The (Block) Macaulay Matrix

Solving Systems of Multivariate Polynomial Equations and Multiparameter Eigenvalue Problems

Christof Vermeersch

Examination committee:
Prof. P. Verbaeten, chair
Prof. B. De Moor, supervisor
Prof. K. Meerbergen
Prof. F. Mohammadi
Prof. B. Mourrain
(Centre Inria d'Université Côte d'Azur)
Prof. B. Plestenjak
(University of Ljubljana)
Dr. O. M. Agudelo

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

December 2023

Abstract

One of the most pervasive tools from (numerical) linear algebra is, without any doubt, the standard eigenvalue decomposition. Eigenvalues describe the intrinsic system dynamics of many natural and scientific phenomena; they explain how a system evolves along the eigenvector direction. This makes eigenvalues and eigenvectors indispensable in a wide array of problems. More importantly, at least in the context of this dissertation, the standard eigenvalue decomposition is also essential when addressing two critical, mathematical problems: finding the common roots of a system of multivariate polynomials and computing the eigenvalues of a multiparameter eigenvalue problem. Multivariate polynomials and multiparameter eigenvalue problems offer natural tools for representing the real world through mathematical models. As a consequence, these two mathematical problems are critical in diverse fields from science and engineering, including systems theory, computational biology and chemistry, robotics, computer vision, and economics.

Solving multivariate polynomial systems remains a very challenging task, in particular when dealing with high total degrees and many variables. There exist various methodologies to compute the solutions, including symbolic methods, which use Gröbner bases or resultants, and efficient numerical iterative methods like homotopy continuation, which track solution paths in continuously deformed systems. In this dissertation, the problem is approached from a (numerical) linear algebra perspective: we leverage the Macaulay matrix, a structured matrix constructed from the coefficients of the polynomials, to formulate a multidimensional realization problem within the right null space of that Macaulay matrix. This multidimensional realization problem yields standard eigenvalue problems, the eigenvalues of which correspond to the common roots of the multivariate polynomials. Although the Macaulay matrix provides an interesting (numerical) linear algebra tool for solving multivariate polynomial systems, the computation of a basis matrix for its right null space is computationally expensive and limits its application potential. This motivates the development of a complementary column space based Macaulay matrix approach, which translates the multidimensional realization problem to the column space (avoiding the right null space). Both subspaces of the Macaulay matrix provide algorithms that find the common roots of the multivariate polynomials.

Multiparameter spectral theory generalizes the classic spectral theory of linear operators to multiple linear operators linked by multiple spectral pa-

rameters. Its origins can be traced back to the problem of using separation of variables to solve boundary-value problems for partial differential equations. One way to solve such problems is via the eigenvalues of a multiparameter eigenvalue problem. Recently, it has been shown that (rectangular) multiparameter eigenvalue problem are also useful in the identification of linear time-invariant dynamical systems and model order reduction. Despite the extensive literature about one-parameter eigenvalue problems, the multiparameter case has not yet fully penetrated the general scientific community. Furthermore, the advent of rectangular multiparameter eigenvalue problems in various applications has uncovered the need for solution approaches. To fill this hiatus, this text extends the Macaulay matrix to the block Macaulay matrix, by replacing the coefficients of the polynomials with the coefficient matrices of the rectangular multiparameter eigenvalue problem. The translation of the solution approaches in the right null space and column space of the Macaulay matrix to the block Macaulay setting, resulting in some of the first (numerical) linear algebra tools for solving rectangular multiparameter eigenvalue problems, is a major contribution of this dissertation. The introduction of this block Macaulay matrix is the final piece of the puzzle to assemble a uniform block Macaulay matrix approach.

Both mathematical problems–systems of multivariate polynomial equations and multiparameter eigenvalue problems–are at the heart of numerous applications. This text illustrates the critical nature of these two problems via several motivational examples from systems theory. These examples originate from the quest to find globally optimal models for single-input/single-output, linear time-invariant dynamical systems, which involves tackling multivariate polynomial optimization problems that minimize a certain polynomial cost function subject to polynomial model constraints. Tackling these motivational examples immediately demonstrates the computational challenges that come with the (block) Macaulay matrix algorithms. Therefore, this dissertation also covers the development of new techniques to exploit the structure and sparsity of the involved (block) Macaulay matrices, which enable more efficient, yet still reliable, solution methods. These techniques are combined into the novel MacaulayLab toolbox and are used to solve the motivational examples in a globally optimal way. Despite having its origin in a curiosity to understand the dynamical behavior of systems, this research integrates insights from various scientific domains and provides novel solution tools for very diverse applications, while relying on tool from numerical linear algebra.

Nederlandse Samenvatting

De standaard eigenwaardeontbinding is, ongetwijfeld, één van de krachtigste gereedschappen uit (numerieke) lineaire algebra. Eigenwaarden beschrijven de intrinsieke systeemdynamica van veel natuurlijke en wetenschappelijke fenomenen; ze verklaren hoe een systeem zich ontwikkelt langs de richting van de eigenvector. Dit maakt eigenwaarden en eigenvectoren onmisbaar in een breed scala van problemen. Belangrijker nog, tenminste binnen de context van dit proefschrift, is dat de standaard eigenwaardeontbinding ook essentieel is bij het oplossen van twee fundamentele, wiskundige problemen: het vinden van de gemeenschappelijke wortels van een stelsel van multivariate veeltermen en het berekenen van de eigenwaarden van een multiparameter eigenwaardeprobleem. Multivariate veeltermen en multiparameter eigenwaardeproblemen bieden een natuurlijke manier aan om de echte wereld via wiskunde modellen te beschrijven. Als een gevolg hiervan zijn deze twee wiskundige problemen cruciaal in diverse vakgebieden, waaronder systeemtheorie, computationele biologie en chemie, robotica, computervisie en economie.

Het oplossen van stelsels van multivariate veeltermvergelijkingen blijft een zeer uitdagende taak, zeker bij hoge graden en veel veranderlijken. Er bestaan verschillende manieren om de oplossingen te vinden, bijvoorbeeld symbolische methoden, die gebruik maken van Gröbnerbasissen of resultanten, en efficiënte numerieke iteratieve methoden zoals homotopievoortzetting, die oplossingspaden volgen in stelsels die continu worden vervormd. In dit proefschrift benaderen we het probleem vanuit een (numerieke) lineaire algebra hoek: we maken gebruik van de Macaulaymatrix, een gestructureerde matrix opgebouwd uit de coëfficiënten van de veeltermen, om een multidimensionaal realisatieprobleem te formuleren binnen de rechter nulruimte van deze Macaulaymatrix. Dit multidimensionel realisatieprobleem resulteert in standaard eigenwaardeproblemen, de eigenwaarden hiervan komen overeen met de gemeenschappelijke wortels van de multivariate veeltermen. Hoewel de Macaulaymatrix een interessante (numerieke) lineaire algebra techniek aanbiedt om stelsels van multivariate veeltermvergelijkingen op te lossen, beperkt de nodige berekeningstijd voor het bepalen van een basismatrix van de rechter nulruimte de toepassingsmogelijkheden. Dit motiveert onze ontwikkeling van een complementaire kolomruimte-gebaseerde Macaulaymatrixmethode, waarin het multidimensionaal realisatieprobleem wordt vertaald naar de kolomruimte (wat het gebruik van de rechter nulruimte vermijdt). Beide deelruimten van de Macaulaymatrix creëren op deze manier algoritmes om de gemeenschappelijke wortels van

multivariate veeltermen te vinden.

Multiparameter spectrale theorie veralgemeent de klassieke spectrale theorie van enkele lineaire operatoren naar meerdere lineaire operatoren verbonden door middel van meerdere spectrale parameters. De oorsprong ervan kan worden teruggevoerd naar het gebruik van scheiding van veranderlijken om randwaardeproblemen op te lossen voor partiële differentiaalvergelijkingen. Dergelijke problemen kunnen worden opgelost door middel van het bepalen van de eigenwaarden van multiparameter eigenwaardeproblemen. Recent is aangetoond dat (rechthoekige) multiparameter eigenwaardeproblemen ook hun net hebben tijdens de identificatie van lineaire tijdsinvariante dynamische systemen en modelreductie. Ondanks de uitgebreide literatuur beschikbaar over één-parameter eigenwaardeproblemen, blijft het multiparameter geval onbekend voor het merendeel van de wetenschappelijke gemeenschap. Bovendien heeft de opkomst van rechthoekige multiparameter eigenwaardeproblemen in verschillende toepassingen de behoefte aan oplossingsmethoden aan het licht gebracht. Om deze lacune op te vullen, breidt dit proefschrift de Macaulaymatrix uit tot de blok-Macaulaymatrix, door de coëfficiënten van de veeltermen te vervangen door de coëfficiëntmatrices van het rechthoekige multiparameter eigenwaardeprobleem. De vertaalstap van de oplossingsalgoritmes in de rechter nulruimte en kolomruimte van de Macaulaymatrix naar de blok-Macaulaymatrix, wat resulteert in enkele van de eerste (numerieke) lineaire algebramethoden voor het oplossen van rechthoekige multiparameter eigenwaardeproblemen, is een belangrijke bijdrage in dit proefschrift. De introductie van deze matrix is het laatste puzzelstuk om een uniform blok-Macaulaymatrixkader te vormen.

Beide wiskundige problemen–stelsels van multivariate veeltermvergelijkingen en (rechthoekige) multiparameter eigenwaardeproblemen–staan centraal in talloze toepassingen. Deze tekst illustreert de fundamentele aard van deze twee problemen aan de hand van verschillende motiverende voorbeelden uit systeemtheorie. Deze voorbeelden komen voort uit de zoektocht naar globaal optimale modellen voor lineaire tijdsinvariante modellen met één ingang en één uitgang, waarbij multivariate veeltermoptimalisatieproblemen worden opgelost door een bepaalde kostfunctie (een veelterm) onderworpen aan modelrestricties (ook veeltermen) te minimaliseren. De confrontatie met deze motiverende voorbeelden legt onmiddellijk de computationele uitdagingen bloot die gepaard gaan met de (blok-)Macaulaymatrixmethoden. Daarom behandelt deze tekst ook de ontwikkeling van nieuwe technieken om de structuur en ijlheid van de betrokken (blok-)Macaulaymatrices uit te buiten, wat efficiëntere, maar toch betrouwbare, oplossingsmethoden mogelijk maakt. Deze technieken zijn gecombineerd in het MacaulayLab-softwarepakket en worden gebruikt om de motiverende voorbeelden op een globaal optimale manier op te lossen. Ondanks dat dit onderzoek zijn oorsprong heeft in een nieuwsgierigheid om het dynamische gedrag van systemen beter te begrijpen, integreert het inzichten uit verschillende wetenschappelijke domeinen en biedt het nieuwe oplossingsmethoden aan voor zeer uiteenlopende toepassingen, gebruikmakend van gereedschappen uit numerieke lineaire algebra.

Contents

Abstract iii

Nederlandse Samenvatting v

List of Figures xv

List of Tables xviii

List of Symbols xix

List of Abbreviations xxi

Introduction 1

1 Introduction and Outline 3
- 1.1 Motivational prologue . . . 6
- 1.2 Four block Macaulay matrices . . . 12
- 1.3 Research objectives . . . 17
- 1.4 MacaulayLab: About the implementation . . . 22
- 1.5 Outline of the dissertation . . . 23

I Fundamentals 27

2 Root-Finding via the Macaulay Matrix 29
- 2.1 Introduction . . . 32
- 2.2 Multivariate polynomials . . . 35
  - 2.2.1 Vector space of polynomials . . . 38
  - 2.2.2 Affine varieties and ideals . . . 41
  - 2.2.3 Number of solutions . . . 44
- 2.3 Macaulay matrix and its subspaces . . . 48
- 2.4 Null space based approach . . . 52
  - 2.4.1 Multidimensional realization theory . . . 52
  - 2.4.2 Solutions with multiplicity greater than one . . . 57
  - 2.4.3 About the notion of a large enough degree . . . 60

2.4.4 Null space based root-finding algorithm 67
2.5 Column space based approach 69
2.5.1 Complementarity with the null space 69
2.5.2 Equivalent column space realization theory 71
2.5.3 Complementary column compression 77
2.5.4 Column space based root-finding algorithm 77
2.6 Extensions to the root-finding approaches 79
2.6.1 Positive-dimensional solution sets at infinity 79
2.6.2 Special shift polynomials 81
2.6.3 Another polynomial basis or monomial ordering 84
2.7 Conclusion 86

3 Eigenvalue-Finding via the Block Macaulay Matrix 95
3.1 Introduction 98
3.2 Multiparameter eigenvalue problems 101
3.2.1 Different matrix pencils and linearizations 103
3.2.2 Relations with square problem formulation 105
3.2.3 Number of solutions 111
3.3 Block Macaulay matrix and its subspaces 114
3.4 Null space based approach 118
3.4.1 Multidimensional realization theory 118
3.4.2 Solutions with multiplicity greater than one 122
3.4.3 About the notion of a large enough degree 123
3.4.4 Null space based eigenvalue-finding algorithm 128
3.5 Column space based approach 130
3.5.1 Equivalent column space realization theory 130
3.5.2 Column space based eigenvalue-finding algorithm 134
3.6 Conclusion 135

4 Fundamental Subspaces of the Macaulay Matrix 143
4.1 Introduction 146
4.2 Projective space 148
4.2.1 Homogeneous polynomials and projective varieties . . . 149
4.2.2 Homogeneous Macaulay matrix 151
4.3 Row space interpretation 152
4.4 Left null space interpretation 155
4.5 Right null space interpretation 159
4.5.1 Affine solutions and multiplication maps 160
4.5.2 Projective solutions and the dual vector space 161
4.6 Column space interpretation 165
4.7 Conclusion 166

II Algorithms 169

5 Recursive Algorithms for the (Block) Macaulay Matrix 171
5.1 Introduction 174
5.2 Recursive null space computations 176

5.2.1 Recursive algorithm . . . 179
5.2.2 Computational complexity . . . 181
5.2.3 Some numerical examples . . . 185
5.3 Sparse null space computations . . . 189
5.3.1 Macaulay matrix algorithm . . . 189
5.3.2 Block Macaulay matrix algorithm . . . 190
5.4 Recursive rank checks . . . 192
5.4.1 Recursive algorithm . . . 195
5.4.2 Computational complexity . . . 197
5.4.3 Some numerical examples . . . 199
5.5 Double recursive algorithms . . . 204
5.5.1 Polynomial system solving . . . 205
5.5.2 Multiparameter eigenvalue solving . . . 209
5.6 Conclusion . . . 212

6 MacaulayLab: About the Implementation 215
6.1 Introduction . . . 218
6.2 Other software packages . . . 219
6.2.1 Systems of multivariate polynomial equations . . . 219
6.2.2 Multiparameter eigenvalue problems . . . 220
6.3 Implementation of MacaulayLab . . . 220
6.3.1 Enlarge the solution subspace . . . 225
6.3.2 Check the rank structure . . . 225
6.3.3 Perform the column compression . . . 226
6.3.4 Shift in the chosen solution subspace . . . 226
6.3.5 Cluster the affine solutions . . . 227
6.3.6 Compute the residual errors . . . 227
6.4 Polynomial basis and monomial ordering . . . 227
6.5 Database with test problems . . . 228
6.6 Comparison of approaches and solvers . . . 228
6.6.1 Different solution approaches in MacaulayLab . . . 230
6.6.2 Other polynomial system solvers . . . 232
6.6.3 Other multiparameter eigenvalue solvers . . . 233
6.7 Conclusion . . . 237

III Applications 243

7 Applications in Systems Theory 245
7.1 Introduction . . . 248
7.2 Examples of possible application areas . . . 249
7.2.1 Common roots of systems of polynomials . . . 249
7.2.2 Eigenvalues and eigenvectors . . . 252
7.3 Complex multivariate optimization . . . 254
7.3.1 Problem formulation . . . 255
7.3.2 Wirtinger derivatives . . . 256
7.3.3 Globally optimal multivariate optimization . . . 260
7.3.4 About ghost solutions . . . 264

7.3.5 Numerical example . . . 265
7.4 Model order reduction . . . 266
7.4.1 Problem formulation . . . 268
7.4.2 Globally optimal model order reduction . . . 269
7.4.3 Numerical example . . . 274
7.5 System identification . . . 275
7.5.1 Problem formulation . . . 278
7.5.2 Globally optimal system identification . . . 279
7.5.3 Numerical example . . . 282
7.6 Conclusion . . . 283

Conclusion **289**

8 Conclusion and Future Work **291**
8.1 Research contributions . . . 294
8.2 Future work . . . 299

Appendices **303**

A Algebraic Geometry and Commutative Algebra **305**
A.1 Group, ring, field, and vector space . . . 308
A.2 Elementary definitions . . . 310
A.3 Solution bounds . . . 312
A.3.1 Univariate polynomials . . . 312
A.3.2 Multivariate polynomial systems . . . 314
A.4 Gröbner bases and Buchberger's algorithm . . . 320
A.4.1 Multivariate division . . . 320
A.4.2 Buchberger's algorithm . . . 322
A.5 Stetter's eigenvalue-eigenvector approach . . . 323
A.5.1 Multiplication maps in the quotient space . . . 323
A.5.2 Normal forms . . . 324

B Numerical Linear Algebra **327**
B.1 Vectors and matrices . . . 330
B.2 Conditioning and stability . . . 331
B.2.1 Conditioning and condition numbers . . . 332
B.2.2 Numerical stability . . . 333
B.2.3 Error measures . . . 335
B.3 Rank, determinant, and inverse of a matrix . . . 336
B.4 Eigenvalues and eigenvectors . . . 338
B.5 Four important matrix decompositions . . . 343
B.5.1 Eigenvalue decomposition . . . 343
B.5.2 Schur decomposition . . . 344
B.5.3 QR decomposition . . . 345
B.5.4 Singular value decomposition . . . 346

C Systems Theory and Shift-Invariant Subspaces **349**
C.1 Systems theory 352
C.1.1 Description of a dynamical system 352
C.1.2 Ho–Kalman's realization algorithm 353
C.2 Shift-invariant subspaces 354
C.3 Multidimensional realization theory 357

D User-Manual of the MacaulayLab Toolbox **359**
D.1 Getting started 362
D.1.1 Installation 362
D.1.2 Help and documentation 362
D.1.3 Quick start 363
D.1.4 Tests to check all functionality 363
D.2 Representation of a problem 364
D.2.1 Systems of multivariate polynomial equations 364
D.2.2 Rectangular multiparameter eigenvalue problems 367
D.2.3 Database with test problems 368
D.3 Solutions via the (block) Macaulay matrix 369
D.3.1 Step-by-step solution approach 369
D.3.2 Direct solution approach 371
D.4 Monomial ordering and polynomial basis 374
D.5 Other useful functions 374

Bibliography **377**

List of Publications **399**

List of Figures

1.1 Unifying (block) Macaulay framework: four seed problems are related to four instances of the (block) Macaulay matrix 10
1.2 Summary of the different steps in solving the motivational example of this dissertation 13
1.3 Overview of the different chapters, with a schematic depiction of the dependencies and related research objectives 24

2.1 Combinatorial explosion of the number of monomials with respect to the maximum total degree and number of variables . . 39
2.2 Real picture of the system of multivariate polynomial equations in Example 2.9 46
2.3 Real picture of the system of multivariate polynomial equations in Example 2.10 47
2.4 Real picture of the system of multivariate polynomial equations in Example 2.11 48
2.5 Spy plot of the Macaulay matrix of degree 5 for the system of multivariate polynomial equations in Example 2.9 50
2.6 Basis matrix of the null space of a Macaulay matrix, for increasing degree, when the solution set is zero-dimensional 63
2.7 Complementarity of the null space and the column space of a Macaulay matrix 71
2.8 Basis matrix of the null space of a Macaulay matrix, for increasing degree, when the solution set is positive-dimensional 82
2.9 Macaulay matrix for a system of random multivariate polynomial system in two different monomial orderings 87

3.1 Real picture of the system of multivariate characteristic polynomial equations in Example 3.10 112
3.2 Spy plot of the block Macaulay matrix of degree 6 for the quadratic two-parameter eigenvalue problem in Example 3.13 116
3.3 Basis matrix of the null space of a block Macaulay matrix, for increasing degree, when the solution set is zero-dimensional . . 125
3.4 Computation time per degree for the first-order autoregressive moving-average (ARMA) model identification problem 136

4.1 Graphical depiction of the fundamental subspaces of an arbitrary matrix and their relations 147

5.1 Annotated visualization of the iterative construction of a block Macaulay matrix 178
5.2 Annotated visualization of the iterative construction of a Macaulay matrix and block banded (Toeplitz) matrix 182
5.3 Experimental complexity analysis of the iteration on the null space computation for the block Macaulay matrix 184
5.4 Experimental complexity analysis of the seed matrix size on the null space computation for the block Macaulay matrix 185
5.5 Experimental complexity analysis of the variables on the null space computation for the block Macaulay matrix 186
5.6 Computation time and relative error for a block Macaulay matrix generated by a linear two-parameter problem 187
5.7 Iteration-wise versus shift-wise algorithm for a block Macaulay matrix generated by a quadratic three-parameter problem . . . 188
5.8 Computation time for a block Toeplitz matrix generated by a linear problem 189
5.9 Annotated last iteration of the Macaulay matrix in Figure 5.2a 191
5.10 Comparison of memory usage of the recursive and sparse algorithm for a Macaulay matrix 191
5.11 Comparison of memory usage of the recursive and sparse algorithm for a block Macaulay matrix 193
5.12 Graphical example of the iterative procedure to determine the rank structure of a block row matrix 194
5.13 Experimental complexity analysis of the iteration on the null space computation for the block row matrix 199
5.14 Experimental complexity analysis of the seed matrix size on the null space computation for the block row matrix 200
5.15 Computation time and relative error for a block row matrix generated by a increasing rank problem 201
5.16 Computation time and relative error for a block row matrix generated by a stabilizing rank problem 203

6.1 Schematic depiction of the construction of the data structure to represent a problem in MacaulayLab 221
6.2 Diagram of the main functions implemented in MacaulayLab and their dependencies 223
6.3 Output of MacaulayLab when solving the `noon5` system from the database via the default Macaulay matrix approach 224
6.4 Output of MacaulayLab when displaying information `noon3` system in the database 230
6.5 Computation time to solve the `noon3` system via the different sets of options 232
6.6 Computation time to solve the `conform` system via the different sets of options 233

6.7 Computation time to solve the `h2f5` multiparameter eigenvalue problem (MEP) via the different sets of options 234
6.8 Computation time to solve the `realization` MEP via the different set of options 235
6.9 Comparison of the relative computation time to solve five different systems of multivariate polynomial equations 236
6.10 Comparison of the relative computation time to solve five different rectangular MEPs 238
6.11 Comparison of the computation time to solve linear MEPs of increasing seed matrix size 240
6.12 Comparison of the computation time to solve linear MEPs with increasing number of parameters 241

7.1 Possible configurations of the elbow of a planar robot arm with a fixed shoulder in the origin and two arm segments 250
7.2 Simple instance of a camera pose estimation problem with a camera centered at an unknown position 252
7.3 Visualization of a mass-spring system with two masses, three springs, and two dampers. 253
7.4 Contour lines of the real-valued polynomial cost function in Example 7.1 259
7.5 Contour lines of the real-valued polynomial cost function $f(z, \bar{z})$ in Example 7.3 263
7.6 Spy plot of the Macaulay matrix that generates the coefficient matrices of the MEP in Example 7.5 267
7.7 Contour plot of the cost function of the globally optimal first-order model order reduction problem in Example 7.8 276
7.8 Magnitude plot of the Bode diagrams of the original transfer function and the computed first-order approximants 277
7.9 Contour plot of the cost function of the first-order ARMA model in Example 7.10 284
7.10 Scaling properties of the globally optimal identification approach for a growing number of data points 286

A.1 Visualization of the Newton polytope of the polynomial in Example A.8 316
A.2 Minkowski sum of two Newton polytopes and several δ-dilations of a Newton polytope 318

D.1 Representation of a system of multivariate polynomial equations or rectangular MEP in MacaulayLab 365

List of Tables

2.1 Solutions for system (2.73) obtained before the clustering step to improve the accuracy. 61
2.2 Computation time and maximum absolute residual error for solving the motivational example 89

3.1 Four different types of (multiparameter) eigenvalue problems, according to the structure of the (multivariate) matrix pencil . 103
3.2 Numerical solutions and absolute residual errors of Example 3.4 obtained via the null space based algorithm 126
3.3 Affine numerical solutions and absolute residual errors of Example 3.5 obtained via the null space based algorithm 128

4.1 Overview of the four fundamental subspaces of a Macaulay matrix of certain degree 149

5.1 Computational complexity to compute a numerical basis matrix of the null space of a block Macaulay matrix 183
5.2 Computational complexity to compute a numerical basis matrix of the null space of a block row matrix 198
5.3 Linearly independent rows of the basis of the null space of the Macaulay matrix that comprises the system in Example 5.14 . 205
5.4 Obtained results of solving the system of multivariate polynomials in Example 5.15 via different recursive combinations . . . 206
5.5 Obtained results of solving the system of multivariate polynomials in Example 5.16 via different recursive combinations . . . 207
5.6 Obtained results of solving the system of multivariate polynomials in Example 5.17 via different recursive combinations . . . 207
5.7 Obtained results of solving the system of multivariate polynomials in Example 5.14 via different recursive combinations . . . 208
5.8 Obtained results of solving the system of multivariate polynomials in Example 5.19 via different recursive combinations . . . 209
5.9 Obtained results of solving the linear two-parameter problem in Example 5.20 via different recursive combinations 210
5.10 Obtained results of solving the quadratic three-parameter problem in Example 5.21 via different recursive combinations 210

5.11 Obtained results of solving the quadratic three-parameter problem in Example 5.22 via different recursive combinations 211
5.12 Obtained results of solving the quadratic two-parameter problem in Example 5.23 via different recursive combinations 213
5.13 Obtained results for solving the motivational example via different recursive combinations 214

6.1 Small selection of systems in the database, with some of their key properties 229
6.2 Small selection of MEPs in the database, with some of their key properties 229
6.3 Summary of the five different sets of options used to compare the capabilities of MacaulayLab 231
6.4 Comparison of the computation time to solve five different systems of multivariate polynomial equations 234
6.5 Comparison of MacaulayLab with PHClab and Maple 237
6.6 Comparison of the computation time to solve five different rectangular MEPs 239

7.1 Numerical values of the candidate solutions of the optimization problem in Example 7.3 265
7.2 Stabilization diagram of the block Macaulay matrix in Example 7.8, for increasing degrees 275
7.3 Identified real model parameters of the globally optimal first-order model order reduction problem in Example 7.8 275
7.4 Identified real model parameters of the first-order ARMA model in Example 7.10 283
7.5 Obtained results of solving the identification problem in Example 7.10 via different recursive combinations 283

A.1 Numerical solutions of system (A.44) obtained via Stetter's eigenvalue-eigenvector approach 325

D.1 Overview of the different options that can be used by the MacaulayLab solver 373

List of Symbols

Basic notation.

a	scalar
$\boldsymbol{a}$	vector
$\boldsymbol{A}$	matrix
$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$	matrix pencil

Algebraic structures.

$\emptyset$	empty set
$\mathbb{N}$	set of nonnegative integers
$\mathbb{N}_0$	set of positive integers
$\mathbb{Z}$	ring of integer numbers
$\mathbb{R}$	field of real numbers
$\mathbb{C}$	field of complex numbers
$\mathbb{F}_q$	finite field with q elements
K	arbitrary field
V	vector space
V'	dual of the vector space
V°	annihilator of the vector space

Rings, varieties, and ideals.

$\boldsymbol{x}^{\boldsymbol{\alpha}}$	monomial
$p(\boldsymbol{x})$	polynomial
$\mathcal{C}^n$	set of all monomials in n variables
$\mathcal{C}^n_d$	$\mathcal{C}^n$ with degree up to d
$\mathcal{P}^n$	set of all complex polynomials in n variables
$\mathcal{P}^n_d$	$\mathcal{P}^n$ with degree up to d
$\widetilde{\mathcal{P}}^n$	set of all homogeneous complex polynomials in n variables
$\widetilde{\mathcal{P}}^n_d$	$\widetilde{\mathcal{P}}^n$ with degree up to d
$\mathbb{C}^n$	affine complex space
$\mathbb{P}^n$	projective space
$\mathcal{V}(P)$	variety defined by P
$\mathcal{I} = \langle P \rangle$	ideal generated by P
$[p]_{\mathcal{I}}$	residue class of $p(\boldsymbol{x})$ modulo the ideal $\mathcal{I}$
$\mathcal{R}[\mathcal{I}] = \mathcal{P}^n/\mathcal{I}$	quotient space

Operations.

$\overline{\cdot}$	complex conjugate
$.^{-1}$	inverse
$.^{\dagger}$	pseudo-inverse
$.^{\mathrm{T}}$	transpose
$.^{\mathrm{H}}$	Hermitian transpose
$\mathrm{vec}(\cdot)$	vectorization
$\mathrm{rank}(\cdot)$	rank
$\det(\cdot)$	determinant
$\mathrm{supp}(\cdot)$	support of a (matrix) polynomial
$\deg(\cdot)$	maximum total degree of a (matrix) polynomial
$\otimes$	Kronecker product
$\oplus$	Kronecker sum
$\lvert\cdot\rvert_{\otimes}$	Kronecker determinant
$\lVert\cdot\rVert_1$	Manhattan norm (1-norm)
$\lVert\cdot\rVert_2$	Euclidean norm (2-norm)
$\lVert\cdot\rVert_{\mathrm{F}}$	Frobenius norm

Error measures.

$\widetilde{e}$	absolute reconstruction error
$\widetilde{e}^{(r)}$	relative reconstruction error
$\lVert\boldsymbol{e}\rVert_2$	absolute residual error
$\lVert\boldsymbol{e}\rVert_2^{(r)}$	relative residual error

Other symbols.

i	imaginary unit
$>$	monomial ordering
d_*	degree of regularity (except when denoted otherwise)
$d_\circ$	solution degree
$i_\circ$	desired iteration
$\boldsymbol{I}_s$	identity matrix of size $s \times s$
$(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$	state space representation
$H(s)$	transfer function
$h(t)$	impulse response in continuous time
h_k	impulse response in discrete time

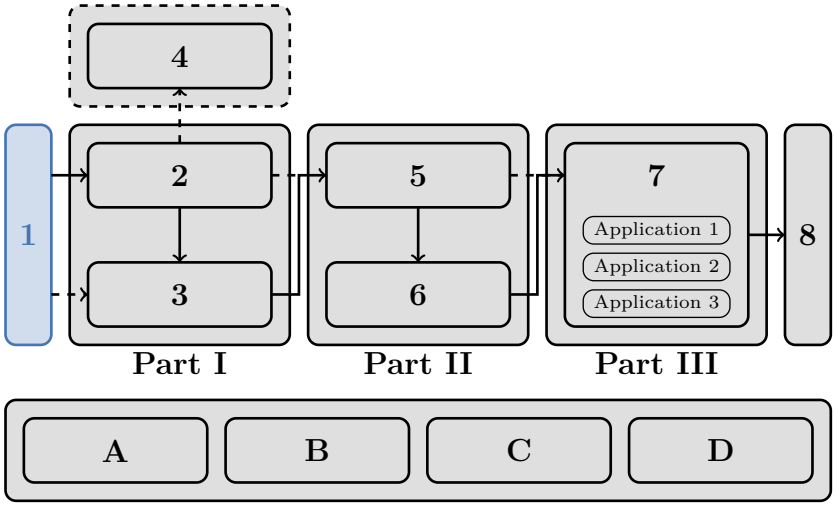
List of Abbreviations

AR	autoregressive
ARIMA	autoregressive integrated moving-average
ARMA	autoregressive moving-average
ARMAX	autoregressive moving-average with exogeneous input
BKK	Bernstein–Khovanskii–Kushnirenko
FLOP	floating-point operation
FSR	forward shift recursion
GEP	generalized eigenvalue problem
GREVLEX	graded reverse lexicographic
GRINVLEX	graded inverse lexicographic
GRLEX	graded lexicographic
HKP	Hochstenbach–Košir–Plestenjak
LEX	lexicographic
LTI	linear time-invariant
MA	moving-average
MEP	multiparameter eigenvalue problem
MIMO	multiple-input/multiple-output
OE	output-error
PEP	polynomial eigenvalue problem
RAM	random-access memory
RC	research contribution
RO	research objective
SEP	standard eigenvalue problem
SISO	single-input/single-output
SVD	singular value decomposition

Introduction

Introduction and Outline

This dissertation dives into the fascinating worlds of multivariate polynomials and multiparameter eigenvalues problems. We approach these two, seemingly unrelated, problems from a (numerical) linear algebra point of view and tackle them via the novel, unifying block Macaulay matrix approach, which provides us with new (numerical) linear algebra algorithms. The text uses applications from systems theory as motivational examples, but it also has the bold ambition to introduce, from a (numerical) linear algebra point of view, some fundamental concepts from the abstract and technical literature of algebraic geometry and multiparameter spectral theory. The contributions of this research enable a new perspective on applications from various fields of science and engineering; a perspective with the ambition to tackle problems with exactness or global optimality in mind. We believe that this dissertation will lead to many new, interesting research challenges that can (and should) be tackled by mathematicians and engineers in the (near) future.



Outline. We start with a motivation in Section 1.1, building up to the central research question of this dissertation. Afterwards, Section 1.2 presents the novel, unifying block Macaulay matrix approach and shows how the Toeplitz matrix, Macaulay matrix, block Toeplitz matrix, and block Macaulay matrix are related. Next, in Section 1.3, we phrase the different research objectives of this dissertation and, in Section 1.4, we advocate the software toolbox that we have developed alongside the text. Finally, Section 1.5 outlines the different chapters of the dissertation.

1.1 Motivational prologue

This dissertation considers two mathematical problems that are critical in different applications from science and engineering: solving systems of multivariate polynomial equations and computing the eigenvalues of multiparameter eigenvalue problems (MEPs). The motivation for tackling these two, seemingly unrelated, problems in one text stems from the fact that we encounter them both while dealing with the applications that we tackle in our research. Although this research lives at the interplay of various fields of science and engineering, it all started with a curiosity in understanding the dynamical behavior of systems.

As this motivational prologue intends to clarify, the road that has led to the insights and algorithms in this dissertation was paved by the ambition to truly understand the dynamical behavior of systems. However, the road quickly took a detour and has evolved into the direction of considering the underlying critical, mathematical problems that were encountered during this endeavor. Since then, the goal of this research has been to solve these two mathematical problems via novel (numerical) linear algebra techniques. In that sense, this dissertation can be seen as a contribution to the field of (numerical) linear algebra, motivated by examples that stem from the field of systems theory.

From dynamical behavior to measured data points

The dynamical behavior of systems is the topic of systems theory. One sound, all-embracing definition for systems theory does not really exist, but Antoulas [9] has done a nice attempt of summarizing its importance in science and engineering:

> *"[Let] us attempt to define what is meant by [systems theory]. It is a science which deals with phenomena whose complexity cannot be described by simple laws. It is concerned not with the actual world but with models of the actual world. System[s] theory is not only descriptive like the natural sciences, but prescriptive as well. This means that system[s] theory does not only tell us how systems are (analysis), but how systems should be (synthesis)."*

Systems theory is, thus, a science that examines and analyzes behavior, interactions, and properties of complex systems, which are combinations of components that act together and perform a certain objective [192]. A system does not need to be physical, for example an industrial or biological system; the concept of the system can also be applied to abstract, dynamical phenomena such as those encountered in economics or sociology [192, 254]. An important tool in systems theory is a mathematical model of a system. Via these mathematical models, it is possible to describe the dynamical behavior of a system as a function of, for example, time or space. In turn, these mathematical models are used in all scientific disciplines to simulate, analyze, monitor, predict, and control systems [254].

There are two main roads to construct a mathematical model of a dynamical system: a rigorous development from first principles (i.e., white box modeling) or an experimental development from observed input/output data points of the system (i.e., black box modeling or system identification). While physicists (and others) are often more interested in the underlying physical laws when deriving the mathematical models, engineers typically infer model parameters of a chosen model structure from the available data points. Moreover, it is in a practical setting not always possible to determine the exact model or the model could lead to a too complex design. Engineers, therefore, typically resort to system identification techniques to build their models.

System identification is the field of modeling dynamical systems from measured data points, or in the words of Ljung [158]:

> *"Inferring models from observations and studying their properties is really what science is about. [...] System identification deals with the problem of building mathematical models of dynamical systems based on observed data from the system."*

There exist many types of model classes that can be considered in such a system identification attempt, and perhaps even more techniques to obtain the parameters of these models. In general, an identified model tries to capture the relations between input, output, and noise. It depends on a set of model parameters, which are selected to obtain the best fit between the model and the measured data points. We refer the interested reader to [95, 158, 254] for an introduction into that subject.

Motivational example. A particular model, which we use throughout this text as a motivational example, is the autoregressive moving-average (ARMA) model. This model regresses an observed output sequence $y_k \in \mathbb{R}$, for $k = 1, \ldots, N$, on its own lagged values $y_{k-i} \in \mathbb{R}$ $(i = 1, \ldots, n_a)$ and on a linear combination of unobserved, latent input samples $e_{k-j} \in \mathbb{R}$ $(j = 1, \ldots, n_c)$:

$$y_k + \alpha_1 y_{k-1} + \cdots + \alpha_{n_a} y_{k-n_a} = e_k + \gamma_1 e_{k-1} + \cdots + \gamma_{n_c} e_{k-n_c}, \tag{1.1}$$

where α_i and γ_j are the $n_a + n_c$ model parameters [40]. It is an appropriate model to describe a system that has its own behavior (i.e., the autoregressive part) as well as a series of unobserved shocks (i.e., the moving-average part). For example, stock prices may be shocked/influenced by some random external factors, next to the autoregressive financial market indicators [210, 240]. The task of system identification is now in obtaining the model parameters α_i and γ_j for a given series of N measured data points y_k.

From measured data points to seed equations

In our research, we only consider single-input/single-output (SISO), linear time-invariant (LTI) models, more specifically, misfit-versus-latency models, which is a broad model class that combines an unobserved, latent input (cf.,

latency models, like the ARMA model) with a misfit on the input and output (cf., misfit models, like the output-error (OE) model): an altered input/output sequence must satisfy the imposed model equation, but this adaptation can be achieved via a combination of a latent input and a misfit on the input/output data points [155].

One important objective in our research is the aim for global optimality. This optimality is captured in finding the global minimizers of the cost function that describes the fit between the identified model and the measured data points, while the imposed model equations form the constrains of that optimization problem. For the specific class of misfit-versus-latency models, the optimization problems that need to be solved are multivariate polynomial optimization problems, which are optimization problems in which both the cost function and the equality constraints are (multivariate) polynomials. The first-order necessary conditions for optimality (i.e., the Karush–Kuhn–Tucker conditions) correspond to a system of multivariate polynomial equations, the solutions of which are the stationary points of the original cost function under the constraints [190]. So, the system identification task of finding the globally optimal model parameters boils down to the problem of finding the solutions of a system of multivariate polynomial equations. Furthermore, in our research [70, 71, 259], we have shown that in many cases this multivariate polynomial system contains a certain structure (e.g., many of the variables only appear "linearly" in the system) and we have exploited this structure to reformulate the problem as a (rectangular) MEP. These insights have led to the central research question of this dissertation:

Central Research Question

How do we solve systems of multivariate polynomial equations and (rectangular) multiparameter eigenvalue problems?

We call multivariate root-finding and multiparameter eigenvalue-finding the two **critical problems of this dissertation**.

The practical need of people for algorithms that solve these two critical problems is motivated by, for example, the applications from systems theory that we use as an illustration in this text.

> **Motivational example.** If we retake our motivational example, then the system identification task is about finding the unknown model parameters α_i and γ_j of the ARMA model for a given sequence of N output samples y_k, while minimizing the latent inputs e_k (i.e., while making sure that the model fits the measured data points as good as possible). This identification problem is a multivariate polynomial optimization problem, in which we minimize the squared 2-norm of the latent input vector $\boldsymbol{e}$, with the constraint

that the parameters must satisfy the model constraint in (1.1), i.e.,

$$\begin{aligned} &\min \|\boldsymbol{e}\|_2^2 \\ &\text{subject to } \sum_{i=0}^{n_a} \alpha_i y_{k-i} = \sum_{j=0}^{n_c} \gamma_j e_{k-j}, \end{aligned} \tag{1.2}$$

for $k = 0, \ldots, N - n_a$. As we show in Chapter 7, this problem can be rephrased both as a multivariate polynomial root-finding problem (i.e., a system of multivariate polynomial equations) and as a multiparameter eigenvalue-finding problem (i.e., an MEP). The solution approaches that we develop in this dissertation need to find very accurate solutions, in order to retrieve the globally optimal parameters of the ARMA model.

From seed equations to standard eigenvalue problems

In order to solve the two critical, mathematical problems, we have considered one very special object: **the (block) Macaulay matrix**. By considering the univariate/one-parameter and multivariate/multiparameter case of each of these two problems, they break down into four different seed problems: univariate polynomial equations, multivariate polynomial equations, polynomial eigenvalue problems, and multiparameter eigenvalue problems. We call them seed problems because their (seed) equation(s) generate the (block) Macaulay matrix that solve them. Figure 1.1 highlights the each of these four cases of the (block) Macaulay matrix.

The block Macaulay matrix is the block (i.e., multiple-output) generalization of the well-known Macaulay matrix from algebraic geometry. The *Macaulay matrix*, on the one hand, was first introduced in 1902 by Macaulay [159] and has, since then, become an important tool in the study of multivariate polynomials. When dealing with systems of multivariate polynomial equations, a Macaulay matrix constructed from the coefficients of the polynomials of that system is never too far away. In the univariate case, the Macaulay matrix reduces to the (banded[1]) Toeplitz matrix (one univariate polynomial) or Sylvester matrix (two univariate polynomials). Its fundamental subspaces (in a linear algebra sense) are intrinsically linked with the properties of the generating polynomials. On the other hand, the *block Macaulay matrix* has been developed within the scope of this dissertation to tackle the rectangular MEPs that emerged from the problems from systems theory that we were trying to solve. Instead of (scalar) coefficients, a block Macaulay matrix is constructed from the coefficient matrices of a rectangular MEP. In that sense, it extends the (scalar) Macaulay matrix from the multivariate polynomial setting to the MEP setting. When the eigenvalue problem only contains one spectral parameter,

[1] In the literature, this type of Toeplitz matrix is often called a banded Toeplitz matrix, in order to make the distinction with full and circulant Toeplitz matrices. Since we only consider the banded Toeplitz matrix, which is the univariate simplification of the Macaulay matrix, we leave out the term "banded" in the remainder of this chapter. This is the same for the block (banded) Toeplitz matrix.

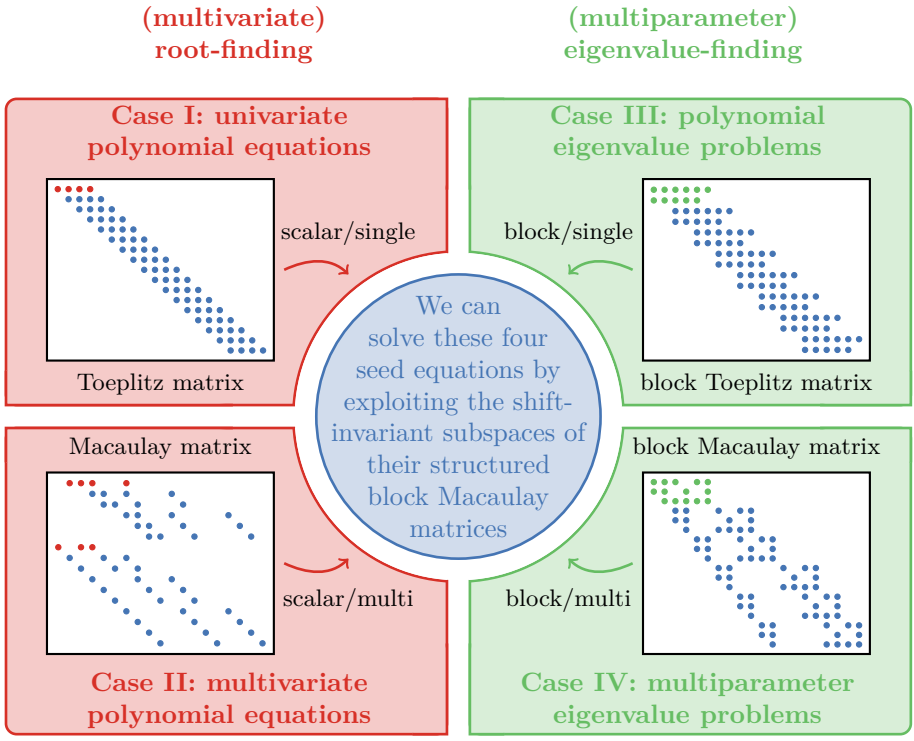


Figure 1.1. Unifying (block) Macaulay framework: the four different appearances of the (block) Macaulay matrix allow us to solve four different cases of seed problems by exploiting the scalar/block single/multi-shift-invariant subspaces of the structured (block) Macaulay matrices that they generate.

the block Macaulay matrix is a block Toeplitz matrix. In a certain sense, the Toeplitz matrix, Macaulay matrix, block Toeplitz matrix, and block Macaulay matrix are four appearances of the same matrix and they constitute, together, the so-called **block Macaulay matrix framework**, which allows us to investigate four different cases of seed problems via a novel, unifying approach (Section 1.2).

In this text, we take a look at the fundamental subspaces of the (block) Macaulay matrix to answer some fundamental questions about the seed equations. In particular, the right null space and column space of the (block) Macaulay matrix are very interesting, since their structure provides us with standard eigenvalue problems that yield the solutions of the original seed equation(s). In that sense, we continue along the approach of Stetter [229]:

> *"[...] matrix eigenproblems are not just some tool in the solution of polynomial systems of equations [but] they represent the weakly nonlinear nucleus to which the original, strongly nonlinear task may be reduced."*

However, we deal not only with systems of multivariate polynomial equations: the (block) Macaulay matrix allows us to also tackle univariate polynomials, polynomial eigenvalue problems, and rectangular MEPs.

This text dives into the fascinating worlds of (multivariate) polynomials and (multiparameter) eigenvalues problems, via the novel, (unifying) block Macaulay matrix approach. Contrary to the above-mentioned approach of Stetter (which still contains a symbolic step), we consider these problems via a pure numerical linear algebra point of view, with inspirations from realization theory, similar as in Dreesen et al. [80]:

> *"Notice that this [approach] is a system-theoretical interpretation of Stetter's eigenvector method, which has been discovered independently by several researchers in the 1980s and 1990s. [While these researchers] employ a Groebner basis approach to find the system matrices, the proposed method in this article uses a linear algebra formulation and does not require the computation of a Groebner basis, and is therefore more reminiscent of matrix-based methods like the ones of Jónsson and Vavasis [127] and Mourrain [182]."*

We only use (numerical) linear algebra techniques, leveraging the many decades of advancements in that field with respect to computational efficiency and numerical robustness. Our philosophy is identical to that of Batselier [21] and Dreesen [78]; we extend their pure (numerical) linear algebra methodology in this dissertation to other seed problems/equations. The algorithms presented in this text do not always (or necessarily) outperform existing, dedicated methods, e.g., the homotopy continuation methods for solving systems of multivariate polynomial equations are typically much more efficient than a Macaulay matrix approach. However, we present a unifying approach that sticks as close as possible to familiar language of and the well-understood techniques from (numerical) linear algebra. Furthermore, the seed equation(s) may be obtained from a noisy experimental setting, which requires taking into account the limited accuracy of the experiment. This (numerical) linear algebra approach allows for a careful consideration of the numerical aspects while using finite-precision arithmetics, contrary to, for example, symbolic methods.

This text uses applications from systems theory as motivational examples and has the bold ambition to be a starting point to get introduced, from a (numerical) linear algebra point of view, in the abstract and technical literature of algebraic geometry and multiparameter spectral theory[2]. We think that this block Macaulay matrix framework is also very powerful from a didactical point of view: mathematicians and engineers with a working knowledge of linear algebra will easily understand the concepts disseminated in this text and will be

[2]When writing this text, we have noticed the challenge of citing the original articles correctly. Sometimes Stigler's law [233] was just lurking behind the corner, sometimes the available information was rather scarce or ambiguous. Many of the references cited in scientific papers have not been read by the authors that cited them, leading to a propagation of errors through the literature [16]. Thanks to digitization of (old) sources, the task of consulting scientific papers and books has become easier than ever. However, some sources that have been cited frequently in the literature were not possible to retrieve, and we have added the "only citation" note to those references in the bibliography.

able to implement our methods without (a lot of) additional expert knowledge. We believe that this dissertation[3] will open many interesting research challenges that can (and should) be tackled in the (near) future!

Motivational example. In order to solve the system of multivariate polynomial equations or the rectangular MEP that we encounter when identifying the model parameters of the ARMA model, we use one of the (block) Macaulay matrix approaches explained in this dissertation:

- The system of multivariate polynomial equations can be tackled via its associated Macaulay matrix.
- The block Macaulay matrix generated from the (rectangular) MEP can be used to find the eigenvalues.

Multidimensional realization problems in the right null space or column space of the (block) Macaulay matrix yield the solutions to the seed equation(s), one of which corresponds to the global minimizer of the underlying multivariate polynomial optimization problem (1.2). For a given sequence of output data points, this (numerical) linear algebra approach results in the least-squares globally optimal ARMA model. A summary of the different steps needed to tackle this motivational example can be found in Figure 1.2.

1.2 Four block Macaulay matrices

What do "rooting (univariate/systems of multivariate) polynomials" and "solving (one-parameter/multiparameter) eigenvalue problems" have in common? They are, in essence, four cases of related seed problems with increasing complexity for which the solution set can be obtained via a shift-invariant subspace of a structured (block) Macaulay matrix generated from that problem. By considering all four cases of seed problems in relation to the shift-invariant subspaces of the structured matrices that they generate, we can provide a novel, unifying block Macaulay matrix approach[4], which constitutes the core of this dissertation. Recall that Figure 1.1 shows how these four cases are connected through the (block) Macaulay matrix.

[3] The contributions of this dissertation fit in the scope of the related ERC project (i.e., "Back to the roots of data-driven dynamical system identification") of the candidate's supervisor, which aims at connecting, understanding, and using the structured (block) Macaulay matrix to solve various problems in system identification. This ERC project lives at the interface of algebraic geometry, operator theory, systems theory, and numerical linear algebra. It combines notions from all four mathematical disciplines to create a coherent framework, via the (block) Macaulay matrix, that can deal with inexact data. This dissertation, which was also supported in part by an FWO Strategic Basic Research fellowship, has focussed on the (numerical) linear algebra part of that interplay, with motivational examples from systems theory in mind.

[4] An exposition of the intrinsic connection between these four cases, with numerical examples, can be found in [69]. The relations between these four cases of problems (and possible application areas) was also the topic of the candidate's plenary talk at the 2022 ERNSI Workshop in System Identification.

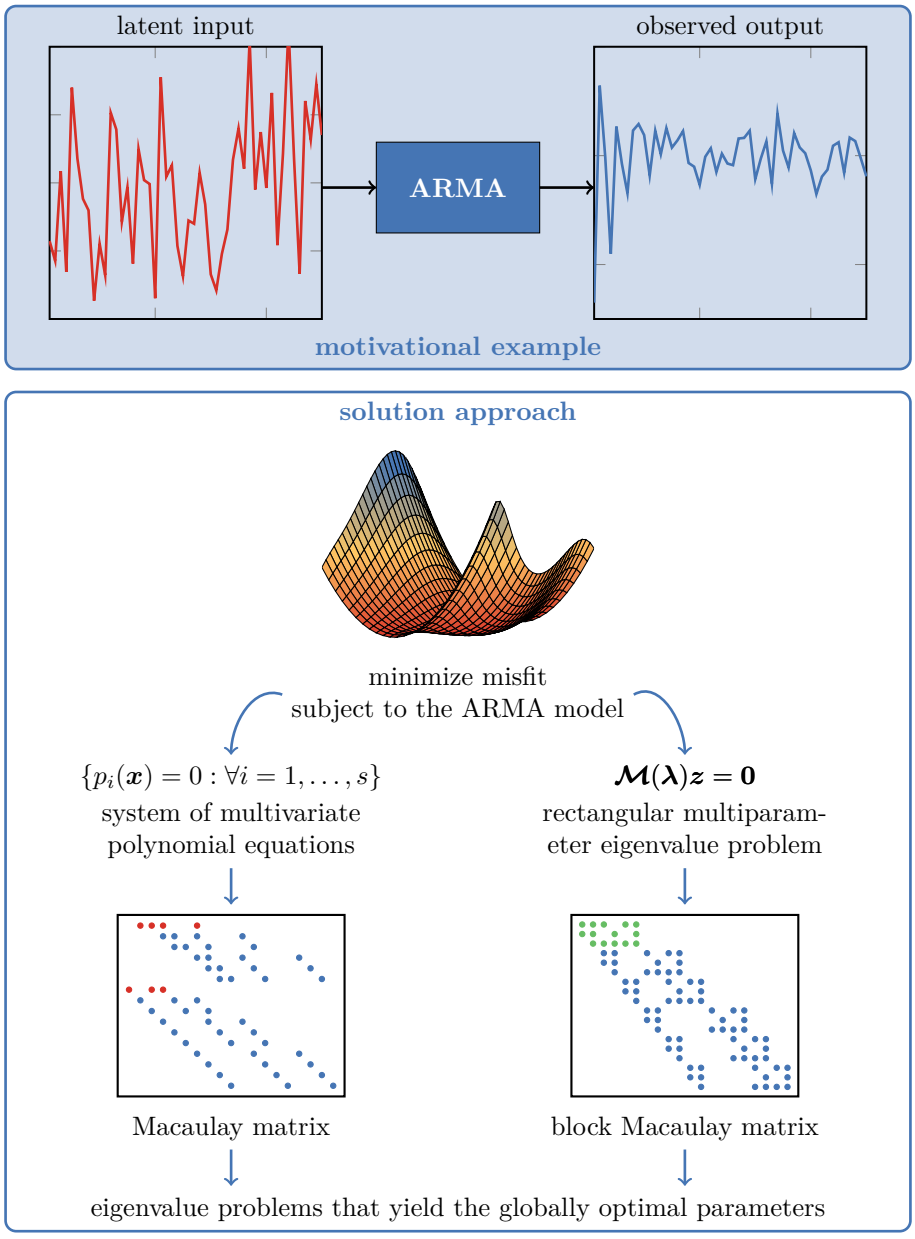


Figure 1.2. Summary of the different steps in (globally optimal) solving the motivational example of this dissertation: The problem of finding the optimal parameters of the ARMA model corresponds to a multivariate polynomial optimization problem, which can be rephrased as a system of multivariate polynomial equations or a rectangular MEP. The (block) Macaulay matrix approaches in this dissertation translate these seed problems into eigenvalue problems that yield the globally optimal parameters of the motivational problem.

Each new case adds an additional layer of complexity, but the same steps to go from the seed equation(s) to the solutions can be taken for each case [69]: Firstly, we generate additional equations by multiplying the given seed equation by monomials of increasing degree. We call this process forward shift recursion (FSR), because it generates a (block) Macaulay matrix from this seed equation(s) by shifting the coefficients or coefficient matrices in a structured fashion. Next, the right null space of this structured matrix is computed, which exhibits for each case of seed equation(s) a specific type of shift-invariance. By exploiting this shift-invariant structure, we finally obtain the solutions via (one of multiple) multidimensional realization problem(s). This provides a very didactical and unifying approach to tackling these four cases of seed problems. In essence, it transforms more difficult problems into standard eigenvalue problems, via multidimensional realization theory, using (numerical) linear algebra techniques only.

Case I: univariate polynomial equation(s). The first seed equation is the **univariate polynomial equation**, e.g.,

$$p(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 = 0. \tag{1.3}$$

We want to find the roots $x|_{(j)} \in \mathbb{C}$, $j = 1, \ldots, 3$, of the polynomial $p(x)$. Starting from the seed equation (1.3), it is possible to generate new equations by multiplying $p(x)$ by monomials x^α (α is the power of x), i.e.,

$$\begin{aligned} \big(c_0 + c_1 x + c_2 x^2 + c_3 x^3\big) &= 0, \\ x\big(c_0 + c_1 x + c_2 x^2 + c_3 x^3\big) &= 0, \\ x^2\big(c_0 + c_1 x + c_2 x^2 + c_3 x^3\big) &= 0, \\ &\vdots \end{aligned} \tag{1.4}$$

This process is called **scalar forward single-shift recursion** and, when we gather these equations in a structured matrix, we obtain a **Toeplitz matrix**. For the seed equation in (1.3), the Toeplitz matrix has the following structure:

$$\underbrace{\begin{array}{c} \\ 1 \\ x \\ x^2 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} 1 & x & x^2 & x^3 & x^4 & x^5 \end{array} \\ \begin{bmatrix} c_0 & c_1 & c_2 & c_3 & 0 & 0 \\ 0 & c_0 & c_1 & c_2 & c_3 & 0 \\ 0 & 0 & c_0 & c_1 & c_2 & c_3 \end{bmatrix} \end{array}}_{\text{Toeplitz matrix } \boldsymbol{T}} \underbrace{\begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \end{bmatrix}}_{\boldsymbol{v}} = \mathbf{0}. \tag{1.5}$$

The vector $\boldsymbol{v}$ is a vector in the right null space of this Toeplitz matrix. In the special structure of $\boldsymbol{v}$ lies the key to solving the univariate polynomial equation. As we explain in Appendix C, the null space of a Toeplitz matrix is **scalar single-shift-invariant**, which means that shifting (i.e., multiplying)

some elements of $\boldsymbol{v}$ by the variable x results in elements of that vector:

$$\begin{bmatrix} 1 \\ x \\ x^2 \end{bmatrix} \xrightarrow{x} \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}. \tag{1.6}$$

Suppose that we know the three (affine and simple) solutions of (1.3), the vector $\boldsymbol{v}$ evaluated in the different solutions spans a basis for the right null space, which means that

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}|_{(1)} & \boldsymbol{v}|_{(2)} & \boldsymbol{v}|_{(3)} \end{bmatrix} \tag{1.7}$$

annihilates $\boldsymbol{T}$. Shifting the first five rows of $\boldsymbol{V}$, indicated by $\underline{\boldsymbol{V}}$, by the variable x results in the last five rows of $\boldsymbol{V}$, indicated by $\overline{\boldsymbol{V}}$, or

$$\underline{\boldsymbol{V}} \begin{bmatrix} x|_{(1)} & 0 & 0 \\ 0 & x|_{(2)} & 0 \\ 0 & 0 & x|_{(3)} \end{bmatrix} = \overline{\boldsymbol{V}}. \tag{1.8}$$

The eigenvalues of the matrix pencil $\left(\overline{\boldsymbol{V}}, \underline{\boldsymbol{V}}\right)$ are the solutions of (1.3). Of course, we do not know $\boldsymbol{V}$ in advance, since it is constructed from the unknown solutions $x|_{(j)}$. In Chapter 2, we show how to overcome this issue, among many more important details (e.g., how the approach changes when the problem has multiple solutions or solutions at infinity).

Case II: multivariate polynomial equations. When we consider a **system of multivariate polynomial equations**, the polynomials consist of two or more variables. For example,

$$\begin{cases} p_1(\boldsymbol{x}) = c_{00} + c_{10}x_1 + c_{01}x_2 = 0, \\ p_2(\boldsymbol{x}) = d_{00} + d_{20}x_1^2 + d_{02}x_2^2 = 0, \end{cases} \tag{1.9}$$

is a system of two bivariate polynomial equations. Similar to Case I, we generate new (multivariate) polynomial equations by multiplying $p_1(\boldsymbol{x})$ and $p_2(\boldsymbol{x})$ with monomials of increasing total degree (Section 2.2.1), i.e.,

$$1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, \ldots, \tag{1.10}$$

and obtain a structured matrix, also known as the **Macaulay matrix**. For the seed equations in (1.9), the Macaulay matrix has the following structure:

$$\underbrace{\begin{array}{c} \\ 1 \\ x_1 \\ x_2 \\ 1 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{array} \\ \begin{bmatrix} c_{00} & c_{10} & c_{01} & 0 & 0 & 0 \\ 0 & c_{00} & c_{10} & c_{01} & 0 & 0 \\ 0 & 0 & c_{00} & 0 & c_{10} & c_{01} \\ d_{00} & 0 & 0 & d_{20} & 0 & d_{02} \end{bmatrix} \end{array}}_{\text{Macaulay matrix } \boldsymbol{M}} \underbrace{\begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}}_{\boldsymbol{v}} = \boldsymbol{0}. \tag{1.11}$$

We call this process **scalar forward multi-shift recursion**, because shifts in multiple variables of the coefficients generate the pattern of the Macaulay matrix. The structure of the null space now can be shifted by two different variables instead of only one:

$$\begin{bmatrix} x_1 \\ x_1^2 \\ x_1x_2 \end{bmatrix} \xleftarrow{x_1} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \xrightarrow{x_2} \begin{bmatrix} x_2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}. \tag{1.12}$$

Therefore, we call the right null space of the Macaulay matrix **scalar multi-shift-invariant**. In Chapter 2, we exploit this shift-invariance to solve systems of multivariate polynomial equations. Not only its right null space, but also the column space of the Macaulay matrix allows us to retrieve the common roots of the generating seed polynomials (Section 2.5).

Case III: polynomial eigenvalue problems. The third case leaves the field of algebraic geometry and enters the area of linear algebra. The **polynomial eigenvalue problem**, for example,

$$\boldsymbol{\mathcal{M}}(\lambda)\boldsymbol{z} = \big(\boldsymbol{A}_0 + \boldsymbol{A}_1\lambda + \boldsymbol{A}_2\lambda^2 + \boldsymbol{A}_3\lambda^3\big)\boldsymbol{z} = \boldsymbol{0}, \tag{1.13}$$

consists in finding the scalars λ and non-zero vectors $\boldsymbol{z}$ that solve this matrix equation. Evidently, we can multiply (1.13) with monomials λ^α of increasing degree (i.e., $1, \lambda, \lambda^2, \lambda^3, \dots$). The structured matrix that corresponds to this **block forwards single-shift recursion** is the so-called **block Toeplitz matrix**. For example, for (1.13),

$$\underbrace{\begin{array}{c} \\ 1 \\ \lambda \\ \lambda^2 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} 1 & \lambda\boldsymbol{z} & \lambda^2\boldsymbol{z} & \lambda^3\boldsymbol{z} & \lambda^4\boldsymbol{z} & \lambda^5\boldsymbol{z} \end{array} \\ \begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_0 & \boldsymbol{A}_1 & \boldsymbol{A}_2 & \boldsymbol{A}_3 \end{bmatrix} \end{array}}_{\text{block Toeplitz matrix } \boldsymbol{T}} \underbrace{\begin{bmatrix} \boldsymbol{z} \\ \lambda\boldsymbol{z} \\ \lambda^2\boldsymbol{z} \\ \lambda^3\boldsymbol{z} \\ \lambda^4\boldsymbol{z} \\ \lambda^5\boldsymbol{z} \end{bmatrix}}_{\boldsymbol{v}} = \boldsymbol{0}, \tag{1.14}$$

which is the same as (1.5) but with coefficient matrices instead of (scalar) coefficients. While the right null space of the Toeplitz matrix is scalar single-shift-invariant, the block Toeplitz matrix has a **block single-shift-invariant** right null space, meaning that entire block rows can be shifted (instead of rows):

$$\begin{bmatrix} \boldsymbol{z} \\ \lambda\boldsymbol{z} \\ \lambda^2\boldsymbol{z} \end{bmatrix} \xrightarrow{\lambda} \begin{bmatrix} \boldsymbol{z}\lambda \\ \lambda^2\boldsymbol{z} \\ \lambda^3\boldsymbol{z} \end{bmatrix}. \tag{1.15}$$

Chapter 3 dives deeper into the extension of scalar shifts to block shifts.

Case IV: multiparameter eigenvalue problems. Finally, the most general problem that fits into this block Macaulay matrix is the **rectangular**

MEP. The rectangular MEP is discussed in-depth in Section 3.2. A linear rectangular two-parameter eigenvalue problem is given by

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = (\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{01}\lambda_2)\boldsymbol{z} = \boldsymbol{0}, \tag{1.16}$$

where we search for the eigenvalues $\boldsymbol{\lambda}$ and non-zero eigenvectors $\boldsymbol{z}$ that solve this matrix equation. The corresponding FSR consists of multivariate monomials $\boldsymbol{\lambda}^{\boldsymbol{\omega}}$ and we speak about the **block forward multi-shift-recursion**, because the shifts of the coefficient matrices with the different eigenvalues generate the pattern of the corresponding **block Macaulay matrix**. For the seed equation in (1.16), the block Macaulay matrix has the following structure:

$$\begin{array}{c} \\ 1 \\ \lambda_1 \\ \lambda_2 \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} 1 & \lambda_1 z & \lambda_2 z & \lambda_1^2 z & \lambda_1\lambda_2 z & \lambda_2^2 z \end{array} \\ \underbrace{\begin{bmatrix} \boldsymbol{A}_{00} & \boldsymbol{A}_{10} & \boldsymbol{A}_{01} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{00} & \boldsymbol{0} & \boldsymbol{A}_{10} & \boldsymbol{A}_{01} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{00} & \boldsymbol{0} & \boldsymbol{A}_{10} & \boldsymbol{A}_{01} \end{bmatrix}}_{\text{block Macaulay matrix } \boldsymbol{M}} \end{array} \underbrace{\begin{bmatrix} \boldsymbol{z} \\ \lambda_1 \boldsymbol{z} \\ \lambda_2 \boldsymbol{z} \\ \lambda_1^2 \boldsymbol{z} \\ \lambda_1\lambda_2 \boldsymbol{z} \\ \lambda_2^2 \boldsymbol{z} \end{bmatrix}}_{\boldsymbol{v}} = \boldsymbol{0}. \tag{1.17}$$

The block Macaulay matrix enforces its right null space to have a **block multi-shift-invariant** structure, which combines the block single-shift-invariant structure of the block Toeplitz matrix with the scalar multi-shift-invariant structure of the Macaulay matrix. For example, the structure of the right null space of the block Macaulay matrix in (1.17) gives the following shift options:

$$\begin{bmatrix} \lambda_1 \boldsymbol{z} \\ \lambda_1^2 \boldsymbol{z} \\ \lambda_1\lambda_2 \boldsymbol{z} \end{bmatrix} \xleftarrow{\lambda_1} \begin{bmatrix} \boldsymbol{z} \\ \lambda_1 \boldsymbol{z} \\ \lambda_2 \boldsymbol{z} \end{bmatrix} \xrightarrow{\lambda_2} \begin{bmatrix} \lambda_2 \boldsymbol{z} \\ \lambda_1\lambda_2 \boldsymbol{z} \\ \lambda_2^2 \boldsymbol{z} \end{bmatrix}. \tag{1.18}$$

The shift-invariance of the right null space and column space of the block Macaulay matrix is exploited in Chapter 3, in order to solve rectangular MEPs.

1.3 Research objectives

The central research question of this dissertation is twofold; it is about multivariate root-finding and multiparameter eigenvalue-finding. Before mapping both critical, mathematical problems into clear research objectives (ROs) that we want to tackle in this dissertation, we need to phrase these two problems more rigorously.

Multivariate root-finding. Let $\mathcal{P}^n = \mathbb{C}[x_1, \ldots, x_n]$ denote the ring of n-variate polynomials with coefficients in $\mathbb{C}$. A polynomial $p(\boldsymbol{x})$ is an element of this ring and defines a function (summation runs over non-zero coefficients $c_{\boldsymbol{\alpha}}$)

$$p : \mathbb{C}^n \to \mathbb{C} : \boldsymbol{x} \mapsto p(\boldsymbol{x}) = \sum_{c_{\boldsymbol{\alpha}} \neq 0} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}} \tag{1.19}$$

that maps a point from the affine space $\mathbb{C}^n$ onto a function value, with complex coefficients $c_{\boldsymbol{\alpha}} \in \mathbb{C}$. Given s elements $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \in \mathcal{P}^n$, we define the map

$$\mathcal{S} : \mathbb{C}^n \to \mathbb{C}^s : \boldsymbol{x} \mapsto (p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})). \tag{1.20}$$

We are interested in the pre-image of the origin in $\mathbb{C}^s$ under this map,

$$\mathcal{S}^{-1}(\mathbf{0}) = \{\boldsymbol{x} \in \mathbb{C}^n : \mathcal{S}(\boldsymbol{x}) = 0\}, \tag{1.21}$$

which are the common roots of the polynomials. Therefore, $\mathcal{S}^{-1}(\mathbf{0})$ is called the set of solutions of the system of polynomial equations defined by these polynomials. In this context, by **multivariate root-finding**, we mean computing $\mathcal{S}^{-1}(\mathbf{0})$.

Multiparameter eigenvalue-finding. Consider an n-variate polynomial matrix pencil (summation runs over all non-zero coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}}$):

$$\boldsymbol{\mathcal{M}} : \mathbb{C}^n \to \mathbb{C}^{k \times l} : \boldsymbol{\lambda} \mapsto \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) = \sum_{\boldsymbol{A}_{\boldsymbol{\omega}} \neq \mathbf{0}} \boldsymbol{A}_{\boldsymbol{\omega}} \boldsymbol{\lambda}^{\boldsymbol{\omega}}, \tag{1.22}$$

that maps a point from the affine space $\mathbb{C}^n$ onto a matrix value, with matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{C}^{k \times l}$ and $k \geq l + n - 1$. Similar to the multivariate root-finding problem, we can define another map

$$\mathcal{S}' : \mathbb{C}^n \times \left(\mathbb{C}^l \setminus \{\mathbf{0}\}\right) \to \mathbb{C}^k : (\boldsymbol{\lambda}, \boldsymbol{z}) \mapsto \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z}. \tag{1.23}$$

We are interested in the pre-image of the origin in $\mathbb{C}^k$ under this map,

$$\mathcal{S}'^{-1}(\mathbf{0}) = \left\{(\boldsymbol{\lambda}, \boldsymbol{z}) \in \mathbb{C}^n \times \left(\mathbb{C}^l \setminus \{\mathbf{0}\}\right) : \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \mathbf{0}\right\}. \tag{1.24}$$

This set consists of all pairs that satisfy the relation $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \mathbf{0}$, which are called eigenpairs. Note that this set is always infinite, because the eigenvectors $\boldsymbol{z}$ associated to an eigenvalue $\boldsymbol{\lambda}$ span an eigenspace. We are only interested in the basis vectors for the eigenspace associated with every eigenvalue $\boldsymbol{\lambda}$. To alleviate this issue, we formulate a different map

$$\mathcal{S} : \mathbb{C}^n \to \mathbb{S}^{k \times l} : \boldsymbol{\lambda} \mapsto \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}), \tag{1.25}$$

where $\mathbb{S}^{k \times l}$ is the subspace of rank deficient matrices in $\mathbb{C}^{k \times l}$. We are interested in the pre-image of $\mathbb{S}^{k \times l}$ under this map,

$$\mathcal{S}^{-1}(\mathbf{0}) = \{\boldsymbol{\lambda} \in \mathbb{C}^n : \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) \text{ is rank deficient}\}. \tag{1.26}$$

Therefore, $\mathcal{S}^{-1}(\mathbf{0})$ is called the set of eigenvalues of the MEP defined by this matrix polynomial. In this context, by **multiparameter eigenvalue-finding**, we mean computing $\mathcal{S}^{-1}(\mathbf{0})$. To obtain $\mathcal{S}'^{-1}(\mathbf{0})$, a basis for the eigenspace associated with every $\boldsymbol{\lambda}$ has to be computed.

What does "solving" mean? For both mathematical problems, we have to ask ourselves what does "solving" mean? The answer is not that simple! There are some issues when "solving" both problems:

- The set $\mathcal{S}^{-1}(\mathbf{0})$ may be infinite.
- There may be no algorithm that computes the coordinates of $\mathcal{S}^{-1}(\mathbf{0})$ exactly in finite time.

In this dissertation, we (mainly) consider problems where the solution set is finite, i.e., zero-dimensional. The text formulates necessary conditions for both fundamental problems to have a zero-dimensional solution set. The second issue means that there is no hope for developing algorithms for computing the coordinates of the solutions of one of the two fundamental problems exactly in finite time[5]. Motivated by this, by "solving", we mean computing satisfactory approximations of the coordinates of the solutions of each of the mathematical problems via numerical algorithms. To determine whether an approximation is satisfactory, we introduce error measurements later on in this text.

Different research objectives. Now, we give an overview of the different ROs considered in this dissertation and explain how the two parts of the central research question return in every RO[6]. The outline in Section 1.5 relates these ROs to the corresponding chapters and publications. We want to stress that every RO can be mapped onto one of the obtained research contributions (RCs): every ROx.y leads to an RCx.y (Section 8.1).

RO1: Unifying (block) Macaulay matrix approach

The obvious first step to extend the Macaulay matrix to the block Macaulay matrix and to develop the so-called unifying (block) Macaulay matrix approach. This quest is the subject of the first research objective (RO1). The goal is to obtain a unifying approach to tackle both systems of multivariate polynomials and rectangular MEPs. The desired result of this RO is very straightforward: develop novel approaches to tackle the four seed problems of Section 1.2. We want to establish connections between the four seed equations and (better) understand the unifying (block) Macaulay matrix. This results into three smaller, more specific, ROs: RO1.1, RO1.2, and RO1.3.

RO1	Create the unifying (block) Macaulay matrix framework and develop novel algorithms to solve the different seed problems.

[5] In the case of the univariate root-finding problem, for example, it is known that there does not always exist an expression in radicals to determine the solutions of the polynomial equations.

[6] Notice that the color of every RO agrees with the mathematical problem that it belongs to: **red** has to do with multivariate root-finding, **green** is related to multiparameter eigenvalue-finding, and **blue** combines aspects from both problems.

For multivariate root-finding, the (numerical) linear algebra connection between the seed equations and the Macaulay matrix has already been established by Batselier [21] and Dreesen [78]. However, leveraging other aspects of this connection, together with new insights from numerical linear algebra, should provide us with better understanding and novel algorithms.

RO1.1 Create new Macaulay matrix algorithms to solve systems of multivariate polynomials more efficiently.

While (multivariate) polynomials are intimately linked with the Macaulay matrix, much less is known about rectangular MEPs. These problems have not yet penetrated the linear algebra community as much as the well-known one-parameter eigenvalue problems, although they are a quite natural generalization. Moreover, multiparameter spectral theory typically (except for some scarce exceptions, e.g., [118, 134, 216]) only considers square formulations of the MEP. Therefore, we want to formalize the rectangular MEP and explore its properties rigorously in this RO. By extending the Macaulay matrix to the block Macaulay matrix, we aim to develop tools for solving this type of problems. Together with the existing Macaulay matrix, this extension will create the necessary tools for a unifying (block) Macaulay matrix approach.

RO1.2 Extend the Macaulay matrix to the block Macaulay matrix in order to solve polynomial rectangular MEPs.

While we approach the Macaulay matrix mainly from a multivariate polynomial system solving perspective, and hence, not consider all the fundamental subspaces of this important matrix, we still want to create a better understanding of this matrix. Therefore, an investigation from a linear algebra point of view must shed some light on the connections between the fundamental subspaces of the Macaulay matrix and the (algebraic geometry) properties of its generating polynomials. The goal is to create a complete overview of the known (and a few new) connections.

RO1.3 Explore the different fundamental subspaces of the Macaulay matrix and relate them to the generating polynomials.

RO2: Efficient (block) Macaulay matrix software

While the theoretical development of the (block) Macaulay matrix provides a nice tool to investigate a wide variety of problems, off-the-shelf implementations of (block) Macaulay matrix algorithms are limited to small toy problems. To tackle more interesting problems, we need efficient algorithms collected in a coherent, user-friendly toolbox. This is the topic of the second research objective (RO2), which breaks up into two smaller ROs: RO2.1 and RO2.2.

RO2	Develop efficient and reliable algorithms for the (block) Macaulay matrix approaches.

The first step towards this toolbox is the adaptation of the naive algorithms into algorithms that exploit the sparsity and structure of the (block) Macaulay matrix. This is a necessary step to improve the computation time and memory usage. Batselier et al. [22] have already developed a recursive approach to compute the right null space of a (scalar) Macaulay matrix, but in this RO we want to create more efficient algorithms and deal with the block Macaulay matrix.

RO2.1	Exploit sparsity and structure in the (block) Macaulay matrix to improve computation time and memory usage.

The second step is to combine all the developed algorithms into a software toolbox that is user-friendly and fast. This requires a careful implementation of the available and new algorithms, while keeping user-friendliness in mind.

RO2.2	Combine all advancements in one coherent software toolbox that is user-friendly and fast.

RO3: Applications from systems theory

Finally, we want to leverage the gained insights and use the novel (block) Macaulay matrix algorithms that we develop in RO1 and RO2 to tackle the problems that emerge in applications, for example from systems theory. These applications have, in a certain sense, created the need for (block) Macaulay matrix algorithms. In the third research objective (RO3), we deal with applications in a globally optimal way, leveraging the fact that we work with accurate tools from (numerical) linear algebra. We consider three specific key examples:

- Rephrasing the multivariate polynomial optimization problem in complex variables as a rectangular MEP.
- Finding the globally optimal $\mathcal{H}_2$-norm reduced-order model from a high-order SISO LTI model.
- Identifying the globally optimal least-squares model parameters of misfit-versus-latency models (cf., the motivational example in Section 1.1).

The overarching question of RO3 is "Can we retrieve globally optimal solutions for these key examples from systems theory?" To answer this question positively, we rephrase the problems as a system of multivariate polynomial equations or rectangular MEP. As it turns out, the reformulation as a rectangular MEP has certain advantages.

RO3 Use the novel (block) Macaulay matrix algorithms to tackle problems from systems theory with global optimality in mind.

1.4 MacaulayLab: About the implementation

An essential deliverable of this dissertation is the software that has been developed in parallel with the theoretical research. The new algorithms that have been developed in this dissertation (but also adaptations of some existing algorithms) are implemented in MacaulayLab, which is a toolbox entirely written in Matlab. The current version of the toolbox contains almost 8000 lines of code and uses (block) Macaulay matrices to solve systems of multivariate polynomial equations and rectangular MEPs, using well-established numerical linear algebra tools in floating-point arithmetics (for example, the singular value, eigenvalue and QR decomposition). Some features of MacaulayLab are:

- iterative, recursive, and sparse algorithms to solve the four seed problems;
- a monomial ordering and polynomial basis independent implementation (the user can decide to work in the standard monomial basis, the Chebyshev polynomial basis, or in his/her own polynomial basis);
- different approaches to determine upper bounds on the number of (affine) solutions; and
- a database with many examples and benchmark problems.

Chapter 6 discusses the different capabilities and implementation of the toolbox in more detail. It also positions the toolbox with respect to other solvers available in Matlab.

In this text, (numerical) examples are often accompanied by a code block that contains the necessary steps to replicate the results with MacaulayLab. Matlab functions, like `macaulay`, are also highlighted in the body of the text to differentiate them from other words. We want to encourage the reader to experiment with the available toolbox.

Code 1.1. This is an example of a code block, in which the Matlab code is highlighted in green. Typically, the code block also contains some additional information about the involved functions or expected result.

```
>> M = macaulay(toy1,3);
>> size(M)

ans =
    9     10
```

MacaulayLab is available at `www.macaulaylab.net` and we provide the user-manual of the toolbox in Appendix D. All numerical examples presented in this dissertation are performed on a MacBook Pro that has an M1 CPU (2020) working at 3.2 GHz (8 cores) and 16 GB random-access memory (RAM) or on a Red Hat Enterprise Linux server infrastructure with 177 nodes that have each two Intel Xeon Platinum 8360Y CPUs working at 2.4 GHz (36 Ice Lake cores each) and 256 GB RAM (or 2048 GB RAM for the "big memory nodes"). We only used the server infrastructure for large-scale experiments. We indicate results obtained via these high-performance nodes always with the symbol ♣.

1.5 Outline of the dissertation

Now, we give the outline of the dissertation and summarize the content of each of the next chapters. The body of the text is organized into three different parts: Part I: Fundamentals, Part II: Algorithms, and Part III: Applications. Figure 1.3 contains a schematic overview of the different chapters, in which links between chapters (and appendices) are indicated by (dashed) arrows. The tackled ROs are also indicated per chapter. Most of the chapters are derived from (one or multiple) published articles, for that reason every chapter is rather self-containing. Note that in some chapters, for the sake of readability, some of the historical information and additional references are moved to a dedicated *historical and bibliographical notes* section.

Part I: Fundamentals

The first part of the dissertation develops the novel, unifying block Macaulay matrix approach (RO1). It considers the four cases of seed problems that we want to tackle in this dissertation: (univariate and multivariate) polynomial equations in Chapter 2 and (one-parameter and multiparameter) eigenvalue problems in Chapter 3. Both chapters formalize the problems clearly, set the required notation, and provide the necessary tools, like important definitions and theorems, to tackle them. Chapters 2 and 3 develop (block) Macaulay matrix approaches to solve these seed problems. Furthermore, the algebraic properties of the different fundamental subspaces of the Macaulay matrix are investigated, from a linear algebra point of view, in Chapter 4.

- In Chapter 2, we revisit the Macaulay matrix and discuss some of its properties. We focus on the right null space and column space to solve systems of multivariate polynomial equations, using numerical linear algebra tools only (RO1.1).

  (article relevant for this chapter: [258])

- Afterwards, in Chapter 3, we investigate the rectangular MEP and extend the Macaulay matrix to the block Macaulay matrix. We propose two novel algorithms to solve rectangular MEPs (RO1.2).

  (article relevant for this chapter: [261])

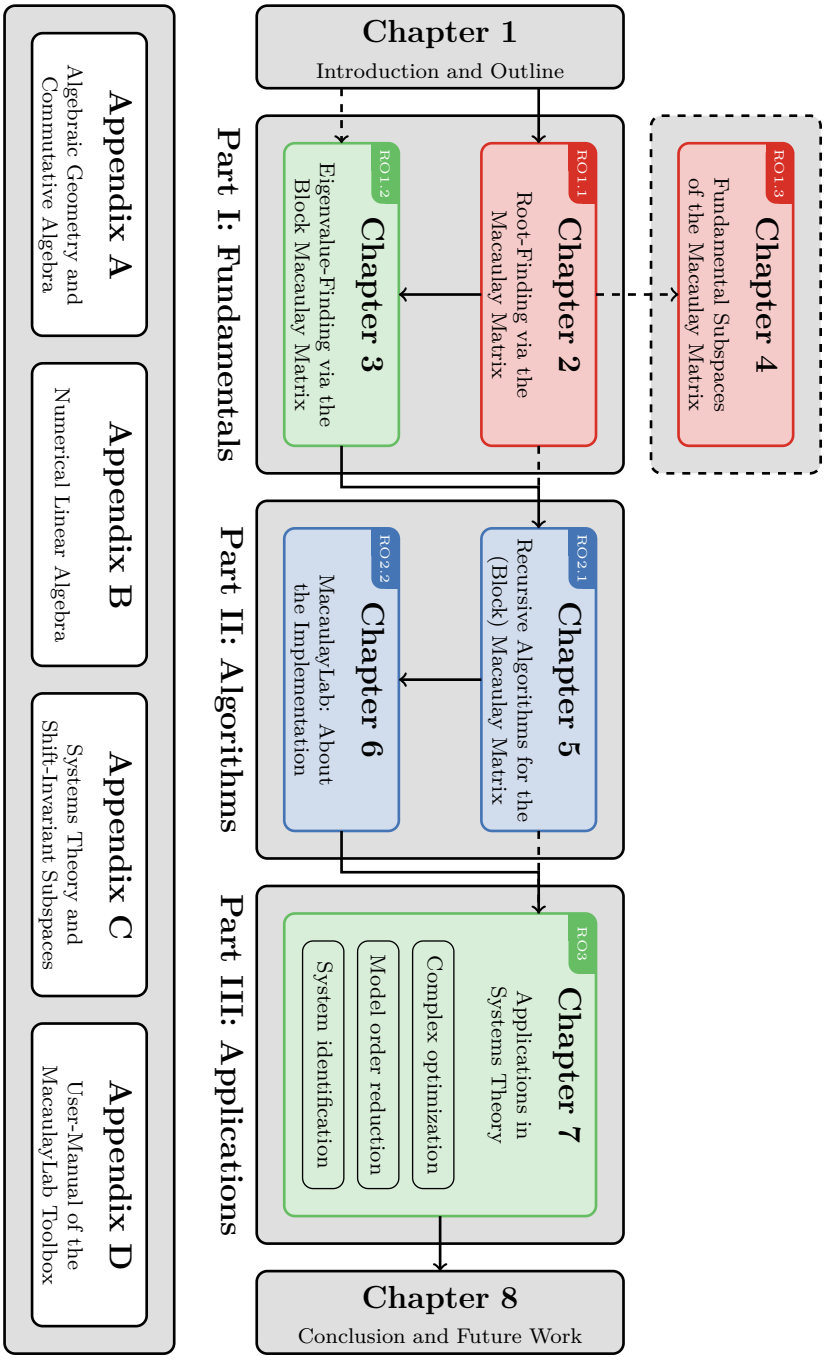


Figure 1.3. Overview of the the different chapters of this dissertation. The dashed arrows indicate the link between the different parts of the text, while the full arrows propose a natural reading order of the chapters.

- Chapter 4 summarizes the algebraic properties of the polynomials that are hiding in the different fundamental subspaces of the Macaulay matrix (RO1.3).

Part II: Algorithms

- Chapter 5 contains recursive algorithms to update a numerical basis matrix of the null space of the block row, block Toeplitz, and block Macaulay matrix. These recursive algorithms give rise to double recursive (block) Macaulay matrix algorithms to deal with systems of multivariate polynomial equations and MEPs. The techniques from this chapter help to exploit the sparsity and structure in the (block) Macaulay matrix, leading to more efficient solution algorithms (RO2.1).

(articles relevant for this chapter: [260, 262])

- Chapter 6 gives an overview of the MacaulayLab toolbox; its structure, the most important implementation decisions capabilities, a comparison with other Matlab toolboxes, and the test database (RO2.2).

(technical report relevant for this chapter: [257])

Part III: Applications

The third part of the text applies the algorithms described above to applications in systems theory (RO3). These are the applications that have pushed us in the direction of algebraic geometry and multiparameter spectral theory. The insights we gain and algorithms we develop in Parts I and II are essential tools to solve the resulting systems of multivariate polynomial equations and rectangular MEPs. In Chapter 7, we give an overview of a wide range of problems from various scientific domains that can be phrased as a system of multivariate polynomial equations or an eigenvalue problem, but we focus on three key examples from systems theory in particular (RO3). Although each of the globally optimal methodologies is a very interesting research problem in its own right, we consider them here as motivational examples to evaluate the contributions from Parts I and II.

- In essence, many of the problems in systems theory are multivariate polynomial optimization problems. Therefore, we consider multivariate polynomial optimization problems and show how to deal with real-valued polynomial cost functions in complex variables. We extend the relation between complex polynomial optimization of univariate polynomial cost functions and polynomial eigenvalue problems to the multivariate case (a rectangular MEP).

(article relevant for this section: [263])

- Next, we show a reformulation of the $\mathcal{H}_2$-norm optimal model order reduction problem of SISO LTI high-order models into a rectangular MEP. While the original article ends up with an inhomogeneous rectangular MEP, we show that the globally optimal reduced-order model can be found as one of the eigenvalues of a homogeneous rectangular MEP, allowing us to use the algorithms from Chapter 3.

  (article relevant for this section: [3])

- Lastly, we consider least-squares system identification problems that can be rephrased as rectangular MEPs. We deal with misfit-versus-latency models, but the focus of Chapter 7 is on ARMA models.

  (articles relevant for this section: [152, 259])

Conclusion and Appendices

Afterwards, Chapter 8 summarizes the different RCs of this dissertation and points at ideas for future work. The text is supported by four appendices that contain additional, but non-essential, material.

- In Appendix A, we provide the reader with a rehearsal of some relevant concepts from algebraic geometry and commutative algebra can be encountered in the text.
- The essential tools from numerical linear algebra are summarized in Appendix B.
- Appendix C discusses the concept of (backward) shift-invariance in-depth, an important notion within the block Macaulay matrix framework.
- Finally, Appendix D contains the user-manual of MacaulayLab, the software toolbox that has been developed in parallel with the theoretical research in this dissertation.

PART

I

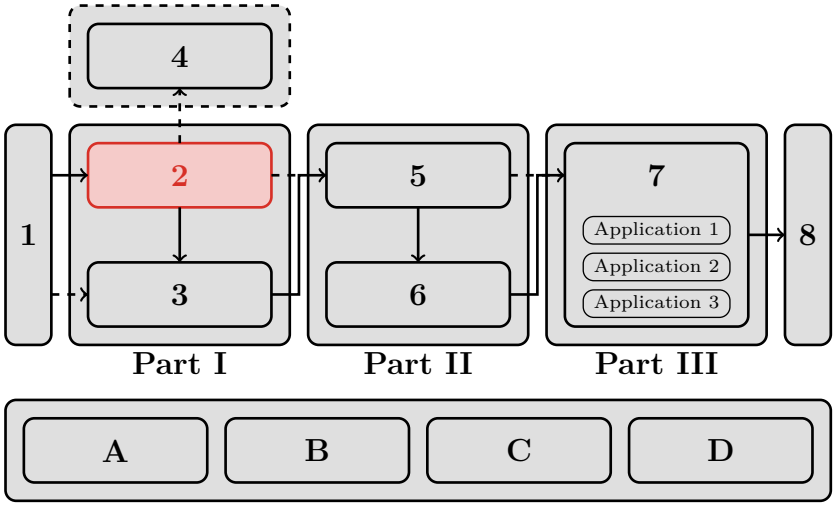
Fundamentals

٢

Root-Finding via the Macaulay Matrix

We revisit the intricate relation between multivariate polynomials and the Macaulay matrix that is constructed from the coefficients of these polynomials. While each of the four fundamental subspaces of the Macaulay matrix has a profound algebraic connection with its generating polynomials, the focus of this chapter is solely on the right null space and column space, since they provide a way to retrieve the common roots of the system of multivariate polynomials. These common roots often play an important role in applications, for example, when identifying the globally optimal parameters of an autoregressive moving-average model.

The right null space and column space of the Macaulay matrix enable numerical linear algebra algorithms to solve the system of multivariate polynomial equations that generates this Macaulay matrix. We show in this chapter how a multidimensional realization problem in the right null space of the Macaulay matrix results in one (or multiple) eigenvalue problem(s), the eigenvalues and eigenvectors of which yield the solutions of the polynomial equations. We also develop a complementary solution approach, which considers the column space of the Macaulay matrix instead of its right null space. This chapter includes an in-depth discussion of the different aspects of multivariate root-finding via the Macaulay matrix.



Contribution. Our main contribution is a novel, complementary approach that finds the common roots of a system of multivariate polynomials from the information in the column space of the Macaulay matrix, without computing a numerical basis matrix of its right null space.

Relevant article. This chapter is an adaptation of [258]. The candidate was the main author of the original article, developed the theoretical contributions, and implemented the accompanying code and experiments. This chapter differs in several parts from the original article, in order to give a more complete and didactic overview.

Outline. Firstly, in Section 2.1, we introduce the problem of multivariate root-finding and sketch the current state-of-the-art. Section 2.2, subsequently, contains a summary about multivariate polynomials and their most important properties, i.e., the algebraic geometry concepts that are used throughout the remainder of the text. Afterwards, we rigorously define the Macaulay matrix and highlight two of its four fundamental subspaces in Section 2.3. We show how to solve systems of multivariate polynomial equations using the right null space of the Macaulay matrix in Section 2.4, and we translate this multivariate system solving approach to the column space in Section 2.5. Section 2.6 includes several extensions to both multivariate root-finding approaches. Finally, in Section 2.7, we conclude this chapter and point at ideas for future research.

2.1 Introduction

Determining the roots of a univariate polynomial or the common roots of a system of multivariate polynomials is a very old problem [66, 194]. It has a long and rich history that can be traced back to the Ancient Near East. For example, the Babylonians and Egyptians (about 2000 B.C.) already solved linear and quadratic equations by methods similar to those we teach to high school students today [194]. The difficulty of the equations and the performance of the solution approaches have evolved a lot since then. Because many problems in science and engineering require some kind of mathematical modeling step and (multivariate) polynomials provide a natural tool for doing this, they appear in a myriad of today's applications, e.g., in robotics, economics, computational chemistry and biology, computer vision, and systems theory. Section 7.2.1 contains a more elaborate overview of possible application domains with examples. Also in this dissertation, we study multivariate[1] polynomials, in particular the numerical linear algebra methods that find their common roots.

Within mathematics, algebraic geometry is the branch that studies (multivariate) polynomial equations and varieties, which are the geometrical objects defined by the zero sets of these polynomials [66]. The roots of algebraic geometry go back to Descartes' introduction of coordinates to describe points in the Euclidean space and the idea in his book *La Géométrie* of describing curves and surfaces by algebraic equations [66, 75, 211]. For most of the 19th and 20th century, the main focus of algebraic geometry was more on abstract algebra than on polynomial system solving, resulting powerful general theorems and striking conjectures [66].

Although the centuries of advancements in the field had developed a detailed understanding of many specific examples, the symbolic tools available at that time were unfeasible for polynomial systems coming from applications, due to the scale of such problems and the inexact nature of the coefficients of the polynomials. The scale of the problems would require too much time for the symbolic tools to terminate, while the inexact coefficients in measurements would result in very large integers when represented as rational numbers. This clearly establishes the need for more robust numerical algorithms that produce reliable results in floating-point arithmetic. Surprisingly, there was not much attention for these numerical algorithms until the second half of the 20th century, when the development of Buchberger's algorithm sparked a renewed interest in the computational aspects of algebraic geometry [66, 242]. Also from numerical analysis and numerical linear algebra, new algorithms for multivariate polynomial system solving have emerged since the end of the previous century [78, 194].

[1] The problem of solving a multivariate polynomial system has two very recognizable special cases: univariate polynomial equations (in which the number of variables and polynomial equations is equal to one) and systems of multivariate linear equations (in which the total degree of every equation is equal to one). Both special cases are linear algebra problems and there exist dedicated solution methods that can tackle very large problem instances. Note that the Macaulay matrix approaches presented in this chapter could also be used to tackle these two special cases.

Multivariate polynomial system solving

Typically, multivariate polynomial system solving approaches are divided into two classes of methods: (i) algebraic root-finding methods transform the polynomials into a univariate root-finding problem or eigenvalue problem via algebraic manipulations, while (ii) iterative root-finding methods employ numerical techniques to identify (some of) the solutions. A more elaborate overview can be found in the "historical and bibliographical notes" of this chapter.

Algebraic root-finding methods

These methods use algebraic operations to deduce the structure of the quotient ring described by the system of multivariate polynomials, which contains the common roots of these polynomials. One way to uncover this structure is via a Gröbner basis. The methods of Faugère [88, 89] and their extensions are currently the most efficient methods to compute a Gröbner basis. Gröbner bases have dominated computer algebra since their inception, but remain computationally very expensive and are based upon infinite precision (symbolic) computations and, therefore, employ rational numbers. This often results in huge coefficients (for example, coefficients with tens or hundreds of digits), which means that their extensions to floating-point arithmetic are known to be rather cumbersome [138, 230]. The numerical instability of Gröbner basis computations have limited their use in a numerical context [242]. This major drawback has motivated the introduction of border bases [14, 181] and truncated normal forms [185, 243], which greatly improve the numerical stability.

Another algebraic way to solve multivariate polynomial equations is by using (numerical) linear algebra operations to deduce the structure of the quotient ring. This approach finds its origin with the work on resultants in the 18th and 19th century, but has been neglected in most of the algebraic geometry literature until the work of Lazard and Stetter (and coworkers) during the 1980s: Lazard [153, 154] observed the resemblance between Buchberger's algorithm and Gaussian elimination of a Macaulay matrix, while Auzinger and Stetter [14] rigorously established the connection between multivariate polynomial system solving and eigenvalue decompositions. This was further explored by, among others, Corless et al. [61], Emiris and Mourrain [87], Hanzon and Jibetean [106], Mourrain and Pan [183], and Stetter [229]. However, Stetter [230] observed that, at that time, the only way to obtain the common roots of multivariate polynomials as the eigenvalues of a matrix using commonly available software was via Gröbner basis methods. Hence, the approach was partially symbolic of nature. Batselier, De Moor, and Dreesen [21, 78, 80] have overcome this hurdle and developed a pure (numerical) linear algebra approach to multivariate root-finding, using the right null space of a rectangular Macaulay matrix.

Iterative root-finding methods

Iterative root-finding methods, e.g., Newton and quasi-Newton methods, do not suffer from the floating-point issues that Gröbner basis computations experi-

ence, but are heuristic: they do not guarantee to find the correct solutions and strongly depend on their initial guesses. One particular type of iterative algorithms to solve systems of multivariate polynomial equations are the homotopy continuation methods, which employ a hybrid mixture of techniques from algebraic geometry and nonlinear optimization to continuously deform a starting system with known solutions into the required system with unknown solutions, while tracking the solutions [157, 265]). Homotopy continuation methods are inherently parallel, i.e., each isolated solution can be computed independently. They are currently among the most competitive algorithms to solve systems of multivariate polynomial equations, although issues with ill-conditioning still exist (i.e., path jumping). Their main disadvantage is that they only work for square systems, in which the number of equations is equal to the number of variables.

Another type of iterative root-finding methods are subdivision methods, which employ domain reductions for an iterative refinement of the subregions where the common roots of the multivariate polynomials may be located [184].

A numerical Macaulay matrix perspective

In this chapter, we (re-)consider the intimate link between multivariate polynomial equations and linear algebra. We tackle the multivariate root-finding problem via the Macaulay matrix, as set out in Section 1.3:

> *We present numerical linear algebra approaches to find the common roots of a system of multivariate polynomials, where we assume that the set of common roots is zero-dimensional. These approaches aim to compute satisfactory approximations of the coordinates of the common roots in the affine (complex) space.*

The central object of this chapter is the Macaulay matrix, a sparse and structured matrix that is constructed from the coefficients of multivariate polynomials. Macaulay [159, 160] introduced this matrix for the first time in 1902 in the context of resultant theory, generalizing earlier work done by Sylvester [239]. Each of the four fundamental subspaces of the Macaulay matrix has a profound algebraic connection with the multivariate polynomials that generate that matrix, but, in this chapter, we focus solely on the right null space and column space, since they enable numerical linear algebra approaches to retrieve the common roots of the polynomial system.

We revisit the multivariate root-finding approach via right null space of the Macaulay matrix, which has originally been formulated by Batselier, De Moor, and Dreesen [21, 78, 80], in a clear and didactic fashion. A multidimensional realization problem in that right null space results in one (or multiple) eigenvalue problem(s), the eigenvalues and eigenvectors of which yield the common roots of the multivariate polynomials that generate the Macaulay matrix. Recently, we have shown that it is also possible to retrieve the common roots from an equivalent multidimensional realization problem in the column space of that Macaulay matrix [258]. Both numerical linear algebra approaches rely on the decades of advancements in numerical linear algebra, using well-established floating-point algorithms to compute the necessary (intermediate) results.

Motivational example. As already mentioned in Section 1.1, the problem of finding the globally optimal parameters of the autoregressive moving-average (ARMA) model corresponds to solving a system of multivariate polynomial equations (Section 7.5). Indeed, a multivariate polynomial optimization problem can be cast as a polynomial system via the Lagrangian [190]. For example, if we consider a sequence of $N = 4$ output samples $y_k \in \mathbb{R}$ and a first-order ARMA model, then we can rewrite (1.2) via the Lagrangian as a system of $s = 9$ polynomial equations in $n = 9$ variables,

$$\widehat{\boldsymbol{x}} = (\alpha, \gamma, e_1, e_2, e_3, e_4, l_1, l_2, l_3), \tag{2.1}$$

where $l_i \in \mathbb{C}$ are the Lagrange multipliers. The corresponding polynomial system that yields the optimal first-order ARMA model for the given output sequence (after eliminating the latent input samples e_k) is

$$\begin{cases} p_1(\boldsymbol{x}) = y_2 + \alpha_1 y_1 + 4l_1 + 4\gamma_1 l_2 + 4\gamma_1^2 l_1 = 0, \\ p_2(\boldsymbol{x}) = y_3 + \alpha_1 y_2 + 4l_2 + 4\gamma_1 l_3 + 4\gamma_1^2 l_2 + 4\gamma_1 l_1 = 0, \\ p_3(\boldsymbol{x}) = y_4 + \alpha_1 y_3 + 4l_3 + 4\gamma_1 l_2 + 4\gamma_1^2 l_3 = 0, \\ p_4(\boldsymbol{x}) = \gamma_1 l_1^2 + \gamma_1 l_2^2 + l_1 l_2 + \gamma_1 l_3^2 + l_2 l_3 = 0, \\ p_5(\boldsymbol{x}) = y_1 l_1 + y_2 l_2 + y_3 l_3 = 0, \end{cases} \tag{2.2}$$

where the $n = 5$ variables are

$$\boldsymbol{x} = (\alpha, \gamma, l_1, l_2, l_3). \tag{2.3}$$

We describe in this chapter numerical linear algebra approaches to solve such a polynomial system.

2.2 Multivariate polynomials

A multivariate polynomial consists of a finite number of monomials.

Definition 2.1. A **monomial** in n variables $x_1, \ldots, x_n$ is the power product

$$\boldsymbol{x}^{\boldsymbol{\alpha}} = \prod_{i=1}^{n} x_i^{\alpha_i} = x_1^{\alpha_1} \cdots x_n^{\alpha_n}, \tag{2.4}$$

where the multi-index $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ is an n-tuple of nonnegative exponents and $|\boldsymbol{\alpha}| = \sum_{i=1}^{n} \alpha_i$ is the **total degree** of the monomial $\boldsymbol{x}^{\boldsymbol{\alpha}}$.

Example 2.1. Some examples of monomials are:

- in one variable, x^2 corresponds to $\alpha = (2)$, with $|\alpha| = 2$;

- in two variables, $x_1^3 x_2^4$ corresponds to $\boldsymbol{\alpha} = (3, 4)$, with $|\boldsymbol{\alpha}| = 7$; and
- in three variables, $x_1^2 x_3^8$ corresponds to $\boldsymbol{\alpha} = (2, 0, 8)$, with $|\boldsymbol{\alpha}| = 10$.

Note that $\boldsymbol{x}^{\mathbf{0}} = 1$ corresponds to $\boldsymbol{\alpha} = (0, \ldots, 0)$.

Definition 2.2. The **set of all monomials** in n variables is indicated by $\mathcal{C}^n$, while $\mathcal{C}_d^n \subset \mathcal{C}^n$ denotes the set of all the monomials in n variables with total degree $\leq d$.

A complex multivariate polynomial is a finite linear combination of different monomials with complex coefficients.

Definition 2.3. A **complex multivariate polynomial** $p(x_1, \ldots, x_n)$, or $p(\boldsymbol{x})$, in n variables is a finite linear combination of monomials $\boldsymbol{x}^{\boldsymbol{\alpha}} \in \mathcal{C}^n$ with coefficients $c_{\boldsymbol{\alpha}} \in \mathbb{C}$:

$$p(\boldsymbol{x}) = \sum_{\mathcal{A}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}, \tag{2.5}$$

where the summation runs over all the multi-indices $\boldsymbol{\alpha}$ in the set $\mathcal{A}$.

The set $\mathcal{A} = \{\boldsymbol{\alpha} : c_{\boldsymbol{\alpha}} \neq 0\} \subset \mathbb{N}^n$ contains all multi-indices $\boldsymbol{\alpha}$ present in $p(\boldsymbol{x})$ (i.e., all the multi-indices with non-vanishing coefficient $c_{\boldsymbol{\alpha}}$). This set $\mathcal{A} = \text{supp}(p(\boldsymbol{x}))$ is called the **support** of $p(\boldsymbol{x})$. The **total degree** of a (complex) multivariate polynomial, denoted by $\deg(p(\boldsymbol{x}))$, corresponds to the maximum total degree of the monomials in its support. Notice that different (complex) multivariate polynomials can have the same support.

Example 2.2. Consider the complex multivariate polynomial

$$p_1(\boldsymbol{x}) = 1 + 2x_1^2 x_2 + 3x_1 x_2^2, \tag{2.6}$$

which has a total degree equal to 3 and its support is

$$\mathcal{A} = \{(0, 0), (2, 1), (1, 2)\}. \tag{2.7}$$

Every tuple (i, j) in $\mathcal{A}$ corresponds to a monomial $x_1^i x_2^j$. The complex multivariate polynomial

$$p_2(\boldsymbol{x}) = 3\mathrm{i} + 10x_1^2 x_2 + (3 + \mathrm{i})x_1 x_2^2 \tag{2.8}$$

has the same total degree and support.

Definition 2.4. The **set of all complex polynomials** $p(\boldsymbol{x})$ in n variables is denoted by $\mathcal{P}^n$, while the set of all complex polynomials in n variables of total degree $\leq d$ is denoted by $\mathcal{P}_d^n \subset \mathcal{P}^n$.

The set of all complex polynomials $p(\boldsymbol{x})$ in n variables, $\mathcal{P}^n$, is a ring (Appendix A.1 contains more information about rings, fields, and vector spaces). In the literature, this polynomial ring is often denoted by $\mathbb{C}[x_1, \ldots, x_n]$, which we only use in this text when we need to stress the field $K = \mathbb{C}$ or the order of the variables[2]. The elements of $\mathcal{P}^n$ are multivariate polynomials, which can be seen as functions

$$p : \mathbb{C}^n \to \mathbb{C} : \boldsymbol{x} = (x_1, \ldots, x_n) \mapsto p(\boldsymbol{x}) = \sum_{\mathcal{A}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}. \tag{2.9}$$

In words: given a point $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{C}^n$ in the affine space (Definition 2.7), a multivariate polynomial (by replacing every x_i by a_i) yields a point $p(\boldsymbol{a})$ in $\mathbb{C}$.

Multivariate polynomials appear typically in systems of multivariate polynomial equations:

$$\begin{cases} p_1(\boldsymbol{x}) = \displaystyle\sum_{\mathcal{A}_1} c_{\boldsymbol{\alpha}_1} \boldsymbol{x}^{\boldsymbol{\alpha}_1} = 0, \\ \qquad \vdots \\ p_s(\boldsymbol{x}) = \displaystyle\sum_{\mathcal{A}_s} c_{\boldsymbol{\alpha}_s} \boldsymbol{x}^{\boldsymbol{\alpha}_s} = 0. \end{cases} \tag{2.10}$$

Given these s elements $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ of $\mathcal{P}^n$, the polynomial system defines the map

$$\mathcal{S} : \mathbb{C}^n \to \mathbb{C}^s : \boldsymbol{x} \mapsto (p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})). \tag{2.11}$$

In the multivariate root-finding problem, we are interested in the pre-image of the origin in $\mathbb{C}^s$ under this map, i.e., in

$$\mathcal{S}^{-1}(\mathbf{0}) = \{\boldsymbol{a} \in \mathbb{C}^n : \mathcal{S}(\boldsymbol{a}) = 0\}. \tag{2.12}$$

This set consists of all points $\boldsymbol{a} \in \mathbb{C}^n$ satisfying the relations

$$p_1(\boldsymbol{x}) = \cdots = p_s(\boldsymbol{x}) = 0. \tag{2.13}$$

Therefore, $\mathcal{S}^{-1}(\mathbf{0})$ is called the set of solutions of the polynomial equations. In the context of this dissertation, by solving the system, we mean computing $S^{-1}(\mathbf{0})$. Because it is in general not possible to compute $S^{-1}(\mathbf{0})$ exactly, we mean computing satisfactory approximations of the coordinates of the solutions in the affine space $\mathbb{C}^n$ via numerical algorithms. Furthermore, we assume that set of solutions is zero-dimensional (i.e., finite[3]), which typically happens when $s \geq n$.

The set of solutions $S^{-1}(\mathbf{0})$ of a system of multivariate polynomial equations corresponds to an affine variety[4]. After taking a closer look at the vector space

[2] In this text, we mainly consider polynomials with complex coefficients. We always consider the polynomial ring $\mathcal{P}^n = \mathbb{C}[x_1, \ldots, x_n]$, except when denoted otherwise. We drop, therefore, the term *complex* in the remainder of this text. However, there exist applications where the field K of the polynomial ring $K[x_1, \ldots, x_n]$ is not the field of complex numbers. Examples are the real field $\mathbb{R}$ in robotics [222] and the finite field $\mathbb{F}_q$ in cryptography [214].

[3] However, systems with infinitely many solutions can also be "solved", in a certain sense [221, 222].

[4] Some nuance is added in Footnote 10.

of polynomials (Section 2.2.1), we take a closer look at affine varieties and ideals, two important objects from algebraic geometry (Section 2.2.2). Afterwards, the number of solutions in $S^{-1}(\mathbf{0})$ is investigated (Section 2.2.3).

2.2.1 Vector space of polynomials

It is easy to show that the set of all polynomials in n variables of total degree $\leq d$, $\mathcal{P}_d^n$, forms a vector space over $\mathbb{C}$: addition and multiplication by a scalar are defined in a natural way. The zero element of the vector space is the zero polynomial $p(\boldsymbol{x}) \equiv 0$ (i.e., $c_{\boldsymbol{\alpha}} = 0$ for all $\boldsymbol{\alpha}$). The monomials in $\mathcal{C}_d^n$ form a canonical basis for $\mathcal{P}_d^n$: this canonical basis is called the **standard monomial basis**. The number of monomials in the set $\mathcal{C}_d^n$, which is also the dimension of $\mathcal{P}_d^n$, can be counted via combinatorics. The number of k-combinations, i.e., a selection of k elements where the order of selection does not matter, from a set with n elements is given by

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{2.14}$$

Proposition 2.1. The set $\mathcal{C}_d^n$ contains $\binom{d+n}{n}$ monomials, $\binom{d+n-1}{n}$ of them have a total degree exactly equal to d.

Proof. This proposition follows from formulas in combinatorics. □

Note that in practice (the cardinality of) the support of a multivariate polynomial may be very small compared to (the cardinality of) the associated basis. Such polynomials are called **sparse**, in analogy with the word *sparse* in linear algebra. Indeed, the number of different monomials in the set $\mathcal{C}_d^n$ grows rapidly with the number of variables n and the maximum total degree d. This combinatorial explosion is visible in Figure 2.1 and resurfaces several times in this dissertation.

Example 2.3. There are 1 221 759 monomials in the set $\mathcal{C}_d^n$ for degree $d = 40$ and $n = 5$ variables:

$$\#\mathcal{C}_{40}^5 = \binom{45}{5} = \frac{45!}{5!40!} = 1\,221\,759. \tag{2.15}$$

Code 2.1. Via `nbmonomials(d,n)`, it is possible to determine the number of monomials for degree d and n variables.

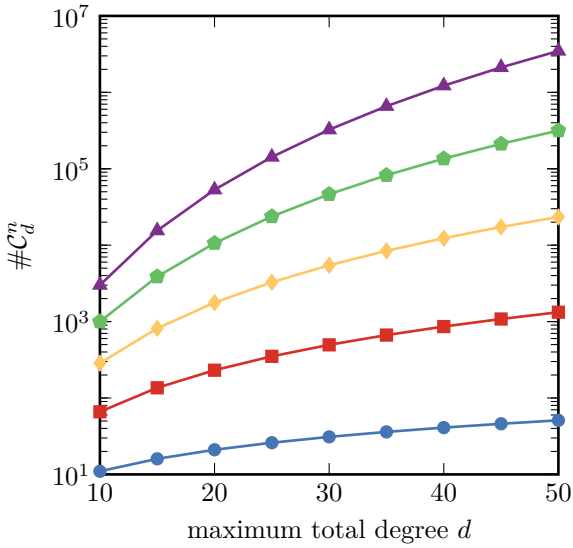


Figure 2.1. Combinatorial explosion of the number of monomials in the set $\mathcal{C}_d^n$ with respect to the maximum total degree d and number of variables n. The cardinality of $\mathcal{C}_d^n$ is given for $n = 1$ (—●—), $n = 2$ (—■—), $n = 3$ (—◆—), $n = 4$ (—●—), and $n = 5$ (—▲—).

```
>> nbmonomials(40,5)

ans =
    1221759
```

With respect to a basis, the coefficients of a multivariate polynomial $p(\boldsymbol{x})$ are the components of $p(\boldsymbol{x})$ as an element of the vector space $\mathcal{P}_d^n$. Of course, it is also possible to use another polynomial basis than the standard monomial basis for $\mathcal{P}_d^n$, e.g., the basis given by the multivariate Chebyshev or Legendre polynomials (Section 2.6.3). A polynomial can be written as its coefficient vector $\boldsymbol{p}$ multiplied by a vector $\boldsymbol{v}$ that contains all the basis polynomials. For example, the univariate polynomial

$$p(x) = 3 + 2x + x^2 = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & x & x^2 \end{bmatrix}^{\mathrm{T}}. \tag{2.16}$$

This univariate polynomial then corresponds to $p(x) = \boldsymbol{p}^{\mathrm{T}}\boldsymbol{v}$. For multivariate polynomials, something similar exists:

$$p(\boldsymbol{x}) = p(x_1, x_2) = 6 + 5x_1 + 4x_2 + 3x_1^2 + 2x_1x_2 + x_2^2 \tag{2.17}$$

can be represented by the vector

$$\boldsymbol{p} = \begin{bmatrix} 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix}^{\mathrm{T}}, \tag{2.18}$$

where the vector of monomials is

$$\boldsymbol{v} = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 \end{bmatrix}^{\mathrm{T}}. \tag{2.19}$$

It is possible to order the terms of multivariate polynomials in different ways, and (algebraic) results, like the Gröbner basis, often depend on that monomial ordering[5]. In order to have an unambiguous notation, this representation requires a consensus about the ordering of the monomials.

Definition 2.5. Given a monomial ordering, a multivariate polynomial $p(\boldsymbol{x}) \in \mathcal{P}_d^n$,

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} c_i \boldsymbol{x}^{\alpha_i}, \tag{2.20}$$

can be represented unambiguously by its **vector representation** $\boldsymbol{p}$, which is a vector that contains its coefficients ordered according to the selected monomial ordering, i.e.,

$$\boldsymbol{p} = [\![p(\boldsymbol{x})]\!]^{\mathrm{T}} = \begin{bmatrix} c_1 & c_2 & \cdots & c_{k-1} & c_k \end{bmatrix}^{\mathrm{T}}, \tag{2.21}$$

where c_i is the ith coefficient $c_{\boldsymbol{\alpha}}$ of $p(\boldsymbol{x})$ in the selected monomial ordering ($i = 1, \ldots, k$). The operator $[\![\cdot]\!]$ denotes the arrangement of the coefficients $c_{\boldsymbol{\alpha}}$ (not the associated monomials) of $p(\boldsymbol{x})$ as a row vector.

The monomial ordering used throughout this text is the graded inverse lexicographic (GRINVLEX) ordering, which is sometimes also known as the *graded xel ordering* or *degree negative lexicographic ordering* [27, 65]. It is graded because it first compares the degrees of the two monomials/multi-indices and applies the inverse/negative lexicographic ordering when there is a tie. Although we mainly use the GRINVLEX ordering in this text, the remainder remains valid for any graded monomial ordering[6].

Definition 2.6. When considering two n-tuples $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}^n$ and $|\boldsymbol{\alpha}| > |\boldsymbol{\beta}|$ or $|\boldsymbol{\alpha}| = |\boldsymbol{\beta}|$ where in the element-wise difference $\boldsymbol{\alpha} - \boldsymbol{\beta} \in \mathbb{Z}^n$ the left-most non-zero element of the tuple is negative, two monomials are ordered $\boldsymbol{x}^{\boldsymbol{\alpha}} > \boldsymbol{x}^{\boldsymbol{\beta}}$ by the **graded inverse lexicographic (GRINVLEX) ordering**.

Example 2.4. The GRINVLEX ordering orders the monomials in $n = 3$ variables as

$$1 < x_1 < x_2 < x_3 < x_1^2 < x_1x_2 < x_1x_3 < x_2^2 < x_2x_3 < x_3^2 < x_1^3 < \ldots \tag{2.22}$$

[5] It is well-known that a Gröbner basis with respect to the lexicographic monomial ordering is typically more complex than with respect to the reverse lexicographic ordering [90].

[6] For a more detailed overview of relevant monomial orderings in algebraic geometry, we refer the interested reader to [65, 66].

2.2.2 Affine varieties and ideals

The two main objects of algebraic geometry are affine varieties and ideals. However, before we consider the definition of an affine variety, we must specify the affine space.

Definition 2.7. Given the field of complex numbers $\mathbb{C}$ and a positive integer n, the n-dimensional **affine space** over $\mathbb{C}$ is the set

$$\mathbb{C}^n = \{\boldsymbol{a} = (a_1, \ldots, a_n) : a_1, \ldots, a_n \in \mathbb{C}\}. \tag{2.23}$$

As a set, the n-dimensional space $\mathbb{C}^n$ consists of all n-tuples of complex numbers. For low n, these affine spaces look very familiar: $n = 1$ is the affine line and $n = 2$ is the affine plane. Affine varieties are points, curves, and surfaces (and higher-dimensional objects) defined by the (common) roots of (multivariate) polynomials. An affine variety is a subspace of the affine space[7], namely the subspace defined by zero sets of polynomials, as explained below.

Definition 2.8. If $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ are polynomials in $\mathcal{P}^n$, then it is possible to define the set

$$\mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})) = \{\boldsymbol{a} \in \mathbb{C}^n : p_i(\boldsymbol{a}) = 0 \text{ for all } 1 \leq i \leq s\}. \tag{2.24}$$

The set $\mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}))$ is called the **affine variety** defined by the polynomials $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$.

Thus, an affine variety is the set of all affine solutions of a given system of multivariate polynomial equations.

Example 2.5. Two trivial affine varieties are $\mathcal{V}(\boldsymbol{0}) = \mathbb{C}^n$ and $\mathcal{V}(1) = \emptyset$. Every point $\boldsymbol{a} = (a_1, \ldots, a_n)$ is also an affine variety: $\mathcal{V}(x_1 - a_1, \ldots, x_n - a_n) = \boldsymbol{a}$.

Affine varieties have many interesting properties, for which we refer the reader to [65, 66]. When the affine variety is zero-dimensional, it consists of isolated points. In this case, the different affine solutions are written as

$$\boldsymbol{x}|_{(j)} = \boldsymbol{a}_j \in \mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})) \subset \mathbb{C}^n, \quad j = 1, \ldots, m_a, \tag{2.25}$$

which means that the variables are evaluated in each of the affine solutions (with some arbitrary order assumed on the affine solutions).

Another important object in algebraic geometry is the ideal. Where varieties are geometric objects, ideals are their algebraic counterparts (there exist a lot of equivalent properties between them, see [65]). The importance of ideals lies in the fact that they allow computing with varieties.

[7] The affine complex space is in this setting also often called the ambient space of the affine variety, sometimes denoted explicitly as $\mathcal{V}_{\mathbb{C}^n}$.

Definition 2.9. A subset $\mathcal{I} \subseteq \mathcal{P}^n$ is an **ideal** if it satisfies the following three requirements:

1. $0 \in \mathcal{I}$ (with 0 the zero polynomial).
2. If $p(\boldsymbol{x}), q(\boldsymbol{x}) \in \mathcal{I}$, then $(p+q)(\boldsymbol{x}) \in \mathcal{I}$.
3. If $p(\boldsymbol{x}) \in \mathcal{I}$ and $r(\boldsymbol{x}) \in \mathcal{P}^n$, then $(rp)(\boldsymbol{x}) \in \mathcal{I}$.

Lemma 2.1. Consider the polynomials $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \in \mathcal{P}^n$. The set

$$\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle = \left\{ \sum_{i=1}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x}) : h_1(\boldsymbol{x}), \ldots, h_s(\boldsymbol{x}) \in \mathcal{P}^n \right\} \tag{2.26}$$

is an ideal.

Proof. In the proof of this lemma, we check each of the requirements of Definition 2.9:

1. $0 \in \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle$, because we can set $h_i(\boldsymbol{x}) = 0$ for all i.
2. For $p(\boldsymbol{x}) = \sum_{i=1}^{s} f_i(\boldsymbol{x}) p_i(\boldsymbol{x})$ and $q(\boldsymbol{x}) = \sum_{i=1}^{s} g_i(\boldsymbol{x}) p_i(\boldsymbol{x})$, it holds that $(p+q)(\boldsymbol{x}) = \sum_{i=1}^{s} (f_i(\boldsymbol{x}) + g_i(\boldsymbol{x})) p_i(\boldsymbol{x}) = \sum_{i=1}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x}) \in \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle$.
3. For $p(\boldsymbol{x}) = \sum_{i=1}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x})$ and $r(\boldsymbol{x}) \in \mathcal{P}^n$, it holds that $(rp)(\boldsymbol{x}) = \sum_{i=1}^{s} (r(\boldsymbol{x}) h_i(\boldsymbol{x})) p_i(\boldsymbol{x}) = \sum_{i=1}^{s} \widetilde{h}_i(\boldsymbol{x}) p_i(\boldsymbol{x}) \in \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle$.

This proofs that $\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle$ is an ideal. □

Example 2.6. Consider the multivariate polynomials

$$p_1(\boldsymbol{x}) = x_1 - x_2^2 \quad \text{and} \quad p_2(\boldsymbol{x}) = x_1 x_2. \tag{2.27}$$

The set

$$\langle p_1(\boldsymbol{x}), p_2(\boldsymbol{x}) \rangle = \left\{ \sum_{i=1}^{2} h_i(\boldsymbol{x}) p_i(\boldsymbol{x}) : h_1(\boldsymbol{x}), h_2(\boldsymbol{x}) \in \mathcal{P}^2 \right\} \tag{2.28}$$

is an ideal. It is possible to show that

$$p_3(\boldsymbol{x}) = x_1^2 \tag{2.29}$$

is an element of this ideal, because

$$x_1^2 = x_1 p_1(\boldsymbol{x}) + x_2 p_2(\boldsymbol{x}) = x_1 \big(x_1 - x_2^2\big) + x_2 (x_1 x_2). \tag{2.30}$$

The set $\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle$ denotes the ideal generated by $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ and, therefore, the polynomials $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ are often called the **generators** of the ideal. Every other polynomial in this ideal can be constructed from these polynomials: they are polynomial **consequences**. We say that an ideal $\mathcal{I}$ is finitely generated if there exists a finite number of polynomials $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ so that $\mathcal{I} = \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle$ and say that $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ form a basis of the ideal. There is an amazing fact, also known as Hilbert's basis theorem, that states that every ideal of $\mathcal{P}^n$ is finitely generated[8].

Theorem 2.1 (Hilbert's basis theorem). Every ideal $\mathcal{I} \subset \mathcal{P}^n$ has a finite generating set. That is, $\mathcal{I} = \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle$ for some $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \in \mathcal{I}$.

Proof. A proof of this theorem can be found in [65, p. 77]. □

Note that an ideal may have many different polynomial bases, for example, Gröbner bases obtained via different monomial orderings.

Example 2.7. The ideal $\mathcal{I} = \left\langle 2x_1^2 + 3x_2^2 - 11, x_1^2 - x_2^2 - 3\right\rangle$ is constructed from the polynomials $p_1(\boldsymbol{x}) = 2x_1^2 + 3x_2^2 - 11$ and $p_2(\boldsymbol{x}) = x_1^2 - x_2^2 - 3$. One can show that the polynomials $q_1(\boldsymbol{x}) = x_1^2 - 4$ and $q_2(\boldsymbol{x}) = x_2^2 - 1$ generate the same ideal. Both sets of polynomials are polynomial bases of that $\mathcal{I}$.

Ideals are closely related to affine varieties (cf., the geometry-algebra equivalence), as the following proposition shows.

Proposition 2.2. If $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ and $q_1(\boldsymbol{x}), \ldots, q_t(\boldsymbol{x})$ are bases of the same ideal in $\mathcal{P}^n$, so that $\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle = \langle q_1(\boldsymbol{x}), \ldots, q_t(\boldsymbol{x})\rangle$, then holds that $\mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})) = \mathcal{V}(q_1(\boldsymbol{x}), \ldots, q_t(\boldsymbol{x}))$.

Proof. Every polynomial $q_j(\boldsymbol{x})$, $j = 1, \ldots, t$, can be written as a linear combination $\sum_{i=1}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x})$, since it is a polynomial in the ideal $\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle$. $\forall \boldsymbol{a} \in \mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}))$, $q_j(\boldsymbol{a}) = \sum_{i=1}^{s} h_i(\boldsymbol{a}) p_i(\boldsymbol{a}) = 0$. Hence, $\boldsymbol{a} \in \mathcal{V}(q_1(\boldsymbol{x}), \ldots, q_t(\boldsymbol{x}))$, which implies that $\mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})) \subseteq \mathcal{V}(q_1(\boldsymbol{x}), \ldots, q_t(\boldsymbol{x}))$. Equality follows from repeating the same argument in the other direction. □

Proposition 2.2 shows that affine varieties are actually determined by ideals, not only polynomials. We can even go one step further and define a special ideal.

[8] As discussed in [66], Hilbert's basis theorem expands beyond the ring of complex polynomials. We restrict the definition in this text to $\mathcal{P}^n$ for readability. Note that a polynomial basis does have a fixed minimum number of elements, like in linear algebra. Two different minimal polynomial bases can have a different number of basis elements [230].

Definition 2.10. Let $\mathcal{V} \subset \mathbb{C}^n$ be an affine variety and define the set

$$\mathcal{I}(\mathcal{V}) = \{p(\boldsymbol{x}) \in \mathcal{P}^n : p(\boldsymbol{a}) = 0 \text{ for all } \boldsymbol{a} \in \mathcal{V}\}. \tag{2.31}$$

Lemma 2.2. If $\mathcal{V} \in \mathbb{C}^n$ is an affine variety, then $\mathcal{I}(\mathcal{V})$ is an ideal.

Proof. A proof of this lemma can be found in [65, p. 32]. □

So, we go from polynomials, which form an ideal, to an affine variety, which defines again an ideal[9]. The goal in this text is to identify the affine solutions of systems of multivariate polynomial equations (in the specific situation when the affine solution set is zero-dimensional). Since the solutions of the entire ideal generated by the polynomials of this system are the same as the solutions of these polynomials (Proposition 2.2), we can rephrase our objective as identifying the affine variety of the ideal generated by the given polynomials[10].

2.2.3 Number of solutions

An important question immediately rises: "How many solutions does the system of multivariate polynomial equations have?" For a univariate equation, the fundamental theorem of algebra answers this question (Theorem A.1). When the number of variables $n > 1$, more advanced results come into the picture. The multivariate extension of the fundamental theorem of algebra is Bézout's theorem.

Theorem 2.2 (Bézout's theorem). For any square system (i.e., $s = n$) of multivariate polynomial equations $p_1(\boldsymbol{x}) = \cdots = p_n(\boldsymbol{x}) = 0$, the number of isolated solutions when the solution set is zero-dimensional, i.e., the number of isolated points in the zero-dimensional affine variety $\mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_n(\boldsymbol{x})) \subset \mathbb{C}^n$, is at most

$$m_b = d_1 \cdots d_n = \prod_{i=1}^{n} d_i, \tag{2.32}$$

[9] Of course, this ideal again defines a new affine variety. Immediately a new question arises: "Is this new affine variety the same as the initial affine variety?" The answer, however, is not always yes and relies on *Hilbert's Nullstellensatz* [65, 66].

[10] For any ideal $\mathcal{I} \subset \mathcal{P}^n$ generated by polynomials $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$, we can consider the affine variety $\mathcal{V}(\mathcal{I})$. However, some information can get lost in making this association. Two different ideals $\mathcal{I}_1 \neq \mathcal{I}_2 \subset \mathcal{P}^n$ may have the same affine variety, $\mathcal{V}(\mathcal{I}_1) = \mathcal{V}(\mathcal{I}_2)$, but a different geometric behavior. There can be points with a multiplicity greater than one, multiple curves, embedded points, etc. An extension of the affine variety that deals with these issues is the *affine scheme*, of which the affine variety can be seen as a subcategory [242]. We ignore affine schemes in this dissertation and only consider zero-dimensional affine varieties together with the multiplicity structure of each isolated solution. The interested reader can find more about affine schemes in [82].

where d_i is the total degree of $p_i(\boldsymbol{x})$. Moreover, there exists a subvariety $\nabla_{d_1,\ldots,d_n} = \mathcal{V}(\Delta_{d_1},\ldots,\Delta_{d_n}) \subsetneq \mathbb{C}^n$ such that, when the coefficients of the square system $p_1(\boldsymbol{x}),\ldots,p_n(\boldsymbol{x}) \notin \nabla_{d_1,\ldots,d_n}$, the variety $\mathcal{V}(p_1(\boldsymbol{x}),\ldots,p_n(\boldsymbol{x}))$ consists of precisely $m_a = m_b$ different affine points.

Proof. A proof of this theorem can be found in [82, Theorem III-71]. □

The subvariety $\nabla_{d_1,\ldots,d_n}$ can be described by using resultants and discriminants. It describes the polynomial systems for which the number of points in the zero-dimensional affine variety is not equal to m_b.

Example 2.8. For the trivial case of one univariate quadratic polynomial $p(x) = c_0 + c_1 x + c_2 x^2$, the subvariety ∇_2 is given by

$$\nabla_2 = \mathcal{V}\big(c_2\big(c_1^2 - 4c_0c_2\big)\big), \tag{2.33}$$

where we recognize the discriminant for a univariate quadratic polynomial. This means that polynomials for which $c_2\big(c_1^2 - 4c_0c_2\big) \neq 0$ are not inside the subvariety ∇_2 and have exactly $m_a = d = 2$ different affine roots.

Theorem 2.2 is an important result and provides an easy way to bound the number of isolated affine solutions of a square system of n multivariate polynomial equations in n variables. The difference between the actual number of affine solutions m_a and the Bézout number m_b can be considered as the number of solutions at infinity m_∞:

$$m_b = m_a + m_\infty. \tag{2.34}$$

The bound in Theorem 2.2 is almost always tight, in the sense that the only polynomial systems with fewer affine solutions lie in that particular subvariety. Unfortunately, applications often lead to systems in that subvariety, due to their structure [241]. This has led to more advanced bounds on the number of isolated affine solutions, like Kushnirenko's bound and the Bernstein–Khovanskii–Kushnirenko theorem (Appendix A.3). In practice, we could try to solve the polynomial equations (for example, with the algorithms presented in this dissertation) and count the number of obtained affine solutions.

Remark 2.1. In a sense, Bézout's theorem more naturally counts solutions in the projective space $\mathbb{P}^n$, where the number of solutions (counted with multiplicity) for a zero-dimensional variety is always equal to m_b. The projective space and projective varieties are defined in Section 4.2. A formulation of Bézout's theorem in the projective space is given in Appendix A.3.

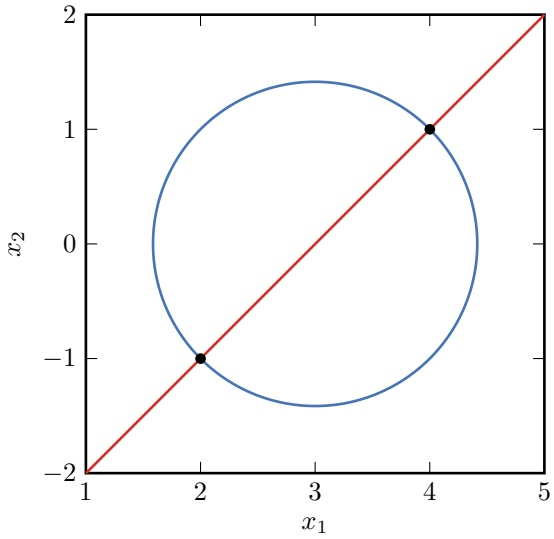


Figure 2.2. Real picture of the system of multivariate polynomial equations in Example 2.9: $p_1(\boldsymbol{x}) = x_1^2 + x_2^2 - 6x_1 + 7 = 0$ (——) and $p_2(\boldsymbol{x}) = x_1 - x_2 - 3 = 0$ (——). These two polynomial equations have 2 affine solutions (•).

Example 2.9. Consider the intersection of a circle with a line (Figure 2.2),

$$\begin{cases} p_1(\boldsymbol{x}) = x_1^2 + x_2^2 - 6x_1 + 7 = 0, \\ p_2(\boldsymbol{x}) = x_1 - x_2 - 3 = 0. \end{cases} \tag{2.35}$$

The Bézout number for this polynomial system equals $m_b = 2 \cdot 1 = 2$, which agrees with the fact that the multivariate polynomials have two common roots $(2, -1)$ and $(4, 1)$.

Code 2.2. It is possible to compute the Bézout number for a `system` via `bezout(system)`. The polynomial system `toy1` is part of MacaulayLab's database.

```
>> mb = bezout(toy1)

mb =
     2
```

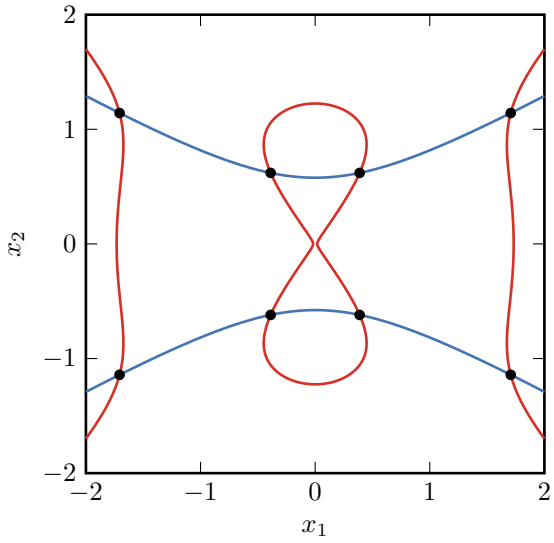


Figure 2.3. Real picture of the system of multivariate polynomial equations in Example 2.10: $p_1(\boldsymbol{x}) = x_1^2 - 3x_2^2 + 1 = 0$ (——) and $p_2(\boldsymbol{x}) = 3x_1^2 - 1.5x_2^2 - x_1^4 + x_2^4 = 0$ (——). These two polynomial equations have 8 affine solutions (•).

Example 2.10. Consider the variety $\mathcal{V}(p_1(\boldsymbol{x}), p_2(\boldsymbol{x}))$ defined by the following system of multivariate polynomials equations:

$$\begin{cases} p_1(\boldsymbol{x}) = x_1^2 - 3x_2^2 + 1 = 0, \\ p_2(\boldsymbol{x}) = 3x_1^2 - 1.5x_2^2 - x_1^4 + x_2^4 = 0. \end{cases} \tag{2.36}$$

The variety of this polynomial system is zero-dimensional and contains 8 affine points in $\mathbb{C}^2$ and no points at infinity (Figure 2.3), in accordance with Bézout's theorem:

$$m_b = d_1 \cdot d_2 = 2 \cdot 4 = 8. \tag{2.37}$$

Example 2.11. A system of multivariate polynomial equations that lies in the subvariety $\nabla_{2,2}$ is

$$\begin{cases} p_1(\boldsymbol{x}) = x_1^2 + x_1 x_2 - 2 = 0, \\ p_2(\boldsymbol{x}) = x_2^2 + x_1 x_2 - 2 = 0. \end{cases} \tag{2.38}$$

The Bézout number of this polynomial system is $m_b = 2^2 = 4$, but it only has $m_a = 2$ affine solutions (Figure 2.4). The two curves defined by (2.38) have an asymptote and seem to intersect at infinity. In the projective space $\mathbb{P}^2$, these two intersection points, given by $(0, 1, -1)$ and $(0, -1, 1)$, are also solutions of the polynomial system.

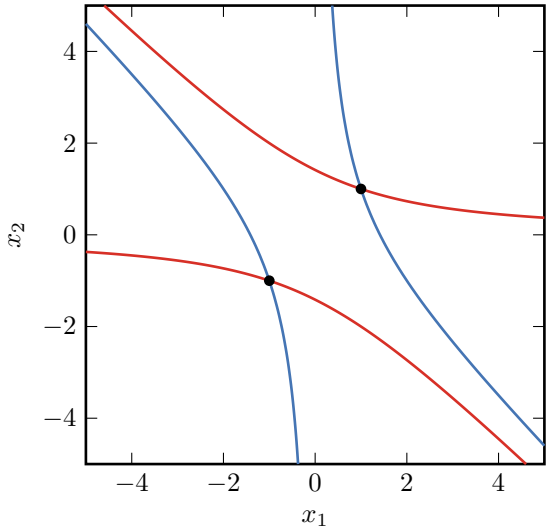


Figure 2.4. Real picture of the system of multivariate polynomial equations in Example 2.11: $p_1(\boldsymbol{x}) = x_1^2 + x_1x_2 - 2 = 0$ (—) and $p_2(\boldsymbol{x}) = x_2^2 + x_1x_2 - 2 = 0$ (—). These two polynomial equations have 2 affine solutions (•).

2.3 Macaulay matrix and its subspaces

The polynomials $p_i(\boldsymbol{x}) \in \mathcal{P}^n$, for $i = 1, \ldots, s$, of a system of multivariate polynomial equations in n variables $\boldsymbol{x} \in \mathbb{C}^n$ constitute the so-called **seed equations** of the corresponding Macaulay matrix. The Macaulay matrix is generated by these seed equations via a forward shift recursion (FSR): multiplying the seed equations (i.e., the generating polynomial equations) with different monomials $\{\boldsymbol{x}^{\boldsymbol{\alpha}_i}\}$ of increasing total degree d_{r_i} leads to "new" polynomial equations, the coefficients of which are organized as the rows of the Macaulay matrix[11].

Example 2.12. To introduce the Macaulay matrix, consider the polynomial system (2.35) from Example 2.9, where $p_1(\boldsymbol{x}) = 0$ and $p_2(\boldsymbol{x}) = 0$ constitute the seed equations. Multiplying $p_1(\boldsymbol{x}) = 0$ by x_1 and x_2 (i.e., $\{\boldsymbol{x}^{\boldsymbol{\alpha}_1}\} = \{x_1, x_2\}$), leads to two "new" polynomial equations:

$$\begin{aligned} x_1\big(x_1^2 + x_2^2 - 6x_1 + 7\big) &= 0, \\ x_2\big(x_1^2 + x_2^2 - 6x_1 + 7\big) &= 0. \end{aligned} \tag{2.39}$$

We apply the same procedure for $p_2(\boldsymbol{x}) = 0$ with different monomials $\{\boldsymbol{x}^{\boldsymbol{\alpha}_2}\} = \{x_1, x_2, x_1^2, x_1x_2, x_2^2\}$ to obtain all polynomial equations up to total

[11] This particular type of FSR is more precisely termed *scalar forward multi-shift recursion*. Next to scalar forward multi-shift recursion, there also exist scalar forward single-shift recursion, block forward single-shift recursion, and block forward multi-shift recursion (Section 1.2). Chapter 3 considers the last type of FSR, when extending the Macaulay matrix to the block Macaulay matrix.

degree $d = 3$. When we structure the coefficients of these polynomial equations in a matrix according to a certain monomial ordering, we obtain the Macaulay matrix of degree 3 (seed equations in red):

$$\begin{array}{c} \\ 1 \\ x_1 \\ x_2 \\ 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{array}\!\!\begin{array}{c} \begin{array}{cccccccccc} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \end{array} \\ \left[\begin{array}{cccccc|cccc} 7 & -6 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & -6 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 7 & 0 & -6 & 0 & 0 & 1 & 0 & 1 \\ -3 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -3 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -3 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 1 & -1 \end{array}\right] \end{array}. \tag{2.40}$$

Of course, we can further enlarge the Macaulay matrix by considering monomials $\{\boldsymbol{x}^{\boldsymbol{\alpha}_1}\}$ and $\{\boldsymbol{x}^{\boldsymbol{\alpha}_2}\}$ of higher total degrees d_{r_1} and d_{r_2}, respectively. We visualize the result for degree $d = 5$ in Figure 2.5.

Code 2.3. A Macaulay matrix of degree d can easily be constructed via `macaulay(system,d)`.

```
>> M = macaulay(toy1,5);
```

Definition 2.11. Consider the system of multivariate polynomial equations with polynomials $p_i(\boldsymbol{x}) \in \mathcal{P}^n$, for $i = 1, \ldots, s$, which serve as the seed equations. Let the total degree of every polynomial $p_i(\boldsymbol{x})$ be denoted by d_i. The **Macaulay matrix** of degree d, $\boldsymbol{M}_d \in \mathbb{C}^{p_d \times q_d}$, contains the coefficients of the seed equations and the equations generated by the FSR with monomials of increasing total degrees $d_{\mathrm{r}_i} = 1, \ldots, (d - d_i)$, i.e.,

$$\boldsymbol{M}_d = \left[\!\!\left[\begin{array}{c} \{\boldsymbol{x}^{\boldsymbol{\alpha}_1}\} p_1(\boldsymbol{x}) \\ \vdots \\ \{\boldsymbol{x}^{\boldsymbol{\alpha}_s}\} p_s(\boldsymbol{x}) \end{array} \right]\!\!\right], \tag{2.41}$$

where $[\![\cdot]\!]$ is the (stacked) vector representation of the polynomials (Definition 2.5).

These shifted coefficients are indexed both in row direction (different monomials of FSR) and column direction (different associated monomials) by the different monomials of total degree at most d . The number of rows p_d and columns q_d

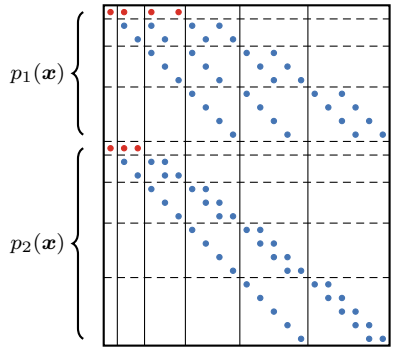


Figure 2.5. Spy plot of the Macaulay matrix of degree 5 for the system of multivariate polynomial equations in Example 2.9. The polynomial $p_1(\boldsymbol{x})$ is multiplied by all monomials of total degree up to $d_{\mathrm{r}_1} = 3$ and the polynomial $p_2(\boldsymbol{x})$ is multiplied by all monomials of total degree up to $d_{\mathrm{r}_2} = 4$. The red dots (●) indicate the coefficients of the seed equations, while the blue dots (●) correspond with the shifted coefficients. The elements not shown are zero. Vertical lines indicate the different degree blocks of the Macaulay matrix, while horizontal dashed lines separate the monomials of different total degrees d_{r_i}, $i = 1, \ldots, s$ in the FSR.

of $\boldsymbol{M}_d$ are given by

$$p_d = \sum_{i=1}^{s} \binom{d - d_i + n}{n} = \sum_{i=1}^{s} \frac{(d - d_i + n)!}{n!(d - d_i)!} \tag{2.42}$$

and

$$q_d = \binom{d+n}{n} = \frac{(d+n)!}{n!d!}. \tag{2.43}$$

The actual structure of the Macaulay matrix depends on its monomial ordering (an example is given in Section 2.6.3).

The system of multivariate polynomial equations and the "new" polynomial equations obtained via the FSR can be written as the matrix-vector product of the generated Macaulay matrix $\boldsymbol{M}_d \in \mathbb{C}^{p_d \times q_d}$ and a structured vector $\boldsymbol{v}_d \in \mathbb{C}^{q_d \times 1}$:

$$\boldsymbol{M}_d \underbrace{\begin{bmatrix} 1 \\ \hline x_1 \\ \vdots \\ x_n \\ \hline \vdots \\ \hline x_1^d \\ \vdots \\ x_n^d \end{bmatrix}}_{\boldsymbol{v}(d)} = \boldsymbol{0}. \tag{2.44}$$

Algorithm 2.1 Iterative root-finding via the Macaulay matrix

Require: $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$
$d \leftarrow d_{\max} = \max(d_1, \ldots, d_s)$
Construct the Macaulay matrix of degree d
while $d < d_\circ$ **do**
    Check the structure of null space or column space
    if it is possible to find the affine common roots (Section 2.4.3) **then**
        $d = d_\circ$
    else
        $d \leftarrow d + 1$
        Construct the Macaulay matrix of degree d
    end if
end while
Find the affine common roots $\boldsymbol{x}|_{(j)}$, for $j = 1, \ldots, m_a$, in the null space (Algorithm 2.2) or in the column space (Algorithm 2.3)
return $\boldsymbol{x}|_{(j)}$, for $j = 1, \ldots, m_a$

We increase the degree d of $\boldsymbol{M}_d$ until it is *large enough* and reaches the **solution degree** $d_\circ$, a notion on which we elaborate in Section 2.4.3. The vector $\boldsymbol{v}_d$ is a vector in the right null space of $\boldsymbol{M}_d$ and has a special multivariate Vandermonde structure, which is *enforced* by the FSR that generates the rows of $\boldsymbol{M}_d$.

In the structure of both the right null space and the column space lies the key to solving the generating system of multivariate polynomial equations. Note that also the other two fundamental subspaces of the Macaulay matrix (i.e., the row space and left null space) have an interpretation in terms of the algebraic properties of the generating polynomial equations. Chapter 4 dives deeper into the algebraic geometry interpretations of all four fundamental subspaces of the Macaulay matrix, while we focus in this chapter solely on multivariate polynomial system solving; hence, we only consider the (right[12]) null space (Section 2.4) and column space (Section 2.5) to develop two complementary multivariate root-finding algorithms. The skeleton of both multivariate root-finding algorithms is very similar (Algorithm 2.1): the Macaulay matrix is iteratively enlarged until the structure of the null space or column space yields the affine common roots of the polynomials.

Remark 2.2. Note that horizontal and vertical lines separate degree blocks in matrices and vectors (e.g., in (2.44) or in Figure 2.5). A **degree block** contains all the rows/columns that correspond to monomials of the same total degree (e.g., the rows that correspond to x_1^2, x_1x_2, and x_2^2).

Remark 2.3. To alleviate the notational complexity, we no longer specify the degree d explicitly in the remainder of this chapter (unless when necessary), but we assume it to be *large enough*, i.e., $d \geq d_\circ$ (Section 2.4.3).

[12]In the remainder of this chapter, we no longer mention the qualification *right* explicitly. We always consider the right null space, except when denoted otherwise.

2.4 Null space based approach

We now exploit the structure of the null space of the Macaulay matrix in order to find the solutions of its seed equations, i.e., the system of multivariate polynomial equations. Firstly, we consider only polynomial equations with simple, affine, and isolated solutions, which allows us to show that a multidimensional realization problem in the structured null space yields the exact affine solutions (Section 2.4.1). Secondly, we study the influence of solutions with multiplicity greater than one (Section 2.4.2). Thirdly, we explain the notion of a *large enough* degree (i.e., the solution degree $d_\circ$) and show how to deflate the solutions at infinity (Section 2.4.3). Finally, we summarize the entire null space based root-finding algorithm (Section 2.4.4).

2.4.1 Multidimensional realization theory

We start our explanation with the multivariate Vandermonde basis matrix of the null space of the Macaulay matrix where we assume that we know all the solutions (Section 2.4.1.1). Afterwards, we generalize this procedure to any (numerical) basis matrix of the null space (Section 2.4.1.2).

2.4.1.1 Multivariate Vandermonde basis matrix

We consider, for didactic purposes, a system of multivariate polynomial equations that only has m_a simple (i.e., algebraic multiplicity is one), affine (i.e., non-infinite), and isolated solutions (i.e., the solution set is zero-dimensional). When recursively increasing the degree d of the Macaulay matrix $\boldsymbol{M}$, the nullity (i.e., the dimension of the null space) grows, until it stabilizes at the Bézout number m_b ($= m_a$, in this case). For a Macaulay matrix of degree $d \geq d_\circ$ (Section 2.4.3), there exists a multivariate Vandermonde vector $\boldsymbol{v}|_{(j)}$ ($j = 1, \ldots, m_a$) in the null space of $\boldsymbol{M}$ for each solution of the system and, together, these basis vectors span the entire null space of $\boldsymbol{M}$. They naturally form the multivariate Vandermonde basis matrix $\boldsymbol{V} \in \mathbb{C}^{q \times m_a}$ of degree $d \geq d_\circ$ (same degree as $\boldsymbol{M}$):

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}|_{(1)} & \cdots & \boldsymbol{v}|_{(m_a)} \end{bmatrix} = \left[\begin{array}{ccc} 1 & \cdots & 1 \\ \hline x_1|_{(1)} & \cdots & x_1|_{(m_a)} \\ \vdots & & \vdots \\ x_n|_{(1)} & \cdots & x_n|_{(m_a)} \\ \hline x_1^2|_{(1)} & \cdots & x_n^2|_{(m_a)} \\ \vdots & & \vdots \end{array}\right]. \tag{2.45}$$

The structured $\boldsymbol{V}$ does suspect that the (affine) null space of the Macaulay matrix has a "special shift structure". Mathematically, this "special shift structure" can be written as (when we shift some rows with the first variable x_1)

$$\underbrace{\boldsymbol{S}_1 \boldsymbol{V}}_{\text{before shift}} \boldsymbol{D}_{x_1} = \underbrace{\boldsymbol{S}_{x_1} \boldsymbol{V}}_{\text{after shift}}, \tag{2.46}$$

where the diagonal matrix $\boldsymbol{D}_{x_1} \in \mathbb{C}^{m_a \times m_a}$ contains the different solutions for the variable x_1 and the row selection matrices $\boldsymbol{S}_1 \in \mathbb{R}^{m_a \times q}$ and $\boldsymbol{S}_{x_1} \in \mathbb{R}^{m_a \times q}$ select the rows before and after the shift, respectively. In order for this expression to cover all the affine solutions, the row selection matrix $\boldsymbol{S}_1$ has to select m_a linearly independent rows from $\boldsymbol{V}$ (then $\boldsymbol{S}_1\boldsymbol{V}$ is square and nonsingular). Actually, from algebraic geometry, it follows that these linearly independent rows correspond to the (affine) standard monomials [21, 66, 78]. We say that the rows in $\boldsymbol{S}_{x_1}\boldsymbol{V}$ are *hit* by the shift with x_1. Notice that the construction of $\boldsymbol{S}_{x_1}$ depends on the chosen $\boldsymbol{S}_1$. The next example illustrates this interesting property.

Example 2.13. Consider again system (2.35), which has only $m_a = 2$ simple, affine, and isolated solutions $(2, -1)$ and $(4, 1)$. The multivariate Vandermonde basis matrix $\boldsymbol{V}$ of degree $d = 2$ is equal to (we can construct it from the known solutions)

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}|_{(1)} & \boldsymbol{v}|_{(2)} \end{bmatrix} = \left[\begin{array}{cc} 1 & 1 \\ \hline 2 & 4 \\ -1 & 1 \\ \hline 4 & 16 \\ -2 & 4 \\ 1 & 1 \end{array}\right]. \tag{2.47}$$

When we take a vector from $\boldsymbol{V}$, i.e., $\boldsymbol{v}|_{(j)}$, and multiply the first two elements by $x_1|_{(j)}$, the elements obtained after the multiplication are again part of that vector:

$$\begin{bmatrix} 1 \\ x_1 \end{bmatrix}_{(j)} \xrightarrow{x_1|_{(j)}} \begin{bmatrix} x_1 \\ x_1^2 \end{bmatrix}_{(j)}.$$

We can also write this multiplication, by means of two row selection matrices $\boldsymbol{S}_1$ and $\boldsymbol{S}_{x_1}$, as

$$\left(\boldsymbol{S}_1\, \boldsymbol{v}|_{(j)}\right) x_1|_{(j)} = \left(\boldsymbol{S}_{x_1}\, \boldsymbol{v}|_{(j)}\right), \tag{2.48}$$

with

$$\boldsymbol{S}_1 = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{2.49}$$

and

$$\boldsymbol{S}_{x_1} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}. \tag{2.50}$$

Or, numerically, for all solutions of (2.35) at once as

$$\underbrace{\begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}}_{\boldsymbol{S}_1\boldsymbol{V}} \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}}_{\boldsymbol{D}_{x_1}} = \underbrace{\begin{bmatrix} 2 & 4 \\ 4 & 16 \end{bmatrix}}_{\boldsymbol{S}_{x_1}\boldsymbol{V}}. \tag{2.51}$$

Code 2.4. We need to construct the `vandermonde(d,n,X)` basis matrix from the solutions `X = [2 -1; 4 1]` before we can shift.

```
>> V = vandermonde(2,2,[2 -1; 4 1]); % d = n = 2
```

We can construct shift matrices via `shiftmatrix(d,n,rows,shift)`, where `shift` determines the specific shift operation. If we shift the first two rows, then we need to use `rows = [1 2]`.

```
>> rows = [1 2];
>> S1 = shiftmatrix(2,2,rows,[1 0 0]); % shift with 1
>> Sx1 = shiftmatrix(2,2,rows,[1 1 0]); % shift with 1*x1
```

The "special shift structure" from (2.51) can then be retrieved:

```
>> (S1*V)*diag([2 4])

ans =
    2     4
    4     16

>> Sx1*V =

ans =
    2     4
    4     16
```

This "special shift structure" does not restrict itself to the variable x_1, but applies to all variables. It even holds for a shift polynomial $g(\boldsymbol{x})$ in the variables $\boldsymbol{x}$ of the system of multivariate polynomial equations. When we shift some rows of $\boldsymbol{V}$ with the shift polynomial $g(\boldsymbol{x}) = c_{\boldsymbol{\alpha}_1}\boldsymbol{x}^{\boldsymbol{\alpha}_1} + \cdots + c_{\boldsymbol{\alpha}_k}\boldsymbol{x}^{\boldsymbol{\alpha}_k}$, then

$$\underbrace{\boldsymbol{S}_1\boldsymbol{V}}_{\text{before shift}} \quad \boldsymbol{D}_g = \underbrace{c_{\boldsymbol{\alpha}_1}\boldsymbol{S}_{\boldsymbol{x}^{\boldsymbol{\alpha}_1}}\boldsymbol{V} + \cdots + c_{\boldsymbol{\alpha}_k}\boldsymbol{S}_{\boldsymbol{x}^{\boldsymbol{\alpha}_k}}\boldsymbol{V}}_{\text{after shift}}, \tag{2.52}$$

where the diagonal matrix $\boldsymbol{D}_g \in \mathbb{C}^{m_a \times m_a}$ contains the evaluations of the shift polynomial $g(\boldsymbol{x})$ in the different solutions of the system. Hence, (2.52) corresponds to the expression

$$(\boldsymbol{S}_g\boldsymbol{V}) = (\boldsymbol{S}_1\boldsymbol{V})\boldsymbol{D}_g, \tag{2.53}$$

when combining the different row selection matrices in the right-hand side. The *row combination matrix*[13] $\boldsymbol{S}_g \in \mathbb{R}^{m_a \times q}$ selects the linear combination of rows

[13] When the shift is merely a monomial of (some of the) variables, the row combination matrix $\boldsymbol{S}_g$ is a row selection matrix because every shift only hits one row.

hit by the shift with $g(\boldsymbol{x})$.

Example 2.13 (continuing from p. 53). If we multiply the first two rows of (2.47) by the shift polynomial $g(\boldsymbol{x}) = 2x_1 + 3x_2$, then the row combination matrix $\boldsymbol{S}_g$ equals

$$\boldsymbol{S}_g = \begin{matrix} \scriptstyle 1 & \scriptstyle 2 & \scriptstyle 3 & \scriptstyle 4 & \scriptstyle 5 & \scriptstyle 6 \end{matrix} \\ \begin{bmatrix} 0 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 3 & 0 \end{bmatrix}, \tag{2.54}$$

while $\boldsymbol{S}_1$ is given by (2.49). Numerically, we obtain instead of (2.51) the expression

$$\underbrace{\begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}}_{\boldsymbol{S}_1\boldsymbol{V}} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 11 \end{bmatrix}}_{\boldsymbol{D}_g} = \underbrace{\begin{bmatrix} 1 & 11 \\ 2 & 44 \end{bmatrix}}_{\boldsymbol{S}_g\boldsymbol{V}}, \tag{2.55}$$

because the evaluation of the shift polynomial in the two common roots is equal to $g(\boldsymbol{x})|_{(1)} = 1$ and $g(\boldsymbol{x})|_{(2)} = 11$.

Code 2.5. Shifting with a shift polynomial $g(\boldsymbol{x}) = 2x_1 + 3x_2$ is performed by supplementing a shift polynomial `shift = [2 1 0; 3 0 1]` to `shiftmatrix`.

```
>> Sg = shiftmatrix(2,2,rows,[2 1 0; 3 0 1]);
>> (S1*V)*diag([1 11])

ans =
    1     11
    2     44

>> Sg*V =

ans =
    1     11
    2     44
```

2.4.1.2 Any numerical basis matrix

In practice, the multivariate Vandermonde basis matrix $\boldsymbol{V}$ of the null space is not known in advance, since it is constructed from the unknown solutions. The practical multidimensional realization problem uses a numerical basis matrix $\boldsymbol{Z} \in \mathbb{C}^{q \times m_a}$ of the null space of the Macaulay matrix $\boldsymbol{M}$ instead. This numerical basis matrix is obtained, for example, via the singular value or QR decomposition [97]. Before translating this theoretical multidimensional real-

ization problem[14] into a practical one, the next proposition makes this "special shift structure" more concrete.

Proposition 2.3 (Appendix C). The (affine) null space of the Macaulay matrix is **(backward) scalar multi-shift-invariant**. This means that if we select a row of a basis matrix of the null space and multiply/shift this row with one of the variables, then we obtain another row of that basis matrix (when the degree is *large enough*, i.e., $d \geq d_\circ$).

Therefore, as the scalar multi-shift-invariance is a property of the null space as a vector space and not of its specific basis matrix (Appendix C), a numerical basis matrix $\boldsymbol{Z}$ can be used. There exists a relation between these two bases, namely $\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{T}$, with $\boldsymbol{T} \in \mathbb{C}^{m_a \times m_a}$ a nonsingular transformation matrix. This relation transforms (2.53) into a generalized eigenvalue problem (GEP)

$$(\boldsymbol{S}_g\boldsymbol{Z})\boldsymbol{T} = (\boldsymbol{S}_1\boldsymbol{Z})\boldsymbol{T}\boldsymbol{D}_g, \tag{2.56}$$

where $\boldsymbol{T}$ contains the eigenvectors and $\boldsymbol{D}_g$ the eigenvalues of the matrix pencil $(\boldsymbol{S}_g\boldsymbol{Z}, \boldsymbol{S}_1\boldsymbol{Z})$. This can also be written as a standard eigenvalue problem (SEP):

$$(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_g\boldsymbol{Z})\boldsymbol{T} = \boldsymbol{T}\boldsymbol{D}_g. \tag{2.57}$$

The matrix of eigenvectors $\boldsymbol{T}$ could be used to retrieve the multivariate Vandermonde matrix $\boldsymbol{V}$, via $\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{T}$ and a normalization of the first row. This approach to set-up an eigenvalue problem that yields the solutions of the polynomial system is closely related to traditional eigenvalue computation of the multiplication matrices in algebraic geometry. Section 4.5 explains in more depth how both eigenvalue problems are related.

Example 2.14. Assume that the degree d of the Macaulay matrix for system (2.35) is the solution degree $d_\circ = 2$. We compute a numerical basis matrix $\boldsymbol{Z}$ of the null space of its Macaulay matrix: $\boldsymbol{Z} \in \mathbb{C}^{6\times 2}$ because this example has two (affine) solutions. The first two rows of $\boldsymbol{Z}$ are linearly independent, so when we shift with x_1, we can use $\boldsymbol{S}_1$ and $\boldsymbol{S}_{x_1}$ from (2.49) and (2.50), respectively. If we decide to shift the first two rows with x_2, then $\boldsymbol{S}_{x_2} \in \mathbb{R}^{2\times 6}$ is given by

$$\boldsymbol{S}_{x_2} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}. \tag{2.58}$$

Solving (2.57) twice results in the diagonal matrices

$$\boldsymbol{D}_{x_1} = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix} \tag{2.59}$$

[14] We actually know that, when the nullity of the Macaulay matrix stabilizes at the total number of solutions, its null space can be modeled as the column space of an observability matrix of a multidimensional descriptor system [80]. From that observation also stems the title of Section 2.4.1; the null space based root-finding approach could be considered as a multidimensional realization problem in the column space of that observability matrix.

and

$$\boldsymbol{D}_{x_2} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.60}$$

By combining both eigenvalues, we obtain the common roots of (2.35). We could also use $\boldsymbol{T}$ to reconstruct $\boldsymbol{V}$:

$$\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{T} = \left[\begin{array}{cc} 1 & 1 \\ \hline 2 & 4 \\ -1 & 1 \\ \hline 4 & 16 \\ -2 & 4 \\ 1 & 1 \end{array}\right] \tag{2.61}$$

after a normalization of the first row.

Code 2.6. We repeat the eigenvalue computation, but now start from a numerical basis matrix of the null space, via `null`.

```
>> Z = null(macaulay(toy1,2));
>> eig(Sx1*Z,S1*Z)

ans =
    2.0000
    4.0000
```

We can repeat this for the other variable:

```
>> Sx2 = shiftmatrix(2,2,rows,[1 0 1]); % shift with 1*x2
>> eig(Sx2*Z,S1*Z)

ans =
   -1.0000
    1.0000
```

2.4.2 Solutions with multiplicity greater than one

When all solutions are simple, every solution of the system corresponds to exactly one column in the multivariate Vandermonde basis matrix $\boldsymbol{V}$ of the null space, and every column contributes to the nullity of the Macaulay matrix. However, if solutions with multiplicity greater than one exist, the null space of the Macaulay matrix no longer contains only the multivariate Vandermonde solution vectors $\boldsymbol{v}|_{(j)}$, but also linear combinations of the partial derivatives of these solution vectors, i.e., it has become a confluent Vandermonde matrix [78].

Möller and Stetter [176] and Dayton et al. [68] elaborate in more detail on the consequences of solutions with multiplicity greater than one. Except for a loss of numerical accuracy in computing eigenvalues with multiplicity greater than one, multiplicity poses no problem for the above-described null space based root-finding approach [79].

However, to counter this loss of accuracy, Corless et al. [62] have proposed to use $n+1$ shift polynomials: one random shift polynomial,

$$g_0(\boldsymbol{x}) = \sum_{i=1}^{n} c_i x_i, \tag{2.62}$$

with every $c_i \in \mathbb{C}$ a random complex number, and n shift polynomials

$$g_i(\boldsymbol{x}) = x_i, \tag{2.63}$$

for $i = 1, \ldots, n$. The first random shift polynomial $g_0(\boldsymbol{x})$ has only multiplicities that come from common roots that are really identical (with probability one). A Schur decomposition of $(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{g_0}\boldsymbol{Z})$ yields the upper triangular matrix $\boldsymbol{D}_{g_0}$ and orthonormal matrix $\boldsymbol{Q}$ via planar rotations or Householder transformations:

$$\boldsymbol{Q}\boldsymbol{D}_{g_0}\boldsymbol{Q}^{\mathrm{H}} = (\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{g_0}\boldsymbol{Z}), \tag{2.64}$$

where $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{\mathrm{H}}$ because the matrix $\boldsymbol{Q}$ is orthonormal. The fact that the triangular form in (2.64) is obtained via planar rotations or Householder transformations results in a more accurate decomposition in the presence of multiplicities. By re-using the matrix $\boldsymbol{Q}$, every component $x_i|_{(j)}$ of the solution of the system is on the jth position of the diagonal of the upper triangular matrices $\boldsymbol{D}_{x_i}$, i.e.,

$$\begin{aligned} \boldsymbol{D}_{x_1} &= \boldsymbol{Q}^{\mathrm{H}}(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{g_1}\boldsymbol{Z})\boldsymbol{Q}, \\ &\vdots \\ \boldsymbol{D}_{x_n} &= \boldsymbol{Q}^{\mathrm{H}}(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{g_n}\boldsymbol{Z})\boldsymbol{Q}. \end{aligned} \tag{2.65}$$

Every upper triangular matrix $\boldsymbol{D}_{x_i}$ contains the different evaluations in one variable:

$$\boldsymbol{D}_{x_i} = \begin{bmatrix} x_i|_{(1)} & \times & \times \\ 0 & \ddots & \times \\ 0 & 0 & x_i|_{(m_a)} \end{bmatrix} \tag{2.66}$$

Example 2.14 (continuing from p. 56). Instead of a single shift polynomial or using a reconstruction of the Vandermonde basis matrix, we now use $n+1$ shift polynomials to retrieve the solutions. We shift with $g_0(\boldsymbol{x}) = 2x_1 + 3x_2$ to determine $\boldsymbol{D}_0$ and $\boldsymbol{Q}$ via (2.64), where $\boldsymbol{S}_{g_0}$ is given in (2.54). By re-using the matrix $\boldsymbol{Q}$, we can retrieve the common roots by combining the diagonal elements of

$$\begin{aligned} \boldsymbol{D}_{x_1} &= \boldsymbol{Q}^{\mathrm{H}}(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{x_1}\boldsymbol{Z})\boldsymbol{Q}, \\ \boldsymbol{D}_{x_2} &= \boldsymbol{Q}^{\mathrm{H}}(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{x_2}\boldsymbol{Z})\boldsymbol{Q}, \end{aligned} \tag{2.67}$$

where the row selection matrices for this example are already given in (2.49), (2.50) and (2.58). Since the solution components of a solution are at the same position on the diagonal of $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, we do not need to reconstruct $\boldsymbol{V}$.

Code 2.7. We repeat the previous eigenvalue computations, but now use the Schur decomposition:

```
>> [Q,~] = schur(inv(S1*Z)*(Sg*Z), `complex');
>> diag(Q'*inv(S1*Z)*(Sx1*Z)*Q)

ans =
    2.0000
    4.0000
```

An additional clustering step can be used to refine the solutions further [62]. After all, when working with floating-point algorithms, it is possible that exact multiplicity (i.e., different solutions that are numerically identical) is destroyed. By clustering the obtained solutions, for example, based on the their values in $g_0(\boldsymbol{x})$, similar solutions are considered to belong to the same cluster. If the solutions are clustered based on their values in the random shift polynomial $g_0(\boldsymbol{x})$, then the solutions in the same cluster should be very similar. The accuracy of the solutions could be improved by taking the mean value of every obtained x_i in the same cluster. However, this clustering step remains a heuristic: things can go wrong. For example, when different clusters are not well-separated, the clustering algorithm can consider two different clusters as one cluster and using the mean of the two clusters for every x_i in both clusters can deteriorate the overall accuracy [62].

Example 2.15. Consider the system with multiplicities larger than one

$$\begin{cases} p_1(\boldsymbol{x}) = (x_2 - 2)^3 = 0, \\ p_2(\boldsymbol{x}) = x_1 - x_2 + 1 = 0. \end{cases} \tag{2.68}$$

This system has a solution $(1, 2)$ with multiplicity 3. If the random shift polynomial is $g_0(\boldsymbol{x}) = 1.4193x_1 + 0.2916x_2$, then

$$\boldsymbol{Q} = \begin{bmatrix} -0.0000 - 0.6531\mathrm{i} & 0.0000 + 0.1785i & -0.7359 + 0.0000\mathrm{i} \\ 0.0000 - 0.3106\mathrm{i} & 0.0000 - 0.9495\mathrm{i} & 0.0453 + 0.0000\mathrm{i} \\ -0.0000 - 0.6907\mathrm{i} & 0.0000 + 0.2581i & 0.6755 + 0.0000\mathrm{i} \end{bmatrix} \tag{2.69}$$

and

$$\boldsymbol{D}_{g_0} = \begin{bmatrix} 2.0025 + 0.0000\mathrm{i} & 3.7559 + 0.0000\mathrm{i} & 0.0001 + 5.1484\mathrm{i} \\ 0.0000 + 0.0000\mathrm{i} & 2.0025 - 0.0000\mathrm{i} & 0.0001 + 4.6904\mathrm{i} \\ 0.0000 + 0.0000\mathrm{i} & 0.0000 + 0.0000\mathrm{i} & 2.0024 + 0.0000\mathrm{i} \end{bmatrix}. \tag{2.70}$$

Re-using the matrix $\boldsymbol{Q}$ from (2.69) for the other shift problems results in

$$\boldsymbol{D}_{x_1} = \begin{bmatrix} 1.0000 + 0.0000\mathrm{i} & 2.1953 + 0.0000\mathrm{i} & 0.0000 + 3.0092\mathrm{i} \\ 0.0000 + 0.0000\mathrm{i} & 1.0000 - 0.0000\mathrm{i} & 0.0000 + 2.7415\mathrm{i} \\ 0.0000 - 0.0000\mathrm{i} & 0.0000 - 0.0000\mathrm{i} & 1.0000 + 0.0000\mathrm{i} \end{bmatrix} \tag{2.71}$$

and

$$\boldsymbol{D}_{x_2} = \begin{bmatrix} 2.0000 + 0.0000\mathrm{i} & 2.1953 + 0.0000\mathrm{i} & 0.0000 + 3.0092\mathrm{i} \\ 0.0000 - 0.0000\mathrm{i} & 2.0000 - 0.0000\mathrm{i} & 0.0000 + 2.7415\mathrm{i} \\ 0.0000 - 0.0000\mathrm{i} & 0.0000 - 0.0000\mathrm{i} & 2.0000 + 0.0000\mathrm{i} \end{bmatrix}, \tag{2.72}$$

from which the triple solution $(1, 2)$ can be retrieved with accuracy 4.8×10^{-5}. A clustering algorithm considers the evaluations of the random shift polynomial $g_0(\boldsymbol{x})$ in the three solutions and identifies them as one cluster. For every variable, x_i, the geometric mean of the values of the solution components in that cluster can be used as an improved value for all $x_i|_{(j)}$, $j = 1, \ldots, 3$. Using this clustering step, we improve the accuracy to 4.6×10^{-15}.

Example 2.16. A larger example shows how the clustering step helps to improve the accuracy, especially when real solutions are split into conjugate pairs. The system

$$\begin{cases} p_1(\boldsymbol{x}) = -2 - 7x_1 + 14x_1^3 - 7x_1^5 + x_1^7 - x_2^7 - x_2^8 \\ \qquad + \big(7 - 42x_1^2 + 35x_1^4 - 7x_1^6\big)x_2 + \big(16 + 42x_1 - 70x_1^3 + 21x_1^5\big)x_2^2 \\ \qquad + \big(-14 + 70x_1^2 - 35x_1^4\big)x_2^3 + \big(-20 - 35x_1 + 35x_1^3\big)x_2^4 \\ \qquad + \big(7 - 21x_1^2\big)x_2^5 + (8 + 7x_1)x_2^6, \\ p_2(\boldsymbol{x}) = 7 - 42x_1^2 + 35x_1^4 - 7x_1^6 - 7x_2^6 - 8x_2^7 \\ \qquad + 2\big(16 + 42x_1 - 70x_1^3 + 21x_1^5\big)x_2 + 3\big(-14 + 70x_1^2 - 35x_1^4\big)x_2^2 \\ \qquad + 4\big(-20 - 35x_1 + 35x_1^3\big)x_2^3 + 5\big(7 - 21x_1^2\big)x_2^4 \\ \qquad + 6(8 + 7x_1)x_2^5 \end{cases} \tag{2.73}$$

is taken from [165] and has 49 real solutions (13 simple solutions and 18 solutions with multiplicity equal to 2). However, after computing the eigenvalues via the Schur decompositions, we obtain 13 simple, real solutions and 36 complex solutions, which appear in 18 complex conjugate pairs (Table 2.1). After clustering the evaluations of the random shift polynomial $g_0(\boldsymbol{x}) = -2.9443x_1 + 1.4384x_2$, we notice that the conjugate pairs belong to the same cluster. The geometric mean of each conjugate pair results in 18 real solutions with multiplicity equal to 2.

2.4.3 About the notion of a large enough degree

One central question in the above-described approach remains unanswered: "What is the solution degree $d_\circ$?" When increasing the degree d by invoking

Table 2.1. Solutions for system (2.73) obtained before the clustering step. There are 13 simple, real solutions, while the 18 real solutions with multiplicity equal to 2 split into complex conjugate pairs. Clustering the evaluations of the random shift polynomial $g_0(\boldsymbol{x}) = -2.9443x_1 + 1.4384x_2$ gathers these complex conjugate pairs in the same cluster, after which the geometric mean of the cluster results in the 18 real solutions with multiplicity equal to 2.

$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{g_0(x)}$
$-0.0086 \pm 0.0881\mathrm{i}$	$-1.8845 \mp 0.1825\mathrm{i}$	$-2.6853 \mp 0.5220\mathrm{i}$
$-0.3838 \mp 0.1787\mathrm{i}$	$1.5422 \mp 0.3258\mathrm{i}$	$3.3482 \pm 0.0576\mathrm{i}$
$0.0555 + 0.0000\mathrm{i}$	$-1.5814 + 0.0000\mathrm{i}$	$-2.4380 + 0.0000\mathrm{i}$
$0.6279 \pm 0.1760\mathrm{i}$	$1.8408 \mp 0.0458\mathrm{i}$	$0.7991 \mp 0.5842\mathrm{i}$
$-1.3614 \pm 0.4866\mathrm{i}$	$-1.8952 \mp 0.0806\mathrm{i}$	$1.2823 \mp 1.5485\mathrm{i}$
$-0.2011 + 0.0000\mathrm{i}$	$1.2425 + 0.0000\mathrm{i}$	$2.3795 + 0.0000\mathrm{i}$
$-0.5781 \pm 0.0395\mathrm{i}$	$0.5092 \mp 0.7348\mathrm{i}$	$2.4344 \mp 1.1733\mathrm{i}$
$0.9986 \mp 0.0368\mathrm{i}$	$-0.8187 \mp 0.0500\mathrm{i}$	$-4.1179 \pm 0.0365\mathrm{i}$
$0.8470 + 0.0000\mathrm{i}$	$1.4849 + 0.0000\mathrm{i}$	$-0.3581 + 0.0000\mathrm{i}$
$1.8930 + 0.0000\mathrm{i}$	$1.8758 + 0.0000\mathrm{i}$	$-2.8753 + 0.0000\mathrm{i}$
$-1.8224 \mp 0.4311\mathrm{i}$	$0.0993 \mp 0.5550\mathrm{i}$	$5.5085 \pm 0.4710\mathrm{i}$
$-0.3159 \pm 0.1849\mathrm{i}$	$-0.8770 \mp 0.5556\mathrm{i}$	$-0.3313 \mp 1.3435\mathrm{i}$
$0.9936 + 0.0000\mathrm{i}$	$1.3975 + 0.0000\mathrm{i}$	$-0.9152 + 0.0000\mathrm{i}$
$-0.7182 + 0.0000\mathrm{i}$	$-1.3384 + 0.0000\mathrm{i}$	$0.1896 + 0.0000\mathrm{i}$
$-0.9966 + 0.0000\mathrm{i}$	$-1.4220 + 0.0000\mathrm{i}$	$0.8888 + 0.0000\mathrm{i}$
$-2.2254 \pm 0.1452\mathrm{i}$	$-1.1306 \mp 0.5494\mathrm{i}$	$4.9260 \mp 1.2176\mathrm{i}$
$1.1654 + 0.0000\mathrm{i}$	$-0.2382 + 0.0000\mathrm{i}$	$-3.7741 + 0.0000\mathrm{i}$
$-3.1906 \pm 0.4831\mathrm{i}$	$-1.9452 \mp 0.0800\mathrm{i}$	$6.5962 \mp 1.5374\mathrm{i}$
$2.7581 \pm 0.2466\mathrm{i}$	$1.6080 \mp 0.2319\mathrm{i}$	$-5.8077 \mp 1.0597\mathrm{i}$
$2.5896 \mp 0.3197\mathrm{i}$	$0.6983 \mp 0.4135\mathrm{i}$	$-6.6199 \pm 0.3465\mathrm{i}$
$1.3211 \pm 0.0184\mathrm{i}$	$0.3496 \mp 0.3334\mathrm{i}$	$-3.3869 \mp 0.5339\mathrm{i}$
$3.0356 \pm 0.0913\mathrm{i}$	$1.8383 \pm 0.0142\mathrm{i}$	$-6.2933 \mp 0.2482\mathrm{i}$
$-1.1374 \mp 0.2113\mathrm{i}$	$0.0632 \mp 0.1137\mathrm{i}$	$3.4397 \pm 0.4586\mathrm{i}$
$1.1235 + 0.0000\mathrm{i}$	$0.8992 + 0.0000\mathrm{i}$	$-2.0145 + 0.0000\mathrm{i}$
$-3.3685 \mp 0.3429\mathrm{i}$	$-1.3803 \mp 0.4465\mathrm{i}$	$7.9323 \pm 0.3674\mathrm{i}$
$3.8185 \mp 0.1335\mathrm{i}$	$1.8324 \mp 0.2014\mathrm{i}$	$-8.6072 \pm 0.1033\mathrm{i}$
$2.3703 + 0.0000\mathrm{i}$	$0.9587 + 0.0000\mathrm{i}$	$-5.5999 + 0.0000\mathrm{i}$
$1.3435 + 0.0000\mathrm{i}$	$0.4718 + 0.0000\mathrm{i}$	$-3.2771 + 0.0000\mathrm{i}$
$-0.0865 + 0.0000\mathrm{i}$	$-0.2216 + 0.0000\mathrm{i}$	$-0.0640 + 0.0000\mathrm{i}$
$-2.6299 \pm 0.5018\mathrm{i}$	$-1.3988 \pm 0.2009\mathrm{i}$	$5.7310 \mp 1.1885\mathrm{i}$
$-4.0934 + 0.0000\mathrm{i}$	$-2.0236 + 0.0000\mathrm{i}$	$9.1415 + 0.0000\mathrm{i}$

multiplications with more monomials in the FSR, the nullity of the Macaulay matrix $\boldsymbol{M}$ eventually stabilizes at the total number of solutions m_b in the case of a zero-dimensional solution set (Section 2.6.1 discusses the case when the solution set is not zero-dimensional). It is possible to monitor this behavior by checking the nullity of $\boldsymbol{M}$ for increasing d. When d is equal to the **degree of regularity** d_*, any basis matrix of the null space has m_b linearly independent columns and, when checking the rank of this basis matrix from top to bottom, at least one linearly independent row per degree block[15]. The structure of a basis matrix for $d > d_*$ depends on whether the system has only affine solutions (Section 2.4.3.1) or affine solutions and solutions at infinity (Section 2.4.3.2).

2.4.3.1 Only affine solutions

When the system only has affine solutions ($m_b = m_a$), these linearly independent rows correspond to the affine standard monomials. For larger degrees $d > d_*$, they remain stable at their respective positions and new degree blocks contain no additional linearly independent rows. The basis matrix consists of two zones: a **regular zone** that contains the linearly independent rows related to the affine standard monomials and a **gap zone** without additional linearly independent rows. The degree d is equal to the solution degree $d_\circ$ when the gap zone exists of d_g degree blocks, for a shift polynomial with total degree equal to d_g.

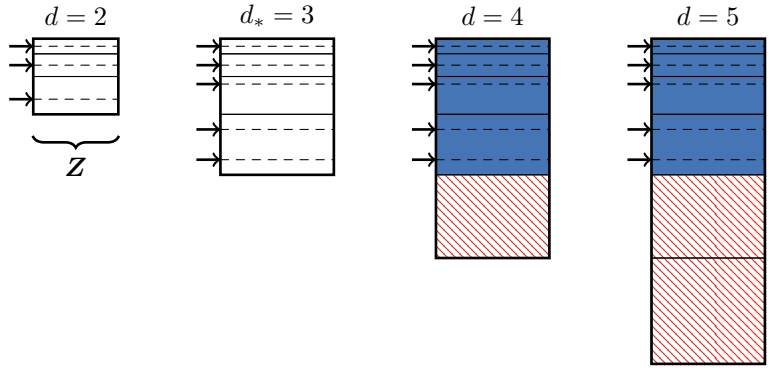
Example 2.17. Consider again system (2.35) and use the linear shift polynomial $g(\boldsymbol{x}) = 2x_1 + 3x_2$. We iteratively build a Macaulay matrix $\boldsymbol{M}$ for increasing degrees $d = 2, \ldots, 4$ (as described in Algorithm 2.1), which results in the following properties:

d	size	rank	nullity
2	4×6	4	2
3	9×10	8	2
4	16×15	13	2

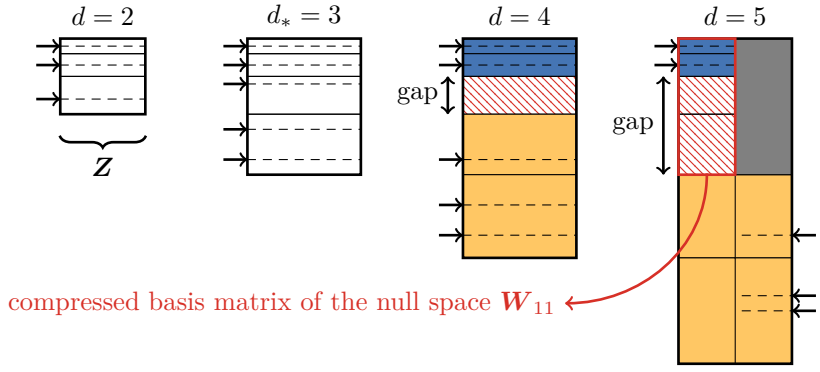
For this example, we notice that the nullity of $\boldsymbol{M}$ has already stabilized for degree $d = 2$. If we want to shift with a linear shift polynomial, the solution degree $d_\circ$ is equal to 2. We compute a numerical basis matrix $\boldsymbol{Z}$ of the null space. Performing row-wise rank checks from top to bottom shows that for $d = 2$ the gap zone can indeed accommodate the shift polynomial ($*$ indicates a degree block without any additional linearly independent rows):

d	standard monomials
2	$1 \mid x_1 \mid *$
3	$1 \mid x_1 \mid * \mid *$
4	$1 \mid x_1 \mid * \mid * \mid *$

[15] These linearly independent rows are related to the standard monomials, which is discussed in more depth in Section 4.5. In that section, we also link the row selection procedure with the Gröbner basis, border basis, and truncated normal forms.



(a) Only affine solutions



(b) Affine solutions and solutions at infinity

Figure 2.6. Basis matrix of the null space of a Macaulay matrix $\boldsymbol{M}$, which grows by invoking more multiplications with monomials in the FSR (increasing degree d). At a certain degree d_* (in this example $d_* = 3$), the nullity stabilizes at the total number of solutions m_b. In the situation with only affine solutions (Figure 2.6a), the linearly independent rows of the basis matrix, checked from top to bottom, correspond to the affine standard monomials and stabilize at their respective positions (indicated by dashed lines). New degree blocks contain no additional linearly independent rows when $d > d_*$. The basis matrix consists of a regular zone (■) and a gap zone (▨). However, when the system has solutions at infinity (Figure 2.6b), the linearly independent rows of the basis matrix that correspond to the standard monomials related to the solutions at infinity (also indicated by dashed lines) move to higher degree blocks when $d > d_*$; they constitute the singular zone (■) of the basis matrix. A gap zone emerges in the rows that separates these two types of linearly independent rows, and the influence of the solutions at infinity can be deflated via a column compression.

We continue with a numerical basis matrix $\boldsymbol{Z} \in \mathbb{C}^{10\times 2}$ of its null space (for $d = 2$) and observe that the first two rows of $\boldsymbol{Z}$, which correspond to the variables 1 and x_1, are linearly independent. As the nullity is 2, there are no solutions at infinity. The matrix $\boldsymbol{Z}$ has a structure similar as in Figure 2.6a: the first two degree blocks form the regular zone, while the other degree blocks belong to the gap zone. An application of the multiple Schur decompositions on this $\boldsymbol{Z}$, as in Example 2.14, results in the common roots of the system.

2.4.3.2 Affine solutions and solutions at infinity

Sometimes, a system also has solutions at infinity, due to the sparsity of the polynomials or interactions of some higher degree coefficients. The nullity of the Macaulay matrix after stabilization corresponds to the total number of solutions m_b of the system, which is now the sum of the number of affine solutions and the number of solutions at infinity ($m_b = m_a + m_\infty$). Every solution spans one basis vector in this null space, hence all the columns of the numerical basis matrix are linear combinations of affine solutions and solutions at infinity. Next to the affine standard monomials, also linearly independent rows related to the standard monomials that correspond to solutions at infinity appear in the basis matrix. When we increase the degree ($d > d_*$), the linearly independent rows that correspond to the affine standard monomials remain again stable at their respective positions, but the standard monomials that correspond to the solutions at infinity move to higher degree blocks when the FSR proceeds. Eventually, a gap in the rows emerges that separates both types of linearly independent rows. This gap grows when we keep increasing the degree $d > d_*$. Now, the basis matrix consists of three zones: a regular zone, a gap zone, and a **singular zone** that contains the linearly independent rows related to the standard monomials that correspond to the solutions at infinity. A column compression [78, 251] can remove the influence of the solutions at infinity from the null space.

Theorem 2.3. A numerical basis matrix $\boldsymbol{Z} = \begin{bmatrix}\boldsymbol{Z}_1^{\mathrm{T}} & \boldsymbol{Z}_2^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$ of the null space of the Macaulay matrix $\boldsymbol{M}$ is a $q \times m_b$ matrix, which can be partitioned into a $s \times m_b$ matrix $\boldsymbol{Z}_1$ (that contains the regular zone and gap zone) and a $(q-s)\times m_b$ matrix $\boldsymbol{Z}_2$ (that contains the singular zone), with $\operatorname{rank}(\boldsymbol{Z}_1) = m_a < m_b$. Furthermore, let the singular value decomposition of $\boldsymbol{Z}_1 = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{Q}^{\mathrm{T}}$. Then, $\boldsymbol{W} = \boldsymbol{Z}\boldsymbol{Q}$ is called the **column compression** of $\boldsymbol{Z}$ and can be partitioned as

$$\boldsymbol{W} = \begin{bmatrix}\boldsymbol{W}_{11} & \boldsymbol{0} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22}\end{bmatrix}, \tag{2.74}$$

where $\boldsymbol{W}_{11}$ is the $s \times m_a$ compressed numerical basis matrix of the (regular and gap zone of the) null space.

Proof. A proof of this theorem can be found in [78, p. 97]. □

Remark 2.4. It is also possible to compute $\boldsymbol{W}_{11}$ as the m_a left-most left singular vectors of $\boldsymbol{Z}_1$ in Theorem 2.3.

This column compression deflates the solutions at infinity and enables a straightforward use of the above-described affine null space based root-finding approach as if no solutions at infinity were present (we simply replace $\boldsymbol{Z}$ in (2.56) by $\boldsymbol{W}_{11}$), provided that the gap zone can *accommodate* for the shift polynomial $g(\boldsymbol{x})$, which means that the monomials with highest total degree hit by the shift must be present in the gap zone. Hence, the solution degree $d_\circ$ corresponds to the degree d for which the gap zone consists of d_g degree blocks, for a shift polynomial with total degree equal to d_g. This happens at $d_\circ = d_* + d_g$, where d_* typically corresponds to the degree for which the nullity has stabilized, but there also are more difficult situations (Example 2.19).

Example 2.18. Consider the polynomial system (2.38), which has 2 affine solutions and 2 solutions at infinity. We iteratively build the Macaulay matrix $\boldsymbol{M}$ for degrees $d = 2, \ldots, 4$ (as described in Algorithm 2.1) and obtain the following properties:

d	size	rank	nullity
2	2×6	2	4
3	6×10	6	4
4	12×15	11	4

The nullity remains stable from degree $d_* = 2$ onwards. When we compute a numerical basis matrix $\boldsymbol{Z}$ of the null space and perform row-wise rank checks from top to bottom, we notice that the null space has a different structure than in the previous example ($*$ indicates a degree block without any additional linearly independent rows):

d	standard monomials
2	$1 \mid x_1 \mid x_1^2, x_2^2$
3	$1 \mid x_1 \mid * \mid x_1^3, x_2^3$
4	$1 \mid x_1 \mid * \mid * \mid x_1^4, x_2^4$

Similar to Figure 2.6b, the null space consists, from degree $d = 3$ onwards, of three zones. A degree $d_\circ = 3$ Macaulay matrix suffices to find a gap zone of degree $d_g = 1$ and to shift with a linear shift polynomial $g(\boldsymbol{x})$. We perform a column compression on $\boldsymbol{Z} \in \mathbb{C}^{10 \times 4}$ in order to remove the solutions at infinity and obtain $\boldsymbol{W}_{11} \in \mathbb{C}^{6 \times 2}$ (Theorem 2.3). Schur decompositions (2.65) result in the affine solutions of the system: $(1, 1)$ and $(-2, -2)$.

Code 2.8. From the numerical basis matrix of the null space, we can find the degree of the gap zone and number of affine solutions via `gap(Z,d,n)`.

```
>> Z = null(macaulay(toy3,4));
>> [dgap, ma] = gap(Z,4,2)

dgap =
    2

ma =
    1
```

We remove the solutions at infinity via a column compression:

```
>> W11 = columncompression(Z,n,dgap);
```

The function `shiftnullspace(W11,shiftpoly)` sets-up and solves the different shift problems. `shiftpoly` is the polynomial $g_0(\boldsymbol{x})$ that determines the orthonormal matrix $\boldsymbol{Q}$ used in the other n shift problems with $g_i(\boldsymbol{x}) = x_i$.

```
>> D = shiftnullspace(W11,[randn(2,1) eye(2)]);
>> solutions = [D{2}, D{3}] % combine x1 and x2

solutions =
    1.0000     1.0000
   -2.0000    -2.0000
```

Example 2.19. Sometimes, the situation is more difficult. Consider the system of multivariate polynomial equations

$$\begin{cases} p_1(\boldsymbol{x}) = x_1 x_2 - 3 = 0, \\ p_2(\boldsymbol{x}) = x_1^2 - x_3^2 + x_1 x_3 - 5 = 0, \\ p_3(\boldsymbol{x}) = x_3^3 - 2x_1 x_2 + 7, \end{cases} \tag{2.75}$$

and iteratively build the Macaulay matrix $\boldsymbol{M}$ for degree $d = 3, 4, \ldots$ (as described in Algorithm 2.1):

d	size	rank	nullity
3	9×20	9	11
4	24×35	23	12
5	50×56	44	12
6	90×84	72	12
7	147×120	108	12

Algorithm 2.2 Null space based root-finding algorithm

Require: Macaulay matrix $\boldsymbol{M} \in \mathbb{C}^{p \times q}$ of degree $d_\circ$ (Algorithm 2.1)

1: Compute a numerical basis matrix $\boldsymbol{Z}$ of the null space of $\boldsymbol{M}$
2: Identify the gap zone and the number of affine solutions m_a via row-wise rank tests on $\boldsymbol{Z}$ (Section 2.4.3)
3: Use Theorem 2.3 to obtain the compressed numerical basis $\boldsymbol{W}_{11}$ of the null space (note that if $m_b = m_a$, then $\boldsymbol{W}_{11} = \boldsymbol{Z}$)
4: For a user-defined shift polynomial $g_0(\boldsymbol{x})$, solve the Schur decomposition

$$\boldsymbol{Q}\boldsymbol{D}_{g_0}\boldsymbol{Q}^{-1} = (\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{g_0}\boldsymbol{Z}), \tag{2.76}$$

where the matrices $\boldsymbol{S}_1$, $\boldsymbol{S}_{g_0}$, $\boldsymbol{Q}$, and $\boldsymbol{D}_{g_0}$ are defined as in (2.64)
5: Retrieve the different components $x_i|_{(j)}$ of the solutions from $\boldsymbol{D}_{x_i}$ in (2.65)
6: **return** $\boldsymbol{x}|_{(j)}$, for $j = 1, \ldots, m_a$

The nullity remains stable from degree $d = 4$ onwards, but when we compute a numerical basis matrix $\boldsymbol{Z}$ of the null space and perform row-wise rank checks from top to bottom, we notice that the null space only has a *large enough* gap zone from degree $d_\circ = 7$ onwards ($*$ indicates a degree block without any additional linearly independent rows):

d	standard monomials
3	$1 \mid x_1, x_2, x_3 \mid x_1^2, x_1x_3, x_2^2, x_2x_3 \mid x_1^3, x_2^3, x_2^2x_3$
4	$1 \mid x_1, x_2, x_3 \mid x_1^2, x_1x_3, x_2^2 \mid x_1^3, x_2^3, x_2^2x_3 \mid x_2^4, x_2^3x_3$
5	$1 \mid x_1, x_2, x_3 \mid x_1^2, x_1x_3, x_2^2 \mid x_2^3 \mid x_2^4, x_2^3x_3 \mid x_2^5, x_2^4x_3$
6	$1 \mid x_1, x_2, x_3 \mid x_1^2, x_1x_3 \mid x_2^3 \mid x_2^4 \mid x_2^5, x_2^4x_3 \mid x_2^6, x_2^5x_3$
7	$1 \mid x_1, x_2, x_3 \mid x_1^2, x_1x_3 \mid * \mid x_1^4 \mid x_2^5 \mid x_2^6, x_2^5x_3 \mid x_2^7, x_2^6x_3$

An analysis of this polynomial system learns that the degree of regularity $d_* = 6$, which is higher than the degree for which the nullity has stabilized.

2.4.4 Null space based root-finding algorithm

We summarize the different steps of the null space based root-finding algorithm in Algorithm 2.2.

Remark 2.5. Algorithm 2.2 only selects the linearly independent rows that correspond to the affine standard monomials. However, the row-wise rank checks from top to bottom to identify the linearly independent rows are numerically not very robust. Instead of selecting m_a linearly independent rows of $\boldsymbol{Z}$, the row selection matrix $\boldsymbol{S}_1 \in \mathbb{R}^{l \times q}$ can also select entire degree blocks of $\boldsymbol{Z}$ (but needs to contain at least m_a linearly independent rows to cover all the affine solutions). The row combination matrix $\boldsymbol{S}_g \in \mathbb{R}^{l \times q}$ selects again the rows in $\boldsymbol{Z}$ that are hit by the shift. Mathematically, a rectangular matrix pencil $(\boldsymbol{S}_g\boldsymbol{Z}, \boldsymbol{S}_1\boldsymbol{Z})$ is considered or the pseudo-inverse $(.^\dagger)$ is used to

obtain a solvable SEP

$$(\boldsymbol{S}_1\boldsymbol{Z})^\dagger(\boldsymbol{S}_g\boldsymbol{Z})\boldsymbol{T} = \boldsymbol{T}\boldsymbol{D}_g. \tag{2.77}$$

Shifting entire degree blocks replaces the row-wise rank checks by more efficient degree block-wise rank checks. For accuracy reasons, a Schur decomposition replaces the eigenvalue decomposition in (2.77):

$$\boldsymbol{Q}\boldsymbol{D}_g\boldsymbol{Q}^{-1} = (\boldsymbol{S}_1\boldsymbol{Z})^\dagger(\boldsymbol{S}_g\boldsymbol{Z}). \tag{2.78}$$

Example 2.20. To wrap up the null space based approach, we consider a polynomial system from magnetism [131],

$$\begin{cases} p_1(\boldsymbol{x}) = 2(x_2 + x_3 + x_4 + x_5 + x_6 + x_7) + x_1 - 1 = 0, \\ p_2(\boldsymbol{x}) = 2(x_1x_6 + x_2x_5 + x_2x_6 + x_3x_4) - x_6 = 0, \\ p_3(\boldsymbol{x}) = 2(x_1x_5 + x_2x_4 + x_2x_6 + x_3x_7) + x_3^2 - x_5 = 0, \\ p_4(\boldsymbol{x}) = 2(x_1x_4 + x_2x_3 + x_2x_5 + x_3x_6 + x_4x_7) - x_4 = 0, \\ p_5(\boldsymbol{x}) = 2(x_1x_3 + x_2x_4 + x_3x_5 + x_4x_6 + x_5x_7) + x_2^2 - x_3 = 0, \\ p_6(\boldsymbol{x}) = 2(x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_5x_6) - x_2 = 0, \\ p_7(\boldsymbol{x}) = 2\big(x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 + x_7^2\big) + x_1^2 - x_1 = 0, \end{cases} \tag{2.79}$$

which is a polynomial system in $n = 7$ variables. Via Algorithm 2.1, we can determine that the solution degree $d_\circ = 7$. The corresponding Macaulay matrix $\boldsymbol{M}$ is a 6468×3432 matrix. Since (2.79) has $m_b = 64$ solutions, the basis matrix of the null space is a 3432×64 matrix. The rank tests reveal that all the solutions are affine, hence no column compression is necessary. This polynomial system can be solved in 6.12 s (averaged over 30 experiments) via the row-wise approach, while it only takes 4.65 s (averaged over 30 experiments) via the block-wise approach mentioned in Remark 2.5. The maximum absolute residual error[16] of the row-wise approach and block-wise approach is 3.71×10^{-11} and 3.44×10^{-12}, respectively (both averaged over 30 experiments).

Remark 2.6. A correct basis matrix of the null space is essential in obtaining good absolute residual errors[16] for the solutions: Using a less accurate basis matrix to compute the solutions directly influences these errors. The accuracy of the computed basis matrix limits the maximum absolute residual errors of the solutions that can be obtained, but approximations of the solutions can still be obtained. The influence of a wrong rank check, however,

[16] We calculate the absolute residual error by substituting the computed solutions $(x_1^*, \ldots, x_n^*)$ in the polynomials and taking the sum of the 2-norm of the residuals, i.e., $\|\boldsymbol{e}\|_2 = \sum_{i=1}^{s} \|p_i(\boldsymbol{x}^*)\|_2$. More information about the error measures used in this text can be found in Appendix B.2.3.

is more severe. Wrongly estimating the number of affine solutions can have devastating consequences for the obtained solutions (with cases in which the obtained solutions are completely wrong).

2.5 Column space based approach

In this section, we consider the column space of the Macaulay matrix instead of its null space. The complementarity between both subspaces (Section 2.5.1) enables an equivalent column space based root-finding approach (Section 2.5.2). Although not strictly necessary, we also translate the column compression of the null space to a complementary compression of the column space of the Macaulay matrix (Section 2.5.3). Afterwards, we summarize the different steps of the column space based root-finding algorithm (Section 2.5.4).

2.5.1 Complementarity with the null space

The null space and column space of an arbitrary matrix share an intrinsic complementarity.

Lemma 2.3. Consider a matrix $\boldsymbol{M} \in \mathbb{R}^{p\times q}$, with $\text{rank}(\boldsymbol{M}) = r \leq \min(p, q)$. Let $\boldsymbol{Z} \in \mathbb{C}^{q\times(q-r)}$ be a full column rank matrix, the columns of which generate a basis matrix of the null space of $\boldsymbol{M}$, such that $\boldsymbol{MZ} = \boldsymbol{0}$. Using a row permutation matrix $\boldsymbol{P}$, reorder the rows of $\boldsymbol{Z}$ into $\boldsymbol{PZ} = \begin{bmatrix}\boldsymbol{Z}_A^{\mathrm{T}} & \boldsymbol{Z}_B^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$ and partition the columns of the matrix $\boldsymbol{M}$ accordingly with $\boldsymbol{P}^{-1}$ so that $\boldsymbol{MP}^{-1} = \begin{bmatrix}\boldsymbol{M}_A & \boldsymbol{M}_B\end{bmatrix}$. Consequently, $\boldsymbol{MZ} = \boldsymbol{MP}^{-1}\boldsymbol{PZ} = \boldsymbol{M}_A\boldsymbol{Z}_A + \boldsymbol{M}_B\boldsymbol{Z}_B = \boldsymbol{0}$ and the following property holds:

$$\text{rank}(\boldsymbol{M}_B) = r \Leftrightarrow \text{rank}(\boldsymbol{Z}_A) = q - r. \tag{2.80}$$

Proof. A proof of this lemma can be found in [78, p. 41]. □

The choice of the row permutation matrix $\boldsymbol{P}$ is not unique, there exist many possibilities to identify $q - r$ linearly independent rows in $\boldsymbol{Z}$. However, this lemma is more than a simple rank property, it expresses a complementarity for maximal sets[17] of linearly independent rows in $\boldsymbol{Z}$ with respect to maximal sets of linearly independent columns in $\boldsymbol{M}$. Obviously, we have $\boldsymbol{M}_A\boldsymbol{Z}_A = -\boldsymbol{M}_B\boldsymbol{Z}_B$, such that $\boldsymbol{M}_A = -\boldsymbol{M}_B\left(\boldsymbol{Z}_B\boldsymbol{Z}_A^{-1}\right)$ expresses the linearly dependent columns of $\boldsymbol{M}$ as a linear combination of the linearly independent ones and $\boldsymbol{Z}_B = -\left(\boldsymbol{M}_B^{\dagger}\boldsymbol{M}_A\right)\boldsymbol{Z}_A$ expresses the linearly dependent rows of $\boldsymbol{Z}$ as a linear combination of the selected linearly independent ones. This lemma leads to an

[17] A set of linearly independent rows is said to be maximal when it can not be expanded to another set of linearly independent rows by addition of any linearly independent row.

important observation when applying it on growing submatices: when indexing the linearly independent rows of $\boldsymbol{Z}$ (row-wise from top to bottom), it turns out that the "corresponding" columns of $\boldsymbol{M}$ are linearly dependent columns on the other columns of $\boldsymbol{M}$ (column-wise from right to left), as the next example illustrates.

Example 2.21. Consider a matrix $\boldsymbol{M} \in \mathbb{R}^{4\times 7}$ and a basis matrix of its null space $\boldsymbol{Z} \in \mathbb{R}^{7\times 3}$:

$$\boldsymbol{M}\boldsymbol{Z} = \begin{bmatrix} -2 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & -2 & 0 & 0 & 1 & 1 \\ -2 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ -1 & 1 & 0 \\ -1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ -1 & 1 & 1 \\ -1 & 1 & -1 \end{bmatrix} = \boldsymbol{0}. \quad (2.81)$$

The linearly independent rows of $\boldsymbol{Z}$, checked from top to bottom, are indexed as $\{1, 2, 6\}$. At the same time, the linearly dependent columns of $\boldsymbol{M}$, checked from right to left, are also indexed as $\{1, 2, 6\}$, in accordance to Lemma 2.3.

We can now apply Lemma 2.3 to the Macaulay matrix and any basis matrix of its null space. Observe that we can replace $\boldsymbol{Z}$ by a linear transformation $\boldsymbol{Z}\boldsymbol{T}$, so Lemma 2.3 is independent of the choice of basis matrix. The solutions of a system of multivariate polynomial equations give rise to standard monomials that correspond to the linearly independent rows of the basis matrix of the null space. When checking the rank of this basis matrix row-wise from top to bottom, every linearly independent row corresponds to one standard monomial. If we gather these linearly independent rows in a matrix $\boldsymbol{Z}_A$, which has full rank $q-r$, then we know, because of Lemma 2.3, that there exists a partitioning $\boldsymbol{M}_B$ of the columns of the Macaulay matrix, which has full rank r. Consequently, the remaining columns $\boldsymbol{M}_A$ of the Macaulay matrix linearly depend on the columns of $\boldsymbol{M}_B$. They correspond to the linearly independent rows of the basis matrix of the null space, and hence, every linearly dependent column of the Macaulay matrix, checked from right to left, also corresponds to exactly one standard monomial. This results in the following corollary.

Corollary. The solutions of a system of multivariate polynomial equations are related to both the linearly dependent columns of the Macaulay matrix (checked column-wise from right to left) and to the linearly independent rows of the basis matrix of its null space (checked row-wise from top to bottom).

Figure 2.7 visualizes the complementarity between both fundamental subspaces. Note that the gap zone in the basis matrix of the null space is a gap of linearly dependent rows, while the gap zone in the Macaulay matrix is a gap of linearly independent columns.

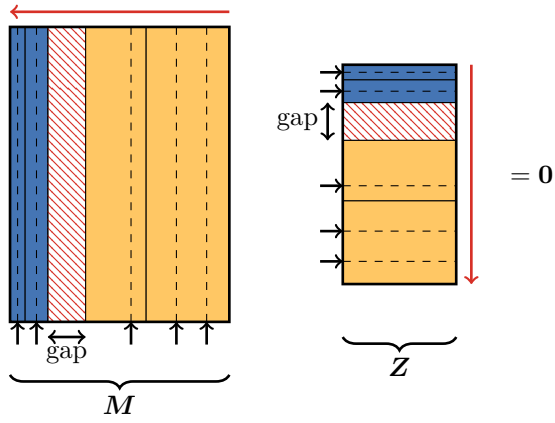


Figure 2.7. Complementarity of the null space and the column space of a Macaulay matrix: the standard monomials related to the solutions of a system of multivariate polynomial equations give rise to linearly independent rows in the basis matrix of the null space and to linearly dependent columns in the Macaulay matrix. If we check the rank of this basis row-wise from top to bottom, every linearly independent row corresponds to one solution. Because of the complementarity between the null space and column space of the Macaulay matrix, the linearly dependent columns of the Macaulay matrix, checked column-wise from right to left, correspond to the same standard monomials. In both matrices, we identify three zones: the regular zone (■), the gap zone (▨), and the singular zone (■).

2.5.2 Equivalent column space realization theory

Consider, for a polynomial system with zero-dimensional solution set, again a Macaulay matrix $\boldsymbol{M} \in \mathbb{C}^{p \times q}$, with degree equal to its solution degree $d = d_\circ$, and a numerical basis matrix $\boldsymbol{W} \in \mathbb{C}^{q \times m_b}$ of its null space after a column compression (Theorem 2.3). A shift of the linearly independent rows of the compressed basis matrix $\boldsymbol{W}_{11}$ with a shift polynomial $g(\boldsymbol{x})$ results in the expression

$$(\boldsymbol{S}_g \boldsymbol{W}_{11})\boldsymbol{T} = (\boldsymbol{S}_1 \boldsymbol{W}_{11})\boldsymbol{T}\boldsymbol{D}_g, \tag{2.82}$$

where the matrices $\boldsymbol{S}_1$, $\boldsymbol{S}_g$, $\boldsymbol{T}$, and $\boldsymbol{D}_g$ are defined as in (2.56).

Next, we define two new matrices $\boldsymbol{B}$ and $\boldsymbol{C}$. The matrix $\boldsymbol{B} \in \mathbb{C}^{m_a \times m_a}$ contains all the linearly independent rows of the matrix $\boldsymbol{W}_{11}$, which corresponds to the selection $\boldsymbol{S}_1 \boldsymbol{W}_{11}$ in (2.82), and is partitioned so that shifting with $g(\boldsymbol{x})$ each of its top m_h rows (denoted by $\boldsymbol{B}_1$) only hits rows inside $\boldsymbol{B}$ and shifting with $g(\boldsymbol{x})$ each of its bottom $m_c = m_a - m_h$ rows (denoted by $\boldsymbol{B}_2$) hits at least one row not in $\boldsymbol{B}$. We gather the m_c linear combination of rows hit by shifting the rows of $\boldsymbol{B}_2$ in the matrix $\boldsymbol{C} \in \mathbb{C}^{m_c \times m_a}$ and rewrite (2.82) as a

matrix pencil $(\boldsymbol{A}, \boldsymbol{B})$,

$$\underbrace{\begin{bmatrix} \boldsymbol{S}_g' \boldsymbol{B} \\ \boldsymbol{C} \end{bmatrix}}_{\boldsymbol{A}} \boldsymbol{T} = \underbrace{\begin{bmatrix} \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \end{bmatrix}}_{\boldsymbol{B}} \boldsymbol{T} \boldsymbol{D}_g, \tag{2.83}$$

where $\boldsymbol{S}_g'$ is the $m_h \times m_a$ row combination matrix that selects the $m_h = m_a - m_c$ linear combinations of rows in $\boldsymbol{B}$ hit by shifting the rows of $\boldsymbol{B}_1$. Shifting the rows in $\boldsymbol{B}_1$ leads to linear combinations of rows only in $\boldsymbol{B}$ ($\boldsymbol{B}_1 \to \boldsymbol{S}_g' \boldsymbol{B}$) and shifting the rows in $\boldsymbol{B}_2$ leads to linear combinations of rows in $\boldsymbol{B}$ and/or not in $\boldsymbol{B}$ ($\boldsymbol{B}_2 \to \boldsymbol{C}$). For example, if we shift the ith row of $\boldsymbol{B}_2$ and $g(\boldsymbol{x})$ hits the μth row of $\boldsymbol{B}$ ($\boldsymbol{b}_\mu$) and the νth row of $\boldsymbol{W}$ ($\boldsymbol{w}_\nu$ – not in $\boldsymbol{B}$), then the ith row of $\boldsymbol{C}$ is equal to $c_\mu \boldsymbol{b}_\mu + c_\nu \boldsymbol{w}_\nu$ (the coefficients c_μ and c_ν come from the shift polynomial). The matrix $\boldsymbol{D}_g$ is again a diagonal matrix that contains the evaluations of the shift polynomial $g(\boldsymbol{x})$ in the different solutions. The matrix $\boldsymbol{B}$ can be extracted from the column matrix in the left-hand side, after which an SEP appears (with $\boldsymbol{BT}$ as its matrix of eigenvectors):

$$\begin{bmatrix} \boldsymbol{S}_g' \\ \boldsymbol{C}\boldsymbol{B}^{-1} \end{bmatrix} \boldsymbol{B}\boldsymbol{T} = \boldsymbol{B}\boldsymbol{T}\boldsymbol{D}_g. \tag{2.84}$$

The matrix $\boldsymbol{B}$ is invertible because it contains m_a linearly independent rows by construction (the rows that correspond to the affine standard monomials). In the remainder of this section, we translate (2.84) to the column space via Lemma 2.3, avoiding the computation of a numerical basis matrix of the null space.

Example 2.22. When we consider system (2.35), we start by identifying the linearly dependent columns (from right to left) of $\boldsymbol{M}$. Like in Example 2.9, we notice that $d_\circ = 2$ and that the two left-most columns are linearly dependent on the other columns (from right to left), which means that the first two rows of $\boldsymbol{W}_{11}$ (i.e., 1 and x_1) constitute $\boldsymbol{B}$. The rows in $\boldsymbol{C}$ depend on the particular shift:

- For a shift with x_1, the rows that correspond to x_1 and x_1^2 are hit. This means that $\boldsymbol{C}$ contains only the 4th row (i.e., x_1^2). The matrix $\boldsymbol{B}$ could be split into $\boldsymbol{B}_1$ with the first row and $\boldsymbol{B}_2$ with the second row. The row selection matrix $\boldsymbol{S}_g'$ selects the second row from $\boldsymbol{B}$.
- When we shift with the polynomial $g(\boldsymbol{x}) = x_1^2$, the matrix $\boldsymbol{C}$ contains the 4th (x_1^2) and 7th (x_1^3) row. The matrix $\boldsymbol{B}_1$ is empty and no row selection matrix $\boldsymbol{S}_g'$ is needed.
- Choosing the same shift polynomial as in Example 2.17, $g(\boldsymbol{x}) = 2x_1 + 3x_2$, results again in an empty $\boldsymbol{B}_1$ matrix. The matrix $\boldsymbol{C}$ contains two combinations of rows: two times the 2nd (x_1) row with three times the 3rd (x_2) row and two times the 4th (x_1^2) row with three times the 5th ($x_1 x_2$) row.

The matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ contain rows (or linear combination of rows) of the matrix $\boldsymbol{W}_{11}$. We define the matrix $\boldsymbol{D} \in \mathbb{C}^{m_r \times m_a}$ (with $m_r = s - m_a - m_c$) as the matrix that contains the remaining rows of $\boldsymbol{W}_{11}$, such that every row of $\boldsymbol{W}_{11}$ is *represented* once in $\boldsymbol{B}$, $\boldsymbol{C}$, or $\boldsymbol{D}$. For example, if a row in $\boldsymbol{C}$ contains a linear combination of multiple rows of $\boldsymbol{W}$, then that row of $\boldsymbol{C}$ represents only one of those rows in the linear combination. The other rows of the linear combination need to be represented by other rows of $\boldsymbol{C}$, or they are included in $\boldsymbol{B}$ or $\boldsymbol{D}$. Consequently, we can reorder the basis matrix $\boldsymbol{W}$ as

$$\boldsymbol{P}\boldsymbol{W} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} \end{bmatrix}, \tag{2.85}$$

where the matrix $\boldsymbol{P}$ is a $q \times q$ row combination matrix that is invertible (because it is square and of full column rank by construction) and does not alter the rank structure of $\boldsymbol{W}$ (because it takes linear combinations of rows that already depend linearly on the rows in $\boldsymbol{B}$). Using Lemma 2.3, the columns of the Macaulay matrix can be reordered in accordance to the reordered basis matrix of the null space and obtain

$$\underbrace{\begin{bmatrix} \boldsymbol{N}_1 & \boldsymbol{N}_2 & \boldsymbol{N}_3 & \boldsymbol{N}_4 \end{bmatrix}}_{\boldsymbol{N}} \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} \end{bmatrix} = \boldsymbol{0}, \tag{2.86}$$

where every matrix $\boldsymbol{N}_i \in \mathbb{C}^{p \times q_i}$ corresponds to a subset of the columns (or linear combinations of columns) of $\boldsymbol{M}$. We call $\boldsymbol{N} = \boldsymbol{M}\boldsymbol{P}^{-1} \in \mathbb{C}^{p \times q}$ the **reordered Macaulay matrix**. Now, we apply a backward QR decomposition[18] on $\boldsymbol{N}$, which yields

$$\boldsymbol{Q} \begin{bmatrix} \boldsymbol{R}_{14} & \boldsymbol{R}_{13} & \boldsymbol{R}_{12} & \boldsymbol{R}_{11} \\ \boldsymbol{R}_{24} & \boldsymbol{R}_{23} & \boldsymbol{R}_{22} & \boldsymbol{0} \\ \boldsymbol{R}_{34} & \boldsymbol{R}_{33} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{R}_{44} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} \end{bmatrix} = \boldsymbol{0}, \tag{2.87}$$

or, if we pre-multiply both sides by $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{\mathrm{H}}$ (the labels denote the number

[18] Essentially, the backward QR decomposition triangularizes the reordered Macaulay matrix $\boldsymbol{N}$ as the traditional forward QR decomposition, but starts with the last column of $\boldsymbol{N}$ and iteratively works towards the first column of $\boldsymbol{N}$. Its result is similar to the result of the traditional forward QR decomposition of the matrix with all its columns flipped.

of rows/columns of the different blocks[19]),

$$\begin{array}{c} \\ q-s \\ m_r \\ m_c \\ p-q+m_a \end{array} \begin{array}{c} \begin{array}{cccc} m_a & m_c & m_r & q-s \end{array} \\ \begin{bmatrix} \boldsymbol{R}_{14} & \boldsymbol{R}_{13} & \boldsymbol{R}_{12} & \boldsymbol{R}_{11} \\ \boldsymbol{R}_{24} & \boldsymbol{R}_{23} & \boldsymbol{R}_{22} & \boldsymbol{0} \\ \boldsymbol{R}_{34} & \boldsymbol{R}_{33} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{R}_{44} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \end{array} \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} \end{bmatrix} = \boldsymbol{0}. \tag{2.88}$$

Note that $\boldsymbol{R}_{44}$ is always a zero matrix, since the rows of $\boldsymbol{B}$ are linearly independent and the complementarity of Lemma 2.3. From (2.88), it follows that $\boldsymbol{R}_{33}\boldsymbol{C} = -\boldsymbol{R}_{34}\boldsymbol{B}$, which means that

$$\boldsymbol{C}\boldsymbol{B}^{-1} = -\boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34}, \tag{2.89}$$

because $\boldsymbol{R}_{33}$ is always of full rank (since the rows of $\boldsymbol{C}$ depend linearly on the rows of $\boldsymbol{B}$ and the complementarity of Lemma 2.3). This relation helps to remove the dependency on the null space in (2.84) and yields a solvable SEP in the column space (with $\boldsymbol{H} = \boldsymbol{B}\boldsymbol{T}$),

$$\begin{bmatrix} \boldsymbol{S}_g' \\ -\boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34} \end{bmatrix} \boldsymbol{H} = \boldsymbol{H}\boldsymbol{D}_g, \tag{2.90}$$

or a GEP (to avoid the computation of the inverse of $\boldsymbol{R}_{33}$),

$$\begin{bmatrix} \boldsymbol{S}_g' \\ -\boldsymbol{R}_{34} \end{bmatrix} \boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}_{m_h} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_{33} \end{bmatrix} \boldsymbol{H}\boldsymbol{D}_g, \tag{2.91}$$

with $\boldsymbol{I}_{m_h} \in \mathbb{N}^{m_h \times m_h}$ the identity matrix. The matrix of eigenvectors $\boldsymbol{H} = \boldsymbol{B}\boldsymbol{T}$ corresponds to the partitioned linearly independent rows of the (affine) Vandermonde basis matrix $\boldsymbol{V}$, because the nonsingular transformation matrix $\boldsymbol{T}$ relates the rows of the numerical basis matrix $\boldsymbol{W}_{11}$ (or $\boldsymbol{B}$) to the rows of $\boldsymbol{V}$. Consequently, the eigenvalues in $\boldsymbol{D}_g$ and eigenvectors in $\boldsymbol{H}$ yield the solutions of the system of multivariate polynomial equations. Note that this complementary column space based approach does not require a column compression (although it does exist, see Section 2.5.3) to deflate the solutions at infinity, because the backward (Q-less) QR decomposition already separates them implicitly.

Example 2.22 (continuing from p. 72). We now continue the previous example and set up the GEP in (2.91) for a shift with $g(\boldsymbol{x}) = 2x_1 + 3x_2$. We

[19] A closer analysis of the flipped upper triangular matrix $\boldsymbol{R}$ reveals two special cases. Firstly, $\boldsymbol{W}_{21}$ and $\boldsymbol{W}_{22}$ are absent from $\boldsymbol{W}$ when there are no solutions at infinity. As a consequence, (2.86) no longer contains $\boldsymbol{N}_4$ and $q - s = 0$, which means that we can ignore the first block row and last block column of $\boldsymbol{R}$ (Example 2.22). Secondly, the size of the Macaulay matrix $\boldsymbol{M}$ determines the size of $\boldsymbol{R}_{44}$: when $p > q$ the matrix $\boldsymbol{R}_{44}$ is tall, but when $p < q$ the matrix $\boldsymbol{R}_{44}$ is wide and absent when $p = q - m_a$. Note that it can even happen that $p = q - m_b < q - m_a$ when the system has solutions at infinity (Example 2.23): then we need to use a larger degree d or remove the linearly dependent columns in $\boldsymbol{N}_4$, which correspond to the standard monomials related to the solutions at infinity, to ensure that $p \geq q - m_a$ and to obtain the structure of $\boldsymbol{R}$ as presented in (2.88).

build the row combination matrix $\boldsymbol{P} \in \mathbb{R}^{6\times 6}$,

$$\boldsymbol{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.92}$$

where the first two rows of $\boldsymbol{P}$ select the linearly independent rows of $\boldsymbol{B}$, the next two rows of $\boldsymbol{P}$ creates the linear combinations of rows of $\boldsymbol{C}$, and the remaining rows of $\boldsymbol{P}$ result in $\boldsymbol{D}$. We do not multiply $\boldsymbol{Z}$ by $\boldsymbol{P}$, but we use $\boldsymbol{P}^{-1}$ to reorder the columns of $\boldsymbol{M}$ into the matrices $\boldsymbol{N}_1$, $\boldsymbol{N}_2$, and $\boldsymbol{N}_3$. Since there are no solutions at infinity, we do not have a matrix $\boldsymbol{N}_4$ (Footnote 19). The backward (Q-less) QR decomposition of $\boldsymbol{N} = \boldsymbol{M}\boldsymbol{P}^{-1}$ results in the matrix

$$\boldsymbol{R} = \begin{matrix} {\scriptstyle 2} \\ {\scriptstyle 2} \end{matrix} \begin{bmatrix} \overset{2}{\boldsymbol{R}_{24}} & \overset{2}{\boldsymbol{R}_{23}} & \overset{2}{\boldsymbol{R}_{22}} \\ \boldsymbol{R}_{34} & \boldsymbol{R}_{33} & \boldsymbol{0} \end{bmatrix}, \tag{2.93}$$

from which the matrices $\boldsymbol{R}_{33}$ and $\boldsymbol{R}_{34}$ of (2.91) can be extracted. The GEP yields the same matrix $\boldsymbol{D}_g$ as in Example 2.13, namely

$$\boldsymbol{D}_g = \begin{bmatrix} 1 & 0 \\ 0 & 11 \end{bmatrix}. \tag{2.94}$$

Code 2.9. After creating the `rowcombinationmatrix(d,n,rows,shift)`, we can reorder the Macaulay matrix. We use the same shift polynomial as in Code 2.5. We use the same rows as in the null space based approach, so `rows = [1 2]`.

```
>> P = rowcombinationmatrix(2,2,rows,[2 1 0; 3 0 1]);
>> N = M*inv(P);
```

The Q-less backward QR decomposition results in the same eigenvalues:

```
>> [~,R] = qr(fliplr(N)); R = fliplr(R);
>> R33 = R(3:4,3:4); R34 = R(3:4,1:2);
>> eig(-R34,R33)

ans =
    1.0000
   11.0000
```

Example 2.23. Next, we consider system (2.38) to illustrate the influence of solutions at infinity. A similar analysis as in Example 2.11 (but now on the columns from right to left), shows us that, for degree $d = 3$, the 6×10 corresponding Macaulay matrix already has a gap zone. However, for this Macaulay matrix $p = q - m_b = 6 < q - m_a = 8$, the special case mentioned in Footnote 19. We have two options to proceed:

- We can alleviate this problem by considering a larger degree of the Macaulay matrix; the Macaulay matrix eventually will become tall.
- We can remove the linearly dependent columns that correspond to standard monomials related to the solutions at infinity. Since we identify these $m_\infty = 2$ linearly dependent columns with the column-wise rank checks, it is easy to remove them. We continue with a new Macaulay matrix of size 6×8, where $p = \widetilde{q} - m_a = 6$ because only the affine solutions remain.

Remark 2.7. Contrary to the null space based approach, where all the different components of the solutions can be retrieved from the (affine) multivariate Vandermonde basis matrix $\boldsymbol{V}$, the matrix $\boldsymbol{H}$ in the column space based approach does not necessarily contain all the components of the solutions. Therefore, a good choice of one or multiple shift polynomials $g_i(\boldsymbol{x})$ is indispensible, so that the matrices $\boldsymbol{D}_{g_i}$ yield the remaining components. Another strategy is to always shift with a random shift polynomial $g_0(\boldsymbol{x})$,

$$\boldsymbol{Q}\boldsymbol{D}_{g_0}\boldsymbol{Q}^{-1} = \begin{bmatrix} \boldsymbol{S}'_{g_0} \\ \boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34} \end{bmatrix}, \tag{2.95}$$

and the n different variables $g_i(\boldsymbol{x}) = x_i$,

$$\begin{aligned} \boldsymbol{Q}\boldsymbol{D}_{x_1}\boldsymbol{Q}^{-1} &= \begin{bmatrix} \boldsymbol{S}'_{x_1} \\ \boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34} \end{bmatrix}, \\ &\vdots \\ \boldsymbol{Q}\boldsymbol{D}_{x_n}\boldsymbol{Q}^{-1} &= \begin{bmatrix} \boldsymbol{S}'_{x_n} \\ \boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34} \end{bmatrix}, \end{aligned} \tag{2.96}$$

which are similar Schur decompositions as in Section 2.4.2. The matrices $\boldsymbol{R}_{33}$ and $\boldsymbol{R}_{34}$ have to be re-computed for every shift.

Example 2.24. Remember that for (2.35), the first two columns of a degree $d = 2$ Macaulay matrix are linearly dependent on the other columns (from right to left), which means that the matrix $\boldsymbol{H}$ only contains the evaluations of the standard monomials 1 and x_1 in the solutions. If we use the shift polynomial $g(\boldsymbol{x}) = x_1^2$, then (2.91) only contains information about the first variable

x_1 in the solutions. Using $n+1$ Schur decompositions, as in Remark 2.7, alleviates this issue.

2.5.3 Complementary column compression

Because of the structure of the backward QR decomposition, the influence of the solutions at infinity disappear implicitly when working in the column space. However, there also exists a complementary column compression in the column space that compresses the Macaulay matrix and can reduce the computational complexity of the column space based approach in some situations.

Theorem 2.4. The Macaulay matrix $\boldsymbol{M} = \begin{bmatrix}\boldsymbol{M}_1 & \boldsymbol{M}_2\end{bmatrix}$ of appropriate degree d is a $p \times q$ matrix, which can be partitioned into a $p \times (q-l)$ matrix $\boldsymbol{M}_1$ (that contains the columns that correspond to the affine solutions and the gap) and a $p \times l$ matrix $\boldsymbol{M}_2$ (that contains the columns that corresponds to the solutions at infinity), with $\operatorname{rank}(\boldsymbol{M}_2) = l - m_\infty$. Furthermore, let the QR decomposition with column pivoting of $\boldsymbol{M}_2\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix}\boldsymbol{Q}_1 & \boldsymbol{Q}_2\end{bmatrix}\boldsymbol{R}$. The matrix $\boldsymbol{Q}_2 \in \mathbb{C}^{p \times (p-l+m_\infty)}$ is an orthogonal basis of the left null space of $\boldsymbol{M}_2$. Then, $\boldsymbol{N} = \boldsymbol{Q}_2^{\mathrm{H}}\boldsymbol{M}$ is called the **complementary column compression** of $\boldsymbol{M}$ and can be partitioned as

$$\boldsymbol{N} = \begin{bmatrix}\boldsymbol{N}_1 & \boldsymbol{0}\end{bmatrix}, \tag{2.97}$$

where $\boldsymbol{N}_1$ is the $(p - l + m_\infty) \times (q - l)$ matrix that corresponds to the compressed Macaulay matrix.

Proof. Partition the Macaulay matrix $\boldsymbol{M} = \begin{bmatrix}\boldsymbol{M}_1 & \boldsymbol{M}_2\end{bmatrix}$ and pre-multiply by the matrix $\boldsymbol{Q}_2^{\mathrm{H}}$. Consequently, $\boldsymbol{N} = \boldsymbol{Q}_2^{\mathrm{H}}\boldsymbol{M} = \begin{bmatrix}\boldsymbol{Q}_2^{\mathrm{H}}\boldsymbol{M}_1 & \boldsymbol{Q}_2^{\mathrm{H}}\boldsymbol{M}_2\end{bmatrix}$. Since the matrix $\boldsymbol{Q}_2$ is an orthogonal basis matrix of the left null space of $\boldsymbol{M}_2$, the matrix $\boldsymbol{Q}_2^{\mathrm{H}}\boldsymbol{M}_2 = \boldsymbol{0}$ and the theorem is proven. □

Note that this matrix $\boldsymbol{Q}_2$ does not have to be calculated explicitly, but is computed at a certain point.

2.5.4 Column space based root-finding algorithm

We summarize the different steps of the column space based root-finding algorithm in Algorithm 2.3.

Remark 2.8. Note that, when the shift polynomial $g(\boldsymbol{x})$ is merely a monomial of (some of the) variables, the row combination matrix $\boldsymbol{P}$ is a row permutation matrix (every hit consists of only one row), and its inverse $\boldsymbol{P}^{-1}$ is equal to its transpose $\boldsymbol{P}^{\mathrm{T}}$. Applying $\boldsymbol{P}^{\mathrm{T}}$ to the Macaulay matrix $\boldsymbol{M}$ corresponds to reordering the columns of $\boldsymbol{M}$ in accordance to $\boldsymbol{P}\boldsymbol{W}$, which is quite easy to implement (no explicit construction of $\boldsymbol{P}$ is necessary).

Algorithm 2.3 Column space based root-finding approach

Require: Macaulay matrix $\boldsymbol{M} \in \mathbb{C}^{p\times q}$ of degree $d_\circ$ (Algorithm 2.1)

1: Replace $\boldsymbol{M}$ by the flipped upper triangular matrix $\boldsymbol{R}$ of its QR decomposition (optional)
2: Determine the linear dependent columns from right to left and reorder $\boldsymbol{M}$ or $\boldsymbol{R}$ as in (2.86)
3: Use Theorem 2.4 to obtain the compressed reordered matrix $\boldsymbol{N}_1$ (optional)
4: Compute the (Q-less) backward QR decomposition of the reordered matrix $\boldsymbol{N}$ (or the compressed $\boldsymbol{N}_1$)
5: For a user-defined shift polynomial $g_0(\boldsymbol{x})$, solve the Schur decomposition

$$\boldsymbol{Q}\boldsymbol{D}_{g_0}\boldsymbol{Q}^{-1} = \begin{bmatrix} \boldsymbol{I}_{m_h} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_{33} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{S}'_{g_0} \\ -\boldsymbol{R}_{34} \end{bmatrix}, \tag{2.98}$$

where the matrices $\boldsymbol{S}'_{g_0}$, $\boldsymbol{R}_{34}$, $\boldsymbol{R}_{34}$, $\boldsymbol{Q}$, and $\boldsymbol{D}_{g_0}$ are defined as in (2.91) and (2.95)
6: Retrieve the different components $x_i|_{(j)}$ of the solutions from $\boldsymbol{D}_{x_i}$ in (2.96)
7: **return** $\boldsymbol{x}|_{(j)}$, for $j = 1, \ldots, m_a$

Example 2.24 (continuing from p. 76). When the shift is a monomial, for example, $g(\boldsymbol{x}) = x_1$, the permutation and inverse permutation matrix are

$$\boldsymbol{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.99}$$

and

$$\boldsymbol{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.100}$$

which do not have to be constructed explicitly. We can simply select the corresponding columns of $\boldsymbol{M}$ directly.

Example 2.25. We repeat Example 2.20 and solve (2.79) via the column space of the Macaulay matrix. The solution degree $d_\circ = 7$ and we construct a Macaulay matrix $\boldsymbol{M} \in \mathbb{C}^{6468\times 3432}$. It is possible to replace $\boldsymbol{M}$ by the flipped upper triangular matrix $\boldsymbol{R} \in \mathbb{C}^{3432\times 3432}$ of its QR decomposition, which is a size reduction of almost 50%. The rank tests, on the columns

of the Macaulay matrix, reveal that the polynomial system has 64 affine solutions. Since we only use linear shift polynomials, we do not build $\boldsymbol{P}$ explicitly, but re-order the columns of $\boldsymbol{M}$ instead. This polynomial system can be solved in 3708.51 s (averaged over 30 experiments), which is much slower than via the null space. The main culprit is performing the column-wise rank checks on the Macaulay matrix, which considers larger matrices than the row-wise rank checks on the basis matrix of the null space. Note that, by using $\boldsymbol{R}$ instead of $\boldsymbol{M}$ for the rank checks, it is possible to speed up the computations, the polynomial system can then be solved in 2032.88 s (averaged over 30 experiments). The maximum absolute residual error[16] of this approach is 1.21×10^{-10} (averaged over 30 experiments).

2.6 Extensions to the root-finding approaches

In this section, we present some extensions to the above-described root-finding approaches. We show how to deal with a positive-dimensional solution set at infinity (Section 2.6.1), how a special shift polynomial might be useful (Section 2.6.2), and how to adapt the Macaulay matrix when the polynomials are given in another polynomial basis (Section 2.6.3).

2.6.1 Positive-dimensional solution sets at infinity

Due to sparsity and interactions of the higher-degree coefficients of the polynomials, not only isolated solutions at infinity, but also positive-dimensional solution sets at infinity may prevail. When the solution set of a system of multivariate polynomial equations is positive-dimensional (so, the number of solutions is infinite), the nullity of the Macaulay matrix does not stabilize. However, if the affine part of the solution set is zero-dimensional and the positive-dimensional part lives entirely at infinity, then the Macaulay matrix approaches can in some situations still be used to solve the system. Figure 2.8 sketches how the nullity and a basis matrix of the null space of the Macaulay matrix behave in this case. On the one hand, the linearly independent rows that correspond to the standard monomials associated with the affine solutions stabilize at their respective positions from a certain degree d_* on, as described in Section 2.4.3. The linearly independent rows that correspond to the standard monomials associated with the solutions at infinity, on the other hand, keep moving to higher degree blocks when further increasing the degree $d > d_*$ and, since the solution set at infinity is positive-dimensional, more linearly independent rows keep appearing in these higher degree blocks (Figure 2.8). A gap zone in the rows without any additional linearly independent rows still emerges in some situations (when $d > d_*$). Similar to the zero-dimensional case, the solution degree $d_\circ$ implies that the basis matrix can accommodate the shift polynomial, which means again that the gap zone must be able to accommodate the shift polynomial, so that a column compression or backward QR decomposition can deflate the (infinitely many) solutions at infinity. Afterwards, Algorithm 2.2 or Algorithm 2.3 yield the solutions similar as in the zero-dimensional case.

Example 2.26. Consider a system of multivariate polynomial equations with a positive-dimensional solution set at infinity,

$$\begin{cases} p_1(\boldsymbol{x}) = x_1 + x_2 - 1 = 0, \\ p_2(\boldsymbol{x}) = x_1 x_3 + x_2 x_4 = 0, \\ p_3(\boldsymbol{x}) = x_1 x_3^2 + x_2 x_4^2 - 1 = 0, \\ p_4(\boldsymbol{x}) = x_1 x_3^3 + x_2 x_4^3 = 0. \end{cases} \tag{2.101}$$

When iteratively building the Macaulay matrix $\boldsymbol{M}$ for degree $d = 4, \ldots 20$, the nullity does not stabilize:

d	size	rank	nullity
4	56×70	50	20
5	125×126	103	23
6	246×210	185	25
7	441×330	303	27
$\vdots$	$\vdots$	$\vdots$	$\vdots$
18	17766×7315	7266	49
19	22021×8855	8804	51
20	27000×10626	10573	53

However, when we compute a numerical basis matrix $\boldsymbol{Z}$ of the null space and perform row-wise rank checks from top to bottom for every degree, we notice that there emerges a gap zone at a certain degree ($*$ indicates a degree block without any additional linearly independent rows):

d	standard monomials
4	$1 \,\vert\, x_1, x_3 \,\vert\, x_1^2, x_1x_3, x_3^2, x_4^2 \,\vert\, x_1^3 \cdots$
5	$1 \,\vert\, x_1, x_3 \,\vert\, x_1^2 \,\vert\, x_1^3, x_1^2x_3, x_3^3, x_4^3 \cdots$
6	$1 \,\vert\, x_3 \,\vert\, x_1^2 \,\vert\, x_1^3 \,\vert\, x_1^4, x_1^3x_3, x_3^4, x_4^4 \cdots$
7	$1 \,\vert\, x_3 \,\vert\, * \,\vert\, x_1^3 \,\vert\, x_1^4 \,\vert\, x_1^5, x_1^4x_3, x_3^5, x_4^5 \cdots$

So, from the basis matrix $\boldsymbol{Z} \in \mathbb{C}^{330\times 27}$ of the null space of the Macaulay matrix for degree $d_\circ = 7$, we can retrieve the solutions when we use a column compression to deflate the (infinitely many) solutions at infinity. The resulting compressed basis matrix $\boldsymbol{W}_{11} \in \mathbb{C}^{15\times 2}$ can be used to set-up the Schur decompositions in Algorithm 2.2.

Code 2.10. When dealing with a positive-dimensional solution set, the numerical basis matrix of the null space must be checked for every degree. If there is no gap zone, then `gap` throws an error.

```
>> Z = null(macaulay(toy4,6));
>> [dgap, ma] = gap(Z,6,4)

    No gap found! Maybe increase the degree?
```

If there is a gap zone, then the positive-dimensional solution set at infinity can be removed and the isolated affine solutions are computed as before.

```
>> Z = null(macaulay(toy4,7));
>> [dgap, ma] = gap(Z,7,4) >> dgap =
    2

ma =
    = 2

>> W11 = columncompression(Z,ma,dgap);
>> D = shiftnullspace(W11,[randn(2,1) eye(2)],1,NaN);
>> solutions = [D{2}, D{3}, D{4}, D{5}]

solutions =
    0.5000     0.5000    -1.0000     1.0000
    0.5000     0.5000     1.0000    -1.0000
```

2.6.2 Special shift polynomials

In the previous sections, we have not yet spent much attention to the shift polynomial. Clearly, from a numerical perspective, it is interesting to use a random shift polynomial together with n shifts in the different variables. The random shift polynomial avoids the situation in which distinct solutions are wrongly considered to be equal, deteriorating the accuracy of the eigenvalue or Schur decompositions. However, in some applications, selecting a special shift polynomial may yield an additional benefit [78]. Since the eigenvalues of (2.56) and (2.91) correspond to the evaluations of the shift polynomial in the different solutions, we can use this GEP or Schur decomposition to evaluate a polynomial, e.g., the cost function of the underlying optimization problem or an additional constraint on the variables. For example, when considering a multivariate polynomial optimization problem, shifting with the cost function makes it possible to only compute the minimizer, avoiding the need for checking all the stationary points against the cost function.

There is an important remark when using a special shift polynomial: Although it is possible to use a shift polynomial of arbitrary high degree, a degree $d_g > 1$ is not interesting from a computational perspective since it requires a gap zone of more degree blocks, thus, implies using a higher solution degree

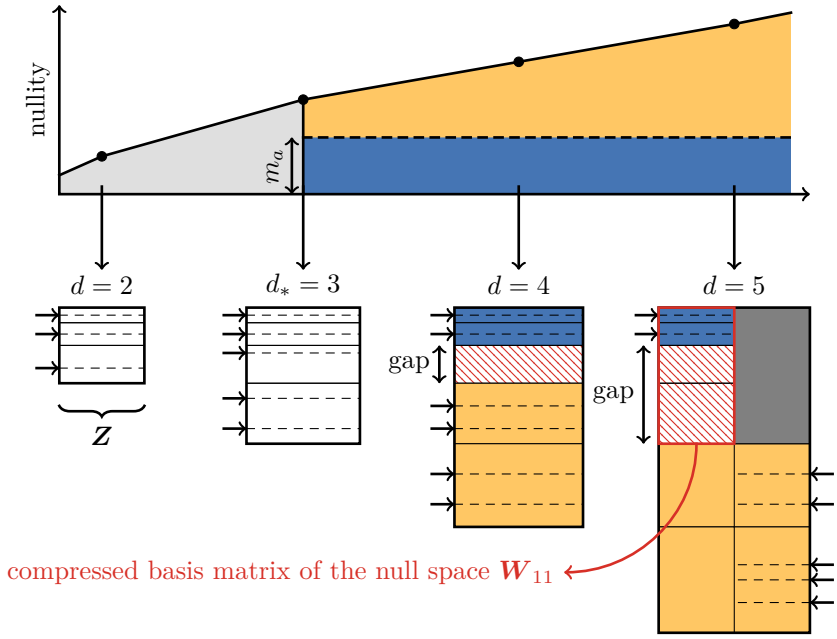


Figure 2.8. Nullity and basis matrix of the null space of a Macaulay matrix $\boldsymbol{M}$, which grows by invoking more multiplications with monomials in the FSR (increasing degree d), when the solution set at infinity is positive-dimensional. Because the solution set at infinity is positive-dimensional, the nullity does not stabilize: more linearly independent rows that correspond to the standard monomials related to the solutions at infinity keep appearing in the higher degree blocks. However, since the affine solution set is zero-dimensional, at a certain degree d_* (in this example $d_* = 3$), the linearly independent rows that correspond to the standard monomials associated with the affine solutions stabilize at their respective positions (■). Similar to the zero-dimensional case, the solution degree $d_\circ$ corresponds to the degree of $\boldsymbol{M}$ for which the gap zone (▨) must be able to accommodate the shift polynomial, so that we can deflate the (infinitely many) solutions at infinity via a column compression or backward QR decomposition (■).

$d_\circ$. This naive approach of using a special shift polynomial may pose computational restrictions on the applicability. However, there are two alternative approaches that circumvent this high degree of the shift polynomial:

- We can incorporate the cost function $\sigma(\boldsymbol{x})$ in the polynomial system as an additional variable x_{n+1} and add the equation

$$p_{x+1}(\widetilde{\boldsymbol{x}}) = x_{n+1}\sigma(\boldsymbol{x}) - 1 = 0 \tag{2.102}$$

  to the polynomial system. If we now shift with this new variable $g_0(\boldsymbol{x}) = x_{n+1}$, the eigenvalues correspond to the reciprocals of the cost function evaluated in the different stationary points [71, 262].

- We can combine different shift matrices as a matrix polynomial and deduce a high degree shift from the eigenvalues of that matrix polynomial [230].

These two approaches result in a linear shift polynomial, so a gap zone of only one degree block. The former approach considers a system of multivariate polynomial equations with an additional equation and variable, while the later approach requires combining multiple shift matrices.

Example 2.27. Consider a multivariate polynomial optimization problem, which minimizes the so-called two-dimensional three-hump camel back function [43, 177],

$$\begin{aligned} \min_{\boldsymbol{x}} \; & 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2, \\ \text{subject to } \; & x_1^3 - x_2 = 0. \end{aligned} \tag{2.103}$$

The stationary points of the cost function subject to the constraint can be found by solving the polynomial system

$$\begin{cases} p_1(\boldsymbol{x}) = 4x_1 - 4.2x_1^3 + x_1^5 + x_2 + 3x_1^2x_3 = 0, \\ p_2(\boldsymbol{x}) = x_1 + 2x_2 - x_3 = 0, \\ p_3(\boldsymbol{x}) = x_1^3 - x_2 = 0, \end{cases} \tag{2.104}$$

where x_3 represents the Lagrange multiplier. The polynomial system has $m_a = 5$ affine solutions. In order to shift with

$$g_0(\boldsymbol{x}) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2, \tag{2.105}$$

which has $d_g = 6$, the gap zone has to contain 6 degree blocks. Indeed, that puts a computational strain on the Macaulay matrix algorithm, since $d_\circ = 12$ results in a 704×455 Macaulay matrix, compared to a 80×84 Macaulay matrix when a linear shift polynomial is used.

In order to work with smaller Macaulay matrices, one of the two abovementioned alternatives can be used:

- We can introduce a new variable x_4, which corresponds to the function value of the cost function. After adding a new polynomial equation, namely $x_4 g_0(\boldsymbol{x}) - 1 = 0$, the largest eigenvalue of

$$(\boldsymbol{S}_1 \boldsymbol{Z})^{-1} (\boldsymbol{S}_{x_4} \boldsymbol{Z}) \tag{2.106}$$

  corresponds to the global minimum.

- Because the matrices $\boldsymbol{D}_i$ commute, we could also build the matrix polynomial $g_0\Big((\boldsymbol{S}_1 \boldsymbol{Z})^{-1} (\boldsymbol{S}_{x_1} \boldsymbol{Z}), (\boldsymbol{S}_1 \boldsymbol{Z})^{-1} (\boldsymbol{S}_{x_2} \boldsymbol{Z})\Big)$ and compute its smallest eigenvalue.

2.6.3 Another polynomial basis or monomial ordering

Within this text, we mainly consider the standard monomial basis, representing a polynomial $p(\boldsymbol{x}) \in \mathcal{P}_d^n$ as a finite linear combination of monomials $\boldsymbol{x}^{\boldsymbol{\alpha}}$ with complex coefficients $c_{\boldsymbol{\alpha}}$,

$$p(\boldsymbol{x}) = \sum_{\mathcal{A}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}, \tag{2.107}$$

where the summation runs over all the exponents in the support $\mathcal{A}$ (Definition 2.3). However, a polynomial can also be represented in a different polynomial basis

$$p(\boldsymbol{x}) = \sum_{\mathcal{B}} c_{\boldsymbol{\beta}} b_{\boldsymbol{\beta}}(\boldsymbol{x}), \tag{2.108}$$

where the polynomials $b_{\boldsymbol{\beta}}(\boldsymbol{x})$ are part of a set of polynomials that form a basis for $\mathcal{P}_d^n$. The complex coefficients $c_{\boldsymbol{\beta}}$ and support $\mathcal{B}$ might be different when using another polynomial basis. Two examples are the multivariate Chebyshev or Legendre polynomials.

When a multiplication of two basis polynomials can be written as a linear combination of basis polynomials, i.e.,

$$b_{\boldsymbol{\alpha}}(\boldsymbol{x}) b_{\boldsymbol{\gamma}}(\boldsymbol{x}) = \sum_{\mathcal{B}} c_{\boldsymbol{\beta}} b_{\boldsymbol{\beta}}(\boldsymbol{x}), \tag{2.109}$$

it is also possible to construct a Macaulay matrix for a system of multivariate polynomial equations in that polynomial basis. The null space based and column space based root-finding algorithm can be constructed in terms of this polynomial basis. While in some areas of mathematics, other polynomial bases are well established (e.g., in approximation theory [247, 248]), more research is necessary to fully grasp the consequences of working in another polynomial basis with the Macaulay matrix.

Example 2.28. Consider a system of multivariate polynomial equations in the multivariate Chebyshev basis of the first kind, i.e.,

$$\begin{cases} p_1(\boldsymbol{x}) = a_{00} t_{00}(\boldsymbol{x}) + a_{01} t_{01}(\boldsymbol{x}) + a_{10} t_{10}(\boldsymbol{x}) = 0, \\ p_2(\boldsymbol{x}) = b_{00} t_{00}(\boldsymbol{x}) + b_{01} t_{01}(\boldsymbol{x}) + b_{10} t_{10}(\boldsymbol{x}) = 0, \end{cases} \tag{2.110}$$

where the polynomials $t_{ij}(\boldsymbol{x}) \in \mathcal{P}^2$ are multivariate Chebyshev polynomials of the first kind [247], defined as

$$t_{ij}(\boldsymbol{x}) = t_i(x_1)t_j(x_2), \tag{2.111}$$

with $t_i(x_1) \in \mathcal{P}^1$ the ith univariate Chebyshev polynomial of the first kind in x_1 and $t_j(x_2) \in \mathcal{P}^1$ the jth univariate Chebyshev polynomial of the first kind in x_2. The first six bivariate Chebyshev polynomials of the first kind (in GRINVLEX ordering) are

$$t_{00}(\boldsymbol{x}) = t_0(x_1)t_0(x_2) = 1, \tag{2.112}$$
$$t_{10}(\boldsymbol{x}) = t_1(x_1)t_0(x_2) = x_1, \tag{2.113}$$
$$t_{01}(\boldsymbol{x}) = t_0(x_1)t_1(x_2) = x_2, \tag{2.114}$$
$$t_{20}(\boldsymbol{x}) = t_2(x_1)t_0(x_2) = 2x_1^2 - 1, \tag{2.115}$$
$$t_{11}(\boldsymbol{x}) = t_1(x_1)t_1(x_2) = x_1x_2, \tag{2.116}$$
$$t_{02}(\boldsymbol{x}) = t_0(x_1)t_2(x_2) = 2x_2^2 - 1. \tag{2.117}$$
$$\tag{2.118}$$

For Chebyshev polynomials, the property

$$t_k(x_i)t_l(x_i) = \frac{1}{2}\big(t_{k+l}(x_i) + t_{|k-l|}(x_i)\big) \tag{2.119}$$

holds, which means that we can create a Macaulay matrix by applying a FSR, using (2.119) to rewrite (2.44) for (2.110) as

$$\begin{array}{c} \\ t_{00} \\ t_{10} \\ t_{01} \\ t_{00} \\ t_{10} \\ t_{01} \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} t_{00} & t_{10} & t_{01} & t_{20} & t_{11} & t_{02} \end{array} \\ \left[\begin{array}{c|cc|c|cc} a_{00} & a_{10} & a_{01} & 0 & 0 & 0 \\ \frac{1}{2}a_{10} & a_{00} & 0 & \frac{1}{2}a_{10} & a_{01} & 0 \\ \frac{1}{2}a_{01} & 0 & a_{00} & 0 & a_{10} & \frac{1}{2}a_{01} \\ b_{00} & b_{10} & b_{01} & 0 & 0 & 0 \\ \frac{1}{2}b_{10} & b_{00} & 0 & \frac{1}{2}b_{10} & b_{01} & 0 \\ \frac{1}{2}b_{01} & 0 & b_{00} & 0 & b_{10} & \frac{1}{2}b_{01} \end{array}\right] \end{array}\!\!\begin{bmatrix} t_{00} \\ t_{10} \\ t_{01} \\ t_{20} \\ t_{11} \\ t_{02} \end{bmatrix} = \mathbf{0}. \tag{2.120}$$

For example, in order to create the second row of the Macaulay matrix, the polynomial $p_1(\boldsymbol{x})$ has to be multiplied with $t_{10}(\boldsymbol{x})$,

$$p_1(\boldsymbol{x})t_{10}(\boldsymbol{x}) = a_{00}t_{10}(\boldsymbol{x})t_{00}(\boldsymbol{x}) + a_{10}t_{10}(\boldsymbol{x})t_{10}(\boldsymbol{x}) + a_{01}t_{10}(\boldsymbol{x})t_{01}(\boldsymbol{x}) \tag{2.121}$$
$$= a_{00}t_{10}(\boldsymbol{x}) + \frac{a_{10}}{2}(t_{20}(\boldsymbol{x}) + t_{00}(\boldsymbol{x})) + a_{01}t_{11}(\boldsymbol{x}), \tag{2.122}$$

via property (2.119). Clearly, the structure of the Macaulay matrix is entirely different. But, also for other basis polynomials, the shift-invariant structure of the null space and column space allow us to set-up multidimensional realization problems that yield the common roots of the generating polynomials.

It is also possible to consider another monomial ordering. As mentioned before, the actual structure of the Macaulay matrix depends on the chosen

monomial ordering.

Example 2.29. To visualize the influence of the monomial ordering on the structure of the Macaulay matrix, we consider a system of random multivariate polynomials in three variables,

$$\begin{cases} p_1(\boldsymbol{x}) = \sum_{\mathcal{A}} a_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}} = 0, \\ p_2(\boldsymbol{x}) = \sum_{\mathcal{A}} b_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}} = 0, \\ p_3(\boldsymbol{x}) = \sum_{\mathcal{A}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}} = 0, \end{cases} \tag{2.123}$$

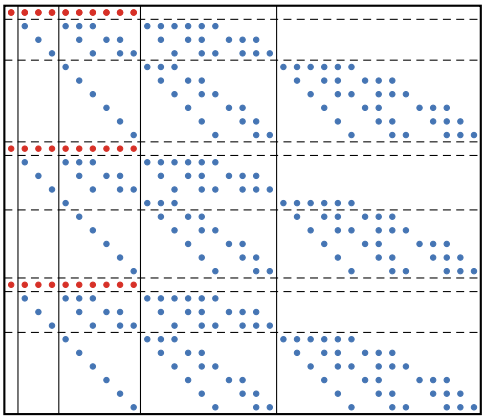
where the support $\mathcal{A}$ consists of all the monomials of total degree ≤ 3 and the coefficients $a_{\boldsymbol{\alpha}}, b_{\boldsymbol{\alpha}}, c_{\boldsymbol{\alpha}} \in \mathbb{C}$ are random numbers. The structure of the Macaulay matrix in the graded lexicographic (GRLEX) ordering is different from the structure when the GRINVLEX ordering is used, as visualized in Figure 2.9

Code 2.11. It is possible to construct the Macaulay matrix in any polynomial basis or monomial ordering. `basis` and `order` should be two functions that implement the basis multiplication and the position of a monomial in the monomial ordering. The system `walsh_cheb` is given in the Chebyshev polynomial basis and is part of MacaulayLab's database.

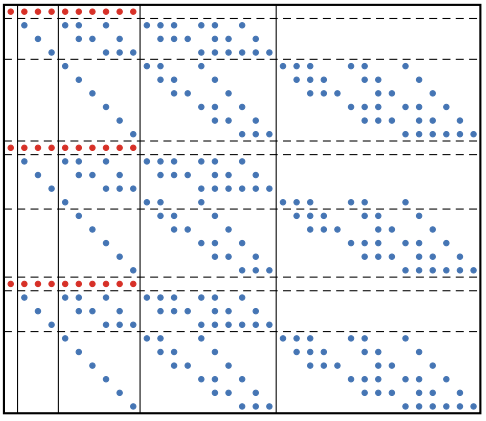
```
>> basis = @basischeb; % Chebyshev polynomial basis
>> order = @ordergrevlex; % grevlex monomial ordering
>> M = macaulay(walsh_cheb,3,basis,order);
```

2.7 Conclusion

We revised the numerical linear algebra approach that uses the right null space of a Macaulay matrix to solve systems of multivariate polynomial equations. We pointed at the complementarity of the right null space and column space of this Macaulay matrix and proposed a novel, complementary root-finding algorithm that considers the columns space of the Macaulay matrix instead. Contrary to null space based approach, the column space based approach does not require an explicit computation of a numerical basis matrix of the right null space, i.e., an expensive singular value decomposition, but directly works on the data in the columns of the Macaulay matrix. In that context, we also proposed the complementary column space compression, which compresses the Macaulay matrix and removes the influence of the solutions at infinity. This compression step is not strictly necessary in the column space based approach,



(a) Macaulay matrix in GRINVLEX ordering



(b) Macaulay matrix in GRLEX ordering

Figure 2.9. Macaulay matrix in two different monomial orderings for the random multivariate polynomial system (2.123). The Macaulay matrix has a different structure in the GRINVLEX ordering than in the GRLEX ordering, while the seed equations are the same polynomial equations in both figures. While the coefficients of the seed equations, indicated by red dots (●), are at the same positions in both Macaulay matrices, the shifted coefficients, indicated by blue dots (●), appear at different positions.

since the backward QR decomposition already removes the solutions at infinity implicitly. We provided three extensions to both root-finding algorithms that can be useful in particular applications: (i) dealing with positive-dimensional solution sets at infinity, (ii) using special shift polynomials, and (iii) changing the polynomial basis and monomial ordering.

Motivational example. To illustrate the above-mentioned contributions, we retake the motivational example and solve the polynomial system in (2.2) via the developed numerical linear algebra approaches for a random sequence of output data, namely

$$\boldsymbol{y} = \begin{bmatrix} 0.1001 \\ -0.5445 \\ 0.3035 \\ -0.6003 \end{bmatrix}. \tag{2.124}$$

The solutions of the polynomial system are the critical points of the optimization problem in (1.2). The solution that results in the smallest value of the cost function corresponds to the globally optimal parameters of the first-order ARMA model. Both the null space based and column space based approach retrieve the different critical points for a 1800×1287 Macaulay matrix of degree $d_\circ = 8$. There is only one real solution for this data sequence, namely $\alpha_1 = 0.3817$ and $\gamma_1 = -0.5789$. Table 2.2 contains a comparison of the different solution approaches. Using block-wise rank checks clearly has a computational advantage in this example, while the numerical robustness is not clear from this polynomial system (there are no difficult rank checks). These block-wise rank checks are not yet available in the column space based approach, resulting in a slower computation of the solutions. Furthermore, the fact that the Macaulay matrix is tall results in more expensive rank checks in the column space based approach, compared to the row-wise null space based approach. The current column space based approach does not yet exploit the fact that the Macaulay matrix is very sparse and structured.

Two of the above-mentioned extensions to the root-finding algorithms come into the picture while solving the motivational example:

- The solution set of the polynomial system is not zero-dimensional. However, since the affine part of the solution set is zero-dimensional, the solutions can be retrieved after deflating the (infinite number of) solutions at infinity, either via a column compression or the backward QR decomposition.

- Since the cost function of the related optimization problem can be written as a polynomial

$$g_0(\boldsymbol{x}) = l_1^2\gamma_1^2 + l_2^2\gamma_1^2 + 2l_1l_2\gamma_1 + l_1^2 + l_3^2\gamma_1^2 + 2l_2l_3\gamma_1 + l_2^2 + l_3^2 \tag{2.125}$$

  it can be used as special shift polynomial. The resulting eigenvalues then correspond to the evaluations of the cost function in the stationary point, resulting in a good approach to identify the globally optimal parameters of the ARMA model.

Table 2.2. Mean computation time and maximum absolute residual error[16] for solving the motivational example via the different numerical linear approaches developed in this chapter, averaged over 30 experiments. The block-wise null space based approach performs the rank checks on entire degree blocks, leading to a much faster execution. Because the Macaulay matrix is tall, the column-wise column space based approach has to perform more expensive rank checks than the row-wise null space based approach, while the number of rank checks is exactly the same.

approach	**time**	$\mathbf{max}\|\boldsymbol{e}\|_{\mathbf{2}}$
block-wise null space	0.86 s	1.7×10^{-12}
row-wise null space	9.00 s	1.6×10^{-12}
column-wise column space	288.05 s	2.0×10^{-13}

Considering both solution approaches, it would be interesting to gain more insight in the numerical properties of the algorithms; making claims about the numerical stability and assessing how every step influences the absolute residual errors[16] of the solutions. This insight may result in techniques to circumvent troublesome steps in the algorithms. Furthermore, the complementary column space based root-finding algorithm has created several research opportunities:

- As highlighted by the motivational example, the column space based approach only works column-wise, while the null space based approach considers entire degree blocks. The latter is preferred, both from a computational and numerical perspective. One of our current research efforts exists in enabling also a degree block-wise column space based approach. Furthermore, the column space based approach does not yet exploit the structure and sparsity of the Macaulay matrix, which can provide a computational advantage compared to working with the unstructured and dense (right) null space.

- Furthermore, systems of multivariate polynomial equations in applications are often structured. The most common structure that appears in practical problems is the sparsity of the support. Clearly, implementations that exploit this inherent structure of the polynomial systems have the potential to be computational much more efficient than general purpose solvers [30, 32, 34].

- An additional research question emerges from the observation that it is possible to build the Macaulay matrix with a different polynomial basis and monomial ordering: “Can the change of polynomial basis or monomial ordering result in superior numerical and/or computational properties?”

Historical and bibliographical notes

The available literature on (multivariate) root-finding is enormous. In the remainder of this chapter, we try to give an overview of the existing algorithms and position their historical evolution. More information and references can be found in standard books, such as [65, 66, 83, 222, 237].

Gröbner basis and Buchberger's algorithm

As mentioned in the introduction, the main focus of algebraic geometry in most of the 19th and 20th century was more on abstract algebra than on multivariate polynomial system solving. The computational aspects only re-emerged in the 1960s with Buchberger's work [49] on manipulating systems of multivariate polynomial equations. Buchberger's algorithm computes a so-called Gröbner basis[20], which has been one of the main tools to solve systems of multivariate polynomial equations ever since. A Gröbner basis is a set of multivariate polynomials that have desirable properties, especially useful in computations with polynomials [238]. Loosely speaking, the Buchberger's can be understood as the polynomial generalization of the Gaussian elimination algorithm or as the multivariate generalization of the Euclidean greatest common divisor (GCD) algorithm [78]. Via its Gröbner basis, it is possible to reduce the problem of solving the multivariate polynomial system into several univariate root-finding problems.

Gröbner bases have become the backbone of nearly all computational algorithms in algebraic geometry. Since then, more computational and applied results came forth from algebraic geometry, and a dedicated subarea, called computational algebraic geometry or computer algebra, gained more attention. Computer algebra has become very important in the past 50 years, thanks to the exploded available computing power [66].

The methods of Faugère [88, 89] and their extensions, like G^2V [92] and $ImpG^2V$ [94], are currently considered to be the most efficient methods to compute a Gröbner basis. Gröbner bases have dominated computer algebra since their inception, but remain computationally very expensive[21] and are based upon infinite precision (symbolic) computations and, therefore, employ rational numbers. This often results in huge coefficients (e.g., coefficients with tens or hundreds of digits), which means that their extensions to floating-point arithmetic are known to be rather cumbersome [138, 230]. The numerical instability of Gröbner basis computations have limited their use in numerical context [242]. This major drawback has motivated the introduction of border

[20] Hironaka [115] independently discovered this concept one year earlier and called it a standard basis. The interested reader finds nice introductions to the Gröbner basis in the short text by Sturmfels [238] or in the books by Cox et al. [65, 66].

[21] It is well-known that, in the worst-case, the complexity of computing a Gröbner basis is doubly exponential in the number of variables [175]. However, it is important to note that this is only an upper bound on the complexity. These worst-case estimates have led to the unfortunately widespread belief that using Gröbner bases is not useful for systems beyond simple toy problems, but for some special classes of systems of multivariate polynomial equations, implementations like Faugère's F_5 methods, can be very efficient [17].

bases [14, 181] and truncated normal forms [185, 243], which greatly improve the numerical stability.

Solutions via resultants

This approach finds its origin with the work by, for example, Sylvester [239] and Macaulay [159, 160], in the 18th and 19th century. There are several ways of using resultants for solving a system of polynomial equations. One strategy involves employing the u-resultant, which recovers the solution coordinates through a GEP [127]. Another approach is to use resultants to eliminate variables from the equations, the so-called hidden variable resultants [52, 163, 164, 188, 224]. This leads to a polynomial eigenvalue problem (PEP), which can be solved again via standard techniques from linear algebra. Diverse resultant constructions, e.g., Sylvester, Macaulay, Bézoutian, and Cayley resultants, are available. The resultant approach is especially useful in the bivariate setting [224]. Despite yielding satisfactory results in practice, these methods are intrinsically numerically unstable due to the inherent "projecting away some variables", as shown by Noferini and Townsend [191].

Other Macaulay matrix algorithms

Despite their manifest common historical ground, the linear algebra link between multivariate polynomials and eigenvalue problems has been neglected in most of the algebraic geometry literature since the end of the nineteenth century until well into the twentieth century. During the 1980s, the work of Lazard and Stetter (and coworkers) revived the interest in matrix-based methods for solving systems of multivariate polynomial equations: Lazard [153, 154] observed the resemblance between Buchberger's algorithm and Gaussian elimination of a Macaulay matrix, while Auzinger and Stetter [14] rigorously established the connection between multivariate polynomial system solving and eigenvalue decompositions only a few years later. This was further explored by, among others, Corless et al. [61], Emiris and Mourrain [87], Hanzon and Jibetean [106], Mourrain and Pan [183], and Stetter [229].

However, Stetter [230] observed that, at that time, the only way to obtain the common roots as the eigenvalues of a matrix using commonly available software was via Gröbner basis methods; hence, the approach was partially symbolic of nature. Batselier, De Moor, and Dreesen [21, 78, 80] have overcome this hurdle and developed a pure (numerical) linear algebra approach to solve systems of multivariate polynomial equations, using the right null space of a rectangular Macaulay matrix and techniques from systems theory and linear algebra to create a multidimensional realization problem in the right null space of the Macaulay matrix that contains the solutions of the system.

Recently, some new adaptations to the Macaulay matrix approach have been considered. Truncated normal forms have been introduced in [185, 243, 244] to counter the numerical issues with the selection of the standard monomials. The Macaulay matrix can also be adapted such that the resulting matrices are smaller, which is interesting from computational point of view. By only

considering the monomials in the dilated Newton polytope of the support of the polynomials, the approach of Bender [30] and Bender and Telen [32] does not require to construct all the rows/columns of the Macaulay matrix. In [31], systems of multivariate polynomials are solved over the toric variety.

Another interesting adaptation to mention is the work of Vanderstukken and De Lathauwer [255], Vanderstukken et al. [256], and Widdershoven et al. [274], in which the shift problems in the right null space of the Macaulay matrix are combined into a tensor and a tensor decomposition yields the common roots of the generating polynomials.

Iterative root-finding methods

One particular type of iterative root-finding methods are the homotopy continuation methods, which employ a hybrid mixture of techniques from algebraic geometry and nonlinear optimization to continuously deform a starting system with known solutions into the required system with unknown solutions, while tracking the solutions (see, for example, [157] and [265]). Homotopy continuation methods are inherently parallel, i.e., each isolated solution can be computed independently. They are currently among the most competitive algorithms to solve systems of multivariate polynomial equations, although issues with ill-conditioning still exist (i.e., path jumping). Their main disadvantage is that they only work for square polynomial systems, which have as many polynomial equations as variables.

An additional category of iterative root-finding methods are subdivision methods. These methods employ domain reductions for an iterative refinement of the subregions where the solutions may be located. An example of such a method is the algorithm presented in [184].

Deterimental representations

It is a well-known fact that the roots of a univariate polynomial can be computed as the eigenvalues of its companion matrix (Appendix A.3.1). This allows us to numerically compute the roots of a univariate polynomial via efficient tools from numerical linear algebra, an approach used in, for example, Matlab. In [197, 201], this idea is extended to bivariate polynomial systems. Given two bivariate polynomials, $p_1(\boldsymbol{x})$ and $p_2(\boldsymbol{x})$, matrices $\boldsymbol{B}_{ij}$ are constructed such that

$$\begin{cases} \det(\boldsymbol{B}_{10} + x_1\boldsymbol{B}_{11} + x_2\boldsymbol{B}_{12}) = p_1(\boldsymbol{x}), \\ \det(\boldsymbol{B}_{20} + x_1\boldsymbol{B}_{21} + x_2\boldsymbol{B}_{22}) = p_2(\boldsymbol{x}). \end{cases} \tag{2.126}$$

This problem corresponds to solving the square multiparameter eigenvalue problem

$$\begin{cases} (\boldsymbol{B}_{10} + x_1\boldsymbol{B}_{11} + x_2\boldsymbol{B}_{12})\boldsymbol{x}_1 = \boldsymbol{0}, \\ (\boldsymbol{B}_{20} + x_1\boldsymbol{B}_{21} + x_2\boldsymbol{B}_{22})\boldsymbol{x}_2 = \boldsymbol{0}, \end{cases} \tag{2.127}$$

which could be solved again with standard tools from numerical linear algebra, or even with the algorithms from the next chapter (Section 3.2.2.1).

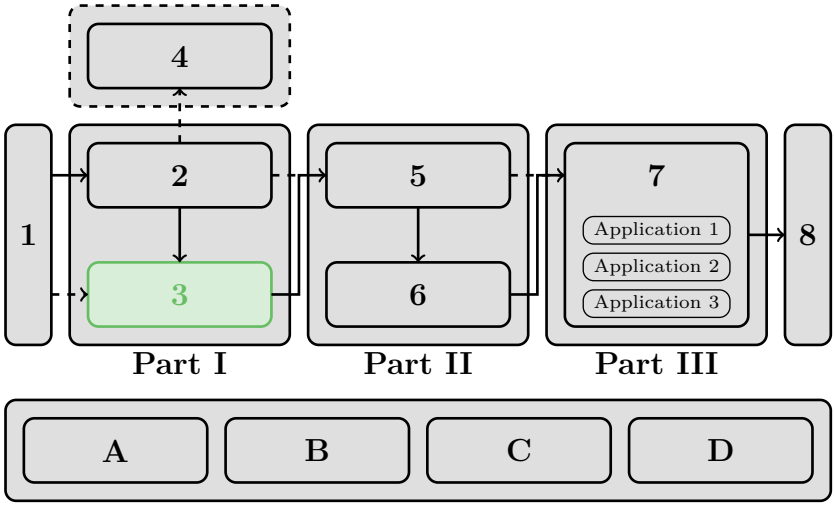
Real algebraic geometry

Although we compute in this text all the complex solutions of the system, some applications are only concerned with the real solutions. The number of real solutions can be much smaller than the number of complex solutions. Real solutions of polynomial systems and specific real solution methods are studied in the field of real algebraic geometry [38, 226]. Finding only the real solutions without computing all complex solutions first (the approach taken in this dissertation) is a hard problem that is still largely open. One reason is the fact that field of complex numbers $\mathbb{C}$ is algebraically closed, while the field of real numbers $\mathbb{R}$ is not. In fact, $\mathbb{C}$ is the algebraic closure $\overline{\mathbb{R}}$ of $\mathbb{R}$, which means that $\mathbb{C}$ is the smallest of all fields K containing $\mathbb{R}$ such that every nonconstant polynomial in $\mathbb{R}[x_1, \ldots, x_n]$ has a solution in K. One possible set of methods to compute the solutions in $\mathbb{R}^n$ are the above-mentioned subdivision methods.

Eigenvalue-Finding via the Block Macaulay Matrix

We extend the traditional (scalar) Macaulay matrix from multivariate polynomial system solving to the block Macaulay matrix in the multiparameter eigenvalue problem setting. The introduction of the block Macaulay matrix enables a natural methodology to deal with (polynomial) rectangular multiparameter eigenvalue problems, but can also be used to solve one-parameter eigenvalue problems and (linear) square multiparameter eigenvalue problems.

We develop, in this chapter, two approaches that use the block Macaulay matrix to solve rectangular multiparameter eigenvalue problems. On the one hand, a multidimensional realization problem in the right null space of the block Macaulay matrix constructed from the coefficient matrices of a rectangular multiparameter eigenvalue problem results in a standard eigenvalue problem. The eigenvalues and eigenvectors of that standard eigenvalue problem yield the solutions of that rectangular multiparameter eigenvalue problem. On the other hand, we propose a complementary approach to solve rectangular multiparameter eigenvalue problems that considers the data in the column space of block Macaulay matrix directly, avoiding the computation of a numerical basis matrix of the right null space. This chapter includes an in-depth discussion of the different aspects of multiparameter eigenvalue-finding via the block Macaulay matrix.



Contributions. We try to create an enhanced understanding of the rectangular multiparameter eigenvalue problem and introduce the block Macaulay matrix. By levering two of the fundamental subspaces of the block Macaulay matrix constructed from its coefficient matrices, we propose two complementary approaches to solve a rectangular multiparameter eigenvalue problem.

Related article. This chapter is an adaptation of [261]. The candidate was the main author of the original article, developed the theoretical contributions, and implemented the accompanying code and experiments. This chapter differs in several parts from the original article, in order to give more complete and didactic overview and avoid repetition with Chapter 2.

Outline. Firstly, we introduce the problem and sketch the current state-of-the-art in Section 3.1. Section 3.2, subsequently, contains the definition and most important properties of the rectangular multiparameter eigenvalue problem. Afterwards, we rigorously define the block Macaulay matrix and highlight two of its four fundamental subspaces in Section 3.3. The right null space of this block Macaulay matrix has a special (backward) block multi-shift-invariant structure, which allows us to find the eigenvalues of the problem via a multidimensional realization problem in that right null space. We develop the null space based algorithm to solve rectangular multiparameter eigenvalue problems in Section 3.4 and translate it to the column space in Section 3.5. Finally, in Section 3.6, we conclude this chapter and suggest interesting future research paths.

3.1 Introduction

Many natural and scientific phenomena exhibit intrinsic system dynamics that can be captured in a one-parameter eigenvalue problem. The eigenvalues and eigenvectors that correspond to those phenomena describe the proper[1] evolution of the system dynamics along the eigenvector directions. Eigenvalue problems prevail often in nature and science, in problems where the understanding of the underlying system's behavior is crucial. More particularly, eigenvalues form the cornerstone of systems theory: They characterize stability, controllability, and observability of linear time-invariant (LTI) dynamical systems [128], arise in the steady-state solutions to linear-quadratic regulators and Kalman filtering problems [129], solve model order reduction problems, like modal approximation [8, 208], etc. Section 7.2.2 provides other examples of applications in which eigenvalue and eigenvectors problems play an important role. For some phenomena, however, a single spectral parameter does not capture the system dynamics entirely and multiple spectral parameters, or tuples of eigenvalues, come into the picture.

Multiparameter spectral theory generalizes the classic spectral theory of linear operators to multiple linear operators linked by multiple spectral parameters [217]. Historically, multiparameter spectral theory has its roots in the classical problem of solving boundary-value problems for partial differential equations by the method of separation of variables [13, 96, 200, 217]. For example, the vibration problem of an elliptic membrane in two elliptic coordinates, i.e., the two-dimensional Helmholtz equation, leads to the study of a pair of ordinary differential equations, both of which share two spectral parameters. This corresponds to a two-parameter spectral problem [200, 217]. The presence of multiple spectral parameters links the evolution of the different ordinary differential equations obtained from the separation of variables in an elementary fashion. Other boundary-value problems give rise to more involved multiparameter spectral problems.

One approach of solving these multiparameter spectral problems numerically is by considering the related algebraic multiparameter eigenvalue problem (MEP), for example, obtained after a discretization or collocation method [96, 200]. The obtained problems are so-called **square multiparameter eigenvalue problems**: square MEPs (sometimes called standard MEPs) are systems of multiple multiparameter square matrix pencils [118]. Some of the early work with respect to this multiparameter formulation are by Atkinson [12], Atkinson and Mingarelli [13], Carmichael [54–56], and Volkmer [267].

Recently, we have shown within our research group that the least-squares identification of LTI dynamical systems is, in essence, also an MEP, more specifically, a **rectangular multiparameter eigenvalue problem** [70, 71,

[1] The prefix *eigen-* is adopted from the German word *eigen*, which means *proper* and was presumably first coined by Hilbert [114]. The use of the German prefix *eigen* may seem odd in the English language, but is now standard in the English literature to refer to the solutions of a matrix pencil. However, and this might sound surprising to some readers, its use in the United States has not always been so common. In fact, over the past two centuries the words *proper*, *latent*, *characteristic*, *secular*, and *singular* have all been used as alternatives for the prefix *eigen* [105, 246].

259]. Also the $\mathcal{H}_2$-norm optimal model order reduction problem can be recast as a rectangular MEP [3, 6, 148]. These rectangular problems differ from square MEPs in that they consist of a single rectangular matrix pencil. The first detailed studies[2] of the rectangular MEP were by Khazanov [134, 136], who rigorously analyzed the spectrum of polynomial multiparameter problems, and by Shapiro and Shapiro [216], who looked at the multivariate linear matrix pencil in the context of the Heine–Stieltjes spectral problem.

Multiparameter eigenvalue computing

The literature about one-parameter eigenvalue problems is quite expanded. There exist many techniques to solve standard eigenvalue problems (SEPs) and generalized eigenvalue problems (GEPs), with adaptations to tackle very large matrices when there is structure or sparsity involved [213, 270] or the matrices are not square [110, 281]. Polynomial eigenvalue problems (PEPs) are usually linearized into larger GEPs [113, 161, 162, 245] and the resulting matrix pencils are solved via one of the many available, efficient SEP or GEP solvers. However, despite their applicability and the natural relation to one-parameter eigenvalue problems, MEPs have not yet been widely diffused among the general scientific community. This chapter tries to alleviate this hiatus. Below, we discuss solution approaches for both square and rectangular problems. A more elaborate overview can be found in the "historical and bibliographical notes" of this chapter.

Square multiparameter eigenvalue problems

Square MEPs are typically solved via simultaneous triangularization of the associated system of coupled GEPs [117, 126, 198, 218]. This approach works for any number of spectral parameters and retrieves all the solutions, but is limited by the size of the matrices in the system of coupled GEPs. Also iterative nonlinear optimization algorithms can be used to retrieve one (or some) of the solutions, but these optimization approaches are heuristic (they depend on an initial guess) and may have issues with convergence. More efficient algorithms, like homotopy continuation methods and subspace techniques (e.g., Jacobi–Davidson or Arnoldi), have been developed in the last two decades to counter issues with scalability and convergence.

Furthermore, it is even possible to convert quite a few square MEPs into equivalent rectangular MEPs (Section 3.2.2). In that sense, the two block Macaulay matrix approaches described in this chapter also supplement the set of existing techniques to solve square problems.

Rectangular multiparameter eigenvalue problems

While they provided a theoretical analysis of the problem and established an initial bound on the number of affine solutions, Shapiro and Shapiro [216] did

[2]Before that, a more general problem with $m \times n$ matrices, where $m \geq n$, appeared under the name *eigentuple-eigenvector problem* in [35, 36] and as a rank-reducing perturbation problem in [273].

not present any numerical algorithms. Khazanov [132, 133, 137] has proposed iterative algorithms (with and without linearization step) to determine the eigenvalues of multivariate polynomial matrix pencils. The iterative algorithms in [35–37, 273] could also be applied to rectangular MEPs. In our work [69–71, 261, 262], we have introduced the block Macaulay matrix, which allows us to solve rectangular MEPs via numerical linear algebra techniques.

Very recently, two additional approaches have emerged to tackle linear rectangular MEPs. These approaches transform the problem into linear square MEPs [6, 118]. These transformations (Section 3.2.2) allows us to also use the (efficient) algorithms developed for square MEPs [118].

A numerical block Macaulay matrix perspective

We focus in this chapter on rectangular MEPs. We tackle the multiparameter eigenvalue-finding problem via the block Macaulay matrix, as we set out in Section 1.3:

> *We present numerical linear algebra approaches to find the tuples of eigenvalues of a rectangular multiparameter eigenvalue problem, where we assume that the solution set is zero-dimensional. These approaches aim to compute satisfactory approximations of the coordinates of the eigenvalues in the affine (complex) space.*

The central object of this chapter is the block Macaulay matrix, a sparse and structured matrix that is constructed from the coefficient matrices of the rectangular MEP. In [71, 259], we have introduced this matrix for the first time in order to solve the rectangular MEPs that we obtained in a system identification context. The complementarity of the right null space and column space has enabled two multiparameter eigenvalue-finding approaches that use a different fundamental subspace of the block Macaulay matrix. This observation stems from a similar complementarity in multivariate polynomial system solving[3], in which the right null space and column space of the traditional (scalar) Macaulay matrix both give rise to a multivariate root-finding algorithm [258].

Motivational example. Where we consider in Chapter 2 a reformulation of the globally optimal autoregressive moving-average (ARMA) model identification problem as a system of multivariate polynomial equations, the underlying optimization problem (1.2) can also be rephrased as a rectangular MEP (as we explain in Section 7.5). For example, if we consider a sequence of $N = 4$ output samples $y_k \in \mathbb{R}$ and a first-order ARMA model, then we can rewrite (1.2) as a quadratic rectangular two-parameter eigenvalue problem

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \big(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{01}\lambda_2 + \boldsymbol{A}_{02}\lambda_2^2\big)\boldsymbol{z} = \boldsymbol{0}, \tag{3.1}$$

where the $n = 2$ spectral parameters correspond to $\boldsymbol{\lambda} = (\alpha, \gamma)$ and the coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}}$ are 11×10 real matrices. We can use the block

[3] MEPs are essentially disguised systems of multivariate polynomial equations with some variables that only appear "linearly". In that sense, the block Macaulay matrix approach is related to other sparse resultant matrices, like Newton matrices [85, 87].

Macaulay matrix approaches developed in this chapter to solve this problem and obtain the globally optimal ARMA model parameters.

3.2 Multiparameter eigenvalue problems

In linear algebra, a matrix-valued function with complex coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{C}^{k\times l}$ for some (multivariate) variable $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n) \in \mathbb{C}^n$,

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) = \sum_{\mathcal{W}} \boldsymbol{A}_{\boldsymbol{\omega}} \boldsymbol{\lambda}^{\boldsymbol{\omega}}, \tag{3.2}$$

where the summation runs over all the multi-indices $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \in \mathbb{N}^n$ in the set $\mathcal{W} = \{\boldsymbol{\omega} : \boldsymbol{A}_{\boldsymbol{\omega}} \neq \boldsymbol{0}\}$, is called a **(multivariate) matrix pencil**. The multi-indices $\boldsymbol{\omega}$ label the powers of the eigenvalues in the monomials $\boldsymbol{\lambda}^{\boldsymbol{\omega}} = \prod_{i=1}^{n} \lambda_i^{\omega_i} = \lambda_1^{\omega_1} \cdots \lambda_n^{\omega_n}$ and index the associated coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} = \boldsymbol{A}_{(\omega_1,\ldots,\omega_n)}$. The total degree of a monomial is equal to the sum of its powers, denoted by $|\boldsymbol{\omega}| = \sum_{i=1}^{n} \omega_i$, and the highest total degree of all the monomials is the degree of a matrix pencil, denoted by $d_{\max}$ or $\deg(\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}))$. To keep the notation unambiguous, we use the graded inverse lexicographic (GRINVLEX) ordering to order different (multivariate) monomials (Definition 2.6). However, the remainder of this chapter remains valid for any graded monomial ordering.

Example 3.1. A multi-index $\boldsymbol{\omega} = (0, 2, 5)$ labels the monomial $\lambda_2^2\lambda_3^5$ (with total degree 7) and indexes the associated coefficient matrix $\boldsymbol{A}_{025}$ in the matrix pencil (3.2).

The corresponding problem, in which we look for the values of the (multivariate) variable $\boldsymbol{\lambda}$ that make one (or multiple) matrix pencil(s) column rank deficient is called the MEP. In that sense, the scalar variables $\lambda_i \in \mathbb{C}$, for $i = 1, \ldots, n$, are called the **eigenvalue parameters** of the MEP, and we organize them together in tuples of eigenvalues $\boldsymbol{\lambda}$. Notice that the definition of the rectangular MEP uses the important assumption that the rectangular matrix pencil has full normal rank, an explanation of which is given in the following definition. When the normal rank is not full, the situation becomes more difficult to analyze and solve [135].

Definition 3.1. The **normal rank** of a matrix pencil $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ with rectangular coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{C}^{k\times l}$, $k \geq l$ is defined as

$$\operatorname{nrank}(\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})) = \max_{\boldsymbol{\lambda}} \operatorname{rank}(\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})). \tag{3.3}$$

The matrix pencil has **full normal rank** when $\operatorname{nrank}(\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})) = l$.

Definition 3.2. Given coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{C}^{k\times l}$ (with $k \geq l + n - 1$) that lead to a full normal rank matrix pencil $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$, the **rectangular multiparameter eigenvalue problem (rectangular MEP)** consists in finding all n-tuples $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n) \in \mathbb{C}^n$ and corresponding vectors $\boldsymbol{z} \in \mathbb{C}^{l\times 1} \setminus \{\boldsymbol{0}\}$, so that

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \left(\sum_{\mathcal{W}} \boldsymbol{A}_{\boldsymbol{\omega}} \boldsymbol{\lambda}^{\boldsymbol{\omega}}\right)\boldsymbol{z} = \boldsymbol{0}, \tag{3.4}$$

where the summation runs over all the multi-indices in the set $\mathcal{W}$. The n-tuples $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ and (non-zero) vectors $\boldsymbol{z}$ are the eigenvalues and eigenvectors of the rectangular MEP, respectively.

The set $\mathcal{W} = \{\boldsymbol{\omega} : \boldsymbol{A}_{\boldsymbol{\omega}} \neq \boldsymbol{0}\} \subset \mathbb{N}^n$ contains all multi-indices $\boldsymbol{\omega}$ present in the matrix pencil (i.e., all the multi-indices with non-vanishing coefficient matrix $\boldsymbol{A}_{\boldsymbol{\omega}}$). Therefore, $\mathcal{W} = \operatorname{supp}(\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}))$ is sometimes called the **support** of the matrix pencil (analogue to the definition of the support of a polynomial).

Another way to phrase the (rectangular[4]) MEP is by considering the eigenvalues for which the rank drops below the normal rank of the matrix pencil.

Corollary. Given the problem in (3.4), the tuple $\boldsymbol{\lambda} \in \mathbb{C}^n$ is an eigenvalue if

$$\operatorname{rank}(\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})) < \operatorname{nrank}(\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})). \tag{3.5}$$

Given Definition 3.2, a matrix pencil can be seen as an n-variate matrix-valued function

$$\boldsymbol{\mathcal{M}} : \mathbb{C}^n \to \mathbb{C}^{k\times l} : \boldsymbol{\lambda} \mapsto \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) = \sum_{\mathcal{W}} \boldsymbol{A}_{\boldsymbol{\omega}} \boldsymbol{\lambda}^{\boldsymbol{\omega}} \tag{3.6}$$

that maps a point from the affine space $\mathbb{C}^n$ onto a matrix value. In words: given a point $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{C}^n$ in the affine space (Definition 2.7), a matrix pencil yields a rectangular matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{a})$ in $\mathbb{C}^{k\times l}$. We can define a map

$$\mathcal{S} : \mathbb{C}^n \to \mathbb{S}^{k\times l} : \boldsymbol{\lambda} \mapsto \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}), \tag{3.7}$$

where $\mathbb{S}^{k\times l}$ is the subspace of rank deficient matrices in $\mathbb{C}^{k\times l}$. We are interested in the pre-image of $\mathbb{S}^{k\times l}$ under this map,

$$\mathcal{S}^{-1}(\boldsymbol{0}) = \{\boldsymbol{\lambda} \in \mathbb{C}^n : \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) \text{ is rank deficient}\}. \tag{3.8}$$

Therefore, $\mathcal{S}^{-1}(\boldsymbol{0})$ is called the set of eigenvalues of the MEP defined by this matrix polynomial. In this context, by multiparameter eigenvalue-finding, we mean computing $\mathcal{S}^{-1}(\boldsymbol{0})$. Because it is in general not possible to compute

[4]In the remainder of this chapter, we no longer mention the qualification *rectangular* explicitly. We always consider rectangular problems, except when denoted otherwise (for example, in Section 3.2.2, when comparing with the square problem formulation).

Table 3.1. Four different types of (multiparameter) eigenvalue problems, organized according to the structure of the monomials in the (multivariate) matrix pencil $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$.

	linear	**polynomial**
single eigenvalue ($n = 1$)	Type I	Type II
	$\{1, \lambda\}$	λ^{ω}
	SEP/GEP	PEP
multiple eigenvalues ($n > 1$)	Type III	Type IV
	λ_i	$\boldsymbol{\lambda}^{\boldsymbol{\omega}} = \prod_{i=1}^{n} \lambda_i^{\omega_i}$
	linear MEP	polynomial MEP

$\mathcal{S}^{-1}(\mathbf{0})$ exactly, we mean computing satisfactory approximations of the coordinates of the solutions in the affine space $\mathbb{C}^n$ via numerical algorithms. Furthermore, we assume that the set of solutions is zero-dimensional (i.e., finite), which results in the size condition on the coefficient matrices. The size condition on the coefficient matrices is a necessary (but not a sufficient) condition in order for the MEP to have a zero-dimensional solution set: there are k equations and one non-triviality constraint on $\boldsymbol{z}$ (e.g., $\|\boldsymbol{z}\|_2 = 1$) in $l+n$ unknowns (l elements in the eigenvectors $\boldsymbol{z}$ and n eigenvalues), thus $k + 1 \geq l + n$. The problem of solving the MEP, in which we look for the eigenpairs $(\boldsymbol{\lambda}, \boldsymbol{z})$ that solve (3.4), is closely related to the multiparameter eigenvalue-finding problem. Given the eigenvalues, it is possible to also compute an associated eigenvector for every obtained eigenvalue.

Now, we give an overview of the different MEPs and mention the linearization of a polynomial MEP to a linear one (Section 3.2.1). We relate the rectangular problem with the square MEP and show to transform one problem formulation into the other one (Section 3.2.2). The number of points in $\mathcal{S}^{-1}(\mathbf{0})$, i.e., the number of solutions of a MEP, is discussed afterwards (Section 2.2.3).

3.2.1 Different matrix pencils and linearizations

When the matrix pencil is a univariate matrix-valued polynomial in λ, we recognize the well-known one-parameter eigenvalue problem. Particular cases are the SEP and GEP for $d_{\max} = 1$ and PEP for $d_{\max} > 1$. Multiparameter eigenvalue problems have multivariate matrix pencils that involve multiple eigenvalue parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ instead of single eigenvalues λ. Also in the multiparameter case, we differentiate linear ($d_{\max} = 1$) and polynomial ($d_{\max} > 1$) problems. Based on the structure of the monomials in the (multivariate) matrix pencil $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$, we can organize (multiparameter) eigenvalue problems according to a two-dimensional grid: single eigenvalues versus multiple eigenvalues and linear versus polynomial (Table 3.1).

Example 3.2 (Type I – SEP/GEP). There are two types of linear one-parameter eigenvalue problems, namely the SEP $\boldsymbol{A}_0\boldsymbol{z} = \boldsymbol{z}\lambda$, or $(\boldsymbol{A}_0 - \boldsymbol{I}\lambda)\boldsymbol{z} = \boldsymbol{0}$, and the GEP $\boldsymbol{A}_0\boldsymbol{z} = \boldsymbol{A}_1\boldsymbol{z}\lambda$, or $(\boldsymbol{A}_0 - \boldsymbol{A}_1\lambda)\boldsymbol{z} = 0$.

Example 3.3 (Type II – PEP). PEPs of degree $d_{\max} > 1$ fit in the grid of Table 3.1 under Type II. The monomials have a univariate power structure λ^i,

$$\left(\sum_{\mathcal{W}} \boldsymbol{A}_\omega \lambda^\omega\right)\boldsymbol{z} = \left(\sum_{i=0}^{d_{\max}} \boldsymbol{A}_i \lambda^i\right)\boldsymbol{z} = \boldsymbol{0}, \tag{3.9}$$

which means that the single-index ω runs from 0 to $d_{\max}$. For example, a polynomial eigenvalue problem of degree $d_{\max} = 4$ has five coefficient matrices $\boldsymbol{A}_i \in \mathbb{C}^{k\times l}$ $(k \geq l)$ and is given by

$$\left(\boldsymbol{A}_0 + \boldsymbol{A}_1\lambda + \boldsymbol{A}_2\lambda^2 + \boldsymbol{A}_3\lambda^3 + \boldsymbol{A}_4\lambda^4\right)\boldsymbol{z} = \boldsymbol{0}. \tag{3.10}$$

Example 3.4 (Type III – linear MEP). Consider the linear two-parameter eigenvalue problem ($d_{\max} = 1$),

$$(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{01}\lambda_2)\boldsymbol{z} = \boldsymbol{0}, \tag{3.11}$$

with coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{R}^{3\times 2}$, for example,

$$\boldsymbol{A}_{00} = \begin{bmatrix} 2 & 6 \\ 4 & 5 \\ 0 & 1 \end{bmatrix}, \boldsymbol{A}_{10} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \text{ and } \boldsymbol{A}_{01} = \begin{bmatrix} 4 & 2 \\ 0 & 8 \\ 1 & 1 \end{bmatrix},$$

This MEP has $m_a = 3$ affine solutions (Table 3.2).

Example 3.5 (Type IV – polynomial MEP). Finally, consider a quadratic two-parameter eigenvalue problem ($d_{\max} = 2$) with only four monomials,

$$\left(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{11}\lambda_1\lambda_2 + \boldsymbol{A}_{02}\lambda_2^2\right)\boldsymbol{z} = \boldsymbol{0}, \tag{3.12}$$

which has four coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{R}^{3\times 2}$,

$$\boldsymbol{A}_{00} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 3 & 4 \end{bmatrix}, \boldsymbol{A}_{10} = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ 1 & 3 \end{bmatrix}, \boldsymbol{A}_{11} = \begin{bmatrix} 3 & 4 \\ 2 & 1 \\ 0 & 1 \end{bmatrix}, \text{ and } \boldsymbol{A}_{02} = \begin{bmatrix} 1 & 2 \\ 4 & 2 \\ 2 & 1 \end{bmatrix}.$$

This MEP has $m_b = 12$ solutions, $m_a = 9$ of which are affine (Table 3.3). Notice that the summation in (3.12) only runs over multi-indices $\boldsymbol{\omega} = (0,0), (1,0), (1,1)$, and $(0,2)$. Hence, the support of this matrix pencil does not contain all combinations of indices.

Similar to linearizations of PEPs [113, 161, 162, 245] and polynomial square MEPs [120, 186], the rectangular matrix pencil can also be linearized. In Example 3.6, we show how to linearize a quadratic two-parameter eigenvalue problem without taking into account any additional structure or sparsity pattern in the coefficient matrices. Other, more compact, linearizations are possible, for example, when certain coefficient matrices are not present in the support [118].

Example 3.6. If we consider a quadratic two-parameter eigenvalue problem,

$$\left(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{01}\lambda_2 + \boldsymbol{A}_{20}\lambda_1^2 + \boldsymbol{A}_{11}\lambda_1\lambda_2 + \boldsymbol{A}_{02}\lambda_2^2\right)\boldsymbol{z} = \boldsymbol{0}, \tag{3.13}$$

with $k \times l$ coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{C}^{k\times l}$, then we can linearize this problem into a $(k + 2l) \times 3l$ linear two-parameter eigenvalue problem,

$$\left(\widehat{\boldsymbol{A}}_{00} + \widehat{\boldsymbol{A}}_{10}\lambda_1 + \widehat{\boldsymbol{A}}_{01}\lambda_2\right)\begin{bmatrix}\boldsymbol{z}\\ \lambda_1\boldsymbol{z}\\ \lambda_2\boldsymbol{z}\end{bmatrix} = \boldsymbol{0}, \tag{3.14}$$

with "new" coefficient matrices $\widehat{\boldsymbol{A}} \in \mathbb{C}^{(k+2l)\times 3l}$,

$$\widehat{\boldsymbol{A}}_{00} = \begin{bmatrix}\boldsymbol{A}_{00} & \boldsymbol{A}_{10} & \boldsymbol{A}_{01}\\ \boldsymbol{0} & -\boldsymbol{I}_l & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{I}_l\end{bmatrix}, \widehat{\boldsymbol{A}}_{10} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{A}_{20} & \boldsymbol{A}_{11}\\ \boldsymbol{I}_l & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\end{bmatrix},$$
$$\text{and } \widehat{\boldsymbol{A}}_{01} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{02}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{I}_l & \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}.$$

Note that the block Macaulay matrix introduced in Section 3.3 implicitly linearizes the generating MEP. Therefore, it allows us to tackle polynomial MEPs directly, without explicit linearization step.

3.2.2 Relations with square problem formulation

Recently, it has been shown that square and rectangular multiparameter eigenvalue problem formulations are related. A good understanding of this relation is currently limited to linear problems. Of course, because of the existing linearization techniques, it is also possible to rephrase a polynomial square MEP as a linear rectangular MEP. Before showing these relations, we first need to give a short summary about linear square MEP.

Definition 3.3. Given square coefficient matrices $\boldsymbol{B}_{ij} \in \mathbb{C}^{l_i\times l_i}$, the **linear square multiparameter eigenvalue problem (linear square MEP)** consists in finding all n-tuples $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n) \in \mathbb{C}^n$ and corresponding vectors $\boldsymbol{x}_i \in \mathbb{C}^{l_i\times 1} \setminus \{\boldsymbol{0}\}$, so that

$$\boldsymbol{\Delta}_i(\boldsymbol{\lambda})\boldsymbol{x}_i = \left(\boldsymbol{B}_{i0} + \sum_{j=1}^{n}\boldsymbol{B}_{ij}\lambda_j\right)\boldsymbol{x}_i = \boldsymbol{0}, \tag{3.15}$$

for $i = 1, \ldots, n$. The n-tuples $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_n)$ and (non-zero) vectors $\boldsymbol{z} = \boldsymbol{x}_1 \otimes \cdots \otimes \boldsymbol{x}_n \in \mathbb{C}^{l \times 1} \setminus \{\mathbf{0}\}$ (with $l = l_1 \cdots l_n$) are the eigenvalues and eigenvectors of the linear square MEP, respectively.

In the generic case, (3.15) has $l = l_1 \cdots l_n$ solutions, which are the common roots of the system of n multivariate characteristic polynomial equations $\det(\boldsymbol{\mathcal{N}}_i(\boldsymbol{\lambda})) = 0$, for $i = 1, \ldots, n$.

Example 3.7. On the first page of his book, Volkmer [267] used the following linear square two-parameter eigenvalue problem to introduce several aspects of multiparameter spectral theory:

$$\begin{cases} \boldsymbol{\mathcal{N}}_1(\boldsymbol{\lambda})\boldsymbol{x}_1 = (\boldsymbol{B}_{10} + \boldsymbol{B}_{11}\lambda_1 + \boldsymbol{B}_{12}\lambda_2)\boldsymbol{x}_1 = \mathbf{0}, \\ \boldsymbol{\mathcal{N}}_2(\boldsymbol{\lambda})\boldsymbol{x}_2 = (\boldsymbol{B}_{20} + \boldsymbol{B}_{21}\lambda_1 + \boldsymbol{B}_{22}\lambda_2)\boldsymbol{x}_2 = \mathbf{0}, \end{cases} \tag{3.16}$$

with square coefficient matrices

$$\boldsymbol{B}_{10} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \boldsymbol{B}_{11} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{B}_{12} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$
$$\boldsymbol{B}_{20} = \begin{bmatrix} 20 & 0 \\ 0 & 0 \end{bmatrix}, \boldsymbol{B}_{21} = \begin{bmatrix} 0 & \sqrt{3} \\ \sqrt{3} & 0 \end{bmatrix}, \text{ and } \boldsymbol{B}_{22} = \begin{bmatrix} 7 & 0 \\ 0 & 1 \end{bmatrix}.$$

An important family of matrices constructed from the coefficient matrices of the linear square MEP (3.15) are the Kronecker operator determinants $\boldsymbol{\Delta}_i \in \mathbb{C}^{l \times l}$. These matrices are given by

$$\boldsymbol{\Delta}_0 = \begin{vmatrix} \boldsymbol{B}_{11} & \cdots & \boldsymbol{B}_{1n} \\ \vdots & & \vdots \\ \boldsymbol{B}_{n1} & \cdots & \boldsymbol{B}_{nn} \end{vmatrix}_{\otimes} \tag{3.17}$$
$$= \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \boldsymbol{B}_{1\sigma_1} \otimes \cdots \otimes \boldsymbol{B}_{n\sigma_n}, \tag{3.18}$$

and, for $i = 1, \ldots, n$,

$$\boldsymbol{\Delta}_i = -\begin{vmatrix} \boldsymbol{B}_{11} & \cdots & \boldsymbol{B}_{1,i-1} & \boldsymbol{B}_{10} & \boldsymbol{B}_{1,i+1} & \cdots & \boldsymbol{B}_{1n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \boldsymbol{B}_{n1} & \cdots & \boldsymbol{B}_{n,i-1} & \boldsymbol{B}_{n0} & \boldsymbol{B}_{n,i+1} & \cdots & \boldsymbol{B}_{nn} \end{vmatrix}_{\otimes}, \tag{3.19}$$
$$= -\sum_{\substack{\sigma \in S_n \\ \sigma_i \to 0}} \operatorname{sgn}(\sigma) \boldsymbol{B}_{1\sigma_1} \otimes \cdots \otimes \boldsymbol{B}_{n\sigma_n}, \tag{3.20}$$

where $|\cdot|_{\otimes}$ corresponds to the determinant operation in which every element is a coefficient matrix and every multiplication is replaced by a Kronecker product. When $\boldsymbol{\Delta}_0$ is a singular matrix, the linear square MEP is called singular. A linear square MEP with generic matrices is nonsingular.

Theorem 3.1. A linear square MEP, as defined in Definition 3.3, is equivalent with its associated system of coupled GEPs,

$$\begin{cases} \boldsymbol{\Delta}_1 \boldsymbol{z} = \lambda_1 \boldsymbol{\Delta}_0 \boldsymbol{z}, \\ \quad \vdots \\ \boldsymbol{\Delta}_n \boldsymbol{z} = \lambda_n \boldsymbol{\Delta}_0 \boldsymbol{z}, \end{cases} \tag{3.21}$$

when the problem is nonsingular, i.e., when $\det(\boldsymbol{\Delta}_0) \neq 0$.

Proof. A proof and explanation of this theorem can be found in [11]. □

When the problem is nonsingular, the matrices $\boldsymbol{\Delta}_0^{-1}\boldsymbol{\Delta}_i$, for $i = 1, \ldots, n$, commute and the eigenvalues of (3.21) and (3.15) agree.

Remark 3.1. The situation is more complicated when the problem is singular. In this case, the eigenvalues of (3.15) correspond to the common regular part of (3.21), see, for more information, [139, 187]. There are several applications in which singular linear square MEPs appear (sometimes after a linearization step), e.g., [120, 139, 187].

Now, we have all the prerequisites to consider the translation of a square problem into a rectangular one via the Kronecker operator determinants, as proposed by Vermeersch and De Moor [261] (Section 3.2.2.1), and in the other direction via the compressed Kronecker operator products or sketching, as proposed by Alsubaie [6] and Hochstenbach et al. [118] (Section 3.2.2.2).

3.2.2.1 From square to rectangular problems

As shown by Atkinson [12], a regular linear square MEP is equivalent to its associated system of coupled GEPs (Theorem 3.1):

$$(3.15) \Leftrightarrow \begin{cases} \boldsymbol{\Delta}_1 \boldsymbol{z} = \lambda_1 \boldsymbol{\Delta}_0 \boldsymbol{z}, \\ \quad \vdots \\ \boldsymbol{\Delta}_n \boldsymbol{z} = \lambda_n \boldsymbol{\Delta}_0 \boldsymbol{z}. \end{cases} \tag{3.22}$$

Via this system of associated coupled GEPs, it is possible to transform the linear square MEP into its equivalent rectangular form:

$$\left(\begin{bmatrix} \boldsymbol{\Delta}_1 \\ \vdots \\ \boldsymbol{\Delta}_n \end{bmatrix} - \begin{bmatrix} \boldsymbol{\Delta}_0 \\ \vdots \\ \boldsymbol{0} \end{bmatrix} \lambda_1 - \cdots - \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{\Delta}_0 \end{bmatrix} \lambda_n \right) \boldsymbol{z} = \boldsymbol{0}. \tag{3.23}$$

This transformation leads to a linear rectangular MEP where the number of rows k is strictly larger than the minimum $l + n - 1$. Since the Kronecker

operator determinants $\boldsymbol{\Delta}_i$ typically result in large coefficient matrices, this approach should only be applied to small problems. Furthermore, we need to be careful in the case of singular problems (i.e., when $\boldsymbol{\Delta}_0$ is a singular matrix), where the equivalence between the square problem and the associated system of coupled GEPs is not straightforward, as discussed in Remark 3.1.

Example 3.8. We illustrate this relation via the linear square two-parameter eigenvalue problem in Example 3.7. The associated system of coupled GEPs is given by

$$\begin{cases} \boldsymbol{\Delta}_1 \boldsymbol{z} = \boldsymbol{\Delta}_0 \lambda_1 \boldsymbol{z}, \\ \boldsymbol{\Delta}_2 \boldsymbol{z} = \boldsymbol{\Delta}_0 \lambda_2 \boldsymbol{z}, \end{cases} \tag{3.24}$$

where the Kronecker operator determinants $\boldsymbol{\Delta}_i$ are 6×6 matrices:

$$\boldsymbol{\Delta}_0 = \boldsymbol{B}_{11} \otimes \boldsymbol{B}_{22} - \boldsymbol{B}_{12} \otimes \boldsymbol{B}_{21}, \tag{3.25}$$
$$\boldsymbol{\Delta}_1 = \boldsymbol{B}_{12} \otimes \boldsymbol{B}_{20} - \boldsymbol{B}_{10} \otimes \boldsymbol{B}_{22}, \tag{3.26}$$
$$\boldsymbol{\Delta}_2 = \boldsymbol{B}_{10} \otimes \boldsymbol{B}_{21} - \boldsymbol{B}_{11} \otimes \boldsymbol{B}_{20}. \tag{3.27}$$

The equivalent linear rectangular MEP in (3.23),

$$\left(\begin{bmatrix} \boldsymbol{\Delta}_1 \\ \boldsymbol{\Delta}_1 \end{bmatrix} - \begin{bmatrix} \boldsymbol{\Delta}_0 \\ \boldsymbol{0} \end{bmatrix} \lambda_1 - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\Delta}_0 \end{bmatrix} \lambda_2 \right) \boldsymbol{z} = \boldsymbol{0}, \tag{3.28}$$

has the same solutions as (3.16) and can be solved via one of the block Macaulay matrix algorithms developed in this chapter.

3.2.2.2 From rectangular to square problems

It is also possible to transform a linear rectangular[5] MEP into a related linear square problem. There are two techniques to obtain this related linear square MEP, via sketching [118] or via compressed Kronecker products [6]. The former retrieves the coefficient matrices of a linear square MEP, which allows the use of iterative solvers, while the latter solves the associated system of coupled GEPs. The latter is thus appropriate for problems small enough such that these GEPs can be solved via a direct solver.

Randomized or deterministic sketching. The idea behind sketching is simple, yet very powerful: via n probabilistic or deterministic projections, square matrix pencils are obtained that form together a related linear square MEP:

$$(3.4) \Rightarrow \begin{cases} \boldsymbol{\mathcal{N}}_1(\boldsymbol{\lambda}) \boldsymbol{x}_1 = \boldsymbol{P}_1 (\boldsymbol{A}_{0\cdots0} + \boldsymbol{A}_{1\cdots0}\lambda_1 + \cdots + \boldsymbol{A}_{0\cdots1}\lambda_n) \boldsymbol{x}_1 = \boldsymbol{0}, \\ \qquad\qquad \vdots \\ \boldsymbol{\mathcal{N}}_n(\boldsymbol{\lambda}) \boldsymbol{x}_n = \boldsymbol{P}_n (\boldsymbol{A}_{0\cdots0} + \boldsymbol{A}_{1\cdots0}\lambda_1 + \cdots + \boldsymbol{A}_{0\cdots1}\lambda_n) \boldsymbol{x}_n = \boldsymbol{0}, \end{cases} \tag{3.29}$$

[5] Hochstenbach et al. [118] have also discussed quadratic rectangular MEPs, but in essence this extension simply corresponds to adding a linearization step before or after the transformation.

where the matrices $\boldsymbol{P}_i \in \mathbb{C}^{l\times k}$ can be random matrices, which is called randomized sketching, or row selection matrices, which is called deterministic sketching, chosen with as goal to obtain a nonsingular square problem. The implication in (3.29) only goes in one direction: the related linear square MEP has more solutions than the original linear rectangular MEP, due to the fact that the vectors $\boldsymbol{x}_i$ are not necessarily colinear (i.e., they do not correspond to the same eigenvector $\boldsymbol{z}$ in (3.4)). For a large n, only a modest portion of the obtained solutions is also a solution to the rectangular MEP (see, for example, the numerical experiment in [118]). Note that a lot of the sparsity gets lost when using random sketching matrices $\boldsymbol{P}_i$ and the sketching procedure ends up with more coefficient matrices than in the original rectangular MEP. However, this transformation allows us to employ the existing efficient (subspace) methods for linear square MEPs to tackle the original rectangular MEP. For more details, we refer the interested reader to [118].

Example 3.9. We illustrate this transformation via the linear rectangular two-parameter eigenvalue problem in (3.11), which has three affine solutions (Table 3.2). In order to obtain a related linear square MEP via sketching, we use two 2×3 row-selection matrices $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$,

$$\boldsymbol{P}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \text{ and } \boldsymbol{P}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.30}$$

to pre-multiply (3.11):

$$\begin{cases} \boldsymbol{\mathcal{N}}_1(\boldsymbol{\lambda})\boldsymbol{x}_1 = (\boldsymbol{P}_1\boldsymbol{A}_{00} + \boldsymbol{P}_1\boldsymbol{A}_{10}\lambda_1 + \boldsymbol{P}_1\boldsymbol{A}_{01}\lambda_2)\boldsymbol{x}_1 = \boldsymbol{0}, \\ \boldsymbol{\mathcal{N}}_2(\boldsymbol{\lambda})\boldsymbol{x}_2 = (\boldsymbol{P}_2\boldsymbol{A}_{00} + \boldsymbol{P}_2\boldsymbol{A}_{10}\lambda_1 + \boldsymbol{P}_2\boldsymbol{A}_{01}\lambda_2)\boldsymbol{x}_2 = \boldsymbol{0}, \end{cases} \tag{3.31}$$

The related linear square MEP has four solutions, three of them correspond to the solutions of the original rectangular MEP, as in Table 3.2. The spurious solution $(-2.2580, 1.0369)$ is a solution with two non-colinear vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and is, thus, not a solution of (3.11).

Compressed Kronecker products. Alsubaie [6] has derived a numerical method to solve linear rectangular MEPs by transforming the problem into a GEP that yields one of the spectral parameters. In that numerical approach to reduce a rectangular MEP into a square GEP lies the idea of the second technique, as highlighted by Hochstenbach et al. [118]. From the rectangular coefficient matrices, rectangular Kronecker operator determinants $\widetilde{\boldsymbol{\Delta}}_i \in \mathbb{C}^{k^n\times l^n}$ can be built that contain the same solutions as the original rectangular MEP, but also other solutions. A subsequent reduction step [6] of the associated system of coupled GEPs yields a problem with the same solutions as (3.4):

$$(3.4) \Rightarrow \begin{cases} \widetilde{\boldsymbol{\Delta}}_1\boldsymbol{w} = \lambda_1\widetilde{\boldsymbol{\Delta}}_0\boldsymbol{w}, \\ \quad\vdots \\ \widetilde{\boldsymbol{\Delta}}_n\boldsymbol{w} = \lambda_1\widetilde{\boldsymbol{\Delta}}_0\boldsymbol{w}, \end{cases} \Rightarrow \begin{cases} \widehat{\boldsymbol{\Delta}}_1\widehat{\boldsymbol{w}} = \lambda_1\widehat{\boldsymbol{\Delta}}_0\widehat{\boldsymbol{w}}, \\ \quad\vdots \\ \widehat{\boldsymbol{\Delta}}_n\widehat{\boldsymbol{w}} = \lambda_1\widehat{\boldsymbol{\Delta}}_0\widehat{\boldsymbol{w}}. \end{cases} \tag{3.32}$$

The reduction step in (3.32) uses the observation of Alsubaie [6] that the vectors $\boldsymbol{w} = \boldsymbol{z} \otimes \cdots \otimes \boldsymbol{z}$ span a subspace of dimension $\binom{k}{l}$ in $\mathbb{C}^l \otimes \cdots \otimes \mathbb{C}^l$. Indeed, if $\boldsymbol{z} = \begin{bmatrix} z_1 & \cdots & z_l \end{bmatrix}^{\mathrm{T}}$, then the elements of $\boldsymbol{w}$ are $w_{i_1 \cdots i_l} = z_{i_1} \cdots z_{i_l}$. In this tensor product, many coefficients are permutations, i.e., $w_{\sigma_1 \cdots \sigma_l} = w_{i_1 \cdots i_l}$. Via the definition of an $l^n \times \binom{k}{l}$ matrix $\boldsymbol{T}$ that extracts only one permutation per coefficient, we can create smaller operator determinants

$$\widehat{\boldsymbol{\Delta}}_i = \boldsymbol{L} \widetilde{\boldsymbol{\Delta}}_i \boldsymbol{T} \in \mathbb{C}^{\binom{k}{l} \times \binom{k}{l}}, \tag{3.33}$$

where each row of $\boldsymbol{T}$ has exactly one non-zero element and each column contains at least one non-zero element and at most $l!$ non-zero elements. The matrix $\boldsymbol{L}$ is a generic nonsingular matrix[6] of size $\binom{k}{l} \times k^n$. The matrix $\boldsymbol{T}$ compresses the problem, since $\boldsymbol{T}\boldsymbol{w} = \boldsymbol{z}$, and therefore

$$\boldsymbol{\Delta}_0^{-1} \boldsymbol{\Delta}_i \boldsymbol{T} = \boldsymbol{T} \boldsymbol{G}_i, \tag{3.34}$$

where $\boldsymbol{G}_i$ is a restriction of $\boldsymbol{\Delta}_0^{-1} \boldsymbol{\Delta}_i$ to the invariant subspace [118]. In an efficient implementation of this transformation, it is even possible to exploit the sparsity of the pre-multiplication (viz., $\boldsymbol{L}$ can be sparse and have the same effect) and post-multiplication or even to avoid explicitly building most of these matrices altogether ([6] provides such a sparse approach). This is important, since $\widetilde{\boldsymbol{\Delta}}_i$ can become very large. Note that this is not a full transformation of a rectangular MEP into a linear square MEP, i.e., the coefficient matrices $\boldsymbol{B}_{ij}$ are not obtained explicitly, only the associated system of coupled reduced Kronecker operator determinants $\widehat{\boldsymbol{\Delta}}_i$. Hence, we can not use the efficient numerical methods designed for square MEPs that operate directly on the coefficient matrices. For more details, we refer the interested reader to [6, 118].

Example 3.9 (continuing from p. 109). In order to obtain these solutions via a system of coupled GEPs, we construct the rectangular Kronecker operator determinants $\widetilde{\boldsymbol{\Delta}}_i \in \mathbb{C}^{9 \times 4}$ from the rectangular coefficient matrices of (3.11). After pre-multiplication with

$$\boldsymbol{L} = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{bmatrix} \tag{3.35}$$

and post-multiplication with

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.36}$$

[6] In deriving the second technique, Hochstenbach et al. [118] have actually used this fact, by starting from the previous technique and constructing $\boldsymbol{L}$ as $\boldsymbol{Q}(\boldsymbol{P}_1 \otimes \cdots \otimes \boldsymbol{P}_n)$, where $\boldsymbol{Q}$ is a generic matrix. This is an intermediate result in [118] and provides an different derivation for the matrix $\boldsymbol{L}$ than in [6]. Futhermore, it shows that it is possible to combine the first technique with this compression step in order to remove the spurious solutions that appear in the sketching procedure.

the reduced Kronecker operator determinants $\widehat{\boldsymbol{\Delta}}_i = \boldsymbol{L}\widetilde{\boldsymbol{\Delta}}_i\boldsymbol{T} \in \mathbb{C}^{3\times 3}$ are obtained. Solving (3.32) results in the three solutions of (3.11) in Table 3.2.

3.2.3 Number of solutions

The fact that the matrix pencil consists of rectangular matrices results in a different definition of the multivariate characteristic polynomial equations. Before giving this definition, consider the following selection procedure:

$$[\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})]_i = \boldsymbol{S}_{\sigma_i}\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}), \tag{3.37}$$

where $\boldsymbol{S}_{\sigma_i} \in \mathbb{N}^{l\times k}$ is a row selection matrix that selects the rows that correspond to the ith l-combination σ_i of the k row indices of $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$. There are exactly $\binom{k}{l}$ such l-combinations, hence i runs from 1 to $\binom{k}{l}$, cf., (2.14).

Definition 3.4. If we consider an MEP with matrix pencil $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ as defined in Definition 3.2, then the multivariate polynomial equations

$$\chi_i(\boldsymbol{\lambda}) = \det([\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})]_i) = 0, \tag{3.38}$$

for $i = 1, \ldots, \binom{k}{l}$, are the **multivariate characteristic polynomial equations** of that MEP.

The common roots of the resulting system of multivariate characteristic polynomial equations[7] correspond to the eigenvalues of the MEP. This system is an overdetermined system, because the vectors in the null space of every square matrix $[\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})]_i$ need to be colinear, for $i = 1, \ldots, \binom{k}{l}$.

Example 3.10. When looking at Example 3.4, the multivariate characteristic polynomial equations are

$$\begin{cases} \chi_1(\boldsymbol{\lambda}) = 14 + 7\lambda_1 - 28\lambda_2 - \lambda_1^2 - 12\lambda_1\lambda_2 - 32\lambda_2^2 = 0, \\ \chi_2(\boldsymbol{\lambda}) = 2 - 3\lambda_1 + \lambda_1^2 + \lambda_1^2 + 3\lambda_1\lambda_2 + 2\lambda_2^2 = 0, \\ \chi_3(\boldsymbol{\lambda}) = 4 - \lambda_1 - \lambda_2 - \lambda_1^2 - 9\lambda_1\lambda_2 - 8\lambda_2^2 = 0. \end{cases} \tag{3.39}$$

This overdetermined system of polynomial equations has three affine solutions, as visualized in Figure 3.1. The common roots of (3.39) correspond to the eigenvalues of (3.11), which can be verified with the numerical results in Table 3.2.

Applying Bézout's theorem directly on the obtained system of multivariate characteristic polynomial equations is not possible, since the system is overdetermined. However, two bounds on the number of affine eigenvalues do exist. The first bound has been given by Shapiro and Shapiro [216] and considers

[7] Sometimes the term *secular equation* is used to denote a characteristic polynomial equation. This comes from the fact that the characteristic polynomial was used to calculate the secular perturbations of planetary orbits in, for example, the work of Lagrange [44].

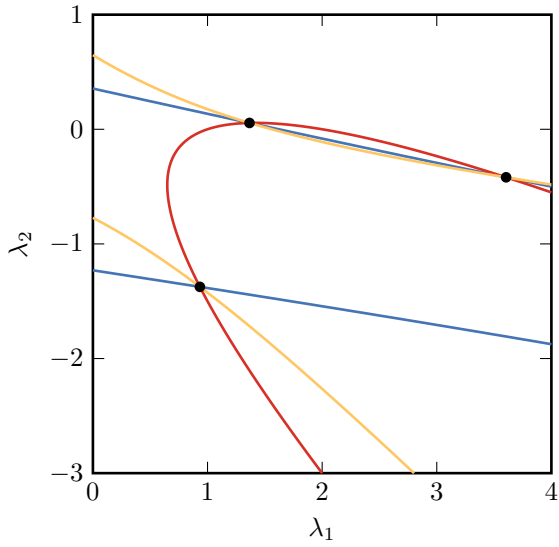


Figure 3.1. Real picture of the system of three multivariate characteristic polynomial equations derived in Example 3.10. The common roots (•) of the polynomials $\chi_1(\boldsymbol{\lambda})$ (—), $\chi_2(\boldsymbol{\lambda})$ (—), and $\chi_3(\boldsymbol{\lambda})$ (—) correspond to the eigenvalues of the linear two-parameter eigenvalue problem in (3.11).

linear MEPs , while the second bound by Hochstenbach et al. [118] gives an upper bound on the number of affine solutions for a polynomial MEP.

Theorem 3.2 (Shapiro–Shapiro bound). For a linear n-parameter eigenvalue problem ($d_{\max} = 1$), as defined in Definition 3.2, with a zero-dimensional solution set, the number of affine eigenvalues is at most equal to

$$m_b = \binom{l+n-1}{l}. \tag{3.40}$$

Proof. A proof of this theorem can be found in [216]. □

A sufficient condition for the linear problem to have finitely many solutions for an arbitrary matrix $\boldsymbol{A}_{0\cdots0}$ is that

$$\operatorname{rank}(\boldsymbol{A}_{1\cdots0}\lambda_1 + \cdots + \boldsymbol{A}_{0\cdots1}\lambda_n) = n \tag{3.41}$$

for all $\boldsymbol{\lambda} \neq \mathbf{0}$.

Example 3.11. Applying Theorem 3.2 to the linear two-parameter eigenvalue problem in (3.11), we retrieve the exact number of affine solutions of

the problem as

$$m_b = \binom{2+2-1}{2} = \binom{3}{2} = 3, \tag{3.42}$$

where the number of columns $l = 2$. This agrees also with the number of common roots of (3.39).

Code 3.1. The Shapiro–Shapiro bound for a linear MEP can be computed via `shapiro(mep)`. `toymep1` corresponds to problem (3.11) and is part of MacaulayLab's database.

```
>> shapiro(toymep1)

ans =
    3
```

Theorem 3.3 (Hochstenbach–Košir–Plestenjak bound). For a polynomial n-parameter eigenvalue problem, as defined in Definition 3.2, with zero-dimensional solution set, the number of affine eigenvalues is at most equal to

$$m_b = d_{\max}^n \binom{l+n-1}{n}. \tag{3.43}$$

Proof. A proof of this theorem can be found in [118]. □

Example 3.12. For polynomial MEPs, Theorem 3.3 gives an upper bound on the number of affine eigenvalues. For example, for the quadratic two-parameter eigenvalue problem in (3.12), an upper bound is given by

$$m_b = 2^2 \binom{2+2-1}{2} = 4\binom{3}{2} = 12, \tag{3.44}$$

where the number of columns $l = 2$. Notice that the number of affine solutions for this problem is $9 < 12$, since the MEP has also 3 solutions at infinity.

Code 3.2. The HKP bound can be computed via `hkp(mep)`. `toymep2` is part of MacaulayLab's database.

```
>> hkp(toymep2)

ans =
    12
```

Similar as for the system of multivariate polynomial equations (Remark 2.1), the eigenvalue solutions of the MEP can also be considered in the projective space $\mathbb{P}^n$, where we consider the homogeneous MEP,

$$\boldsymbol{\mathcal{M}}^h(\widetilde{\boldsymbol{\lambda}})\boldsymbol{z} = \left(\sum_{\mathcal{W}} \boldsymbol{A}_{\boldsymbol{\omega}} \lambda_0^{d_{\max}-|\boldsymbol{\omega}|} \boldsymbol{\lambda}^{\boldsymbol{\omega}}\right)\boldsymbol{z} = \boldsymbol{0}, \tag{3.45}$$

and the projective eigenvalues $\widetilde{\boldsymbol{\lambda}} = (\lambda_0, \lambda_1, \ldots, \lambda_n)$, for which an equivalence relation $\sim$ exists on the non-zero points of $\mathbb{C}^{n+1}$ (Section 4.2). The above-mentioned solutions at infinity can then be interpreted as the solutions for which $\lambda_0 = 0$.

3.3 Block Macaulay matrix and its subspaces

An essential result of this dissertation is the development of methods to solve MEPs. In order to achieve this result, we need to introduce a new matrix, the block Macaulay matrix. Similar to Macaulay matrix for polynomial systems, the right null space and column space of the block Macaulay matrix lead to the solutions of the generating MEP.

The MEP $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \boldsymbol{0}$ constitutes the so-called *seed equation* of the corresponding block Macaulay matrix. The block Macaulay matrix is generated by this seed equation via a forward shift recursion (FSR)[8]: multiplying the seed equation (i.e., the MEP) with different monomials $\{\boldsymbol{\lambda}^{\boldsymbol{\alpha}}\}$ of increasing total degree d_{r} leads to "new" matrix equations, the coefficient matrices of which are organized as the block rows of the block Macaulay matrix.

Example 3.13. To introduce the block Macaulay matrix, consider the quadratic two-parameter eigenvalue problem in (3.12),

$$\left(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{11}\lambda_1\lambda_2 + \boldsymbol{A}_{02}\lambda_2^2\right)\boldsymbol{z} = \boldsymbol{0}. \tag{3.46}$$

Multiplying the MEP by the two eigenvalues λ_1 and λ_2, leads to two "new" matrix equations:

$$\begin{aligned} \lambda_1\left(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{11}\lambda_1\lambda_2 + \boldsymbol{A}_{02}\lambda_2^2\right)\boldsymbol{z} &= \boldsymbol{0} \\ \lambda_2\left(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{11}\lambda_1\lambda_2 + \boldsymbol{A}_{02}\lambda_2^2\right)\boldsymbol{z} &= \boldsymbol{0}. \end{aligned} \tag{3.47}$$

[8]This block forward multi-shift recursion is the block extension of the scalar forward multi-shift recursion in Section 2.3.

We can continue this process with monomials of increasing total degree d_r, i.e.,

$$\underbrace{\lambda_1, \lambda_2}_{d_\mathrm{r}=1}, \underbrace{\lambda_1^2, \lambda_1\lambda_2, \lambda_2^2}_{d_\mathrm{r}=2}, \underbrace{\lambda_1^3, \lambda_1^2\lambda_2, \dots}_{d_\mathrm{r}\geq 3} \tag{3.48}$$

and arrange the resulting coefficient matrices in a block Macaulay matrix (seed equation in red):

$$\begin{array}{c|ccc|ccc|ccc|c} & \boldsymbol{z} & \lambda_1\boldsymbol{z} & \lambda_2\boldsymbol{z} & \lambda_1^2\boldsymbol{z} & \lambda_1\lambda_2\boldsymbol{z} & \lambda_2^2\boldsymbol{z} & \lambda_1^3\boldsymbol{z} & \lambda_1^2\lambda_2 & \lambda_1\lambda_2^2 & \\ 1 & \boldsymbol{A}_{00} & \boldsymbol{A}_{10} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{11} & \boldsymbol{A}_{02} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots \\ \lambda_1 & \boldsymbol{0} & \boldsymbol{A}_{00} & \boldsymbol{0} & \boldsymbol{A}_{10} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{11} & \boldsymbol{A}_{02} & \cdots \\ \lambda_2 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{00} & \boldsymbol{0} & \boldsymbol{A}_{10} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{11} & \cdots \\ \lambda_1^2 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{00} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{10} & \boldsymbol{0} & \boldsymbol{0} & \cdots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array} . \tag{3.49}$$

When we further *enlarge* the block Macaulay matrix with monomials of higher total degree d_r in the FSR, we obtain a sparse and structured matrix, as visualized in Figure 3.2 (for degree $d = 6$).

Code 3.3. A block Macaulay matrix of degree d can easily be constructed via `macaulay(mep,d)`. Notice that this is the same function as for the (scalar) Macaulay matrix.

```
>> M = macaulay(toymep2,6);
```

Definition 3.5. Consider the MEP $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \boldsymbol{0}$, which serves as the seed equation. Let the total degree of the MEP be denoted by d_max. The **block Macaulay matrix** of degree d, $\boldsymbol{M}_d \in \mathbb{C}^{p_d \times q_d}$, contains the coefficient matrices of the seed equation and the matrix equations generated by the FSR with monomials of increasing total degree $d_\mathrm{r} = 1, \dots, (d - d_\mathrm{max})$, i.e.,

$$\boldsymbol{M}_d = \left[\!\!\left[\left\{ \prod_{i=1}^{n} \lambda_i^{d_i} \right\} \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} \right]\!\!\right], \tag{3.50}$$

where $[\![\cdot]\!]$ denotes the arrangement of the shifted coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}}$ (not the associated eigenvalues or eigenvectors) of the matrix equations in block rows.

These shifted coefficient matrices are indexed both in row (different monomials of the FSR) and column (different associated monomials) direction by the different monomials in the eigenvalues of total degree at most d. The number of rows p_d and columns q_d of $\boldsymbol{M}_d$ are given by

$$p_d = k \binom{d - d_\mathrm{max} + n}{n} = k \frac{(d - d_\mathrm{max} + n)!}{n!(d - d_\mathrm{max})!} \tag{3.51}$$

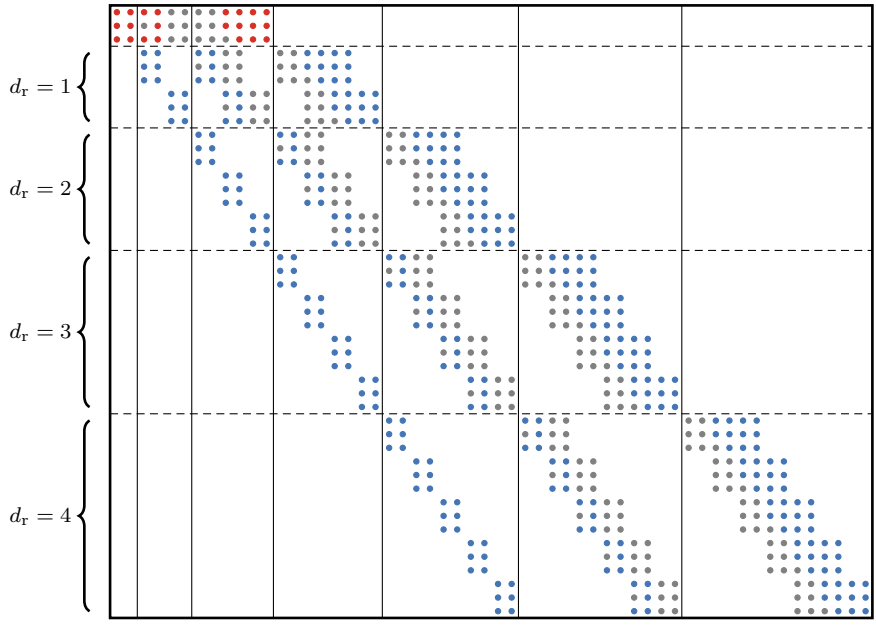


Figure 3.2. Spy plot of the block Macaulay matrix of degree 6 for the quadratic two-parameter eigenvalue in Example 3.13. The elements of the seed equation, i.e., the generating MEP, are indicated with red dots (●), while the elements of the "new" matrix equations obtained by multiplying with monomials of total degree $d_r \geq 1$ in the FSR are indicated with blue dots (●). For didactic purposes, we indicate the zero elements/matrices that come from the seed equation with gray dots (●). The elements not shown are zero. Vertical lines indicate the different degree blocks, while horizontal dashed lines separate the monomials of different total degree d_r in the FSR.

and

$$q_d = l\binom{d+n}{n} = l\frac{(d+n)!}{n!d!}. \tag{3.52}$$

The actual structure of the block Macaulay matrix depends on its monomial ordering (Definition 2.6).

The MEP and the "new" matrix equations obtained via the FSR can be written as the matrix-vector product of the generated block Macaulay matrix $\boldsymbol{M}_d \in \mathbb{C}^{p_d \times q_d}$ and a structured vector $\boldsymbol{v}_d \in \mathbb{C}^{q_d \times 1}$:

$$\boldsymbol{M}_d \begin{bmatrix} \boldsymbol{z} \\ \hline \boldsymbol{z}\lambda_1 \\ \vdots \\ \boldsymbol{z}\lambda_n \\ \hline \vdots \\ \hline \boldsymbol{z}\lambda_1^d \\ \vdots \\ \boldsymbol{z}\lambda_n^d \end{bmatrix} = \boldsymbol{0}. \tag{3.53}$$

We increase the degree d of the block Macaulay matrix $\boldsymbol{M}_d$ until it reaches the **solution degree** $d_\circ$, a notion on which we elaborate in Section 3.4.3. The vector $\boldsymbol{v}_d$ is a vector in the right null space of $\boldsymbol{M}_d$ and has a special block multivariate Vandermonde structure, which is *enforced* by the consecutive block FmSRs that generate the block rows of $\boldsymbol{M}_d$. In the structure of both the (right[9]) null space (Section 3.4) and the column space (Section 3.5) of $\boldsymbol{M}_d$ lies the key to solving its generating MEP. The iterative block Macaulay matrix solution approach (Algorithm 3.1) looks very similar as in the Macaulay matrix approach in Chapter 2.

Remark 3.2. Similar to Remark 2.2, we also use horizontal and vertical lines to separate different degree blocks in matrices and vectors (e.g., in Figure 3.2). Furthermore, we make a distinction between **block rows/columns** and **degree blocks**. A block row/column gathers all the rows/columns that correspond to one monomial (e.g., all the rows that belong to λ_1^2), while a degree block contains all the block rows/columns that correspond to monomials of the same total degree (e.g., all the rows that belong to λ_1^2, $\lambda_1\lambda_2$, and λ_2^2). A degree block contains multiple block rows/columns (except when the total degree is zero or the number of variables is equal to one).

Remark 3.3. To alleviate the notational complexity, we no longer specify the degree d explicitly in the remainder of this chapter (unless when necessary), but we assume it to be *large enough*, i.e., $d \geq d_\circ$ (Section 3.4.3).

[9] In the remainder of this chapter, we no longer mention the qualification *right* explicitly. We always consider the right null space, except when denoted otherwise.

Algorithm 3.1 Iterative eigenvalue-finding via the block Macaulay matrix

Require: $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$
$d \leftarrow d_{\max}$
Construct the block Macaulay matrix of degree d
while $d < d_\circ$ **do**
    Check structure of null space or column space
    if it is possible to find the affine eigenvalues (Section 3.4.3) **then**
        $d = d_\circ$
    else
        $d \leftarrow d + 1$
        Construct the block Macaulay matrix of degree d
    end if
end while
Find the affine eigenvalues $\boldsymbol{\lambda}|_{(j)}$, for $j = 1, \ldots, m_a$, in the null space (Algorithm 3.2) or column space (Algorithm 3.3)
return $\boldsymbol{\lambda}|_{(j)}$, for $j = 1, \ldots, m_a$

3.4 Null space based approach

We now exploit the structure of the null space of the block Macaulay matrix in order to find the solutions of its seed equation, i.e., the MEP that we want to solve. We show again that a multidimensional realization problem in the structured null space[10] yields the affine solutions of the MEP (Section 3.4.1). Next, we use Schur decompositions to improve the accuracy in the presence of solutions with multiplicity greater than one (Section 3.4.2) and translate the notion of a *large enough* degree (i.e., the solution degree $d_\circ$) to the block Macaulay matrix in order to deal with the solutions at infinity (Section 3.4.3). Finally, we summarize the different steps of the null space based eigenvalue-finding algorithm (Section 3.4.4).

3.4.1 Multidimensional realization theory

We use the same two-step exposition as in Section 2.4.1: we look at the block multivariate Vandermonde basis matrix (Section 3.4.1.1) and generalize it to any (numerical) basis matrix of the null space of the block Macaulay matrix (Section 3.4.1.2).

[10] Similar to the null space of the Macaulay matrix, the null space of the block Macaulay matrix (for a sufficiently large degree) can be modeled as the column space of an observability matrix of a multidimensional descriptor system [259]. The observability matrix is constructed with output matrices instead of output vectors. From this observation also stems the title of Section 3.4.1; the null space based eigenvalue-finding approach could be considered as a multidimensional realization problem in the column space of that observability matrix.

3.4.1.1 Block multivariate Vandermonde basis matrix

We consider, again for didactic purposes, an MEP that only has m_a simple (i.e., algebraic multiplicity is one), affine (i.e., non-infinite), and isolated solutions (i.e., the solution set is zero-dimensional). For a block Macaulay matrix $\boldsymbol{M}$ of degree $d \geq d_\circ$ (Section 3.4.3), there exists a block multivariate Vandermonde vector $\boldsymbol{v}|_{(j)}$ $(j = 1, \ldots, m_a)$ in the null space of $\boldsymbol{M}$ for every solution of the MEP and, together, these basis vectors span the entire null space of $\boldsymbol{M}$. They naturally form the block multivariate Vandermonde basis matrix $\boldsymbol{V} \in \mathbb{C}^{q \times m_a}$ of degree $d \geq d_\circ$ (same degree as $\boldsymbol{M}$):

$$\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}|_{(1)} & \cdots & \boldsymbol{v}|_{(m_a)} \end{bmatrix} = \left[\begin{array}{ccc} \boldsymbol{z}|_{(1)} & \cdots & \boldsymbol{z}|_{(m_a)} \\ \hline (\lambda_1 \boldsymbol{z})|_{(1)} & \cdots & (\lambda_1 \boldsymbol{z})|_{(m_a)} \\ \vdots & & \vdots \\ (\lambda_n \boldsymbol{z})|_{(1)} & \cdots & (\lambda_n \boldsymbol{z})|_{(m_a)} \\ \hline (\lambda_1^2 \boldsymbol{z})|_{(1)} & \cdots & (\lambda_1^2 \boldsymbol{z})|_{(m_a)} \\ \vdots & & \vdots \end{array}\right]. \tag{3.54}$$

Comparing $\boldsymbol{V}$ with (2.45) supports the hypothesis that $\boldsymbol{V}$ has a similar "special shift structure" as the basis matrix of the null space of the (scalar) Macaulay matrix. Mathematically, this "special shift structure" can again be written as (when we shift some (block) rows with the shift polynomial $g(\boldsymbol{\lambda})$)

$$\underbrace{\boldsymbol{S}_1 \boldsymbol{V}}_{\text{before shift}} \quad \boldsymbol{D}_g = \underbrace{c_{\boldsymbol{\alpha}_1} \boldsymbol{S}_{\boldsymbol{\lambda}^{\boldsymbol{\omega}_1}} \boldsymbol{V} + \cdots + c_{\boldsymbol{\omega}_k} \boldsymbol{S}_{\boldsymbol{\lambda}^{\boldsymbol{\omega}_k}} \boldsymbol{V}}_{\text{after shift}} = \boldsymbol{S}_g \boldsymbol{V}, \tag{3.55}$$

where the diagonal matrix $\boldsymbol{D}_g \in \mathbb{C}^{m_a \times m_a}$ contains the evaluations of the shift polynomial $g(\boldsymbol{\lambda}) = c_{\boldsymbol{\omega}_1} \boldsymbol{\lambda}^{\boldsymbol{\omega}_1} + \cdots + c_{\boldsymbol{\omega}_k} \boldsymbol{\lambda}^{\boldsymbol{\omega}_k}$ in the different solutions of the MEP and the row selection matrices $\boldsymbol{S}_1$ and $\boldsymbol{S}_{\boldsymbol{\lambda}^{\boldsymbol{\omega}}}$ select the (block) rows before and after the shift, respectively. Hence, (3.55) corresponds to the expression

$$(\boldsymbol{S}_g \boldsymbol{V}) = (\boldsymbol{S}_1 \boldsymbol{V}) \boldsymbol{D}_g, \tag{3.56}$$

when combining the different row selection matrices in the right-hand side. We say that the rows in $\boldsymbol{S}_g \boldsymbol{V}$ are *hit* by the shift with $g(\boldsymbol{\lambda})$. In order for this expression to cover all the affine solutions, the row selection matrix $\boldsymbol{S}_1 \in \mathbb{R}^{m_a \times q}$ has to select m_a linearly independent rows from $\boldsymbol{V}$ (then $\boldsymbol{S}_1 \boldsymbol{V}$ is square and nonsingular). The *row combination matrix*[11] $\boldsymbol{S}_g \in \mathbb{R}^{m_a \times q}$, on the other hand, simply selects the linear combination of rows *hit* by the shift with $g(\boldsymbol{\lambda})$.

Example 3.14. Consider the linear two-parameter eigenvalue problem from Example 3.4 and construct a block multivariate Vandermonde matrix (we suppose that the solutions are known). In order to shift the first three (linearly independent) rows $(z_1, z_2, \lambda_1 z_1)$ with a shift polynomial $g(\boldsymbol{\lambda}) = 4\lambda_2^3$,

[11] When the shift is merely a monomial of (some of the) eigenvalues, the row combination matrix $\boldsymbol{S}_g$ is a row selection matrix because every shift only hits one row.

we need a degree $d_\circ = 4$ basis matrix of the null space. The row selection matrix $\boldsymbol{S}_1 \in \mathbb{R}^{3\times 30}$ corresponds to

$$\boldsymbol{S}_1 = \begin{matrix} \\ \end{matrix}\overset{\begin{matrix}1 & 2 & 3 & 4 & & 30\end{matrix}}{\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \end{bmatrix}}. \tag{3.57}$$

After shifting $z_1, z_2, \lambda_1 z_1$ by $g(\boldsymbol{\lambda}) = 4\lambda_2^3$, the monomials hit correspond to the 19th ($\lambda_2^3 z_1$), 20th ($\lambda_2^3 z_2$), and 27th ($\lambda_1\lambda_2^3 z_1$) row of the basis matrix. They can be selected by $\boldsymbol{S}_g \in \mathbb{R}^{3\times 30}$,

$$\boldsymbol{S}_g = \begin{matrix} \\ \end{matrix}\overset{\begin{matrix}1 & & 18 & 19 & 20 & 21 & & 26 & 27 & 28 & 29 & 30\end{matrix}}{\begin{bmatrix} 0 & \cdots & 0 & 4 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 4 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 4 & 0 & 0 & 0 \end{bmatrix}}. \tag{3.58}$$

The shift problem is given by

$$\boldsymbol{S}_1\boldsymbol{V}\begin{bmatrix} -10.3979 & 0 & 0 \\ 0 & 0.0007 & 0 \\ 0 & 0 & -0.2928 \end{bmatrix} = \boldsymbol{S}_g\boldsymbol{V}. \tag{3.59}$$

Code 3.4. The block Vandermonde basis matrix of the null space is constructed from the different eigenvalues `lambda` and eigenvectors `z`.

```
>> V = vandermonde(4,2,lambda,z);
```

If we want to shift the first three rows in the block Vandermonde basis matrix, we also need to supplement the length of the eigenvector `l = 2` to `shiftmatrix(d,n,rows,shift,l)`.

```
>> rows = [1 2 3]; % shift row 1, 2, and 3
>> l = 2; % eigenvector have length 2
>> S1 = shiftmatrix(2,2,rows,[1 0 0],l); % shift with 1
>> Sg = shiftmatrix(2,2,rows,[4 0 3],l); % shift with g(x)
```

This gives us the shift problem from (3.59):

```
>> diag(inv(S1*V)*(Sg*V))

ans =
   -10.3979
    0.0007
   -0.2928
```

3.4.1.2 Any numerical basis matrix

Similar as in Section 2.4.1.2, a numerical basis matrix $\boldsymbol{Z} \in \mathbb{C}^{q \times m_a}$ of the null space of the block Macaulay matrix $\boldsymbol{M}$ can be used instead of the unknown block multivariate Vandermonde basis matrix $\boldsymbol{V}$. Before translating the theoretical multidimensional realization problem into a practical one, the next proposition makes this "special shift structure" more concrete for the block case.

Proposition 3.1 (Appendix C). The (affine) null space of the block Macaulay matrix is **(backward) block multi-shift-invariant**. This means that if we select a block row of a basis matrix of the null space and multiply/shift this block row with one of the eigenvalues, then we obtain another block row of that basis matrix (when the degree is *large enough*, i.e., $d \geq d_\circ$).

The block multi-shift-invariance is a property of the null space as a vector space and not of its specific basis matrix (Appendix C). Via a linear transformation $\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{T}$, the block multivariate Vandermonde basis matrix $\boldsymbol{V}$ can be replaced by any numerical basis matrix $\boldsymbol{Z}$, with $\boldsymbol{T} \in \mathbb{C}^{m_a \times m_a}$ a nonsingular transformation matrix, transforming (3.56) into a solvable GEP,

$$(\boldsymbol{S}_g\boldsymbol{Z})\boldsymbol{T} = (\boldsymbol{S}_1\boldsymbol{Z})\boldsymbol{T}\boldsymbol{D}_g, \tag{3.60}$$

where $\boldsymbol{T}$ contains the eigenvectors and $\boldsymbol{D}_g$ the eigenvalues of the matrix pencil $(\boldsymbol{S}_g\boldsymbol{Z}, \boldsymbol{S}_1\boldsymbol{Z})$. Alternatively, we can also consider the SEP

$$(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_g\boldsymbol{Z})\boldsymbol{T} = \boldsymbol{T}\boldsymbol{D}_g. \tag{3.61}$$

Then, the matrix of eigenvectors $\boldsymbol{T}$ could be used again to obtain $\boldsymbol{V}$ (via $\boldsymbol{V} = \boldsymbol{Z}\boldsymbol{T}$ and a normalization), and hence, to find the affine solutions of the MEP from the null space of the block Macaulay matrix.

Example 3.15. Assume that the solution degree $d_\circ$ of the block Macaulay matrix generated by (3.11) is 2. We compute a numerical basis matrix $\boldsymbol{Z}$ of its null space: $\boldsymbol{Z} \in \mathbb{C}^{12\times 3}$ because this MEP has three (affine) solutions. The first three rows are linearly independent, so we shift these rows with λ_1 and λ_2 to obtain the eigenvalues of the MEP. We can use $\boldsymbol{S}_1$ from (3.57), while $\boldsymbol{S}_{\lambda_1}$ and $\boldsymbol{S}_{\lambda_2}$ are given by

$$\boldsymbol{S}_{\lambda_1} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{3.62}$$

and

$$\boldsymbol{S}_{\lambda_2} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}. \tag{3.63}$$

Solving (3.61) twice results in the diagonal matrices

$$\boldsymbol{D}_{\lambda_1} = \begin{bmatrix} 0.9338 & 0 & 0 \\ 0 & 1.3683 & 0 \\ 0 & 0 & 3.6026 \end{bmatrix} \tag{3.64}$$

and

$$\boldsymbol{D}_{\lambda_2} = \begin{bmatrix} -1.3750 & 0 & 0 \\ 0 & 0.0552 & 0 \\ 0 & 0 & -0.4183 \end{bmatrix}. \tag{3.65}$$

Code 3.5. Via a numerical basis matrix of the null space, we can set-up the different eigenvalue problems.

```
>> Z = null(macaulay(toymep1),2);
>> Sl1 = shiftmatrix(2,2,rows,[1 1 0],1); % shift with 1*l1
>> Sl2 = shiftmatrix(2,2,rows,[1 0 1],1); % shift with 1*l2
>> eig(Sl1*Z,S1*Z)

ans =
    0.9338
    1.3683
    3.6026
>> eig(Sl2*Z,S1*Z)

ans =
   -1.3750
    0.0552
   -0.4183
```

3.4.2 Solutions with multiplicity greater than one

Dealing with solutions in the block multivariate Vandermonde basis matrix $\boldsymbol{V}$ of the null space with a multiplicity greater than one is a similar problem as in the (scalar) multivariate Vandermonde basis matrix. Multiplicity greater than one results again in a Jordan normal form, which leads to inaccurate solutions when working with floating-point algorithms (Section 2.4.2). In order to obtain more accurate solutions, the Schur decomposition approach from Section 2.4.2 can also be used for the block Macaulay matrix. We extend the technique of Corless et al. [62] to the MEP setting and use $n+1$ different shift polynomials: one random shift polynomial,

$$g_0(\boldsymbol{\lambda}) = \sum_{i=1}^{n} c_i \lambda_i, \tag{3.66}$$

with $c_i \in \mathbb{C}$ is a random complex number, and n linear shift polynomials

$$g_i(\boldsymbol{\lambda}) = \lambda_i, \tag{3.67}$$

for $i = 1, \ldots, n$. A Schur decomposition of $(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{g_0}\boldsymbol{Z})$ yields the upper triangular matrix $\boldsymbol{D}_{g_0}$ and orthonormal matrix $\boldsymbol{Q}$:

$$\boldsymbol{Q}\boldsymbol{D}_{g_0}\boldsymbol{Q}^{\mathrm{H}} = (\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{g_0}\boldsymbol{Z}), \tag{3.68}$$

where $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^{\mathrm{H}}$ because the matrix $\boldsymbol{Q}$ is orthonormal. The unitary transformations that create the triangular form (3.68) result in more accurate eigenvalues than (3.60) in the presence of multiplicities. By re-using the matrix $\boldsymbol{Q}$, every eigenvalue parameter $\lambda_i|_{(j)}$ of the MEP is on the jth position of the diagonal of the upper triangular matrices $\boldsymbol{D}_{\lambda_i}$, i.e.,

$$\begin{aligned} \boldsymbol{D}_{\lambda_1} &= \boldsymbol{Q}^{\mathrm{H}}(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{\lambda_1}\boldsymbol{Z})\boldsymbol{Q}, \\ &\vdots \\ \boldsymbol{D}_{\lambda_n} &= \boldsymbol{Q}^{\mathrm{H}}(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{\lambda_n}\boldsymbol{Z})\boldsymbol{Q}, \end{aligned} \tag{3.69}$$

where every upper triangular matrix $\boldsymbol{D}_{\lambda_i}$ contains the different evaluations in one eigenvalue parameter:

$$\boldsymbol{D}_{\lambda_i} = \begin{bmatrix} \lambda_i|_{(1)} & \times & \times \\ 0 & \ddots & \times \\ 0 & 0 & \lambda_i|_{(m_a)} \end{bmatrix} \tag{3.70}$$

An additional clustering step could also be used to even further improve the accuracy (cf., the clustering step in Section 2.4.2).

Example 3.15 (continuing from p. 121). Instead of combining the different eigenvalues in Example 3.15 or reconstructing the block Vandermonde basis matrix, we can use $n+1$ shift polynomials to retrieve the solutions. We shift with a random linear polynomial in the eigenvalues $g_0(\boldsymbol{\lambda})$ to determine $\boldsymbol{D}_{g_0}$ and $\boldsymbol{Q}$ via (3.68). By re-using the matrix $\boldsymbol{Q}$, we can retrieve the same solutions as in Example 3.15 via the diagonal elements of the $\boldsymbol{D}_{\lambda_i}$ in (3.69):

$$\begin{aligned} \boldsymbol{D}_{\lambda_1} &= \boldsymbol{Q}^{-1}(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{\lambda_1}\boldsymbol{Z})\boldsymbol{Q}, \\ \boldsymbol{D}_{\lambda_2} &= \boldsymbol{Q}^{-1}(\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{\lambda_2}\boldsymbol{Z})\boldsymbol{Q}. \end{aligned} \tag{3.71}$$

3.4.3 About the notion of a large enough degree

When we move from the Macaulay matrix to the block Macaulay matrix, we need to pose an important question: "Is the notion of the solution degree $d_\circ$ the same for the block Macaulay matrix?" The answer is simply *yes*! An MEP can also have solutions at infinity, due to the singularity of some higher

degree coefficient matrix or sparsity of the support. When increasing the degree d by multiplying with more monomials in the FSR, the nullity of the block Macaulay matrix $\boldsymbol{M}$ stabilizes for a certain d at the total number of solutions $m_b = m_a + m_\infty$ in the case of a zero-dimensional solution set. We denote this degree again by d_*. Every solution spans one basis vector in this null space; hence, all the columns of the numerical basis matrix are linear combinations of affine solutions and solutions at infinity.

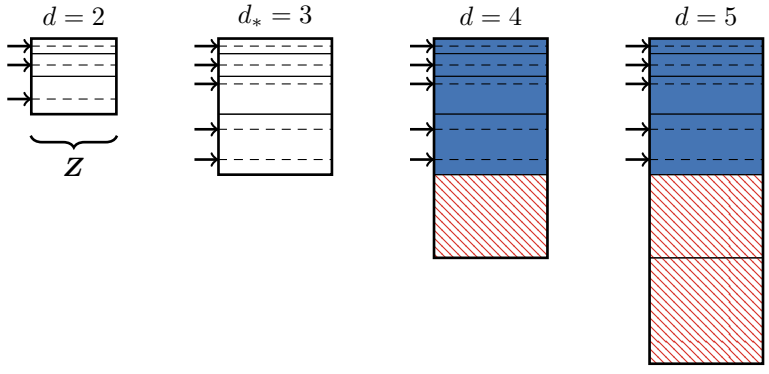
For higher degrees $d > d_*$, the linearly independent rows that correspond to the affine solutions remain stable at their respective positions and new degree blocks contain no additional linearly independent rows (Figure 3.3a). Next to the linearly independent rows related to the affine solutions, the basis matrix of the null space can also contain linearly independent rows related to the solutions at infinity. These linearly independent rows move to higher degree blocks when we increase the degree (Figure 3.3b). Eventually, a gap in the rows emerges that separates both types of linearly independent rows. The basis matrix consists of two or three zones for $d > d_*$: a **regular zone** that contains the linearly independent rows related to the affine solutions, a **gap zone** without additional linearly independent rows, and, when the MEP has solutions at infinity, a **singular zone** that contains the linearly independent rows that correspond to the solutions at infinity. Via a column compression (Theorem 2.3), we can deflate again the solutions at infinity and replace $\boldsymbol{Z}$ in (3.60) by the compressed basis matrix $\boldsymbol{W}_{11}$.

When we want to shift the linearly independent rows that correspond to the affine solutions (3.60) with a shift polynomial $g(\boldsymbol{\lambda})$ of degree d_g, the gap zone needs to be able to *accommodate* this shift, which means that the monomials with highest total degree hit by the shift must be present in the gap zone. Hence, the degree d of the block Macaulay matrix corresponds to the solution degree $d_\circ$ when the gap zone consists of d_g degree blocks.

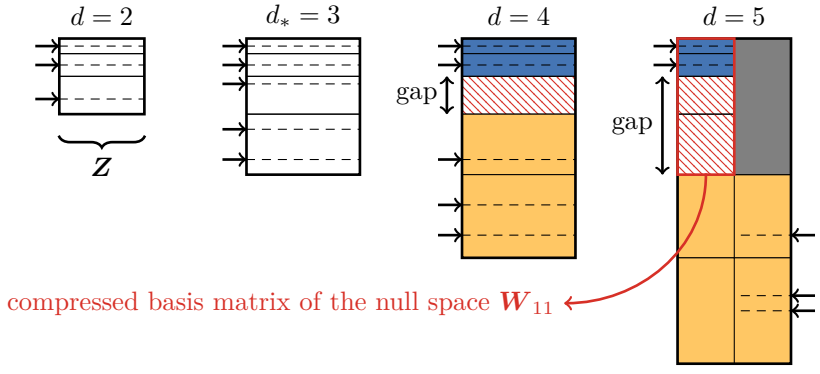
Example 3.14 (continuing from p. 119). To solve the MEP in (3.11) with only affine solutions, we iteratively build a block Macaulay matrix $\boldsymbol{M}$ for increasing degree $d = 1, \ldots, 4$ (as described in Algorithm 3.1). The successive matrices $\boldsymbol{M}$ have the following properties:

d	size	rank	nullity
1	3×6	3	3
2	9×12	9	3
3	18×20	17	3
4	30×30	17	3

For this example, we notice that the nullity of $\boldsymbol{M}$ has already stabilized for degree $d_* = 1$. If we want to shift with a cubic shift polynomial, the solution degree $d_\circ$ of $\boldsymbol{M}$ is equal to 4. Computing a numerical basis matrix $\boldsymbol{Z}$ of the null space and performing row-wise rank checks from top to bottom learn that the solution degree $d_\circ = 4$ results in a gap zone that can accommodate the shift ($*$ indicates a degree block without any additional linearly independent rows):



(a) Only affine solutions



(b) Affine solutions and solutions at infinity

Figure 3.3. Basis matrix of the null space of a block Macaulay matrix $\boldsymbol{M}$, which grows by multiplying with monomials of higher total degrees in the FSR (increasing degree d). At a certain degree d_* (in this example $d_* = 3$), the nullity stabilizes at the total number of solutions m_b. In the situation with only affine solutions (Figure 3.3a), the linearly independent rows of the basis matrix, checked from top to bottom, stabilize at their respective positions (indicated by dashed lines). New degree blocks contain no additional linearly independent rows when $d > d_*$. The basis matrix consists of a regular zone (■) and a gap zone (▨). However, when the system has solutions at infinity (Figure 3.3b), the linearly independent rows of the basis matrix that are related to the solutions at infinity (also indicated by dashed lines) move to higher degree blocks when $d > d_*$; they constitute the singular zone (■) of the basis matrix. A gap zone emerges in the rows that separates these two types of linearly independent rows, and the influence of the solutions at infinity can be deflated via a column compression.

Table 3.2. Numerical solutions and absolute residual errors[12] of Example 3.4 obtained via the null space based algorithm.

$\boldsymbol{\lambda_1}$	$\boldsymbol{\lambda_2}$	$\max\|\boldsymbol{e}\|_2$
0.9338	−1.3750	2.8×10^{-14}
1.3683	0.0552	6.0×10^{-15}
3.6026	−0.4183	2.6×10^{-14}

d	standard monomials
1	$z_1, z_2 \,\vert\, \lambda_1 z_1$
2	$z_1, z_2 \,\vert\, \lambda_1 z_1 \,\vert\, *$
3	$z_1, z_2 \,\vert\, \lambda_1 z_1 \,\vert\, * \,\vert\, *$
4	$z_1, z_2 \,\vert\, \lambda_1 z_1 \,\vert\, * \,\vert\, * \,\vert\, *$

We continue with a numerical basis matrix $\boldsymbol{Z} \in \mathbb{C}^{30\times 3}$ of its null space (for $d = 4$) and observe that the first three rows, which correspond to the variables z_1, z_2, and $\lambda_1 z_1$, are linearly independent. As the nullity is 3, there are no solutions at infinity. After constructing the shift matrices, we can use (3.71) to obtain the numerical solutions, which are shown in Table 3.2.

Code 3.6. It is possible to determine the degree of the gap zone and number of affine solutions via `gap(Z,d,n,l)`, but now the length of the eigenvector `l` is required to correctly iterate over the degree blocks.

```
>> [dgap,ma] = gap(Z,4,2,l)

dgap =
     2

ma =
     3
```

If there are only affine solutions, then there is no need for a column compression. `shiftnullspace(Z,shiftpoly,rows,l)` determines the solutions of the MEP.

[12] We calculate the absolute residual error by substituting the computed eigenvalues $(\lambda_1^*, \ldots, \lambda_n^*)$ and eigenvectors $\boldsymbol{z}^*$ in the MEP and determining the 2-norm of the residual vector $\|\boldsymbol{e}\|_2 = \|\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^*)\boldsymbol{z}^*\|_2$. More information about the error measures used in this text can be found in Appendix B.2.3.

```
>> D = shiftnullspace(Z,[4 0 3],rows,1);
>> solution = [D2, D3]

solution =
    0.9338   -1.3750
    1.3683    0.0552
    3.6026   -0.4183
```

Example 3.16. Next, to illustrate the influence of solutions at infinity, we consider the polynomial two-parameter eigenvalue problem from Example 3.5 and use a shift polynomial $g(\boldsymbol{\lambda}) = \lambda_1$. We iteratively build the block Macaulay matrix $\boldsymbol{M}$ for increasing degree $d = 2, \ldots, 5$ (as described in Algorithm 3.1) and obtain the following properties:

d	size	rank	nullity
2	3×12	3	9
3	9×20	9	11
4	18×30	18	12
5	30×42	30	12

The nullity remains stable from degree $d_* = 4$ onwards. When we compute a numerical basis matrix $\boldsymbol{Z}$ of the null space and perform row-wise rank checks from top to bottom, we notice that the null space contains three different zones for $d \geq 5$ ($*$ indicates a degree block without any additional linearly independent rows):

d	standard monomials
2	$z_1, z_2 \,\vert\, \lambda_1 z_1, \lambda_1 z_2, \lambda_2 z_1, \lambda_2 z_2 \,\vert\, \lambda_1^2 z_1, \lambda_1^2 z_2, \lambda_1\lambda_2 z_1$
3	$z_1, z_2 \,\vert\, \lambda_1 z_1, \lambda_1 z_2, \lambda_2 z_1, \lambda_2 z_2 \,\vert\, \lambda_1^2 z_1, \lambda_1^2 z_2, \lambda_1\lambda_2 z_1 \,\vert\, \lambda_1^3 z_1, \lambda_1^3 z_2$
4	$z_1, z_2 \,\vert\, \lambda_1 z_1, \lambda_1 z_2, \lambda_2 z_1, \lambda_2 z_2 \,\vert\, \lambda_1^2 z_1, \lambda_1^2 z_2, \lambda_1\lambda_2 z_1 \,\vert\, \lambda_1^3 z_1 \,\vert\, \lambda_1^4 z_1, \lambda_1^4 z_2$
5	$z_1, z_2 \,\vert\, \lambda_1 z_1, \lambda_1 z_2, \lambda_2 z_1, \lambda_2 z_2 \,\vert\, \lambda_1^2 z_1, \lambda_1^2 z_2, \lambda_1\lambda_2 z_1 \,\vert\, * \,\vert\, \lambda_1^4 z_1 \,\vert\, \lambda_1^5 z_1, \lambda_1^5 z_2$

A degree $d = 5$ block Macaulay matrix $\boldsymbol{M}$ suffices to solve this MEP with a linear shift polynomial and is, thus, equal to $d_\circ$. We can compute a numerical basis matrix $\boldsymbol{Z} \in \mathbb{C}^{42\times 12}$ of the null space (since we have $m_b = 12$ solutions) and determine the gap via row-wise rank checks from top to bottom. The gap indicates that this problem has $m_a = 9$ affine solutions and $m_\infty = 3$ solutions at infinity. After a column compression, we are left with $\boldsymbol{W}_{11} \in \mathbb{C}^{20\times 9}$. The $n + 1$ Schur decompositions in (3.69) give us the solutions in Table 3.3.

Code 3.7. For a MEP with solutions at infinity, the gap zone separates the regular zone and singular zone of the numerical basis matrix of the null space.

Table 3.3. Affine numerical solutions and absolute residual errors[12] of Example 3.5 obtained via the null space based algorithm.

$\boldsymbol{\lambda_1}$	$\boldsymbol{\lambda_2}$	$\mathbf{max}\|e\|_2$
$1.4027 \pm 0.3941\mathrm{i}$	$-1.3835 \mp 0.8431\mathrm{i}$	5.1×10^{-14}
$-0.9699 \pm 0.7168\mathrm{i}$	$-0.1113 \pm 0.5741\mathrm{i}$	1.1×10^{-14}
$0.8543 + 0.0000\mathrm{i}$	$-0.9341 + 0.0000\mathrm{i}$	2.8×10^{-15}
$0.2737 + 0.0751\mathrm{i}$	$-0.1917 \mp 0.2408\mathrm{i}$	1.0×10^{-14}
$-0.4497 \pm 0.0662\mathrm{i}$	$0.6094 \mp 1.0534\mathrm{i}$	7.6×10^{-14}

```
>> Z = null(macaulay(toymep2,5));
>> [dgap,ma] = gap(Z,5,2,1)

dgap =
    3

ma =
    9
```

In this case, a column compression is necessary to retrieve the solutions.

```
>> W11 = columncompression(Z,n,dgap);
>> D = shiftnullspace(W11,[randn(2,1), eye(2)],rows,1);
>> solution = [D2, D3]

solution =
    1.4027 + 0.3941i    -1.3835 - 0.8431i
    1.4027 - 0.3941i    -1.3835 + 0.8431i
   -0.9699 + 0.7168i    -0.1113 + 0.5741i
   -0.9699 - 0.7168i    -0.1113 - 0.5741i
    0.8543 + 0.0000i    -0.9341 + 0.0000i
    0.2737 + 0.0751i    -0.1917 - 0.2408i
    0.2737 - 0.0751i    -0.1917 + 0.2408i
   -0.4497 + 0.0662i     0.6094 - 1.0534i
   -0.4497 - 0.0662i     0.6094 + 1.0534i
```

3.4.4 Null space based eigenvalue-finding algorithm

We summarize the different steps of the null space based eigenvalue-finding algorithm in Algorithm 3.2.

Algorithm 3.2 Null space based approach eigenvalue-finding algorithm

Require: Block Macaulay matrix $\boldsymbol{M}$ of degree $d_\circ$ (Algorithm 3.1)

1: Compute a numerical basis matrix $\boldsymbol{Z}$ of the null space of $\boldsymbol{M}$
2: Determine the gap and the number of affine solutions m_a via row-wise rank checks from top to bottom in $\boldsymbol{Z}$ (Section 3.4.3)
3: Use Theorem 2.3 to obtain the compressed numerical basis matrix $\boldsymbol{W}_{11}$ of the null space
4: For a (user-defined) shift polynomial $g_0(\boldsymbol{\lambda})$, solve the Schur decomposition

$$\boldsymbol{Q}\boldsymbol{D}_{g_0}\boldsymbol{Q}^{-1} = (\boldsymbol{S}_1\boldsymbol{Z})^{-1}(\boldsymbol{S}_{g_0}\boldsymbol{Z}), \tag{3.72}$$

where the matrices $\boldsymbol{S}_1$, $\boldsymbol{S}_{g_0}$, $\boldsymbol{Q}$, and $\boldsymbol{D}_{g_0}$ are defined as in (3.68)
5: Retrieve the different components $\lambda_i|_{(j)}$ of the solutions from $\boldsymbol{D}_{\lambda_i}$ in (3.69)
6: **return** $\boldsymbol{\lambda}|_{(j)}$, for $j = 1, \ldots, m_a$

Remark 3.4. To fully exploit the block multi-shift-invariance of the null space, the row selection matrix $\boldsymbol{S}_1 \in \mathbb{R}^{l\times q}$ can select entire block rows of $\boldsymbol{Z}$ (with at least m_a linearly independent rows to cover all the affine solutions):

$$\boldsymbol{Q}\boldsymbol{D}_{g}\boldsymbol{Q}^{-1} = (\boldsymbol{S}_1\boldsymbol{Z})^{\dagger}(\boldsymbol{S}_{g}\boldsymbol{Z}). \tag{3.73}$$

where the row combination matrix $\boldsymbol{S}_g \in \mathbb{R}^{l\times q}$ also selects entire block rows of $\boldsymbol{Z}$. The pseudo-inverse $(.^\dagger)$ is used to obtain a solvable SEP. Shifting entire blocks replaces the row-wise rank checks by more efficient block-wise rank checks, which is interesting for both the numerical robustness and the computational efficiency of the algorithm.

Example 3.17. To wrap up the null space based approach, we consider the quadratic three-parameter eigenvalue problem constructed in [148] from a model order reduction problem [101],

$$\begin{aligned}\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \Big(&\boldsymbol{A}_{000} + \boldsymbol{A}_{100}\lambda_1 + \boldsymbol{A}_{010}\lambda_2 + \boldsymbol{A}_{001}\lambda_3 \\ &+ \boldsymbol{A}_{200}\lambda_1^2 + \boldsymbol{A}_{110}\lambda_1\lambda_2 + \boldsymbol{A}_{101}\lambda_1\lambda_3 + \boldsymbol{A}_{020}\lambda_2^2 \\ &+ \boldsymbol{A}_{011}\lambda_2\lambda_3 + \boldsymbol{A}_{002}\lambda_3^2\Big)\boldsymbol{z} = \boldsymbol{0},\end{aligned} \tag{3.74}$$

the 10×8 coefficient matrices of which are given in (3.91) to (3.100) at the end of this chapter. Via Algorithm 3.1, we determine that the solution degree $d_\circ = 11$. The block Macaulay matrix for this degree is $\boldsymbol{M} \in \mathbb{C}^{2200\times 2912}$. Rank tests on the basis matrix of the null space of $\boldsymbol{M}$ reveal that this problem has a positive-dimensional solution set at infinity, but the affine part of the solution set is zero-dimensional (cf., Section 2.6.1). So, the solutions at infinity can be deflated via a column compression to construct $\boldsymbol{W}_{11} \in \mathbb{C}^{1760\times 209}$ and solve the different Schur decompositions in (3.68). This MEP can be solved in 361.44 s (averaged over 30 experiments) via the row-wise approach,

while it only takes 3.04 s (averaged over 30 experiments) via the block-wise approach of Remark 3.4. The absolute residual errors[12] of the row-wise approach are (in most of the experiments) very bad, due to possible wrong rank decisions, while the maximum absolute residual error of the block-wise approach is equal to 4.41×10^{-8} (averaged over 30 experiments).

Remark 3.5. A correct selection of the linearly independent rows in the row-wise approach is essential for obtaining satisfactory solutions (cf., Remark 2.6). The previous example illustrates that wrong rank decisions have fatal consequences for the algorithm. The block-wise approach is numerically more robust.

3.5 Column space based approach

In this section, we consider the column space of the block Macaulay matrix instead of its null space. The intrinsic complementarity between both fundamental subspaces (Section 2.5.1) enables a new, complementary algorithm to solve MEPs, which works directly on the data in the columns (Section 3.5.1). After explaining the approach, we summarize the different steps of the column space based algorithm (Section 3.5.2).

3.5.1 Equivalent column space realization theory

Consider, for an MEP with zero-dimensional solution set, again a block Macaulay matrix $\boldsymbol{M} \in \mathbb{C}^{p \times q}$, with degree equal to its solution degree $d = d_\circ$, and a numerical basis matrix $\boldsymbol{W} \in \mathbb{C}^{q \times m_b}$ of its null space after a column compression (Theorem 2.3). A shift of the linearly independent rows of the compressed basis matrix $\boldsymbol{W}_{11}$ with a shift polynomial $g(\boldsymbol{\lambda})$ results in the expression

$$(\boldsymbol{S}_g \boldsymbol{W}_{11})\boldsymbol{T} = (\boldsymbol{S}_1 \boldsymbol{W}_{11})\boldsymbol{T}\boldsymbol{D}_g, \tag{3.75}$$

where the matrices $\boldsymbol{S}_1$, $\boldsymbol{S}_g$, $\boldsymbol{T}$, and $\boldsymbol{D}_g$ are defined as in (3.60). After introducing two new matrices $\boldsymbol{B} \in \mathbb{R}^{m_a \times m_a}$, which contains all the linearly independent rows of $\boldsymbol{W}_{11}$, and $\boldsymbol{C} \in \mathbb{R}^{m_c \times m_a}$, with the linear combinations of rows hit by the shift that contain a row not present in $\boldsymbol{B}$, as in Section 2.5.2, we can re-write (3.75) as

$$\begin{bmatrix} \boldsymbol{S}'_g \\ \boldsymbol{C}\boldsymbol{B}^{-1} \end{bmatrix} \boldsymbol{B}\boldsymbol{T} = \boldsymbol{B}\boldsymbol{T}\boldsymbol{D}_g, \tag{3.76}$$

where $\boldsymbol{S}'_g \in \mathbb{R}^{m_h \times m_a}$ is the row combination matrix that constructs the linear combinations of rows hit inside $\boldsymbol{B}$. The matrix $\boldsymbol{B}$ is invertible because it contains m_a linearly independent rows by construction. In this section, we translate (3.76) to the column space via Lemma 2.3, avoiding the computation of a numerical basis matrix of the null space.

Example 3.18. When we consider again the MEP (3.11), we start by identifying the linearly dependent columns (from right to left) of $\boldsymbol{M}$. Like in Example 3.15, we notice that $d_* = 1$ and that the three left-most columns are linearly dependent on the other columns (from right to left), which means that the first three rows of $\boldsymbol{W}_{11}$ (i.e., z_1, z_2 and $\lambda_1 z_1$) constitute $\boldsymbol{B}$. The rows in $\boldsymbol{C}$ depend on the particular shift:

- For a shift with λ_1, the rows that correspond to $\lambda_1 z_1$, $\lambda_1 z_2$, and $\lambda_1^2 z_1$ are hit. This means that $\boldsymbol{C}$ contains the 4th and 7th row (i.e., $\lambda_1 z_2$ and $\lambda_1^2 z_1$). The matrix $\boldsymbol{B}$ could be split into $\boldsymbol{B}_1$ with the first row and $\boldsymbol{B}_2$ with the second and third row. The row selection matrix $\boldsymbol{S}'_g$ selects the third row from $\boldsymbol{B}$.
- When choosing $g(\boldsymbol{\lambda}) = 4\lambda_2^3$ as in Example 3.14, $\boldsymbol{B}_1$ is empty and $\boldsymbol{C}$ contains 4 times the 19th, 20th, and 27th row.

We define, similar as in Section 2.5.2, the matrix $\boldsymbol{D} \in \mathbb{C}^{m_r \times m_a}$ (with $m_r = s - m_a - m_c$) as the matrix that contains the remaining rows of $\boldsymbol{W}_{11}$, such that every row of $\boldsymbol{W}_{11}$ is *represented* once in $\boldsymbol{B}$, $\boldsymbol{C}$, or $\boldsymbol{D}$. Consequently, we can reorder the basis matrix $\boldsymbol{W}$ as

$$\boldsymbol{P}\boldsymbol{W} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} \end{bmatrix}, \tag{3.77}$$

where the matrix $\boldsymbol{P}$ is a $q \times q$ row combination matrix that is invertible and does not alter the rank structure of $\boldsymbol{W}$. Using Lemma 2.3, we can reorder the columns of the block Macaulay matrix in accordance to $\boldsymbol{P}\boldsymbol{W}$ and obtain

$$\underbrace{\begin{bmatrix} \boldsymbol{N}_1 & \boldsymbol{N}_2 & \boldsymbol{N}_3 & \boldsymbol{N}_4 \end{bmatrix}}_{\boldsymbol{N}} \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} \end{bmatrix} = \boldsymbol{0}, \tag{3.78}$$

where every matrix $\boldsymbol{N}_i \in \mathbb{C}^{p \times q_i}$ corresponds to a subset of the columns (or linear combinations of columns) of $\boldsymbol{M}$. We call $\boldsymbol{N} = \boldsymbol{M}\boldsymbol{P}^{-1} \in \mathbb{C}^{p \times q}$ the **reordered block Macaulay matrix**. Now, we apply a backward (Q-less) QR decomposition on $\boldsymbol{N}$, which yields[13]

$$\begin{array}{c} \\ q-s \\ m_r \\ m_c \\ p-q+m_a \end{array} \begin{array}{c} \begin{array}{cccc} m_a & m_c & m_r & q-s \end{array} \\ \begin{bmatrix} \boldsymbol{R}_{14} & \boldsymbol{R}_{13} & \boldsymbol{R}_{12} & \boldsymbol{R}_{11} \\ \boldsymbol{R}_{24} & \boldsymbol{R}_{23} & \boldsymbol{R}_{22} & \boldsymbol{0} \\ \boldsymbol{R}_{34} & \boldsymbol{R}_{33} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{R}_{44} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \end{array} \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} \\ \boldsymbol{C} & \boldsymbol{0} \\ \boldsymbol{D} & \boldsymbol{0} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} \end{bmatrix} = \boldsymbol{0}, \tag{3.79}$$

[13] A similar analysis of the upper triangular matrix $\boldsymbol{R}$ as in Footnote 19 of Chapter 2 reveals that $\boldsymbol{N}_4$ is not present in $\boldsymbol{N}$ when there are no solutions at infinity (Example 3.18) and that the size of the block Macaulay matrix $\boldsymbol{M}$ determines the size of $\boldsymbol{R}_{44}$ (Example 3.19).

where the labels denote the number of rows/columns of the different blocks. We notice that $\boldsymbol{R}_{33}\boldsymbol{C} = -\boldsymbol{R}_{34}\boldsymbol{B}$, which means that

$$\boldsymbol{C}\boldsymbol{B}^{-1} = -\boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34}, \tag{3.80}$$

because $\boldsymbol{R}_{33}$ is always of full rank. Note that $\boldsymbol{R}_{44}$ is always a zero matrix. Relation (3.80) helps to remove the dependency on the null space in (3.76) and yields a solvable SEP in the column space (with $\boldsymbol{H} = \boldsymbol{B}\boldsymbol{T}$),

$$\begin{bmatrix} \boldsymbol{S}_g' \\ -\boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34} \end{bmatrix} \boldsymbol{H} = \boldsymbol{H}\boldsymbol{D}_g, \tag{3.81}$$

or a GEP (to avoid the computation of the inverse of $\boldsymbol{R}_{33}$),

$$\begin{bmatrix} \boldsymbol{S}_g' \\ -\boldsymbol{R}_{34} \end{bmatrix} \boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}_{m_h} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_{33} \end{bmatrix} \boldsymbol{H}\boldsymbol{D}_g, \tag{3.82}$$

with $\boldsymbol{I}_{m_h} \in \mathbb{N}^{m_h \times m_h}$ the identity matrix. The matrix of eigenvector $\boldsymbol{H} = \boldsymbol{B}\boldsymbol{T}$ corresponds to the partitioned linearly independent rows of the (affine) block multivariate Vandermonde basis matrix $\boldsymbol{V}$, because the nonsingular transformation matrix $\boldsymbol{T}$ relates the rows of the numerical basis matrix $\boldsymbol{W}_{11}$ (or $\boldsymbol{B}$) to the rows of $\boldsymbol{V}$. Consequently, the eigenvalues in $\boldsymbol{D}_g$ and eigenvectors in $\boldsymbol{H}$ yield the solutions of the MEP. Note that this complementary column space based approach does not require a column compression to deflate the solutions at infinity, because the backward (Q-less) QR decomposition already separates them implicitly.

Example 3.18 (continuing from p. 131). We now continue the previous example and try to set up the GEP in (3.82) for a shift with $g(\boldsymbol{\lambda}) = 4\lambda_2^3$. The corresponding row combination matrix $\boldsymbol{P} \in \mathbb{R}^{30\times 30}$ is

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{I}_3 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & 4\boldsymbol{I}_2 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 4\boldsymbol{I}_1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{15} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_6 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_3 \end{bmatrix}, \tag{3.83}$$

where the first three rows of $\boldsymbol{P}$ select the linearly independent rows of $\boldsymbol{B}$, the next three rows of $\boldsymbol{P}$ select the rows of $\boldsymbol{C}$ (notice the factor 4), and the remaining rows of $\boldsymbol{P}$ result in $\boldsymbol{D}$. We do not multiply $\boldsymbol{W}$ by $\boldsymbol{P}$, but we use $\boldsymbol{P}^{-1}$ to reorder the columns of $\boldsymbol{M}$ into the matrices $\boldsymbol{N}_1$, $\boldsymbol{N}_2$, and $\boldsymbol{N}_3$. Since there are no solutions at infinity, we do not have a matrix $\boldsymbol{N}_4$ (Footnote 13). The backward (Q-less) QR decomposition of $\boldsymbol{N} = \boldsymbol{M}\boldsymbol{P}^{-1}$ results in the matrix

$$\boldsymbol{R} = \begin{array}{r} \\ 24 \\ 3 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 3 & 3 & 24 \end{array} \\ \begin{bmatrix} \boldsymbol{R}_{24} & \boldsymbol{R}_{23} & \boldsymbol{R}_{22} \\ \boldsymbol{R}_{34} & \boldsymbol{R}_{33} & \boldsymbol{0} \\ \boldsymbol{R}_{44} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \end{array}, \tag{3.84}$$

from which the matrices $\boldsymbol{R}_{33}$ and $\boldsymbol{R}_{34}$ of (3.82) can be extracted. The GEP yields the same matrix $\boldsymbol{D}_g$ as in Example 3.14:

$$\boldsymbol{D}_g = \begin{bmatrix} -10.3979 & 0 & 0 \\ 0 & 0.0007 & 0 \\ 0 & 0 & -0.2928 \end{bmatrix}. \tag{3.85}$$

Code 3.8. Similar as in Code 2.9, a row combination matrix can be built for the block Macaulay matrix. Of course, the length of the eigenvector `l = 2` is required.

```
>> P = rowcombinationmatrix(4,2,rows,[4 0 3],l);
>> N = M*inv(P);
>> [~,R] = qr(fliplr(N)); R = fliplr(R);
>> R33 = R(25:27,4:6); R34 = R(25:27,1:3);
>> eig(-inv(R33)*R34)

ans =
   -10.3979
    0.0007
   -0.2928
```

Example 3.19. Next, we revisit the MEP in Example 3.5. Obviously, checking the block Macaulay matrix at degree $d_\circ = 5$ for linearly dependent columns from right to left results in a similar analysis as before. Because of the backward (Q-less) QR decomposition, we do not need to deflate the solutions at infinity via a column compression. Note that $\boldsymbol{M}$ is not yet tall and $p = q - m_b < q - m_a$ (Footnote 13). We have two options to proceed:

- We can consider a block Macaulay matrix of a larger degree, since the matrix will eventually become tall.
- We can remove the linearly dependent columns that correspond to solutions at infinity in $\boldsymbol{N}_4$.

We choose in this numerical example the latter option and remove the 21st ($\lambda_1^4 z_1$), 31st ($\lambda_1^5 z_1$), and 32nd ($\lambda_1^5 z_2$) column from $\boldsymbol{M}$ when splitting into $\boldsymbol{N}_1$, $\boldsymbol{N}_2$, $\boldsymbol{N}_3$, and $\boldsymbol{N}_4$. The numerical results are the same as with the null space based approach (Table 3.3).

In the previous example, any shift polynomial with a power of λ_2 yields a perfectly reconstructible solution. But as mentioned in Remark 2.7, the situation is sometimes more difficult, because the matrix $\boldsymbol{H}$ in the column space based approach does not necessarily contain all the eigenvalue parameters. The

same strategy as for the column space based root-finding approach can be used: shift with a random shift polynomial $g_0(\boldsymbol{\lambda})$,

$$\boldsymbol{Q}\boldsymbol{D}_{g_0}\boldsymbol{Q}^{\mathrm{H}} = \begin{bmatrix} \boldsymbol{S}'_{g_0} \\ \boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34} \end{bmatrix}, \tag{3.86}$$

and the n different variables $g_i(\boldsymbol{\lambda}) = \lambda_i$,

$$\begin{aligned} \boldsymbol{Q}\boldsymbol{D}_{\lambda_1}\boldsymbol{Q}^{\mathrm{H}} &= \begin{bmatrix} \boldsymbol{S}'_{\lambda_1} \\ \boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34} \end{bmatrix}, \\ &\vdots \\ \boldsymbol{Q}\boldsymbol{D}_{\lambda_n}\boldsymbol{Q}^{\mathrm{H}} &= \begin{bmatrix} \boldsymbol{S}'_{\lambda_n} \\ \boldsymbol{R}_{33}^{-1}\boldsymbol{R}_{34} \end{bmatrix}, \end{aligned} \tag{3.87}$$

which are similar Schur decompositions as in Section 3.4.2. The matrices $\boldsymbol{R}_{33}$ and $\boldsymbol{R}_{34}$ have to be re-computed for every shift.

Example 3.20. To illustrate, let us consider the following linear three-parameter eigenvalue problem:

$$(\boldsymbol{A}_{000} + \boldsymbol{A}_{100}\lambda_1 + \boldsymbol{A}_{010}\lambda_2 + \boldsymbol{A}_{001}\lambda_3)\boldsymbol{z} = \boldsymbol{0}, \tag{3.88}$$

with four coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{C}^{4\times 2}$,

$$\boldsymbol{A}_{000} = \begin{bmatrix} 2 & 3 \\ 2 & 5 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \boldsymbol{A}_{100} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{bmatrix}, \boldsymbol{A}_{010} = \begin{bmatrix} 4 & 2 \\ 2 & 3 \\ 3 & 1 \\ 3 & 1 \end{bmatrix}, \text{ and } \boldsymbol{A}_{001} = \begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 2 & 1 \\ 4 & 2 \end{bmatrix}.$$

When we check the columns of the block Macaulay matrix $\boldsymbol{M}$ of degree $d = d_\circ = 2$ from right to left, we observe that the first four columns (which correspond to z_1, z_2, $\lambda_1 z_1$, and $\lambda_1 z_2$) are linearly dependent on the other columns. Thus, the matrix $\boldsymbol{H}$ contains references to the variables z_1, z_2, and λ_1 evaluated in each of the affine solutions, but no references to λ_2 or λ_3. Hence, not one, but two shift polynomials (with references to λ_2 and λ_3) are required to find all the components of the solutions via the eigenvalues of two different GEPs. We can shift in this numerical example with $g_1(\lambda_1, \lambda_2, \lambda_3) = \lambda_2$ and $g_2(\lambda_1, \lambda_2, \lambda_3) = \lambda_3$ to obtain also the two remaining eigenvalues in $\boldsymbol{D}_{g_1}$ and $\boldsymbol{D}_{g_2}$. In order to match the different eigenvalues λ_2 and λ_3, we can use the matrix of eigenvectors $\boldsymbol{H}$ in both GEPs (because this problem only has simple solutions) or re-use the matrix $\boldsymbol{Q}$ of the Schur decomposition.

3.5.2 Column space based eigenvalue-finding algorithm

We summarize the different steps of the column space based eigenvalue-finding algorithm in Algorithm 3.3.

Algorithm 3.3 Column space based eigenvalue-finding algorithm

Require: Block Macaulay matrix $\boldsymbol{M}$ of degree $d_\circ$ (Algorithm 3.1)

1: Replace $\boldsymbol{M}$ by the flipped upper triangular matrix $\boldsymbol{R}$ of its QR decomposition (optional)
2: Determine the linearly dependent columns via rank checks from right to left and reorder $\boldsymbol{M}$ or $\boldsymbol{R}$ as in (3.78)
3: Compute the (Q-less) backward QR decomposition of the reorderd block Macaulay matrix $\boldsymbol{N}$
4: For a user-defined shift polynomial $g_0(\boldsymbol{\lambda})$, solve the Schur decomposition

$$\boldsymbol{Q}\boldsymbol{D}_{g_0}\boldsymbol{Q}^{-1} = \begin{bmatrix} \boldsymbol{I}_{m_h} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_{33} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{S}'_g \\ -\boldsymbol{R}_{34} \end{bmatrix}, \tag{3.89}$$

where the matrices $\boldsymbol{S}'_g$, $\boldsymbol{R}_{34}$, $\boldsymbol{R}_{34}$, $\boldsymbol{Q}$, and $\boldsymbol{D}_{g_0}$ are defined as in (3.82) and (3.86)
5: Retrieve the different eigenvalues $\lambda_i|_{(j)}$ of the solutions from $\boldsymbol{D}_{\lambda_i}$ in (3.87)
6: **return** $\boldsymbol{\lambda}|_{(j)}$, for $j = 1, \ldots, m_a$

Remark 3.6. Similar to Remark 2.8 from the Macaulay matrix, the inverse permutation also reduces to a simple column re-ordering procedure when the shift polynomial $g(\boldsymbol{\lambda})$ is a monomial of (some of) the eigenvalues.

Example 3.21. We repeat Example 3.17 and solve (3.74) via the column space of the block Macaulay matrix. The solution degree $d_\circ = 11$ and the constructed block Macaulay matrix $\boldsymbol{M}$ is of size 2200×2912. It is possible to replace $\boldsymbol{M}$ by the flipped upper triangular matrix $\boldsymbol{R} \in \mathbb{C}^{2200\times 2912}$ of its QR decomposition, but this leads not to a size reduction in this example. The rank tests, on the columns of the block Macaulay matrix, reveal again that the system has 209 affine solutions and a positive-dimensional solution set at infinity. Since we only use linear shift polynomials, we do not build $\boldsymbol{P}$ explicitly, but re-order the columns of $\boldsymbol{M}$ instead. This MEP can be solved in 1206.15 s (averaged over 30 experiments), which is much slower than via the null space. The maximum absolute residual error[12] of this approach is 1.11×10^{-8} when the column-wise rank checks succeed (cf., row-wise approach in Example 3.17).

3.6 Conclusion

We introduced the block Macaulay matrix, which is an extension of the traditional (scalar) Macaulay matrix from polynomial system solving to the multi-parameter eigenvalue setting. We exploited the complementarity between the right null space and column space of the block Macaulay matrix to propose two

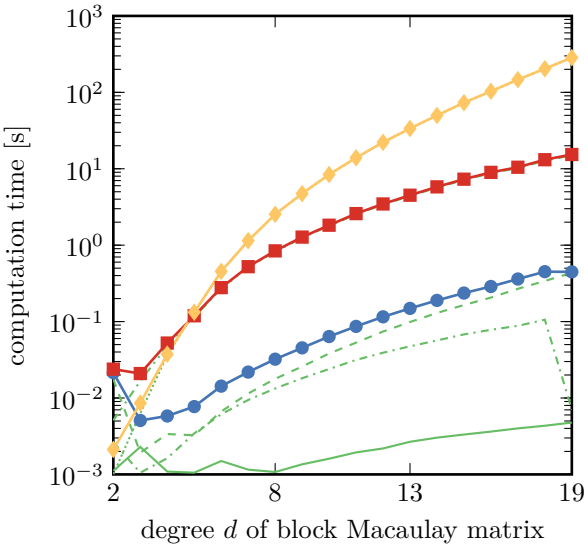


Figure 3.4. Mean computation time per degree for the first-order ARMA model identification problem with $N = 4$ data points, using the block-wise null space based approach (—●—), row-wise null space based approach (—■—), and column-wise column space based approach (—◆—), averaged over 30 experiments (without standard deviations to keep the figure readable). We have also added the necessary mean computation time to construct the Macaulay matrix (——), to compute a basis matrix for the null space (- - -), and to perform the different rank checks, when performed block-wise (-·-·-) or row-wise (·····) on the basis matrix of the null space or column-wise (-···-) on the block Macaulay matrix.

approaches to solve (rectangular) multiparameter eigenvalue problems. The null space based approach yields, after deflating the solutions at infinity with a column compression, the affine eigenvalues of the problem. Contrary to the null space based approach, the column space based approach does not require an explicit computation of a numerical basis matrix of the right null space, but considers the data in the columns of the block Macaulay matrix directly. The influence of the solutions at infinity is removed implicitly in the column space based approach via a backward (Q-less) QR decomposition. Note that the extensions to the root-finding algorithms discussed in Section 2.6 can also be incorporated in both block Macaulay matrix eigenvalue-finding algorithms: these algorithms can also (i) deal with a positive-dimensional solution set at infinity, (ii) use special shift polynomials, and (iii) work with a different polynomial basis or monomial ordering.

Motivational example. Let us consider the first-order ARMA model identification problem for the same data sequence $\boldsymbol{y} \in \mathbb{R}^{4\times 1}$ as in (2.124). We solve the quadratic two-parameter eigenvalue problem,

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \left(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{01}\lambda_2 + \boldsymbol{A}_{02}\lambda_2^2\right)\boldsymbol{z} = \boldsymbol{0}, \tag{3.90}$$

with 11×10 coefficient matrices as described in Section 7.5. One of the solutions of this MEP corresponds to the globally optimal parameters of the ARMA model, namely $\alpha_1 = 0.3817$ and $\gamma_1 = -0.5789$. Both the null space based and column space based solution approach find this solution. Note that this MEP has a positive-dimensional solution set at infinity.

Similar to Table 2.2, the block-wise null space based approach for solving the MEP is much faster than the row-wise null space based approach and column-wise column space based approach. We compare (the iterative part of) the different solution approaches in Figure 3.4. Using block-wise rank checks clearly has a computational advantage in this example, while the numerical robustness is not clear from this MEP (there are no difficult rank checks). These block-wise rank checks are not yet available in the column space based approach, resulting in a slower execution. Furthermore, the fact that the Macaulay matrix is tall results more expensive rank checks in the column space based approach, compared to the row-wise null space based approach, while the number of rank checks is the same. The current column space based approach does not yet exploit the fact that the Macaulay matrix is very sparse and structured.

Notice that it takes more time to solve the first-order ARMA model identification problem for $N = 4$ data points via the block Macaulay matrix approaches than via the Macaulay matrix approaches. However, for larger N, the reformulation as an MEP scales better than the reformulation as a polynomial system.

When the coefficient matrices grow, the computational complexity of both algorithms increases rapidly. In particular the rank checks to determine the linearly independent rows of the basis matrix or the linearly dependent columns of the block Macaulay matrix are computationally expensive. This observation opens several new research paths:

- The backward (Q-less) QR decomposition, which constitutes the core of the column space based approach, has created several algorithmic opportunities. We want to improve the column space based algorithm by exploiting the sparsity and structure of the block Macaulay matrix, by considering block columns instead of columns (i.e., by taking fully advantage of the backward block multi-shift-invariance), by looking into rank-revealing QR decompositions, and by developing recursive techniques.

- Furthermore, the complementarity between both fundamental subspaces may even yield more useful properties in the column space. Together with a better understanding of MEPs, these properties could give us the machinery to tackle much larger problems in the future.

- Advancements for the Macaulay matrix for polynomial system solving are also an inspiration for the block Macaulay matrix algorithms.

- For square MEPs, subspace techniques make the transition from small problems to large problems possible. Several subspace algorithms have been explored for this type of problems, see, for example, [117, 121, 122].

This immediately fuels the next question: "Can we also develop subspace methods for rectangular MEPs?" The answer is *yes*! We are currently developing subspace techniques for rectangular MEPs. Preliminary results are quite promising, so we hope that subspace techniques will enable us to also solve large problems with the block Macaulay matrix algorithms.

Historical and bibliographical notes

Below, we give some additional historical and bibliographical notes about solving (one-parameter and square multiparameter) eigenvalue problems.

One-parameter eigenvalue problems

There exist many techniques to solve SEPs and GEPs, with adaptations to tackle very large matrices when there is structure or sparsity involved [213, 270]. When the problem is not square, a different solution approach is necessary, e.g., a combination with projections, as in [110, 281].

One of the most common strategies for solving a PEP is via a linearization, which replaces the matrix polynomial by a matrix pencil with the same spectrum, and then computes the eigenvalues of the linearized pencil [113, 161, 162, 245]. Many matrix polynomials arising from applications have additional structure, leading to symmetries in the spectrum that are important for computational methods to respect. It is useful to employ a structured linearization for a matrix polynomial with structure [161, 162, 245].

Square multiparameter eigenvalue problems

Square MEPs are typically solved via simultaneous triangularization (i.e., via the QZ algorithm) of the associated system of coupled GEPs [117, 126, 198, 218]. This approach works for any number of spectral parameters and retrieves all the solutions, but is limited by the size of the matrices in the system of coupled GEPs. Also iterative nonlinear optimization algorithms can be used to retrieve one (or some) of the solutions, e.g., gradient descent techniques [36, 37, 48], minimal residual quotient iterations [35], or Newton-based methods [39], but these optimization approaches are heuristic (they depend on an initial guess) and result in numerical approximations of the eigenvalues and eigenvectors.

In the last two decades, a renewed interest in the topic has led to several efficient homotopy continuation algorithms [77, 196, 206] and subspace approaches [117, 119, 121, 122, 212] to overcome issues with scalability and convergence. These algorithms can also solve polynomial square MEPs, either directly or after a linearization step [120, 206].

The problems that we discuss in this chapter are polynomial problems. However, there exists an even more general class of problems: nonlinear MEPs, which arise, for example, in the computation of critical delays of delay-differential equations with multiple delays [199]. Some approaches to deal with (square) nonlinear MEPs are outlined in [199].

Coefficient matrices of Example 3.17

The coefficient matrices of the MEP in (3.74) are

$$\boldsymbol{A}_{000} = \begin{bmatrix} b_1 & -a_1 & 0 & 0 & -1 & 0 & 0 & 0 \\ b_2 & -a_2 & -a_1 & 0 & 0 & -1 & 0 & 0 \\ b_3 & -a_3 & -a_2 & -a_1 & 0 & 0 & -1 & 0 \\ b_4 & -a_4 & -a_3 & -a_2 & 0 & 0 & 0 & -1 \\ b_5 & -a_5 & -a_4 & -a_3 & 0 & 0 & 0 & 0 \\ b_6 & -a_6 & -a_5 & -a_4 & 0 & 0 & 0 & 0 \\ b_7 & -a_7 & -a_6 & -a_5 & 0 & 0 & 0 & 0 \\ 0 & -a_8 & -a_7 & -a_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & -a_8 & -a_7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -a_8 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.91}$$

$$\boldsymbol{A}_{100} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ b_2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ b_3 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ b_4 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ b_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.92}$$

$$\boldsymbol{A}_{010} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ b_2 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ b_3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 \\ b_4 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \\ b_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.93}$$

$$\boldsymbol{A}_{001} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ b_2 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ b_3 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ b_4 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ b_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ b_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.94}$$

$$\boldsymbol{A}_{200} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{3.95}$$

$$\boldsymbol{A}_{110} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.96}$$

$$\boldsymbol{A}_{101} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.97}$$

$$\boldsymbol{A}_{020} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.98}$$

$$\boldsymbol{A}_{011} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.99}$$

and

$$\boldsymbol{A}_{002} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.100}$$

The elements a_i and b_i are elements of the vectors

$$\boldsymbol{a} = \begin{bmatrix} 1 & 10 & 46 & 130 & 239 & 280 & 194 & 60 \end{bmatrix}^{\mathrm{T}} \tag{3.101}$$

and

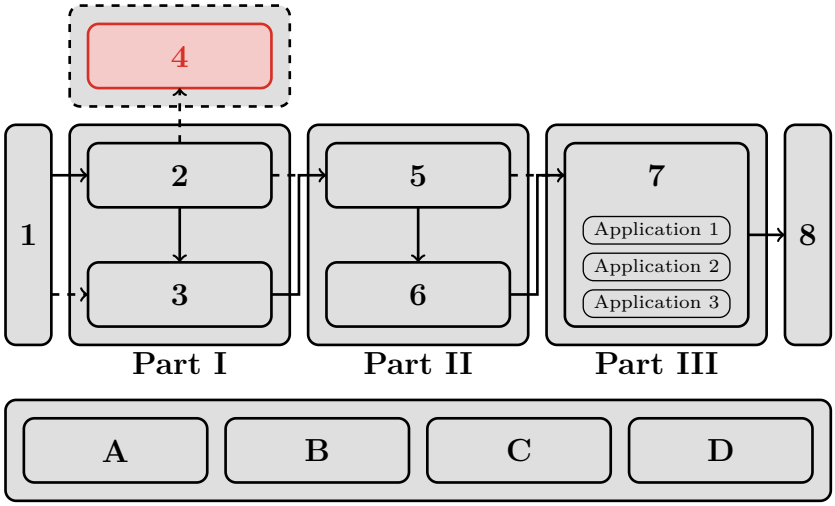
$$\boldsymbol{b} = \begin{bmatrix} 2 & 11.5 & 57.75 & 178.625 & 345.5 & 323.625 & 94.5 \end{bmatrix}^{\mathrm{T}}, \tag{3.102}$$

respectively.

4

Fundamental Subspaces of the Macaulay Matrix

As mentioned in Chapter 2, each of the four fundamental subspaces of the Macaulay matrix has a profound algebraic connection with its generating polynomials. In this chapter, we visit each of these fundamental subspaces and interpret them in the language of algebraic geometry. We show how basic linear algebra concepts, like linear independence and matrix rank, coincide with the properties of the polynomials in the Macaulay matrix. Throughout Chapter 2, the Macaulay matrix has always defined for a certain degree and it is important to understand how the dimensions and interpretations of its fundamental subspaces change as a function of this degree.



Contribution. We give an overview of the different fundamental subspaces of the Macaulay matrix: revisiting, summarizing, and extending the different algebraic interpretations of this matrix. Notice that we gather in this chapter contributions of numerous other authors.

Outline. The introduction in Section 4.1 presents the different fundamental subspaces of the Macaulay matrix. Before we can consider the four different fundamental subspaces, we give some essential information about the projective space in Section 4.2. Subsequently, we focus on each of the fundamental subspaces separately: In Section 4.3, we consider the row space of the Macaulay matrix and provide an interpretation of that subspace. Section 4.4 is about the left null space and the link with the occurrence of syzygies in the row space, which gives an expression for the degree of regularity. Section 4.5 contains an interpretation of the null space based root-finding algorithm in the language of algebraic geometry. The complementarity between the right null space and column space creates the possibility for an interpretation of the column space in Section 4.6. Finally, in Section 4.7, we draw some conclusions and point at ideas for future work.

4.1 Introduction

Throughout this dissertation, the Macaulay matrix (and its block extension) play an important role. The Macaulay matrix is generated by a system of multivariate polynomials. One can only image that some of the properties of these polynomials should transfer to this matrix. Indeed, the linear algebra properties of the Macaulay matrix are deeply intertwined with the polynomials that generate it. We can use the Macaulay matrix for various problems, such as elimination of variables [27], computing an approximate greatest common divisor [23], computing a Gröbner and border basis [26], solving the ideal membership problem [26], and finding the affine/projective common roots of a polynomial system [24, 258].

In this chapter, we approach the Macaulay matrix and the generating polynomials from a linear algebra point of view. We relate basic linear algebra concepts, like linear independence and matrix rank, with concepts from algebraic geometry. The key objects that we consider are the four fundamental subspaces of the Macaulay matrix, connected with each other through two important theorems from linear algebra. We want to show that many question about the polynomials can be answered via the Macaulay matrix, and that the answers are often hiding in one of its fundamental subspaces. For didactic purposes, we restrict ourselves in this chapter to the situation where the generating system of multivariate polynomial equations has a zero-dimensional solution set, although many touched concepts can be extended to the positive-dimensional case.

Four fundamental subspaces

The idea of thinking about the four fundamental subspaces and their relations goes back to Gilbert Strang in the 1970s [173]. Suppose that we consider a matrix $\boldsymbol{A} \in \mathbb{C}^{p\times q}$, which is a linear transformation from $\mathbb{C}^p$ to $\mathbb{C}^q$, we can associate four fundamental subspaces [234] with that matrix, two in $\mathbb{C}^p$ and two in $\mathbb{C}^q$. These subspaces are often called the fundamental subspaces of $\boldsymbol{A}$:

1. The column space of $\boldsymbol{A}$, denoted by $\mathcal{C}(\boldsymbol{A})$, is the set of all $\boldsymbol{y}_c = \boldsymbol{A}\boldsymbol{x} \in \mathbb{C}^p$ for which $\boldsymbol{x}$ is a nonzero vector in $\mathbb{C}^q$. It is spanned by the columns of the matrix $\boldsymbol{A}$ and its dimension is called the rank of the matrix.

2. The row space of $\boldsymbol{A}$, denoted by $\mathcal{R}(\boldsymbol{A})$, is set of all $\boldsymbol{x}_r = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} \in \mathbb{C}^q$ for which $\boldsymbol{y}$ is a nonzero vector in $\mathbb{C}^p$, which corresponds to $\mathcal{C}(\boldsymbol{A}^{\mathrm{T}})$. It is spanned by the rows of the matrix $\boldsymbol{A}$.

3. The right null space of $\boldsymbol{A}$, denoted by $\mathcal{N}(\boldsymbol{A})$, is the set of all $\boldsymbol{x}_n \in \mathbb{C}^q$ for which $\boldsymbol{A}\boldsymbol{x}_n = \boldsymbol{0}$. Its dimension is called the nullity of the matrix.

4. The left null space of $\boldsymbol{A}$, denoted by $\mathcal{L}(\boldsymbol{A})$, is the set of all $\boldsymbol{y}_l \in \mathbb{C}^p$ for which $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}_l = \boldsymbol{0}$, which corresponds to $\mathcal{N}(\boldsymbol{A}^{\mathrm{T}})$.

Figure 4.1 contains a graphical depiction of the fundamental subspaces of an arbitrary matrix. They are connected through the fundamental theorem of linear algebra [236] and rank-nullity theorem [123]:

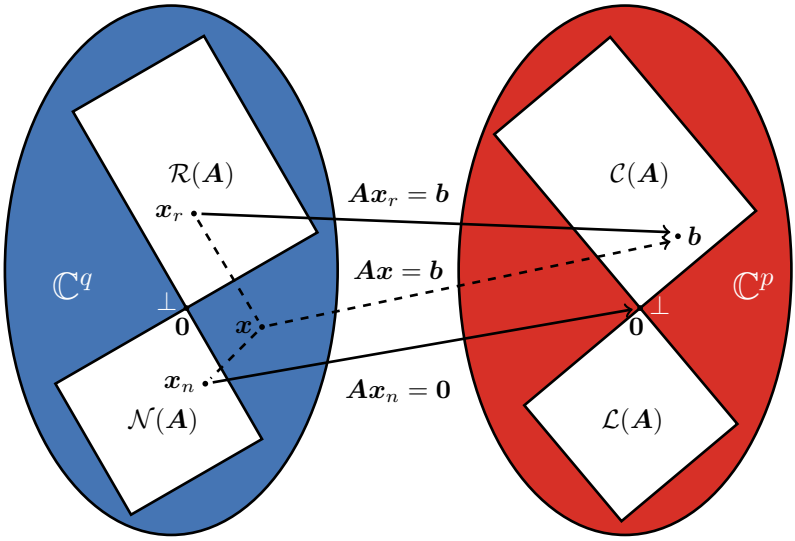


Figure 4.1. Graphical depiction of the fundamental subspaces of an arbitrary matrix $\boldsymbol{A} \in \mathbb{C}^{p \times q}$. These spaces are the row space $\mathcal{R}(\boldsymbol{A})$, the right null space $\mathcal{N}(\boldsymbol{A})$, the column space $\mathcal{C}(\boldsymbol{A})$, and the left null space $\mathcal{L}(\boldsymbol{A})$. The vectors $\boldsymbol{x}_r \in \mathbb{C}^q$ and $\boldsymbol{x}_n \in \mathbb{C}^q$ are vectors in the row space and right null space of $\boldsymbol{A}$, respectively. The vector $\boldsymbol{b} \in \mathbb{C}^p$ lies in the column space of $\boldsymbol{A}$, while $\boldsymbol{x} \in \mathbb{C}^q$ is a linear combination of $\boldsymbol{x}_r$ and $\boldsymbol{x}_n$.

Theorem 4.1 (Fundamental theorem of linear algebra). Let us consider an arbitrary complex matrix $\boldsymbol{A} \in \mathbb{C}^{p \times q}$ with rank r. The dimension of the row space is equal to the dimension of the column space, i.e.,

$$\dim(\mathcal{R}(\boldsymbol{A})) = \dim(\mathcal{C}(\boldsymbol{A})) = r. \tag{4.1}$$

Theorem 4.2 (Rank-nullity theorem). For any complex matrix $\boldsymbol{A} \in \mathbb{C}^{p \times q}$, the rank-nullity theorem states that

$$\operatorname{rank}(\boldsymbol{A}) + \operatorname{nullity}(\boldsymbol{A}) = q. \tag{4.2}$$

A natural approach to compute bases for these four fundamental is via the singular value decomposition, where the left and right singular vectors provide basis vectors for the different subspaces (more information about how to compute these basis vectors in Appendix B). Also other approaches are often used to retrieve a basis, for example, the QR decomposition with column pivoting can be used to compute a basis for the right null space [97, 249]. There exist more advanced techniques to compute a basis when the matrix has a particular structure [99, 167] or is constructed iteratively [22, 260].

Applied to the Macaulay matrix

In this chapter, we want to focus on each of the fundamental subspaces of the Macaulay matrix $\boldsymbol{M}_d \in \mathbb{C}^{p_d \times q_d}$, for various degrees d. The Macaulay matrix $\boldsymbol{M}_d$ is defined for a certain degree d and it is important to understand how its dimensions and its fundamental subspaces change as a function of that degree d. From the connections between the fundamental subspaces, it follows that

$$\mathcal{R}(\boldsymbol{M}_d) = \mathcal{C}(\boldsymbol{M}_d^{\mathrm{T}}), \tag{4.3}$$
$$\mathcal{L}(\boldsymbol{M}_d) = \mathcal{N}(\boldsymbol{M}_d^{\mathrm{T}}), \tag{4.4}$$

while Theorems 4.1 and 4.2 for $\boldsymbol{M}_d$ and $\boldsymbol{M}_d^{\mathrm{T}}$ yield

$$p_d = r_d + l_d, \tag{4.5}$$
$$q_d = r_d + n_d, \tag{4.6}$$

where r_d is the rank, l_d the dimension of the left null space, and n_d the nullity of the Macaulay matrix for degree d.

The fundamental subspaces of the Macaulay matrix (Table 4.1) possess interesting algebraic properties related to the generating system of multivariate polynomial equations:

- **Row space** (Section 4.3): the row space can be interpreted in terms of the consequences of the related homogeneous ideal.
- **Left null space** (Section 4.4): the left null space is linked to the syzygies that occur in the row space and its dimension leads to an expression for the degree of regularity d_*, which is of vital importance for the root-finding algorithms in Chapter 2.
- **Right null space** (Section 4.5): the structure of the right null space, when the degree $d \geq d_*$, can be described in terms of differential functionals (algebraic language), which is a key observation to solve the generating system of multivariate polynomial equations via this subspace.
- **Column space** (Section 4.6): the complementarity between the null space and column space allows us to also rephrase this solution approach in the column space.

4.2 Projective space

Before diving into the different fundamental subspaces, we leave the affine complex space and enter the projective space. This helps us to talk about homogeneous polynomials and projective varieties (Section 4.2.1), and it allows us to consider the homogeneous Macaulay matrix (Section 4.2.2).

Table 4.1. Overview of the four fundamental subspaces of the Macaulay matrix $\boldsymbol{M}_d \in \mathbb{C}^{p_d \times q_d}$ of degree d with rank r_d. Because of the fundamental theorem of linear algebra, the dimension of the column space and row space are equal to the rank of the matrix. The dimension of the left null space and right null space (i.e., the nullity) are l_d and n_d, respectively.

subspace	symbol	dimension	interpretation
row space	$\mathcal{R}(\boldsymbol{M}_d) \subset \mathbb{C}^{q_d}$	r_d	Section 4.3
left null space	$\mathcal{L}(\boldsymbol{M}_d) \subset \mathbb{C}^{p_d}$	l_d	Section 4.4
right null space	$\mathcal{N}(\boldsymbol{M}_d) \subset \mathbb{C}^{q_d}$	n_d	Section 4.5
column space	$\mathcal{C}(\boldsymbol{M}_d) \subset \mathbb{C}^{p_d}$	r_d	Section 4.6

4.2.1 Homogeneous polynomials and projective varieties

Systems of multivariate polynomial equations can, due to sparsity and algebraic interactions among the coefficients of the polynomials, also have solutions at infinity (Section 2.2.3). Because these points at infinity play an important didactic role in this text, we discuss here projective varieties. A necessary ingredient in this discussion is the homogeneous polynomial.

Definition 4.1. A polynomial $p(\boldsymbol{x}) \in \mathcal{P}^n$ with a support $\mathcal{A}$, such that $|\boldsymbol{\alpha}|$ is equal to the total degree of the polynomial for each $\boldsymbol{\alpha} \in \mathcal{A}$, is called a **homogeneous polynomial**.

A nonhomogeneous polynomial can easily be made homogeneous by introducing an additional variable x_0. The homogenization $p^h(\widetilde{\boldsymbol{x}})$ of a polynomial $p(\boldsymbol{x}) \in \mathcal{P}_d^n$ is the polynomial obtained by multiplying each term (i.e., each monomial) of $p(\boldsymbol{x})$ with a power of x_0 so that its total degree becomes d. The vector space of all homogeneous polynomials in $n+1$ variables and of degree d is denoted by $\widetilde{\mathcal{P}}_d^{n+1}$ and its dimension is the same as $\mathcal{P}_d^n$. To make clear when we also consider the additional variable x_0, we make a distinction between $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\widetilde{\boldsymbol{x}} = (x_0, x_1, \ldots, x_n)$.

Example 4.1. Consider two polynomials

$$p_1(\boldsymbol{x}) = x_1^2 + 2x_1x_2 + 3x_2^2 \tag{4.7}$$

and

$$p_2(\boldsymbol{x}) = x_2 + x_1^2 - x_1^3. \tag{4.8}$$

The first polynomial $p_1(\boldsymbol{x})$ is a homogeneous polynomial, while the second polynomial $p_2(\boldsymbol{x})$ is nonhomogeneous, because the monomials have different total degrees. However, the second polynomial can be made homogeneous by introducing the additional variable x_0:

$$p_2^h(\widetilde{\boldsymbol{x}}) = x_0^2x_2 + x_0x_1^2 - x_1^3. \tag{4.9}$$

Moreover, dehomogenizing $p_2^h(\widetilde{\boldsymbol{x}})$ by setting $x_0 = 1$ results again in the original polynomial $p_2(\boldsymbol{x})$.

In order to describe solution sets of systems of multivariate homogeneous polynomial equations, the projective space needs to be introduced [65].

Definition 4.2. The n-dimensional projective space $\mathbb{P}^n$ is the set of equivalence classes on $\mathbb{C}^{n+1} \setminus \{\mathbf{0}\}$. Each non-zero $(n+1)$-tuple $\widetilde{\boldsymbol{a}}$ defines a point in $\mathbb{P}^n$ with homogeneous coordinates $\widetilde{\boldsymbol{a}}$.

The above-mentioned equivalence relation $\sim$ on the non-zero points of $\mathbb{C}^{n+1}$ is defined by setting

$$(x'_0, x'_1, \ldots, x'_n) \sim (x_0, x_1, \ldots, x_n) \tag{4.10}$$

if there is a non-zero scalar $\delta \in \mathbb{C}$ such that

$$(x'_0, x'_1, \ldots, x'_n) = (\delta x_0, \delta x_1, \ldots, \delta x_n). \tag{4.11}$$

The origin $\mathbf{0} \in \mathbb{C}^{n+1}$ is not a point in the projective space $\mathbb{P}^n$, because of the equivalence relation $\sim$ this point would be associated with an infinite number of projective points. The affine space $\mathbb{C}^n$, however, can be retrieved as a "slice" of the projective space:

$$\mathbb{C}^n = \{(1, x_1, \ldots, x_n) \in \mathbb{P}^n\}. \tag{4.12}$$

This means that given a projective point $\widetilde{\boldsymbol{a}} = (a_0, a_1, \ldots, a_n)$ with $a_0 \neq 0$, its affine counterpart is $\left(1, \frac{a_1}{a_0}, \ldots, \frac{a_n}{a_0}\right)$. The projective points for which $a_0 = 0$ are called points at infinity.

Just like affine varieties are subsets of the affine space $\mathbb{C}^n$ given by polynomials in $\mathcal{P}^n$, we can define projective varieties as subsets in the projective space $\mathbb{P}^n$ given by homogeneous polynomials in $\widetilde{\mathcal{P}}^{n+1}$.

Definition 4.3. If $p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})$ are homogeneous polynomials in $\widetilde{\mathcal{P}}^{n+1}$, then it is possible to define the set

$$\mathcal{V}\big(p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})\big) = \big\{\widetilde{\boldsymbol{a}} \in \mathbb{P}^n : p_i^h(\widetilde{\boldsymbol{a}}) = 0 \text{ for all } 1 \leq i \leq s\big\}. \tag{4.13}$$

The set $\mathcal{V}\big(p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})\big)$ is called the **projective variety** defined by the polynomials $p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})$.

We would like to also generalize the definition of an ideal to the projective setting. However, being homogeneous is not preserved under the sum operation in $\widetilde{\mathcal{P}}^{n+1}$ [65]. If we sum homogeneous polynomials of different total degrees, the sum will never be homogeneous. The ideal $\mathcal{I}$ generated by these homogeneous polynomials $p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})$ contains nonhomogeneous polynomials. Nevertheless, each element of this $\mathcal{I}$ vanishes on all homogeneous coordinates of the variety $\mathcal{V}\big(p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})\big)$ defined by the original homogeneous polynomials. This builds up the following definition for a homogeneous ideal:

Definition 4.4. An ideal $\mathcal{I} \subset \widetilde{\mathcal{P}}^{n+1}$ is said to be a **homogeneous ideal** if for each $p(\widetilde{\boldsymbol{x}}) \in \mathcal{I}$, the homogeneous components of $p(\widetilde{\boldsymbol{x}})$ are in $\mathcal{I}$ as well.

Unfortunately, most ideals do not have this property and are not homogeneous, but the ideal $\mathcal{I} = \langle p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}}) \rangle$ generated by the homogeneous polynomials $p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})$ does [65]!

4.2.2 Homogeneous Macaulay matrix

The construction of the Macaulay matrix in Definition 2.11 can also be interpreted as the Macaulay matrix of the corresponding homogenized polynomials.

Definition 4.5. Consider the system of homogeneous multivariate polynomial equations with polynomials $p_i^h(\widetilde{\boldsymbol{x}}) \in \widetilde{\mathcal{P}}^{n+1}$, for $i = 1, \ldots, s$, which serve as the seed equations. Let the total degree of every polynomial $p_i^h(\widetilde{\boldsymbol{x}})$ be denoted by d_i. The **homogeneous Macaulay matrix** of degree d, $\boldsymbol{M}_d \in \mathbb{C}^{p_d \times q_d}$, contains the coefficients of the seed equations and the equations generated by the FSR with all monomials of total degree $d_{\mathrm{r}_i} = d - d_i$ in the variables $x_0, x_1, \ldots, x_n$, i.e.,

$$\boldsymbol{M}_d^h = \begin{bmatrix}\!\!\begin{bmatrix} \left\{\widetilde{\boldsymbol{x}}^{\widetilde{\boldsymbol{\alpha}}_1}\right\} p_1^h(\boldsymbol{x}) \\ \vdots \\ \left\{\widetilde{\boldsymbol{x}}^{\widetilde{\boldsymbol{\alpha}}_s}\right\} p_s^h(\boldsymbol{x}) \end{bmatrix}\!\!\end{bmatrix}, \tag{4.14}$$

where $[\![\cdot]\!]$ is the (stacked) vector representation of the polynomials (Definition 2.5).

This adaptation is possible because changing to homogeneous coordinates is nothing more than a relabelling of the rows and columns of the Macaulay matrix, its coefficients/elements remain the same.

Proposition 4.1. The Macaulay matrix $\boldsymbol{M}$ of degree d for the system of multivariate polynomials $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ is identical to the homogeneous Macaulay matrix of degree d for the system of homogeneous polynomials $p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})$, obtained after homogenization, i.e.,

$$\boldsymbol{M}_d^h = \boldsymbol{M}_d. \tag{4.15}$$

We continue in this chapter with the homogeneous interpretation of the Macaulay matrix and drop the superscript.

Example 4.2. Let us homogenize the polynomials in (2.35), yielding

$$\begin{cases} p_1^h(\widetilde{\boldsymbol{x}}) = x_1^2 + x_2^2 - 6x_1x_0 + 7x_0^2 = 0, \\ p_2^h(\widetilde{\boldsymbol{x}}) = x_1 - x_2 - 3x_0 = 0. \end{cases} \tag{4.16}$$

The construction of the Macaulay matrix give us the same matrix as in (2.40), but with a different labelling of the rows and columns.

$$
\begin{array}{c}
 \\
x_0 \\
x_1 \\
x_2 \\
x_0^2 \\
x_0x_1 \\
x_0x_2 \\
x_1^2 \\
x_1x_2 \\
x_2^2
\end{array}
\left[\begin{array}{ccc|ccc|ccc|c}
x_0^3 & x_0^2x_1 & x_0^2x_2 & x_0x_1^2 & x_0x_1x_2 & x_0x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \\
\hline
7 & -6 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 7 & 0 & -6 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 7 & 0 & -6 & 0 & 0 & 1 & 0 & 1 \\
-3 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -3 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -3 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -3 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -3 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 1 & -1
\end{array}\right].
$$

4.3 Row space interpretation

Consider a system of polynomials $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \in \mathcal{P}^n$ and the Macaulay matrix $\boldsymbol{M}_d$ of degree d generated by these polynomials. Every row of $\boldsymbol{M}_d$ corresponds to a polynomial $\boldsymbol{x}^{\boldsymbol{\alpha}} p_i(\boldsymbol{x})$ of total degree $\leq d$. The row space $\mathcal{R}(\boldsymbol{M})$ contains all linear combinations of rows with maximum total degree equal to d, hence all the multivariate polynomials

$$
\left\{ \sum_{i=0}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x}) : h_i(\boldsymbol{x}) \in \mathcal{P}^n_{d-d_i} \right\}. \tag{4.17}
$$

If we recall (2.26), then a polynomial ideal is defined as the set

$$
\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle = \left\{ \sum_{i=0}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x}) : h_i(\boldsymbol{x}) \in \mathcal{P}^n \right\}. \tag{4.18}
$$

So, it is now very tempting to say that

$$
\mathcal{R}(\boldsymbol{M}) = \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle \cap \mathcal{P}^n_d \triangleq \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle_d, \tag{4.19}
$$

or in words: the row space of the Macaulay matrix contains all the polynomials of the ideal $\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle$ from degree 0 up to d. However, this is not necessarily true. The row space does not, in general, contain all polynomials of degree d that can be written as a consequence of the original polynomials [24].

Example 4.3 (taken from [24]). Consider the following system of multivariate polynomial equations:

$$
\begin{cases} p_1(x) = x^2 + 2x + 1 = 0, \\ p_2(x) = x^2 + x + 1 = 0. \end{cases} \tag{4.20}
$$

It follows that $1 \in \langle p_1(x), p_2(x) \rangle$, because

$$(-1-x)p_1(x) + (2+x)p_2(x) = 1. \tag{4.21}$$

However, $1 \notin \mathcal{R}(\boldsymbol{M}_2)$, since $\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ is not a vector in the row space of $\boldsymbol{M}_2$, i.e.,

$$\boldsymbol{M}_2 = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{4.22}$$

In fact, (4.21) tells us that $1 \in \mathcal{R}(\boldsymbol{M}_3)$:

$$\boldsymbol{M}_3 = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}. \tag{4.23}$$

Deciding whether a given multivariate polynomial lies in the ideal generated by some polynomials is called the ideal membership problem. A numerical algorithm to check this via the Macaulay matrix is presented in [26].

As the previous counter example shows, the reason that not all polynomials of degree d lie in the row space of the Macaulay matrix $\boldsymbol{M}_d$ of degree d is because it is possible that a polynomial combination of degree higher than d is required. However, when we consider only homogeneous polynomials of degree d, we can interpret the row space properly. Given a set of (possibly nonhomogenous) polynomials $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$, we can interpret the row space as the vector space

$$\mathcal{R}(\boldsymbol{M}_d) = \left\{ \sum_{i=1}^{s} h_i(\widetilde{\boldsymbol{x}}) p_i^h(\widetilde{\boldsymbol{x}}) : h_i(\widetilde{\boldsymbol{x}}) \in \widetilde{\mathcal{P}}_{d-d_i}^{n+1} \right\}, \tag{4.24}$$

where $p_i^h(\widetilde{\boldsymbol{x}})$ denotes the homogenization of $p_i(\boldsymbol{x})$ and the polynomials $h_i(\widetilde{\boldsymbol{x}})$ are also homogeneous. The homogeneity of the homogeneous ideal guarantees that all homogeneous polynomials of degree d are contained in the $\mathcal{R}(\boldsymbol{M}_d)$, which corresponds to

$$\mathcal{R}(\boldsymbol{M}_d) = \left\langle p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}}) \right\rangle_d. \tag{4.25}$$

Then, an important consequence is that

$$\dim\left(\left\langle p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}}) \right\rangle_d \right) = r_d. \tag{4.26}$$

When considering a matrix, the (row) rank of the matrix is often smaller than the number of rows. What makes that $r_d \leq p_d$ for the Macaulay matrix? This means that there are linearly dependent rows in the matrix. The occurrence of linearly dependent rows is linked to polynomial combinations of polynomials equal to zero, i.e.,

$$\sum_{i=1}^{s} h_i(\widetilde{\boldsymbol{x}}) p_i^h(\widetilde{\boldsymbol{x}}) = 0, \tag{4.27}$$

or, after dehomogenizing the polynomials,

$$\sum_{i=1}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x}) = 0. \tag{4.28}$$

A polynomial combination as in (4.28) is called a *syzygy*[1]. It is possible to identify with each syzygy a linearly dependent row of the Macaulay matrix. The linear dependence of this particular row is with respect to the other rows of the Macaulay matrix. An important observation is that

$$\boldsymbol{x}^{\boldsymbol{\beta}} \sum_{i=1}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x}) = 0, \tag{4.29}$$

which means that all monomial multiples of a syzygy will also be a syzygy and that the corresponding rows in the Macaulay matrix will also be linearly dependent. We call (4.28) a *standard syzygy*, and the monomial multiples (4.29) are called *derived syzygies*. It can be shown that the number of standard syzygies is finite, this is linked with Hilbert's basis theorem [24, 66]. From linear algebra perspective this means that a standard syzygy, which corresponds to a linear combination of rows, generates more linear combinations of rows for higher degrees, which results in linearly dependent rows associated with the derived syzygies.

Example 4.4. Let us consider the system of multivariate polynomials

$$\begin{cases} p_1(\boldsymbol{x}) = -3x_2 + 6x_1x_2 - 1x_2^2 + x_1^2x_2 - 3x_1x_2^2 + x_2^3 = 0, \\ p_2(\boldsymbol{x}) = 1 + 2x_1 - 2x_2 + x_1^2 + -2x_1x_2 + x_2^2 = 0, \\ p_3(\boldsymbol{x}) = -4x_2 + x_2^2 = 0. \end{cases} \tag{4.30}$$

The corresponding Macaulay matrix of degree $d = 3$ is equal to

$$\begin{array}{c} \\ 1(\boldsymbol{x}) \\ x_0(\boldsymbol{x}) \\ x_1(\boldsymbol{x}) \\ x_2(\boldsymbol{x}) \\ x_0(\boldsymbol{x}) \\ x_1(\boldsymbol{x}) \\ x_2(\boldsymbol{x}) \end{array} \begin{array}{c} \begin{array}{cccccccccc} x_0^3 & x_0^2x_1 & x_0^2x_2 & x_0x_1^2 & x_0x_1x_2 & x_0x_2^2 & x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 \end{array} \\ \left[\begin{array}{c|c|c|c|c|c|c|c|c|c} 0 & 0 & -3 & 0 & 6 & -1 & 0 & 1 & -3 & 1 \\ 1 & 2 & -2 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & -2 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 2 & -2 & 0 & 1 & -2 & 1 \\ 0 & 0 & -4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & 1 \end{array}\right] \end{array}, \tag{4.31}$$

where we notice that the 6th row is a linear combination of the 1st, 4th, and 5th row. This corresponds to the polynomial combination

$$(x_0 - x_1)p_3(\boldsymbol{x}) + x_2 p_2(\boldsymbol{x}) - p_1(\boldsymbol{x}) = 0, \tag{4.32}$$

[1]The Greek word $\sigma\upsilon\zeta\upsilon\gamma\iota\alpha$ (syzygy) refers to an alignment of celestial bodies. Cayley introduced the word in the field of mathematics, to denote linear relations between the minors of a matrix. The words was later used by Hilbert in 1890 for polynomials: the polynomials are thought to be in syzygy with each other, so that their polynomial combination is equal to zero.

which leads evidently to the standard syzygy

$$(1 - x_1)p_3(\boldsymbol{x}) + x_2p_2(\boldsymbol{x}) - p_1(\boldsymbol{x}) = 0, \tag{4.33}$$

Unsurprisingly, derived syzygies, like

$$x_1^2((1 - x_1)p_3(\boldsymbol{x}) + x_2p_2(\boldsymbol{x}) - p_1(\boldsymbol{x})) = 0, \tag{4.34}$$

also generate linearly dependent rows in (homogeneous) Macaulay matrices of higher degrees.

4.4 Left null space interpretation

We can learn more about these syzygies by looking at the left null space of the (homogeneous) Macaulay matrix. The left null space gives us more information about the coefficients in these polynomial combinations. Remember that the left null space of the Macaulay matrix $\boldsymbol{M}_d$ is the vector space

$$\mathcal{L}(\boldsymbol{M}_d) = \left\{\boldsymbol{h} \in \mathbb{C}^{p_d \times 1} : \boldsymbol{h}^{\mathrm{T}}\boldsymbol{M}_d = \boldsymbol{0}\right\}. \tag{4.35}$$

The multiplications $\boldsymbol{h}^{\mathrm{T}}\boldsymbol{M}_d$ are equivalent with

$$\sum_{i=1}^{s} h_i(\widetilde{\boldsymbol{x}})p_i^h(\widetilde{\boldsymbol{x}}) = 0, \tag{4.36}$$

where the vector $\boldsymbol{h} \in \mathcal{L}(\boldsymbol{M}_d)$ contains the coefficients of all the polynomials $h_i(\widetilde{\boldsymbol{x}})$ in the linear combination (4.36).

Example 4.5. The polynomial combination (4.32) is equivalent with

$$-p_1(\boldsymbol{x}) + x_2p_2(\boldsymbol{x}) + x_0p_3(\boldsymbol{x}) - x_1p_3(\boldsymbol{x}) = \boldsymbol{0}, \tag{4.37}$$

which corresponds with a vector in the left null space

$$\boldsymbol{h} = \begin{bmatrix}-1 & 0 & 0 & 1 & 1 & -1 & 0\end{bmatrix}^{\mathrm{T}}. \tag{4.38}$$

This brings us to the interpretation of the dimension l_d of the left null space: it counts the total number of syzygies that occur in the row space. After identifying the syzygies in the row space of the (homogeneous) Macaulay matrix, by determining the linearly dependent rows, it is possible to write down an expression for the dimension l_d of the left null space [24]. Indeed, suppose that we have identified a linearly dependent row in the Macaulay matrix with respect to all the rows above it, this linear dependence expresses a certain syzygy (4.28) and introduces a term to l_d.

Lemma 4.1 (Lemma 3.1 from [24]). If a standard syzygy has a total degree d_l, then it introduces a term

$$\binom{d - d_l + n}{n} \tag{4.39}$$

to the dimension l_d of the left null space of the (homogeneous) Macaulay matrix.

Proof. This follows from $x_j^{\beta_j} \sum_{i=1}^{s} h_i(\boldsymbol{x}) p_i(\boldsymbol{x}) = 0$ and the fact that the total number of monomials $x_j^{\beta_j}$ at a degree $d \geq d_l$ is given by (4.39). □

Unfortunately, standard syzygies of the same set (i.e., that consider the same polynomials) count some linearly dependent rows more than once, as the next example illustrates.

Example 4.6 (taken from [24]). Consider the following system of multivariate polynomial equations:

$$\begin{cases} p_1(\boldsymbol{x}) = x_1^2 x_2^2 + x_3 = 0, \\ p_2(\boldsymbol{x}) = x_1 x_2 - 1 = 0, \\ p_3(\boldsymbol{x}) = x_1^2 + x_3 = 0. \end{cases} \tag{4.40}$$

The first standard syzygy can be found in the rows of the corresponding (homogeneous) Macaulay matrix of degree $d = 4$ and corresponds to the row

$$x_1 x_2 p_3^h(\widetilde{\boldsymbol{x}}) \to x_1 x_2 p_3(\boldsymbol{x}). \tag{4.41}$$

The remaining standard syzygies are all found for degree $d = 6$ and correspond to the rows

$$x_1^3 x_2 p_2^h(\widetilde{\boldsymbol{x}}) \to x_1^3 x_2 p_2(\boldsymbol{x}), \tag{4.42}$$

$$x_1^2 x_2^2 p_2^h(\widetilde{\boldsymbol{x}}) \to x_1^2 x_2^2 p_2(\boldsymbol{x}), \tag{4.43}$$

$$x_1 x_2^2 x_3 p_2^h(\widetilde{\boldsymbol{x}}) \to x_1 x_2^2 x_3 p_2(\boldsymbol{x}). \tag{4.44}$$

The first group of standard syzygies, $\{x_1 x_2 p_3(\boldsymbol{x})\}$, has only one element and describes a syzygy of degree 4. It introduces a term

$$\binom{d - 4 + 3}{3} \tag{4.45}$$

to the dimension l_d of the left null space. Moreover, the second group of three standard syzygies, $\{x_1^3 x_2 p_2(\boldsymbol{x}), x_1^2 x_2^2 p_2(\boldsymbol{x}), x_1 x_2^2 x_3 p_2(\boldsymbol{x})\}$, introduces three times the term

$$\binom{d - 6 + 3}{3}. \tag{4.46}$$

to l_d. It is now tempting to sum both contributions in l_d, but taking three times the binomial coefficient in (4.46) counts too many contributions. Take for example the standard syzygies corresponding with the rows $x_1^3 x_2 p_2^h(\widetilde{\boldsymbol{x}})$ and $x_1^2 x_2^2 p_2^h(\widetilde{\boldsymbol{x}})$. Their least common multiple is $x_1^3 x_2^2 p_2^h(\widetilde{\boldsymbol{x}})$, which means that this row is counted twice if we take (4.46) three times.

One notices that each of these groups can be analyzed separately, because they involve different polynomials. The derived syzygies can not involve more polynomials $p_i^h(\widetilde{\boldsymbol{x}})$ than in the standard syzygy, so there is no interference between rows of different groups possible. Via the inclusion-exclusion principle, Batselier et al. [24] have derived an algorithm that determines the expression for l_d correctly.

Theorem 4.3 (Inclusion-exclusion principle). Let $A_1, \ldots, A_n$ be a collection of finite sets, with $|A_i|$ the cardinality of A_i. Then

$$\left| \bigcup_{i=1}^{n} A_i \right| = \sum_{k=1}^{n} (-1)^{k+1} \left(\sum_{i_1 \leq \ldots \leq i_k} |A_{i_1} \cap \cdots \cap A_{i_k}| \right). \tag{4.47}$$

Proof. A proof of this theorem can be found in [65, p. 454]. □

Applying this inclusion-exclusion principle on the standard syzygies removes the derived syzygies that are counted more than once and results in a correct expression for l_d [24].

Example 4.6 (continuing from p. 156). For (4.40), the sets A_1, A_2 and A_3 are the monomial multiples of $x_1^3 x_2 p_2(\boldsymbol{x})$, $x_1^2 x_2^2 p_2(\boldsymbol{x})$, and $x_1 x_2^2 x_3 p_2(\boldsymbol{x})$. Their cardinality is given by (4.46), the cardinality of their intersections correspond to

$$A_1 \cap A_2 = \binom{d-7+3}{3}, \tag{4.48}$$

$$A_1 \cap A_3 = \binom{d-8+3}{3}, \tag{4.49}$$

$$A_2 \cap A_3 = \binom{d-7+3}{3}, \tag{4.50}$$

$$A_1 \cap A_2 \cap A_3 = \binom{d-8+3}{3}, \tag{4.51}$$

which uses the same formula as (4.39), but with the degree equal to the degree of the least common multiple of the involved syzygies. A correct expression for l_d can now be found as

$$\begin{aligned} l_d = &\binom{d-4+3}{3} + 3\binom{d-6+3}{3} - 2\binom{d-7+3}{3} \\ &- \binom{d-8+3}{3} + \binom{d-8+3}{3} \end{aligned} \tag{4.52}$$

$$
\begin{aligned}
&= \binom{d-4+3}{3} + 3\binom{d-6+3}{3} - 2\binom{d-7+3}{3} \\
&= \frac{1}{3}d^3 - 2d^2 + \frac{2}{3}d + 9 \quad (d \geq 4).
\end{aligned}
$$

Once l_d is known for all degrees d, the rank r_d and nullity n_d of the Macaulay matrix are also fully determined, via (4.5) and (4.6). An important consequence of the dimension of the left null space is the degree of regularity d_*.

Definition 4.6. The minimal degree d_* for which the dimension l_d can be computed correctly via the number of obtained syzygies is called the **degree of regularity**.

The degree of regularity (sometimes called the Hilbert regularity) is of vital importance when using the right null space or column space to solve systems of multivariate polynomial equations. From the rank-nullity theorem of $\boldsymbol{M}_d$ (Theorem 4.2), it follows that

$$
\begin{aligned}
n_d &= q_d - r_d \\
&= \dim \widetilde{\mathcal{P}}_d^{n+1} - \dim\left\langle p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})\right\rangle_d \\
&= \dim \widetilde{\mathcal{P}}_d^{n+1} / \left\langle p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})\right\rangle_d.
\end{aligned} \tag{4.53}
$$

When d_* is reached, the nullity n_d of the Macaulay matrix corresponds to the projective Hilbert polynomial, meaning that it describes the dimension of the projective variety [24].

Definition 4.7. The polynomial

$$
n_d = \mathrm{HP}(d) = \dim \widetilde{\mathcal{P}}_d^{n+1} / \left\langle p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})\right\rangle_d, \quad \forall d \geq d_* \tag{4.54}
$$

is called the **projective Hilbert polynomial**. The degree of $\mathrm{HP}(d)$ equals the dimension of the projective variety [65, p. 493].

This leads to the following theorem in the zero-dimensional situation:

Theorem 4.4. Consider a homogeneous ideal $\left\langle p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}})\right\rangle_d$ with exactly m_b isolated projective common roots (counting multiplicities) and degree of regularity d_*. We have that

$$
n_d = m_b, \quad \forall d \geq d_*. \tag{4.55}
$$

Proof. This follows from (4.53) and Definition 4.6. □

Notice that, for $s = n$ and a zero-dimensional ideal, then $n_d = m_b$ is given by Bézout's theorem when $d \geq d_*$.

The number of standard syzygies is finite [24]. This finiteness is intimately linked with the occurrence of a Gröbner basis in the row space of the Macaulay matrix [26]: the reduction to zero of every S-polynomial[2] of a pair of polynomials in the Gröbner basis provides one standard syzygy. This implies that it is required to construct a Macaulay matrix $\boldsymbol{M}_d$ for a degree which contains all these S-polynomials. So, a stop criterion is needed to be able to decide whether we have found all these S-polynomials or syzygies. Lazard [154] showed that, in the zero-dimensional case, the maximal degree of a reduced Gröbner basis is at most

$$\left(\sum_{i=1}^{n} d_i\right) + d_0 - n + 1, \tag{4.56}$$

with $d_0 = 1$ when $n = s$. This provides an upper bound on the degree at which all standard syzygies can be found. However, in order to find d_* via the standard syzygies, it is required to construct a Macaulay matrix for a degree larger than d_*. Macaulay [159] showed that it is possible to determine whether a homogeneous polynomial system has a nontrivial common root by computing the determinant of a submatrix of $\boldsymbol{M}_d$ for $d = (\sum_{i=1}^{n} d_i) - n + 1$. This results in the maximal degree to check for d_* [24], which is given by

$$\left(\sum_{i=1}^{n} d_i\right) + 1. \tag{4.57}$$

In practice, it may not be useful to use the left null space to determine d_*, but one can directly monitor the nullity of the right null space instead. Note that Batselier et al. [24] have proposed two numerical algorithms to determine this d_*, taking into account possible cancellations of the large combinatorial terms involved in determining l_d.

4.5 Right null space interpretation

In this section, we show how the projective common roots of a system of multivariate polynomials are connected to the right null space of the (homogeneous) Macaulay matrix generated by these polynomials. The goal of this section is to show that the matrices obtained via the shift-invariance approach of Chapter 2 are related to the multiplication matrices used in traditional normal form methods.

It is a classic result that for a polynomial system $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ with a finite number of affine common roots (i.e., an affine zero-dimensional variety), the quotient ring $\mathcal{R}[\mathcal{I}] = \mathcal{P}^n / \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x}) \rangle$ is a finite-dimensional vector space [21, 66, 230]. Through the multiplication structure of $\mathcal{R}[\mathcal{I}]$, it is possible

[2] An essential ingredient in Buchberger's algorithm to compute a Gröbner basis is the S-polynomial. It name comes from "substraction" or "syzygy" polynomial and it is used to eliminate leading terms from a system of polynomials. Its mathematical definition is given in Definition A.10.

to retrieve the common roots of the multivariate polynomials. This multiplication structure can also be described via the right null space of the Macaulay matrix. Therefore, the solution algorithm in Section 2.4 can be interpreted, in a certain sense, as an extension/adaptation of the traditional eigenvalue-eigenvector approach described by [229] to solve systems of multivariate polynomial equations.

First, we show how the multiplicative structure of the quotient space leads to a link between the affine common roots of a zero-dimensional system of multivariate polynomials and the eigenvalues of a matrix (Section 4.5.1). Next, we show how to set up similar "projective multiplication matrices" via the right null space of the Macaulay matrix and interpret the associated row selection procedure (Section 4.5.2).

4.5.1 Affine solutions and multiplication maps

Let us first consider the case where the ideal $\mathcal{I} = \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle \subset \mathcal{P}^n$ describes an affine zero-dimensional variety $\mathcal{V}(\mathcal{I}) \subset \mathbb{C}^n$. The set

$$[p]_{\mathcal{I}} = \{q \in \mathcal{P}^n : p(\boldsymbol{x}) - q(\boldsymbol{x}) \in \mathcal{I}\} \tag{4.58}$$

is the set of all remainders of $p(\boldsymbol{x}) \in \mathcal{P}^n$ modulo the ideal $\mathcal{I}$. It is called the residue class of $p(\boldsymbol{x})$ modulo $\mathcal{I}$. Every polynomial $p(\boldsymbol{x})$ defines such a residue class and we call $p(\boldsymbol{x})$ a representative of $[p]_{\mathcal{I}}$. The set $\mathcal{R}[\mathcal{I}]$ is the set of all residue classes modulo $\mathcal{I}$ and is a vector space over $\mathbb{C}$, since the scalar multiplication and addition operations are well-defined [230]. Moreover, since the multiplication is commutative, $\mathcal{R}[\mathcal{I}]$ is a commutative ring and is called often the **quotient ring**, denoted by $\mathcal{P}^n/\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle$.

Next, observe that, given a polynomial $g(\boldsymbol{x})$, we can use multiplication to define a linear map $\mathcal{A}_g$ from $\mathcal{R}[\mathcal{I}]$ to itself:

Definition 4.8. For any $g(\boldsymbol{x}) \in \mathcal{P}^n$, we define the **multiplication map** representing the multiplication with $g(\boldsymbol{x})$ as the linear map

$$\mathcal{A}_g : \mathcal{R}[\mathcal{I}] \to \mathcal{R}[\mathcal{I}] : [f]_{\mathcal{I}} \mapsto [f]_{\mathcal{I}}[g]_{\mathcal{I}} = [fg]_{\mathcal{I}}. \tag{4.59}$$

Since $\mathcal{R}[\mathcal{I}]$ is a finite-dimensional vector space over $\mathbb{C}$, we can represent the multiplication map by its matrix with respect to a basis $B = \{[b_1]_{\mathcal{I}}, \ldots, [b_{m_b}]_{\mathcal{I}}\}$. This $m_b \times m_b$ matrix, where the number of projective solutions m_b is equal to the dimension of $\mathcal{R}[\mathcal{I}]$, is called the multiplication matrix and denoted by $\boldsymbol{A}_g$. Its eigenvalues and eigenvectors have an important property:

Theorem 4.5 (Eigenvalue-eigenvector theorem). Let $\mathcal{I} \subset \mathcal{P}^n$ be an ideal with zero-dimensional variety $\mathcal{V}(\mathcal{I}) = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{m_b})$ and $\boldsymbol{A}_g$ the multiplication matrix for a multiplication map $\mathcal{A}_g$ with respect to a given basis $B = \{[b_1]_{\mathcal{I}}, \ldots, [b_{m_b}]_{\mathcal{I}}\}$. The eigenvalues of $\boldsymbol{A}_g$ are the evaluations of $g(\boldsymbol{x})$ in the m_b points of the zero-dimensional variety and the row vector

$$\begin{bmatrix} b_1(\boldsymbol{a}_j) & \cdots & b_{m_b}(\boldsymbol{a}_j) \end{bmatrix} \tag{4.60}$$

lies in the left eigenspace for the eigenvalue $g(\boldsymbol{a}_j)$, $j = 1, \ldots, m_b$.

Proof. A proof of this theorem can be found in [66, Chapter 2]. □

This means that, if the coordinate functions $x_1, \ldots, x_n$ are among the basis elements, then the coordinates of the solutions can be read of directly from the left eigenvectors. Theorem 4.5 implies that the solutions of a system of multivariate polynomial equations can be retrieved from the multiplication structure of the quotient space by computing the multiplication matrices $\boldsymbol{A}_{x_1}, \ldots, \boldsymbol{A}_{x_n}$ in the variables $x_1, \ldots, x_n$ and computing their eigenvalues. Note that the matrix $\boldsymbol{A}_{x_i}$ and $\boldsymbol{A}_{x_j}$, for any i and j, commute since $x_i x_j = x_j x_i$. As a result, the matrices $\boldsymbol{A}_{x_i}$ and $\boldsymbol{A}_{x_j}$ have common eigenspaces. Thus, the traditional procedure to find all affine common roots can be summarized as:

1. Compute the multiplication matrices $\boldsymbol{A}_{x_1}, \ldots, \boldsymbol{A}_{x_n}$ for a particular basis B of the quotient space $\mathcal{R}[\mathcal{I}]$.

2. Perform a simultaneous triangularization of $\boldsymbol{A}_{x_1}, \ldots, \boldsymbol{A}_{x_n}$ to retrieve the common roots of the system.

Where the second step can be tackled via standard algorithms from numerical linear algebra (for example via Schur decompositions), the first step requires the construction of these multiplication matrices. This involves computing so-called **normal forms**, which map the obtained polynomials after multiplication onto the basis along the ideal; hence, the name of these methods. Computing normal forms can be done using symbolic techniques or via linear algebra methods. We now present an approach via the right null space of the Macaulay matrix, while Appendix A.5.2 considers the more "traditional" approach.

4.5.2 Projective solutions and the dual vector space

The attentive readers may see a lot of similarities between the above-mentioned approach and the null space based root-finding algorithm that we have developed in Section 2.4. Indeed, as soon as $d \geq d_*$ and the number of projective common roots is finite, it is possible to obtain the homogeneous multiplication matrices via the right null space of the Macaulay matrix [24]. Let us consider now a homogeneous ideal $\mathcal{I}_d = \langle p_1^h(\widetilde{\boldsymbol{x}}), \ldots, p_s^h(\widetilde{\boldsymbol{x}}) \rangle_d$ that describes a projective zero-dimensional variety[3] $\mathcal{V}(\mathcal{I})$. The quotient space is given by $\mathcal{R}[\mathcal{I}]_d = \widetilde{\mathcal{P}}_d^{n+1} / \mathcal{I}_d$.

Theorem 4.5 can also be phrased in the projective space for homogeneous multiplication maps [242]:

[3]Note that we consider the variety together with information about the multiplicity 'tructure of each point in that variety as a solution for the generating polynomials. This is related to Footnote 10 of Chapter 2 about schemes.

Definition 4.9. For any $g(\boldsymbol{x}) \in \mathcal{P}^n_{d'}$, we define the **homogeneous multiplication map** representing the multiplication with $g(\boldsymbol{x})$ as the linear map

$$\mathcal{A}_g : \mathcal{R}[\mathcal{I}]_d \to \mathcal{R}[\mathcal{I}]_{d+d'} : [f]_{\mathcal{I}} \mapsto [f]_{\mathcal{I}_d}[g]_{\mathcal{I}_{d'}} = [fg]_{\mathcal{I}_{d+d'}}. \tag{4.61}$$

Since $\mathcal{R}[\mathcal{I}]_d$ is a finite-dimensional vector space over $\mathbb{C}$, we can represent the multiplication map after fixing a basis $B = \left\{[b_1]_{\mathcal{I}_d}, \ldots, [b_{m_b}]_{\mathcal{I}_d}\right\}$ by its matrix. This $m_b \times m_b$ matrix is called the multiplication matrix and is denoted by $\boldsymbol{A}_g$. Its eigenvalues and eigenvectors again have an important property:

Theorem 4.6 (Projective eigenvalue-eigenvector theorem). Let $\mathcal{I}_d \subset \widetilde{\mathcal{P}}^n_d$, with $d \geq d_*$ be a homogeneous ideal with zero-dimensional projective variety $\mathcal{V}(\mathcal{I}) = (\widetilde{\boldsymbol{a}}_1, \ldots, \widetilde{\boldsymbol{a}}_{m_b}) \subset \mathbb{P}^n$ and $\boldsymbol{A}_{g/h}$ the homogeneous multiplication matrix for a homogeneous multiplication map $\mathcal{A}_{g/h} = \mathcal{A}_h^{-1} \circ \mathcal{A}_g$ with respect to a given basis $B = \left\{[b_1]_{\mathcal{I}_d}, \ldots, [b_{m_b}]_{\mathcal{I}_d}\right\}$, where for the homogeneous polynomial $h(\widetilde{\boldsymbol{x}})$, it holds that $h(\widetilde{\boldsymbol{a}}_j) \neq 0$, for $j = 1, \ldots, m_b$. The eigenvalues of $\boldsymbol{A}_{g/h}$ are the evaluations of $g(\widetilde{\boldsymbol{x}})/h(\widetilde{\boldsymbol{x}})$ in the m_b points of the zero-dimensional projective variety and the row vector

$$\begin{bmatrix} b_1(\widetilde{\boldsymbol{a}}_j) & \cdots & b_{m_b}(\widetilde{\boldsymbol{a}}_j) \end{bmatrix} \tag{4.62}$$

lies in the left eigenspace for the eigenvalue $g(\widetilde{\boldsymbol{a}})/h(\widetilde{\boldsymbol{a}})$, $j = 1, \ldots, m_b$.

Proof. A proof of this theorem can be found in [242]. □

4.5.2.1 Shift-invariance structure of the right null space

As soon as $d \geq d_*$ and the number of projective common roots is finite, a basis matrix of the right null space can be written explicitly in terms of these projective common roots [24]. This requires the notion of the dual vector space. The dual space allows to describe the relation between the multiplication structure and the composition of the multivariate Vandermonde basis matrix of the right null space, by means of linear combinations of partial derivatives of the monomial vectors. We denote the dual vector space of the set of projective multivariate monomials $\widetilde{\mathcal{C}}^{n+1}_d$ by $\widetilde{\mathcal{C}}^{n+1'}_d$

Definition 4.10. If $j \in \mathbb{N}^n_0$ and $\widetilde{\boldsymbol{a}} \in \mathbb{P}^n$, then the **differential functional** $\partial_j(\cdot)|_{\boldsymbol{a}} \in \widetilde{\mathcal{C}}^{n+1'}_d$ is defined by

$$\partial_j(\cdot)|_{\boldsymbol{a}} = \frac{1}{j_1! \cdots j_n!} \frac{\partial^{j_1+\cdots+j_n}(\cdot)}{\partial x_1^{j_1} \ldots \partial x_n^{j_n}}\bigg|_{\widetilde{\boldsymbol{a}}} \tag{4.63}$$

where $(\cdot)|_{\widetilde{\boldsymbol{a}}}$ stands for the evaluation of $\widetilde{\boldsymbol{x}}$ in an affine point $\widetilde{\boldsymbol{a}}$.

A basis of the dual vector space of the row space $\mathcal{R}'(\boldsymbol{M}_d)$ can be given by differential functionals evaluated in the projective common roots. Being elements of $\mathcal{R}'(\boldsymbol{M}_d)$, these differential functionals can be represented as column vectors. Applying the differential functionals to the elements of the row space is simply the matrix vector multiplication of the Macaulay matrix with these functionals. A basis for the annihilator of the row space, which corresponds to the right null space, consist of the differential functionals evaluated into each projective common roots, taking into account multiplicities. We call this basis the canonical right null space of $\boldsymbol{M}_d$, which can be seen as a generalization of the multivariate Vandermonde basis that we have used in Section 2.4.1.1.

Definition 4.11. Consider the set of homogeneous polynomials $p_1^h(\widetilde{\boldsymbol{x}}), \dots, p_s^h(\widetilde{\boldsymbol{x}})$ with a zero-dimensional projective solution set. Let $m_1, \dots m_t$ be the multiplicities of the t different projective common roots $\widetilde{\boldsymbol{x}}|_{(j)}$, such that sum $m_b = m_1 + \cdots + m_t$ is equal to the Bézout number. Then, for all $d \geq d_*$, there exists a matrix $\boldsymbol{V}_d$ of m_b linearly independent columns, so that

$$\boldsymbol{M}_d \boldsymbol{V}_d = \boldsymbol{0}. \tag{4.64}$$

Furthermore, $\boldsymbol{V}_d$ can be partitioned into

$$\boldsymbol{V}_d = \begin{bmatrix} \boldsymbol{V}_d^{(1)} & \cdots & \boldsymbol{V}_d^{(t)} \end{bmatrix}, \tag{4.65}$$

such that each $\boldsymbol{V}_d^{(j)}$ consists of m_j linear combinations of differential functionals. We call this basis matrix of the right null space the **canonical right null space** of the Macaulay matrix.

Note that the multiplicity structure of a certain common root is not unique. Finding that multiplicity structure is an active area of research and goes way beyond the scope of this chapter [68, 109, 284]. We suppose that we know this structure for now and eliminate its need later on, similar to the approach taken in Chapter 2.

Given the matrix $\boldsymbol{V}_d$, we can describe the monomial multiplicative structure of $\widetilde{\mathcal{P}}_d^{n+1}$ as

$$\boldsymbol{S}_{x_i,x_j} \boldsymbol{V}_d \boldsymbol{D}_{x_i} = \boldsymbol{S}_{x_j,x_i} \boldsymbol{V}_d \boldsymbol{D}_{x_j}, \tag{4.66}$$

where the matrices $\boldsymbol{S}_{x_i,x_j}$ and $\boldsymbol{S}_{x_j,x_i}$ select rows from the canonical right null space $\boldsymbol{V}_d$. Since the multiplication with monomials is commutative, the multiplication matrices $\boldsymbol{D}_{x_i}$ and $\boldsymbol{D}_{x_j}$ must commute. This corresponds to computing the homogeneous multiplication matrices for a monomial basis (selected via $\boldsymbol{S}_{x_i,x_j}$ and $\boldsymbol{S}_{x_j,x_i}$) of $\mathcal{R}[\mathcal{I}]_d$.

Example 4.7. Let us consider a system that has only one projective common root and suppose that $d_\circ = 2$. We can write in this situation the

multiplication property as

$$\boldsymbol{S}_{x_1,x_0} \underbrace{\begin{bmatrix} x_0^2|_{(1)} \\ \hline x_0x_1|_{(1)} \\ x_0x_2|_{(1)} \\ \hline x_1^2|_{(1)} \\ x_1x_2|_{(1)} \\ x_2^2|_{(1)} \end{bmatrix}}_{\partial_{00}(\boldsymbol{v})|_{(1)}} x_1|_{(1)} = \boldsymbol{S}_{x_0,x_1} \underbrace{\begin{bmatrix} x_0^2|_{(1)} \\ \hline x_0x_1|_{(1)} \\ x_0x_2|_{(1)} \\ \hline x_1^2|_{(1)} \\ x_1x_2|_{(1)} \\ x_2^2|_{(1)} \end{bmatrix}}_{\partial_{00}(\boldsymbol{v})|_{(1)}} x_0|_{(1)}, \tag{4.67}$$

where $\boldsymbol{S}_{x_1,x_0}$ and $\boldsymbol{S}_{x_0,x_1}$ select the correct rows for the equality to hold. Of course, when the system has more than one projective common root, the single shift is replaced with multiple shifts in the matrices $\boldsymbol{D}_{x_i}$ and $\boldsymbol{D}_{x_j}$. The canonical right null space, for two simple projective common roots, corresponds to

$$\boldsymbol{V} = \begin{bmatrix} \partial_{00}(\boldsymbol{v})|_{(1)} & \partial_{00}(\boldsymbol{v})|_{(2)} \end{bmatrix}, \tag{4.68}$$

resulting in

$$\boldsymbol{S}_{x_1,x_0}\boldsymbol{V}_d \underbrace{\begin{bmatrix} x_1|_{(1)} & 0 \\ 0 & x_1|_{(2)} \end{bmatrix}}_{\boldsymbol{D}_{x_1}} = \boldsymbol{S}_{x_0,x_1}\boldsymbol{V}_d \underbrace{\begin{bmatrix} x_0|_{(1)} & 0 \\ 0 & x_0|_{(2)} \end{bmatrix}}_{\boldsymbol{D}_{x_0}}. \tag{4.69}$$

When the second projective common root is the same as the first one, i.e., a solution with multiplicity equal to two, the canonical right null space contains two differential functionals evaluated in the same solution, for example,

$$\boldsymbol{V} = \begin{bmatrix} \partial_{00}(\boldsymbol{v})|_{(1)} & \partial_{10}(\boldsymbol{v})|_{(1)} \end{bmatrix}, \tag{4.70}$$

which leads to upper-triangular matrices $\boldsymbol{D}_{x_i}$ and $\boldsymbol{D}_{x_j}$ in

$$\boldsymbol{S}_{x_1,x_0}\boldsymbol{V}_d \underbrace{\begin{bmatrix} x_1|_{(1)} & \times \\ 0 & x_1|_{(1)} \end{bmatrix}}_{\boldsymbol{D}_{x_1}}. = \boldsymbol{S}_{x_0,x_1}\boldsymbol{V}_d \underbrace{\begin{bmatrix} x_0|_{(1)} & \times \\ 0 & x_0|_{(1)} \end{bmatrix}}_{\boldsymbol{D}_{x_0}}. \tag{4.71}$$

When we know that the common roots are affine, we can retrieve the shift structure from (2.53) by setting $x_0 = 1$, resulting in

$$\boldsymbol{S}_1\boldsymbol{V}_d \begin{bmatrix} x_1|_{(1)} & \times \\ 0 & x_1|_{(1)} \end{bmatrix} = \boldsymbol{S}_{x_1}\boldsymbol{V}_d\boldsymbol{I}. \tag{4.72}$$

As shown in Section 2.4.1.2, we can replace the dependency on $\boldsymbol{V}_d$ by a numerical basis matrix of the right null space $\boldsymbol{Z}_d$:

$$(\boldsymbol{S}_{x_i,x_j}\boldsymbol{Z}_d)\boldsymbol{T}\boldsymbol{D}_{x_i} = (\boldsymbol{S}_{x_j,x_i}\boldsymbol{Z}_d)\boldsymbol{T}\boldsymbol{D}_{x_j}, \tag{4.73}$$

Let $\boldsymbol{J}_{x_i,x_j} = \boldsymbol{P}^{-1}\boldsymbol{D}_{x_i}\boldsymbol{D}_{x_j}^{-1}\boldsymbol{P}$ be the Jordan normal form of $\boldsymbol{D}_{x_i}\boldsymbol{D}_{x_j}^{-1}$, then we

obtain

$$(\boldsymbol{S}_{x_i,x_j}\boldsymbol{Z}_d)\boldsymbol{T}\boldsymbol{P}\boldsymbol{J}_{x_i,x_j} = (\boldsymbol{S}_{x_j,x_i}\boldsymbol{Z}_d)\boldsymbol{T}\boldsymbol{P}. \tag{4.74}$$

When there are only affine solutions, $\boldsymbol{D}_{x_0}$ is always diagonalizable. Solutions at infinity require a more careful choice of monomials to shift with, or must be deflated first (Theorem 2.3). This multiplication corresponds with the approach we have taken in Section 2.4. Of course, many details are hidden in the numerical implementation of this idea, many of them are covered in Section 2.4.

4.5.2.2 Explanation of the row selections

One question remains unanswered: "How do we choose a basis for the quotient space $\mathcal{R}[\mathcal{I}]_d$?" Or similarly: "Which rows does the row selection matrix $\boldsymbol{S}_1$ need to pick?" There are several choices possible to create an invertible matrix $(\boldsymbol{S}_1\boldsymbol{Z}_d)$. Typically, one chooses the basis elements to be monomials. This could be determined as the standard monomials from a Gröbner basis computation or the linearly independent rows of the basis matrix of the null space [26].

In a numerical setting, however, it is better to use border bases or more general bases to avoid amplifications of rounding errors [243]. Recently, Telen and Van Barel [244] have presented an adaptive selection of the basis elements via numerical linear algebra techniques, like the QR decomposition with optimal column pivoting. Note that in our numerical implementation, we do not select some basis elements, but take all monomials up to a certain degree (cf., Remark 2.5). This is numerically more robust than selecting only the rows that correspond to the standard monomials. The advantages of this selection is clearly visible in Example 2.20.

4.6 Column space interpretation

In Section 2.5, we have shown that it is also possible to solve a system of multivariate polynomial equations via the column space of the (homogeneous) Macaulay matrix. The fact that there are equivalent eigenvalue problems in the column space suggests that there must be also be an interpretation of this subspace in terms of the projective common roots. While this novel solution approach gives an ubiquitous use to that fundamental subspace, it is at this moment still unclear how to interpret it in the language of algebraic geometry.

Furthermore, the columns of the Macaulay matrix are labeled by the monomials in $\mathcal{C}_d^n$ or $\widetilde{\mathcal{C}}_d^{n+1}$. The complementarity between the column space and the right null space result in a correspondence between the linearly dependent columns from right to left in the Macaulay matrix with the linearly independent rows from top to bottom in the right null space; hence, they correspond also to the standard monomials. The fact that the rank counts both the number of linearly independent rows in the row space, which correspond to the linearly independent leading monomials in that row space [26], and the linearly independent columns in the colum space points also at a link with the canonical decomposition of the set of monomials for a system of multivariate polynomials [26].

4.7 Conclusion

It is clear from this chapter that each of the fundamental subspaces of the (homogeneous) Macaulay matrix has a purpose. Many properties/questions from algebraic geometry hide in one of the fundamental subspaces of the Macaulay matrix. Although a full treatment of the column space is not yet available, it is clear that there must also be an interpretation of this subspace in terms of the generating polynomials.

The attempt to summarize the different fundamental subspaces of the Macaulay matrix has created three potential research avenues:

- One of the current research efforts is to investigate the properties of the column space of the Macaulay matrix. This may yield us a better interpretation of the matrix and maybe also provide with additional algorithmic opportunities.
- Since the right null space of the Macaulay matrix can also be interpreted in a system theoretic language as the column space of the observability matrix of a multidimensional descriptor system, for which the eigenvalues of the system matrices corresponds to the common roots of the polynomials of the Macaulay matrix, a full system theoretic treatment of the subspaces of the Macaulay matrix is also very interesting.
- Furthermore, we could ask ourselves the question whether we can come up with a similar treatment of the block Macaulay matrix in the future. The fact that the right null space of the block Macaulay matrix can also be interpreted as the column space of a multidimensional observability matrix fuels the idea that this might be possible [259].

Historical and bibliographical notes

We refer at several places in this dissertation to the eigenvalue-eigenvector approach as Stetter's eigenvalue-eigenvector approach (sometimes also called the Stetter–Möller approach). The fundamental relation between eigenvalues and the quotient ring of the ideal was probably already known in the late 19th and early 20th century, being it in the specific mathematical language used at that point in time [230]. The fact that there were numerical tools to solve eigenvalue problems may have been a reason why this eigenvalue approach to solve multivariate polynomial systems was not elaborated further upon at that time. The eigenvalue-eigenvector theorem has been linked also to Ludwig Stickelberger, who has formulated a theorem similar to the one presented here [64, 230]. However, his theorem has been remained concealed for a long time through history. In the 1980s, there were new allusions by Lazard [153, 154] in the direction of a link between eigenvalues and the common roots of polynomials, but only in terms of the eigenvalues. Stetter [229] established a few years later the connection between the eigenvalue decomposition (eigenvalues and eigenvectors) and multivariate polynomial system solving.

PART

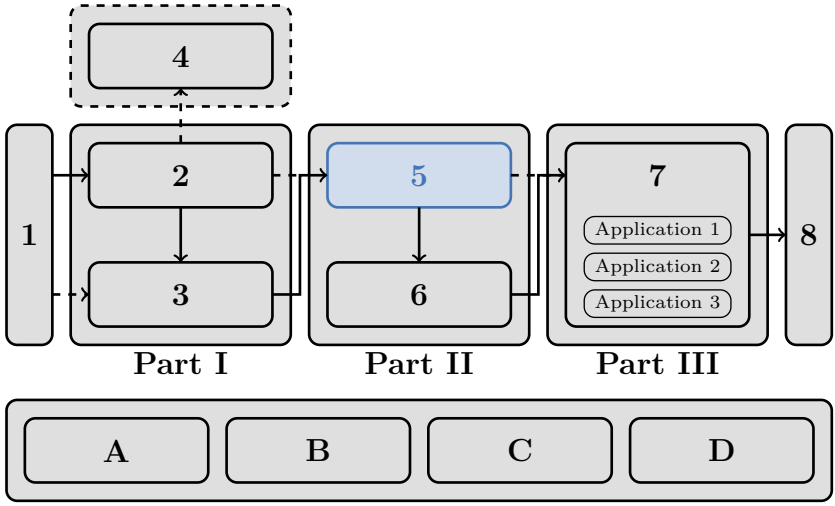
II

Algorithms

Recursive Algorithms for the (Block) Macaulay Matrix

As demonstrated in the previous chapters, the construction of the (block) Macaulay matrix typically follows an iterative approach. In order to know whether the solution degree has been reached, i.e., whether the right null space of the (block) Macaulay matrix in the current iteration can accommodate the shift polynomial, it is necessary to compute a basis matrix of this right null space in every iteration and to examine the dimension or rank structure of that computed basis matrix. Consequently, recursively updating the numerical basis matrix of the right null space, while exploiting the inherent structure of the matrices involved, can induce large savings in computation time. Moreover, the process of checking the rank structure of the computed basis matrix is again an iterative procedure: this basis matrix can be interpreted as a block row matrix, where every degree block corresponds to one block row, and the rank structure must be checked by adding these block rows one by one.

In this chapter, we zoom in on the development of recursive techniques for the null space based (block) Macaulay matrix approach. We discuss the recursive computation of a basis matrix of the null space and the recursive rank checks. Combining of both recursive techniques leads to double recursive algorithms for polynomial system solving and multiparameter eigenvalue computing.



Contributions. Firstly, we introduce recursive algorithms to update a basis matrix of the right null space of the (block) Macaulay matrix and block row matrix. Secondly, the proposed sparse adaptations in this chapter of the recursive algorithms to update a basis matrix of the right null space for the (block) Macaulay matrix results in a considerable reduction of the required memory. Finally, by combining sparse and recursive techniques, double recursive algorithms are able to solve systems of multivariate polynomial equations and multiparameter eigenvalue problems more efficiently.

Relevant articles. This chapter contains the combined content of [260, 262]. The candidate was the main author of both original articles, developed the theoretical contributions, and implemented the accompanying code and experiments. The notation is changed compared to the original text for consistency, some sections of the original articles are (re)moved from this chapter to avoid redundancy, and other numerical examples are added to highlight the improvements over the other chapters.

Outline. After introducing the different ingredients of the double recursive algorithms in Section 5.1, we develop in Section 5.2 a recursive algorithm to determine a numerical basis matrix of the right null space of the block Macaulay matrix, which reduces to an algorithm that can be used for the Macaulay matrix, when considering coefficients instead of coefficient matrices. Section 5.3 contains sparse adaptations for updating this basis matrix. Next, in Section 5.4, we consider a recursive algorithm to update a numerical basis matrix of the right null space of the block row matrix, i.e., to perform recursive rank checks. In Section 5.5, we combine the different recursive and sparse techniques of this chapter into double recursive algorithms. Section 5.6 concludes this chapter and contains some ideas for future research.

5.1 Introduction

The (right[1]) null space and column space of the Macaulay matrix play a crucial role when solving the generating system of multivariate polynomial equations. In Chapter 2, we have shown how to leverage the structure of the null space in order to retrieve the common roots of the system of polynomials: a multi-dimensional realization problem in a basis matrix of the null space that can accommodate the shift yields the solutions. Similarly, the null space of the block Macaulay matrix can be used to retrieve the eigenvalues of the (rectangular[2]) multiparameter eigenvalue problem (MEP) that generate that block Macaulay matrix, as presented in Chapter 3. The computation of a numerical basis matrix of the null space is an important step in both null space based solution approaches.

As demonstrated in the previous chapters, the (block) Macaulay matrix is typically constructed in an iterative fashion, since the solution degree is not known in advance. In order to know whether the solution degree has been reached (i.e., degree for which the null space can accommodate the shift), the rank structure of the basis matrix of the null space needs to be checked in every iteration. Checking that rank structure is again an iterative procedure, which poses two computational problems: (i) efficiently updating the basis matrix for subsequent degrees of the (block) Macaulay matrix and (ii) efficiently checking the rank structure of that basis matrix in every iteration. In this chapter, we address these two computational problems via recursive[3] algorithms.

Ingredient 1: null space of a (block) Macaulay matrix

The required degree of the (block) Macaulay matrix in applications depends on the properties of its null space. Because these properties can not be deduced in advance, we need to enlarge the (block) Macaulay matrix iteratively and compute in every iteration a new numerical basis matrix of the null space. Several authors have already addressed the direct null space computation of structured matrices [142, 167], but a recursive approach that exploits, next to the structure and sparsity, the iterative nature of these special matrices clearly has a lot of potential. Batselier et al. [22] have developed, therefore, a recursive algorithm to update a numerical basis matrix of the null space of the Macaulay matrix. However, algorithms to update a basis matrix of the null space of the block Macaulay matrix were absent in the literature until our extension in [260]. Note that these algorithms can also be used to tackle the other matrices of the (block) Macaulay approach: the algorithms can also deal with block (banded) Toeplitz matrices, Macaulay matrices, and (banded) Toeplitz matrices.

[1] In the remainder of this chapter, we no longer mention the qualification *right* explicitly. We always consider the right null space, except when denoted otherwise.

[2] In the remainder of this chapter, we no longer mention the qualification *rectangular* explicitly. We always consider rectangular problems, except when denoted otherwise.

[3] We do not use the term *recursion* in its strict computer science meaning ("an algorithm that calls itself on smaller input values"), but see it as an algorithm that performs the same steps on different input values ("an algorithm that uses in every iteration the same approach on new input values"), cf., the recursive least-squares algorithm [129].

Ingredient 2: null space of a block row matrix

The question whether the null space can accommodate the shift can only be answered by checking its rank structure. In the case of a zero-dimensional solution set (i.e., every solution of the problem is an isolated point in the solution space), the nullity of the (block) Macaulay matrix reveals the total number of solutions, both affine and at infinity. After stabilization of that nullity (i.e., when all solutions are present in the null space), rank checks on growing submatrices of the numerical basis matrix are required to separate the affine solutions from the solutions at infinity. Moreover, for problems with a positive-dimensional solution set at infinity, these rank checks are required in every iteration, because there is no stabilization of the nullity anymore. Since a numerical basis matrix of the null space of a (block) Macaulay matrix is typically a dense (i.e., non-sparse) tall matrix, it can be considered as a block row matrix, where we iterate over its subsequent (degree) blocks in order to determine the rank structure, i.e., we check the change of the rank for every additional degree block of the numerical basis matrix in order to know whether the null space has a gap zone to separate the affine solutions from the solutions at infinity (Sections 2.4.3 and 3.4.3).

The iterative construction of a block row matrix is a very active subject in the signal processing literature: in many signal processing applications [2, 178, 179], new data vectors in the (block) rows are appended continuously. The process of appending new (block) rows induces the iterative structure naturally. A mature body of literature already covers the (block) row-wise updating of the singular value decomposition [50, 179] or tracking of a subspace [2, 179, 231, 232]. In this chapter, we restrict ourselves to the particular subproblem where we only update in every iteration a basis matrix of the null space of the block row matrix using results from the previous iteration, which we have presented also in [260]. The block row matrix mainly under consideration in this chapter is the basis matrix of the null space of a (block) Macaulay matrix: we want to iterate over the different degree blocks of that matrix to obtain the rank structure. A recursive algorithm that takes into account the iterative nature of the problem can be very beneficial.

Combining both ingredients in one algorithm

In this chapter, we thus address these two computational problems and propose recursive algorithms to update an orthogonal numerical basis matrix of the null space of the (block) Macaulay and block row matrix, using results from the previous iteration. We extend in this chapter the algorithm of Batselier et al. [22] to the block Macaulay matrix and develop a sparse adaptation for both the Macaulay and block Macaulay matrix, which avoids the explicit construction of the (block) Macaulay matrix and results in a considerable memory improvement compared to its dense counterparts. A similar insight for the block row matrix allows us to recursively update a basis matrix of the null space of that matrix. The combination of the recursive or sparse technique to compute a basis matrix of the null space of the (block) Macaulay matrix with a recursive approach to perform the rank checks in that basis matrix of the null space leads to double

recursive algorithms to solve systems of multivariate polynomial equations and MEPs. The computational results are impressive: when we use the null space based block Macaulay matrix approach to solve, for example, a linear two-parameter eigenvalue problem, we notice that the proposed double recursive approaches are 95–276 times faster than the standard non-recursive approach.

Motivational example. The size of the multivariate polynomial system and MEP that result in the globally optimal parameters of the first-order autoregressive moving-average (ARMA) model identification problem in Chapters 2 and 3 depends on the considered number of data points. When more data points are considered, the problems quickly grow to become untractable for the naive (block) Macaulay matrix algorithms in the previous chapters. The (block) Macaulay matrix algorithms clearly exhibit an iterative characteristic; different iterations are necessary before the degree d of the (block) Macaulay matrix reaches the solution degree $d_\circ$. For example, in order to solve the quadratic two-parameter eigenvalue problem in (3.90), a block Macaulay matrix of degree $d = 19$ is necessary. From Figure 3.4, it is clear that computing a basis matrix of the null space and performing the necessary rank checks create a bottleneck for these (block) Macaulay approaches. However, by applying the above-mentioned ingredients, it should be possible to solve the polynomial system in (2.2) and MEP in (3.90) much faster!

Remark 5.1. Because of the different setting (i.e., we focus more on the computational aspects), we use a different indexing in this chapter. We use i for the iteration index and j to iterate over the coefficients and coefficient matrices. The former index is related to the degree of the block row and (block) Macaulay matrix, while the latter is related to the multi-index $\boldsymbol{\alpha}$ or $\boldsymbol{\omega}$. For example, a block Macaulay matrix in iteration $i = 0$ has degree $d_{\max}$ and, if the generating MEP has two parameters, then $j = 1, 2, 3$ correspond to $\boldsymbol{\omega} = (0, 0), (1, 0), (0, 1)$, since we use the graded inverse lexicographic (GRINVLEX) ordering.

Remark 5.2. In order to construct a random matrix $\boldsymbol{M} \in \mathbb{R}^{p \times q}$ with a specific rank r, we multiply two random matrices $\boldsymbol{N} \in \mathbb{R}^{p \times r}$ and $\boldsymbol{P} \in \mathbb{R}^{r \times q}$, which have by construction a rank equal to r. Throughout this chapter, we always use Matlab's `randn` function to generate normally distributed (pseudo)random matrices.

5.2 Recursive null space computations

In this section, we study the computation of a basis matrix of the null space of the block Macaulay matrix[4], which is an essential step in the null space based

[4]Note that the approach presented in this section has similarities with the recursive algorithm presented in [22, Algorithm 3.1] when considering the (scalar) Macaulay matrix, although it was developed independently (Remark 5.4).

approach to solve MEPs. In accordance to Remark 5.1, the seed matrices are indicated by a single subscript j, i.e., $\boldsymbol{A}_j \in \mathbb{C}^{k\times l}$ $(j = 1, \ldots, x+y)$. When we are solving MEPs via the block Macaulay matrix, these seed matrices correspond to the coefficient matrices of that MEP, e.g.,

$$\begin{aligned} \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) &= \big(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\lambda_1 + \boldsymbol{A}_{01}\lambda_2 + \boldsymbol{A}_{20}\lambda_1^2 + \boldsymbol{A}_{11}\lambda_1\lambda_2 + \boldsymbol{A}_{02}\lambda_2^2\big)\boldsymbol{z} = \boldsymbol{0} \\ &\Downarrow \\ \boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}) &= \big(\boldsymbol{A}_1 + \boldsymbol{A}_2\lambda_1 + \boldsymbol{A}_3\lambda_2 + \boldsymbol{A}_4\lambda_1^2 + \boldsymbol{A}_5\lambda_1\lambda_2 + \boldsymbol{A}_6\lambda_2^2\big)\boldsymbol{z} = \boldsymbol{0}. \end{aligned} \tag{5.1}$$

We make, in the remainder of this chapter, abstraction of the monomial that a certain coefficient matrix is associated with. The structure of a block Macaulay matrix is visualized in Figure 5.1. It is clear that this matrix is very structured and sparse. This structure leads to a recursive definition of the block Macaulay matrix $\boldsymbol{M}_i \in \mathbb{C}^{p_i\times q_i}$ in iteration i:

$$\boldsymbol{M}_i = \begin{bmatrix} \boldsymbol{M}_{i-1}^1 & \boldsymbol{M}_{i-1}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_i & \boldsymbol{Y}_i \end{bmatrix}, \tag{5.2}$$

where the matrix $\boldsymbol{X}_i \in \mathbb{C}^{m_i\times s_i}$ gathers all the seed matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_x$ (but also some zero matrices) below $\boldsymbol{M}_{i-1}^2$ and the matrix $\boldsymbol{Y}_i \in \mathbb{C}^{m_i\times t_i}$ contains the remaining seed matrices $\boldsymbol{A}_{x+1}, \ldots, \boldsymbol{A}_{x+y}$ (and also some zero matrices) under the zero block. Notice that the matrices $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ depend on i, because every iteration consists of a different number of shifts $\sigma_{\max}$. The matrices $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$, together with the iteration i and shift σ, are annotated in Figure 5.1. The sizes[5] of $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ depend on the number of shifts $\sigma_{\max}$ in that particular i:

$$m_i = k\binom{i+n-1}{n-1} = \frac{k}{(n-1)!}i^{n-1} + \mathcal{O}\big(i^{n-2}\big), \tag{5.3}$$

$$s_i = l\sum_{d=0}^{d_{\max}-1}\binom{i+d+n-1}{n-1} = \frac{l}{(n-1)!}i^{n-1} + \mathcal{O}\big(i^{n-2}\big), \tag{5.4}$$

$$t_i = l\binom{i+d_{\max}+n-1}{n-1} = \frac{l}{(n-1)!}i^{n-1} + \mathcal{O}\big(i^{n-2}\big), \tag{5.5}$$

where n is the number of eigenvalues of the generating MEP (i.e., the number of variables in the block Macaulay matrix) and $d_{\max}$ is the degree of the generating MEP (i.e., the highest total degree of the monomials in the MEP). The block Macaulay matrix $\boldsymbol{M}_i$ in iteration i has degree $d_{\max} + i$. The number of rows p_i and columns q_i with respect to i can be deduced from (3.51) and (3.52) and grow quickly, due to the combinatorial explosion of the number of shifts:

$$p_i = k\binom{i+n}{n} = \frac{k}{n!}i^n + \mathcal{O}\big(i^{n-1}\big), \tag{5.6}$$

$$q_i = l\binom{i+d_{\max}+n}{n} = \frac{l}{n!}i^n + \mathcal{O}\big(i^{n-1}\big). \tag{5.7}$$

[5]The notation $\mathcal{O}$ describes the limit behavior of the size. The "Big-Oh" notation $\varphi(t) = \mathcal{O}(\xi(t))$ has a precise meaning in mathematics. It asserts that there exists some positive constant C such that, for all t sufficiently close to an understood limit (e.g., $t \to 0$ or $t \to \infty$), $\varphi(t) \leq C\xi(t)$ [193, 249]. The focus is typically on the highest order operations that are involved, as they tend to dominate the overall complexity [97].

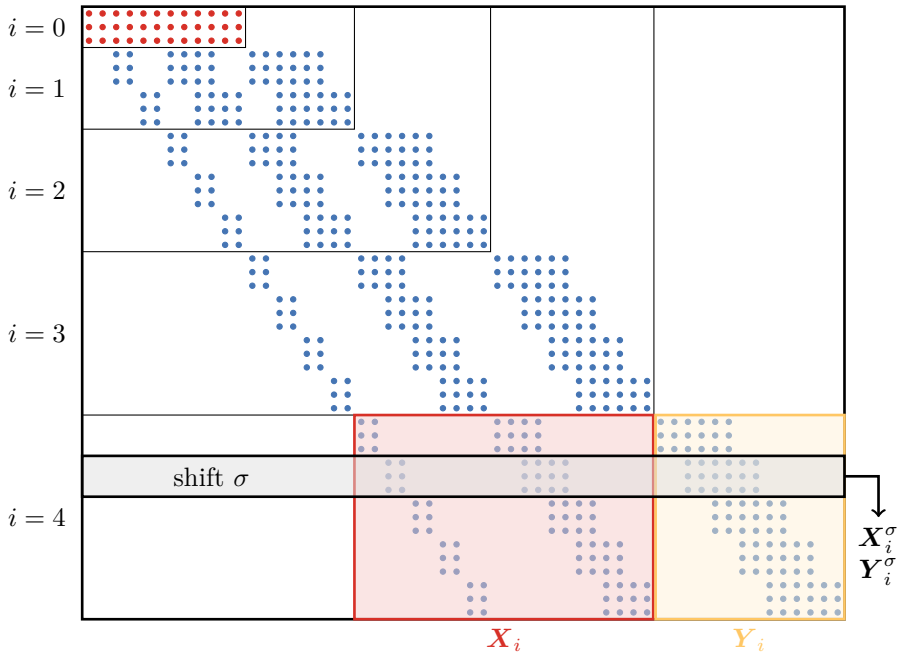


Figure 5.1. Annotated visualization of the iterative construction of a block Macaulay matrix $\boldsymbol{M}_4$ ($n = 2$ and $d_{\max} = 2$) of the quadratic two-parameter eigenvalue problem in (5.1) with rectangular seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{3\times 2}$, for $j = 1, \ldots, 6$. Due to the combinatorial explosion of the number of shifts, the block Macaulay matrix grows quickly very large. Notice that the sizes of $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ of the block Macaulay matrix depend on i, since the maximum number of shifts $\sigma_{\max}$ changes in every iteration.

Typically, the desired iteration $i_\circ$ of the block Macaulay matrix depends on the structure of its null space and is not known in advance. Hence, when we want to compute a numerical basis matrix of the null space for every iteration i, e.g., in order to determine the solutions of the generating MEP, we have to extend the block Macaulay matrix in an iterative way and recompute a numerical basis matrix of its null space in every iteration. Clearly, a recursive algorithm to update this numerical basis matrix poses itself useful in this type of practical situations. We sketch the problem of iteratively updating the numerical basis matrix of the null space of the block Macaulay matrix in Algorithm 5.1, which is similar to the outline provided in Algorithm 3.1. Algorithm 5.1 also fits in the polynomial system solving setting when using the Macaulay matrix, as described in Algorithm 2.1.

In the remainder of this section, we develop a recursive algorithm to determine an orthogonal numerical basis matrix $\boldsymbol{Z}_i$ of the null space of the block Macaulay matrix $\boldsymbol{M}_i$ (Section 5.2.1) and determine the computational com-

Algorithm 5.1 Iterative updating of the (block) Macaulay matrix null space

Require: $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ or $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$
1: $i \leftarrow 0$
2: Construct the (block) Macaulay matrix $\boldsymbol{M}_0$
3: **while** $i < i_\circ$ **do**
4: $\quad \boldsymbol{Z}_i \leftarrow \text{NULL}(\boldsymbol{M}_i)$, for example, via Algorithm 5.2
5: $\quad$ **if** structure of the null space contains the solutions **then**
6: $\quad\quad i = i_\circ$
7: $\quad$ **else**
8: $\quad\quad i \leftarrow i + 1$
9: $\quad\quad$ Determine $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$
10: $\quad\quad \boldsymbol{M}_i \leftarrow \begin{bmatrix} \boldsymbol{M}_{i-1}^1 & \boldsymbol{M}_{i-1}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_i & \boldsymbol{Y}_i \end{bmatrix}$
11: $\quad$ **end if**
12: **end while**
13: **return** $\boldsymbol{Z}_{i_\circ}$

plexity afterwards (Section 5.2.2). numerical examples illustrate the properties of the standard and recursive algorithm (Section 5.2.3).

5.2.1 Recursive algorithm

Consider a block Macaulay matrix $\boldsymbol{M}_{i-1} \in \mathbb{C}^{p_{i-1} \times q_{i-1}}$ after $i-1$ iterations and an orthogonal numerical basis matrix $\boldsymbol{Z}_{i-1} \in \mathbb{C}^{q_{i-1} \times n_{i-1}}$ of its null space, with nullity n_{i-1}, such that

$$\boldsymbol{M}_{i-1}\boldsymbol{Z}_{i-1} = \boldsymbol{0}. \tag{5.8}$$

If we now extend $\boldsymbol{M}_{i-1}$ with t_i zero columns, then we can write

$$\begin{bmatrix} \boldsymbol{M}_{i-1} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}_{i-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{t_i} \end{bmatrix} = \boldsymbol{0}. \tag{5.9}$$

The nullity of this extended matrix $\begin{bmatrix} \boldsymbol{M}_{i-1} & \boldsymbol{0} \end{bmatrix}$ equals $n_{i-1} + t_i$. If we add the next block row of the block Macaulay matrix, i.e., we consider the block Macaulay matrix $\boldsymbol{M}_i$, then we know that there exists an orthogonal matrix $\boldsymbol{V}_i \in \mathbb{C}^{(n_{i-1}+t_i) \times n_i}$, such that

$$\begin{bmatrix} \boldsymbol{M}_{i-1}^1 & \boldsymbol{M}_{i-1}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_i & \boldsymbol{Y}_i \end{bmatrix} \begin{bmatrix} \boldsymbol{Z}_{i-1}^1 & \boldsymbol{0} \\ \boldsymbol{Z}_{i-1}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{t_i} \end{bmatrix} \boldsymbol{V}_i = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{X}_i\boldsymbol{Z}_{i-1}^2 & \boldsymbol{Y}_i \end{bmatrix} \boldsymbol{V}_i = \boldsymbol{0}, \tag{5.10}$$

where $\boldsymbol{Z}_{i-1}$ is partitioned in accordance with $\boldsymbol{M}_{i-1}$. The top part is simplified by the observation that $\boldsymbol{M}_{i-1}^1\boldsymbol{Z}_{i-1}^1 + \boldsymbol{M}_{i-1}^2\boldsymbol{Z}_{i-1}^2 = \boldsymbol{M}_{i-1}\boldsymbol{Z}_{i-1} = \boldsymbol{0}$. Hence, from the bottom part of (5.10), it follows that

$$\begin{bmatrix} \boldsymbol{X}_i\boldsymbol{Z}_{i-1}^2 & \boldsymbol{Y}_i \end{bmatrix} \boldsymbol{V}_i = \boldsymbol{0}, \tag{5.11}$$

Algorithm 5.2 Recursive null space algorithm for the block Macaulay matrix

Require: $\boldsymbol{Z}_{i-1}$, $\boldsymbol{X}_i$, and $\boldsymbol{Y}_i$
1: $\boldsymbol{W}_i \leftarrow \boldsymbol{X}_i \boldsymbol{Z}_{i-1}^2$
2: $\boldsymbol{V}_i \leftarrow \text{NULL}\left(\begin{bmatrix}\boldsymbol{W}_i & \boldsymbol{Y}_i\end{bmatrix}\right)$
3: $\boldsymbol{Z}_i \leftarrow \begin{bmatrix}\boldsymbol{Z}_{i-1}\boldsymbol{V}_i^1 \\ \boldsymbol{V}_i^2\end{bmatrix}$
4: **return** $\boldsymbol{Z}_i$

which means that the $\boldsymbol{V}_i$ is a basis matrix of the null space of $\begin{bmatrix}\boldsymbol{X}_i\boldsymbol{Z}_{i-1}^2 & \boldsymbol{Y}_i\end{bmatrix}$ and

$$\underbrace{\begin{bmatrix}\boldsymbol{M}_{i-1}^1 & \boldsymbol{M}_{i-1}^2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_i & \boldsymbol{Y}_i\end{bmatrix}}_{\boldsymbol{M}_i} \underbrace{\begin{bmatrix}\boldsymbol{Z}_{i-1}^1\boldsymbol{V}_i^1 \\ \boldsymbol{Z}_{i-1}^2\boldsymbol{V}_i^1 \\ \boldsymbol{V}_i^2\end{bmatrix}}_{\boldsymbol{Z}_i} = \boldsymbol{0}, \tag{5.12}$$

where $\boldsymbol{V}_i$ is partitioned into $\boldsymbol{V}_i^1 \in \mathbb{C}^{n_{i-1}\times n_i}$ and $\boldsymbol{V}_i^2 \in \mathbb{C}^{t\times n_i}$. Consequently, an orthogonal numerical basis matrix $\boldsymbol{Z}_i$ of $\boldsymbol{M}_i$ can be computed as

$$\boldsymbol{Z}_i = \begin{bmatrix}\boldsymbol{Z}_{i-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{t\times t}\end{bmatrix}\boldsymbol{V}_i = \begin{bmatrix}\boldsymbol{Z}_{i-1}\boldsymbol{V}_i^1 \\ \boldsymbol{V}_i^2\end{bmatrix}. \tag{5.13}$$

We summarize in Algorithm 5.2 the different steps of the entire recursive approach. An efficient implementation, of course, tries to avoid the zero blocks and uses fast multiplications that exploit the available structure, an improvement that is naturally incorporated in a sparse adaptation (Algorithm 5.5).

Remark 5.3. Algorithm 5.2 considers an iteration-wise growth of the block Macaulay matrix and recomputes the numerical basis matrix in an iteration-wise fashion (cf., every iteration corresponds to one degree). One notices easily that the same idea could also work if the recursive approach is applied in a shift-wise fashion (i.e., for every shifted block row in the block Macaulay matrix). Moreover, in a shift-wise fashion, the zero blocks are easier to identify and avoid. The main drawback of this alternative shift-wise implementation is the fact that, for every iteration, multiple multiplications and null space computations are necessary, which cancels the above-mentioned computational advantages. Section 5.2.3.1 contains a numerical example that compares both approaches.

Remark 5.4. Algorithm 5.2 can also be used for (scalar) Macaulay matrices and block banded (Toeplitz) matrices:

- For the former type of matrices, the recursive computation of a basis matrix of the null space reduces to the algorithm presented in [22, Algorithm 3.1], when the coefficient matrices are replaced by scalar coefficients. One slight difference is that Algorithm 5.2 avoids the zero matrix under $\boldsymbol{M}_{i-1}^1$ (Figure 5.2a).

- When the number of parameters $n = 1$, the block Macaulay matrix reduces to the latter type of matrices (Figure 5.2b). Since Algorithm 5.2 does not explicitly make use of the repetitive structure in the block Macaulay matrix (i.e., the fact that same seed matrices appear in every block row), it can be applied to tackle both block banded Toeplitz matrices (the matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the same in every iteration) and block banded matrices without fixed seed matrices (the matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are different in every iteration). Example 5.6 contains a numerical example with such block (banded) Toeplitz matrices.

5.2.2 Computational complexity

The computational complexity of both the standard and the recursive approach is analyzed in this section. We derive theoretical expressions first (Section 5.2.2.1) and, afterwards, we verify the results experimentally (Section 5.2.2.2).

5.2.2.1 Theoretical complexity analysis

To determine the computational complexity, we substitute the number of rows p_i and columns q_i of the block Macaulay matrix $\boldsymbol{M}_i$ (5.6) and (5.7) into the computational cost of computing the singular values and right singular vectors, i.e., $4p_iq_i^2 + 8q_i^3$ floating-point operations (FLOPs) [97, p. 493], which results in the computational cost of the **standard algorithm** (in FLOPs):

$$\frac{4kl^2}{n!^3}i^{3n} + \frac{8l^3}{n!^3}i^{3n} + \mathcal{O}\big(i^{3n-1}\big) = \mathcal{O}\big(i^{3n}\big). \tag{5.14}$$

Most of the times, the seed matrices $\boldsymbol{A}_j$ are square or close to square (i.e., $k \approx l$):

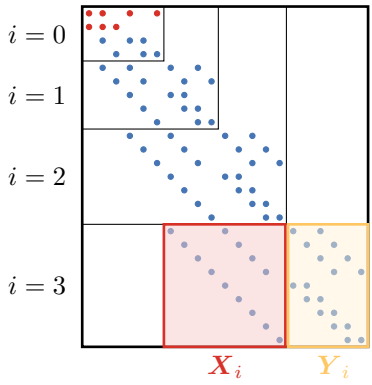
$$\frac{12l^3}{n!^3}i^{3n} + \mathcal{O}\big(i^{3n-1}\big) = \mathcal{O}\big(i^{3n}\big). \tag{5.15}$$

The proposed **recursive algorithm** consists of three main steps (Algorithm 5.2 – in FLOPs):

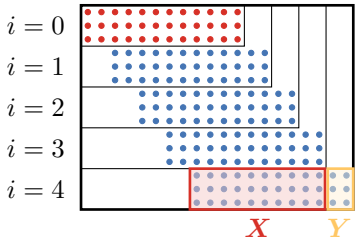
$$\begin{aligned} 2m_is_in_{i-1} &\quad \text{(multiplication – line 1)},\\ 4m_i(n_{i-1}+t_i)^2 + 8(n_{i-1}+t_i)^3 &\quad \text{(null space computation – line 2)},\\ 2q_{i-1}n_{i-1}n_i &\quad \text{(multiplication – line 3)}. \end{aligned}$$

The polynomial n_i describes the nullity of $\boldsymbol{M}_i$ with respect to the iteration i:

$$\begin{aligned} n_i &= q_i - r_i && (5.16)\\ &= q_i - p_i && (5.17)\\ &= \frac{l}{n!}i^n - \frac{k}{n!}i^n + \mathcal{O}\big(i^{n-1}\big) && (5.18)\\ &\leq \frac{\phi}{(n-1)!}i^{n-1} = \mathcal{O}\big(i^{n-1}\big), && (5.19) \end{aligned}$$



(a) Macaulay matrix (iteration $i = 3$ and degree $d = 5$)



(b) Block banded (Toeplitz) matrix (iteration $i = 4$ and degree $d = 10$)

Figure 5.2. Annotated visualization of a Macaulay matrix $\boldsymbol{M}_3$ and block banded (Toeplitz) $\boldsymbol{T}_4$. $\boldsymbol{M}_3$ consists of the same polynomials as in Figure 2.5, but the rows have a different ordering: instead of ordering the different shifts per polynomial, the Macaulay matrix is build iteratively and every iteration corresponds to a higher degree. $\boldsymbol{T}_4$ is the block (banded) Toeplitz matrix that is generated from a polynomial eigenvalue problem with rectangular seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{3\times 2}$, for $j = 1, \ldots, 6$. Notice that the sizes of $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ of the (block) Macaulay matrix depend on i, while this is not the case for $\boldsymbol{X}$ and $\boldsymbol{Y}$ in the block banded (Toeplitz) matrix.

Table 5.1. Dominant term(s) of the computational complexity (in FLOPs per iteration i) of the standard and recursive algorithm to compute a numerical basis matrix of the null space of the block Macaulay matrix $\boldsymbol{M}_i$, for both rectangular $k \times l$ and square $l \times l$ seed matrices $\boldsymbol{A}_j$. The rank r_i is assumed to be equal to the number of rows p_i of $\boldsymbol{M}_i$ for iteration $i \leq i_\circ$ and two factors ϕ and $\phi^{'}$ are introduced that do not depend on i.

algorithm	**rectangular**	**square**
standard	$\frac{4kl^2+8l^3}{n!^3} i^{3n}$	$\frac{12l^3}{n!^3} i^{3n}$
recursive	$\left(\frac{\phi^{'3}}{(n-1)!^3} + \frac{2l\phi^2}{n(n-1)!^3}\right) i^{3n-2}$	$\frac{2k\phi^2}{n(n-1)!^3} i^{3n-2}$

where we assume in (5.17) that the rank is equal to the number of rows for $i < i_\circ$ and introduce a factor ϕ (and also $\phi^{'}$ below) in (5.19) that does not depend on i, but depends linearly on the size of the seed matrices (i.e., $\mathcal{O}(k, l)$). We remove the highest order terms in our upper bound, since $k \geq l$ in practical applications (otherwise the nullity does not stabilize). The computational complexity of the recursive algorithm is then bounded above by (in FLOPs)

$$\frac{\phi^{'3}}{(n-1)!^3} i^{3n-3} + \frac{2l\phi^2}{n(n-1)!^3} i^{3n-2} = \mathcal{O}\big(i^{3n-2}\big), \tag{5.20}$$

which remains the same expression when $k = l$ (only the factors ϕ and $\phi^{'}$ change).

The computational complexity of the recursive algorithm (per iteration i) corresponds to $\mathcal{O}\big(i^{3n-2}\big)$, which is due to the dominating multiplication. If we compare this to the standard singular value decomposition, which has a computational complexity $\mathcal{O}(i^{3n})$, the recursive algorithm gains two orders of magnitude. We summarize the computational complexities in Table 5.1.

5.2.2.2 Experimental complexity analysis

We use numerical examples to verify whether the derived computational complexities in Table 5.1 are correct. First, we consider, for different iterations i, the block Macaulay matrix generated by a linear 2-parameter eigenvalue problem.

Example 5.1. We build, for different iterations i, the block Macaulay matrix $\boldsymbol{M}_i$ of a linear 2-parameter eigenvalue problem with 3 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{21\times 20}$. For every iteration i, we consider the standard algorithm, which computes the null space from $\boldsymbol{M}_i$, and the recursive algorithm, which uses the new block rows of $\boldsymbol{M}_i$ and previous basis matrix $\boldsymbol{Z}_{i-1}$ of the null space. The log-log plot in Figure 5.3 shows that computation time of the recursive algorithm with respect to i is two powers smaller than the standard algorithm, as in Table 5.1.

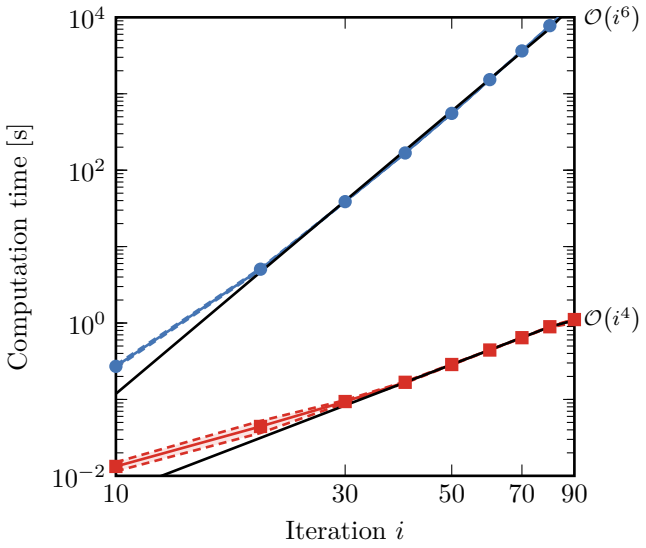


Figure 5.3♣. Experimental complexity analysis of the iteration i on the null space computation for the block Macaulay matrix. The mean computation time for the standard (—●—) and recursive (—■—) algorithm to compute a numerical basis matrix of the null space of a block Macaulay matrix $\boldsymbol{M}_i$ is averaged over 30 experiments (the dashed lines indicate one standard deviation). $\boldsymbol{M}_i$ is generated by a linear 2-parameter eigenvalue problem with 3 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{21\times 20}$. The computation times of both algorithms follow the theoretical complexities (——) in Table 5.1.

Next, we consider the influence of the size of the seed matrices on the computation time, which should be cubic (Table 5.1).

Example 5.2. In Figure 5.4, we visualize the total time to compute a numerical basis matrix of the null space of a block Macaulay matrix $\boldsymbol{M}_{15}$ from $i = 0$ to the desired iteration $i_\circ = 15$, i.e., the total computation time to iteratively reach $i_\circ$. We consider a linear 2-parameter eigenvalue problem with 3 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{(l+1)\times l}$, where we increase the size of the seed matrices during the numerical example. The computation time grows indeed cubically with the number of columns of the seed matrices, as in Table 5.1.

Table 5.1 claims that the computation time strongly depends on the number of variables. The following example supports this claim.

Example 5.3. We repeat the previous example, but now we consider a linear n-parameter eigenvalue problem with $n + 1$ random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{(4+n)\times 5}$, where we increase the number of parameters in every step. We visualize in Figure 5.4 the total time to compute a numerical basis matrix of the null space of a block Macaulay matrix for desired iteration $i_\circ = 15$.

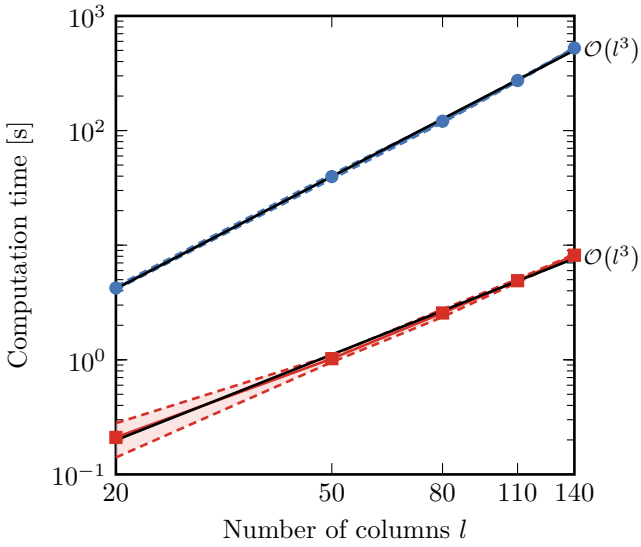


Figure 5.4♣. Experimental complexity analysis of the seed matrix size $k \times l$ on the null space computation for the block Macaulay matrix. The mean total computation time for the standard (—●—) and recursive (—■—) algorithm to compute a numerical basis matrix of the null space of a block Macaulay matrix $\boldsymbol{M}_{15}$ is averaged over 30 experiments (the dashed lines indicate one standard deviation). $\boldsymbol{M}_i$ is generated by a linear 2-parameter eigenvalue problem with 3 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{(l+1)\times l}$. The computation times of both algorithms follow the theoretical complexities (——) in Table 5.1.

The influence of the number of parameters corresponds (more or less) to the computational complexities derived in Table 5.1.

5.2.3 Some numerical examples

We illustrate the properties of the recursive null space computation via several numerical examples with random seed matrices. We use the algorithms as we would do in a practical setting: starting from the initial block Macaulay matrix ($i = 0$) and increasing that matrix with one degree in every iteration. Note that we have performed many more numerical examples, which the interested reader can find in [260].

5.2.3.1 Random block Macaulay matrix

In the first numerical example, we consider random block Macaulay matrices generated by a random linear 2-parameter eigenvalue problem.

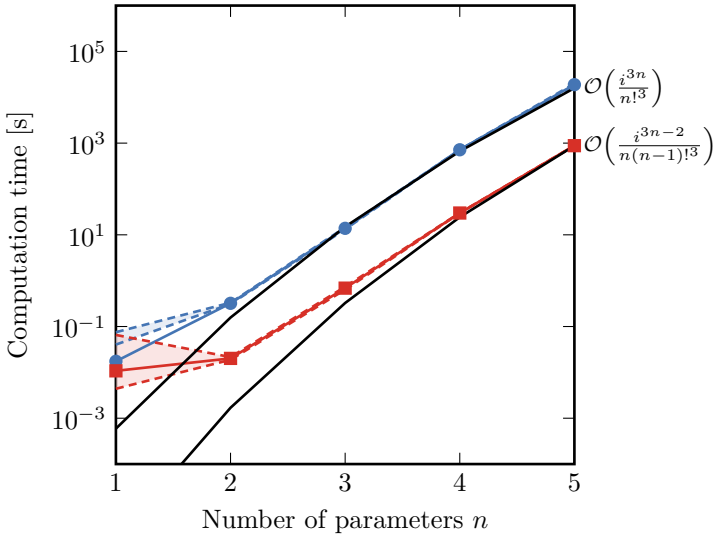


Figure 5.5♣. Experimental complexity analysis of the number of variables n on the null space computation for the block Macaulay matrix. The mean total computation time for the standard (—●—) and recursive (—■—) algorithm to compute a numerical basis matrix of the null space of a block Macaulay matrix $\boldsymbol{M}_{15}$ is averaged over 30 experiments (the dashed lines indicate one standard deviation). $\boldsymbol{M}_i$ is generated by a linear n-parameter eigenvalue problem with $n+1$ random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{(4+n)\times 5}$. The computation times of both algorithms follow the theoretical complexities (——) in Table 5.1.

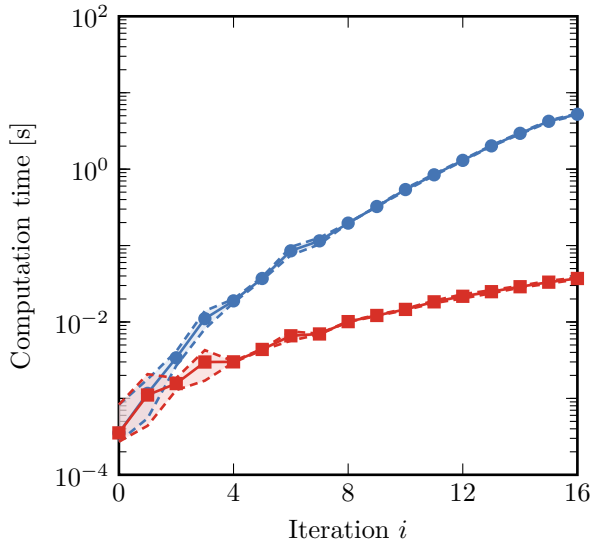
Example 5.4. We iteratively build a block Macaulay matrix $\boldsymbol{M}_i$ and compute a numerical basis matrix $\boldsymbol{Z}_i$ of its null space. We consider a linear 2-parameter eigenvalue problem with 3 random full rank seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{21\times 20}$. In Figure 5.6, we visualize the computation time and relative reconstruction error[6].

We observe experimentally that we gain two orders of magnitude in the computational complexity, as Table 5.1 indicates, while the relative reconstruction error[6] remains more or less the same, close to machine precision.

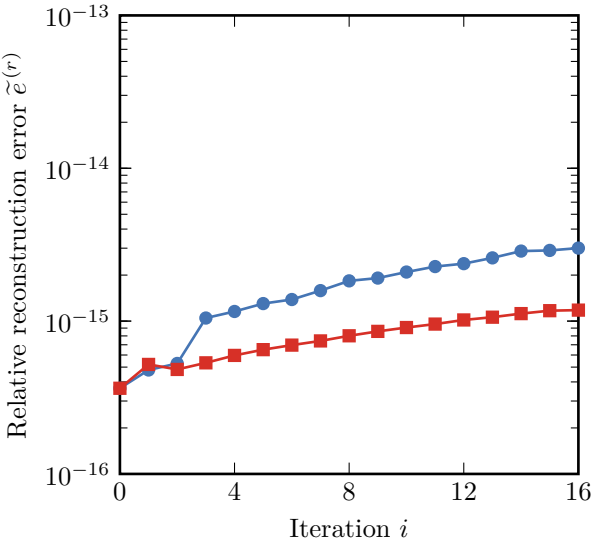
Next, we take a closer look at Remark 5.3 and look at the influence of an iteration-wise implementation versus a shift-wise implementation of the recursive algorithm.

Example 5.5. Figure 5.7 contains a comparison the computation time for the recursive algorithm, when applied iteration-wise and shift-wise. The results obtained for the block Macaulay matrices generated by a quadratic

[6] We calculate the relative reconstruction error for the computation of a basis matrix of the null space as $\widetilde{e}^{(r)} = \|\boldsymbol{M}_i\boldsymbol{Z}_i\|/\|\boldsymbol{M}_i\|$. More information about the error measures used in this text can be found in Appendix B.2.3.



(a) Computation time



(b) Relative reconstruction error

Figure 5.6♣. Comparison of the mean computation time and the mean relative reconstruction error $\widetilde{e}^{(r)}$ between the standard (—●—) and recursive (—■—) algorithm applied to a block Macaulay matrix $\boldsymbol{M}_i$, averaged over 30 experiments (the dashed lines indicate one standard deviation). The block Macaulay matrix $\boldsymbol{M}_i$ is generated by a linear 2-parameter eigenvalue problem with 3 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{21\times 20}$.

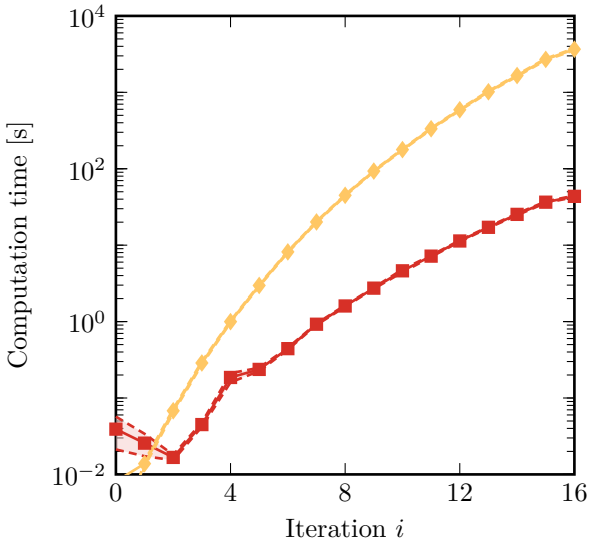


Figure 5.7♣. Comparison of the mean computation time of a block Macaulay matrix, averaged over 30 experiments (the dashed lines indicate one standard deviation), when we apply the recursive algorithm iteration-wise (—■—) and shift-wise (—✦—). The block Macaulay matrix $\boldsymbol{M}_i$ is generated by a quadratic 3-parameter eigenvalue problem with 10 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{22\times 20}$.

3-parameter eigenvalue problem support our claim that an iteration-wise implementation of the recursive algorithm is faster than a shift-wise approach, especially when the desired iteration $i_\circ$ is large. Note that there is no considerable difference in relative reconstruction error[6] between the shift-wise and iteration-wise implementation.

5.2.3.2 Random block banded Toeplitz matrix

To illustrate Remark 5.4, we consider the following example.

Example 5.6. Consider the block banded Toeplitz matrix $\boldsymbol{T}_i$ that consists of two square seed matrices $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathbb{R}^{20\times 20}$ with $\operatorname{rank}\big(\begin{bmatrix}\boldsymbol{A}_1 & \boldsymbol{A}_2\end{bmatrix}\big) = \operatorname{rank}\big(\begin{bmatrix}\boldsymbol{X} & \boldsymbol{Y}\end{bmatrix}\big) = 16$ (which is close to the number of columns $l = 20$). In every iteration i, we compute a numerical basis matrix of the null space of $\boldsymbol{T}_i$ via the standard and recursive algorithm. Clearly, the recursive approach outperforms the full singular value decomposition, as shown in Figure 5.8. Note that the relative reconstruction error[6] of both approaches is more or less the same.

The computation times of the standard and recursive algorithm for a block banded Toeplitz matrix grow cubicly and linearly with respect to i, respectively (as in Table 5.1 for $n = 1$).

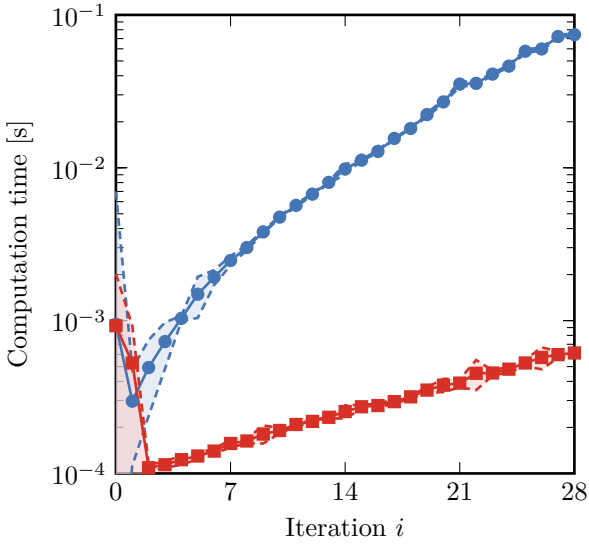


Figure 5.8♣. Comparison of the mean computation time between the standard (—●—) and recursive (—■—) algorithm applied to a block Toeplitz matrix $\boldsymbol{T}_i$, averaged over 30 experiments (the dashed lines indicate one standard deviation). $\boldsymbol{T}_i$ consists of two square random seed matrices $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathbb{R}^{20\times 20}$, such that the rank r of $\begin{bmatrix}\boldsymbol{A}_1 & \boldsymbol{A}_2\end{bmatrix} = \begin{bmatrix}\boldsymbol{X} & \boldsymbol{Y}\end{bmatrix}$ is equal to 16.

5.3 Sparse null space computations

Although an efficient implementation of Algorithm 5.2 may exploit the structure and sparsity pattern of the (block) Macaulay matrix, it still has two defects: (i) it stores the entire matrix in the memory and (ii) it does not consider the fact that the different shifts use the same coefficients or coefficient matrices. This means that we keep a large, sparse (block) Macaulay matrix in memory with a lot of redundancy, so avoiding the explicit construction and exploiting the repetition of the matrix is very interesting. In this section, we adapt the recursive null space computation and create sparse alternatives that do not require to store or construct the (block) Macaulay matrix during execution. We replace the iterative null space updating problem in Algorithm 5.1 by Algorithm 5.3.

5.3.1 Macaulay matrix algorithm

To address both above-mentioned defects in the case of the Macaulay matrix, we propose Algorithm 5.4, which is the sparse adaptation of [22, Algorithm 3.1]. It considers in every iteration the different shifts σ_k for every polynomial $p_k(\boldsymbol{x})$, for $k = 1, \ldots, s$, and divides the coefficients of the polynomials into two groups: the first group (COL_x) belongs to $\boldsymbol{X}_i$ and is multiplied efficiently with the previous basis matrix of the null space $\boldsymbol{Z}_{i-1}$, the second group (COL_y) constructs

Algorithm 5.3 Sparse updating of the (block) Macaulay matrix null space

Require: $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ or $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$

$i \leftarrow 0$
Construct the (block) Macaulay matrix $\boldsymbol{M}_0$
while $i < i_\circ$ **do**
    $\boldsymbol{Z}_i \leftarrow$ UPDATE-NULL$(\boldsymbol{Z}_{i-1})^\dagger$, via Algorithm 5.4 or Algorithm 5.5
    if structure of the null space contains the solutions **then**
        $i = i_\circ$
    else
        $i \leftarrow i + 1$
    end if
end while
return $\boldsymbol{Z}_{i_\circ}$

† For $i = 0$, this line becomes $\boldsymbol{Z}_0 \leftarrow$ NULL$(\boldsymbol{M}_0)$, computed via the standard algorithm.

the matrix $\boldsymbol{Y}_i$. These different variables are indicated on the spy plot of a Macaulay matrix in Figure 5.9 (cf., Figure 5.2a). At no point in this sparse adaptation, the Macaulay matrix $\boldsymbol{M}_i$ is build or stored in memory, but only used implicitly through the position of its shifts. This adaptation leads to a considerable reduction in necessary memory, as illustrated in the next example.

Example 5.7. Consider a system of five polynomials, each of degree $d_i = 5$, with random real coefficients. We recursively build a Macaulay matrix $\boldsymbol{M}_i$ and compute a basis matrix of its null space via Algorithm 5.2 for every iteration. Similarly, we use Algorithm 5.4 to construct the same basis matrix without constructing $\boldsymbol{M}_i$. We do this experiment for iterations $i = 0, \ldots, 20$, and compare the necessary memory for every i. It is clear from Figure 5.10 that the sparse algorithm is much more memory efficient.

5.3.2 Block Macaulay matrix algorithm

In Algorithm 5.5, we propose a sparse adaptation that addresses the same two shortcomings in the case of the block Macaulay matrix. It removes the explicit construction of the block Macaulay matrix $\boldsymbol{M}_i$ and incorporates the formation of $\boldsymbol{X}_i$ and $\boldsymbol{Y}_i$ into the recursive algorithm to build a basis matrix $\boldsymbol{Z}_i$ of the null space. For every shift σ in iteration i, Algorithm 5.5 first determines the position of the shifted seed matrices $\boldsymbol{A}_j$ and partitions them into $\boldsymbol{X}_i^\sigma$ and $\boldsymbol{Y}_i^\sigma$ (cf., Figure 5.9). The blocks $\boldsymbol{X}_i^\sigma$ yield together with the previous numerical basis matrix $\boldsymbol{Z}_{i-1}$ the matrix $\boldsymbol{W}_i^\sigma$, similar to Algorithm 5.2, but now per shift, and the blocks $\boldsymbol{Y}_i^\sigma$ result in $\boldsymbol{Y}_i$. The computation of $\boldsymbol{V}_i$ and $\boldsymbol{Z}_i$ are similar to Algorithm 5.4. At no point in this sparse algorithm, $\boldsymbol{M}_i$ is explicitly built or stored in memory, but is only used implicitly through the position of its shifts. We repeat the numerical example of Example 5.4 to show the memory improvements obtained by this sparse adaptation.

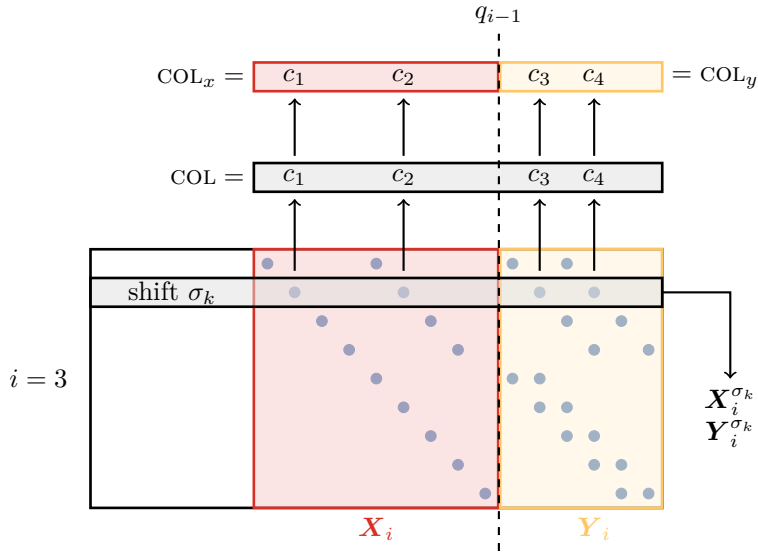


Figure 5.9. Annotated last iteration of the Macaulay matrix in Figure 5.2a. For iteration $i = 3$, one of the shifts, σ_k, is highlighted together with the allocation of the coefficients into COL_x and COL_y. Note that for the block Macaulay matrix, the situation is very similar: only the index k disappears and columns of coefficient matrices are selected instead of single coefficients.

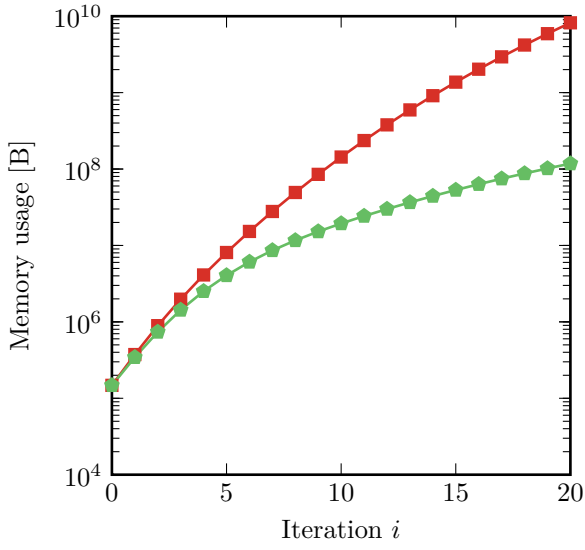


Figure 5.10♣. Comparison of the memory usage between the recursive (—■—) and sparse (—●—) algorithm applied to a Macaulay matrix $\boldsymbol{M}_i$. $\boldsymbol{M}_i$ is generated by a system of 5 multivariate polynomial equations, each of degree $d_i = 5$, with random coefficients.

Algorithm 5.4 Sparse null space algorithm for the Macaulay matrix

Require: $\boldsymbol{Z}_{i-1}$, $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$
1: **for** `every polynomial` $p_k(\boldsymbol{x})$ `with coefficients` c_j **do**
2: **for** `every shift` σ_k `of iteration` i $(\sigma_k = 1, \ldots, \sigma_{k\max})$ **do**
3: $\text{COL} \leftarrow$ positions of columns of c_j at shift σ_k
4: $\text{COL}_x \leftarrow \text{COL} \leq q_{i-1}$ $(c_1, \ldots c_x$ at shift $\sigma_k)$
5: $\text{COL}_y \leftarrow \text{COL} > q_{i-1}$ $(c_{x+1}, \ldots c_{x+y}$ at shift $\sigma_k)$
6: $\boldsymbol{W}_i^{\sigma_k} \leftarrow \begin{bmatrix} c_1 & \cdots & c_x \end{bmatrix} \boldsymbol{Z}_{i-1}(\text{COL}_x)$
7: $\boldsymbol{Y}_i^{\sigma_k}(\text{COL}_y) \leftarrow \begin{bmatrix} c_{x+1} & \cdots & c_{x+y} \end{bmatrix}$
8: **end for**
9: $\boldsymbol{W}_i^{p_k(\boldsymbol{x})} \leftarrow \begin{bmatrix} \boldsymbol{W}_i^1 \\ \vdots \\ \boldsymbol{W}_i^{\sigma_{k\max}} \end{bmatrix}$ and $\boldsymbol{Y}_i^{p_k(\boldsymbol{x})} \leftarrow \begin{bmatrix} \boldsymbol{Y}_i^1 \\ \vdots \\ \boldsymbol{Y}_i^{\sigma_{k\max}} \end{bmatrix}$
10: **end for**
11: $\boldsymbol{W}_i \leftarrow \begin{bmatrix} \boldsymbol{W}_i^{p_1(\boldsymbol{x})} \\ \vdots \\ \boldsymbol{W}_i^{p_s(\boldsymbol{x})} \end{bmatrix}$ and $\boldsymbol{Y}_i \leftarrow \begin{bmatrix} \boldsymbol{Y}_i^{p_1(\boldsymbol{x})} \\ \vdots \\ \boldsymbol{Y}_i^{p_s(\boldsymbol{x})} \end{bmatrix}$
12: $\boldsymbol{V}_i \leftarrow \text{NULL}\left(\begin{bmatrix} \boldsymbol{W}_i & \boldsymbol{Y}_i \end{bmatrix}\right)$
13: $\boldsymbol{Z}_i \leftarrow \begin{bmatrix} \boldsymbol{Z}_{i-1}\boldsymbol{V}_i^1 \\ \boldsymbol{V}_i^2 \end{bmatrix}$
14: **return** $\boldsymbol{Z}_i$

Example 5.8. We repeat Example 5.4, but now compare the recursive and sparse algorithm. The sparse algorithm is much more memory efficient than its recursive counterpart (Figure 5.11).

Note that for some orderings of the monomials in the (block) Macaulay matrix, the structure can be exploited even further. For example, when using the GRINVLEX ordering, like in Figure 5.1, $\boldsymbol{Y}_i$ always contains $\boldsymbol{Y}_{i-1}$.

5.4 Recursive rank checks

The standard way to determine the rank structure of the null space of a (block) Macaulay matrix is by considering the constructed basis matrix $\boldsymbol{Z}_d$ of its null space and compute the rank or nullity for every submatrix that has one more degree block than the previous one, starting with the zeroth degree block (Figure 5.12). In that context, we can consider $\boldsymbol{Z}_d$ as a block row matrix $\boldsymbol{R}_i$ ($i = 0, \ldots, d$ with $\boldsymbol{R}_d = \boldsymbol{Z}_d$) and we use a recursive algorithm to compute a basis matrix of its null space, the intermediate results yielding the rank/nullity structure of $\boldsymbol{Z}_d$.

After i iterations, a block row matrix $\boldsymbol{R}_i \in \mathbb{C}^{p_i \times q_i}$ consists of $i + 1$ consec-

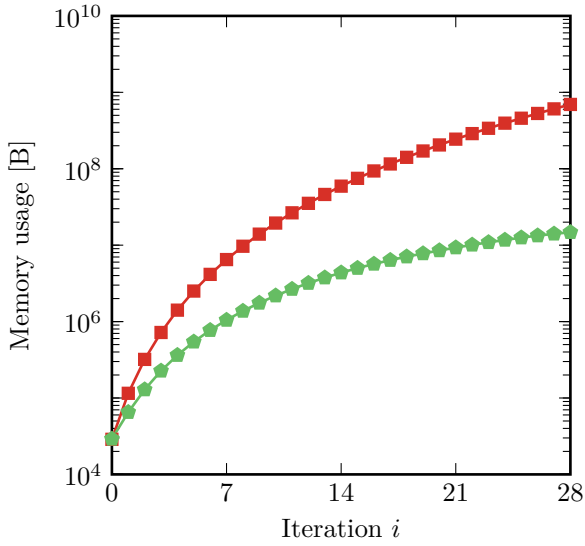


Figure 5.11♣. Comparison of the memory usage between the recursive (—■—) and sparse (—⬟—) algorithm applied to a block Macaulay matrix $\boldsymbol{M}_i$. $\boldsymbol{M}_i$ is generated by a linear 2-parameter eigenvalue problem with 3 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{21\times 20}$.

utive blocks[7] (or block rows) $\boldsymbol{B}_i \in \mathbb{C}^{k\times l}$:

$$\boldsymbol{R}_i = \begin{bmatrix} \boldsymbol{B}_0 \\ \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \\ \vdots \\ \boldsymbol{B}_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{i-1} \\ \boldsymbol{B}_i \end{bmatrix}. \tag{5.21}$$

The number of rows and columns of the block row matrix $\boldsymbol{R}_i$ is

$$p_i = k(i+1), \tag{5.22}$$
$$q_i = l, \tag{5.23}$$

respectively. Block row matrices appear in applications where the data only gradually becomes available (e.g., online signal processing problems) or where intermediate results are required (e.g., to determine the rank structure of the matrix). In the former situation, the desired iteration $i_\circ$ of the block row matrix is often not known in advance. Since $\boldsymbol{R}_i$ grows in every iteration i, its null space also changes with respect to i. We denote an orthogonal numerical basis matrix of the null space of $\boldsymbol{R}_i$ by $\boldsymbol{U}_i \in \mathbb{C}^{q_i\times n_i}$, such that

$$\boldsymbol{R}_i\boldsymbol{U}_i = \boldsymbol{0}, \tag{5.24}$$

[7]Although we consider in this chapter consecutive blocks $\boldsymbol{B}_i$ with an equal number of rows for didactical purposes, the extension to consecutive blocks with a different number of rows is trivial and does not alter the proposed algorithm (Example 5.14).

Algorithm 5.5 Sparse null space algorithm for the block Macaulay matrix

Require: $\boldsymbol{Z}_{i-1}$, $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{x+y}$

for `every shift` σ `of iteration` i `(`$\sigma = 1, \ldots \sigma_{\texttt{max}}$`)` **do**
  $\text{COL} \leftarrow$ positions of columns of $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_{x+y}$ at shift σ
  $\text{COL}_x \leftarrow \text{COL} \leq q_{i-1}$ ($\boldsymbol{A}_1, \ldots, \boldsymbol{A}_x$ at shift σ)
  $\text{COL}_y \leftarrow \text{COL} > q_{i-1}$ ($\boldsymbol{A}_{x+1}, \ldots, \boldsymbol{A}_{x+y}$ at shift σ)
  $\boldsymbol{W}_i^{\sigma} \leftarrow \begin{bmatrix}\boldsymbol{A}_1 & \cdots & \boldsymbol{A}_x\end{bmatrix} \boldsymbol{Z}_{i-1}(\text{COL}_x)$
  $\boldsymbol{Y}_i^{\sigma}(\text{COL}_y) \leftarrow \begin{bmatrix}\boldsymbol{A}_{x+1} & \cdots & \boldsymbol{A}_{x+y}\end{bmatrix}$
end for
$\boldsymbol{W}_i \leftarrow \begin{bmatrix}\boldsymbol{W}_i^1 \\ \vdots \\ \boldsymbol{W}_i^{\sigma_{\max}}\end{bmatrix}$ and $\boldsymbol{Y}_i \leftarrow \begin{bmatrix}\boldsymbol{Y}_i^1 \\ \vdots \\ \boldsymbol{Y}_i^{\sigma_{\max}}\end{bmatrix}$
$\boldsymbol{V}_i \leftarrow \text{NULL}\left(\begin{bmatrix}\boldsymbol{W}_i & \boldsymbol{Y}_i\end{bmatrix}\right)$
$\boldsymbol{Z}_i \leftarrow \begin{bmatrix}\boldsymbol{Z}_{i-1}\boldsymbol{V}_i^1 \\ \boldsymbol{V}_i^2\end{bmatrix}$
return $\boldsymbol{Z}_i$

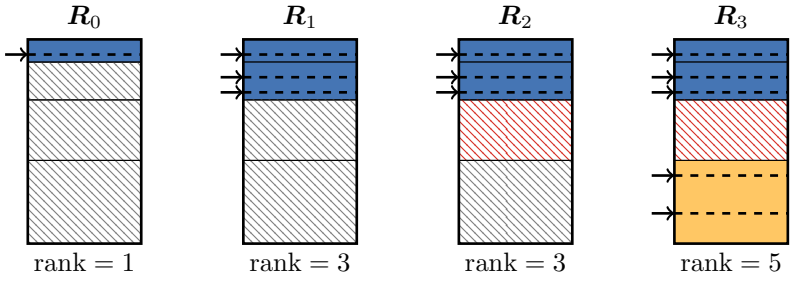


Figure 5.12. Graphical example of the iterative procedure to determine the rank structure of a block row matrix $\boldsymbol{R}_i$. A new seed matrix $\boldsymbol{B}_i$ is added and the rank of $\boldsymbol{R}_i$ is computed. In the case of basis matrix of the null space ($\boldsymbol{R}_d = \boldsymbol{Z}_d$), the different zones of the basis matrix, i.e., the regular zone (■), gap zone (▨), and singular zone (■), can be identified via these iterative rank checks. Dashed lines denote linearly independent rows.

Algorithm 5.6 Iterative updating of the block row matrix null space

Require: $\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots$
1: $\boldsymbol{U}_0 \leftarrow \text{NULL}(\boldsymbol{R}_0)$ with $\boldsymbol{R}_0 = \boldsymbol{B}_0$
2: $i \leftarrow 1$
3: **while** $i \leq i_\circ$ **do**
4: $\quad \boldsymbol{R}_i \leftarrow \begin{bmatrix} \boldsymbol{R}_{i-1} \\ \boldsymbol{B}_i \end{bmatrix}$
5: $\quad \boldsymbol{U}_i \leftarrow \text{NULL}(\boldsymbol{R}_i)$, for example, via Algorithm 5.7
6: $\quad i \leftarrow i + 1$
7: **end while**
8: **return** $\boldsymbol{U}_{i_\circ}$

where n_i corresponds to the nullity of $\boldsymbol{R}_i$. In Algorithm 5.6, the problem is stated more clearly: we extend the block row matrix $\boldsymbol{R}_i$ in an iterative way and compute a numerical basis matrix $\boldsymbol{U}_i$ of its null space in every iteration using $\boldsymbol{U}_{i-1}$, until we reach the desired iteration $i_\circ$.

The standard algorithm to determine this numerical basis matrix is the singular value decomposition and it does not consider the iterative nature of the problem. Therefore, we propose a recursive algorithm that uses the existing numerical basis matrix $\boldsymbol{U}_{i-1} \in \mathbb{C}^{q_{i-1} \times n_{i-1}}$ of the null space of the block row matrix $\boldsymbol{R}_{i-1} \in \mathbb{C}^{p_{i-1} \times q_{i-1}}$ to obtain $\boldsymbol{U}_i$ (Section 5.4.1). We do not assume any structure in the blocks $\boldsymbol{B}_i$ of $\boldsymbol{R}_i$, apart from the iterative construction in (5.21). Afterwards, we asses the computational complexity of this recursive algorithm and compare it with the standard approach (Section 5.4.2). Finally, numerical examples illustrate the theoretical derivations (Section 5.4.3).

5.4.1 Recursive algorithm

Consider a block row matrix $\boldsymbol{R}_{i-1} \in \mathbb{C}^{p_{i-1} \times q_{i-1}}$ after $i-1$ iterations and an orthogonal numerical basis matrix $\boldsymbol{U}_{i-1} \in \mathbb{C}^{q_{i-1} \times n_{i-1}}$ of its null space:

$$\boldsymbol{R}_{i-1}\boldsymbol{U}_{i-1} = \boldsymbol{0}. \tag{5.25}$$

When we append a new block $\boldsymbol{B}_i$ to obtain $\boldsymbol{R}_i$, we know that there exists an orthogonal matrix $\boldsymbol{V}_i \in \mathbb{C}^{n_{i-1} \times n_i}$, so that

$$\underbrace{\begin{bmatrix} \boldsymbol{R}_{i-1} \\ \boldsymbol{B}_i \end{bmatrix}}_{\boldsymbol{R}_i} \boldsymbol{U}_{i-1}\boldsymbol{V}_i = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{B}_i\boldsymbol{U}_{i-1} \end{bmatrix} \boldsymbol{V}_i = \boldsymbol{0}, \tag{5.26}$$

because of (5.25). The matrix $\boldsymbol{V}_i$, on the one hand, is a basis matrix of the null space of the matrix $\boldsymbol{W}_i = \boldsymbol{B}_i\boldsymbol{U}_{i-1} \in \mathbb{C}^{k \times n_{i-1}}$. The nullity n_i of $\boldsymbol{R}_i$ is at most n_{i-1} because the block $\boldsymbol{B}_i$ adds (sometimes zero) linearly independent rows to $\boldsymbol{R}_i$. The matrix product $\boldsymbol{U}_i = \boldsymbol{U}_{i-1}\boldsymbol{V}_i = \prod_{j=0}^{i} \boldsymbol{V}_j \in \mathbb{C}^{l \times n_i}$ (with $\boldsymbol{V}_0 = \boldsymbol{U}_0$), on the other hand, is a numerical basis matrix of the null space of $\boldsymbol{R}_i$. This insight yields a recursive algorithm to update an orthogonal numerical basis matrix of the null space of the block row matrix, which can also be used to

Algorithm 5.7 Recursive null space algorithm for the block row matrix

Require: $\boldsymbol{U}_{i-1}$ and $\boldsymbol{B}_i$
1: $\boldsymbol{W}_i \leftarrow \boldsymbol{B}_i\boldsymbol{U}_{i-1}$
2: $\boldsymbol{V}_i \leftarrow \text{NULL}(\boldsymbol{W}_i)$
3: $\boldsymbol{U}_i \leftarrow \boldsymbol{U}_{i-1}\boldsymbol{V}_i$
4: Return $\boldsymbol{U}_i$

track the rank/nullity of that block row matrix through different iterations. We summarize the different steps to obtain $\boldsymbol{U}_i$ given $\boldsymbol{B}_i$ in Algorithm 5.7, which fits perfectly in Algorithm 5.6.

Remark 5.5. In Algorithm 5.7, a correct rank decision is essential to obtain accurate results. For example, in the (limit) case when we add a new block $\boldsymbol{B}_i$ of which all the rows depend linearly on the rows of the previous blocks $(\boldsymbol{B}_0, \ldots, \boldsymbol{B}_{i-1})$, the numerical basis matrix of the null space of $\boldsymbol{R}_{i-1}$ also annihilates the matrix $\boldsymbol{B}_i$. Hence, $\boldsymbol{W}_i = \boldsymbol{B}_i\boldsymbol{U}_{i-1}$ (theoretically) equals zero. When we determine $\boldsymbol{V}_i$ in Algorithm 5.7 (line 2), we should obtain an orthogonal matrix of full rank n_{i-1}, e.g., an identity matrix. However, due to numerical floating-point errors, the matrix $\boldsymbol{W}_i$ is only close to zero and we need to be very careful when computing $\boldsymbol{V}_i$. A careful rank check in Algorithm 5.7 alleviates this problem in most situations, for example by using an additional absolute tolerance or a more advanced rank decision approach [143, 195].

Example 5.9. Consider a particular rank-10 block row matrix $\boldsymbol{R}_1 \in \mathbb{R}^{40\times 20}$, which consists of two rank-5 blocks $\boldsymbol{B}_0 \in \mathbb{R}^{20\times 20}$ and $\boldsymbol{B}_1 \in \mathbb{R}^{20\times 20}$, and a orthogonal basis matrix of its null space $\boldsymbol{U}_1 \in \mathbb{C}^{20\times 10}$. We create a new block $\boldsymbol{B}_2 = 2\boldsymbol{B}_0 + 3\boldsymbol{B}_1 \in \mathbb{R}^{20\times 20}$ and construct $\boldsymbol{R}_2 \in \mathbb{R}^{60\times 20}$ as

$$\boldsymbol{R}_2 = \begin{bmatrix} \boldsymbol{B}_0 \\ \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{B}_2 \end{bmatrix}. \tag{5.27}$$

Since the rows of $\boldsymbol{B}_2$ depend linearly on the rows of the first two blocks by construction, the matrix $\boldsymbol{W}_2 = \boldsymbol{B}_2\boldsymbol{U}_1$ is close (but not exactly equal) to zero. All singular values have the same order of magnitude and, when using a relative tolerance, the matrix $\boldsymbol{W}_2$ could be considered to be of full rank.

Code 5.1. The illustration of the importance of correct rank decisions in Example 5.9 can be implemented as follows:

```
>> B0 = randn(20,5)*randn(5,20);
>> B1 = randn(20,5)*randn(5,20);
>> B2 = 2*B0 + 3*B1;
>> R1 = [B0; B1]; U1 = null(R1);
>> W2 = B2*U1;
```

While $\boldsymbol{W}_2$ is numerically zero, its rank is equal to 10.

```
>> norm(W2)

ans =
    2.8321e-14

>> rank(W2)

ans =
    10
```

5.4.2 Computational complexity

The computational complexity for both the standard and recursive approach is analyzed in this section; firstly, we derive theoretical expressions (Section 5.4.2.1) and, afterwards, we verify the results experimentally (Section 5.4.2.2).

5.4.2.1 Theoretical complexity analysis

When computing a numerical basis matrix $\boldsymbol{U}_i$ of the null space of the block row matrix $\boldsymbol{R}_i$ via the **standard algorithm** (i.e., the singular value decomposition), we only use the singular values and right singular vectors. We substitute the number of rows (5.22) and columns (5.23) of the block row matrix $\boldsymbol{R}_i$, in iteration i, into the computational cost of computing these singular values and right singular vectors, which is about $4p_iq_i^2 + 8q_i^3$ FLOPs [97, p. 493]. This yields the computational complexity of the standard algorithm (in FLOPs):

$$4kl^2(i+1) + 8l^3 = 4kl^2i + 4kl^2 + 8l^3 = \mathcal{O}(i). \tag{5.28}$$

In some applications, the blocks $\boldsymbol{B}_i$ are square (i.e., $k = l$), which simplifies (5.28):

$$4l^3i + 12l^3 = \mathcal{O}(i). \tag{5.29}$$

The proposed **recursive algorithm** consists of three main steps (Algorithm 5.7), each with their respective number of FLOPs:

$$\begin{aligned} 2kln_{i-1} &\quad \text{(multiplication – line 1)}, \\ 4kn_{i-1}^2 + 8n_{i-1}^3 &\quad \text{(null space computation – line 2)}, \\ 2ln_{i-1}n_i &\quad \text{(multiplication – line 3)}. \end{aligned}$$

Table 5.2. Computational complexity (given in FLOPs per iteration i) of the standard and recursive algorithm to determine a numerical basis matrix of the null space of the block row matrix $\boldsymbol{R}_i$, for both rectangular $k \times l$ and square $l \times l$ blocks $\boldsymbol{B}_i$. The given computational complexity of the recursive algorithm is an upper bound and depends in practice on the rank of the blocks $\boldsymbol{B}_i$ $(i = 0, \dots, i)$.

algorithm	**rectangular**	**square**
standard	$4kl^2i + 4kl^2 + 8l^3$	$4l^3i + 12l^3$
recursive	$6kl^2 + 10l^3$	$16l^3$

The nullity n_i of $\boldsymbol{R}_i$ is equal to $l - r_i \leq l = \mathcal{O}(1)$, where r_i is the rank of $\boldsymbol{R}_i$. The total computational complexity of the recursive algorithm is thus bounded above by (in FLOPs)

$$6kl^2 + 10l^3 = \mathcal{O}(1), \tag{5.30}$$

or when the blocks $\boldsymbol{B}_i$ are square (i.e., $k = l$) by

$$16l^3 = \mathcal{O}(1). \tag{5.31}$$

When we compare the (theoretical) computational complexity of both approaches (Table 5.2), we notice that the number of FLOPs of the recursive algorithm remains constant with respect to the iteration i, while the computational complexity of the standard algorithm depends linearly on i. This behavior, of course, does not sound surprising, as the recursive algorithm uses results from the previous iterations and matrices of (more or less) fixed sizes, while the block row matrix $\boldsymbol{R}_i$ in the standard approach grows in every iteration.

5.4.2.2 Experimental complexity analysis

Firstly, we investigate the computational complexity to compute a basis matrix of the null space with respect to iteration i.

Example 5.10. Consider a block row matrix $\boldsymbol{R}_i \in \mathbb{R}^{(300i)\times 300}$ constructed from random seed matrices $\boldsymbol{B}_i \in \mathbb{R}^{300\times 300}$ of rank $r = 3$. We determine the computation time to compute a basis matrix of the null space of this block row matrix for different i. We visualize in Figure 5.13 the mean computation time for both the standard and recursive algorithm. The recursive algorithm remains under the constant line, while the computation time of the standard algorithm grows linearly, as expected.

Secondly, we investigate the influence of the size of the seed matrices $\boldsymbol{B}_i$ on the computation time.

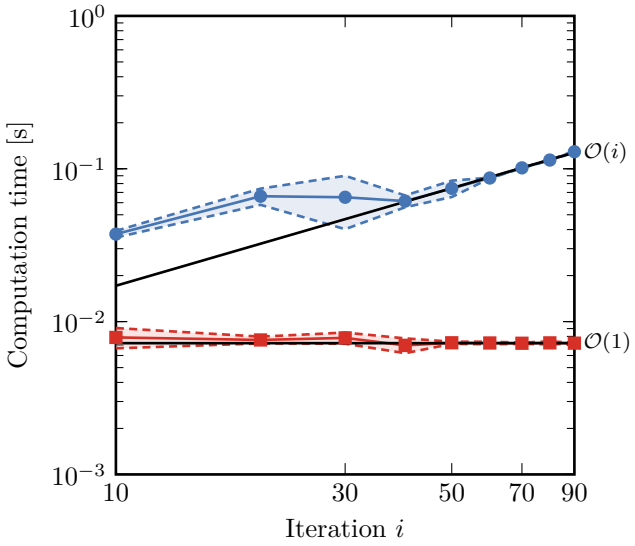


Figure 5.13♣. Experimental complexity analysis of the iteration on the null space computation for the block row matrix. The mean computation time for the standard (—●—) and recursive (—■—) algorithm to compute a numerical basis matrix of the null space of a block row matrix $\boldsymbol{R}_i$ is averaged over 30 experiments (the dashed lines indicate one standard deviation). In every iteration i, we extend the block row matrix $\boldsymbol{R}_{i-1}$ with a random block $\boldsymbol{B}_i \in \mathbb{R}^{300\times 300}$ of rank $r = 3$. The computation time of the standard approach grows linearly $\mathcal{O}(i)$, while the computation time of the recursive algorithms remains constant $\mathcal{O}(1)$, as expected from the theoretical complexities (——).

Example 5.11. We visualize in Figure 5.14 the total time to compute a numerical basis matrix of the null space of a block row matrix $\boldsymbol{R}_{15}$ from $i = 0$ to the desired iteration $i_\circ = 15$, i.e., the total computation time to iteratively reach $i_\circ$. We consider in this numerical example a block row matrix $\boldsymbol{R}_i$ that we extend in every iteration with a seed matrix $\boldsymbol{B}_i \in \mathbb{R}^{l\times l}$ for different values of l. The computation time of both algorithms in Figure 5.14 grows cubicly with respect to l, as expected.

5.4.3 Some numerical examples

We consider three additional experiments to illustrate the numerical properties of the recursive algorithm: a random block row matrix with increasing rank (or decreasing nullity), a random block row matrix of which the rank (and also the nullity) stabilizes after $i = 10$ iterations, and a block row matrix with seed matrices of varying size.

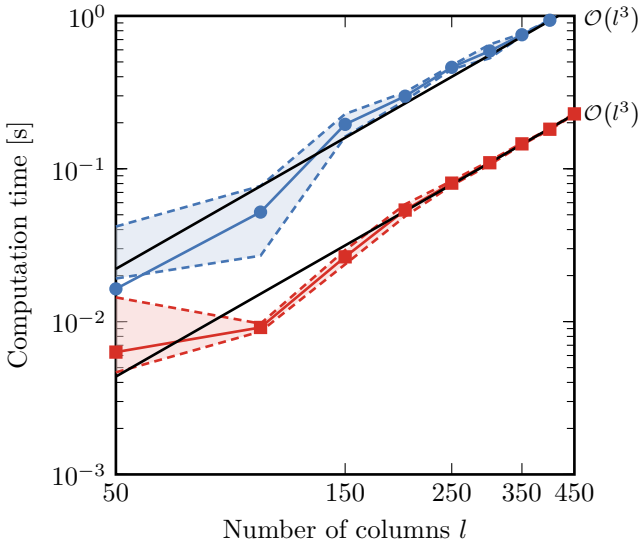


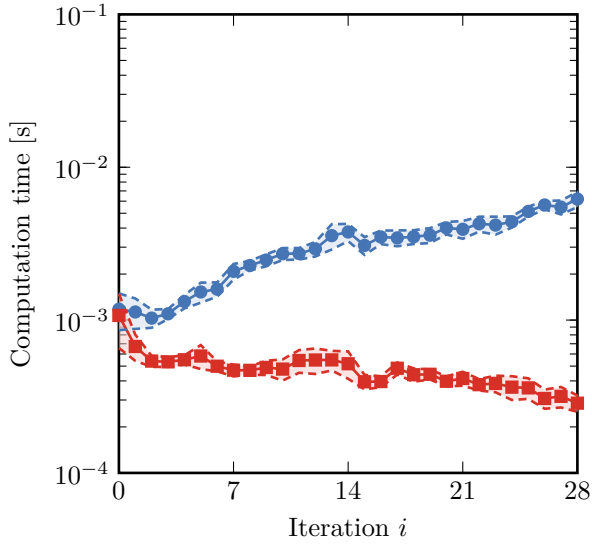
Figure 5.14♣. Experimental complexity analysis of the seed matrix size on the null space computation for the block row matrix. The mean computation time for the standard (—●—) and recursive (—■—) algorithm to compute a numerical basis matrix of the null space of a block row matrix $\boldsymbol{R}_{15}$ is averaged over 30 experiments (the dashed lines indicate one standard deviation). In every iteration i, we extend the block row matrix $\boldsymbol{R}_{i-1}$ with a random block $\boldsymbol{B}_i \in \mathbb{R}^{l\times l}$ of rank $r = 2$, until iteration $i_\circ = 15$. The computation times of both algorithms grow cubicly with respect to l, as expected from the theoretical complexities (——), which is $\mathcal{O}(l^3)$ for the both algorithms.

5.4.3.1 Random block row matrix with increasing rank

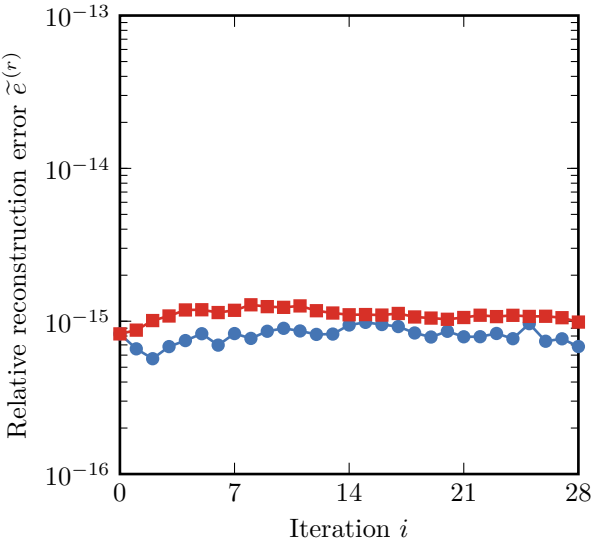
The first numerical example consists of a block row matrix with random seed matrices that increase the total rank in every iteration.

Example 5.12. Consider a block row matrix $\boldsymbol{R}_i \in \mathbb{R}^{(300i)\times 300}$, which we extend in every iteration i by a random matrix $\boldsymbol{B}_i \in \mathbb{R}^{300\times 300}$ with rank $r = 3$. The rank of $\boldsymbol{R}_i$ is equal to $r_i = \max(3(i+1), 300)$. The recursive algorithm clearly outperforms the standard algorithm (Figure 5.15), while the relative reconstruction errors[6] remain stable within the same order of magnitude.

As mentioned in Section 5.4.2, the computation time of the standard algorithm grows linearly with the respect to i, while the computation time of the recursive algorithm remains more or less constant. Figure 5.15 even shows a small decrease in the computation time for higher iterations, which is mainly because of the decrease in the nullity (remember that we used the upper bound of the nullity to determine the computational complexity of the recursive algorithm, which is especially a good approximation when the number of blocks is small).



(a) Computation time



(b) Relative reconstruction error

Figure 5.15♣. Comparison of the mean computation time and the mean relative reconstruction error $\widetilde{e}^{(r)}$ between the standard (—●—) and recursive (—■—) algorithm applied to a block row matrix $\boldsymbol{R}_i$, averaged over 30 experiments (the dashed lines indicate one standard deviation). In every iteration i, we extend the block row matrix $\boldsymbol{R}_{i-1}$ with a random block $\boldsymbol{B}_i \in \mathbb{R}^{100\times 100}$ of rank $r = 2$. The computation time of the recursive algorithm decreases for higher iterations, because the input matrices become smaller in every iteration (since the nullity decreases in every iteration).

5.4.3.2 Random block row matrix with stabilizing rank

In the second numerical example, we look at a block row matrix with random seed matrices where the total rank stabilizes after a certain number of iterations.

Example 5.13. Consider the block row matrix $\boldsymbol{R}_i \in \mathbb{R}^{(100i)\times 100}$ in which the new blocks $\boldsymbol{B}_i$ after $i = 10$ iterations are linear combinations of the previously appended blocks (viz., $\boldsymbol{B}_0, \ldots, \boldsymbol{B}_{10}$). The rank and nullity of $\boldsymbol{R}_i$ stabilize after $i = 10$ iterations, and we notice that the computation time of the recursive algorithm (Figure 5.16) becomes constant, i.e., the computational complexity now follows the theoretical $\mathcal{O}(1)$. Notice that the relative reconstruction errors[6] remain stable within the same order of magnitude.

Notice that the computation time first jumps at $i = 11$ before stabilizing. Due to the rank stabilization after 10 iterations, the matrix $\boldsymbol{W}_{11}$ is numerically zero and considered as a matrix of full rank, the singular value decomposition of which is computationally more expensive than of a low-rank matrix (like $\boldsymbol{W}_{10}$). This is completely in line with our earlier discussion about the importance of a correct rank decision (Remark 5.5): when we are not careful and use wrong rank decisions, the relative error of the recursive algorithm can rise quickly. The combination of a relative and absolute tolerance avoids wrong rank decisions in this numerical example.

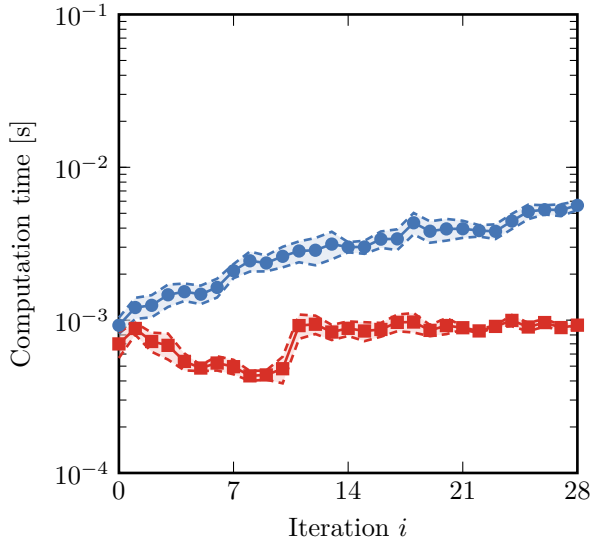
5.4.3.3 Seed matrices with different number of rows

As mentioned in Footnote 7, the recursive algorithm does not explicitly require that the number of rows of the seed matrices is equal. As an illustration of a problem with different number of rows in every seed matrix $\boldsymbol{B}_i$, we consider a basis matrix of the Macaulay matrix and use the recursive algorithm to recover its rank structure.

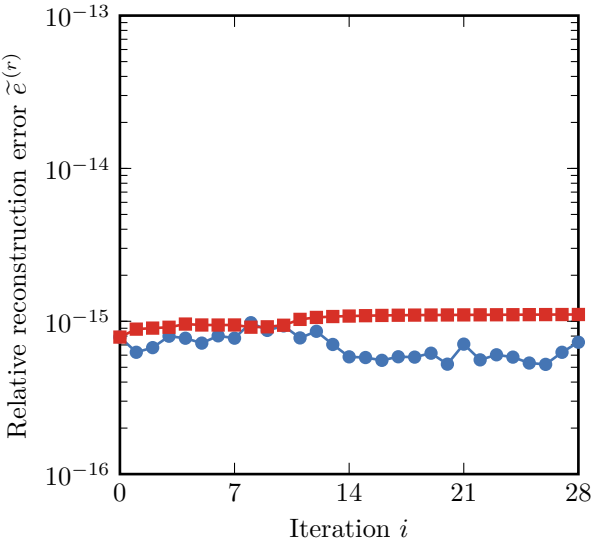
Example 5.14. The following system of 5-variate polynomial equations (with maximum total degree $d_{\max} = 3$) models a neural network by an adaptive Lotka–Volterra system [266]:

$$\begin{cases} p_1(\boldsymbol{x}) = x_1x_2^2 + x_1x_3^2 + x_1x_4^2 + x_1x_5^2 - 1.1x_1 + 1 = 0, \\ p_2(\boldsymbol{x}) = x_2x_1^2 + x_2x_3^2 + x_2x_4^2 + x_2x_5^2 - 1.1x_2 + 1 = 0, \\ p_3(\boldsymbol{x}) = x_3x_1^2 + x_3x_2^2 + x_3x_4^2 + x_3x_5^2 - 1.1x_3 + 1 = 0, \\ p_4(\boldsymbol{x}) = x_4x_1^2 + x_4x_2^2 + x_4x_3^2 + x_4x_5^2 - 1.1x_4 + 1 = 0, \\ p_5(\boldsymbol{x}) = x_5x_1^2 + x_5x_2^2 + x_5x_3^2 + x_5x_4^2 - 1.1x_5 + 1 = 0. \end{cases} \tag{5.32}$$

This system has an isolated zero-dimensional solution set with 233 affine solutions and 10 solutions at infinity. The solution degree of the Macaulay matrix is $d_\circ = 11$, for which we observe a gap in its null space. In order to check this, we consider a basis matrix $\boldsymbol{Z}_d$ of its null space and use the



(a) Computation time



(b) Relative reconstruction error

Figure 5.16♣. Comparison of the mean computation time and the mean relative reconstruction error $\widetilde{e}^{(r)}$ between the standard (—●—) and recursive (—■—) algorithm applied to a block row matrix $\boldsymbol{R}_i$, averaged over 30 experiments (the dashed lines indicate one standard deviation). In every iteration i, we extend the block row matrix $\boldsymbol{R}_{i-1}$ with a random block $\boldsymbol{B}_i \in \mathbb{R}^{100\times 100}$ of rank $r = 2$, until iteration $i = 10$. After 10 iterations, the newly appended blocks are linear combinations of previously added blocks, hence the computational complexity of the recursive algorithm stabilizes. The computation time for the recursive algorithm jumps at $i = 11$ because the matrix $\boldsymbol{W}_{11}$ is numerically zero.

recursive algorithm on the consecutive degree blocks. The basis matrix corresponds with a 4368×243 block row matrix $\boldsymbol{R}_{11}$ that consists of 11 seed matrices $\boldsymbol{B}_i$. The number of rows of $\boldsymbol{B}_i$ and the obtained rank structure can be found in Table 5.3. From the rank checks, it is clear that the 9th degree block corresponds to the gap zone. The standard algorithm executes the rank checks in 0.2343 s, while the recursive algorithms retrieves the correct rank structure in 0.0356 s.

Code 5.2. If we want to determine the rank structure of a basis matrix of the null space recursively, then we need to check the nullity for growing submatrices, adding one degree block in every iteration. The system `noon5` is part of MacaulayLab's database.

```
>> M = macaulay(noon5,11);
>> Z = null(M);
```

We can use the recursive algorithm to update a basis matrix $\boldsymbol{U}$ of its null space, when considering $\boldsymbol{Z}$ as a block row matrix. The function `nullrecrrow(U,B)` builds a numerical basis of the null space of a block row matrix, using a numerical basis `U` of the null space of a lower degree block row matrix and the new block `B`.

```
>> U = null(Z(1,:));
>> nullity = zeros(11,1);
>> for i = 1:11
       B = Z(nbmonomials(i-1,5)+1:nbmonomials(i,5),:)
       U = nullrecrrow(U,B);
       nullity(i) = size(U,2);
   end
```

This numerical example shows that the recursive algorithm is useful in the polynomial system solving context, when checking the rank structure of the null space of the (scalar) Macaulay matrix. Of course, the block row matrix of Example 5.14 could easily be replaced with a basis matrix of the null space of the block Macaulay matrix. This example gives rise to the idea of the next section: double recursive algorithms to solve systems of multivariate polynomial equations and MEPs.

5.5 Double recursive algorithms

By recognizing that the basis matrix of the null space of the (block) Macaulay matrix is a block row matrix, we can combine the recursive/sparse null space

Table 5.3. Summary of the linearly independent rows of the basis of the null space of the Macaulay matrix that comprises the system in Example 5.14.

degree block(s)	rows	lin. indep. rows	increase
0	1	1	1
$0-1$	$1-6$	6	5
$0-2$	$1-21$	21	30
$0-3$	$1-56$	51	30
$0-4$	$1-126$	96	45
$0-5$	$1-252$	147	51
$0-6$	$1-462$	192	45
$0-7$	$1-792$	222	30
$0-8$	$1-1287$	233	11
$0-9$	$1-2002$	233	0
$0-10$	$1-3003$	238	5
$0-11$	$1-4368$	243	5

computations with the recursive rank checks into double recursive algorithms: we use (i) a recursive/sparse technique to construct a numerical basis matrix of the null space of the (block) Macaulay matrix (Algorithms 5.2, 5.4 and 5.5) and we apply (ii) a recursive technique for the block row matrix to check the rank structure of that basis matrix (Algorithm 5.7). In particular, the sparse adaptation is very useful, because it avoids the explicit construction of a large (block) Macaulay matrix. We demonstrate the improvements in both computation time and memory usage of the double recursive algorithms via several numerical examples. We compare five different combinations: standard-standard, standard-recursive, recursive-standard, recursive-recursive, and sparse-recursive.

5.5.1 Polynomial system solving

We start with two systems of multivariate polynomial equations with random coefficients, re-consider the Katsura system from Example 2.20, the Lotka–Volterra system from Example 5.14, and tackle a reduced eight-dimensional economical problem.

5.5.1.1 Systems of random multivariate polynomials

We start with a dense system of random multivariate polynomial equations, meaning that all coefficients of the polynomials are random nonzero real numbers.

Example 5.15. The first system consists of three dense polynomials, each of total degree $d_i = 15$, with random real coefficients. The different Macaulay matrix algorithms require all $i_\circ = 28$ iterations to obtain the common roots of the system of multivariate polynomials. Table 5.4 contains the necessary computation time, memory usage, and maximum absolute residual error[8]for

Table 5.4♣. Obtained results of solving the dense system of three multivariate polynomials in Example 5.15 ($d_i = 15$) via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 28$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual error[8] of the solutions are averaged over 30 experiments.

combination	time	memory	$\max\|e\|_2$
standard-standard	1081.67 s	2.05 GB	2.09×10^{-12}
standard-recursive	922.35 s	2.14 GB	2.39×10^{-12}
recursive-standard	322.94 s	2.05 GB	1.89×10^{-12}
recursive-recursive	167.27 s	2.14 GB	2.05×10^{-12}
sparse-recursive	180.10 s	501 MB	2.38×10^{-12}

the different combinations of techniques. The recursive-recursive and sparse-recursive combination are 6.5 and 4.2 times faster than the standard-standard combination.

In Example 5.15, the double recursive algorithms are clearly more efficient, while the maximum absolute residual errors[8] are of the same order of magnitude. Furthermore, using the sparse approach (avoiding the construction of the Macaulay matrix) leads to a considerable reduction in memory usage. Notice that the experiments are ran on a server with a lot of available memory, so there is no computation time advantage when considering the sparse variant. In the previous example, the recursive-recursive combination is faster than the sparse-recursive combination, because the same singular value decompositions are computed but the sparse approach has some additional overhead. However, when the polynomials are sparse, the sparse approach avoids many multiplications with zero and, hence, is faster than the recursive-recursive combination even with the overhead. The next example demonstrates this behavior.

Example 5.16. The next system is a sparse system,

$$\begin{cases} p_1(\boldsymbol{x}) = x_1^{12} + x_2^{12} + x_3^{12} - 4 = 0, \\ p_2(\boldsymbol{x}) = x_1^{12} + x_2^{12} - 5 = 0, \\ p_3(\boldsymbol{x}) = x_1^6 x_3^6 - 1 = 0, \end{cases} \tag{5.33}$$

which consists of three polynomials, each of total degree 12. Because the polynomials are sparse, the sparse-recursive combination is faster than the recursive-recursive combination, as shown in Table 5.5.

[8] We calculate the absolute residual error for a polynomial system by substituting the computed solutions $(x_1^*, \ldots, x_n^*)$ in the polynomials and taking the sum of the 2-norm of the residuals, i.e., $\|\boldsymbol{e}\|_2 = \sum_{i=1}^{s} \|p_i(\boldsymbol{x}^*)\|_2$.

Table 5.5♣. Obtained results of solving the sparse system of three multivariate polynomials in Example 5.16 ($d_i = 12$) via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 22$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual error[8] of the solutions are averaged over 30 experiments.

combination	time	memory	$\max\|e\|_2$
standard-standard	572.14 s	536.32 MB	5.46×10^{-13}
standard-recursive	462.46 s	560.19 MB	5.55×10^{-13}
recursive-standard	142.22 s	536.32 MB	5.22×10^{-13}
recursive-recursive	41.45 s	560.19 MB	4.82×10^{-13}
sparse-recursive	36.71 s	131.30 MB	4.94×10^{-13}

Table 5.6♣. Obtained results of solving the Katsura problem in Example 5.17 ($d_i = 2$) via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 5$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual error[8] of the solutions are averaged over 30 experiments.

combination	time	memory	$\max\|e\|_2$
standard-standard	3.74 s	179.35 MB	4.02×10^{-12}
standard-recursive	3.57 s	179.38 MB	3.15×10^{-12}
recursive-standard	1.55 s	179.35 MB	3.39×10^{-12}
recursive-recursive	1.56 s	179.38 MB	2.38×10^{-12}
sparse-recursive	0.94 s	1.80 MB	4.22×10^{-12}

5.5.1.2 Katsura, Lotka–Volterra, and reduced economics problem

Now, we take three systems (`katsura6`, `noon5`, and `redeco8`) from the database with test problems (Section 6.5). The examples confirm the conclusions from the examples with random polynomials.

Example 5.17. We use the sparse and recurse techniques to tackle the polynomial system from Example 2.20. As Table 5.6 shows, the recursive-recursive and sparse-recursive combinations are more than 14 times faster than the standard-standard combination, while the maximum absolute residual errors[8] are almost the same. The sparse construction of the basis matrix of the null space avoids the construction of the 6468×3432 Macaulay matrix (177.58 MB). This is a clear improvement over the naive standard-standard approach (both row-wise and block-wise) presented in Chapter 2.

Table 5.7♣. Obtained results of solving the Lotka–Volterra problem in Example 5.18 ($d_i = 12$) via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 8$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual errors[8] of the solutions are averaged over 30 experiments.

combination	time	memory	$\max\|\boldsymbol{e}\|_2$
standard-standard	37.57 s	233.35 MB	6.54×10^{-14}
standard-recursive	37.54 s	233.83 MB	6.54×10^{-14}
recursive-standard	2.78 s	233.35 MB	1.41×10^{-14}
recursive-recursive	2.62 s	233.83 MB	1.41×10^{-14}
sparse-recursive	2.63 s	8.97 MB	7.15×10^{-14}

Example 5.18. We take again the Lotka–Volterra problem from Example 5.14 and solve it via the different combinations of recursive and sparse techniques. As visible in Table 5.7, the recursive-recursive and sparse-recursive combinations are more than 14 times faster than the standard-standard combination, while the maximum absolute residual errors[8] are almost the same. The sparse construction of the basis matrix of the null space avoids the construction of the 6435×4368 Macaulay matrix (224.86 MB).

Example 5.19. The following system of 8 polynomial equations in 8 variables with maximum total degree $d_{\max} = 2$ is a reduced problem from economical modeling [180, 266]:

$$\begin{cases} p_1(\boldsymbol{x}) = x_2x_3 + x_2 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_7 + x_7x_8 - x_1 = 0, \\ p_2(\boldsymbol{x}) = x_2x_4 + x_3x_5 + x_3 + x_4x_6 + x_5x_7 + x_6x_8 - 2x_1 = 0, \\ p_3(\boldsymbol{x}) = x_2x_5 + x_3x_6 + x_4 + x_4x_7 + x_5x_8 - 3x_1 = 0, \\ p_4(\boldsymbol{x}) = x_2x_6 + x_3x_7 + x_4x_8 + x_5 - 4x_1 = 0, \\ p_5(\boldsymbol{x}) = x_2x_7 + x_6 + x_3x_8 - 5x_1 = 0, \\ p_6(\boldsymbol{x}) = x_2x_8 + x_7 - 6x_1 = 0, \\ p_7(\boldsymbol{x}) = x_8 - 7x_1 = 0, \\ p_8(\boldsymbol{x}) = x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + 1 = 0. \end{cases} \tag{5.34}$$

This system has a zero-dimensional solution set with 64 solutions, all affine. After $i_\circ = 5$ iterations, all combinations have constructed a basis matrix of the null space that contains these solutions. The recursive techniques are much faster (more than 5 times faster) and result in similar absolute residual errors[8] (Table 5.8). Using the sparse approach requires less memory.

Table 5.8♣. Obtained results of solving the reduced economics problem in Example 5.19 ($d_i = 12$) via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 5$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual errors[8] of the solutions are averaged over 30 experiments.

combination	time	memory	$\max\|\boldsymbol{e}\|_2$
standard-standard	81.50 s	710.02 MB	1.60×10^{-13}
standard-recursive	79.89 s	710.05 MB	1.70×10^{-13}
recursive-standard	14.57 s	710.02 MB	9.36×10^{-13}
recursive-recursive	14.58 s	710.05 MB	9.97×10^{-13}
sparse-recursive	15.27 s	3.34 MB	8.49×10^{-13}

5.5.2 Multiparameter eigenvalue solving

We start with two MEPs that have random coefficient matrices, but we also tackle two problems in which the coefficient matrices are not random, namely the MEP from Example 3.17 and the MEP that solves the least-squares realization problem. Note that we have performed many more numerical examples, for which we refer the interested reader to [260, 262].

5.5.2.1 Random multiparameter eigenvalue problems

We solve two MEPs with random rectangular coefficient matrices via the different combinations of recursive techniques.

Example 5.20. The first random MEP that we tackle is a linear two-parameter eigenvalue problem,

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = (\boldsymbol{A}_1 + \boldsymbol{A}_2\lambda_1 + \boldsymbol{A}_3\lambda_2)\boldsymbol{z} = \boldsymbol{0}, \tag{5.35}$$

with random coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{R}^{41\times 40}$. In order to solve the MEP, we need a degree $d = 40$ block Macaulay matrix, hence $i_\circ = 39$ iterations. The computational results are summarized in Table 5.9. The recursive-recursive and sparse-recursive combinations are 95 and 276 times faster than the standard-standard combination, respectively. The maximum absolute residual errors[9]are more or less the same.

Example 5.21. Consider the quadratic three-parameter eigenvalue problem,

$$\begin{aligned}\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \big(&\boldsymbol{A}_1 + \boldsymbol{A}_2\lambda_1 + \boldsymbol{A}_3\lambda_2 + \boldsymbol{A}_4\lambda_3 + \boldsymbol{A}_5\lambda_1^2 + \boldsymbol{A}_6\lambda_1\lambda_2 \\ &+\boldsymbol{A}_7\lambda_1\lambda_3 + \boldsymbol{A}_8\lambda_2^2 + \boldsymbol{A}_9\lambda_2\lambda_3 + \boldsymbol{A}_{10}\lambda_3^2\big)\boldsymbol{z} = \boldsymbol{0},\end{aligned} \tag{5.36}$$

[9] We calculate the absolute residual error for an MEP by substituting the computed eigenvalues $(\lambda_1^*, \ldots, \lambda_n^*)$ and eigenvectors $\boldsymbol{z}^*$ in the MEP and determining the 2-norm of the residual vector $\|\boldsymbol{e}\|_2 = \|\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^*)\boldsymbol{z}^*\|_2$.

Table 5.9♣. Obtained results of solving the linear two-parameter eigenvalue problem in Example 5.20 (3 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{41\times 40}$) via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 39$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual errors[9] of the solutions are averaged over 30 experiments.

combination	time	memory	$\max\|\boldsymbol{e}\|_2$
standard-standard	5573.10 s	9.49 GB	8.13×10^{-12}
standard-recursive	5216.19 s	9.49 GB	1.27×10^{-11}
recursive-standard	138.62 s	9.49 GB	5.73×10^{-12}
recursive-recursive	58.79 s	9.49 GB	1.37×10^{-11}
sparse-recursive	20.18 s	231.12 MB	6.21×10^{-12}

Table 5.10♣. Obtained results of solving the quadratic three-parameter eigenvalue problem in Example 5.21 (10 random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{12\times 10}$) via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 23$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual errors[9] of the solutions are averaged over 30 experiments.

combination	time	memory	$\max\|\boldsymbol{e}\|_2$
standard-standard	1033.90 s	4.23 GB	1.20×10^{-11}
standard-recursive	989.54 s	4.26 GB	1.27×10^{-11}
recursive-standard	103.66 s	4.23 GB	3.36×10^{-12}
recursive-recursive	42.43 s	4.26 GB	2.78×10^{-12}
sparse-recursive	29.37 s	348.50 MB	5.93×10^{-12}

with random seed matrices $\boldsymbol{A}_j \in \mathbb{R}^{12\times 10}$. In order to solve the MEP, we need a degree $d = 25$ block Macaulay matrix, hence $i_\circ = 23$ iterations. The computational results are summarized in Table 5.10. The recursive-recursive and sparse-recursive combinations are 24 and 35 times faster than the standard-standard combination, respectively.

The previous examples show clearly that the computation times of the recursive-recursive and sparse-recursive combination are much smaller than the standard-standard approach, while the maximum absolute residual errors[9] of the solutions are more or less the same. Moreover, the computation time required to perform the last iteration with the standard approach takes more time than the total computation time of the recursive and sparse approach. Hence, even if we know the desired iteration $i_\circ$ in advance, a recursive (or sparse) construction of the basis matrix of the null space may still be advantageous. Contrary to double recursive algorithms in polynomial system solving, the sparse-recursive combination is faster than the recursive-recursive combi-

Table 5.11♣. Obtained results of solving the quadratic three-parameter eigenvalue problem in Example 5.22 that originates from a model order reduction problem via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 9$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual errors[9] of the solutions are averaged over 30 experiments.

combination	time	memory	$\max\|e\|_2$
standard-standard	2.67 s	68.01 MB	6.90×10^{-10}
standard-recursive	2.08 s	72.10 MB	6.90×10^{-10}
recursive-standard	1.32 s	68.01 MB	1.73×10^{-11}
recursive-recursive	0.95 s	72.10 MB	1.73×10^{-11}
sparse-recursive	0.96 s	20.86 MB	1.80×10^{-11}

nation, even when the problem has no sparse support. The explanation for this additional speed-up lies in the multiplications that have to be performed during the construction of the basis matrix of the null space of the block Macaulay matrix: for the sparse block Macaulay matrix, the sparse implementation avoids many multiplications with zero matrices, resulting in more multiplications but with smaller matrices, exploiting the available structure of the involved matrices. The sparse approach has the additional advantage of requiring much less memory, which is important on computer systems with limited available memory.

5.5.2.2 Model order reduction and system identification problem

Finally, we deal with two MEPs that do not have random coefficient matrices, the model order reduction problem from Chapter 3 and the least-square realization problem described in [71]. Other examples of applications that can be tackled via MEPs are given in Chapter 7. The MEPs from that chapter are good examples of problems for which these double recursive algorithms are useful.

Example 5.22. We use the different combinations of recursive and sparse techniques to solve the quadratic three-parameter eigenvalue problem (3.74) from Example 3.17. As visible in Table 5.11, we can further reduce the computation time obtained via the naive null space based implementation from Chapter 3 (which corresponds to the standard-standard combination) by applying the techniques developed in this chapter. Furthermore, the naive implementation builds the 2200×2912 block Macaulay matrix explicitly, which requires 51.25 MB of memory.

Example 5.23. Given a data sequence $y_0, \ldots, y_{N-1}$ ($\boldsymbol{y} \in \mathbb{R}^{N\times 1}$), find the adapted data sequence $\hat{y}_0, \ldots, \hat{y}_{N-1}$ so that the misfit $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2$ is minimized and $\hat{\boldsymbol{y}} \in \mathbb{R}^{N\times 1}$ is the output of a model of pre-specified order n [71]:

$$\hat{y}_k = \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{x}_0, \tag{5.37}$$

where $\boldsymbol{x}_0 \in \mathbb{R}^{n\times 1}$ is the initial state, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the system matrix, and $\boldsymbol{C} \in \mathbb{R}^{1\times n}$ is the output vector. This problem is the least-squares realization problem. In [71], it has been shown how this identification problem corresponds to a quadratic MEP, with the number of parameters equal to n. When we consider a model of order $n = 2$, we obtain a quadratic two-parameter eigenvalue problem

$$\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = \big(\boldsymbol{A}_1 + \boldsymbol{A}_2\lambda_1 + \boldsymbol{A}_3\lambda_2 + \boldsymbol{A}_4\lambda_1^2 + \boldsymbol{A}_5\lambda_1\lambda_2 + \boldsymbol{A}_6\lambda_2^2\big)\boldsymbol{z} = \boldsymbol{0}, \tag{5.38}$$

with the coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{R}^{(3N-4)\times(3N-5)}$ as described by De Moor [71]. We consider the MEP in Example 5.23 constructed from a series of $N = 6$ random data points $\boldsymbol{y}$, which results in 14×13 coefficient matrices $\boldsymbol{A}_j$, with $j = 1, \ldots, 6$. This problem has a positive-dimensional solution set at infinity, so we need to compute a basis matrix of the null space for every degree and check its rank structure. A block Macaulay matrix of degree $i_\circ = 22$ has a gap zone that can *accommodate* the shift and allows us to deflate the positive-dimensional solution set at infinity via a column compression. In Table 5.12, we compare the computation time and maximum absolute residual errors[9] of the different combinations of standard/recursive/sparse techniques. The recursive-recursive and sparse-recursive algorithm are much faster than the standard-standard algorithm (non-recursive approach), while resulting in more or less the same absolute residual errors. The recursive-recursive and sparse-recursive combinations are 11 and 14 times faster than the standard-standard combination, respectively.

5.6 Conclusion

We presented recursive techniques to update a numerical basis matrix of the null space of the block Macaulay matrix and block row matrix. These recursive techniques use the numerical basis matrix computed during the previous iteration in order to efficiently determine an update. Furthermore, we also proposed a sparse alternative implementation for the (block) Macaulay matrix, without explicitly constructing this large (block) Macaulay matrix. We provided several numerical examples to illustrate the properties of these algorithms and to compare them with the standard full singular value decomposition. Furthermore, we combined recursive techniques into a double recursive algorithm to solve systems of multivariate polynomial equations and MEPs more efficiently. By exploiting the available structure and sparsity, we obtained impressive reductions in computation time and memory usage compared to the existing

Table 5.12♣. Obtained results of solving the quadratic two-parameter eigenvalue problem in Example 5.23 that originates from the least-squares realization problem (with $N = 6$ random data points) via different recursive combinations. The total computation time to build a numerical basis matrix of the corresponding Macaulay matrix (requires $i_\circ = 22$ iterations), the total memory usage to obtain this basis matrix, and the maximum absolute residual errors[9] of the solutions are averaged over 30 experiments.

combination	**time**	**memory**	$\mathbf{max}\|\boldsymbol{e}\|_\mathbf{2}$
standard-standard	15.56 s	142.81 MB	3.27×10^{-7}
standard-recursive	12.36 s	143.82 MB	3.37×10^{-7}
recursive-standard	4.54 s	142.81 MB	4.31×10^{-7}
recursive-recursive	1.39 s	143.82 MB	3.76×10^{-7}
sparse-recursive	1.05 s	13.22 MB	3.15×10^{-7}

non-recursive approach, while keeping more or less the same accuracy. For example, in our Lotka–Volterra example, we observed a factor 14 improvement in computation time compared to the non-recursive approach and noticed that the sparse implementation avoids the construction of a 6435×4368 Macaulay matrix. For the block Macaulay matrix, even more impressive time reductions were demonstrated in this chapter: Example 5.20 shows how the recursive-recursive and sparse-recursive combinations are 95 and 276 times faster than the standard-standard combination, respectively. Also the memory improvement, going from 9.49 GB to 231.12 MB, is remarkable.

> **Motivational example.** Let us run the different combinations of recursive techniques once more, but now on the polynomial system and MEP that tackle the first-order ARMA model identification problem. To highlight the improvements better, we consider for both problems a data sequence of $N = 6$ random output values y_k. The polynomial system consists of 7 cubic polynomials in 7 variables, while the MEP is a quadratic two-parameter eigenvalue problem with 17×16 coefficient matrices. The results are summarized in Table 5.13: the double recursive algorithms indeed prove to be useful in both multivariate root-finding and multiparameter eigenvalue-finding. As mentioned in Section 3.6, the MEP approach is more efficient when the number of data points N increases.

Numerical examples on practical problems, like the first-order ARMA model identification and the second-order least-square realization problem, motivated the need for faster algorithms: the proposed recursive (and sparse) algorithms clearly outperformed the standard approach. The recursive approach and sparse adaptation have given us the opportunity to solve larger systems of multivariate polynomial equations and MEPs (MEPs) than possible with the standard approach. In the future, we will improve our current algorithms to further push the limits:

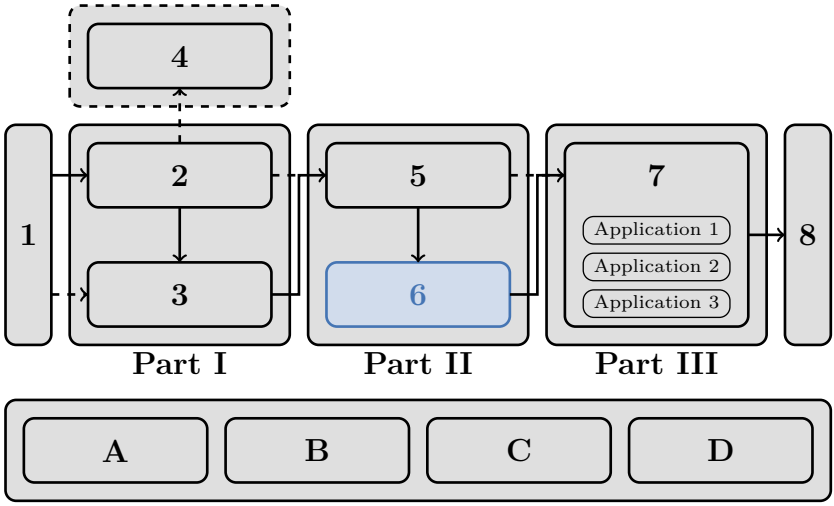
Table 5.13. Obtained results of solving the motivational example for a data sequence of $N = 6$ random data points via a polynomial system or a quadratic two-parameter eigenvalue problem. We apply the different combinations of recursive techniques to both problems. The total computation time to build a numerical basis matrix of the corresponding (block) Macaulay matrix, the total memory usage to obtain this basis matrix, and the maximum absolute residual errors[8,9] of the solutions are averaged over 30 experiments.

	combination	**time**	**memory**	$\mathbf{max\|\|}e\mathbf{\|\|_2}$
Macaulay matrix	standard-standard	545.94 s	1.67 GB	7.82×10^{-15}
	standard-recursive	562.07 s	1.69 GB	9.99×10^{-15}
	recursive-standard	89.88 s	1.67 GB	9.99×10^{-15}
	recursive-recursive	91.34 s	1.69 GB	4.03×10^{-14}
	sparse-recursive	89.74 s	0.16 GB	4.03×10^{-14}
block Macaulay matrix	standard-standard	1191.11 s	644.77 MB	3.27×10^{-8}
	standard-recursive	1200.62 s	647.10 MB	3.37×10^{-8}
	recursive-standard	34.76 s	644.77 MB	4.31×10^{-8}
	recursive-recursive	8.65 s	647.10 MB	3.76×10^{-8}
	sparse-recursive	9.16 s	41.63 MB	3.15×10^{-8}

- Clearly, we do not yet exploit all the available structure. Some promising research is done by various researchers at this time to exploit the quasi-Toeplitz structure of the Macaulay matrix.
- We will also consider memory-efficient implementations (e.g., looking at cache locality).
- Moreover, we currently investigate how to replace the second recursive technique by more efficient procedures to reveal the rank structure (e.g., the URV algorithms in [2]).
- In many problems, this rank checks pose a huge burden on the solutions algorithms. Analogue approaches for Hankel and block Hankel matrices could also be useful in other application areas.
- Furthermore, we want to translate our efforts from the singular value decomposition to the QR decomposition, enabling more efficient column space based solution approaches for systems of multivariate polynomial equations and MEPs. A fast and sparse implementation of the (Q-less) QR decomposition is much faster than the traditional singular value decomposition to compute a numerical basis matrix of the null space. Therefore, one of our current research efforts is to improve current implementations and to exploit both the structure and the sparsity of the (block) Macaulay matrix. Together with more efficient implementations, this will give us the machinery to tackle much larger problems in the future.

MacaulayLab: About the Implementation

To utilize and test the algorithms discussed in the preceding chapters, it is essential to implement them within a comprehensive toolbox. In this chapter, we explore the implementation of these algorithms in the MacaulayLab toolbox, which enables us to solve various applications, like those presented in Chapter 7. For the software toolbox to be useful, it should strive for a good computational efficiency and numerical robustness, while keeping user-friendliness in mind. This requires a careful implementation of the available and new algorithms. MacaulayLab not only provides a platform to solve systems of multivariate polynomial equations and rectangular multiparameter eigenvalue problems, but also includes a database with many test problems. The database serves as a means to test MacaulayLab's algorithms and the test problems provide benchmarks for other toolboxes.



Contributions. This chapter presents the implementation of the different (block) Macaulay matrix algorithms in one coherent and user-friendly toolbox. It provides an overview of the different implementation decisions that have been made, compares the available solution approaches, and positions MacaulayLab with respect to other toolboxes.

Outline. Firstly, we introduce MacaulayLab in Section 6.1, highlighting the link with the other chapters of this dissertation. Section 6.2 contains an overview of other software toolboxes to deal with system of multivariate polynomial equations and (square and rectangular) multiparameter eigenvalue problems. Next, in Section 6.3, we dive into the implementation of the implemented algorithms, highlighting the important implementation choices that have been made. Furthermore, Section 6.4 is about one of MacaulayLab's key features: its independence from the choice of polynomial basis and monomial ordering. We showcase in Section 6.5 the test problem database, while we evaluate the different solution approaches of MacaulayLab and compare them with similar solvers in Section 6.6. Finally, in Section 6.7, we conclude this chapter by outlining possible upgrades in future releases.

6.1 Introduction

An essential product of this research is, next to scientific contributions in the dissertation, the accompanying software toolbox: MacaulayLab is a Matlab toolbox that features algorithms to solve systems of multivariate polynomial equations and rectangular multiparameter eigenvalue problems (MEPs). While both problem types seem unrelated at first, they are connected through the (block) Macaulay matrix and, consequently, solved via a similar numerical linear algebra methodology. In that sense, this toolbox is quite unique; the combined nature of the two types of problems allows for one tool(box) to tackle them both.

In this chapter, we discuss the implementation of the algorithms from Chapters 2, 3 and 5, and we highlight several implementation details[1]. MacaulayLab solves both types of problems by transforming them into multidimensional realization problems in the right null space or column space of the (block) Macaulay matrix constructed from the coefficients of the polynomials or the coefficient matrices of the rectangular MEP. This transformation is obtained via iterative, recursive, or sparse algorithms. Similarly, the required rank checks to determine whether this right null space and column space contain the affine solutions can also be performed recursively. In the end, only numerical linear algebra tools are used, like singular value, QR, or eigenvalue decompositions–no symbolic tools enter the picture. The toolbox relies on the decades of advancements in numerical linear algebra, resulting in computationally efficient and numerically robust algorithms. An important feature of MacaulayLab is that all the algorithms are implemented without depending on a particular polynomial basis or monomial ordering, allowing the user to choose what suits the application best. It is known that in some situations an orthogonal polynomial basis, like the Chebyshev polynomials, has superior numerical properties [247, 248], while results in algebraic geometry often depend on the chosen monomial ordering [90]. Offering the user the choice to select a particular polynomial basis and monomial ordering can thus be very useful in applications. Next to the implementation of the many algorithms developed in the previous chapters, this toolbox also features a database with test problems, which can be used to test the toolbox or to benchmark other software packages. Evidently, the toolbox has been developed to be very user-friendly and easy to use, whether the user simply wants to solve the problem or wants to learn more about its properties.

> **Motivational example.** While the previous chapters provided different algorithms to tackle multivariate polynomial systems or rectangular MEPs, there are many (small and large) implementation details that have to be considered before we can effectively start to tackle challenges like the globally optimal autoregressive moving-average (ARMA) model identification problems. The goal of this chapter is to develop a toolbox that combines the

[1] Notice that this chapter is not a guide on how to use MacaulayLab. For more information about using the toolbox, we refer the interested reader to the user-manual in Appendix D. We repeat once more that MacaulayLab is available online at `www.macaulaylab.net`.

different algorithms from the previous chapters and creates a user-friendly approach to deal with these problems. By the end of this chapter, it should be clear how MacaulayLab yields the globally optimal parameters of the ARMA model.

6.2 Other software packages

Of course, there exist many other software packages to solve systems of multivariate polynomial equations or (square and rectangular) MEPs, mostly dedicated to tackling only one type of problems. We have compiled, to the best of our knowledge, the following two lists of software packages to solve these problems.

6.2.1 Systems of multivariate polynomial equations

Currently, the most efficient way of obtaining the common roots of a system of multivariate polynomials with available software is via homotopy continuation algorithms. These algorithms employ a mixture of techniques from algebraic geometry and nonlinear optimization to continuously deform a starting system with known solutions into the original system with unknown solutions, while tracking the paths of the solutions (see, for example, [157, 221, 265]). Although issues with ill-conditioning still exist, homotopy continuation methods are inherently parallel, i.e., each isolated solution can be computed independently, and are currently among the most competitive algorithms to solve systems of multivariate polynomial equations. Their main disadvantage is that they only work for square (i.e., the number of equations is equal to the number of unknown variables) systems of multivariate polynomial equations. Because of their efficiency and applicability, software for homotopy continuation comes in many flavours. Some toolboxes that use homotopy continuation to tackle polynomial systems are PHCpack [264], HOMPACK [271], PHoM [103, 104], HOM4PS [58], NAG4M2 [156] (part of Macaulay2), Bertini [19, 20], and HomotopyContinuation.jl [45].

Another approach to deal with systems of multivariate polynomial equations is to reduce the problem into subsequent univariate root-finding problems or eigenvalue problems. On the one hand, symbolic software packages, like Maple, Singular, Magma, and msolve, use a symbolic approach to solve polynomial systems, by creating a Gröbner basis to create a triangular system that can be solved via back-substitution or to construct the multiplication matrices (the eigenvalues of which are related to the solutions of the polynomial system). It is, on the other hand, also possible to rely solely on numerical linear algebra techniques to rephrase polynomial systems as eigenvalue problems. MacaulayLab falls into this category of solvers. Batselier [21] and Dreesen [78] have also approached the problem via the Macaulay matrix and have implemented their algorithms in the Matlab packages PNLA and RootFinding[2], respectively.

[2]The Matlab package RootFinding is available upon request with its developer, Philippe Dreesen.

Similar to MacaulayLab, they both use the Macaulay matrix to set up one or multiple multidimensional realization problems. While RootFinding is a quite naive implementation of the Macaulay matrix approach as described in [78], PNLA offers functions that take advantage of the structure and sparsity of the Macaulay matrix. When the system consists of two bivariate polynomials, the Matlab package BiRoots by Plestenjak and Hochstenbach [201] can be used, which transforms the problem into a square MEP and solves this problem via MultiParEig (see below). For users of the Julia language, we want to highlight the packages AlgebraicSolvers.jl by Telen et al. [243] and JuliaEigenvalueSolver.jl by Bender and Telen [32], which tackle the polynomial systems by using the Macaulay (or a related) matrix to set up the multiplication matrices.

6.2.2 Multiparameter eigenvalue problems

In the literature, the available methods to tackle MEPs mainly concern square problems. Small problems can be tackled via simultaneous triangularization of the associated system of coupled generalized eigenvalue problems (GEPs), while for larger problems subspace techniques and homotopy continuation algorithms overcome the scalability issues of this simultaneous triangularization. While implementations of these different approaches are available, the only comprehensive toolbox to deal with square MEPs, to the best of our knowledge, is MultiParEig by Plestenjak [198]. It features a wide array of algorithms (both direct and iterative) and can deal with singular and non-singular problems.

MacaulayLab is the first toolbox that focusses on rectangular MEPs. Very recently, Hochstenbach et al. [118] have shown that it is possible to translate (linear) rectangular problems into a square formulation. This transformation, together with the available methods for solving square MEPs, is added to MultiParEig in one of its last updates[3], adding a second toolbox to list of software packages that can solve rectangular MEPs. When the problem is not linear, however, an additional linearization step is required (before or after the transformation), while MacaulayLab deals with polynomial problems directly.

6.3 Implementation of MacaulayLab

As previously mentioned, MacaulayLab can deal with both systems of multivariate polynomial equations and (rectangular[4]) MEPs. In order to keep the toolbox user-friendly, describing a problem is kept very simple. Both problem types are internally represented by the same class `problemstruct`: all necessary information is stored in the cell arrays `coef` and `supp`[5]. Although it is

[3]Methods to solve linear rectangular MEPs were added in version 2.7.0.0 of MultiParEig (December 6, 2022).

[4]In the remainder of this chapter, we no longer mention the qualification *rectangular* explicitly. We always consider rectangular MEPs, except when denoted otherwise.

[5]A cell array is a data type in Matlab with indexed data containers, called cells, where each cell can contain any type of data. For MacaulayLab, each cell of `coef` and `supp` correspond to one polynomial (matrix) equation. A multidimensional array in the corresponding `coef`

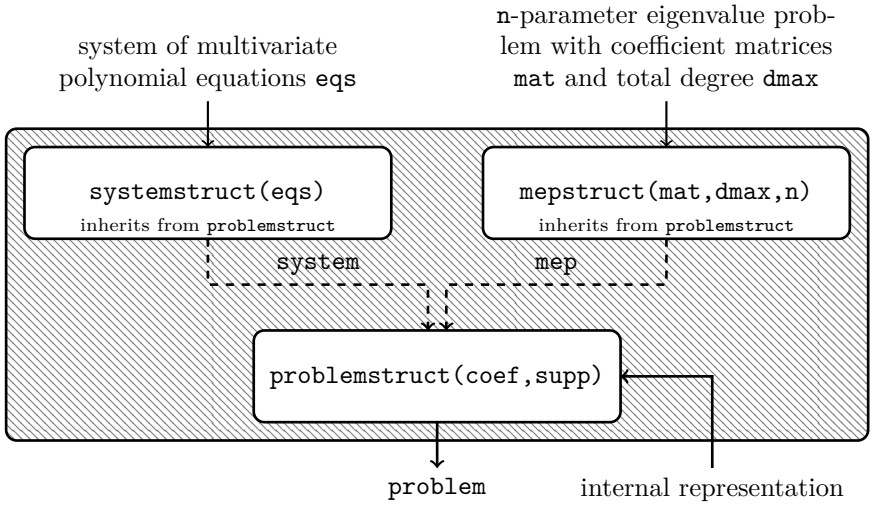


Figure 6.1. Construction of the data structure to represent a system of multivariate polynomial equations or an MEP. Both problem types are internally represented by the same `problemstruct`: all necessary information is stored in the cells `coef` and `supp`. The sub-classes `systemstruct` and `mepstruct` provide constructors to set-up the problems more easily, but it is also possible to submit the problem directly in its internal representation.

also possible to submit the problem directly in its internal representation, the sub-classes `systemstruct` and `mepstruct` provide constructors to set-up the specific problems more easily (Figure 6.1). The reason for using this dedicated, uniform internal data structure is to allow that the functions of MacaulayLab can manipulate both types of problems similarly. The two dedicated sub-classes and related constructors make sure that the user supplies all the necessary information about the problem and allow for a more user-friendly interaction. Especially in the case of a rectangular MEP this constructor is essential: from the number of coefficient matrices, it is impossible to the determine the exact structure of the rectangular MEP. For example, both a linear 5-parameter eigenvalue problem and a quadratic 2-parameter eigenvalue problem comprises of six coefficient matrices.

With the problem given in the correct data structure, the user submit it to the solver, allowing MacaulayLab to identify the problem type and solve the problem accordingly. When comparing the Macaulay matrix algorithm to solve systems of multivariate polynomial equations (Algorithm 2.1) with the block Macaulay matrix algorithm to solve MEPs (Algorithm 3.1), it is immediately clear that both types of problems can be tackled by applying the same six steps (Sections 6.3.1 to 6.3.6):

1. enlarge the solution subspace,

cell contains the coefficients/coefficient matrices, while a two-dimensional array in the `supp` cell stores the support of these coefficients/coefficient matrices.

2. check whether the solution subspace can accommodate the shift,
3. remove the solutions at infinity,
4. exploit the shift-invariance of the solution subspace,
5. cluster the affine solutions to obtain a better accuracy (optional), and
6. compute the residual errors of the obtained affine solutions (optional).

Figure 6.2 visualizes the main functions in MacaulayLab that enable these different steps and shows how they interact with each other. It is easy to identify the six different steps as the key functions of the function diagram. (Notice that `enlarge` and `check` are two local functions inside the solver `macaulaylab`.) The solution methodology for both problem types is built around these six steps and re-uses, because of the similarity between both problem types, most of the functions. For example, the function `macaulay` builds a Macaulay or block Macaulay matrix given the particular problem. While most of the time the same functions can deal with both problem types, this requires a very careful implementation that takes into account the subtle differences between them. The toolbox is built in a modular fashion, which allows easy expansions and improvements in future updates. For example, when we develop a faster approach to compute the right null space of the (block) Macaulay matrix or to perform the necessary rank checks, we can simply replace that function by its more efficient counterpart (or add the option to choose between both approaches).

Clearly, MacaulayLab contains many possibilities and has many options that can be set via an options structure. As mentioned in Chapters 2 and 3, the solutions of the problem can be retrieved using either the right null space or column space of the (block) Macaulay matrix. Depending on this choice, the shifts can be performed in one of those subspaces, indicated by `<sub>` (which is `null` or `column`). How to obtain these solution subspaces can also be chosen; iteratively, recursively, or sparsely. Note that many of the functions depend on the chosen polynomial basis (implemented in `<basis>`) and monomial ordering (implemented in `<position>`), more information about this feature is provided in Section 6.4. To learn more about the different options, we refer the interested reader to the manual of MacaulayLab (Appendix D).

Via the output, the user retrieves the (affine) solutions and additional properties of the given problem. When using the verbose option of MacaulayLab, the output shows in real-time the stabilization diagram of the solution subspace and the current size of the gap zone, among many other things. The verbose option can be very useful when the user wants to learn more about the properties of the problem or the behavior of the algorithms (Figure 6.3). All the necessary code related to the verbose output of the algorithms is put in `verbose`, again with the modularity of the toolbox in mind.

In Sections 6.3.1 to 6.3.6, we dive deeper into the six different steps required to solve a system of multivariate polynomial equations or MEP via the (block) Macaulay matrix. We deal with one step at a time and highlight the different

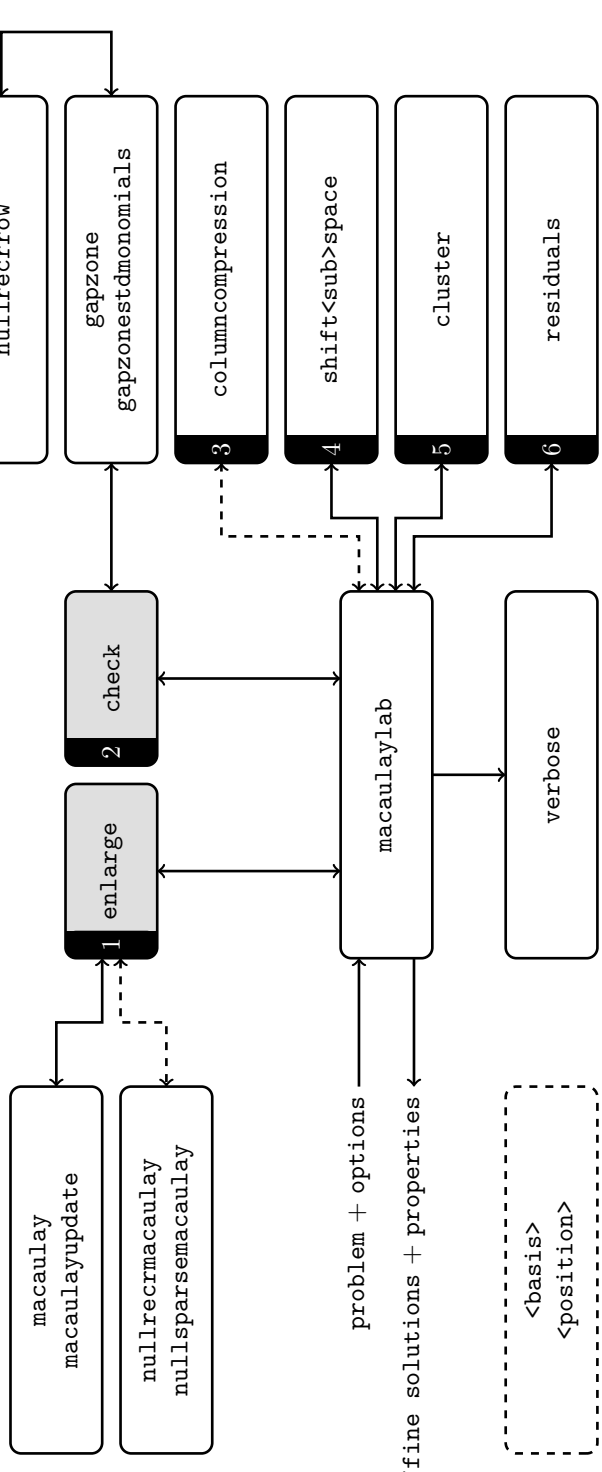


Figure 6.2. Diagram of the main functions (rectangles) implemented in MacaulayLab and their dependencies (arrows). The gray rectangles indicate local functions from `macaulaylab` and the dashed arrows denote dependencies that are only used in the null space based approach. Since the solution approach is similar for both problem types, the same six steps are used and most functions are capable of dealing directly with both problem types (scalars coefficients are replaced by coefficient matrices, and vice versa). Depending on the choice of using the right null space or column space, the solution subspace `<sub>` is `null` or `column` space. Note that most functions depend on the chosen polynomial basis (implemented in `<basis>`) and monomial ordering (implemented in `<position>`).

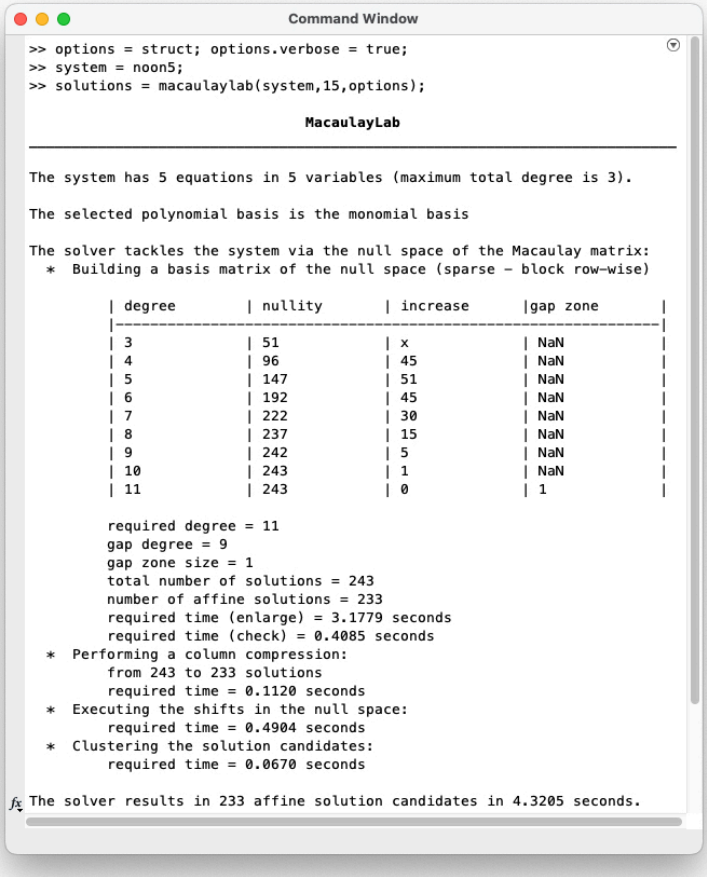


Figure 6.3. Output of MacaulayLab when solving the `noon5` system from the database via the default Macaulay matrix approach. When the verbose option is enabled, the toolbox displays a lot of information about the problem, the solution procedure, and the obtained (affine) solutions.

implementation decisions. It is important to note that implementation decisions always represent making a substantiated choice in the trade-off between computational efficiency and numerical robustness.

6.3.1 Enlarge the solution subspace

The first step in the solution approach, whether we solve a system of multivariate polynomial equations or an MEP, is to build and enlarge the solution subspace. MacaulayLab supports two solution subspaces: the right null space and column space of the (block) Macaulay matrix. When we choose to use the right null space as the solution subspace (Sections 2.4 and 3.4), we need to build a basis matrix of the right null space of the (block) Macaulay matrix and enlarge it until it contains a sufficiently large gap zone, which we check at every iteration (see how in Section 6.3.2). Enlarging this basis matrix can be done iteratively (via `macaulay` and `null`), recursively (via `macaulayupdate` and `nullrecrmacaulay`), or sparsely (via `nullsparsemacaulay`). We have developed these algorithms in Chapter 5. For the column space, we only need to build the (block) Macaulay matrix in every iteration, which can again be done iteratively or recursively (Sections 2.5 and 3.5).

It is important to note that, depending on the chosen polynomial basis and monomial ordering, the resulting matrices can have a very different structure. However, everything is implemented independent of the polynomial basis and monomial ordering, as we explain in Section 6.4.

6.3.2 Check the rank structure

Of course, we need to know whether the solution subspace contains a gap zone that can accommodate the shift used in Section 6.3.4, i.e., whether the degree of the (block) Macaulay subspace is equal to the solution degree (Sections 2.4.3 and 3.4.3). We do this via rank checks with an absolute tolerance, but this can be adapted by the user. There exist many ways to do this: We can look at the individual rows/columns, we can look at entire degree blocks, etc. By default, we work per degree block, since this is numerically more robust and computationally more efficient than row-wise rank checks (Remark 2.5).

Both the basis matrix of the right null space and the flipped transpose of the (block) Macaulay matrix can be interpreted as block row matrices, when we consider the rows/columns degree block-wise. To improve computational efficiency, we use, therefore, the recursive technique developed in Chapter 5 and implemented in `nullrecrrow`. This gives the double recursive nature to the solution algorithm.

Note that we typically do not immediately check the rank structure for all iterations of the enlargement. For zero-dimensional solution sets, we wait until the nullity is stabilized, and we only start checking the rank structure of the solution space after stabilization. This way, we avoid many superfluous rank checks and create a more efficient solution algorithm. When the problem has a positive-dimensional solution space, it is useless to wait for stabilization, since this will never happen (Section 2.6.1). By setting an optional flag (i.e.,

`posdim = true`), the solution algorithm does not wait for the nullity to stabilize, but checks the rank structure of the solution subspace for every iteration, a necessity for problems with a positive-dimensional solution space.

6.3.3 Perform the column compression

For the null space based approach, we need to deflate the solutions at infinity from the right null space via a column compression of its basis matrix (Theorem 2.3), which is implemented via a single singular value decomposition. In the column space based approach, however, this is not necessary, since the backward QR decompositions to set-up the shift problems (Section 6.3.4) remove those infinite solutions implicitly.

6.3.4 Shift in the chosen solution subspace

Given a large-enough solution subspace, the shift problems are set-up as described in Chapters 2 and 3. We consider $n+1$ shift problems: we shift with a random linear shift polynomial and with the n different solutions components (i.e., the variable coordinates x_i or eigenvalue parameters λ_i, for $i = 1, \ldots, n$). Other shift polynomials are possible (Section 2.6.2), but using a random shift polynomial avoids false multiplicities (Section 2.4.2). We use a Schur decomposition to solve the first shift problem and re-use the obtained unitary matrix to obtain the upper triangular matrices of the other n shift problems by premultiplication and post-multiplication (Section 2.4.2). This makes sure that the different components of the same solution are at the same position on the diagonal of the upper triangular matrices, which removes the need for matching the different solution components. Note that the $n+1$ default shift problems only consist of linear shifts. This has a clear advantage: We do only need a gap zone of one degree block. A shift polynomial with a large degree requires a large gap zone, hence more iterations are needed to construct a solution subspace that can accommodate this shift.

When choosing the column space based approach, every shift problem requires one backward QR decomposition to construct the necessary matrices. The backward QR decompositions can be implemented easily (when not considering the structure and sparsity of the (block) Macaulay matrix) as

```
>> [~,R] = qr(fliplr(N)); R = fliplr(R);
```

This backward QR decomposition implicitly removes the solutions at infinity. Furthermore, the shifts with the different solution components are monomials, for which we do not need to build and invert the row permutation matrix, but we can simply re-order the columns of the (block) Macaulay matrix directly (Remark 2.8).

To avoid the sparse multiplication of the row-selection and row-combination matrices with the subspace matrix, we perform these multiplications indirectly

and do not build the row-selection/combination matrices explicitly. It is important to take into account the correct polynomial basis and monomial ordering when shifting, otherwise the results are incorrect, as we explain in Section 6.4.

6.3.5 Cluster the affine solutions

An additional, but optional, step in the solution approach is to cluster the obtained affine solutions (Section 2.4.2). We cluster the different solutions based on the evaluations of the random shift polynomial, which are obtained as the eigenvalues of the first shift problem. Clustering means here that similar evaluations of the shift polynomial are considered to be equal solutions of the problems; hence, we construct clusters of multiple solutions. When the shift polynomial is chosen to be random (for other polynomials, we do not perform this clustering step by default), similar evaluations correspond to the same solutions with a probability equal to one. Every solution for which the evaluation of the random shift polynomial is grouped in the same cluster, is considered to be the same solution. Therefore, we take the geometric mean of the cluster for every component of the solution (i.e., the cluster center). The idea for this additional clustering step comes from Corless et al. [62] in the context of polynomial system solving, but is also very useful when solving MEPs.

6.3.6 Compute the residual errors

Finally, we compute the residual error for every obtained affine solution:

- For a system of multivariate polynomial equations, we evaluate all the polynomials in the solution and sum the 2-norm of these residuals.
- For an MEP, we evaluate the matrix polynomial in the solution and compute the 2-norm of the residual vector. The eigenvector associated with the obtained eigenvalue is computed as the right-most right singular vector of the evaluated matrix pencil.

Depending on the chosen polynomial basis, and monomial ordering, a particular evaluation function `problemvalue` is used to achieve this goal. MacaulayLab returns both a list with all the residual errors and the maximum residual error of all obtained affine solutions.

6.4 Polynomial basis and monomial ordering

MacaulayLab is implemented independently from the specific polynomial basis and monomial ordering. This means that the user can choose the polynomial basis and monomial ordering that suits the problem best, while all the functions keep working out-of-the-box. In Section 2.6.3, it was already explained why the solution approaches still work. But, using a different polynomial basis or monomial ordering greatly influences the structure of the involved matrices. How is this implemented in MacaulayLab?

- Every time it is necessary to shift (i.e., a multiplication of two monomials) in the code, the functions call the chosen function `<basis>`, which implements the basic shift operation in a particular polynomial basis. By using the correct shift function, the other functions do not need to care about the specific polynomial basis. For example, in the standard monomial basis, when shifting $\phi_{11}(\boldsymbol{x}) = x_1x_2$ with $\phi_{10}(\boldsymbol{x}) = x_1$, `basismon` results in $\phi_{21}(\boldsymbol{x}) = x_1^2x_2$, while `basischeb` in the Chebyshev polynomial basis yields $\frac{1}{2}(\phi'_{21}(\boldsymbol{x}) + \phi'_{01}(\boldsymbol{x}))$, where the basis polynomials now correspond to $\phi'_{21}(\boldsymbol{x}) = \big(2x_1^2 - 1\big)x_2$ and $\phi'_{01}(\boldsymbol{x}) = x_2$.
- Similarly, the monomial ordering is important when we need to know the position of a monomial in the polynomial basis (for example, to build the Macaulay matrix or set-up the shift problems). By leaving this computation to a dedicated function that implements the correct monomial ordering `<position>`, the correct position in the chosen monomial ordering is always obtained. For example, using `posgrinvlex` positions $x_1^2x_2$ for $n = 2$ variables at position 8, while `posgrevlex` yields 9.

By default, MacaulayLab uses the standard monomial basis and graded inverse lexicographic (GRINVLEX) ordering. The code also contains the pre-implemented Chebyshev polynomial basis, graded lexicographic (GRLEX) ordering, and graded reverse lexicographic (GREVLEX) ordering. Users can easily implement and use other polynomial bases or monomial orderings.

6.5 Database with test problems

Next to the implementation of the various (block) Macaulay matrix algorithms, MacaulayLab also includes a large set of test problems, which can be used to test the toolbox's features and act as benchmarks for other software. At the time of writing this chapter, the database contains 230 systems of multivariate equations and 30 MEPs, some of which are parameter dependent. In Tables 6.1 and 6.2, we highlight some systems and MEPs from that database. Additional information about the problems is also stored and available in the database, accessible by calling the overloaded function `disp` (Figure 6.4).

Because MacaulayLab is polynomial basis independent (Section 6.4), the database also contains problems that are given in another polynomial basis. When using such a problem, the MacaulayLab solver recognizes the correct polynomial basis automatically and applies the correct shift and evaluation operations (if the corresponding functions are implemented). These problems can be recognized easily in the database, because an identifier for the polynomial basis is appended to the name. For example, `system_cheb.m` and `mep_cheb.m` for the Chebyshev polynomial basis.

6.6 Comparison of approaches and solvers

In this section, the objective is to compare the various solution approaches available in MacaulayLab and to evaluate the performance of its default solution

Table 6.1. Small selection of systems in the database, with some of their key properties: a system has s equations in n variables with maximum total degree equal to $d_{\max}$, resulting in a total of m_b solutions, of which m_a are affine solutions. The database contains a wide variety of problems, sometimes parameter dependent. For example, `arma11eq(y)` depends on a data sequence $\boldsymbol{y}$ with N data points. For such system, the number of solutions depends on the particular parameter values (indicated by /). An underscore is used to indicate whether a problem is given in another polynomial basis than the standard monomial basis.

name	s	$d_{\max}$	n	m_b	m_a
`noon3`	3	3	3	27	21
`batselier5`	3	12	3	1728	1728
`conform`	3	4	3	64	16
`dreesen10`	4	4	4	∞	2
`cyclic5`	5	7	5	120	70
`walsh`	6	7	6	∞	7
`katsura7`	8	2	8	128	128
`overdet1`	8	4	4	10	10
`arma11eq(y)`	$N+1$	3	$N+1$	/	/
`walsh_cheb`	6	3	6	∞	7

Table 6.2. Small selection of MEPs in the database, with some of their key properties: an n-parameter eigenvalue problem has maximum total degree $d_{\max}$ and $k \times l$ coefficient matrices, resulting in a total of m_b solutions, of which m_a are affine solutions. The database contains a wide variety of problems, sometimes parameter dependent. For example, `realization(y,n)` depends on a data sequence $\boldsymbol{y}$ of length N and a parameter n (we use $m = nN - n^2$ to simplify notation). For such system, the number of solutions depends on the particular parameter values (indicated by /). An underscore is used to indicate whether a problem is given in another polynomial basis than the standard monomial basis.

name	$d_{\max}$	n	k	l	m_b	m_a
`volkmer`	1	2	12	6	6	6
`muhic4`	1	2	18	9	∞	4
`alsubaie3`	1	3	5	3	∞	4
`h2fom2r3`	2	3	10	8	∞	209
`hkp2`	2	2	3	2	12	12
`h2f5`	2	2	6	5	∞	17
`h2fourdisk`	2	4	10	7	∞	129
`realization(y,n)`	2	n	$m+n$	$m+1$	/	/
`cube`	3	2	21	20	1890	1890
`wing_cheb`	2	1	3	3	6	6

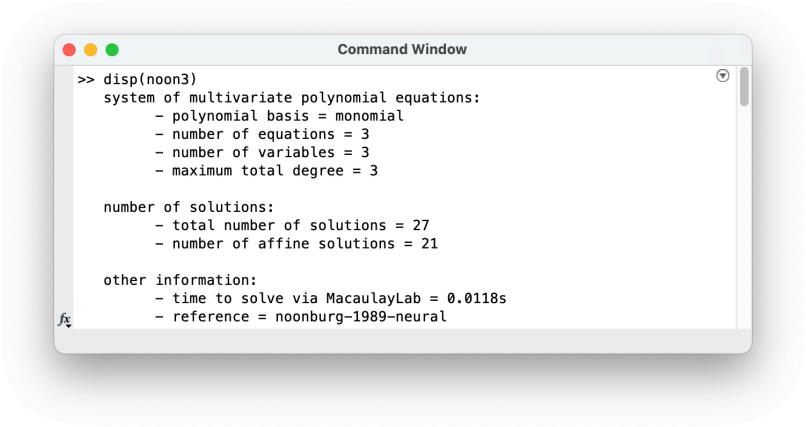


Figure 6.4. Information displayed for the `noon3` system in the database.

approach with respect to other solvers[6]. We start by exploring the different solution approaches of the toolbox (Section 6.6.1). Subsequently, we juxtapose the default solution approach of MacaulayLab next to some other Matlab solvers, both for systems of multivariate polynomial equations (Section 6.6.2) and MEPs (Section 6.6.3).

6.6.1 Different solution approaches in MacaulayLab

There are different solution approaches available to solve a system of multivariate polynomial equations with this toolbox. In order to compare the computational time of the different solutions approaches, we solve two systems via five different sets of options (summarized in Table 6.3):

- option 1: a column space based approach that iteratively builds the (block) Macaulay matrix and uses column-wise rank checks on the columns of that (block) Macaulay matrix,
- option 2: a null space based approach that iteratively builds the (block) Macaulay matrix and uses row-wise rank checks on the rows of the basis matrix of the right null space of that (block) Macaulay matrix,
- option 3: a null space based approach similar to option 2, but now using degree block-wise rank checks on the basis matrix of the right null space,
- option 4: instead of an iterative construction of the (block) Macaulay matrix and a basis matrix of its null space, option 4 uses a recursive approach to enlarge the solution subspace, also with degree block-wise rank checks, and

[6] All computations in this chapter are performed on a MacBook Pro that has an M1 CPU (2020) working at 3.2 GHz (8 cores) and 16 GB random-access memory (RAM). This choice has been made to evaluate the performance and capabilities of the toolboxes under typical working conditions, rather than on a server with extensive memory resources (Section 1.4).

Table 6.3. Summary of the different sets of options. The five sets differ in three options: the solution subspace used, how this subspace is enlarged, and how the rank structure is checked.

set	subspace	enlarge	check
option 1	column space	iterative	columns
option 2	null space	iterative	rows
option 3	null space	iterative	degree blocks
option 4	null space	recursive	degree blocks
option 5	null space	sparse	degree blocks

- option 5: a sparse null space based approach with degree block-wise rank checks, which means that the (block) Macaulay matrix is no longer constructed, but the basis matrix is updated directly.

For the first problem, the `noon3` system, Figure 6.5 clearly shows that column-wise and row-wise rank checks put a lot of stress on the computational complexity of the algorithm. Degree block-wise rank checks are clearly much faster, because less rank checks are necessary (it also provides a better numerical robustness). Furthermore, the column space based approach suffers from two additional difficulties: (i) checking the rank structure column-wise in the column space takes more time than checking the rank structure row-wise in the right null space, due to the larger size of the involved matrices, even though the number of checks is the same, and (ii) shifting in the column space takes more time than in the right null space because of the computation of a backward QR decomposition for every shift. Moreover, from Figure 6.5, it is also clearly visible that using recursive and sparse techniques further reduces the computation time. Similar conclusions can be drawn from the second problem, the slightly more difficult `conform` system, where the computation time of the column space based approach explodes, due to the column-wise rank checks on the large Macaulay matrix (Figure 6.6).

We can also compare the different sets of options when using MacaulayLab to solve MEPs. Starting with the `h2f5` problem, we observe a similar behavior as for the polynomial systems. We notice in Figure 6.7 that most of the computation time of the column space based approach is due to the rank checks. Also for the row-wise null space based approach, the rank checks determine a major part of the computation time. Note that for this example, we have to set an additional option, namely `posdim = true`, since this MEP has a positive-dimensional solution set at infinity. This amplifies the dominant factor of the rank checks even more, since for problems with a positive-dimensional solution set, the algorithm needs to perform the rank checks for every iteration. When we consider the `realization` problem from the database with a random data vector of $N = 4$ data points and $n = 2$ parameters, this behavior is even more outspoken (Figure 6.8).

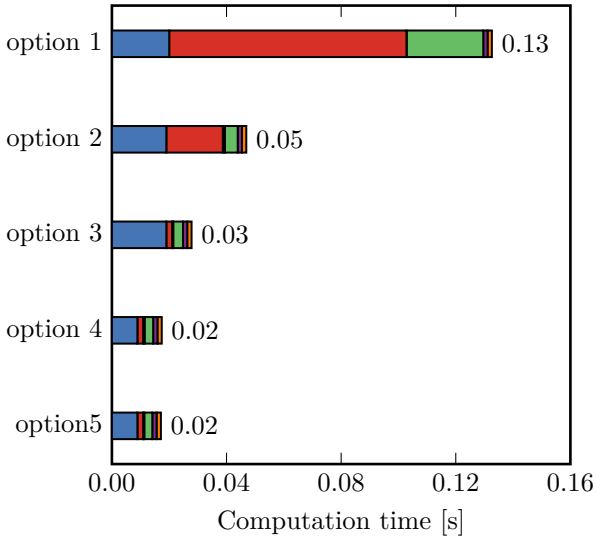


Figure 6.5. Computation time to solve the `noon3` system via the different sets of options (Table 6.3). All computation times are averaged over 30 experiments. The different sub-timings of the implementation are: enlarge the solution subspace (), check the rank structure (), perform the column compression (), shift in the subspace (), cluster the affine solutions (), and evaluate the affine solutions ().

6.6.2 Other polynomial system solvers

Now, we compare the default MacaulayLab solution approach to solve systems of multivariate polynomial equations (i.e., option 5 from Table 6.3) with some other solvers:

- RootFinding is a naive implementation of the null space based Macaulay matrix approach. It uses, for example, row-wise rank checks.
- PNLA-QR is the default implementation of the PNLA toolbox. It uses a sparse QR decomposition to compute a basis matrix of the right null space.
- PNLA-SVD is an adaptation of PNLA-QR, in which we have replaced all the sparse QR decompositions by the numerically more reliable singular value decomposition.

These solvers are included in the comparison because they are also implemented in Matlab, allowing for a fair comparison with MacaulayLab. We tested all algorithms on five different polynomial systems from the database, and the results are presented in Table 6.4 and Figure 6.9

Across all systems, except for `cyclic5` and `katsura7`, MacaulayLab is the fastest solver, consistently providing the solutions with the smallest residual errors (at least two orders of magnitude smaller than the other solvers). For

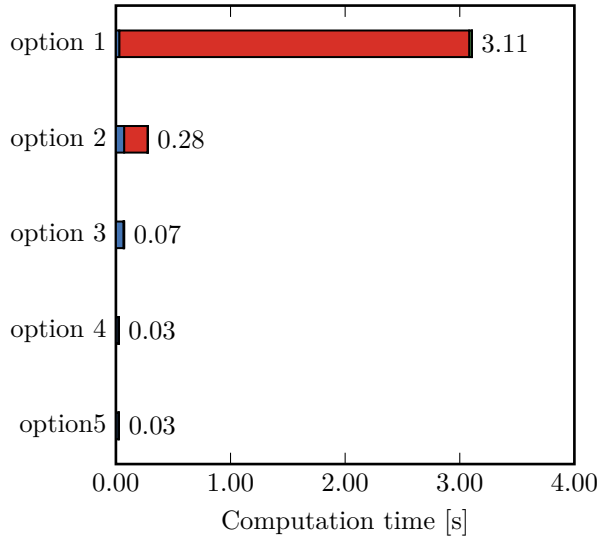


Figure 6.6. Computation time to solve the `conform` system via the different sets of options (Table 6.3). All computation times are averaged over 30 experiments. The different sub-timings of the implementation are: enlarge the solution subspace (■), check the rank structure (■), perform the column compression (■), shift in the subspace (■), cluster the affine solutions (■), and evaluate the affine solutions (■).

two systems, `cyclic5` and `katsura7`, PNLA-QR seems to be a faster alternative than MacaulayLab. However, PNLA-QR does not provide the correct solutions for these problems. Furthermore, both PNLA-SVD and PNLA-QR have difficulties with the positive-dimensional solution space at infinity of `walsh`, while MacaulayLab solves it quite easily (RootFinding only finds the solutions of this system when we supply the number of affine solutions in advance). The computation speed of PNLA-QR suggests an interesting direction for future research: "Why not incorporate sparse QR decompositions in the implementation of MacaulayLab?"

For completeness, we also compare MacaulayLab with PHClab, a Matlab interface to PHCpack, and Maple. PHCpack is a pre-compiled Ada general-purpose solver that uses homotopy continuation. In Maple, we use a naive Gröbner basis implementation to solve the polynomial system in exact arithmetic. The results of this comparison are summarized in Table 6.5. It is important to acknowledge that this comparison is not entirely fair, because the nature of the three toolboxes is quite different.

6.6.3 Other multiparameter eigenvalue solvers

We also compare the block Macaulay matrix approach to solve MEPs in MacaulayLab with the two approaches offered by the MultiParEig toolbox (cf., they are based on the transformations described in Section 3.2.2):

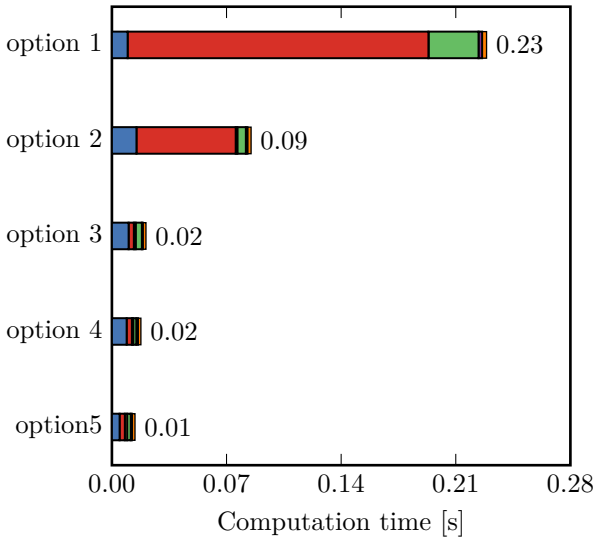


Figure 6.7. Computation time to solve the `h2f5` MEP via the different sets of options (Table 6.3). All computation times are averaged over 30 experiments. The different sub-timings of the implementation are: enlarge the solution subspace (■), check the rank structure (■), perform the column compression (■), shift in the subspace (■), cluster the affine solutions (■), and evaluate the affine solutions (■).

Table 6.4. Comparison of the computation time to solve five different systems of multivariate polynomial equations via MacaulayLab, RootFinding, PNLA-SVD, and PNLA-QR. The most important properties of the systems are summarized in Table 6.1. All computation times are averaged over 30 experiments.

problem	MacaulayLab	RootFinding	PNLA-SVD	PNLA-QR
`conform`	0.05 s	1.31 s	1.56 s	1.37 s
`batselier5`	51.42 s	$> 10\,000.00$ s*	7726.71 s	7155.19 s
`cyclic5`	15.04 s	74.82 s	23.62 s	10.13 s†
`walsh`	0.50 s	3078.27 s	1.37 s†	0.99 s†
`katsura7`	192.91 s	1447.34 s	235.08 s	17.57 s†

† The solver fails to correctly obtain the (affine) solutions of the system.

* Execution was aborted after 10 000.00 s.

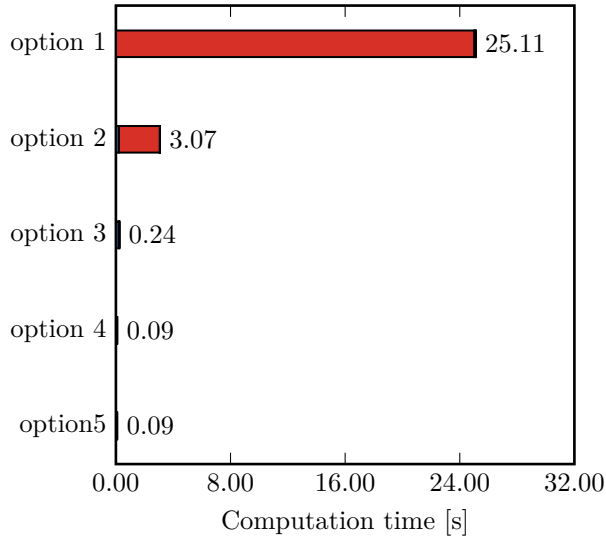


Figure 6.8. Computation time to solve the `realization` MEP via the different set of options (Table 6.3). We consider a data vector $\boldsymbol{y} \in \mathbb{R}^{4\times 1}$ and $n = 2$ parameters. All computation times are averaged over 30 experiments. The different sub-timings of the implementation are: enlarge the solution subspace (■), check the rank structure (■), perform the column compression (■), shift in the subspace (■), cluster the affine solutions (■), and evaluate the affine solutions (■).

- MultiParEig-CMP transforms the rectangular MEP into a related system of coupled rectangular GEPs via Kronecker products of the rectangular coefficient matrices. The large, rectangular Kronecker matrices are compressed into small, square matrices via the pre-multiplication and post-multiplication with compression matrices to create a system of square, coupled GEPs that contain the same solutions as the rectangular MEP.
- MultiParEig-MEP uses randomized sketching (i.e., it multiplies the rectangular coefficient matrices with n random projection matrices to create n square matrix pencils) to create a square MEP from the supplied problem and solves the resulting square problem via its associated system of coupled GEPs, after which the spurious solutions generated by the transformation are pruned (based on the residual error of the solution).

While for moderate-sized problems both approaches are comparable, the MultiParEig approaches are clearly faster when the matrix sizes grow (Table 6.6 and Figure 6.10). Figure 6.11 seems to suggest that the trend in the computational complexity for a fixed number of eigenvalue parameters is similar. However, when the number of eigenvalue parameters grows, MultiParEig-CMP and MultiParEig-MEP have more difficulties to solve the problem. For example, MultiParEig-CMP and MultiParEig-MEP both fail to solve the 4-parameter eigenvalue problem `h2fourdisk`, due to memory constraints:

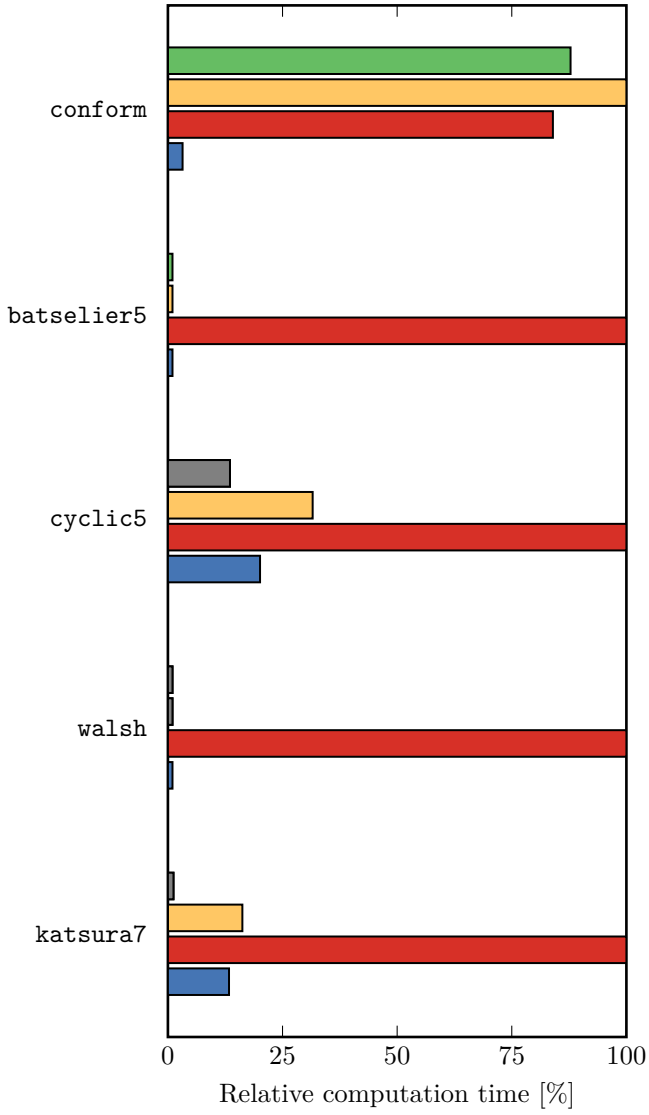


Figure 6.9. Comparison of the relative computation time to solve five different systems of multivariate polynomial equations via MacaulayLab(■), RootFinding (■), PNLA-SVD (■), and PNLA-QR (■). The most important properties of the systems are summarized in Table 6.1. All computation times are averaged over 30 experiments. While Table 6.4 contains the exact computation times per system and solver, this comparison visualizes the relative computation time with respect to the slowest solver (which is always set at 100%). The PNLA solver fails to correctly obtain the (affine) solutions of some systems (■).

Table 6.5♣. Comparison of the computation time to solve ten different systems of multivariate polynomial equations via MacaulayLab, PHClab, and Maple. The most important properties of the first five systems are summarized in Table 6.1, while the other five systems are random polynomial systems `randomsystem(s,d,n)` with s polynomials of total degree d in n variables. All computation times are averaged over 30 experiments.

problem	MacaulayLab	PHClab	Maple
`batselier5`	17.22 s	3.50 s	1.58 s
`dreesen10`	0.09 s	0.23 s	0.55 s
`cyclic5`	4.85 s	0.43 s	4.92 s
`walsh`	0.50 s	0.24 s	0.57 s
`katsura7`	32.02 s	2.77 s	125.94 s
`randomsystem(2,10,2)`	0.16 s	0.78 s	0.24 s
`randomsystem(2,20,2)`	0.09 s	0.23 s	4.79 s
`randomsystem(3,10,3)`	15.90 s	48.21 s	1901.23 s
`randomsystem(4,5,4)`	13.90 s	21.26 s	2801.98 s
`randomsystem(5,3,5)`	6.85 s	5.22 s	330.11 s

- For MultiParEig-CMP, setting up the Kronecker products of the rectangular coefficient matrices results in very large matrices (because the dimension of a Kronecker product of n matrices is the product of all n dimensions).

- For MultiParEig-MEP, the bottleneck is solving the associated set of coupled GEPs of a square MEP with a high number of eigenvalues.

Another motivation for this claim is the example of solving a small linear 5-parameter eigenvalue problem with 11×7 coefficient matrices, which is not possible with MultiParEig-CMP, due to the fact that the Kronecker operator determinants generate matrices that require more than 20 GB of memory. Figure 6.12 shows how, for small n-parameter eigenvalue problems with $(5 + n - 1) \times 5$ coefficient matrices, the memory constraints trouble both approaches from the MultiParEig toolbox.

6.7 Conclusion

This chapter emphasized the versatility and potential of MacaulayLab as a toolbox for solving systems of multivariate polynomial equations and rectangular MEPs. MacaulayLab stands out as a general-purpose polynomial system solver, capable of dealing with systems in different polynomial bases, overdetermined systems, and systems with positive-dimensional solution sets at infinity. While homotopy continuation may offer faster solution algorithms, these methods may suffer from ill-conditioning and (only) deal with square systems, leaving a niche that can be filled by the Macaulay matrix algorithms from this

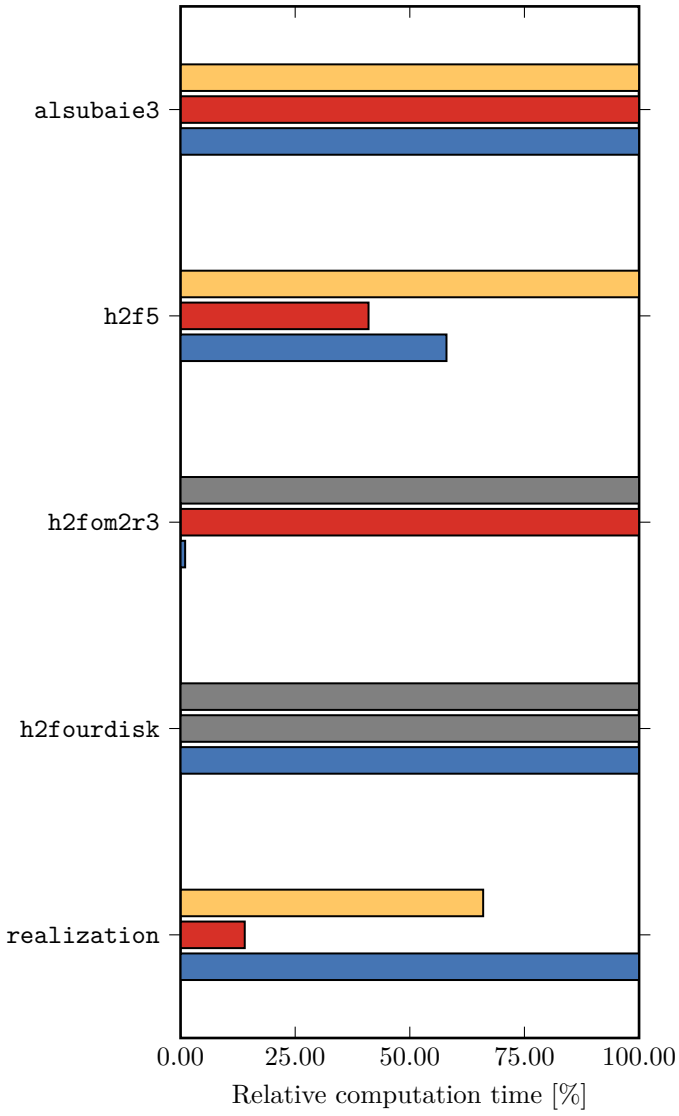


Figure 6.10. Comparison of the relative computation time to solve five different MEPs via MacaulayLab(■), MultiParEig-CMP (■), and MultiParEig-MEP (■). The most important properties of the MEPs are summarized in Table 6.2. All computation times are averaged over 30 experiments. While Table 6.6 contains the exact computation times per problem and solver, this comparison visualizes the relative computation time with respect to the slowest solver (which is always set at 100%). MultiParEig fails to solve the `h2fourdisk` problem, due to memory constraints (■).

Table 6.6. Comparison of the computation time to solve five different rectangular MEPs via MacaulayLab, MultiParEig-CMP, and MultiParEig-MEP. The most important properties of the MEPs are summarized in Table 6.2. All computation times are averaged over 30 experiments. For the `realization` problem, we consider a data vector $\boldsymbol{y} \in \mathbb{R}^{6\times 1}$ and $n = 2$ parameters.

problem	MacaulayLab	MultiParEig-CMP	MultiParEig-MEP
`alsubaie3`	< 0.01 s	< 0.01 s	< 0.01 s
`h2f5`	0.01 s	0.01 s	0.02 s
`h2fom2r3`	2.54 s	247.81 s	n.a.‡
`h2fourdisk`	3.79 s	n.a.‡	n.a.‡
`realization`	5.56 s	0.75 s§	3.55 s

‡ This solver fails to solve the `h2fourdisk` problem, due to memory constraints.

§ MultiParEig also contains a more efficient approach that exploits the structure of this particular MEP.

(and other) toolbox(es), particularly for overdetermined systems of multivariate polynomial equations. In comparison to other Matlab toolboxes that use the Macaulay matrix, MacaulayLab is a clear improvement: while being built modular and offering many different solution approaches and options, it is computationally more efficient and numerically more robust than its historical predecessors RootFinding and PNLA.

By introducing the block Macaulay matrix as natural extension of the (scalar) Macaulay matrix, MacaulayLab becomes the first dedicated toolbox to tackle rectangular MEPs, while maintaining the same flexibility as when solving systems of multivariate polynomial equations. Furthermore, the solution approaches in the toolbox can deal with positive-dimensional solution sets at infinity and work with different shift polynomials. It is important to stress that MacaulayLab is designed to be independent of the polynomial basis and monomial ordering, which provides an easy way of solving problems that are given in a different representation (without needing to change the representation first). This can be very useful in applications where the results depend on the chosen polynomial basis or monomial ordering. Additionally, the MacaulayLab toolbox also contains a database with many test problems. We believe that the inclusion of this comprehensive set of test problems could prove to be highly valuable for its users.

Motivational example. With MacaulayLab, it has become very easy to tackle the first-order ARMA identification problem. The globally optimal parameters can be obtained from the polynomial system by running `macaulaylab` with `options.posdim = true`, because the polynomial system has a positive-dimensional solution set at infinity (`dend = 50` gives an upper bound to the degree of the Macaulay matrix and `y` is the sequence of data points):

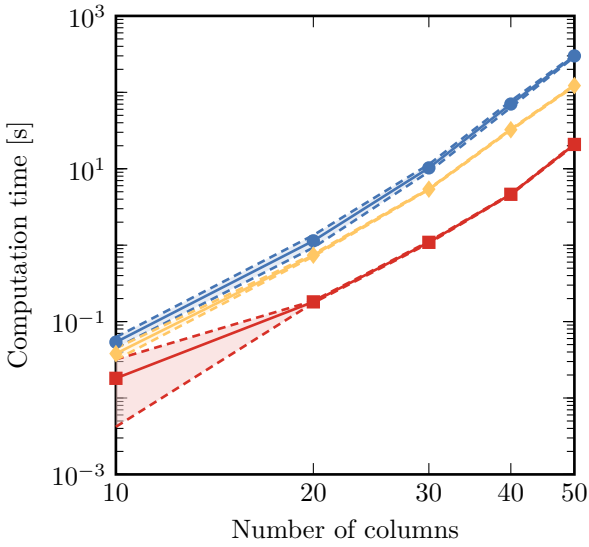


Figure 6.11. Computation time to solve a linear 2-parameter eigenvalue problem via MacaulayLab (—●—), MultiParEig-CMP (—■—), and MultiParEig-MEP (—◆—). All computation times are averaged over 30 experiments (dashed lines indicate one standard deviation). We notice that the toolboxes have a similar complexity, but the two approaches from MultiParEig are faster.

```
>> dend = 50; options = struct; options.posdim = true;
>> commonroots = macaulaylab(arma11eq(y),dend,options);
```

A similar approach can be used to solve the related quadratic rectangular two-parameter eigenvalue problem:

```
>> eigenvalues = macaulaylab(arma11(y),50,options);
```

Both `arma11eq` and `arma11` are part of the database with test problems that accompanies MacaulayLab.

Taking inspiration from other toolboxes and recent advances in the literature, it is evident that future releases of MacaulayLab will contain many additional features:

- We want to consider the sparsity in the involved matrices and investigate the advantages and disadvantages of working with the QR decomposition instead of the singular value decomposition.
- Also looking into the support of the polynomials and polynomial eigenvalue problems to reduce the number of columns in the (block) Macaulay

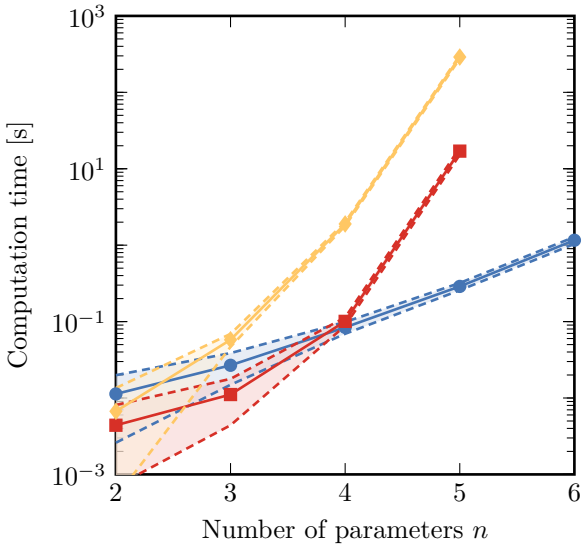


Figure 6.12. Computation time to solve a linear MEP via MacaulayLab (—●—), MultiParEig-CMP (—■—), and MultiParEig-MEP (—◆—). All computation times are averaged over 30 experiments. We increase the number of eigenvalue parameters n, and we notice that the algorithms of MultiParEig struggle to determine the solutions of a small linear n-parameter eigenvalue problems with $(5+n-1)\times 5$ coefficient matrices. Notice that both approaches of MultiParEig fail to compute the solutions of an 6-parameter eigenvalue problem with 10×5 coefficient matrices, due to memory constraints.

matrix sounds promising [30, 32].

- Performing the rank checks is an important step in the (block) Macaulay matrix approach. A lot of care is necessary with these rank decisions. In one of the next releases, more support to take the correct rank decisions (i.e., different types of rank decisions, visual aids, etc.) will be added.
- Subspace methods are also anticipated to be included as an extension to the block Macaulay matrix algorithms in one of the upcoming releases.

PART

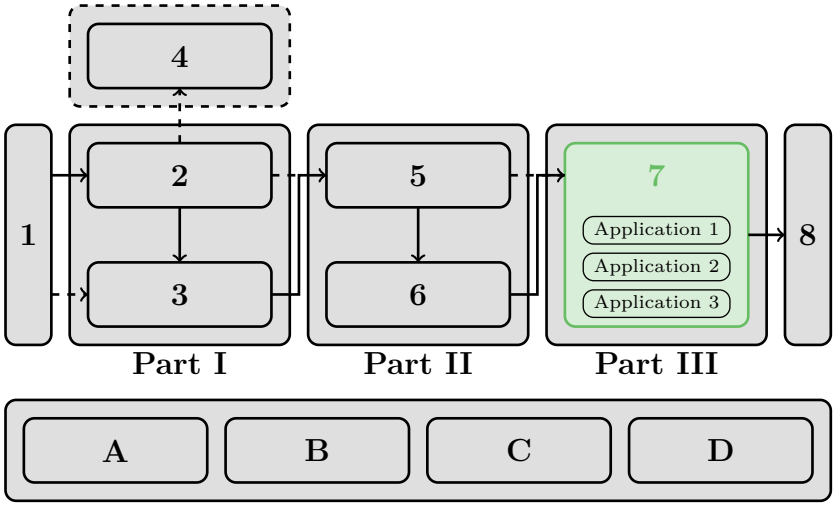
III

Applications

Applications in Systems Theory

The different algorithms described in this dissertation share a common characteristic: their development is driven by a practical need. In this chapter, we address the origin for this need and provide motivational examples from diverse fields that can be addressed using the algorithms proposed in this text. We focus on three key examples from systems theory: multivariate polynomial optimization in complex variables, $\mathcal{H}_2$-norm model order reduction, and least-squares identification of autoregressive moving-average models. The goal is to find globally optimal solutions for each of these three key examples.

Although each of the discussed globally optimal methodologies is a valuable contribution to its respective research area, we present them more as a detailed illustration of what we could do with the algorithms of this dissertation. We use the MacaulayLab toolbox from Chapter 6 and highlight, for example, how the recursive algorithms in Chapter 5 speed up the computations. It is important to note that the potential application area extends well beyond the examples presented here.



Contributions. We show how the algorithms presented in the previous chapters can be useful in a practical setting. Next to their illustrative nature, the proposed globally optimal methodologies to tackle the three key examples in this chapter are also valuable contributions to their respective research areas.

Related articles. Three sections of this chapter (with applications from systems theory) consist of adapted versions of [3, 259, 263]. For [259, 263], the candidate was the main author of the original articles, developed the theoretical contributions, and implemented the accompanying code and experiments. The candidate developed a numerical solution algorithm and implemented the numerical experiments for [3]. He also assisted with writing the text of that article. In order to make the text more consistent, several parts of the original articles are re-written and some sections are removed from this chapter to avoid redundancy (as these sections are present in a more thorough form in the previous chapters). Especially, the notation and terminology has been made uniform with the rest of the dissertation.

Outline. This chapter revolves around three key examples. However, we first take into account the broader context by providing an introduction in Section 7.1 and highlighting possible application areas in Section 7.2. Subsequently, Section 7.3 contains the first key example, namely the multivariate polynomial optimization problem in complex variables. In Section 7.4, we show how the algorithms presented in this dissertation can also be used to tackle the $\mathcal{H}_2$-norm model order reduction problem. Afterwards, in Section 7.5, we demonstrate how the globally optimal least-squares identification of autoregressive moving-average models can be rephrased as a rectangular multiparameter eigenvalue problem. Finally, we conclude this chapter in Section 7.6, where we discuss some potential future research avenues.

7.1 Introduction

In the previous chapters, we have developed an extensive set of algorithms, collected in the Matlab toolbox MacaulayLab, to solve systems of multivariate polynomial equations and to tackle rectangular multiparameter eigenvalue problems (MEPs). While the development of those algorithms are very interesting in their own rights, the reason for their development originates from a practical necessity: the desire to solve practical applications that can be phrased as systems of multivariate polynomial equations or rectangular MEPs. In this chapter, we focus on these practical applications and provide motivational examples from various fields that can be addressed using the algorithms proposed in this text.

At the hearth of many engineering applications is the goal to optimize a certain metric. Frequently, it is possible to express the optimization of this metric as a (multivariate) polynomial optimization problem, which is an optimization problem in which both the cost function and the constraints are (multivariate) polynomials. Many of the applications that we encounter in this chapter are multivariate polynomial optimization problems applied to a specific problem context. The first-order necessary conditions for optimality (i.e., the Karush–Kuhn–Tucker conditions) of such multivariate polynomial optimization problems correspond to a system of multivariate polynomial equations when we only consider equality constraints. The solutions of this multivariate polynomial system are the stationary points of the original cost function under the equality constraints [190].

Recently, our research group has shown that for some multivariate polynomial optimization problems, the resulting system of multivariate polynomial equations is essentially an MEP. The eigenvalues of these MEPs correspond to the stationary points of the related cost function. Hence, the globally optimal solution(s) of the original optimization problem can be found via the eigenvalues of the MEP. During this research, we have focussed on three key examples from systems theory that could be phrased as an MEP:

- the multivariate polynomial optimization problem in complex variables via Wirtinger derivatives;
- the $\mathcal{H}_2$-norm model order reduction problem for single-input/single-output (SISO) linear time-invariant (LTI) models; and
- the least-squares parameter identification of autoregressive moving-average (ARMA) models.

The (block) Macaulay matrix algorithms are exactly developed in the context of these particular problems. Although each of the key examples is a valuable contribution to its respective research area, we present them more as a detailed illustration of what we could do with the algorithms of this dissertation.

> **Motivational example.** In tackling the first-order ARMA model identification problem as a rectangular MEP, one important question has not

been answered yet: "What are the coefficient matrices of the quadratic two-parameter eigenvalue problem?" This chapter contains the answer to that question. In Section 7.5, we show how to derive the (coefficient matrices of the) rectangular MEP that solves the optimization problem (1.2).

7.2 Examples of possible application areas

Before delving into the three problems from systems theory, we provide some additional examples that can be formulated as a system of multivariate polynomial equations or (multiparameter) eigenvalue problem.

7.2.1 Common roots of systems of polynomials

Polynomials, both univariate and multivariate, are powerful tools to model problems from various origins mathematically. Because of their modeling capabilities, polynomials, and in particular systems of (multivariate) polynomial equations, have extensive applications in diverse areas of science and engineering, for example, in robotics [222, 269], game theory [67, 222], computational chemistry and biology [60, 76, 86], computer vision [145, 146], system identification [25, 158, 259], and model order reduction [3, 148]. We give some examples below, but more applications can be found in books like [63, 180, 222].

Robotics and kinematics. An important field that contains many systems of multivariate polynomial equations is robotics (and kinematics). The polynomials that appear are often very sparse and structured [85, 222]. Figure 7.1 shows, for example, how a planar robot arm with a fixed shoulder at the origin and two arm segments leads to a system of two bivariate polynomial equations via Pythagoras' theorem,

$$\begin{cases} p_1(\boldsymbol{x}) = x_1^2 + x_2^2 - l_1^2 = 0, \\ p_2(\boldsymbol{x}) = (a - x_1)^2 + (b - x_2)^2 - l_2^2 = 0, \end{cases} \tag{7.1}$$

describing the possible positions of the robot's elbow. It is easy to image that more complicated robots and kinematic problems lead quickly to more involved polynomial systems. Sommese and Wampler [222] discuss many examples of robots in a polynomial system solving context.

Nash equilibria. In game theory, the Nash equilibrium[1] is the most common way to define the solution of a non-cooperative game involving two or more players. In a Nash equilibrium, each of the players is assumed to know the equilibrium strategies of the other players, and no player has anything to gain

[1]The principle of Nash equilibrium dates back to the time of Antoine Cournot, who applied it in 1838 to competing firms choosing outputs, but it is named after the American mathematician John Nash, who received a Nobel Prize in Economics in 1994 [140, 144].

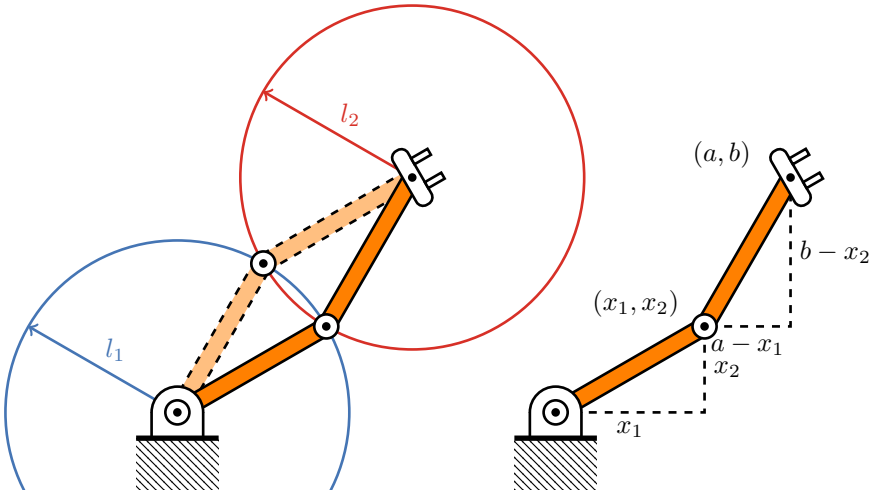


Figure 7.1. Possible configurations of the elbow (x_1, x_2) of a planar robot arm with a fixed shoulder in the origin and two arm segments. Finding the possible configurations of the robot's elbow corresponds to solving a system of bivariate polynomial equations (7.1) that originates from applying Pythagoras' theorem twice.

by changing only one's own strategy. The game is played multiple times. If one player observes that in the last round a change in strategy would have earned him/her a higher payoff, then he/she is likely to change his/her play in the next round. An equilibrium occurs when every player finds that there is no unilateral change of strategy that would have increased his/her payoff. Nash equilibria of finite games (i.e., with finite numbers of players, each with a finite number of pure strategies) can be characterized as solutions to systems of polynomial equalities subject to some inequalities [67].

Chemical reactor networks. A simple, but meaningful example of a chemical reaction network is the T-cell signal transduction model proposed by the immunologist McKeithan [168] and tackled in the context of algebraic geometry by Dickenstein [76]. The main task of the immune system is to recognize that a strange body has entered the organism. T-cell receptors bind to both self-antigens and foreign antigens and the dynamical features of this model give a possible explanation of how T-cells can be sensitive and specific in recognizing self versus foreign antigens. In its simplest case, the network of reactions is given by

$$\begin{array}{ccc} & A+B & \\ {\scriptstyle \kappa_{31}}\nearrow & & \searrow\nearrow {\scriptstyle \kappa_{21}} \\ & {\scriptstyle \kappa_{12}} & \\ D & \xleftarrow[\kappa_{23}]{} & C, \end{array} \tag{7.2}$$

where A denotes the T-cell receptor protein, B denotes the major histocompatibility protein complex of the antigen-presenting cell, C denotes the biochemical species A bound to species B, and D denotes an activated (phosphorylated) form of C. The binding of A and B forms C, which undergoes a modification into its activated form D before transmitting a signal. This example is a (bio)chemical reaction network with 3 complexes ($A + B$, C, and D) and 4 reactions (the four arrows, each with a reaction rate constants $\kappa_{ij} \in \mathbb{R}_{>0}$). If we denote the time dependent concentrations of the species by

$$\boldsymbol{x} = (x_A(t), x_B(t), x_C(t), x_D(t)), \tag{7.3}$$

then the law of mass action gives us a system of four multivariate polynomial equations:

$$\begin{cases} p_1(\boldsymbol{x}) = \dfrac{dx_A(t)}{dt} = -\kappa_{12}x_A(t)x_B(t) + \kappa_{21}x_C(t) + \kappa_{31}x_D(t) = 0, \\ p_2(\boldsymbol{x}) = \dfrac{dx_B(t)}{dt} = -\kappa_{12}x_A(t)x_B(t) + \kappa_{21}x_C(t) + \kappa_{31}x_D(t) = 0, \\ p_3(\boldsymbol{x}) = \dfrac{dx_C(t)}{dt} = \kappa_{12}x_A(t)x_B(t) - \kappa_{21}x_C(t) - \kappa_{23}x_C(t) = 0, \\ p_4(\boldsymbol{x}) = \dfrac{dx_D(t)}{dt} = \kappa_{23}x_C(t) - \kappa_{31}x_D(t) = 0. \end{cases} \tag{7.4}$$

The set $\left\{\boldsymbol{x} \in \mathbb{R}^4_{>0} : p_1(\boldsymbol{x}) = p_2(\boldsymbol{x}) = p_3(\boldsymbol{x}) = p_4(\boldsymbol{x}) = 0\right\}$ is called the steady state variety of the (bio)chemical reaction network. By the structure of the equations, for given initial concentrations, the solution $\boldsymbol{x}$ can not leave the stoichiometric compatibility class, which is an affine subspace $\mathbb{R}^4_{>0}$. Adding the affine equations of the stoichiometric compatibility class to the polynomial system in (7.4), we get the set of all candidate solutions. In this example, we are only interested in the positive solutions, rather than the real solutions. Solving these algebraic relations, i.e., the system of multivariate polynomial equations, gives us the possible stationary concentrations of the different chemical species. Taking advantage of the algebraic structure of these networks has led to advances in understanding their dynamical behavior. We refer the interested reader to [60, 76] and references therein.

Computer vision. Computer vision problems are another type of problems that often result in systems of multivariate polynomial equations [145, 146]. An important example is estimating internal calibration parameters of a camera from point correspondences in a sequence of (noisy) images [79, 204]. Every such point correspondence imposes an algebraic relation on the parameters that are to be estimated; hence, the solution can be found in polynomial system solving. For example, a camera is positioned at the unknown position P and images taken of four calibration points A, B, C, and D put constraints upon the unknown camera distances x_i, for $i = 1, \ldots 4$ (Figure 7.2). We now have

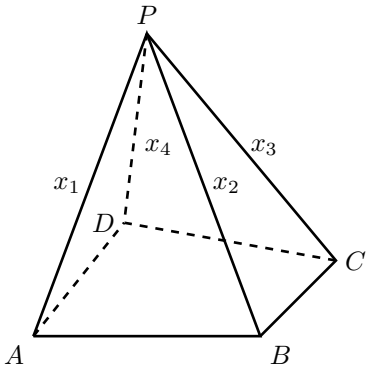


Figure 7.2. Simple instance of a camera pose estimation problem. A camera is centered at an unknown position P. Then, a set of images is taken of four calibration points A, B, C, and D, each of which leading to a constraint on the unknown camera distances x_i, for $i = 1, \ldots, 4$.

an over-constrained system of $s = 6$ equations in $n = 4$ variables:

$$\begin{cases} x_1^2 + x_2^2 - 2\cos(APB)x_1x_2 - \|AB\|^2 \approx 0, \\ x_1^2 + x_3^2 - 2\cos(APC)x_1x_3 - \|AC\|^2 \approx 0, \\ x_2^2 + x_3^2 - 2\cos(BPC)x_2x_3 - \|BC\|^2 \approx 0, \\ x_1^2 + x_4^2 - 2\cos(CPD)x_1x_4 - \|AD\|^2 \approx 0, \\ x_4^2 + x_3^2 - 2\cos(APD)x_1x_2 - \|CD\|^2 \approx 0, \\ x_2^2 + x_4^2 - 2\cos(BPD)x_2x_4 - \|BD\|^2 \approx 0. \end{cases} \tag{7.5}$$

Solving this over-determined system of multivariate polynomial equations yields the unknown camera distances x_i.

7.2.2 Eigenvalues and eigenvectors

Eigenvalue problems, whether involving a single spectral parameter or multiple spectral parameters, are equally pervasive in applications. The significance of eigenvalues and eigenvectors lies in their ability to capture the inherent dynamics of many natural and scientific phenomena, which elucidates their prevalence in numerous domains. Consequently, eigenvalue problems are indispensable for comprehending and modeling the underlying behavior of systems, for example, when considering problems in systems and control [128–130], vibration analysis [149, 245], partial differential equations [96, 200], system identification [70, 71, 259], and model order reduction [3, 208].

Systems and control. Eigenvalues are the cornerstone of systems and control. When we consider LTI dynamical systems, eigenvalues characterize key properties of a system, like stability, controllability, and observability [128].

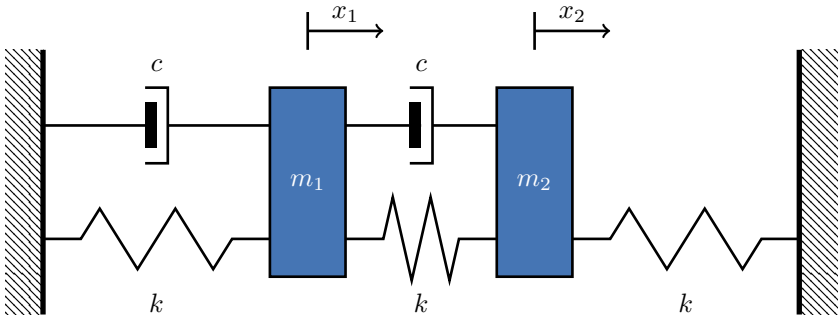


Figure 7.3. Visualization of a mass-spring system with two masses m_1 and m_2. The positions of the two masses, x_1 and x_2, can be found by solving the corresponding quadratic eigenvalue problem in (7.6). The parameters c and k are the damping and stiffness coefficient, respectively.

Standard eigenvalue problems (SEPs) and generalized eigenvalue problems (GEPs) also arise in the steady-state solutions to linear-quadratic regulators and Kalman[2] filtering problems [129].

Vibration analysis. When considering vibration problems, eigenvalues immediately enter the picture. These vibration problems can be interpreted in the broad definition of the word: vibrations in mechanical structures, acoustic systems, electrical circuits, and fluid mechanics [245]. For example, the vibration analysis of the simple mass-spring system in Figure 7.3 can be solved by computing the eigenvalues of the corresponding quadratic eigenvalue problem [149]:

$$\left(\begin{bmatrix} 2k & -k \\ -k & 2k \end{bmatrix} + \begin{bmatrix} 2c & -c \\ -c & c \end{bmatrix} \lambda + \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \lambda^2 \right) \boldsymbol{z} = \mathbf{0}, \tag{7.6}$$

where m_1 and m_2 are the two masses, c is the damping coefficient, and k is the stiffness coefficient. More advanced vibration problems, like the aeroelastic flutter problem [203], can be described via the introduction of multiple spectral parameters.

Partial differential equations. A key motivation for multiparameter spectral theory is solving boundary-value problems for partial differential equations via the method of separation of variables. A discretization via, e.g., spectral collocation methods, finite differences, or finite elements, leads to a solvable (algebraic) square MEP. Even more, problems like Mathieu's system are in the literature often used as a motivation for the introduction of MEPs, for example, in [267]. The spectral parameters arise as separation constants when the separation of variables technique is used. Although Mathieu's system is often

[2]Kalman [130] showed that the control and filtering problem are related to each other; hence, the steady-state solutions to both problems follow from a Riccati equation, which can be tackled by solving a GEP, as described by Van Dooren [250].

used as motivating example, it was not until [96] that it was shown that solving the corresponding two-parameter eigenvalue problem approach has certain numerical advantages when tackling this type of problems. More examples can be found in [200, 276].

Identifying Hopf bifurcations. A Hopf bifurcation is a critical point where a dynamical system's stability changes and periodicity arises. Hopf bifurcations occur in some very famous dynamical systems, e.g., the Lotka–Volterra system, the Lorenz attractor, and the Brusselator. The identification of instability in large-scale dynamical systems caused by a Hopf bifurcation is difficult in practice, because it requires the identification of the right-most pair of complex eigenvalues of a large, sparse GEP [84]. When looking for these Hopf bifurcations, the problem can also be reduced to solving a square two-parameter eigenvalue problem [169, 170].

Heine–Stieltjes problems. The Heine–Stieltjes spectral problem is a generalization of the SEP [41, 215], where the goal is to obtain the polynomial solutions $(\boldsymbol{z}, f)$ such that

$$T(f) = \boldsymbol{z} f, \tag{7.7}$$

where T is a certain second-order differential operator with polynomial coefficients. The multiparameter generalization of this problem can be phrased as a rectangular MEP [216].

7.3 Complex multivariate optimization

Complex-valued signals arise in many areas of science and engineering, like communications, systems theory, oceanography, geophysics, optics, and electromagnetics [1, 225]. Especially in signal processing, one often encounters (nonlinear) functions in complex variables [53], for example, transfer functions of LTI models. An important issue when working with complex-valued signals and complex variables is related to (nonlinear) optimization. Most of the optimization literature deals with real variables only, seemingly suggesting that complex variables are not encountered in practice. However, optimization problems in complex variables appear in various applications, e.g., filter design [1, 42, 53, 141], system identification [70], blind source separation [1], tensor decomposition [223–225], parameter estimation [1], and nonlinear electrical circuit simulation [225].

Cost functions of optimization problems in complex variables are real-valued: it makes no sense to optimize a complex-valued cost function, because the field of complex numbers is not (totally) ordered. So, from an application point of view, these real-valued functions are exactly the kind of cost functions that we expect to encounter. However, real-valued cost functions in complex variables are necessarily non-holomorphic (i.e., the complex generalization of non-analytic) [225]. They have no complex derivatives. An optimization problem in complex variables is typically tackled by reformulating the cost function as a function of the real and imaginary parts of the complex variables, so that

standard real optimization techniques can be used. Wirtinger calculus provides a more elegant solution by relaxing the definition of differentiability and defining a general framework that includes holomorphic functions as a special case [1, 53, 141]. The development of Wirtinger calculus[3] by the Austrian mathematician Wirtinger [280] dates back to 1927. It was rediscovered in 1983, without any reference to Wirtinger, by Brandwood [42]. The advantage of Wirtinger calculus is that the expressions do not become unnecessarily complicated and the derivations are rather similar to the real situation.

Sorber et al. [223, 224] have highlighted an interesting relation between univariate polynomial optimization in one complex variable and the polynomial eigenvalue problem (PEP). The first-order necessary conditions for optimality of the real-valued univariate polynomial cost function obtained via Wirtinger calculus yield a system of two polyanalytic polynomials in the complex variable and its complex conjugate. An elimination of this complex conjugate variable, via the Sylvester matrix, results in a PEP that can be solved with standard techniques from numerical linear algebra. In this section, we extend this relation to the multivariate case: we show that multivariate polynomial optimization problems in multiple complex variables lead to (rectangular[4]) MEPs. At least one of the eigenvalues of this MEP corresponds to the global solution of the optimization problem. This section has not the ambition to provide a competitive alternative with respect to the current state-of-the-art. Rather it serves a tutorial purpose, presenting a novel optimization approach in a didactic way. It highlights several interesting research avenues initiated by this reformulation (Section 7.6), while omitting technical derivations. Note that reformulating a multivariate polynomial optimization problem in real variables as an (one-parameter) eigenvalue problem is a well-established methodology, as discussed in the "historical and bibliographical notes" of this chapter.

In the remainder of this section, we first formulate the first key example mathematically (Section 7.3.1), before giving a short introduction to Wirtinger calculus (Section 7.3.2). Afterwards, we show how the first-order necessary conditions for optimality result in an MEP, via a special Macaulay matrix construction (Section 7.3.3). Finally, we discuss the concept of ghost solutions, which emerge in this globally optimal complex optimization approach (Section 7.3.4), and give a numerical example (Section 7.3.5).

7.3.1 Problem formulation

In this section, we deal with real-valued (multivariate) polynomial cost functions $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$ in n complex (decision) variables $\boldsymbol{z} \in \mathbb{C}^n$ and their complex conjugates $\bar{\boldsymbol{z}} \in \mathbb{C}^n$,

$$f : \mathbb{C}^n \to \mathbb{R} : \boldsymbol{z} \mapsto f(\boldsymbol{z}, \bar{\boldsymbol{z}}), \tag{7.8}$$

[3]The idea of using Wirtinger derivatives can be traced back to at least 1899, with Poincaré [202, 225]. The name *Wirtinger calculus* is especially present in the German literature, while in other sources one often reads $\mathbb{CR}$-*calculus*, referring to the fields $\mathbb{C}$ and $\mathbb{R}$ [141].

[4]In the remainder of this chapter, we no longer mention the qualification *rectangular* explicitly. We always consider rectangular MEPs, except when denoted otherwise.

where we express the dependency of the cost function on the complex variables $\boldsymbol{z}$ and their complex conjugates $\bar{\boldsymbol{z}}$ explicitly to highlight that the polynomial is real-valued (Section 7.3.2). The problems tackled in Sections 7.4 and 7.5 are, for example, multivariate polynomial optimization problems with a real-valued cost function in complex variables (despite the fact that we are only interested in the real-valued stationary points). We consider, primarily, the unconstrained minimization problem, i.e.,

$$\min_{\boldsymbol{z}} f(\boldsymbol{z}, \bar{\boldsymbol{z}}), \tag{7.9}$$

but adaptations to maximization or constrained optimization are straightforward via the Lagrangian (Example 7.1). A prototypical problem with a real-valued polynomial cost function is the minimization of the squared Frobenius-norm of a matrix[5] polynomial $\boldsymbol{\mathcal{F}}(\boldsymbol{z}, \bar{\boldsymbol{z}})$,

$$\boldsymbol{\mathcal{F}} : \mathbb{C}^n \to \mathbb{C}^{m_1 \times m_2} : \boldsymbol{z} \mapsto \boldsymbol{\mathcal{F}}(\boldsymbol{z}, \bar{\boldsymbol{z}}), \tag{7.10}$$

that maps n complex variables $\boldsymbol{z}$ and their complex conjugates $\bar{\boldsymbol{z}}$ onto $m_1 m_2$ function values, i.e.,

$$\min_{\boldsymbol{z}} \|\boldsymbol{\mathcal{F}}(\boldsymbol{z}, \bar{\boldsymbol{z}})\|_{\mathrm{F}}^2, \tag{7.11}$$

which is also known as the complex nonlinear least-squares optimization problem. Because of the imposed Frobenius-norm, the cost function in (7.11) is a real-valued polynomial in $\boldsymbol{z}$.

7.3.2 Wirtinger derivatives

Before we tackle (7.9), we need to take a closer look at the implications of differentiation in the complex domain. Consider a multivariate complex-valued function $f(\boldsymbol{z})$ in n complex variables $\boldsymbol{z} \in \mathbb{C}^n$,

$$f : \mathbb{C}^n \to \mathbb{C} : \boldsymbol{z} = \boldsymbol{x} + \mathrm{i}\boldsymbol{y} \mapsto f(\boldsymbol{z}) = u(\boldsymbol{x}, \boldsymbol{y}) + \mathrm{i}v(\boldsymbol{x}, \boldsymbol{y}), \tag{7.12}$$

where $u(\boldsymbol{x}, \boldsymbol{y})$ and $v(\boldsymbol{x}, \boldsymbol{y})$ are ordinary real-valued functions in $2n$ real variables $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{y} \in \mathbb{R}^n$. The transformation from $(\boldsymbol{z}, \bar{\boldsymbol{z}})$ to $(\boldsymbol{x}, \boldsymbol{y})$ is a simple change of variables for two independent vector variables,

$$\boldsymbol{x} = \frac{\boldsymbol{z} + \bar{\boldsymbol{z}}}{2}, \quad \boldsymbol{y} = \frac{\boldsymbol{z} - \bar{\boldsymbol{z}}}{2\mathrm{i}}, \tag{7.13}$$

and, vice versa,

$$\boldsymbol{z} = \boldsymbol{x} + \mathrm{i}\boldsymbol{y}, \quad \bar{\boldsymbol{z}} = \boldsymbol{x} - \mathrm{i}\boldsymbol{y}. \tag{7.14}$$

The complex-valued function is said to be **differentiable** at a point $\boldsymbol{z}_0 \in \mathbb{C}^n$ if the complex-valued limit operation

$$\lim_{\Delta\boldsymbol{z} \to \boldsymbol{0}} \frac{f(\boldsymbol{z}_0 + \Delta\boldsymbol{z}) - f(\boldsymbol{z}_0)}{\Delta\boldsymbol{z}} \tag{7.15}$$

exists, i.e., when the limit value is independent of the direction in which $\Delta\boldsymbol{z}$ approaches zero. For example, the result of the limit should be the same

[5] The Frobenius-norm of a multidimensional tensor polynomial in complex variables is a straightforward extension of (7.10), see [223–225].

when $\Delta \boldsymbol{z}$ approaches zero on the real axis ($\Delta \boldsymbol{x} \to \boldsymbol{0}$) or on the imaginary axis ($\Delta \boldsymbol{y} \to \boldsymbol{0}$). This requirement is formalized in the **Cauchy–Riemann conditions** [1, 53] for the differentiability at $\boldsymbol{z}_0 = \boldsymbol{x}_0 + \mathrm{i}\boldsymbol{y}_0$:

$$\begin{aligned} \frac{\partial u(\boldsymbol{x}_0, \boldsymbol{y}_0)}{\partial \boldsymbol{x}} &= \frac{\partial v(\boldsymbol{x}_0, \boldsymbol{y}_0)}{\partial \boldsymbol{y}}, \\ \frac{\partial v(\boldsymbol{x}_0, \boldsymbol{y}_0)}{\partial \boldsymbol{x}} &= -\frac{\partial u(\boldsymbol{x}_0, \boldsymbol{y}_0)}{\partial \boldsymbol{y}}. \end{aligned} \tag{7.16}$$

The Cauchy–Riemann conditions (7.16) are necessary and sufficient conditions[6] for the existence of the limit defining the complex differentiation operation in (7.15). A multivariate function in complex variables is said to be **holomorphic** in a domain (i.e., the complex generalization of analytic), if the function is differentiable for all points in that domain. Real-valued functions are, however, non-holomorphic. It is easy to see that the Cauchy–Riemann conditions (7.16) do not hold, except for the constant real-valued polynomial, because $v(\boldsymbol{x}, \boldsymbol{y}) \equiv 0$. In other words, there exists no Taylor series in $\boldsymbol{z}$ of $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$ at $\boldsymbol{z}_0$ so that the series converges to $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$ in a neighborhood of $\boldsymbol{z}_0$ [225].

Wirtinger calculus provides a general framework for differentiating non-holomorphic functions; it is general in the sense that it includes holomorphic functions as a special case. It only requires that $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$ or $f(\boldsymbol{z})$ is **real differentiable**: if $u(\boldsymbol{x}, \boldsymbol{y})$ and $v(\boldsymbol{x}, \boldsymbol{y})$ have continuous partial derivatives with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$, then the function is real differentiable [1]. The idea in Wirtinger calculus is to differentiate functions of the form $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$ by considering the partial derivatives with respect to $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$, which can be formally written as

$$\frac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial \boldsymbol{z}} = \frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} + \frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{z}} \tag{7.17}$$

$$= \frac{1}{2}\left(\frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{x}} - \mathrm{i} \frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{y}} \right), \tag{7.18}$$

$$\frac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial \bar{\boldsymbol{z}}} = \frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \bar{\boldsymbol{z}}} + \frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \bar{\boldsymbol{z}}} \tag{7.19}$$

$$= \frac{1}{2}\left(\frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{x}} + \mathrm{i} \frac{\partial f(\boldsymbol{z})}{\partial \boldsymbol{y}} \right). \tag{7.20}$$

We call $\frac{\partial(\cdot)}{\partial \boldsymbol{z}}$ and $\frac{\partial(\cdot)}{\partial \bar{\boldsymbol{z}}}$ the **cogradient operator** and **conjugate cogradient operator**, respectively. They act as a partial derivative with respect to $\boldsymbol{z}$ (or $\bar{\boldsymbol{z}}$), while treating $\bar{\boldsymbol{z}}$ (or $\boldsymbol{z}$) as a constant vector. Note that, for a complex-valued cost function that satisfies the Cauchy–Riemann conditions (7.16), $\frac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial \bar{\boldsymbol{z}}}$ is equal to zero [53]. Hence, differentiability in a complex domain requires the function $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$ to be solely a function of $\boldsymbol{z}$ and not exhibit any dependency on $\bar{\boldsymbol{z}}$. This is also the reason why we explicitly write real-valued functions in terms of $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$. For a real-valued function $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$, we have that

$$\overline{\left(\frac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial \boldsymbol{z}} \right)} = \frac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial \bar{\boldsymbol{z}}}. \tag{7.21}$$

[6] The Cauchy–Riemann conditions are necessary and sufficient only for continuous functions $u(\boldsymbol{x}, \boldsymbol{y})$ and $v(\boldsymbol{x}, \boldsymbol{y})$, see [1] for more information.

Although their definitions allow the cogradient and conjugate cogradient to be expressed elegantly in terms of $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$, neither contains enough information by itself to express the change in a function with respect to a change in $\boldsymbol{z}$ or $\bar{\boldsymbol{z}}$ as independent variables. Therefore, we define the **complex gradient operator** $\nabla(\cdot)$ as

$$\nabla(\cdot) = \left(\frac{\partial(\cdot)}{\partial \boldsymbol{z}}, \frac{\partial(\cdot)}{\partial \bar{\boldsymbol{z}}} \right). \tag{7.22}$$

Relation (7.21) between both cogradients, however, allows us to only compute one cogradient and obtain the other one by simply taking the complex conjugate of that expression.

Proposition 7.1. For the real-valued multivariate polynomial cost functions in complex variables in (7.8), a complex derivative does not exist, but Wirtinger calculus provides an elegant alternative framework to compute the **first-order necessary conditions for optimality**:

$$\begin{cases} p_i(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \dfrac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial z_i} = 0, & \text{for } i = 1, \dots, n, \\ p_{n+i}(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \dfrac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial \bar{z}_i} = 0, & \text{for } i = 1, \dots, n. \end{cases} \tag{7.23}$$

The common roots $(\boldsymbol{z}_0, \bar{\boldsymbol{z}}_0)$ of this square system of $2n$ multivariate polynomial equations in $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$ correspond to the stationary points of (7.8):

$$\mathcal{V}_{\mathbb{C}} = \{ \boldsymbol{z}_0 \in \mathbb{C}^n : p_i(\boldsymbol{z}_0, \bar{\boldsymbol{z}}_0) = 0, \forall i = 1, \dots, 2n \}. \tag{7.24}$$

Notice that the polynomials in (7.23) are not necessarily real-valued, only the original cost function is. We can illustrate all the above with a didactic example.

Example 7.1. Let us consider the univariate optimization problem:

$$\begin{aligned} &\min_{z, \bar{z}} -\mathrm{i}\big(z^3 + z^2\bar{z} - z\bar{z}^2 - \bar{z}^3\big) \\ &\text{subject to } \|z\|_2^2 - 3 = 0, \end{aligned} \tag{7.25}$$

which amounts to minimizing the real-valued polynomial cost function $f(z, \bar{z}) = -\mathrm{i}\big(z^3 + z^2\bar{z} - z\bar{z}^2 - \bar{z}^3\big) = 8x^2y$, where $x = \mathrm{Re}(z)$ and $y = \mathrm{Im}(z)$, on a circle with radius $\sqrt{3}$. We could approach this constrained optimization problem from the traditional point of view, via (7.13), and consider x and y as two independent real variables:

$$\begin{aligned} &\min_{x, y} 8x^2y \\ &\text{subject to } x^2 + y^2 - 3 = 0. \end{aligned} \tag{7.26}$$

However, since the cost function is real-valued, we can use Wirtinger derivatives. The Lagrangian that corresponds to this optimization problem is

$$\mathcal{L}(z, \bar{z}, \lambda) = -\mathrm{i}\big(z^3 + z^2\bar{z} - z\bar{z}^2 - \bar{z}^3\big) + \lambda(z\bar{z} - 3) \tag{7.27}$$

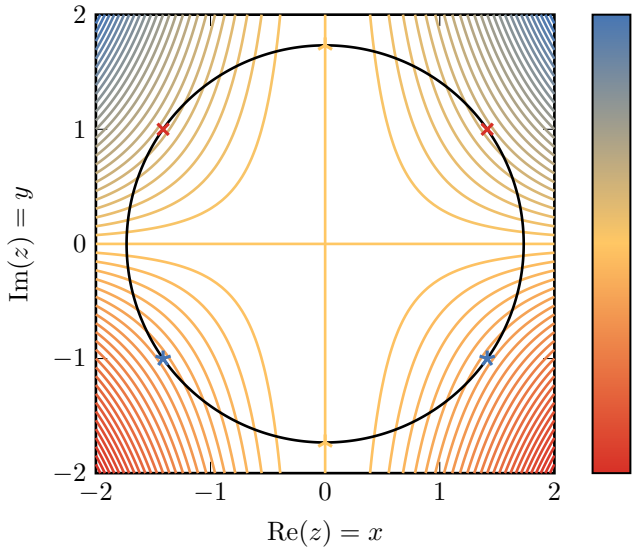


Figure 7.4. Contour lines of the real-valued polynomial cost function $f(z, \bar{z})$ in Example 7.1: the optimization problem has two global minimizers (✱), two global maximizers (×), and two local optima (⅄), subject to (—).

and its first-order necessary conditions for optimality (7.23) are

$$\begin{cases} \dfrac{\partial \mathcal{L}(z, \bar{z}, \lambda)}{\partial z} = -\mathrm{i}\big(3z^2 + 2z\bar{z} - \bar{z}^2\big) + \lambda \bar{z} = 0, \\ \dfrac{\partial \mathcal{L}(z, \bar{z}, \lambda)}{\partial \bar{z}} = -\mathrm{i}\big(z^2 - 2z\bar{z} - 3\bar{z}^2\big) + \lambda z = 0, \\ \dfrac{\partial \mathcal{L}(z, \bar{z}, \lambda)}{\partial \lambda} = z\bar{z} - 3 = 0. \end{cases} \tag{7.28}$$

When we solve this system of multivariate polynomial equations (for example, via the Macaulay matrix approaches proposed in Chapter 2), we obtain six stationary points: two global maximizers $\pm\sqrt{2} + 1\mathrm{i}$, two global minimizers $\pm\sqrt{2} - 1\mathrm{i}$, one local minimizer $\sqrt{3}\mathrm{i}$, and one local maximizer $-\sqrt{3}\mathrm{i}$ (subject to the constraints). This agrees with the visually identified stationary points in Figure 7.4. Notice that the first and second equation in (7.28) are complex conjugates of each other and that they are clearly not real-valued.

Remark 7.1. In the real case ($\boldsymbol{z} = \boldsymbol{x}$), the polynomial system (7.23) corresponds to the well-known real gradient set equal to zero. Suppose that we are only interested in the real stationary points $\boldsymbol{x}_0$ of the real-valued cost function $f(\boldsymbol{z}, \bar{\boldsymbol{z}})$ in $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$, we only consider the real gradient [224]:

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = 2\frac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial \boldsymbol{z}}\bigg|_{\boldsymbol{z}=\boldsymbol{x}} = 2\frac{\partial f(\boldsymbol{z}, \bar{\boldsymbol{z}})}{\partial \bar{\boldsymbol{z}}}\bigg|_{\boldsymbol{z}=\boldsymbol{x}}. \tag{7.29}$$

7.3.3 Globally optimal multivariate optimization

The fact that both the complex (decision) variables $\boldsymbol{z}$ and their complex conjugates $\bar{\boldsymbol{z}}$ are present in (7.23) clearly creates redundancy. After all, solving a system of multivariate polynomial equations is not an easy task at hand (Chapter 2). In this section, we show that the stationary points of (7.8) correspond to (some of) the eigenvalues of an MEP, by eliminating $\bar{\boldsymbol{z}}$ via the Macaulay matrix (Chapter 2).

Firstly, we rewrite every polynomial in (7.23) in terms of the different complex conjugate monomials $\bar{\boldsymbol{z}}^{\boldsymbol{\alpha}}$:

$$p_i(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \sum_{\{\boldsymbol{\alpha}\}} p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z}) \bar{\boldsymbol{z}}^{\boldsymbol{\alpha}}, \tag{7.30}$$

for $i = 1, \ldots, 2n$, where the summation runs over all multi-indices $\boldsymbol{\alpha}$. The multi-index $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ labels the powers of the conjugate variables $\bar{\boldsymbol{z}}$ in the monomials $\bar{\boldsymbol{z}}^{\boldsymbol{\alpha}}$ and indexes the coefficients $p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z})$ of $p_i(\boldsymbol{z}, \bar{\boldsymbol{z}})$, which are themselves also polynomials. The total degree of a monomial with respect to $\bar{\boldsymbol{z}}$ is equal to $|\boldsymbol{\alpha}|$ and the highest total degree with respect to $\bar{\boldsymbol{z}}$ among all the monomials of $p_i(\boldsymbol{z}, \bar{\boldsymbol{z}})$ defines the degree d_i in $\bar{\boldsymbol{z}}$ of that polynomial.

Example 7.2. To clarify, we consider

$$\begin{aligned} p(\boldsymbol{z}, \bar{\boldsymbol{z}}) &= 2 + z_2 + 3 z_1 z_2 \bar{z}_1 + z_1^2 \bar{z}_2 \\ &= p^{(00)}(\boldsymbol{z}) + p^{(10)}(\boldsymbol{z}) \bar{z}_1 + p^{(01)}(\boldsymbol{z}) \bar{z}_2, \end{aligned} \tag{7.31}$$

where the corresponding polynomial coefficients $p^{(\boldsymbol{\alpha})}(\boldsymbol{z})$ are

$$p^{(00)}(\boldsymbol{z}) = 2 + z_2, \tag{7.32}$$

$$p^{(10)}(\boldsymbol{z}) = 3 z_1 z_2, \tag{7.33}$$

$$p^{(01)}(\boldsymbol{z}) = z_1^2. \tag{7.34}$$

The degree of $p(\boldsymbol{z}, \bar{\boldsymbol{z}})$ with respect to $\bar{\boldsymbol{z}}$ is 1.

When we multiply a polynomial $p_i(\boldsymbol{z}, \bar{\boldsymbol{z}})$ by a monomial $\bar{\boldsymbol{z}}^{\boldsymbol{\delta_i}}$, we obtain a "new" polynomial,

$$\bar{\boldsymbol{z}}^{\boldsymbol{\delta_i}} p_i(\boldsymbol{z}, \bar{\boldsymbol{z}}) = \sum_{\{\boldsymbol{\alpha}\}} p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z}) \bar{\boldsymbol{z}}^{\boldsymbol{\alpha} + \boldsymbol{\delta}_i}, \tag{7.35}$$

which is similar to assigning every polynomial coefficient $p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z})$ to a monomial of higher degree in $\bar{\boldsymbol{z}}$. Note that these "new" polynomials, after equating them to zero, do not alter the solution set $\mathcal{V}_{\mathbb{C}}$ in (7.24) when we add them to (7.23).

Secondly, we define the Macaulay matrix $\boldsymbol{\mathcal{M}}_d$ with respect to the conjugate variables $\bar{\boldsymbol{z}}$. This matrix corresponds to the traditional Macaulay matrix from Definition 2.11 when treating $\boldsymbol{z}$ as a constant vector.

Definition 7.1. Consider the polynomials $p_i(\boldsymbol{z}, \bar{\boldsymbol{z}})$ with total degree d_i in $\bar{\boldsymbol{z}}$, for $i = 1, \ldots, 2n$, of the system of multivariate polynomial equations, which serve as the seed equations. The corresponding **Macaulay matrix with respect to the conjugate variables** of degree d in $\bar{\boldsymbol{z}}$, $\boldsymbol{\mathcal{M}}_d(\boldsymbol{z}) \in \mathbb{C}^{k_d \times l_d}$, contains the polynomial coefficients $p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z})$ of the seed equations and the equations $\bar{\boldsymbol{z}}^{\boldsymbol{\delta}_i} p_i(\boldsymbol{z}, \bar{\boldsymbol{z}}) = 0$ generated by a forward shift recursion (FSR) with monomials $\bar{\boldsymbol{z}}^{\boldsymbol{\delta}_i}$ of increasing total degrees in $\bar{\boldsymbol{z}}$ so that $|\boldsymbol{\delta}_i| = 0, \ldots, d - d_i$, for $i = 1, \ldots, 2n$.

Every row of $\boldsymbol{\mathcal{M}}_d(\boldsymbol{z})$ contains one polynomial $\bar{\boldsymbol{z}}^{\boldsymbol{\delta}_i} p_i(\boldsymbol{z}, \bar{\boldsymbol{z}})$, while every column is associated with one monomial $\bar{\boldsymbol{z}}^{\boldsymbol{\alpha}+\boldsymbol{\delta}_i}$, the highest total degree of which is equal to d. The Macaulay matrix with respect to the conjugate variables $\boldsymbol{\mathcal{M}}_d(\boldsymbol{z})$ is clearly a polynomial matrix in $\boldsymbol{z}$, hence the bold calligraphic notation, that gathers the polynomial coefficients $p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z})$ according to a certain pre-defined monomial ordering (for example, monomial ordering in Definition 2.6). The number of rows and columns of $\boldsymbol{\mathcal{M}}_d(\boldsymbol{z})$ depend on the degree d in $\bar{\boldsymbol{z}}$:

$$k_d = \sum_{i=1}^{2n} \binom{d - d_i + 2n}{d - d_i} \tag{7.36}$$

and

$$l_d = \binom{d + 2n}{d}. \tag{7.37}$$

We can use Definition 7.1 to rewrite (7.23) and additional equations (7.35) as a matrix-vector product,

$$\boldsymbol{\mathcal{M}}_d(\boldsymbol{z})\boldsymbol{q} = \boldsymbol{0}, \tag{7.38}$$

where the matrix $\boldsymbol{\mathcal{M}}_d(\boldsymbol{z})$ is the Macaulay matrix with respect to $\bar{\boldsymbol{z}}$ and the vector $\boldsymbol{q}$ contains the different complex conjugate monomials $\bar{\boldsymbol{z}}^{\boldsymbol{\alpha}+\boldsymbol{\delta}_i}$ ordered the same as the columns of $\boldsymbol{\mathcal{M}}_d(\boldsymbol{z})$.

Example 7.2 (continuing from p. 260). We can multiply $p(\boldsymbol{z}, \bar{\boldsymbol{z}})$ in (7.31) with all monomials $\bar{\boldsymbol{z}}^{\boldsymbol{\delta}}$ for which $|\boldsymbol{\delta}| = 1$, i.e., $\bar{z}_1 p(\boldsymbol{z}, \bar{\boldsymbol{z}})$ and $\bar{z}_2 p(\boldsymbol{z}, \bar{\boldsymbol{z}})$, or, construct the Macaulay matrix with respect to $\bar{\boldsymbol{z}}$ of degree $d = 3$ in $\bar{\boldsymbol{z}}$, to obtain a matrix-vector product as in (7.38):

$$\underbrace{\begin{bmatrix} 2+z_2 & 3z_1z_2 & z_1^2 & 0 & 0 & 0 \\ 0 & 2+z_2 & 0 & 3z_1z_2 & z_1^2 & 0 \\ 0 & 0 & 2+z_2 & 0 & 3z_1z_2 & z_1^2 \end{bmatrix}}_{\boldsymbol{\mathcal{M}}_3(\boldsymbol{z})} \underbrace{\begin{bmatrix} 1 \\ \bar{z}_1 \\ \bar{z}_2 \\ \bar{z}_1^2 \\ \bar{z}_1\bar{z}_2 \\ \bar{z}_2^2 \end{bmatrix}}_{\boldsymbol{q}} = \boldsymbol{0}. \tag{7.39}$$

Finally, notice that (7.38) is an MEP when expanding the multivariate matrix polynomial $\boldsymbol{\mathcal{M}}(\boldsymbol{z})$ in terms of the different complex monomials $\boldsymbol{z}^{\boldsymbol{\beta}}$,

$$\boldsymbol{\mathcal{M}}_d(\boldsymbol{z})\boldsymbol{q} = \left(\sum_{\{\boldsymbol{\beta}\}} \boldsymbol{M}_{\boldsymbol{\beta}} \boldsymbol{z}^{\boldsymbol{\beta}}\right)\boldsymbol{q} = \boldsymbol{0}, \tag{7.40}$$

where the summation runs over all the multi-indices $\boldsymbol{\beta}$. The minimal required degree d of $\boldsymbol{\mathcal{M}}_d(\boldsymbol{z})$ is such that $k_d \geq l_d + n - 1$, which is a necessary condition for the MEP to have a zero-dimensional solution set (Chapter 3). The coefficient matrices $\boldsymbol{M}_{\boldsymbol{\beta}} \in \mathbb{C}^{k_d \times l_d}$ of the MEP impose the structure of $\boldsymbol{q}$ and contain the coefficients of the polynomial coefficients $p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z})$ associated with $\boldsymbol{z}^{\boldsymbol{\beta}}$.

Example 7.2 (continuing from p. 261). For the polynomial $p(\boldsymbol{z}, \bar{\boldsymbol{z}})$ in (7.31), the polynomial coefficient $p^{(00)}(\boldsymbol{z}) = 2 + z_2$ creates coefficients 2 and 1 in the coefficient matrices $\boldsymbol{M}_{00}$ and $\boldsymbol{M}_{01}$, respectively. This results in

$$\boldsymbol{M}_{00} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \text{ and } \boldsymbol{M}_{01} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{7.41}$$

One approach to solve the resulting MEP is via the block Macaulay matrix approach developed in Chapter 3. The following example illustrates the above-described reformulation.

Remark 7.2. In the univariate case, polynomial system (7.23) only consists of two bivariate equations in z and $\bar{z}$. An elimination of the complex conjugate variable $\bar{z}$, via the well-known Sylvester matrix (i.e., the simplification of the Macaulay matrix for two univariate polynomials), results in a PEP instead of the MEP in (7.40). The Macaulay matrix from Definition 7.1 reduces to the well-known Sylvester matrix for $n = 1$. Note that the univariate case of our proposed optimization approach yields a similar PEP as in [223, 224]. The main difference is that the coefficient matrices in [223, 224] are the transpose of the coefficient matrices in (7.40).

Example 7.3. Consider the univariate optimization problem

$$\min_z \left\| z(z - 0.5j)^2 - z \right\|_2^2, \tag{7.42}$$

which clearly has a real-valued polynomial cost function. The corresponding system of Wirtinger derivatives is

$$\begin{cases} p_1(z, \bar{z}) = \dfrac{\partial f(z, \bar{z})}{\partial z} = 0, \\ p_2(z, \bar{z}) = \dfrac{\partial f(z, \bar{z})}{\partial \bar{z}} = 0, \end{cases} \tag{7.43}$$

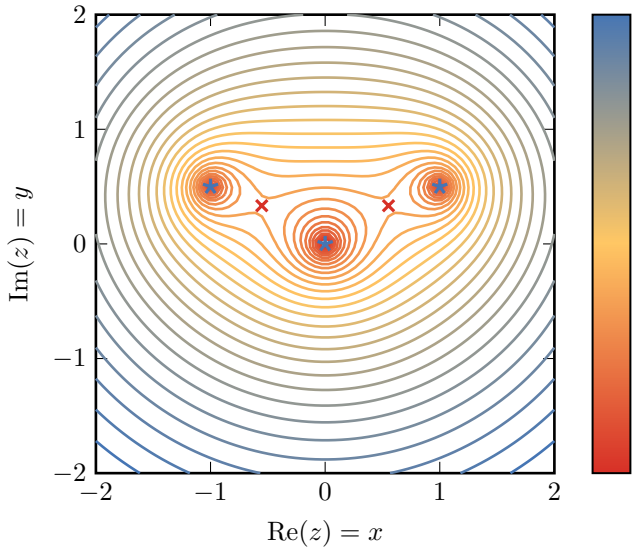


Figure 7.5. Contour lines of the real-valued polynomial cost function $f(z, \bar{z})$ in Example 7.3: the optimization problem has three minimizers (★) and two saddle points (×).

or

$$\begin{cases} p_1(z,\bar{z}) = 1.5625\bar{z} + 2.5\mathrm{i}z\bar{z} - 1.25\mathrm{i}\bar{z}^2 - 3.75z^2\bar{z} + 2z\bar{z}^2 \\ \qquad\quad - 1.25\bar{z}^3 + 3\mathrm{i}z^2\bar{z}^2 - 2\mathrm{i}z\bar{z}^3 + 3z^2\bar{z}^3 = 0, \\ p_2(z,\bar{z}) = 1.5625z + 1.25\mathrm{i}z^2 - 2.5\mathrm{i}z\bar{z} - 1.25z^3 + 2z^2\bar{z} \\ \qquad\quad - 3.75z\bar{z}^2 + 2\mathrm{i}z^3\bar{z} - 3\mathrm{i}z^2\bar{z}^2 + 3z^3\bar{z}^2 = 0. \end{cases} \tag{7.44}$$

We can construct the corresponding Sylvester matrix (of degree $d = 4$ in $\bar{z}$),

$$\boldsymbol{\mathcal{S}}_4(z) = \left[\begin{array}{ccccc} p_1^{(0)}(z) & p_1^{(1)}(z) & p_1^{(2)}(z) & p_1^{(3)}(z) & 0 \\ 0 & p_1^{(0)}(z) & p_1^{(1)}(z) & p_1^{(2)}(z) & p_1^{(3)}(z) \\ \hline p_2^{(0)}(z) & p_2^{(1)}(z) & p_2^{(2)}(z) & 0 & 0 \\ 0 & p_2^{(0)}(z) & p_2^{(1)}(z) & p_2^{(2)}(z) & 0 \\ 0 & 0 & p_2^{(0)}(z) & p_2^{(1)}(z) & p_2^{(2)}(z) \end{array}\right], \tag{7.45}$$

where $p_i^{(\alpha)}(z)$ is the polynomial that is associated with $\bar{z}^\alpha$ of $p_i(z, \bar{z})$ in (7.44). For example, $p_1^{(2)}(z) = -1.25\mathrm{i} + 2z + 3\mathrm{i}z^2$ because that are the monomials of $p_1(z, \bar{z})$ that are associated with $\bar{z}^2$. Subsequently, we create the coefficient matrices of the PEP from the Sylvester matrix by extracting the coefficients that belong to a power of z^β:

$$\big(\boldsymbol{S}_0 + \boldsymbol{S}_1 z + \boldsymbol{S}_2 z^2 + \boldsymbol{S}_3 z^3\big)\boldsymbol{q} = \boldsymbol{0}. \tag{7.46}$$

Taking again $p_1^{(2)}(z) = -1.25\mathrm{i} + 2z + 3\mathrm{i}z^2$, this leads to the coefficients $-1.25\mathrm{i}$ in $\boldsymbol{S}_0$, 2 in $\boldsymbol{S}_1$, and $3\mathrm{i}$ in $\boldsymbol{S}_2$ at the positions of $p_1^{(2)}(z)$ in $\boldsymbol{\mathcal{S}}_4(z)$. For clarity,

we show $\boldsymbol{S}_2$:

$$\boldsymbol{S}_2 = \left[\begin{array}{ccccc} 0 & -3.75 & 3\mathrm{i} & 3 & 0 \\ 0 & 0 & -3.75 & 3\mathrm{i} & 3 \\ \hline 1.25\mathrm{i} & 2 & -3\mathrm{i} & 0 & 0 \\ 0 & 1.25\mathrm{i} & 2 & -3\mathrm{i} & 0 \\ 0 & 0 & 1.25\mathrm{i} & 2 & -3\mathrm{i} \end{array}\right]. \tag{7.47}$$

Solving the resulting PEP, or the system (7.44) directly, yields 13 affine solutions: 3 minimizers, 2 saddle points, and 8 ghost solutions (Table 7.1). In Figure 7.5, we visualize the minimizers and saddle points on the contour lines of the real-valued polynomial cost function. We discuss these ghost solutions in more detail in Section 7.3.4.

7.3.4 About ghost solutions

In the context of complex optimization, **ghost solutions** (sometimes called *spurious solutions*) arise due to the fact that numerical optimization algorithms can not properly deal with complex conjugate variables. Ghost solutions can also arise in our proposed optimization approach, see Example 7.3. They emerge when solving the MEP (7.40), or the system of multivariate polynomial equations (7.23) directly, via numerical linear algebra algorithms that can not impose that $\bar{\boldsymbol{z}}$ is the complex conjugate of $\boldsymbol{z}$. In that case, we essentially tackle the problem as if $\boldsymbol{z}$ and $\bar{\boldsymbol{z}}$ (let us call them $\boldsymbol{u}$ and $\boldsymbol{v}$ here) are independent variables, which results in the candidate solution set (instead of the desired solution set of (7.24))

$$\mathcal{V}_{\widetilde{\mathbb{C}}} = \big\{(\boldsymbol{u}_0, \boldsymbol{v}_0) \in \mathbb{C}^{2n} : p_i(\boldsymbol{u}_0, \boldsymbol{v}_0) = 0, \forall i = 1, \ldots, 2n\big\}. \tag{7.48}$$

Of course, we only want the subset for which $(\boldsymbol{u}_0, \boldsymbol{v}_0) = (\boldsymbol{z}_0, \bar{\boldsymbol{z}}_0)$, i.e., the true stationary points of (7.8), and we need to remove these ghost solutions from (7.48). Luckily, this is not a difficult task, even if we only compute the eigenvalues $\boldsymbol{u}$ ($\boldsymbol{v}$ is then part of the eigenvector): we can (i) substitute the obtained eigenvalues and their complex conjugates in (7.23) and check if $(\boldsymbol{u}_0, \boldsymbol{v}_0)$ is indeed a stationary point of (7.8) or (ii) construct an eigenvector $\boldsymbol{q}_0$ from the complex conjugate of $\boldsymbol{u}_0$ and check if $\boldsymbol{q}_0$ is indeed an eigenvector of $\boldsymbol{\mathcal{M}}(\boldsymbol{u}_0)$. An alternative heuristic technique to filter out ghost solutions (and to prune wrong solutions due to rounding errors) based on the well-known Newton–Raphson method was proposed in [224]. However, this technique is known to fail in some cases [224].

Remark 7.3. Note that using the standard approach for complex optimization, using derivatives with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$ and solving the resulting polynomial system of first-order necessary conditions for optimality, also can result in ghost solutions. In this situation, ghost solutions are candidate solutions that are complex-valued, while $\boldsymbol{x}_0$ and $\boldsymbol{y}_0$ have to be real-valued. These ghost solutions emerge because systems of multivariate polynomial

Table 7.1. Numerical values of the candidate solutions (u_0, v_0) of (7.44). Next to "true" stationary points of the cost function in (7.42), we also obtain ghost solutions.

u_0	v_0	classification
$1.0000 + 0.5000\mathrm{i}$	$1.0000 - 0.5000\mathrm{i}$	minimizer
$-1.0000 + 0.5000\mathrm{i}$	$-1.0000 - 0.5000\mathrm{i}$	minimizer
$0.0000 + 0.0000\mathrm{i}$	$0.0000 + 0.0000\mathrm{i}$	minimizer
$0.5528 + 0.3333\mathrm{i}$	$0.5528 - 0.3333\mathrm{i}$	saddle point
$-0.5528 + 0.3333\mathrm{i}$	$-0.5528 - 0.3333\mathrm{i}$	saddle point
$1.0000 + 0.5000\mathrm{i}$	$-1.0000 - 0.5000\mathrm{i}$	ghost solution
$1.0000 + 0.5000\mathrm{i}$	$0.0000 + 0.0000\mathrm{i}$	ghost solution
$-1.0000 + 0.5000\mathrm{i}$	$1.0000 - 0.5000\mathrm{i}$	ghost solution
$-1.0000 + 0.5000\mathrm{i}$	$0.0000 + 0.0000\mathrm{i}$	ghost solution
$0.0000 + 0.0000\mathrm{i}$	$1.0000 - 0.5000\mathrm{i}$	ghost solution
$0.0000 + 0.0000\mathrm{i}$	$-1.0000 - 0.5000\mathrm{i}$	ghost solution
$-0.5528 + 0.3333\mathrm{i}$	$0.5528 - 0.3333\mathrm{i}$	ghost solution
$0.5528 + 0.3333\mathrm{i}$	$-0.5528 - 0.3333\mathrm{i}$	ghost solution

equations and MEPs, without additional constraints, can also have complex solutions. When considering a specific problem with both approaches, it is possible to show that every ghost solution $(\boldsymbol{x}_0, \boldsymbol{y}_0)$ corresponds to a ghost solution $(\boldsymbol{u}_0, \boldsymbol{v}_0)$, via (7.13), and vice versa, via (7.14).

Example 7.4. When solving the PEP in (7.46) or the system of multivariate polynomial equations in (7.44) with numerical linear algebra algorithms, we obtain 13 affine solutions (Table 7.1): 3 minimizers, 2 saddle points, and 8 ghost solutions. The ghost solutions can be deflated from the candidate solution set by checking for every candidate solution u_0 if the candidate solution u_0 and its complex conjugate $\bar{u}_0$ are indeed a solution of (7.44) or by checking if the eigenvector $\boldsymbol{q}_0$ constructed from the complex conjugate $\bar{u}_0$ of the candidate solution is indeed an eigenvector of the PEP for u_0.

7.3.5 Numerical example

Finally, we try to fit as good as possible a rank-1 matrix to a given complex matrix.

Example 7.5. Consider the problem where we try to fit a rank-1 matrix to a given complex 2×2 matrix $\boldsymbol{A} \in \mathbb{C}^{2\times 2}$:

$$\min_{\boldsymbol{z}} \left\| \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} - \begin{bmatrix} z_1^2 & z_1 z_2 \\ z_1 z_2 & z_2^2 \end{bmatrix} \right\|_{\mathrm{F}}^2, \tag{7.49}$$

which is an example of a nonlinear least-squares optimization problem (7.11).

The system of first-order necessary conditions for optimality for this example is

$$\begin{cases} p_1(\boldsymbol{z},\bar{\boldsymbol{z}}) = \dfrac{\partial f(\boldsymbol{z},\bar{\boldsymbol{z}})}{\partial z_1} = 0, \\ p_2(\boldsymbol{z},\bar{\boldsymbol{z}}) = \dfrac{\partial f(\boldsymbol{z},\bar{\boldsymbol{z}})}{\partial z_2} = 0, \\ p_3(\boldsymbol{z},\bar{\boldsymbol{z}}) = \dfrac{\partial f(\boldsymbol{z},\bar{\boldsymbol{z}})}{\partial \bar{z}_1} = 0, \\ p_4(\boldsymbol{z},\bar{\boldsymbol{z}}) = \dfrac{\partial f(\boldsymbol{z},\bar{\boldsymbol{z}})}{\partial \bar{z}_2} = 0, \end{cases} \tag{7.50}$$

or

$$\begin{cases} p_1(\boldsymbol{z},\bar{\boldsymbol{z}}) = -2\bar{a}_{11}z_1 + 2z_1\bar{z}_1^2 - (\bar{a}_{12}+\bar{a}_{21})z_2 + 2z_2\bar{z}_1\bar{z}_2 = 0, \\ p_2(\boldsymbol{z},\bar{\boldsymbol{z}}) = -2\bar{a}_{22}z_2 + 2z_2\bar{z}_2^2 - (\bar{a}_{12}+\bar{a}_{21})z_1 + 2z_1\bar{z}_1\bar{z}_2 = 0, \\ p_3(\boldsymbol{z},\bar{\boldsymbol{z}}) = -2a_{11}\bar{z}_1 + 2z_1^2\bar{z}_1 - (a_{12}+a_{21})\bar{z}_2 + 2z_1z_2\bar{z}_2 = 0, \\ p_4(\boldsymbol{z},\bar{\boldsymbol{z}}) = -2a_{22}\bar{z}_2 + 2z_2^2\bar{z}_2 - (a_{12}+a_{21})\bar{z}_1 + 2z_1z_2\bar{z}_1 = 0, \end{cases} \tag{7.51}$$

with $\boldsymbol{z} = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^{\mathrm{T}}$ and $\bar{\boldsymbol{z}} = \begin{bmatrix} \bar{z}_1 & \bar{z}_2 \end{bmatrix}^{\mathrm{T}}$. We visualizes the Macaulay matrix $\boldsymbol{\mathcal{M}}_3(\boldsymbol{z})$ of degree $d = 3$ in $\bar{\boldsymbol{z}}$ for these polynomials in Figure 7.6. Each coefficient of $\boldsymbol{\mathcal{M}}_3(\boldsymbol{z})$ is a polynomial coefficient $p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z})$ associated with a monomial $\bar{\boldsymbol{z}}^{\boldsymbol{\alpha}}$. For example, the green dot (●) corresponds to $p_1^{(20)}(\boldsymbol{z}) = 2z_1$ and is associated with $\bar{z}_1^2$. This Macaulay matrix leads to a quadratic two-parameter eigenvalue problem,

$$\big(\boldsymbol{M}_{00} + \boldsymbol{M}_{10}z_1 + \boldsymbol{M}_{01}z_2 + \boldsymbol{M}_{20}z_1^2 + \boldsymbol{M}_{11}z_1z_2 + \boldsymbol{M}_{02}z_2^2\big)\boldsymbol{q} = \boldsymbol{0}, \tag{7.52}$$

which we can solve, for example, via a block Macaulay matrix approach. If we consider the given matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & 1\mathrm{i} \\ 1\mathrm{i} & -2 \end{bmatrix}, \tag{7.53}$$

then we obtain nine stationary points after solving (7.52). The global minimizer is $(0.8507, 1.3764\mathrm{i})$, which also corresponds to the first triplet obtained via the complex singular value decomposition of $\boldsymbol{A}$.

7.4 Model order reduction

In model order reduction, our second key example, we start from a higher-order model of a dynamical system, but we want to reduce the complexity and approximate this higher-order model by a reduced-order model [8]. This is a different setting than in system identification (see next section), where we try to find a model from the available measured data. Model order reduction aims to approximate a large high-order model by a model of lower order (less states). Large models may be too complicated for simulation or for control system design; hence, model order reduction in these scenarios is of crucial importance

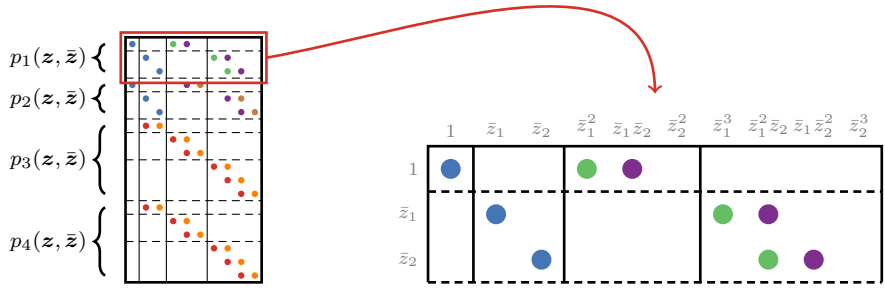


Figure 7.6. Visualization of the Macaulay matrix that generates the coefficient matrices of the MEP in Example 7.5, with (on the right side) a detailed illustration of the polynomial coefficients of $p_1(\boldsymbol{z}, \bar{\boldsymbol{z}})$. The row-labels denote the shifted polynomials $\bar{\boldsymbol{z}}^{\boldsymbol{\delta}_i} p_i(\boldsymbol{z}, \bar{\boldsymbol{z}})$, while the column-labels denote the associated monomials $\bar{\boldsymbol{z}}^{\boldsymbol{\alpha}}$. Every colored dot corresponds to one of the (non-zero) polynomial coefficients $p_i^{(\boldsymbol{\alpha})}(\boldsymbol{z})$ of the polynomials. For example, the green dot (●) corresponds to $p_1^{(20)}(\boldsymbol{z})$, which is shifted throughout the Macaulay matrix after multiplying $p_1(\boldsymbol{z}, \bar{\boldsymbol{z}})$ by 1 (i.e., the original polynomial), $\bar{z}_1$, and $\bar{z}_2$.

(see [8] for some motivating examples). For polynomial models, the approximation problem can in certain norms be rephrased as a multivariate polynomial optimization problem [3, 148], where the stationary points are candidates for model parameters of the reduced-order models.

In this section, we address the model order reduction problem for SISO LTI systems. The approximation problem is considered in a $\mathcal{H}_2$-norm way. The underlying optimization problem is non-convex, implying that there exist many local minima and that obtaining the global minimizers is known to be a very challenging task. State-of-the-art solvers provide a heuristic approach to the optimization problem (see the "historical and bibliographical notes" of this chapter for an overview). These algorithms are not guaranteed to converge to the globally optimal solution, despite the use of several heuristic rules during their initialization. Nonetheless, we can not stress enough their importance in large practical applications. For the particular cases of first-order and second-order SISO approximants, the global optimum can be found by solving a polynomial system in one and two variables, respectively [4, 5]. In [107, 108], it has been shown that the special case of order-one reductions can be solved by computing the common roots of a system of quadratic polynomial equations. This section, on the other hand, provides a unique approach to obtain the globally optimal solution of the $\mathcal{H}_2$-norm model order reduction problem for an approximant of arbitrary order. It reformulates the problem as an MEP, an approach that has also been used in the (very) recent work by Alsubaie [6] and Lagauw et al. [148]. Where Alsubaie [6] has followed an approach inspired by iterative rational Krylov methods and Lagauw et al. [148] have applied Walsh's theorem in the frequency domain to obtain an MEP, this approach exploits the

Lyapunov equation to rewrite the cost function of the underlying optimization problem. The result is an MEP in terms of the unknown parameters of the reduced-order transfer function[7]. It is clear that the block Macaulay matrix approach from Chapter 3 can play an important role in finding these globally optimal reduced-order models.

The remainder of this section is organized as follows: We formulate the second key example mathematically (Section 7.4.1), before deriving the first-order optimality conditions of an appropriately redefined cost function and explaining how this system of first-order optimality conditions can be transformed into an MEP[8] (Section 7.4.2). We also present a numerical example to illustrate our methodology and to show how the block Macaulay matrix can be useful for model order reduction (Section 7.4.3).

7.4.1 Problem formulation

The model order reduction problem for SISO LTI systems can be cast in the following way: For a given nth-order LTI continuous-time stable system with transfer function

$$G(s) = \boldsymbol{C}(s\boldsymbol{I}_n - \boldsymbol{A})^{-1}\boldsymbol{B}, \tag{7.54}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times 1}$, and $\boldsymbol{C} \in \mathbb{R}^{1\times n}$ are the system matrices, we look for an rth-order stable reduced model

$$G_r(s) = \boldsymbol{C}_r(s\boldsymbol{I}_r - \boldsymbol{A}_r)^{-1}\boldsymbol{B}_r, \tag{7.55}$$

with $r < n$, $\boldsymbol{A}_r \in \mathbb{R}^{r\times r}$, $\boldsymbol{B}_r \in \mathbb{R}^{r\times 1}$, and $\boldsymbol{C}_r \in \mathbb{R}^{1\times r}$, so that $G_r(s)$ is a "good approximation" of $G(s)$. In the particular setting of $\mathcal{H}_2$-norm model order reduction, we seek to minimize the squared $\mathcal{H}_2$-norm of $G_e(s) = G(s) - G_r(s)$, i.e.,

$$\min \|G(s) - G_r(s)\|_{\mathcal{H}_2}^2, \tag{7.56}$$

where the $\mathcal{H}_2$-norm of $G_e(s)$ is defined as

$$\|G_e(s)\|_{\mathcal{H}_2} = \sqrt{\frac{1}{2\pi}\int_{-\infty}^{\infty} |G_e(j\omega)|^2 d\omega} \tag{7.57}$$

$$= \sqrt{\int_0^{\infty} g_e(t)^2 dt}. \tag{7.58}$$

Here, $g_e(t)$ is the impulse response of $G_e(s)$, so the $\mathcal{H}_2$-error could be interpreted as the area under the impulse response of the error system. Note that the $\mathcal{H}_2$-norm is only defined (bounded) for stable and strictly proper transfer functions.

[7] Although all three approaches provide a $\mathcal{H}_2$-norm approach to compute the globally optimal approximant of arbitrary order, the computational properties of the resulting MEPs may vary a lot, depending on the specific orders of the original high-order and required reduced-order model. A first attempt to compare the computational properties of all three approaches can be found in [148].

[8] While we rephrase the globally optimal $\mathcal{H}_2$-norm model reduction problem as a homogeneous MEP in this chapter, the original paper [3] resulted in an inhomogeneous MEP, which could be solved via an extension of the block Macaulay matrix, the so-called augmented block Macaulay matrix.

7.4.2 Globally optimal model order reduction

This optimization problem (7.56) is non-convex and obtaining the global minimizer is known to be a very challenging task. In this section, we show that finding the optimal and suboptimal solutions of the $\mathcal{H}_2$-norm model order reduction problem (7.56) is equivalent to finding the common roots of a system of multivariate polynomial equations, and, eventually, to solving an MEP.

7.4.2.1 Redefined cost function

The $\mathcal{H}_2$-norm of the error transfer function $G_e(s)$ can be computed algebraically via its state space realization, instead of evaluating the integral in (7.57). Hence, as shown by Antoulas [8] and Van Dooren et al. [252], we can conveniently express the cost function of the optimization problem (7.56) as

$$\sigma^2(\boldsymbol{a}, \boldsymbol{b}) = \|G_e(s)\|_{\mathcal{H}_2}^2 = \boldsymbol{C}_e \boldsymbol{W} \boldsymbol{C}_e^{\mathrm{T}}, \tag{7.59}$$

where $\boldsymbol{W} = \boldsymbol{W}^{\mathrm{T}} \in \mathbb{R}^{(n+r)\times(n+r)}$ is the controllability Gramian of $G_e(s)$ satisfying the Lyapunov equation

$$\boldsymbol{A}_e \boldsymbol{W} + \boldsymbol{W} \boldsymbol{A}_e^{\mathrm{T}} + \boldsymbol{B}_e \boldsymbol{B}_e^{\mathrm{T}} = \boldsymbol{0}, \tag{7.60}$$

with

$$\boldsymbol{A}_e = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_r \end{bmatrix}, \boldsymbol{B}_e = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{B}_r \end{bmatrix}, \text{ and } \boldsymbol{C}_e = \begin{bmatrix} \boldsymbol{C} & -\boldsymbol{C}_r \end{bmatrix} \tag{7.61}$$

the system matrices of

$$G_e(s) = \boldsymbol{C}_e (s\boldsymbol{I}_{n+r} - \boldsymbol{A}_e)^{-1} \boldsymbol{B}_e. \tag{7.62}$$

Now, we rewrite the cost function (7.59) only in terms of the unknown parameters (a_i and b_i, $\forall i = 1, \ldots, r$) of the transfer function of the reduced-order model (keep in mind that $\boldsymbol{W}$ is also unknown)

$$G_r(s) = \frac{b_1 s^{r-1} + b_2 s^{r-2} + \cdots + b_{r-1} s + b_r}{s^r + a_1 s^{r-1} + \cdots + a_{r-1} s + a_r}. \tag{7.63}$$

As state space representation of $G_r(s)$, we use its control canonical form:

$$\boldsymbol{A}_r = \begin{bmatrix} -a_1 & -a_2 & \cdots & -a_{r-1} & -a_r \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \boldsymbol{B}_r = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{7.64}$$

$$\text{and } \boldsymbol{C}_r = \begin{bmatrix} b_1 & b_2 & b_3 & \cdots & b_r \end{bmatrix}. \tag{7.65}$$

By partitioning $\boldsymbol{W}$, we can rewrite the cost function (7.59) as

$$\sigma^2(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{C}_e \boldsymbol{W} \boldsymbol{C}_e^{\mathrm{T}} \tag{7.66}$$

$$= \begin{bmatrix} \boldsymbol{C} & -\boldsymbol{C}_r \end{bmatrix} \begin{bmatrix} \boldsymbol{W}_{11} & \boldsymbol{W}_{12} \\ \boldsymbol{W}_{21} & \boldsymbol{W}_{22} \end{bmatrix} \begin{bmatrix} \boldsymbol{C}^{\mathrm{T}} \\ -\boldsymbol{C}_r^{\mathrm{T}} \end{bmatrix} \tag{7.67}$$

$$= \boldsymbol{C}_r \boldsymbol{W}_{22} \boldsymbol{C}_r^{\mathrm{T}} - 2\boldsymbol{C}_r \boldsymbol{W}_{21} \boldsymbol{C}^{\mathrm{T}} + \boldsymbol{C} \boldsymbol{W}_{11} \boldsymbol{C}^{\mathrm{T}} \tag{7.68}$$

and the Lyapunov equation (7.60) as

$$\begin{bmatrix} \boldsymbol{A}\boldsymbol{W}_{11} + \boldsymbol{W}_{11}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} & \boldsymbol{A}\boldsymbol{W}_{12} + \boldsymbol{W}_{12}\boldsymbol{A}_r^{\mathrm{T}} + \boldsymbol{B}\boldsymbol{B}_r^{\mathrm{T}} \\ \boldsymbol{A}_r\boldsymbol{W}_{21} + \boldsymbol{W}_{21}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{B}_r\boldsymbol{B}^{\mathrm{T}} & \boldsymbol{A}_r\boldsymbol{W}_{22} + \boldsymbol{W}_{22}\boldsymbol{A}_r^{\mathrm{T}} + \boldsymbol{B}_r\boldsymbol{B}_r^{\mathrm{T}} \end{bmatrix} = \boldsymbol{0}, \quad (7.69)$$

where $\boldsymbol{W}_{11}$ and $\boldsymbol{W}_{22}$ are the controllability Gramians of $G(s)$ and $G_r(s)$, respectively, and $\boldsymbol{W}_{12} = \boldsymbol{W}_{21}^{\mathrm{T}}$ because $\boldsymbol{W} = \boldsymbol{W}^{\mathrm{T}}$. Notice that the term $\boldsymbol{C}\boldsymbol{W}_{11}\boldsymbol{C}^{\mathrm{T}}$ in (7.68) can be dropped, since it does not depend on the parameters of $G_r(s)$. Thus, we can use

$$\widetilde{\sigma}^2(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{C}_r\boldsymbol{W}_{22}\boldsymbol{C}_r^{\mathrm{T}} - 2\boldsymbol{C}_r\boldsymbol{W}_{21}\boldsymbol{C}^{\mathrm{T}} \quad (7.70)$$

as the new cost function.

In what follows, we eliminate $\boldsymbol{W}_{22}$ and $\boldsymbol{W}_{21}$ from $\widetilde{\sigma}^2(\boldsymbol{a}, \boldsymbol{b})$ by using (7.69). It is not difficult to see that (vec$(\cdot)$ denotes the vectorization operator)

$$\boldsymbol{C}_r\boldsymbol{W}_{22}\boldsymbol{C}_r^{\mathrm{T}} = \mathrm{vec}\Big(\boldsymbol{C}_r^{\mathrm{T}}\boldsymbol{C}_r\Big)^{\mathrm{T}}\,\mathrm{vec}(\boldsymbol{W}_{22}) \quad (7.71)$$

and

$$\boldsymbol{C}_r\boldsymbol{W}_{21}\boldsymbol{C}^{\mathrm{T}} = \mathrm{vec}\Big(\boldsymbol{C}_r^{\mathrm{T}}\boldsymbol{C}\Big)^{\mathrm{T}}\,\mathrm{vec}(\boldsymbol{W}_{21}). \quad (7.72)$$

If we introduce the auxiliary vectors $\boldsymbol{g}_r = \mathrm{vec}\Big(\boldsymbol{C}_r^{\mathrm{T}}\boldsymbol{C}_r\Big) \in \mathbb{R}^{r^2\times 1}$ and $\boldsymbol{g}_m = \mathrm{vec}\Big(\boldsymbol{C}_r^{\mathrm{T}}\boldsymbol{C}\Big) \in \mathbb{R}^{(nr)\times 1}$, then we can compactly write $\widetilde{\sigma}^2(\boldsymbol{a}, \boldsymbol{b})$ as

$$\widetilde{\sigma}^2(\boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{g}_r^{\mathrm{T}}\,\mathrm{vec}(\boldsymbol{W}_{22}) - 2\boldsymbol{g}_m^{\mathrm{T}}\,\mathrm{vec}(\boldsymbol{W}_{21}). \quad (7.73)$$

Notice that the Lyapunov equation $\boldsymbol{A}_r\boldsymbol{W}_{22} + \boldsymbol{W}_{22}\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{B}_r\boldsymbol{B}_r^{\mathrm{T}} = \boldsymbol{0}$ can be expressed as (see [124])

$$(\boldsymbol{A}_r \otimes \boldsymbol{I}_r + \boldsymbol{I}_r \otimes \boldsymbol{A}_r)\,\mathrm{vec}(\boldsymbol{W}_{22}) = -\,\mathrm{vec}\Big(\boldsymbol{B}_r\boldsymbol{B}_r^{\mathrm{T}}\Big) \quad (7.74)$$

$$\Downarrow$$

$$\underbrace{(\boldsymbol{A}_r \oplus \boldsymbol{A}_r)}_{\boldsymbol{T}_r}\,\mathrm{vec}(\boldsymbol{W}_{22}) = -\underbrace{\mathrm{vec}\Big(\boldsymbol{B}_r\boldsymbol{B}_r^{\mathrm{T}}\Big)}_{\boldsymbol{f}_r}, \quad (7.75)$$

and the Sylvester equation $\boldsymbol{A}_r\boldsymbol{W}_{21} + \boldsymbol{W}_{21}\boldsymbol{A}_r^{\mathrm{T}} + \boldsymbol{B}_r\boldsymbol{B}^{\mathrm{T}} = \boldsymbol{0}$ as

$$(\boldsymbol{A} \otimes \boldsymbol{I}_r + \boldsymbol{I}_n \otimes \boldsymbol{A}_r)\,\mathrm{vec}(\boldsymbol{W}_{21}) = -\,\mathrm{vec}\Big(\boldsymbol{B}_r\boldsymbol{B}^{\mathrm{T}}\Big) \quad (7.76)$$

$$\Downarrow$$

$$\underbrace{(\boldsymbol{A} \oplus \boldsymbol{A}_r)}_{\boldsymbol{T}_m}\,\mathrm{vec}(\boldsymbol{W}_{21}) = -\underbrace{\mathrm{vec}\Big(\boldsymbol{B}_r\boldsymbol{B}^{\mathrm{T}}\Big)}_{\boldsymbol{f}_m}, \quad (7.77)$$

with $\boldsymbol{T}_r \in \mathbb{R}^{r^2\times r^2}$, $\boldsymbol{f}_r \in \mathbb{R}^{r^2\times 1}$, $\boldsymbol{T}_m \in \mathbb{R}^{(nr)\times(nr)}$, and $\boldsymbol{f}_m \in \mathbb{R}^{(nr)\times 1}$. The operator $\otimes$ is again the Kronecker product and the operator $\oplus$ denotes the Kronecker sum.

Finally, from (7.75) and (7.77), we have that $\text{vec}(\boldsymbol{W}_{22}) = -\boldsymbol{T}_r^{-1}\boldsymbol{f}_r$ and $\text{vec}(\boldsymbol{W}_{21}) = -\boldsymbol{T}_m^{-1}\boldsymbol{f}_m$, and, by substituting them into (7.73), we get $\widetilde{\sigma}^2(\boldsymbol{a},\boldsymbol{b})$ only in terms of the parameters a_i and b_i ($\forall i = 1,\ldots,r$) of $G_r(s)$:

$$\widetilde{\sigma}^2(\boldsymbol{a},\boldsymbol{b}) = -\boldsymbol{g}_r^{\mathrm{T}}\boldsymbol{T}_r^{-1}\boldsymbol{f}_r + 2\boldsymbol{g}_m^{\mathrm{T}}\boldsymbol{T}_m^{-1}\boldsymbol{f}_m. \tag{7.78}$$

This cost function has to be minimized over the unknown parameters a_i and b_i, $\forall i = 1,\ldots,r$.

Remark 7.4. From [124, Theorem 4.4.5], we know that the eigenvalues of the Kronecker sum of two matrices $\boldsymbol{X} \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{Y} \in \mathbb{R}^{n_y \times n_y}$ correspond to all possible pairwise sums of the eigenvalues of $\boldsymbol{X}$ and $\boldsymbol{Y}$, that is, if $\Lambda(\boldsymbol{X}) = \{\lambda_1,\ldots,\lambda_{n_x}\}$ and $\Lambda(\boldsymbol{Y}) = \{\mu_1,\ldots,\mu_{n_y}\}$, then $\Lambda(\boldsymbol{X}\oplus\boldsymbol{Y}) = \{\lambda_i + \mu_j : i = 1,\ldots,n_x, j = 1,\ldots,n_y\}$. A sufficient condition for $\boldsymbol{T}_r$ and $\boldsymbol{T}_m$ to be invertible can be drawn from this result: If all the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{A}_r$ have a negative real part (which is the case for the optimal and suboptimal solutions of (7.56)), then all the eigenvalues of $\boldsymbol{T}_r = \boldsymbol{A}_r \oplus \boldsymbol{A}_r$ and $\boldsymbol{T}_m = \boldsymbol{A} \oplus \boldsymbol{A}_r$ also have a negative real part, implying the non-singularity of the matrices $\boldsymbol{T}_r$ and $\boldsymbol{T}_m$.

7.4.2.2 First-order necessary conditions for optimality

Keeping in mind that the vectors $\boldsymbol{g}_r$ and $\boldsymbol{g}_m$ are only a function of b_i, and the matrices $\boldsymbol{T}_r$ and $\boldsymbol{T}_m$ are only a function of a_i, the first-order necessary conditions for optimality of $\widetilde{\sigma}^2(\boldsymbol{a},\boldsymbol{b})$, $\forall i = 1,\ldots,r$, are given by

$$\frac{\partial\widetilde{\sigma}^2(\boldsymbol{a},\boldsymbol{b})}{\partial a_i} = -\boldsymbol{g}_r^{\mathrm{T}}\frac{\partial\boldsymbol{T}_r^{-1}}{\partial a_i}\boldsymbol{f}_r + 2\boldsymbol{g}_m^{\mathrm{T}}\frac{\partial\boldsymbol{T}_m^{-1}}{\partial a_i}\boldsymbol{f}_m = 0 \tag{7.79}$$

$$\frac{\partial\widetilde{\sigma}^2(\boldsymbol{a},\boldsymbol{b})}{\partial b_i} = -\frac{\partial\boldsymbol{g}_r^{\mathrm{T}}}{\partial b_i}\boldsymbol{T}_r^{-1}\boldsymbol{f}_r + 2\frac{\partial\boldsymbol{g}_m^{\mathrm{T}}}{\partial b_i}\boldsymbol{T}_m^{-1}\boldsymbol{f}_m = 0. \tag{7.80}$$

Since $\frac{\partial\boldsymbol{T}_r^{-1}}{\partial a_i} = -\boldsymbol{T}_r^{-1}\frac{\partial\boldsymbol{T}_r}{\partial a_i}\boldsymbol{T}_r^{-1}$ and $\frac{\partial\boldsymbol{T}_m^{-1}}{\partial a_i} = -\boldsymbol{T}_m^{-1}\frac{\partial\boldsymbol{T}_m}{\partial a_i}\boldsymbol{T}_m^{-1}$, the previous equations become

$$\frac{\partial\widetilde{\sigma}^2(\boldsymbol{a},\boldsymbol{b})}{\partial a_i} = \boldsymbol{g}_r^{\mathrm{T}}\boldsymbol{T}_r^{-1}\boldsymbol{T}_r^{a_i}\boldsymbol{T}_r^{-1}\boldsymbol{f}_r - 2\boldsymbol{g}_m^{\mathrm{T}}\boldsymbol{T}_m^{-1}\boldsymbol{T}_m^{a_i}\boldsymbol{T}_m^{-1}\boldsymbol{f}_m = 0, \tag{7.81}$$

$$\frac{\partial\widetilde{\sigma}^2(\boldsymbol{a},\boldsymbol{b})}{\partial b_i} = -\boldsymbol{g}_r^{b_i\mathrm{T}}\boldsymbol{T}_r^{-1}\boldsymbol{f}_r + 2\boldsymbol{g}_m^{b_i\mathrm{T}}\boldsymbol{T}_m^{-1}\boldsymbol{f}_m = 0, \tag{7.82}$$

with $\boldsymbol{T}_r^{a_i} = \frac{\partial\boldsymbol{T}_r}{\partial a_i}$, $\boldsymbol{T}_m^{a_i} = \frac{\partial\boldsymbol{T}_m}{\partial a_i}$, $\boldsymbol{g}_r^{b_i} = \frac{\partial\boldsymbol{g}_r}{\partial b_i}$, and $\boldsymbol{g}_m^{b_i} = \frac{\partial\boldsymbol{g}_m}{\partial b_i}$. $\boldsymbol{T}_r^{-1} = \frac{\text{adj}(\boldsymbol{T}_r)}{\det(\boldsymbol{T}_r)}$ and $\boldsymbol{T}_m^{-1} = \frac{\text{adj}(\boldsymbol{T}_m)}{\det(\boldsymbol{T}_m)}$, where $\text{adj}(\cdot)$ denotes the adjugate matrix. Given that $\det(\boldsymbol{T}_r) \neq 0$ and $\det(\boldsymbol{T}_m) \neq 0$, the partial derivatives in (7.81) and (7.82) define a system of $2r$ multivariate polynomial equations in $2r$ unknowns ($a_i, b_i, \forall i = 1,\ldots,r$), after "multiplying out" $\det(\boldsymbol{T}_r)$ and $\det(\boldsymbol{T}_m)$. Next, we introduce two auxiliary vectors,

$$\boldsymbol{h} = \boldsymbol{T}_r^{-1}\boldsymbol{f}_r \in \mathbb{R}^{r^2\times 1}, \tag{7.83}$$

$$\boldsymbol{p} = \boldsymbol{T}_m^{-1}\boldsymbol{f}_m \in \mathbb{R}^{(nr)\times 1}, \tag{7.84}$$

to partially linearize these equations. The vectors

$$\boldsymbol{h}^{a_i} = -\boldsymbol{T}_r^{-1}\boldsymbol{T}_r^{a_i}\boldsymbol{h} \in \mathbb{R}^{r^2\times 1}, \tag{7.85}$$

$$\boldsymbol{p}^{a_i} = -\boldsymbol{T}_m^{-1}\boldsymbol{T}_m^{a_i}\boldsymbol{p} \in \mathbb{R}^{(nr)\times 1} \tag{7.86}$$

are the partial derivatives of $\boldsymbol{h}$ and $\boldsymbol{p}$ with respect to the unknown parameters a_i ($\forall i = 1, \ldots, r$), respectively. With these definitions, we can rewrite (7.81) and (7.82) as

$$\frac{\partial \widetilde{\sigma}^2(\boldsymbol{a}, \boldsymbol{b})}{\partial a_i} = -\boldsymbol{g}_r^{\mathrm{T}}\boldsymbol{h}^{a_i} + 2\boldsymbol{g}_m^{\mathrm{T}}\boldsymbol{p}^{a_i} = 0, \tag{7.87}$$

$$\frac{\partial \widetilde{\sigma}^2(\boldsymbol{a}, \boldsymbol{b})}{\partial b_i} = -\boldsymbol{g}_r^{b_i\mathrm{T}}\boldsymbol{h} + 2\boldsymbol{g}_m^{b_i\mathrm{T}}\boldsymbol{p} = 0. \tag{7.88}$$

These partial derivatives, together with the definitions in (7.83) to (7.86), conform a new system of multivariate polynomial equations from which the optimal solution(s) can be retrieved. The common roots of this system of multivariate polynomial equations comprise all the global and local minima as well as all the maxima and saddle points of $\widetilde{\sigma}^2(\boldsymbol{a}, \boldsymbol{b})$ and $\sigma^2(\boldsymbol{a}, \boldsymbol{b})$. In Section 7.4.2.3, we reformulate this system of multivariate polynomial equations as an MEP.

Example 7.6. To illustrate, let us consider the case when $r = 1$, that is, when we look for an $\mathcal{H}_2$-norm optimal first-order approximant of $G(s)$. In this case, $G_r(s)$ only has two parameters,

$$G_r(s) = \frac{b}{s + a}, \tag{7.89}$$

and the polynomial system defined by (7.83) to (7.88) consists of $2n + 4$ multivariate polynomial equations in $2n + 4$ unknowns:

$$2\boldsymbol{g}_m^{\mathrm{T}}\boldsymbol{p}^a - \boldsymbol{g}_r^{\mathrm{T}}\boldsymbol{h}^a = 0, \tag{7.90}$$

$$2\boldsymbol{g}_m^{b\mathrm{T}}\boldsymbol{p} - \boldsymbol{g}_r^{b\mathrm{T}}\boldsymbol{h} = 0, \tag{7.91}$$

$$\boldsymbol{T}_r\boldsymbol{h}^a + \boldsymbol{T}_r^a\boldsymbol{h} = \boldsymbol{0}, \tag{7.92}$$

$$\boldsymbol{T}_m\boldsymbol{p}^a + \boldsymbol{T}_m^a\boldsymbol{p} = \boldsymbol{0}, \tag{7.93}$$

$$\boldsymbol{T}_r\boldsymbol{h} - \boldsymbol{f}_r = \boldsymbol{0}, \tag{7.94}$$

$$\boldsymbol{T}_m\boldsymbol{p} - \boldsymbol{f}_m = \boldsymbol{0}. \tag{7.95}$$

Only two of the unknowns appear not linearly in the polynomial system[9], namely the parameters a and b.

[9]Notice that some of the variables in this system are no longer matrices or vectors in the first-order case. However, to avoid distraction and keep our notation consistent with the higher-order case, we do not remove their boldface appearance.

7.4.2.3 Multiparameter eigenvalue problem

This first-order necessary conditions for optimality (7.83) to (7.88) constitute a system of $r^3 + r^2(n+1) + r(n+2)$ cubic polynomial equations in the same number of unknowns, of which $r^3 + r^2(n+1) + rn$ variables appear linearly in the problem. Given that $\boldsymbol{h}$, $\boldsymbol{p}$, $\boldsymbol{h}^{a_i}$, and $\boldsymbol{p}^{a_i}$, for $i = 1, \ldots, r$, "only appear linearly", we can compactly rewrite this system as a matrix-vector product:

$$\underbrace{\begin{bmatrix} -\left(\boldsymbol{g}_r^{\mathrm{T}}\right)_{\otimes}^{r} & 2\left(\boldsymbol{g}_m^{\mathrm{T}}\right)_{\otimes}^{r} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \left\{-\boldsymbol{g}_r^{b_i\mathrm{T}}\right\}_i & 2\left\{\boldsymbol{g}_m^{b_i\mathrm{T}}\right\}_i & \boldsymbol{0} \\ \left(\boldsymbol{T}_r\right)_{\otimes}^{r} & \boldsymbol{0} & \left\{\boldsymbol{T}_r^{a_i}\right\}_i & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \left(\boldsymbol{T}_m\right)_{\otimes}^{r} & \boldsymbol{0} & \left\{\boldsymbol{T}_m^{a_i}\right\}_i & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{T}_r & \boldsymbol{0} & \boldsymbol{f}_r \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{T}_m & \boldsymbol{f}_m \end{bmatrix}}_{\boldsymbol{\mathcal{M}}(\boldsymbol{a},\boldsymbol{b})} \underbrace{\begin{bmatrix} \left\{\boldsymbol{h}^{a_i}\right\}_i \\ \left\{\boldsymbol{p}^{a_i}\right\}_i \\ \boldsymbol{h} \\ \boldsymbol{p} \\ -1 \end{bmatrix}}_{\boldsymbol{z}} = \boldsymbol{0}, \tag{7.96}$$

where the operator $\left(\cdot\right)_{\otimes}^{s}$ represents the identity matrix Kronecker product $\boldsymbol{I}_s \otimes \left(\cdot\right)$ and the curly brackets $\left\{\boldsymbol{A}_i\right\}_i$ indicate a vertical stack of matrices $\boldsymbol{A}_i$ over the index i, e.g., for $i = 1, 2$, $\left\{\boldsymbol{A}_i\right\}_i = \begin{bmatrix} \boldsymbol{A}_1^{\mathrm{T}} & \boldsymbol{A}_2^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$. The rectangular matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{R}^{k \times l}$ has $2r - 1$ more rows than columns,

$$k = r^3 + r^2(n+1) + r(n+2), \tag{7.97}$$

$$l = r^3 + r^2(n+1) + rn + 1, \tag{7.98}$$

and is a function of the unknown parameters a_i and b_i, which appear quadratically in $\boldsymbol{g}_r$ and linearly in $\boldsymbol{g}_r^{b_i}$, $\boldsymbol{g}_m$, $\boldsymbol{T}_r$, and $\boldsymbol{T}_m$. The matrix-vector product

$$\boldsymbol{\mathcal{M}}(\boldsymbol{a}, \boldsymbol{b})\boldsymbol{z} = \boldsymbol{0} \tag{7.99}$$

is an MEP, where the parameters a_i and b_i constitute the $(2r)$-tuples of eigenvalues $\boldsymbol{\lambda}$ and the vectors $\boldsymbol{h}$, $\boldsymbol{h}^{a_i}$, $\boldsymbol{p}$, and $\boldsymbol{p}^{a_i}$ generate the eigenvectors $\boldsymbol{z}$. By expanding the matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{a}, \boldsymbol{b})$ in terms of the different monomials $\boldsymbol{\lambda}^{\boldsymbol{\omega}}$, where

$$\boldsymbol{\lambda} = (a_1, \ldots, a_r, b_1, \ldots, b_r), \tag{7.100}$$

we obtain the coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}}$ of the MEP. To solve this problem numerically, we rely again on the algorithms developed in Chapter 3.

Example 7.7. For $r = 1$, the matrix-vector product (7.96) corresponds to

$$\begin{bmatrix} -\boldsymbol{g}_r^{\mathrm{T}} & 2\boldsymbol{g}_m^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{g}_r^{b\mathrm{T}} & 2\boldsymbol{g}_m^{b\mathrm{T}} & \boldsymbol{0} \\ \boldsymbol{T}_r & \boldsymbol{0} & \boldsymbol{T}_r^{a} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}_m & \boldsymbol{0} & \boldsymbol{T}_m^{a} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{T}_r & \boldsymbol{0} & \boldsymbol{f}_r \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{T}_m & \boldsymbol{f}_m \end{bmatrix} \begin{bmatrix} \boldsymbol{h}^{a} \\ \boldsymbol{p}^{a} \\ \boldsymbol{h} \\ \boldsymbol{p} \\ -1 \end{bmatrix} = \boldsymbol{0}, \tag{7.101}$$

or, in terms of the model parameters, to

$$\begin{bmatrix} -b^2 & 2b\boldsymbol{C} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -2b & 2\boldsymbol{C} & \boldsymbol{0} \\ -2a & \boldsymbol{0} & -2 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & -a\boldsymbol{I}_n + \boldsymbol{A} & \boldsymbol{0} & -\boldsymbol{I}_n & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -2a & \boldsymbol{0} & 1 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -a\boldsymbol{I}_n + \boldsymbol{A} & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{h}^a \\ \boldsymbol{p}^a \\ \boldsymbol{h} \\ \boldsymbol{p} \\ -1 \end{bmatrix} = \boldsymbol{0}, \tag{7.102}$$

where a and b constitute the 2-tuples of eigenvalues $\boldsymbol{\lambda}$, while $\boldsymbol{h}$, $\boldsymbol{p}$, $\boldsymbol{h}^a$, and $\boldsymbol{p}^a$ generate the eigenvectors $\boldsymbol{z}$.

7.4.3 Numerical example

Now, we present a small numerical proof-of-concept to illustrate the above-described novel model order reduction approach.

Example 7.8. We consider the transfer function

$$G(s) = \frac{s^2 + 9s - 10}{s^3 + 12s^2 + 49s + 78}, \tag{7.103}$$

for which we want to compute the $\mathcal{H}_2$-norm globally optimal first-order approximant

$$G_r(s) = \frac{b}{s + a}. \tag{7.104}$$

For this example, the system of multivariate polynomial equations consists of 10 multivariate polynomial equations in 10 unknowns, of which 8 appear only linearly. This can be reformulated as a quadratic two-parameter eigenvalue problem with coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}} \in \mathbb{R}^{10\times 9}$, where the unknown parameters a and b constitute the eigenvalues $\boldsymbol{\lambda} = (a, b)$ of the problem.

Via the implemented functions in MacaulayLab, we observe that a block Macaulay matrix $\boldsymbol{M} \in \mathbb{R}^{450\times 594}$ of degree $d = 10$ suffices to find the gap in its null space. In this particular example, the nullity does not stabilize, but the nullity change does, which indicates that the solutions at infinity form a one-dimensional variety (Table 7.2). Since we detect 14 linearly independent rows in the regular zone of the null space of $\boldsymbol{M}$, the MEP has $m_a = 14$ affine solutions. Table 7.3 contains these eigenvalues. Only 2 of these solutions lead to stable transfer functions with real coefficients and non-zero numerators, namely

$$G_1(s) = \frac{1.2799}{s + 9.6796} \tag{7.105}$$

and

$$G_2(s) = \frac{-0.0437}{s + 0.2671}. \tag{7.106}$$

The contour plot of the $\mathcal{H}_2$-error for this numerical example is given in Figure 7.7 and the Bode plots of the original model and two first-order models are

Table 7.2. Stabilization diagram for the numerical example, showing the properties of the block Macaulay matrix as a function of its degree d.

degree	size	rank	nullity	nullity change
2	10×54	10	44	/
3	30×90	30	60	16
4	60×135	60	75	15
5	100×189	100	89	14
6	150×252	150	102	13
7	210×324	210	114	12
8	280×405	280	125	11
9	360×495	360	135	10
10	450×594	450	144	9
11	550×702	549	153	9

Table 7.3. Real eigenvalues obtained after solving the model order reduction problem in Example 7.8 for $r = 1$. Only two of the eigenvalues result in a stable reduced-order model $G_r(s)$ with real coefficients and nonzero numerators.

a	b	$\sigma^2(a, b)$
9.6796	1.2799	0.2784
0.2671	−0.0437	0.4082
−16.6189	1.9265	unstable model
−10.0000	0.0000	zero numerator
−6.0000	0.0000	zero numerator
1.0000	0.0000	zero numerator
0.0000	0.0000	zero numerator

given in Figure 7.8. Clearly, the globally optimal first-order approximant of $G(s)$ is $G_1(s)$ (Table 7.3). In order to corroborate the previous results, we used the iterative rational Krylov algorithm (IRKA) [101], available in the sssMOR (sparse state-space and model order reduction) toolbox [57] for Matlab. We observe that, depending on the initialization, the algorithm can converge to one of the two solutions, i.e., (7.105) and (7.106), or to a solution that does not lead to a stable reduced-order model (e.g., $a = -16.6189$ and $b = 1.9265$). Note that we could also directly tackle the polynomial system derived in Example 7.6 via the Macaulay matrix approaches from Chapter 2, but this requires solving a polynomial system with 7 equations in 7 variables.

7.5 System identification

The last key example that we consider in this chapter is the motivational problem used throughout this dissertation. The problem of finding the globally

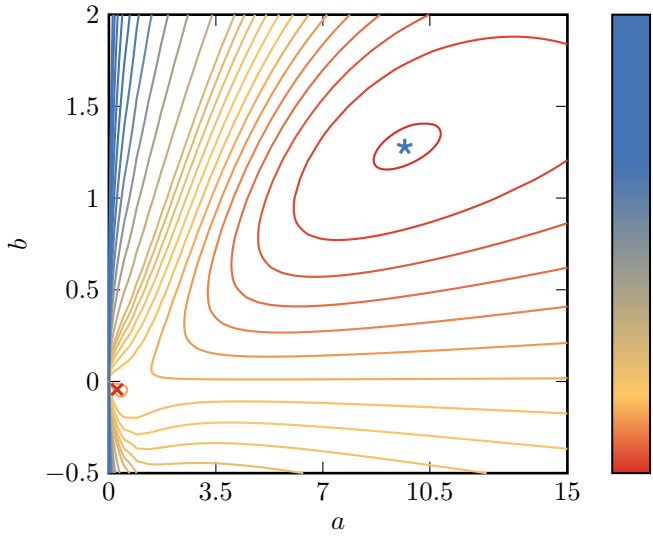


Figure 7.7. Contour plot of the $\mathcal{H}_2$-error $\|G_e(s)\|_{\mathcal{H}_2}$ for Example 7.8. Here, $G_1(s)$ (✱) corresponds to the globally optimal solution and $G_2(s)$ (×) is a local minimizer.

optimal parameters of an ARMA model is a problem from system identification, which aims at constructing models for dynamical systems from measured data [40, 158, 254]. The constructed model tries to capture the relations between input, output, and noise. It depends on a set of model parameters, which are selected to best fit the measured data.

We explore in this section the intimate connection between system identification problems and eigenvalue problems, and we proof that the identification of an ARMA model is an MEP. ARMA models regress an observed output sequence on its own lagged values and on a linear combination of unobserved, latent input samples [40]. In the statistical literature, this sequence of latent inputs is often assumed to be a white Gaussian process [59]. Although our results could be interpreted in an appropriate maximum likelihood framework, we refrain ourselves from those a priori assumptions. ARMA models emerge in a wide variety of domains [40], e.g., in modeling industrial processes, financial time series, or smart utility grid applications (electricity, water, etc.). Moreover, their model structure is an important building block for more sophisticated models [158], e.g., autoregressive moving-average with exogeneous input (ARMAX) models and autoregressive integrated moving-average (ARIMA) models.

Although numerous nonlinear identification techniques for ARMA models already exist, e.g., autocorrelation, penalty function, and innovation regression methods (see the books by Choi [59], Ljung [158], Brockwell and Davis [47]), most of them rely on non-convex numerical optimization and do not guarantee to find the globally optimal model parameters. Stochastic subspace methods, on the other hand, provide a geometric approach by means of projections, which

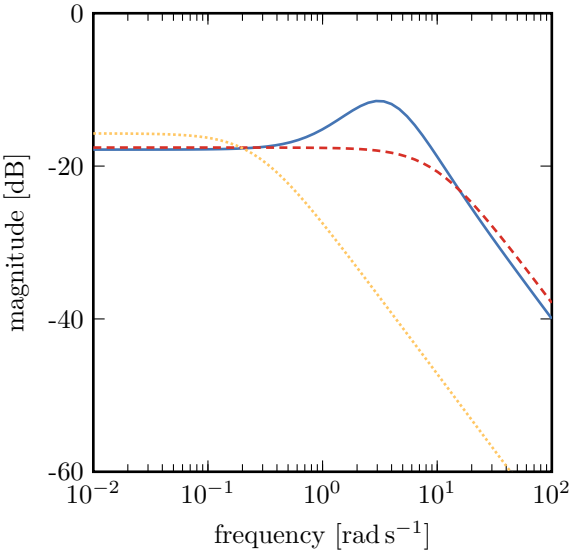


Figure 7.8. Magnitude plot of the Bode diagrams of the original transfer function $G(s)$ (——), the globally optimal first-order approximant $G_1(s)$ (- - -), and the other stable first-order approximant $G_2(s)$ (·····). $G_1(s)$ is also visually a better approximant of $G(s)$ than $G_2(s)$.

work very good in practice, but are not known to be optimal in any sense (see, for example, [253, 254]). Batselier et al. [25] have already approached globally optimal prediction error method identification (and thus also the identification of ARMA models) as an eigenvalue problem. However, they have used the (scalar) Macaulay matrix, which does not exploit the available structure in the problem and scales terribly with the number of output samples.

In this section, we tackle and resolve this hiatus and find the globally optimal least-squares ARMA model parameters using the block Macaulay matrix from Chapter 3. The first-order necessary conditions for optimality of this identification problem constitute a system of multivariate polynomial equations, in which most variables appear linearly. This system is essentially an MEP, which we solve by the block Macaulay matrix. At least one of the eigenvalues corresponds to the global minimum of the original least-squares cost function and thus yields the globally optimal parameters of the ARMA model. Although we focus in this section solely on ARMA models, we want the reader to be aware of the possibility to use this approach for other model classes[10].

[10] In our research group, we do actually not restrict ourselves to ARMA models, but consider the more general misfit-versus-latency model class, which encompasses this ARMA model. The misfit-versus-latency model class was introduced for the first time by Lemmerling and De Moor [155]. It has close links with both Willems' behavioral approach [166, 277–279] and the (structured) total least-squares approach [72, 73]. The framework deals with a broad class of models by allowing not only an unobserved, latent input (cf., latency models, like the ARMA model), but also a misfit on the input and output (cf., misfit models, like the output-error model): an altered input-output sequence must satisfy the imposed model equation,

The remainder of this section proceeds as follows: Firstly, we mathematically formulate the ARMA model structure and set up the ARMA identification problem (Section 7.5.1). We subsequently propose a globally optimal least-squares approach to find the parameters of ARMA models (Section 7.5.2). Finally, we provide a numerical example to support our theoretical derivations (Section 7.5.3).

7.5.1 Problem formulation

A single-input/single-output ARMA model combines a regression of the observed output variable $y_k \in \mathbb{R}$ on its own lagged values y_{k-i} with a linear combination of unobserved, latent inputs $e_{k-r} \in \mathbb{R}$ [40]:

$$\sum_{i=0}^{n_a} \alpha_i y_{k-i} = \sum_{j=0}^{n_c} \gamma_j e_{k-j}, \tag{7.107}$$

where n_a and n_c are the orders of the autoregressive and moving-average part, respectively. The weighting factors $\alpha_i \in \mathbb{R}$, $i = 1, \ldots, n_a$, and $\gamma_j \in \mathbb{R}$, $j = 1, \ldots, n_c$, in the summations are the parameters of the ARMA model. To avoid indeterminacy and without loss of generality, we fix the leading parameters $\alpha_0 = \gamma_0 = 1$.

Given a data sequence of N observed output samples $\boldsymbol{y} \in \mathbb{R}^{N\times 1}$ (not necessarily generated by an ARMA model), we want to find the parameters that satisfy the model structure of (7.107) and minimize the squared 2-norm of the unobserved, latent input vector $\boldsymbol{e} \in \mathbb{R}^{(N-n_a+n_c)\times 1}$, on which we put no a priori unverifiable constraints (like, for example, whiteness or Gaussianity). For this problem, the model structure of (7.107) results in

$$\boldsymbol{T}_\alpha \boldsymbol{y} = \boldsymbol{T}_\gamma \boldsymbol{e}, \tag{7.108}$$

where the two model matrices $\boldsymbol{T}_\alpha \in \mathbb{R}^{(N-n_a)\times N}$ and $\boldsymbol{T}_\gamma \in \mathbb{R}^{(N-n_a)\times(N-n_a+n_c)}$ are banded Toeplitz matrices of appropriate dimensions:

$$\boldsymbol{T}_\alpha = \begin{bmatrix} \alpha_{n_a} & \cdots & \alpha_1 & 1 & 0 & \cdots & 0 \\ 0 & \alpha_{n_a} & \cdots & \alpha_1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_{n_a} & \cdots & \alpha_1 & 1 \end{bmatrix} \tag{7.109}$$

and

$$\boldsymbol{T}_\gamma = \begin{bmatrix} \gamma_{n_c} & \cdots & \gamma_1 & 1 & 0 & \cdots & 0 \\ 0 & \gamma_{n_c} & \cdots & \gamma_1 & 1 & \vdots & \vdots \\ \vdots & \ddots & \ddots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \gamma_{n_c} & \cdots & \gamma_1 & 1 \end{bmatrix}. \tag{7.110}$$

but this adaptation can be achieved via a combination of a latent input and a misfit on the input-output data. Playing around with the hyper-parameters results in very different model classes, see the overview in [155, Table 1]. For certain sets of hyper-parameters, our research group has already published some interesting results, e.g., for the least-squares realization problem [70, 71] and the ARMA model [259]. Also for ARMAX models, we have obtained some (unpublished) results.

The identification problem corresponds to a multivariate polynomial optimization problem in which we minimize the sum of squares of the latent inputs $\sigma^2(\boldsymbol{a}, \boldsymbol{c}) = \|\boldsymbol{e}\|_2^2$, subject to the ARMA model structure of (7.108):

$$\begin{aligned} &\min_{\boldsymbol{a},\boldsymbol{c}} \|\boldsymbol{e}\|_2^2 \\ &\text{subject to } \boldsymbol{T}_\alpha \boldsymbol{y} = \boldsymbol{T}_\gamma \boldsymbol{e}, \end{aligned} \tag{7.111}$$

where the unknown vectors $\boldsymbol{a} \in \mathbb{R}^{n_a \times 1}$ and $\boldsymbol{c} \in \mathbb{R}^{n_c \times 1}$ contain the unknown model parameters α_i and γ_j, respectively.

7.5.2 Globally optimal system identification

This section proves our claim that globally optimal least-squares identification of ARMA models is an MEP. The first-order necessary conditions for optimality constitute a system of multivariate polynomial equations that defines the set of stationary points of the original least-squares cost function (Section 7.5.2.1). Next, we show that this system is basically an MEP (Section 7.5.2.2). At least one of the eigenvalues corresponds to the globally optimal parameters of the ARMA model. It is possible to solve this problem with the block Macaulay matrix algorithms from Chapter 3.

7.5.2.1 First-order necessary conditions for optimality

If the vectors $\boldsymbol{a}$ and $\boldsymbol{c}$ were known, the model structure (7.108) would be a set of under-determined linear equations, the minimum norm solution of which is

$$\boldsymbol{e} = \boldsymbol{T}_\gamma^\dagger \boldsymbol{T}_\alpha \boldsymbol{y}, \tag{7.112}$$

where $\boldsymbol{T}_\gamma^\dagger$ is the pseudo-inverse of the matrix $\boldsymbol{T}_\gamma$. This relationship between the unobserved, latent input vector $\boldsymbol{e}$ and the observed output vector $\boldsymbol{y}$ helps to remove the latent inputs from the least-squares cost function

$$\sigma^2(\boldsymbol{a}, \boldsymbol{c}) = \|\boldsymbol{e}\|_2^2 \tag{7.113}$$

$$= \boldsymbol{e}^{\mathrm{T}} \boldsymbol{e} \tag{7.114}$$

$$= \boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_\alpha^{\mathrm{T}} \boldsymbol{T}_\gamma^{\dagger \mathrm{T}} \boldsymbol{T}_\gamma^\dagger \boldsymbol{T}_\alpha \boldsymbol{y}. \tag{7.115}$$

Since the model matrix $\boldsymbol{T}_\gamma$ is of full row rank, its pseudo-inverse equals

$$\boldsymbol{T}_\gamma^\dagger = \boldsymbol{T}_\gamma^{\mathrm{T}} \left(\boldsymbol{T}_\gamma \boldsymbol{T}_\gamma^{\mathrm{T}} \right)^{-1}, \tag{7.116}$$

where the matrix $\boldsymbol{D}_\gamma = \boldsymbol{T}_\gamma \boldsymbol{T}_\gamma^{\mathrm{T}} \in \mathbb{R}^{(N-n_a)\times(N-n_a)}$ is a symmetric, positive definite, banded Toeplitz matrix. The cost function in (7.115) then reduces to

$$\sigma^2(\boldsymbol{a}, \boldsymbol{c}) = \boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_\alpha^{\mathrm{T}} \boldsymbol{D}_\gamma^{-1} \boldsymbol{T}_\alpha \boldsymbol{y}, \tag{7.117}$$

which has to be minimized over the parameters α_i and γ_j in $\boldsymbol{a}$ and $\boldsymbol{c}$, respectively. The cost function $\sigma^2(\boldsymbol{a}, \boldsymbol{c})$ is clearly nonlinear in the parameters

and its first-order necessary conditions for optimality are ($\forall i = 1, \ldots, n_a$ and $\forall j = 1, \ldots, n_c$):

$$\begin{cases} \dfrac{\partial \sigma^2(\boldsymbol{a}, \boldsymbol{c})}{\partial \alpha_i} = \boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\mathrm{T}} \boldsymbol{D}_{\gamma}^{-1} \boldsymbol{T}_{\alpha}^{\alpha_i} \boldsymbol{y} + \boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\alpha_i \mathrm{T}} \boldsymbol{D}_{\gamma}^{-1} \boldsymbol{T}_{\alpha} \boldsymbol{y} = 0, \\ \dfrac{\partial \sigma^2(\boldsymbol{a}, \boldsymbol{c})}{\partial \gamma_j} = -\boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\mathrm{T}} \boldsymbol{D}_{\gamma}^{-1} \boldsymbol{D}_{\gamma}^{\gamma_j} \boldsymbol{D}_{\gamma}^{-1} \boldsymbol{T}_{\alpha} \boldsymbol{y} = 0, \end{cases} \tag{7.118}$$

where the matrices $\boldsymbol{T}_{\alpha}^{\alpha_i} = \frac{\partial \boldsymbol{T}_{\alpha}}{\partial \alpha_i}$ and $\boldsymbol{D}_{\gamma}^{\gamma_j} = \frac{\partial \boldsymbol{D}_{\gamma}}{\partial \gamma_j}$ contain the element-wise partial derivatives. By introducing an auxiliary vector $\boldsymbol{f} \in \mathbb{R}^{(N-n_a)\times 1}$,

$$\boldsymbol{f} = \boldsymbol{D}_{\gamma}^{-1} \boldsymbol{T}_{\alpha} \boldsymbol{y}, \tag{7.119}$$

we partially linearize this optimization problem. Via its partial derivatives

$$\boldsymbol{f}^{\alpha_i} = \boldsymbol{D}_{\gamma}^{-1} \boldsymbol{T}_{\alpha}^{\alpha_i} \boldsymbol{y}, \tag{7.120}$$

$$\boldsymbol{f}^{\gamma_j} = -\boldsymbol{D}_{\gamma}^{-1} \boldsymbol{D}_{\gamma}^{\gamma_j} \boldsymbol{f}, \tag{7.121}$$

we rewrite (7.118) and obtain ($\forall i = 1, \ldots, n_a$ and $\forall j = 1, \ldots, n_c$)

$$\begin{cases} \dfrac{\partial \sigma^2(\boldsymbol{a}, \boldsymbol{c})}{\partial \alpha_i} = \boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\mathrm{T}} \boldsymbol{f}^{\alpha_i} + \boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\alpha_i \mathrm{T}} \boldsymbol{f} = 0, \\ \dfrac{\partial \sigma^2(\boldsymbol{a}, \boldsymbol{c})}{\partial \gamma_j} = \boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\mathrm{T}} \boldsymbol{f}^{\gamma_j} = 0. \end{cases} \tag{7.122}$$

Finally, the first-order optimality conditions in (7.122), together with the definitions in (7.119) to (7.121), constitute the system of multivariate polynomial equations that defines the set of stationary points of the original least-squares cost function $\sigma^2(\boldsymbol{a}, \boldsymbol{c})$. At least one of the common roots of this system corresponds to the global minimizer of the original multivariate optimization problem, i.e., to the globally optimal least-squares parameters of the ARMA model.

Example 7.9. The first-order necessary conditions for optimality for a first-order ARMA(1,1) model are given by (only model parameters α and γ)

$$\boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\mathrm{T}} \boldsymbol{f}^{\alpha} + \boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\alpha \mathrm{T}} \boldsymbol{f} = 0, \tag{7.123}$$

$$\boldsymbol{y}^{\mathrm{T}} \boldsymbol{T}_{\alpha}^{\mathrm{T}} \boldsymbol{f}^{\gamma} = 0, \tag{7.124}$$

$$\boldsymbol{D}_{\gamma} \boldsymbol{f} - \boldsymbol{T}_{\alpha} \boldsymbol{y} = \boldsymbol{0}, \tag{7.125}$$

$$\boldsymbol{D}_{\gamma} \boldsymbol{f}^{\alpha} - \boldsymbol{T}_{\alpha}^{\alpha} \boldsymbol{y} = \boldsymbol{0}, \tag{7.126}$$

$$\boldsymbol{D}_{\gamma} \boldsymbol{f}^{\gamma} + \boldsymbol{D}_{\gamma}^{\gamma} \boldsymbol{f} = \boldsymbol{0}. \tag{7.127}$$

Only 2 of the unknowns appear not linearly in the polynomial system, namely the two parameters α and β.

7.5.2.2 Multiparameter eigenvalue problem

The system in (7.119) to (7.122) consists of $(N-n_a)(n_a+n_b+1)+n_a+n_c$ cubic multivariate polynomial equations in $(N-n_a)(n_a+n_c+1)+n_a+n_c$ variables, of which $(N-n_a)(n_a+n_c+1)$ variables "appear only linearly" in the problem, a structure that becomes more apparent when we isolate these linear variables in a vector $\boldsymbol{z} \in \mathbb{R}^{l\times 1}$ and rewrite the system as a matrix-vector product

$$\underbrace{\begin{bmatrix} \left(\boldsymbol{g}^{\mathrm{T}}\right)_{\otimes}^{n_a} & \mathbf{0} & \left\{\boldsymbol{g}^{\alpha_i\mathrm{T}}\right\}_i & \mathbf{0} \\ \mathbf{0} & \left(\boldsymbol{g}^{\mathrm{T}}\right)_{\otimes}^{n_a} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{D}_\gamma & \boldsymbol{g} \\ \left(\boldsymbol{D}_\gamma\right)_{\otimes}^{n_a} & \mathbf{0} & \mathbf{0} & \left\{\boldsymbol{g}^{\alpha_i}\right\}_i \\ \mathbf{0} & \left(\boldsymbol{D}_\gamma\right)_{\otimes}^{n_a} & \left\{\boldsymbol{D}_\gamma^{\gamma_j}\right\}_r & \mathbf{0} \end{bmatrix}}_{\boldsymbol{\mathcal{M}}(\boldsymbol{a},\boldsymbol{c})} \underbrace{\begin{bmatrix} \left\{\boldsymbol{f}^{\alpha_i}\right\}_i \\ \left\{\boldsymbol{f}^{\gamma_j}\right\}_r \\ \boldsymbol{f} \\ -1 \end{bmatrix}}_{\boldsymbol{z}}, = \mathbf{0} \tag{7.128}$$

with vectors $\boldsymbol{g} = \boldsymbol{T}_\alpha \boldsymbol{y} \in \mathbb{R}^{(N-n_a)\times 1}$ and $\boldsymbol{g}^{\alpha_i} = \boldsymbol{T}_\alpha^{\alpha_i} \boldsymbol{y} \in \mathbb{R}^{(N-n_a)\times 1}$. The sizes of $\boldsymbol{\mathcal{M}}(\boldsymbol{a},\boldsymbol{c}) \in \mathbb{R}^{k\times l}$ and $\boldsymbol{z} \in \mathbb{R}^{l\times 1}$ are given by

$$k = (N-n_a)(n_a+n_b+1)+n_a+n_c, \tag{7.129}$$

$$l = (N-n_a)(n_a+n_c+1). \tag{7.130}$$

The matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{a},\boldsymbol{c})$ is only a function of the known input-output samples and the unknown model parameters α_i and γ_j. This system of multivariate polynomial equations is basically an MEP, where the nonlinear variables (the parameters α_i and γ_j) constitute the (n_a+n_c)-tuples of eigenvalues $\boldsymbol{\lambda}$ and the linear variables generate the eigenvectors $\boldsymbol{z}$. By expanding the matrix $\boldsymbol{\mathcal{M}}(\boldsymbol{a},\boldsymbol{c})$ in terms of the different monomials $\boldsymbol{\lambda}^{\boldsymbol{\omega}}$, where

$$\boldsymbol{\lambda} = (\alpha_1, \ldots, \alpha_{n_a}, \gamma_1, \ldots, \gamma_{n_c}), \tag{7.131}$$

we obtain the coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}}$ of the MEP.

Example 7.9 (continuing from p. 280). We take again the first-order ARMA model and construct the corresponding MEP that defines the stationary points of the underlying cost function. Simplifying (7.128) results in

$$\begin{bmatrix} \boldsymbol{g} & \mathbf{0} & \boldsymbol{g}^{\alpha} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{g} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{D}_\gamma & \boldsymbol{g}^{\mathrm{T}} \\ \boldsymbol{D}_\gamma & \mathbf{0} & \mathbf{0} & \boldsymbol{g}^{\alpha\mathrm{T}} \\ \mathbf{0} & \boldsymbol{D}_\gamma & \boldsymbol{D}_\gamma^{\gamma} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{f}^{\alpha} \\ \boldsymbol{f}^{\gamma} \\ \boldsymbol{f} \\ -1 \end{bmatrix} = \mathbf{0}. \tag{7.132}$$

This problem is a quadratic two-parameter eigenvalue problem,

$$\left(\boldsymbol{A}_{00} + \boldsymbol{A}_{10}\alpha + \boldsymbol{A}_{01}\gamma + \boldsymbol{A}_{02}\gamma^2\right)\boldsymbol{z} = \mathbf{0}, \tag{7.133}$$

since γ appears quadratically in $\boldsymbol{D}_\gamma$.

7.5.3 Numerical example

In order to illustrate this novel system identification approach, this section provides a numerical example. We show that the block Macaulay matrix approach is able to identify the globally optimal least-squares parameters of a first-order ARMA model.

Example 7.10. We consider a first-order ARMA$(1,1)$ model,

$$y_k + \alpha y_{k-1} = e_k + \gamma e_{k-1}, \tag{7.134}$$

for a sequence of $N = 8$ output samples (without any a priori assumptions):

$$\boldsymbol{y} = \begin{bmatrix} 0.6601 \\ -0.0679 \\ -0.1952 \\ -0.2176 \\ -0.3031 \\ 0.0230 \\ 0.0513 \\ 0.8261 \end{bmatrix}. \tag{7.135}$$

We can use both block Macaulay matrix algorithms to solve the quadratic two-parameter eigenvalue problem in (7.133) with 23×22 coefficient matrices $\boldsymbol{A}_{\boldsymbol{\omega}}$) that yields the optimal model parameters. The solution set of this problem is positive-dimensional at infinity, which means that the nullity of the block Macaulay matrix does not stabilize (there are an infinite number of solutions). Instead of checking the nullity of the block Macaulay matrix for a growing degree d, we monitor the linearly independent rows in a basis matrix of the right null space or linearly dependent columns in the block Macaulay matrix. We observe a gap zone for a degree $d = 43$ block Macaulay matrix. Afterwards, a column compression (null space based approach) or a backward (Q-less) QR decomposition (column space based approach) deflates the positive-dimensional solution set at infinity and we find $m_a = 92$ affine solutions for the MEP. Only three solutions are real and, thus, interesting in this practical setting: we find one minimum and two saddle points (Table 7.4). When we compare this identified minimum with a contour plot (Figure 7.9) of the cost function $\sigma^2(\alpha, \gamma)$, we observe that we indeed have a global minimum within the unit domain $[-1, 1] \times [-1, 1]$. The affine solution with the smallest sum of squares of the latent inputs $\|\boldsymbol{e}\|_2^2$ corresponds to the globally optimal least-squares ARMA model parameters for this given vector of output samples $\boldsymbol{y}$. These parameters result in a smaller $\|\boldsymbol{e}\|_2^2$ than the original parameters and the solution found by the `armax` function of Matlab's system identification toolbox[11].

Next to the system identification aspect, this numerical example highlights another contribution of this dissertation. The full construction of the degree

[11] The `armax` function minimizes the prediction errors in order to find the model parameters. It uses a nonlinear optimization algorithm as described in the book of Ljung [158, Chapter 7].

Table 7.4. Identified real parameters α and γ of the ARMA$(1,1)$ model given by the data in (7.135). The value of the cost function $\sigma^2(\alpha,\gamma)$ in the identified minimum is smaller than in the saddle points and the solution obtained via the `armax` function of Matlab's system identification toolbox.

stationary point	$\boldsymbol{\alpha}$	$\boldsymbol{\gamma}$	$\boldsymbol{\sigma^2(\alpha,\gamma)}$
saddle point	-0.2176	-0.6341	1.0429
saddle point	-0.0267	0.8504	0.9247
minimum	-0.1063	0.1611	0.8016
`armax` function	-0.0939	0.0952	0.8272

Table 7.5. Results of the different combinations of techniques to solve the ARMA$(1,1)$ model identification problem in Example 7.10. The total computation time, the total memory usage, and the maximum absolute residual errors[12]of the real solutions are averaged over 30 experiments.

combination	**time**	**memory**	$\boldsymbol{\max\|e\|_2}$
standard-standard	31 223.95 s	3.62 GB	5.16×10^{-14}
standard-recursive	27 951.57 s	3.62 GB	5.16×10^{-14}
recursive-standard	323.00 s	3.62 GB	1.24×10^{-12}
recursive-recursive	69.41 s	3.62 GB	1.24×10^{-12}
sparse-recursive	41.74 s	24.28 kB	1.48×10^{-13}

$d = 43$ block Macaulay matrix needed to solve the ARMA model identification problem corresponds to a 20769×21780 matrix, and hence, requires 3.62 GB memory. Computing a basis matrix of the right null space of this block Macaulay matrix is a computationally intensive step in the solution approach. The recursive algorithms developed in Chapter 5 are very useful to tackle these MEPs. Table 7.5 shows how the different combinations of recursive techniques influence the computation time, memory usage, and maximum residual error: the sparse-recursive approach is much faster than the standard-standard approach. Furthermore, the sparse adaptation only stores the coefficient matrices of the MEP (24.28 kB) instead of the associated block Macaulay matrix. All computations are done via the functions implemented in MacaulayLab.

7.6 Conclusion

In this chapter, we highlighted several problems that can be tackled via the novel (block) Macaulay matrix algorithms. We focussed on three key examples from systems theory, in which the algorithms of this dissertation are useful to

[12]We calculate the absolute residual error by substituting the computed eigenvalues $(\lambda_1^*, \ldots, \lambda_n^*)$ and eigenvectors $\boldsymbol{z}^*$ in the MEP and determining the 2-norm of the residual vector $\|\boldsymbol{e}\|_2 = \|\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^*)\boldsymbol{z}^*\|_2$. More information about the error measures used in this text can be found in Appendix B.2.3.

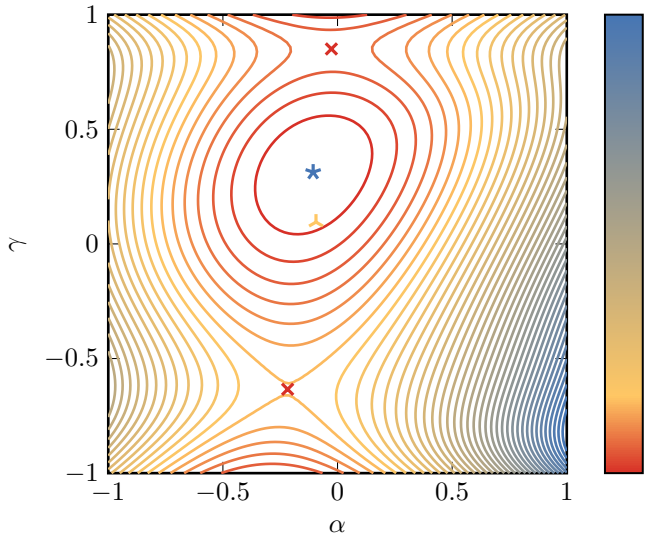


Figure 7.9. Contour plot of the cost function of the ARMA$(1,1)$ model given by the data in (7.135) for the model parameters α and γ in the unit domain $[-1,1] \times [-1,1]$. The value of the cost function σ^2 in the minimum ($*$) is smaller than in the saddle points ($\times$) or in the model obtained via the `armax` function of Matlab's system identification toolbox ($\curlyvee$).

obtain the globally optimal solutions:

- the multivariate polynomial optimization problem in complex variables via Wirtinger derivatives,
- the $\mathcal{H}_2$-norm model order reduction problem for SISO LTI high-order models, and
- the least-squares parameter identification of ARMA models.

We showed that these problems can be rephrased as a system of multivariate polynomial equations or an MEP. Via these systems of multivariate polynomial equations and MEPs, we were able to show that the algorithms that we have developed in this dissertation have a practical use. Although not the goal of this chapter, the reformulation of each of these key examples was a proper contribution to their respective research areas, since the proposed solution methodologies lead to globally optimal solutions to difficult problems that are typically tackled via heuristic techniques.

> **Motivational example.** The reformulation of the ARMA model identification problem in Section 7.5 provides an expression for the coefficient matrices of the quadratic two-parameter eigenvalue problem that we need to solve to obtain the globally optimal model parameters of our motivational example. For a sequence of N output points $\boldsymbol{y}$, the four non-zero coefficient matrices

of the MEP are given by

$$\boldsymbol{A}_{00} = \begin{bmatrix} \overline{\boldsymbol{y}}^{\mathrm{T}} & \boldsymbol{0} & \underline{\boldsymbol{y}}^{\mathrm{T}} & 0 \\ \boldsymbol{0} & \overline{\boldsymbol{y}}^{\mathrm{T}} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_M & \overline{\boldsymbol{y}} \\ \boldsymbol{I}_M & \boldsymbol{0} & \boldsymbol{0} & \underline{\boldsymbol{y}} \\ \boldsymbol{0} & \boldsymbol{I}_M & \boldsymbol{S}_M & \boldsymbol{0} \end{bmatrix}, \boldsymbol{A}_{10} = \begin{bmatrix} \underline{\boldsymbol{y}}^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \underline{\boldsymbol{y}}^{\mathrm{T}} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \underline{\boldsymbol{y}} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix},$$

$$\boldsymbol{A}_{01} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{S}_M & \boldsymbol{0} \\ \boldsymbol{S}_M & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_M & 2\boldsymbol{I}_M & \boldsymbol{0} \end{bmatrix}, \text{ and } \boldsymbol{A}_{02} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & 0 \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I}_M & \boldsymbol{0} \\ \boldsymbol{I}_M & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_M & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix},$$

where $M = N - 1$ and $\boldsymbol{S}_M$ is a $M \times M$ matrix with only ones on its superdiagonal and subdiagonal. The vector of output points $\boldsymbol{y}$ returns in the coefficient matrices via $\overline{\boldsymbol{y}} \in \mathbb{R}^{M \times 1}$, which is $\boldsymbol{y}$ without the first value y_1, and $\underline{\boldsymbol{y}} \in \mathbb{R}^{M \times 1}$, which is $\boldsymbol{y}$ without the last value y_N.

This global optimality, of course, came at a high cost: the resulting problems are computationally hard to solve. For practical problems, the size of the problems quickly grows too large for the standard algorithms. The double recursive algorithms developed in Chapter 5 proved to be very useful, since the recursive and sparse techniques help to (partially) overcome this burden and retrieve the globally optimal solutions. However, it is clear that there is "no such thing as a free lunch". It remains hard to obtain the globally optimal solutions to large-scale problems, even with the double recursive algorithms. In Figure 7.10, we show how the sparse-recursive combination scales for more (random) data points when identifying the parameters of a first-order ARMA$(1, 1)$ model. The required computation time quickly becomes untractable on standard computers. Also for the two approaches of MultiParEig, which scale better with respect to the size of the coefficient matrices, the problem quickly becomes too difficult when increasing the number of data points.

This huge computational complexity of computing the solutions for large-scale applications is a motivation to strive to more efficient solution approaches. The presented reformulations have also invoked a couple of new research challenges:

- A first question when tackling the multivariate optimization problem in complex variables is about the efficiency of this new approach. "What is the computational complexity of this optimization approach and how does it compare to current state-of-the-art solvers for the different applications?" Upper bounds on the computational complexity would be useful for this comparison.

- Closely related with the previous question is the fact that it is not know a priori whether the Macaulay matrix is of a sufficient degree to provide the global minimizers. "Do there exist necessary and sufficient conditions on the degree of the Macaulay matrix in (7.38)?"

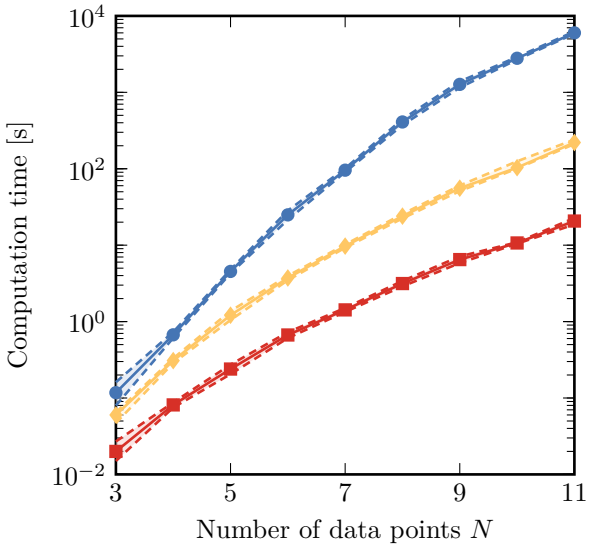


Figure 7.10. Scaling properties of the sparse-recursive approach applied to the first-order ARMA(1, 1) model identification problem for a growing number of (random) data points. The computation time via MacaulayLab (—●—), MultiParEig-CMP (—■—), and MultiParEig-MEP (—◆—) to determine the solutions of the corresponding MEP is displayed. The results are averaged on 30 experiments (the dashed lines indicate one standard deviation).

- The coefficient matrices of the MEPs presented in this chapter exhibit a lot of structure and sparsity (for example, for the first key example, they are constructed via the Macaulay matrix): "Is it possible to exploit the structure and sparsity of the coefficient matrices in the MEP?"
- Furthermore, the solution set of the MEP is also linked to the specific coefficient matrices: "Can we enforce that the MEP has a zero-dimensional solution set?"
- An interesting property of the block Macaulay matrix algorithms to solve MEPs is that a user-defined shift polynomial can be used, as explained in [262]. "Could this property be exploited when the cost function is chosen as shift polynomial?"
- Finally, it is well known that a polynomial's roots can be very sensitive to small changes in the polynomial's coefficients [275]. "Does the MEP approach have better numerical properties than directly solving the system of multivariate polynomial equations that describes the first-order necessary conditions for optimality?"

Furthermore, in the future, we also want to apply our algorithms to more applications. Considering other model classes in the model order reduction or system identification problem are the first steps to take.

Historical and bibliographical notes

We end this chapter with some additional historical and bibliographical notes related to the three key examples of this chapter.

Polynomial optimization problems as eigenvalue problems

Note that reformulating a multivariate polynomial optimization problem in real variables as an (one-parameter) eigenvalue problem is a well-established methodology and there exist several techniques in the optimization literature to obtain such an eigenvalue problem. For example, in the scope of the moment hierarchy approach, the extraction of global minimizers reduces to eigenvalue computations [111, 112, 150, 151]. Furthermore, the use of the first-order necessary conditions for optimality (i.e., the gradient ideal) for polynomial optimization has thoroughly been investigated by, among others, Dreesen and De Moor [81], Nie et al. [189], and Nocedal and Wright [190], and can be combined with a multivariate polynomial system solving approach that resorts on eigenvalue computations (like presented in this dissertation). A reformulation of the cost function as a function of the real and imaginary parts of the complex variables makes it possible to also use these (efficient) numerical optimization techniques in a complex setting.

Different model order reduction methods

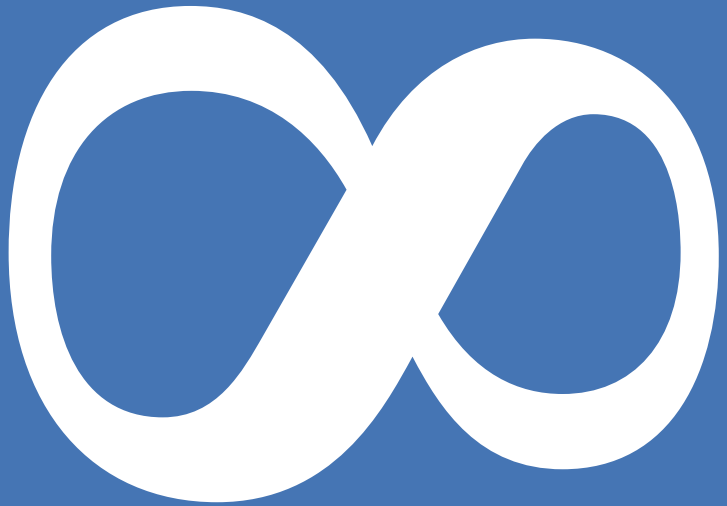
In general, the available methods that address $\mathcal{H}_2$-norm model order reduction can be divided into two main groups: Lyapunov-based methods [227, 282, 285] and interpolation-based methods [7, 10, 101, 102, 171]. Unlike Lyapunov-based methods, which rapidly become infeasible when the dimension increases, interpolation approaches (in which the reduced-order transfer function interpolates the original high-order transfer function at some points in the frequency domain) have proved to be numerically very effective [10]. Although the literature typically makes a distinction between both approaches, the two frameworks are actually equivalent, as shown by Gugercin et al. [101]. Interpolation-based $\mathcal{H}_2$-norm optimality conditions were originally derived by Meier and Luenberger [171] for SISO systems and extended later to the multiple-input/multiple-output (MIMO) case by both Gugercin et al. [101] and Van Dooren et al. [252]. Based on these conditions and results from rational interpolation [28], several iterative numerical algorithms have been proposed (e.g., [7, 51, 102]). However, none of these algorithms are guaranteed to converge to the globally optimal solution, despite the use of several heuristic rules during their initialization. Nonetheless, we can not stress enough their importance in large practical applications.

Inception of autoregressive moving-average models

Already in 1927, Yule [283] proposed a pure autoregressive (AR) process, which only considers a regression of the output sequence on itself. The pure moving-average (MA) model was introduced simultaneously by Yule [283] and Slutzky

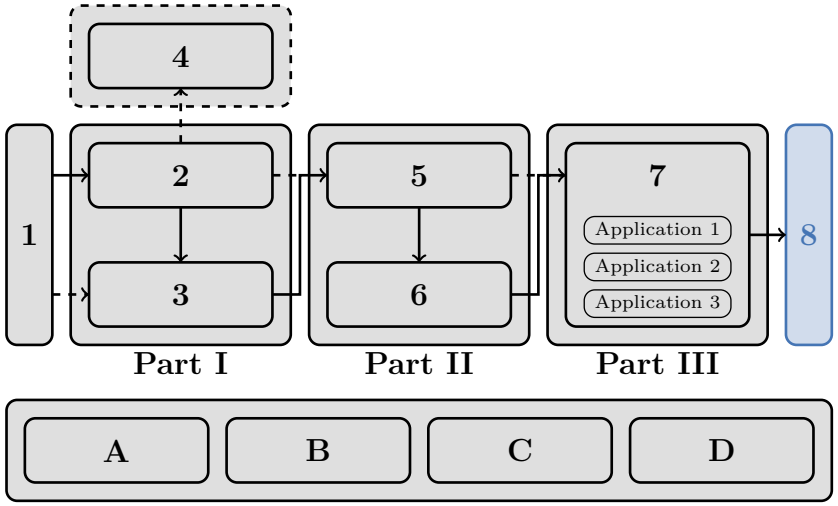
[219]. There exists some dissonance about who was the first to combine these two models, but Whittle [272] and Walker [268] have often been cited as the founding fathers of ARMA modeling. Its popularization, however, was clearly thanks to the famous book by Box and Jenkins [40], who have really propagated these models as a useful tool in time series analysis.

Conclusion



Conclusion and Future Work

In this dissertation, we have tackled both systems of multivariate polynomial equations and rectangular multiparameter eigenvalue problems. The (block) Macaulay matrix has been a central object in that effort. While the traditional (scalar) Macaulay matrix already has a proven track record in polynomial system solving, we have introduced the block Macaulay matrix in this research to address rectangular multiparameter eigenvalue problems. The novel (block) Macaulay matrix approaches can be used to solve both types of problems, proving especially useful when considering the encountered motivational examples from systems theory. We give, in this chapter, an overview of the research contributions of every chapter and highlight several remaining challenges for future research.



Outline. This chapter contains an overview of the dissertation's research contributions (Section 8.1) and highlight some challenges for future research (Section 8.2).

8.1 Research contributions

We give a chapter-by-chapter overview of our research contributions (RCs)[1].

Part I: Fundamentals

In Part I, we introduced the unifying (block) Macaulay matrix approach and developed algorithms to solve systems of multivariate polynomial equations and rectangular multiparameter eigenvalue problems (MEPs). We addressed the different questions of RO1 step-by-step in Part I.

Chapter 2. In Chapter 2, we considered the Macaulay matrix and looked at different approaches to solve systems of multivariate polynomial equations. We revisited the null space based Macaulay matrix approach to solve systems of multivariate polynomial equations and translated the multidimensional realization problem from the right null space to the column space, yielding a complementary column space based Macaulay matrix algorithm to solve systems of multivariate polynomial equations. We also discussed three interesting extensions to both multivariate root-finding approaches: dealing with a positive-dimensional solution set at infinity, using special shift polynomials, and using another monomial ordering or polynomial basis.

RO1.1 Create new Macaulay matrix algorithms to solve systems of multivariate polynomials more efficiently.

↓

RC1.1 We revisited the null space based Macaulay matrix approach and developed the complementary column space based Macaulay matrix approach to solve systems of multivariate polynomial equations.

Chapter 3. While polynomials were the seed problems of Chapter 2, this chapter considered (multiparameter) matrix pencils. Chapter 3 investigated the rectangular MEP, which was, before the exploration within this dissertation, a quite uncharted area. We investigated the rectangular multiparameter eigenvalue problem and its properties. Links between the square and rectangular manifestation were provided in the text. Furthermore, the emergence of rectangular MEP in applications created a need for algorithms to solve them numerically. By considering coefficient matrices instead of coefficients, we introduced in Chapter 3 the so-called block Macaulay matrix. This sparse and structured matrix is the multiple-output extension of the Macaulay matrix and

[1]Recall, from Chapter 1, that all research objectives (ROs) can be mapped onto one of the obtained RCs: every ROx.y leads to an RCx.y.

allowed the development of two complementary approaches to solve polynomial rectangular MEPs in its right null space or column space. The three extensions to the Macaulay matrix approach (cf., RO1.1) were also incorporated in both block Macaulay matrix algorithms: (i) the block Macaulay matrix approaches can deal with a positive-dimensional solution set at infinity, (ii) a special shift function can be used, and (iii) the monomial ordering and polynomial basis for the monomials in the MEP are adjustable. In that sense, Chapter 3 provided an answer to the big challenge of RO1.2.

RO1.2 Extend the Macaulay matrix to the block Macaulay matrix in order to solve polynomial rectangular MEPs.

RC1.2 We developed two block Macaulay matrix approaches to solve rectangular MEPs.

- We explored the properties of the rectangular MEP and compared it with the square problem formulation.
- We extended the Macaulay matrix to its multiple-output generalization, i.e., the block Macaulay matrix.
- Both the right null space and columns space of the block Macaulay matrix were used to retrieve the solutions of the generating MEP.
- The three extensions to the Macaulay matrix approach (cf., RO1.1) were also incorporated in both block Macaulay matrix approaches.

Chapter 4. An important line of research is understanding the Macaulay matrix in an algebraic geometric language. Although a full treatement of the column space is not yet available, we reviewed each of the four fundamental subspaces of the Macaulay matrix in Chapter 4.

RO1.3 Explore the different fundamental subspaces of the Macaulay matrix and relate them to the generating polynomials.

RC1.3 We have summarized the connections between the four different fundamental subspaces of the Macaulay matrix with the (algebraic geometry) properties of the generating polynomials.

Part II: Algorithms

In order to be able to tackle more interesting problems, we developed several recursive techniques that exploit the sparsity and structure of the involved matrices. By combining these different recursive algorithms, together with other important implementation decisions, we obtained a comprehensive toolbox that is user-friendly and fast.

Chapter 5. We developed in Chapter 5 several recursive algorithms to tackle the computational bottlenecks that the naive implementations of the algorithms in Chapters 2 and 3 suffer from. We derived a recursive algorithm to compute a basis matrix of the null space of a block Macaulay matrix. By replacing the coefficient matrices in the block Macaulay matrix by the coefficients of polynomials, this algorithm reduces to an algorithm for the Macaulay matrix, similar to, but more efficient than, the algorithm presented in [22]. We adapted both recursive algorithms such that they are sparse and no longer require an explicit construction of the (block) Macaulay matrix. The recursive nature of these algorithms fits perfectly in the iterative null space based (block) Macaulay approach to solve systems of multivariate polynomial equations or MEPs, where we need to check the structure of the right null space for every iteration in order to know whether we can construct a standard eigenvalue problem that contains the solutions. Furthermore, we also came up with a recursive algorithm to compute a numerical basis matrix of the null space of a block row matrix. This recursive algorithm is, for example, useful to check the rank structure of the obtained numerical basis matrix of the right null space of the (block) Macaulay matrix. A combination of these recursive and sparse algorithms allowed us to solve systems of multivariate polynomial equations and rectangular MEPs more efficiently.

RO2.1 Exploit sparsity and structure in the (block) Macaulay matrix to improve computation time and memory usage.

↓

RC2.1

We developed recursive algorithms that exploit the sparsity and structure of the involved matrices:

- We developed recursive and sparse algorithms to compute a numerical basis matrix of the right null space of a (block) Macaulay matrix.
- Furthermore, we also came up with a recursive algorithm to determine the rank structure of that numerical basis matrix.
- The combination of these two algorithms resulted in double recursive approaches to solve systems of multivariate polynomial equations and rectangular MEPs.

Chapter 6. We have implemented all algorithms from this dissertation in a comprehensive Matlab toolbox, with an eye on the efficiency of the algorithms. Because of the key object in this toolbox, viz., the (block) Macaulay matrix, the toolbox is named MacaulayLab:

- The functions in MacaulayLab exploit the sparsity and structure of the involved matrices, to end up with far more efficient (block) Macaulay matrix solvers than before, while remaining numerically reliable.
- The solution approaches in the toolbox can deal with positive-dimensional solution sets at infinity and work with different shift polynomials.
- MacaulayLab also allows the user to use a different monomial ordering, e.g., graded reverse lexicographic (GREVLEX), and to represent the polynomials in a different polynomial basis, e.g., the multivariate Chebyshev basis.
- The functions of the toolbox are supplemented with many test problems. This database of test problems, both systems of multivariate polynomial equations and multiparameter eigenvalue problems, is useful to test the methods of this and other toolboxes.

We compared the different solution approaches in MacaulayLab and positioned the toolbox with respect to other software packages.

RO2.2 Combine all advancements in one coherent software toolbox that is user-friendly and fast.

↓

RC2.2

We created the MacaulayLab toolbox in Matlab containing efficient implementations of the (block) Macaulay algorithms developed in this dissertation.

- During the implementation, special attention was paid to double recursive algorithms that exploit the sparsity and structure of the involved matrices.
- The option to deal with positive-dimensional solutions sets at infinity and work with different shift polynomials was added to the toolbox. The code was also designed to be independent of the monomial ordering and polynomial basis.
- Many important implementation decisions have been taken, striving for a good trade-off between computation speed an numerical reliability.
- We also included a database with many test problem.

Part III: Applications

The entire dissertation was pushed by a related question: "Can we retrieve the globally optimal solutions to some specific problem from systems theory?" The answer to this question is "Yes, but…" and was tackled in Part III. We focussed in Part III on three applications from systems theory: optimization, model order reduction, and system identification, each looking at one of the suggested problems in RO3. Although each of the discussed globally optimal methodologies is a valuable contribution to its respective research area, we presented them more as a detailed illustration of what we could do with the algorithms of this dissertation.

Chapter 7. We proposed several problems that can be tackled via the novel (block) Macaulay matrix algorithms in Chapter 7. While these algorithms can be applied to a wide range of problem, we focussed on three specific key examples.

- Firstly, we dealt with multivariate polynomial optimization in complex variables. We extended the paper of Sorber et al. [223] to the multivariate case and showed that it is possible to solve these multivariate problems exactly via a reformulation into a rectangular multiparameter eigenvalue problem. Given that the rectangular multiparameter eigenvalue problem can be solved via the block Macaulay matrix, this optimization problem was transformed into a two-layer Macaulay approach.

- Secondly, we rephrased the globally optimal $\mathcal{H}_2$-norm model order reduction problem of a single-input/single-output (SISO) linear time-invariant (LTI) high-order model as a rectangular multiparameter eigenvalue problem. While the original article [3] has posed the problem as an inhomogeneous eigenvalue problem, we reformulated the model order reduction problem as a homogeneous rectangular multiparameter eigenvalue problem (i.e., the problem type considered in this dissertation).

- Thirdly, we showed that the globally optimal least-squares parameter(s) of an autoregressive moving-average (ARMA) model can be found as one of the eigentuples of a rectangular multiparameter eigenvalue problem. While we restricted ourselves to the ARMA problem, the methodology was also extended to other models within the misfit-versus-latency model class.

The reformulation of these three key examples into a system of multivariate polynomial equations or a rectangular multiparameter eigenvalue problem enabled the use of the novel (block) Macaulay matrix algorithms developed in the preceding chapters.

RO3	Use the novel (block) Macaulay matrix algorithms to tackle problems from systems theory with global optimality in mind.

RC3

We showed that it is indeed possible to use the (block) Macaulay matrix to retrieve the globally optimal solutions to three key examples in systems theory.

- We used the Macaulay matrix to write multivariate polynomial optimization problems in complex variables as a rectangular MEP.
- We rephrased the globally optimal $\mathcal{H}_2$-norm model order reduction problem of a SISO LTI high-order model as a rectangular MEP.
- We showed that the globally optimal least-squares identification of ARMA models is a rectangular MEP.

8.2 Future work

The development of the (block) Macaulay matrix algorithms in this dissertation has provided us with a unifying, novel approach for solving systems of multivariate polynomial equations and multiparameter eigenvalue problems. The central matrices of this text are very nice tools to solve problems that emerge, for example, in systems theory. However, we do not consider the work to be done. Einstein [172] once said that

> *"The important thing is to never stop questioning. Curiosity has its own reason for existence. […] Never lose a holy curiosity."*

Indeed, the remarkable, but yet inspiring, thing about research is that answering one question immediately opens up new problems to pursue. Also with creating this unifying block Macaulay matrix approach, many new research questions have popped up: a multitude of "new" research challenges have not been answered in this text. We suggest the following (non-exhaustive) list of future research directions:

Reducing the Macaulay matrices. The Macaulay matrix algorithms to solve systems of multivariate polynomial equations are general-purpose solvers. This means that the algorithms do not consider the structure of the involved polynomials, while polynomials in applications often exhibit structure. One type of structure is the sparsity of the support: not all monomials are present in the support of the polynomials. It could be very beneficial to look at the specific support of the involved polynomials and build smaller Macaulay matrices which only deal with the monomials that are necessary. Reducing the Macaulay matrices is one way to tackle the combinatorial explosion from which our Macaulay matrix algorithms suffer.

Interpreting the subspaces of (block) Macaulay matrices. It is clear from Chapter 4 that the interpretation of the column space is not yet fully understood. Furthermore, the extension of the Macaulay matrix into the block Macaulay matrix leads to the question whether we can do the same for the block Macaulay matrix; Hence, we want to answer the question whether every fundamental subspace has an interpretation in terms of the properties of its generating rectangular MEP.

Exploring other polynomial bases. In, for example, approximation theory, it is a well-known fact that working in another polynomial basis can result in superior numerical properties. A good idea for future work could be a more in-depth analysis of these properties, both with respect to the numerical conditioning and with respect to the computational complexity (since a different polynomial basis induces a different structure of the involved matrices). Using a different polynomial basis is not only a question for systems of multivariate polynomial equations, but could also have implications for multiparameter eigenvalue problems.

Considering the over-determined case. One disadvantage of the very efficient homotopy continuation methods is that they only deal with square systems (i.e., number of equations is equal to the number of variables). The Macaulay matrix algorithms do not have this restriction, so they could be very useful to tackle over-determined systems of multivariate polynomial equations. Furthermore, the linear algebra approach with floating-point work horses suits itself ideally to deal with the possible noise of the experimentally obtained coefficients.

Developing subspace algorithms. Current state-of-the-art algorithms to solve one-parameter eigenvalue problems and square multiparameter eigenvalue problems use projections of large coefficient matrices onto smaller coefficient matrices in order to find some of the eigenvalues/eigentuples faster. Introducing subspace techniques to the block Macaulay matrix algorithms could be a very interesting improvement to our approach to solving rectangular multiparameter eigenvalue problems. First attempts[2] have already delivered very promising results!

Creating more efficient implementations. In any case, the success of the (block) Macaulay matrix algorithms relies on the efficiency of their implementations. Creating more efficient implementations (maybe in a more efficient programming language) could help to tackle much larger applications. The computational efficiency has been a topic that returned in several chapters, so there is clearly a big need for it. Some possible research ideas in this regards are creating recursive algorithms for the column space based approaches, reducing

[2]The development of subspace algorithms for solving rectangular MEPs was a topic of one of the candidate's thesis students. He has shown that we can indeed speed up the computation time of the block Macaulay matrix algorithms by using these subspace techniques, but we still need to tackle some convergence questions.

the size of the (block) Macaulay matrices, considering URV algorithms for rank checks, and using QR decompositions instead of singular value decompositions. Taking inspiration from other toolboxes and looking at the above-mentioned research ideas, it is only natural that future releases of MacaulayLab will contain more efficient implementation and new features. Already in one of the next releases, more support to take the correct rank decisions (i.e., different types of rank decisions, visual aids, etc.) and subspace methods will be added.

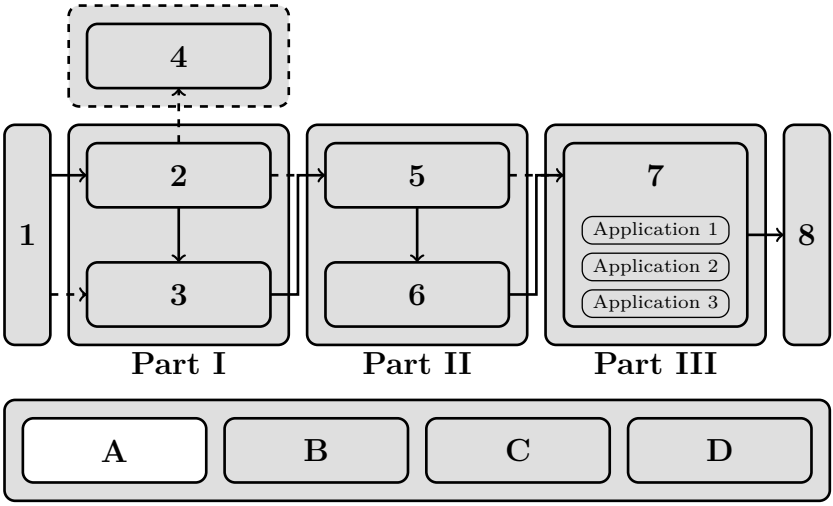
Tackling more applications. We have provided a very interesting approach to solve systems of multivariate polynomial equations and rectangular MEPs. This dissertation has also put forward several applications from systems theory that could be rephrased as one of these fundamental problems. This is only a first attempt to solve problems in globally optimal way. In future work, we want to tackle other model classes, use larger data sequences, exploit structure in the problems, consider different norms in the optimization problems, etc.

Appendices

A

Algebraic Geometry and Commutative Algebra

This chapter supplements the concepts introduced throughout Chapters 2 and 4 and contains some additional topic from algebraic geometry and commutative algebra. Some related textbooks are [65, 66, 205].



Outline. We start by reviewing the definitions of some basic algebraic structures in Appendix A.1. Afterwards, we consider useful elementary definitions from commutative algebra and algebraic geometry in Appendix A.2. In Appendix A.3, we consider bounds on the number of affine solutions for both univariate polynomials and systems of multivariate polynomials, while we introduce Gröbner bases and Buchberger's algorithm in Appendix A.4. Finally, Appendix A.5 sketches Stetter's eigenvalue-eigenvector approach, which is closely related to the approach that we present in this dissertation.

A.1 Group, ring, field, and vector space

Groups, rings, and fields are mathematical objects that share a lot of properties. Every field is also a ring, but not vice versa, while a group can always be found in a ring. A vector space is a set whose elements, often called vectors, can be added and multiplied by scalars, which are elements of a field.

Group. A group consists of a non-empty set A and a binary operation[1] "$\cdot$" defined on A that satisfies the following constraints:

- The operation is associative: $(a \cdot b) \cdot c = a \cdot (b \cdot c), \forall a, b, c \in A$.
- It has an identity element: $\exists 1 \in A : 1 \cdot a = a \cdot 1 = a, \forall a \in A$.
- Every element of the set has an inverse element: $\forall a \in A, \exists b \in A : a \cdot b = b \cdot a = 1$.

In an abelian group, also called a commutative group, the binary operation is also commutative. To qualify as an abelian group, the set and the binary operation, $(A, \cdot)$, must satisfy the abelian group axioms:

- $(A, \cdot)$ is a group.
- The operation is commutative: $\forall a, b \in A : a \cdot b = b \cdot a$.

A group in which the group operation is not commutative is called non-abelian group or non-commutative group. A simple example of a group is given by the set of the integers $\mathbb{Z}$ under addition, i.e., $(\mathbb{Z}, +)$. Note that the set of integers $\mathbb{Z}$ under multiplication, i.e., $(\mathbb{Z}, \times)$, is not a group.

Ring. A ring is an abelian group, whose binary operation is called "addition", with a second binary operation, called the "multiplication", that is associative, is distributive over the addition operation, and has a multiplicative identity element. Thus, formally, a ring is a non-empty set R equipped with two binary operations, the addition "$+$" and multiplication "$\times$", and satisfies three requirements, known as the ring axioms:

- $(R, +)$ is an abelian group under addition (the set and addition satisfy the above-mentioned abelian group axioms).
- R is monoid under multiplication, meaning that the multiplication is associative $(\forall a, b, c \in R : (a \times b) \times c = a \times (b \times c))$ and there exists a multiplicative identity element $(\exists 1 \in R : 1 \times a = a \times 1 = a, \forall a \in R)$.
- The multiplication operation is distributive with respect to the addition operation: $\forall a, b, c \in R : a \times (b + c) = (a \times b) + (a \times c)$ and $\forall a, b, c \in R : (b + c) \times a = (b \times a) + (c \times a)$.

[1] A binary operation is an operation whose input is two elements of the same set A and whose output is also an element of that set A, i.e., $\cdot : A \times A \to A$.

Whether a ring is commutative (that is, whether the order in which two elements are multiplied might change the result) has profound implications on its behavior. These rings are called commutative rings and are very important in algebraic geometry. Examples of commutative rings are the integer numbers $\mathbb{Z}$ and the polynomial ring $K[x_1, \ldots, x_n]$ (this is a ring formed from the set of polynomials in one or more variables with coefficients in another ring K, which is often a field). Note that the set of natural numbers $\mathbb{N}$ with the usual addition and multiplication is not a ring (even not a group), since not all elements are invertible with respect to the addition.

Field. The simplest commutative rings are those that admit division by non-zero elements, such ring are called fields. A field consists of a set K together with two binary operations on that K, called the field operations addition "$+$" and multiplication "$\times$". For K to be a field, these operations are required to satisfy the following properties, referred to the field axioms:

- The addition and multiplication are associative: $(a + b) + c = a + (b + c)$ and $(a \times b) \times c = a \times (b \times c), \forall a, b, c \in K$.
- The addition and multiplication are commutative: $a + b = b + a$ and $a \times b = b \times a, \forall a, b \in K$.
- The operations are distributive: $a \times (b + c) = a \times b + a \times c, \forall a, b, c \in K$.
- There exists an additive and multiplicative identity: $\exists 0, 1 \in K : a + 0 = a \times 1 = a, \forall a \in K$.
- The addition has an additive inverse: $\forall a \in K, \exists b \in K : a + b = 0$.
- The multiplication has a multiplicative inverse: $\forall a \neq 0 \in K, \exists b \in K : a \times b = 1$.

More compact; a field is a commutative ring where $0 \neq 1$ and all non-zero elements are invertible under multiplication. Examples of fields are the rational numbers $\mathbb{Q}$, the real numbers $\mathbb{R}$ and the complex numbers $\mathbb{C}$. The ring of integers $\mathbb{Z}$ is not a field, since the multiplicative inverse (or reciprocal) of an integer is not always itself an integer.

Vector space. A vector space, or linear space, is a set whose elements (i.e., vectors) may be added together and multiplied by numbers (i.e., scalars), which are elements of a field. Formally, a vector space over a field K is a non-empty set V together with two binary operation. The first operation, the (vector) addition "$+$" assigns to any two vectors in V a third vector in V that is the sum of these two vectors. The second operation, the (scalar) multiplication "$\times$", assigns to any scalar in K and any vector in V another vector in V. In order to have a vector space, the set and two binary operations $(V, +, \times)$ must satisfy eight requirements, the vector field axioms:

- Associativity of the vector addition: $\boldsymbol{u}+(\boldsymbol{v} + \boldsymbol{w}) = (\boldsymbol{u} + \boldsymbol{v})+\boldsymbol{w}, \forall \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V$.

- Commutativity of the vector addition: $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}, \forall \boldsymbol{u}, \boldsymbol{v} \in V$.
- Identity element of the vector addition: $\exists \mathbf{0} \in V : \boldsymbol{v} + \mathbf{0} = \boldsymbol{v}, \forall \boldsymbol{v} \in V$.
- Inverse element of vector addition: $\forall \boldsymbol{v} \in V, \exists \boldsymbol{u} \in V : \boldsymbol{v} + \boldsymbol{u} = \mathbf{0}$.
- Compatibility of scalar multiplication with field multiplication: $\forall a, b \in K, \forall \boldsymbol{v} \in K : a \times (b \times \boldsymbol{v}) = (a \times b) \times \boldsymbol{v}$.
- Identity element of scalar multiplication: $\exists 1 \in K, \forall \boldsymbol{v} \in V : 1\boldsymbol{v} = \boldsymbol{v}$.
- Distributivity of scalar multiplication with respect to vector addition: $\forall a \in K, \forall \boldsymbol{u}, \boldsymbol{v} \in V : a \times (\boldsymbol{u} + \boldsymbol{v}) = a \times \boldsymbol{u} + a \times \boldsymbol{v}$.
- Distributivity of scalar multiplication with respect to field addition: $\forall a, b \in K, \forall \boldsymbol{v} \in V : (a + b) \times \boldsymbol{v} = a \times \boldsymbol{v} + b \times \boldsymbol{v}$.

When the scalar field is the complex numbers, the vector space is called a complex vector space. It is clear that the first four requirements (related to vector addition) say that a vector space is an abelian group under addition. The other four requirements (related to the scalar multiplication) are closely related to the field axioms: they say that this operation defines a ring homomorphism from the field into the endomorphism ring of this group.

A.2 Elementary definitions

Definition A.1. Let R and R' be commutative rings with identity element. A **ring homomorphism** is a map $\varphi : R \to R'$ such that for any $f, g \in R$

- $\varphi(f + g) = \varphi(f) + \varphi(g)$,
- $\varphi(fg) = \varphi(f)\varphi(g)$, and
- $\varphi(1) = 1$.

Definition A.2. Let R and A be commutative rings with identity. R is an ***A*-algebra** if there is a ring homomorphism $\varphi : A \to R$. If R and R' are A-algebras with ring homomorphisms $\varphi : A \to R$ and $\varphi' : A \to R'$, then a ring homomorphism $\psi : R \to R'$ is called an A-algebra homomorphism if it satisfies $\psi \circ \varphi = \varphi'$.

Example A.1. The most important A-algebra in this text is the $\mathbb{C}$-algebra, where $A = \mathbb{C}$ and $\varphi : \mathbb{C} \to R$ is the inclusion. An example is the polynomial ring.

For an A-algebra R with homomorphism $\varphi : A \to R$ and $f_1, \ldots, f_s \in R$, we define

$$A[f_1, \ldots, f_s] = \left\{ \sum_{\boldsymbol{\alpha} \in \mathbb{N}^s} \varphi(c_{\boldsymbol{\alpha}}) f^{\boldsymbol{\alpha}} : c_{\boldsymbol{\alpha}} \in A \right\}, \tag{A.1}$$

where the summation is finite and $f^{\boldsymbol{\alpha}} = f_1^{\alpha_1} \cdots f_s^{\alpha_s}$. An A-algebra R with homomorphism φ is finitely generated over A if there is an finite set $\{f_1, \ldots, f_s\} \subset R$ such that $R = A[f_1, \ldots, f_s]$. In this case, the set $\{f_1, \ldots, f_s\}$ is called a set of A-generators of R.

Example A.2. The polynomial ring $\mathcal{P}^n = \mathbb{C}[x_1, \ldots, x_n]$ is finitely generated as a $\mathbb{C}$-algebra: it is generated by the coordinate functions $\{x_1, \ldots, x_s\}$.

Definition A.3. A ring R is called a **Noetherian ring** if all its ideals $\mathcal{I} \subset R$ are finitely generated.

Example A.3. The polynomial ring $\mathcal{P}^n = \mathbb{C}[x_1, \ldots, x_n]$ is Noetherian.

Now, we focus our attention on the polynomial ring $\mathcal{P}^n$ and consider a special set:

$$[p]_{\mathcal{I}} = \{q \in \mathcal{P}^n : p(\boldsymbol{x}) - q(\boldsymbol{x}) \in \mathcal{I}\} \tag{A.2}$$

is the set of all remainders of $p(\boldsymbol{x}) \in \mathcal{P}^n$ modulo the ideal $\mathcal{I}$. It is called the residue class of $p(\boldsymbol{x})$ modulo $\mathcal{I}$. Every polynomial $p(\boldsymbol{x})$ defines such a residue class and we call $p(\boldsymbol{x})$ a representative of $[p]_{\mathcal{I}}$. The set $\mathcal{R}[\mathcal{I}]$ is the set of all residue classes modulo $\mathcal{I}$ and is a vector space over $\mathbb{C}$, since the scalar multiplication and addition operations are well-defined [230]:

$$\alpha[p]_{\mathcal{I}} = [\alpha p]_{\mathcal{I}} \tag{A.3}$$

$$[p]_{\mathcal{I}} + [p]_{\mathcal{I}} = [p + q]_{\mathcal{I}} \tag{A.4}$$

Moreover, since the multiplication $[p]_{\mathcal{I}}[q]_{\mathcal{I}}$ is commutative, $\mathcal{R}[\mathcal{I}]$ is a commutative ring and is often called the quotient ring:

Definition A.4. Let $\mathcal{I} \subset \mathcal{P}^n$ be an ideal. The **quotient ring** of $\mathcal{P}^n$ by $\mathcal{I}$ is the set

$$\mathcal{R}[\mathcal{I}] = \mathcal{P}^n / \mathcal{I} \tag{A.5}$$

modulo the equivalence relation $[p]_{\mathcal{I}} \sim [q]_{\mathcal{I}} \Leftrightarrow p(\boldsymbol{x}) - q(\boldsymbol{x}) \in \mathcal{I}$, with operations

$$\alpha[p]_{\mathcal{I}} = [\alpha p]_{\mathcal{I}} \tag{A.6}$$

$$[p]_{\mathcal{I}} + [p]_{\mathcal{I}} = [p + q]_{\mathcal{I}} \tag{A.7}$$

$$[p]_{\mathcal{I}}[q]_{\mathcal{I}} = [pq]_{\mathcal{I}}. \tag{A.8}$$

If $\mathcal{I} = \mathcal{P}^n$, then $\mathcal{P}^n / \mathcal{I} = \{0\}$ and if $\mathcal{I} = \langle 0 \rangle$ then $\mathcal{P}^n / \mathcal{I} = \mathcal{P}^n$.

A.3 Solution bounds

For a univariate polynomial, the fundamental theorem of algebra gives the number of solutions. When the number of variables $n > 1$, more advanced results come into the picture. We consider in this section solution bounds for both univariate polynomials (Appendix A.3.1) and systems of multivariate polynomials (Appendix A.3.2).

A.3.1 Univariate polynomials

When $n = 1$ in Definition 2.3, we have a univariate polynomial

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{d-1} x^{d-1} + c_d x^d \in \mathcal{P}_d^1. \tag{A.9}$$

Solving a univariate polynomial amounts to finding the solutions of the univariate polynomial equation $p(x) = 0$, i.e., the roots of the univariate polynomial $p(x)$. A standard linear algebra approach to solve a univariate polynomial (in the standard monomial basis) is via the eigenvalues of its companion matrix[2]. For a univariate polynomial $p(x)$ in the standard monomial basis with total degree d, the roots are equal to the eigenvalues of the $d \times d$ companion matrix

$$\boldsymbol{C}_p = \begin{bmatrix} 0 & \cdots & 0 & -\frac{c_0}{c_d} \\ 1 & \cdots & 0 & -\frac{c_1}{c_d} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 1 & -\frac{c_{d-1}}{c_d} \end{bmatrix}, \tag{A.10}$$

whose characteristic polynomial is

$$\det(x\boldsymbol{I} - \boldsymbol{C}_p) = c_d^{-1} p(x). \tag{A.11}$$

Example A.4. If we want to find the solutions of the polynomial equation

$$p(x) = -6 + 5x + 5x^2 - 5x^3 + x^4 = 0, \tag{A.12}$$

then we can compute the eigenvalues of the 4×4 companion matrix

$$\boldsymbol{C}_p = \begin{bmatrix} 0 & 0 & 0 & -6 \\ 1 & 0 & 0 & -5 \\ 0 & 1 & 0 & -5 \\ 0 & 0 & 1 & 5 \end{bmatrix}, \tag{A.13}$$

which are -1, 1, 2, and 3.

[2] When the univariate polynomial is given in another polynomial basis than the standard monomial basis, then other companion-like matrices can be used to find the roots of the polynomial. This family of matrices is called *comrade matrices*, and, in particular, when using the Chebyshev basis, we talk about the *colleague matrix* [18, 98, 228].

The number of solutions/roots in the univariate case follows from a well-known theorem[3].

Theorem A.1 (Fundamental theorem of algebra). Every univariate polynomial $p(x) \in \mathcal{P}_d^1$ of degree d has exactly d roots, counted with multiplicities, in the complex field $\mathbb{C}$.

Example A.5. A quadratic polynomial equation ($d = 2$),

$$c_0 + c_1 x + c_2 x^2 = 0, \tag{A.14}$$

has two (complex) solutions, given by

$$x|_{(1)} = \frac{-c_1 + \sqrt{c_1^2 - 4c_0c_2}}{2c_2} \quad \text{and} \quad x|_{(2)} = \frac{-c_1 - \sqrt{c_1^2 - 4c_0c_2}}{2c_2}. \tag{A.15}$$

Example A.6. The polynomial equation

$$x^d - 1 = 0 \tag{A.16}$$

has d solutions, which are called the *roots of unity* or the *de Moivre numbers*,

$$x|_{(k)} = e^{\frac{2k\pi i}{d}}, \tag{A.17}$$

for $k = 0, \ldots, d-1$.

Unfortunately, there are no such expressions in radicals for the roots of a general polynomial[4]. We can rephrase Theorem A.1 for all univariate polynomials $p(x) \in \mathcal{P}_d^1$ as the following corollary:

Corollary. Every univariate polynomial $p(x) \in \mathcal{P}_d^1$ has at most d distinct roots in the complex field $\mathbb{C}$.

[3]This theorem was proven by Carl Friedrich Gauss in 1799, although the proof contained a topological gap [220]. Sometimes the fundamental theorem of algebra is phrased a little bit differently: "Every non-constant polynomial $p(x) \in \mathcal{P}^1$ has a root in $\mathbb{C}$."

[4]The Abel–Ruffini theorem (also known as Abel's impossibility theorem) states that there is no solution in radicals to general univariate polynomial equations of degree five or higher with arbitrary coefficients [15, 209]. This theorem is named after Paolo Ruffini, who made an incomplete proof in 1799 (refined and accepted by Cauchy in 1813), and Niels Henrik Abel, who provided a proof in 1824. The Abel–Ruffini theorem refers also to the slightly stronger result that there are equations of degree five and higher that cannot be solved by radicals. This does not follow from Abel's statement of the theorem, but is a corollary of his proof, as his proof is based on the fact that some polynomials in the coefficients of the equation are not the zero polynomial. This improved statement follows directly from Galois theory, which says that $x^5 - x - 1 = 0$ is the simplest equation that cannot be solved in radicals, and that almost all polynomials of degree five or higher can not be solved in radicals.

This corollary gives an upper bound for the number of affine solutions of a univariate polynomial. This upper bound is met for generic polynomials, but there exists a subset of non-generic polynomials, for which the upper bound is not met: polynomials $p(x)$ for which the coefficients are in the affine variety $\nabla_d = \mathcal{V}(\Delta_d)$ are non-generic. Δ_d is a polynomial in the coefficients c_i of $p(x)$. Some examples of this polynomial Δ_d are [65, 241]

$$\Delta_1 = c_1, \tag{A.18}$$
$$\Delta_2 = c_2\big(c_1^2 - 4c_0c_2\big), \tag{A.19}$$
$$\Delta_3 = c_3\big(c_1^2c_2^2 - 4c_0c_2^3 - 4c_1^3c_3 + 18c_0c_1c_2c_3 - 27c_0^2c_3^2\big), \tag{A.20}$$
$$\Delta_4 = c_4\big(c_1^2c_2^2c_3^2 - 4c_0c_2^3c_3^2 - 4c_1^3c_3^3 + 18c_0c_1c_2c_3^3 + \ldots + 256c_0^3c_4^3\big). \tag{A.21}$$

One notices that $\Delta_d = c_d\widetilde{\Delta}_d$, with $\widetilde{\Delta}_d$ the discriminant for degree d polynomials. A well-known case is of course the discriminant of a quadratic polynomial $\tilde{\Delta}_2$. Furthermore, it is immediately clear that a polynomial in $\mathcal{P}_d^1$ is non-generic when the polynomial is not of degree d (or when $c_d = 0$).

Example A.7. To illustrate the previous equations, we consider three quadratic polynomials:

- $p_1(x) = 2 + 4x + x^2$ is clearly a generic polynomial, because $\Delta_2 \neq 0$, and has two distinct affine solutions.
- $p_2(x) = 1 + 2x + x^2$ is a non-generic polynomial. Since $c_1^2 - 4c_0c_2 = 0$, the coefficients of the polynomial are in the affine variety $\mathcal{V}(\Delta_2)$. This polynomial has an affine solution with multiplicity equal to 2.
- $p_3(x) = 1+2x$ is also a non-generic quadratic polynomial. Since $c_2 = 0$, the coefficients of the polynomial are in the affine variety $\mathcal{V}(\Delta_2)$. This polynomial has one affine solution and one solution at infinity.

A.3.2 Multivariate polynomial systems

The multivariate extension of the fundamental theorem of algebra is Bézout's theorem.

Theorem A.2 (Bézout's theorem). For any square system (i.e., $s = n$) of multivariate polynomial equations $p_1(\boldsymbol{x}), \ldots, p_n(\boldsymbol{x}) \in \mathcal{P}^n$, the number of isolated solutions in the projective space $\mathbb{P}^n$ when the solution set is zero-dimensional, i.e., the number of isolated points in the zero-dimensional variety $\mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_n(\boldsymbol{x})) \subset \mathbb{P}^n$, is exactly equal to

$$m_b = d_1 \cdots d_n = \prod_{i=1}^{n} d_i, \tag{A.22}$$

where d_i is the total degree of the polynomial $p_i(\boldsymbol{x})$.

Proof. A proof of this theorem can be found in [82, Theorem III-71]. □

Theorem A.2 is an important result and counts the total number of isolated solutions of a square system of n multivariate polynomial equations in n variables. Theorem 2.2 can be phrased as a corollary of the previously stated version of Bézout's theorem.

Corollary. For any square system (i.e., $s = n$) of multivariate polynomial equations $p_1(\boldsymbol{x}), \ldots, p_n(\boldsymbol{x})$, the number of affine, isolated solutions when the solution set is zero-dimensional, i.e., the number of isolated points in the zero-dimensional affine variety $\mathcal{V}(p_1(\boldsymbol{x}), \ldots, p_n(\boldsymbol{x})) \subset \mathbb{C}^n$, is at most equal to

$$m_a = d_1 \cdots d_n = \prod_{i=1}^{n} d_i, \tag{A.23}$$

where d_i is the total degree of the polynomial $p_i(\boldsymbol{x})$.

The Bézout bound is almost always tight, in the sense that most systems have $m_b = m_a$ affine solutions. Unfortunately, many applications lead to polynomial systems with less affine solutions. This has led to more tight, advanced bounds on the number of isolated affine solutions, like Kushnirenko's bound and the Bernstein–Khovanskii–Kushnirenko (BKK) bound. An important observation is that there is a deep connection between lattice/Newton polytopes and polynomials.

Definition A.5. The **Newton polytope** of a polynomial $p(\boldsymbol{x})$ with support $\text{supp}(p(\boldsymbol{x})) = \mathcal{A}$, denoted by $\text{NP}(p(\boldsymbol{x}))$, is the lattice polytope

$$\text{NP}(p(\boldsymbol{x})) = \text{Conv}(\mathcal{A}), \tag{A.24}$$

where $\text{Conv}(\mathcal{A})$ is the convex hull of the support.

In words: the Newton polytope of a polynomial records the "shape" or "sparsity structure" of a polynomial, it tells us which monomials appear in its support. The actual values of the coefficients $c_{\boldsymbol{\alpha}}$ do not matter, since different polynomials can have the same support. Note that, not only different polynomials with the same support, but also polynomials with a slightly different support can have the same Newton polytope, when the the convex hulls of their support are identical.

Example A.8. Consider the multivariate polynomial

$$p(\boldsymbol{x}) = 1 + 2x_1^2 x_2 + 3x_1 x_2^2, \tag{A.25}$$

which has the support

$$\mathcal{A} = \{(0,0), (2,1), (1,2)\}. \tag{A.26}$$

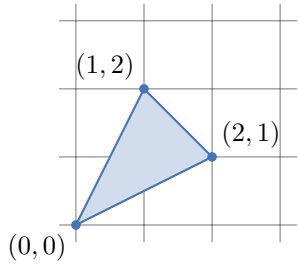


Figure A.1. Visualization of the Newton polytope of the polynomial $p(\boldsymbol{x})$ in Example A.8. The dots represent the monomials in its support.

The Newton polytope of $p(\boldsymbol{x})$ is

$$\mathrm{NP}(p(\boldsymbol{x})) = \mathrm{Conv}(\{(0,0),(2,1),(1,2)\}). \tag{A.27}$$

Figure A.1 visualizes the Newton polytope $\mathrm{NP}(p(\boldsymbol{x}))$ and support $\mathcal{A}$ on a two-dimensional grid. Every monomial in the support corresponds to one dot indicated on the grid. The polynomial

$$q(\boldsymbol{x}) = 1 + 2x_1x_2 + 3x_1^2x_2 + 4x_1x_2^2 \tag{A.28}$$

has the same Newton polytope as $p(\boldsymbol{x})$, since the monomial x_1x_2 lies within the convex hull of $\mathcal{A}$.

In applications, the polynomials often miss some variables, which means that their support is not full. Kushnirenko's theorem counts the number of affine solutions for a system of multivariate polynomial equations in the family

$$\mathcal{F}^n(\mathcal{A}) = \{p_1(\boldsymbol{x}) = \cdots = p_n(\boldsymbol{x}) = 0 : \mathrm{supp}(p_i(\boldsymbol{x})) \subseteq \mathcal{A}\}, \tag{A.29}$$

which are the polynomials with a support that is a subset of $\mathcal{A}$. The theorem expresses this in terms of the volume $\mathrm{Vol}(\mathcal{A}) = \int_{\mathrm{Conv}(\mathcal{A})} d_{\alpha_1} \cdots d_{\alpha_n}$ of the convex polytope

$$\mathrm{Conv}(\mathcal{A}) = \left\{ \sum_{\boldsymbol{\alpha}\in\mathcal{A}} \lambda_{\boldsymbol{\alpha}}\boldsymbol{\alpha} : \lambda_{\boldsymbol{\alpha}} \geq 0, \sum_{\boldsymbol{\alpha}\in\mathcal{A}} \lambda_{\boldsymbol{\alpha}}\boldsymbol{\alpha} = 1 \right\} \subset \mathbb{R}^n. \tag{A.30}$$

The normalized volume $\mathrm{vol}(\mathcal{A})$ is defined as $n!\mathrm{Vol}(\mathcal{A})$. The link with the Newton polytopes of the polynomials in $\mathcal{F}^n(\mathcal{A})$ is easy to see in the previous equation.

Theorem A.3 (Kushnirenko's theorem). For any square system $\mathcal{S} \in \mathcal{F}^n(\mathcal{A})$, the number of isolated, affine solutions, i.e., the number of isolated points in the affine variety, is at most $\mathrm{vol}(\mathcal{A})$. Moreover, there exists a proper variety $\nabla_{\mathcal{A}}$ such that, when $\mathcal{S} \in \mathcal{F}^n(\mathcal{A}) \setminus \nabla_{\mathcal{A}}$, the variety consists of precisely $\mathrm{vol}(\mathcal{A})$ isolated, affine points.

Proof. A proof of this theorem can be found in [147]. □

Example A.9. To illustrate the implications of the previous theorem, we consider a system of two bivariate polynomial equations

$$\begin{cases} p_1(\boldsymbol{x}) = 23x_1^{16}x_2^{14} - x_1^{22}x_2^{18} + 27 = 0, \\ p_2(\boldsymbol{x}) = 35x_1^{16}x_2^{14} - x_1^{22}x_2^{18} + 9 = 0, \end{cases} \tag{A.31}$$

which both have support $\mathcal{A} = \{(16, 14), (22, 18), (0, 0)\}$ and maximum total degree $d_i = 40$. Bézout's theorem indicates that this polynomial system has $m_b = 40^2 = 1600$ solutions, but via Kushnirenko's theorem we know that the system only has $m_a = 20$ affine solutions (the bound is tight in this example).

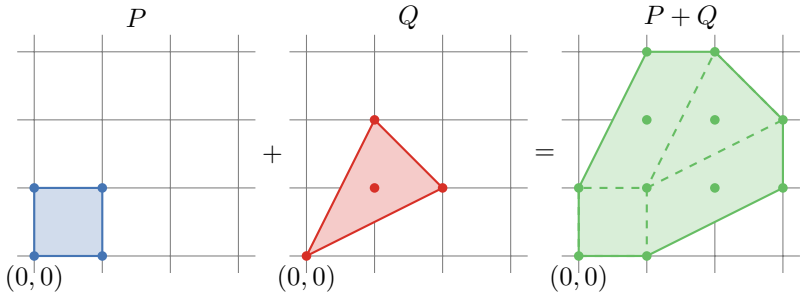
Code A.1. It is possible to compute Kushnirenko's bound via MacaulayLab:

```
>> p1 = [23 16 14; -1 22 18; 27 0 0];
>> p2 = [35 16 14; -1 22 18; 9 0 0];
>> mb = bezout(systemstruct({p1, p2}))

mb =
    1600

>> ma = kushnirenko(systemstruct({p1, p2}))

ma =
    20
```

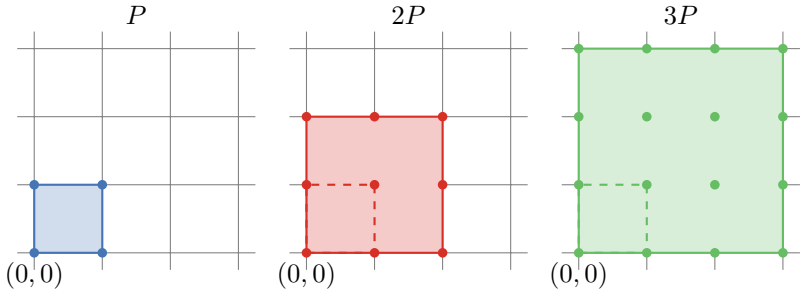
Remark A.1. Theorem A.3 necessarily counts solutions in $\mathbb{C}^n \setminus \{\mathbf{0}\}$, as multiplying all equations with a monomial $\boldsymbol{x}^{\boldsymbol{\alpha}}$ may change the number of solutions in the coordinate hyperplanes (i.e., there may be new solutions with zero-coordinates), but it does not change the normalized volume $\mathrm{vol}(\mathcal{A})$. The statement can be adapted to count solutions in $\mathbb{C}^n$, but then becomes more involved [125].

Remark A.2. If $\mathcal{A} = \{\boldsymbol{\alpha} \in \mathbb{N}^n : |\boldsymbol{\alpha}| \leq d\}$, we have that $\mathcal{F}^n(\mathcal{A}) = \mathcal{P}_d^n$ and $\mathrm{vol}(\mathcal{A}) = d^n$. In that situation, Theorem A.3 coincides with Theorem A.2 for $d_1 = \cdots = d_n = d$.

There exists a generalization of Kushnirenko's theorem to unequal supports (i.e., the polynomials can have different supports), which is called Bernstein's



(a) Minkowski sum



(b) δ-dilation

Figure A.2. Visualization of the Minkowski sum of two Newton polytopes and the δ-dilations for $\delta = 2$ and $\delta = 3$ of a Newton polytope.

theorem. Let us fix n subsets of exponents $\mathcal{A}_1, \ldots, \mathcal{A}_n$ with cardinalities $|\mathcal{A}_i|$ and define the family of systems of polynomial equations

$$\mathcal{F}^n(\mathcal{A}_1, \ldots, \mathcal{A}_n) = \{p_1(\boldsymbol{x}) = \cdots = p_n(\boldsymbol{x}) = 0 : \operatorname{supp}(p_i(\boldsymbol{x})) = \mathcal{A}_i\}. \quad \text{(A.32)}$$

The number of solutions is characterized by the *mixed volume* of these subsets $\mathcal{A}_1, \ldots, \mathcal{A}_n$. In order to compute the mixed volume, we need to define first the *Minkowski sum* and the *δ-dilation* (see Figure A.2): the Minkowski sum of two sets $S, T \subset \mathbb{R}^n$ as $\{s + t : s \in S, t \in T\}$, where $s + t$ is the usual addition of vectors in $\mathbb{R}^n$, while the δ-dilation δS of a set $S \subset \mathbb{R}^n$ is $\{\delta s : s \in S\}$, where δs is the usual scalar multiplication in $\mathbb{R}^n$. Each of the supports $\mathcal{A}_i$ corresponds to a convex polytope $\operatorname{Conv}(\mathcal{A}_i)$ as in (A.30). If we consider the homogeneous polynomial of degree n

$$\begin{gathered} f(\lambda_1, \ldots, \lambda_n) : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0} : \\ (\lambda_1, \ldots, \lambda_n) \to \operatorname{Vol}(\lambda_1 \operatorname{Conv}(\mathcal{A}_1) + \cdots + \lambda_n \operatorname{Conv}(\mathcal{A}_n)), \end{gathered} \quad \text{(A.33)}$$

then the mixed volume $\operatorname{mvol}(\lambda_1, \ldots, \lambda_n)$ is the coefficient of that polynomial associated with the monomial $\lambda_1 \cdots \lambda_n$.

Theorem A.4 (Bernstein's theorem). For any square $\mathcal{S} \in \mathcal{F}^n(\mathcal{A}_1, \ldots, \mathcal{A}_n)$, the number of isolated, affine solutions, i.e., the number of isolated points in the affine variety, is at most $\text{mvol}(\lambda_1, \ldots, \lambda_n)$. Moreover, there exists a proper variety $\nabla_{\mathcal{A}_1,\ldots,\mathcal{A}_n}$ such that, when $\mathcal{S} \in \mathcal{P}(\mathcal{A}_1, \ldots, \mathcal{A}_n) \setminus \nabla_{\mathcal{A}_1,\ldots,\mathcal{A}_n}$, the variety consists of precisely $\text{mvol}(\mathcal{A}_1, \ldots, \mathcal{A}_n)$ isolated affine points.

This theorem was originally proved by Bernstein [33] for the field of complex numbers. The proof by Kushnirenko [147] works for all algebraically closed fields and several alternative proofs were found by Khovanskii. Therefore, Theorem A.4 is sometimes called the BKK theorem and the bound on the number of solutions the BKK bound.

Example A.10. Let us consider the system of two bivariate polynomial equations

$$\begin{cases} p_1(\boldsymbol{x}) = ax_1^3x_2^2 + bx_1 + cx_2^2 + d = 0, \\ p_2(\boldsymbol{x}) = ex_1x_2^4 + fx_1^3 + gx_2 = 0, \end{cases} \tag{A.34}$$

where a, b, c, d, e, f, and g are coefficients in $\mathbb{C}$. The Bézout bound of this system is equal to $m_b = 5 \cdot 5 = 25$, which is not tight. However, the mixed volume is equal to 18, which corresponds exactly to the number of affine solutions of the system (when the coefficients are random scalars).

Code A.2. Also the BKK bound can be determined via MacaulayLab. The code below computes the Bézout bound and BKK bound for a system with a special support and random coefficients.

```
>> c = randn(7,1);
>> p1 = [c(1) 3 2; c(2) 1 0; c(3) 0 2; c(4) 0 0];
>> p2 = [c(5) 1 4; c(6) 3 0; c(7) 0 1];
>> mb = bezout(systemstruct({p1, p2}))

mb =
    25

>> ma = bkk(systemstruct({p1, p2}))

ma =
    18
```

Remark A.3. Like Theorem A.3, Theorem A.4 counts solutions in $\mathbb{C}^n \setminus \{\mathbf{0}\}$. The statement can also be adapted to count solutions in $\mathbb{C}^n$ [125].

Theorem A.4 provides an upper bound on the number of isolated, affine solutions to any system of polynomial equations with $n = s$. Although it significantly improves the upper bound of Theorem A.2, it still often happens in applications that the bound is not tight. There exist even more refined solution counts, but these are far out of the scope of this text.

A.4 Gröbner bases and Buchberger's algorithm

The Gröbner basis algorithm, more commonly referred to as Buchberger's algorithm, computes a Gröbner basis for a set of polynomial equations. Loosely speaking, the algorithm can be understood as the polynomial generalization of the Gaussian elimination algorithm or as the multivariate generalization of the greatest common divisor algorithm [78]. In essence, Buchberger's algorithm proceeds by manipulating the given set of polynomial equations with the objective of elimination certain terms, while at all times the "new" set of equations is algebraically equivalent to the original set, i.e., the define the same ideal.

A typical way to do this is to aim at finding a triangular structure: one or more equations would be univariate, then some equations would involve two or a few variables, until the most complicated equation involves (almost) all variables. This requires that one defines an ordering on the monomials to decide in what order the terms should be eliminated. This triangular structure greatly simplifies the task of solving the system of multivariate polynomial equations. For example, if we use a lexicographic (LEX) ordering, one often end up with polynomials that are univariate. After the univariate equation is solved (via an efficient univariate root-finding algorithm, like the companion matrix), the solutions can be substituted in the next equations which has two unknowns, after which only one unknown remains and it can be solved again, etc. By iterating this way, all unknowns are ultimately determined. The properties of a Gröbner basis with different monomial ordering can be very different.

A.4.1 Multivariate division

As mentioned in Section 2.2, there exist different ways to order multivariate monomials. The literature contains a number of different orderings (see, for example, [66]), two of them are relevant in this section:

Definition A.6. Let $\boldsymbol{x}^{\boldsymbol{\alpha}}$ and $\boldsymbol{x}^{\boldsymbol{\beta}}$ be monomials in $\mathcal{C}^n$. We say that $\boldsymbol{x}^{\boldsymbol{\alpha}} > \boldsymbol{x}^{\boldsymbol{\beta}}$ in the **lexicographic (LEX) ordering** when in the difference $\boldsymbol{\alpha} - \boldsymbol{\beta} \in \mathbb{Z}^n$, the leftmost nonzero entry is positive.

Definition A.7. Let $\boldsymbol{x}^{\boldsymbol{\alpha}}$ and $\boldsymbol{x}^{\boldsymbol{\beta}}$ be monomials in $\mathcal{C}^n$. We say that $\boldsymbol{x}^{\boldsymbol{\alpha}} > \boldsymbol{x}^{\boldsymbol{\beta}}$ in the **graded lexicographic (GRLEX) ordering** when in the difference $|\boldsymbol{\alpha}| > |\boldsymbol{\beta}| \in \mathbb{Z}^n$, or when $\boldsymbol{\alpha} - \boldsymbol{\beta} \in \mathbb{Z}^n$ for $|\boldsymbol{\alpha}| = |\boldsymbol{\beta}| \in \mathbb{Z}^n$.

Let us assume that a monomial ordering is chosen, expressed by $>$, and let us now consider the terms appearing in a given polynomial $p(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}$.

Algorithm A.1 Multivariate division algorithm

Require: $q(\boldsymbol{x}), p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$
$a_1(\boldsymbol{x}) = \cdots = a_s(\boldsymbol{x}) = r(\boldsymbol{x}) = 0$
while $q(\boldsymbol{x}) \neq 0$ **do**
  $i = 1$
  flag $\leftarrow$ false
  while $i \leq s$ and flag = false **do**
    if flag = false **then**
      $a_i(\boldsymbol{x}) = a_i(\boldsymbol{x}) + LT(q)/LT(p_i)$
      $q(\boldsymbol{x}) = q(\boldsymbol{x}) - (LT(q)/LT(p_i))p_i(\boldsymbol{x})$
      flag $\leftarrow$ true
    else
      $i = i + 1$
    end if
  end while
  if flag = true **then**
    $r(\boldsymbol{x}) = r(\boldsymbol{x}) + LT(q)$
    $q(\boldsymbol{x}) = q(\boldsymbol{x}) - LT(q)$
  end if
end while
return $a_1(\boldsymbol{x}), \ldots, a_s(\boldsymbol{x}), r(\boldsymbol{x})$

Definition A.8. The **leading term** of $p(\boldsymbol{x})$ (w.r.t. $>$) is the product $c_{\boldsymbol{\alpha}}\boldsymbol{x}^{\boldsymbol{\alpha}}$ where $\boldsymbol{x}^{\boldsymbol{\alpha}}$ is the largest monomial appearing in $p(\boldsymbol{x})$ in the ordering $>$. The notation $LT(p)$ will be used for the leading term. Furthermore, if $LT(p) = c_{\boldsymbol{\alpha}}\boldsymbol{x}^{\boldsymbol{\alpha}}$, then $c_{\boldsymbol{\alpha}}$ is the **leading coefficient** of $p(\boldsymbol{x})$, denoted by $LC(p)$, and $\boldsymbol{x}^{\boldsymbol{\alpha}}$ is the **leading monomial** of $p(\boldsymbol{x})$, denoted by $LM(p)$

A multivariate division algorithm can now be devised, for which an algorithm is presented in Algorithm A.1.

Definition A.9. Fix any monomial ordering $>$, and let $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$ be an ordered set of polynomials. Then any polynomial $q(\boldsymbol{x}) \in \mathcal{P}^n$ can be written as

$$q(\boldsymbol{x}) = a_1(\boldsymbol{x})p_1(\boldsymbol{x}) + \cdots + a_s(\boldsymbol{x})p_s(\boldsymbol{x}) + r(\boldsymbol{x}), \tag{A.35}$$

where $a_i(\boldsymbol{x}), r(\boldsymbol{x}) \in \mathcal{P}^n$, for each i, $a_i(\boldsymbol{x})p_i(\boldsymbol{x}) = 0$ or $LT(q) \geq LT(a_i \cdot p_i)$, and either $r(\boldsymbol{x}) = 0$ or $r(\boldsymbol{x})$ is a linear combination of monomials, none of which is divisible by any of $LT(p_1), \ldots, LT(p_s)$. We call $r(\boldsymbol{x})$ the remainder of $p(\boldsymbol{x})$ on division by $p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})$.

A.4.2 Buchberger's algorithm

Another important ingredient in Buchberger's algorithm are the S-polynomials[5]. S-polynomials are used to eliminate leading terms from a system of polynomial equations, by considering two polynomials from the system and consequently computing a least common multiple, leading to the vanishing of the leading terms. The multivariate division algorithm is used to reduce the result from this operation to a normal form (Appendix A.5.2) with respect to a set of polynomial equations. This normal form is the remainder in the multivariate division algorithm.

Definition A.10. Let $p(\boldsymbol{x}), q(\boldsymbol{x}) \in \mathcal{P}^n$ be nonzero polynomials. Fix a monomial order $>$ and let

$$LT(p) = c\boldsymbol{x}^{\boldsymbol{\alpha}} \text{ and } LT(q) = d\boldsymbol{x}^{\boldsymbol{\beta}}, \tag{A.36}$$

where $c, d \in \mathbb{K}$. Let $\boldsymbol{x}^{\boldsymbol{\gamma}}$ be the least common multiple of $\boldsymbol{x}^{\boldsymbol{\alpha}}$ and $\boldsymbol{x}^{\boldsymbol{\beta}}$. The **S-polynomial** of $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$, denoted by $S(p, q)(\boldsymbol{x})$, is the polynomial

$$S(p, q) = \frac{\boldsymbol{x}^{\boldsymbol{\gamma}}}{LT(p)} p(\boldsymbol{x}) - \frac{\boldsymbol{x}^{\boldsymbol{\gamma}}}{LT(q)} q(\boldsymbol{x}). \tag{A.37}$$

By definition, $S(p, q)(\boldsymbol{x}) \in \langle p(\boldsymbol{x}), q(\boldsymbol{x}) \rangle$.

Example A.11. Consider $p(\boldsymbol{x}) = x_1^3 x_2 - 2x_1^2 x_2^2 + x_1$ and $q(\boldsymbol{x}) = 3x_1^4 - x_2$. We use the LEX ordering. We have $\boldsymbol{x}^{\boldsymbol{\gamma}} = x_1^4 x_2$ and

$$S(p, q)(\boldsymbol{x}) = x_1 p(\boldsymbol{x}) - \frac{1}{3} x_2 q(\boldsymbol{x}) = -2x_1^3 x_2^2 + x_1^2 + \frac{1}{3} x_2^2. \tag{A.38}$$

If we continue with the equations from the previous example, and now take the remainder on division by $I = \{p(\boldsymbol{x}), q(\boldsymbol{x})\}$, denoted $\overline{S(p, q)}^{I}(\boldsymbol{x})$, we can uncover new leading terms of elements in $\mathcal{I} = \langle p(\boldsymbol{x}), q(\boldsymbol{x}) \rangle$. Note that this step requires Algorithm A.1.

Example A.12. In this case, we find that the remainder is

$$\overline{S(p, q)}^{I}(\boldsymbol{x}) = -4x_1^2 x_2^3 + x_1^2 + 2x_1 x_2 + \frac{x_2^2}{3}, \tag{A.39}$$

and $LT\left(\overline{S(p, q)}^{I}\right) = -4x_1^2 x_2^3$ is divisible by neither $LT(p)$ nor $LT(q)$.

Buchberger's algorithm consists in identifying such interfering polynomials in the polynomial system, computing the remainder in the division algorithm, adding the result to the set of polynomial equations, and repeating this procedure until all reductions yield zero. The following theorem expresses a criterion

[5]The name comes from subtraction polynomials or syzygy polynomials.

Algorithm A.2 Buchberger's rudimentary algorithm

Require: $I = \{p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\}$
1: $G = IF$
2: **while** $G = H$ **do**
3: $H = G$
4: **for** each pair $f(\boldsymbol{x}) \neq g(\boldsymbol{x})$ in H **do**
5: $h(\boldsymbol{x}) = \overline{S(f,g)}^{H}(\boldsymbol{x})$
6: If $h(\boldsymbol{x}) \neq 0$, add $G = G \cup \{h(\boldsymbol{x})\}$
7: **end for**
8: **end while**
9: **return** $G = \{g_1(\boldsymbol{x}), \ldots, g_t(\boldsymbol{x})\}$

by Buchberger that is essential for this procedure and leads directly to a rudimentary version of Buchberger's algorithms (Algorithm A.2).

Theorem A.5. A finite set $G = \{g_1(\boldsymbol{x}), \ldots, g_t(\boldsymbol{x})\}$ is a Gröbner basis of $\mathcal{I} = \langle g_1(\boldsymbol{x}), \ldots, g_t(\boldsymbol{x})\rangle$ if and only if $\overline{S(g_i, g_j)}^{G}(\boldsymbol{x}) = 0$ for all pairs $i \neq j$.

Although Buchberger's rudimentary algorithm is constructive and finishes in a finite number of steps, it requires exact arithmetic, and turns out to be impractical even medium-sized problems. During the last decades, a series of optimizations has been introduced, of which the work by Faugère [88, 89] (and extensions) are currently among the most competitive approaches to compute a Gröbner basis of a polynomial system.

A.5 Stetter's eigenvalue-eigenvector approach

In the current section, we will illustrate the Stetter's eigenvalue-eigenvector approach for solving a system of polynomial equations as an eigenvalue problem. The approach works by computing an eigenvalue decomposition of a matrix representing multiplication in the quotient space (Appendix A.5.1) and requires the concept of normal forms (Appendix A.5.2). Stetter's eigenvalue-eigenvector approach is closely related to the approaches developed in this dissertation.

A.5.1 Multiplication maps in the quotient space

Let us consider an ideal $\mathcal{I} = \langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle$ that describes an affine zero-dimensional variety $\mathcal{V}(\mathcal{I})$. In Appendix A.2, it is explained that the quotient ring $\mathcal{R}[\mathcal{I}] = \mathcal{P}^n/\langle p_1(\boldsymbol{x}), \ldots, p_s(\boldsymbol{x})\rangle$ is a commutative ring. Given a polynomial $g(\boldsymbol{x})$, we can use multiplication to define a linear map $\mathcal{A}_g$ from $\mathcal{R}[\mathcal{I}]$ to itself:

$$\mathcal{A}_g : \mathcal{R}[\mathcal{I}] \to \mathcal{R}[\mathcal{I}] : [f]_{\mathcal{I}} \mapsto [f]_{\mathcal{I}}[g]_{\mathcal{I}} = [fg]_{\mathcal{I}}. \tag{A.40}$$

Since $\mathcal{R}[\mathcal{I}]$ is a finite-dimensional vector space over $\mathbb{C}$, we can represent the multiplication map by its matrix with respect to a basis $B = \left\{[b_1]_{\mathcal{I}}, \ldots, [b_{m_b}]_{\mathcal{I}}\right\}$.

This $m_b \times m_b$ matrix is called the multiplication matrix and denoted by $\boldsymbol{A}_g$. Its eigenvalues and eigenvectors have important properties:

Theorem A.6. Let $\mathcal{I} \subset \mathcal{P}^n$ be an ideal with zero-dimensional variety $\mathcal{V}(\mathcal{I}) = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{m_b})$ and $\boldsymbol{A}_g$ the multiplication matrix for a multiplication map $\mathcal{A}_g$ with respect to a given basis $B = \{[b_1]_{\mathcal{I}}, \ldots, [b_{m_b}]_{\mathcal{I}}\}$. The eigenvalues of $\boldsymbol{A}_g$ are the evaluations of $g(\boldsymbol{x})$ in the m_b points of the zero-dimensional variety and the row vector

$$\begin{bmatrix} b_1(\boldsymbol{z}_i) & \cdots & b_{m_b}(\boldsymbol{z}_i) \end{bmatrix} \tag{A.41}$$

lies in the left eigenspace for the eigenvalue $g(\boldsymbol{z}_i)$, $i = 1, \ldots, m_b$.

Proof. A proof of this theorem can be found in [66, Chapter 4]. □

This theorem implies that the solutions of a system of multivariate polynomial equations can be retrieved from the multiplication structure of the quotient space by computing the multiplication matrices $\boldsymbol{A}_{x_1}, \ldots, \boldsymbol{A}_{x_n}$ in the variables $x_1, \ldots, x_n$ and computing their eigenvalues. Note that the matrix $\boldsymbol{A}_{x_i}$ and $\boldsymbol{A}_{x_j}$, for any i and j, commute since $x_i x_j = x_j x_i$. As a result, the matrices $\boldsymbol{A}_{x_i}$ and $\boldsymbol{A}_{x_j}$ have common eigenspaces. Thus, the traditional procedure to find all affine common roots can be summarized as:

1. Compute the multiplication matrices $\boldsymbol{A}_{x_1}, \ldots, \boldsymbol{A}_{x_n}$ for a particular basis B of the quotient space $\mathcal{R}[\mathcal{I}]$.
2. Perform a simultaneous triangularization of $\boldsymbol{A}_{x_1}, \ldots, \boldsymbol{A}_{x_n}$ to retrieve the common roots of the system.

We elaborate more on the first step in the next paragraph, while the second step can be tackled via standard numerical linear algebra algorithms.

A.5.2 Normal forms

An essential element of the first step are normal forms; hence, the name "normal form algorithms" for this type of algorithms. If the basis $B = \{[b_1]_{\mathcal{I}}, \ldots, [b_{m_b}]_{\mathcal{I}}\}$ of the quotient space $\mathcal{R}[\mathcal{I}]$ is fixed, then, for any $f(\boldsymbol{x})$, there are unique constants c_j such that

$$f(\boldsymbol{x}) - \sum_{j=1}^{m_b} c_j b_j(\boldsymbol{x}) \in \mathcal{I}. \tag{A.42}$$

These are the coefficients in the unique expansion of $[f]_{\mathcal{I}} = \sum_{j=1}^{m_b} c_j b_j(\boldsymbol{x})$ in the basis B. This map

$$\mathcal{M} : \mathcal{P}^n \to B : f(\boldsymbol{x}) \mapsto \sum_{j=1}^{m_b} c_j b_j(\boldsymbol{x}), \tag{A.43}$$

which sends $f(\boldsymbol{x})$ to $\sum_{j=1}^{m_b} c_j b_j(\boldsymbol{x})$, is called a normal form. Its key property is that $\mathcal{M}$ projects $\mathcal{P}^n$ onto B along $\mathcal{I}$. The multiplication matrix $\boldsymbol{A}_g : B \to B$ is

Table A.1. Numerical solutions of system (A.44) obtained via Stetter's eigenvalue-eigenvector approach.

$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\boldsymbol{x_3}$
$1.8570 \mp 0.1760\mathrm{i}$	$1.6000 \pm 0.1510\mathrm{i}$	$0.5000 \pm 0.8666\mathrm{i}$
$-2.0000 + 0.0000\mathrm{i}$	$-1.5000 + 0.0000\mathrm{i}$	$-1.0000 + 0.0000\mathrm{i}$
$-2.3570 \pm 0.6890\mathrm{i}$	$1.1720 \mp 0.3430\mathrm{i}$	$0.5000 \mp 0.8666\mathrm{i}$
$3.0000 + 0.0000\mathrm{i}$	$1.0000 + 0.0000\mathrm{i}$	$-1.0000 + 0.0000\mathrm{i}$

simply given by $\mathcal{A}_g(b_j) = \mathcal{M}(g \cdot b_j)$. This means that the ith column of the multiplication matrix contains the coefficients $c_j(g \cdot b_i)$ of $\mathcal{M}(g \cdot b_i)$. Computing normal forms can be done via symbolic computations or linear algebra (using the Macaulay matrix) [243]. The following example illustrate the symbolic normal form approach.

Example A.13 (taken from [78]). Let us consider a small example, namely the system of multivariate polynomial equations

$$\begin{cases} p_1(\boldsymbol{x}) = x_1x_2 - 3 = 0, \\ p_2(\boldsymbol{x}) = x_1^2 - x_3^2 + x_1x_3 - 5 = 0, \\ p_3(\boldsymbol{x}) = x_3^3 - 2x_1x_2 + 7 = 0. \end{cases} \tag{A.44}$$

A Gröbner basis G for this polynomial system can be computed (using the graded inverse lexicographic (GRINVLEX) ordering):

$$\begin{cases} g_1(\boldsymbol{x}) = x_1x_2 - 3 = 0, \\ g_2(\boldsymbol{x}) = 14x_2^2 - 25 - 5x_1x_3 + 2x_3 - 5x_2 + x_1 = 0, \\ g_3(\boldsymbol{x}) = 5x_3x_2 + 15 - 6x_1x_3 - 3x_1^2 - x_2 = 0, \\ g_4(\boldsymbol{x}) = x_3^2 + 5 - x_1x_3 - x_1^2 = 0, \\ g_5(\boldsymbol{x}) = 5x_1^3 - 25 + 4x_1x_3 + 2x_1^2 - 25x_3 + 84x_2 - 75x_1 = 0, \\ g_6(\boldsymbol{x}) = 5x_3x_1^2 + 15 - 2x_1x_3 - x_1^2 - 42x_2 + 25_1 = 0. \end{cases} \tag{A.45}$$

The leading terms of G are

$$\left\{x_1x_2, x_2^2, x_2x_3, x_3^2, x_1^3, x_1^3x_3\right\}, \tag{A.46}$$

so the normal set B is

$$B = \left\{1, x_1, x_2, x_3, x_1^2, x_1x_3\right\}. \tag{A.47}$$

If we choose $g(\boldsymbol{x}) = x_1 + 2x_2 + 3x_3$ as the shift polynomial, then the multiplication matrix $\boldsymbol{A}_g$ is equal to

$$\boldsymbol{A}_g = \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 \\ 6 & 0 & 0 & 0 & 1 & 3 \\ -17/7 & -1/7 & 46/35 & -2/7 & 9/5 & 151/35 \\ -21 & 0 & 2/5 & 0 & 21/5 & 32/5 \\ -4 & 6 & 42/5 & 5 & 1/5 & 2/5 \\ 3 & 10 & -84/5 & 21 & -2/5 & -4/5 \end{bmatrix}, \tag{A.48}$$

which is obtained by computing for each element B its multiplication with g and reducing the result with respect to G; the remainder is linear in the elements of B and composes a row of $\boldsymbol{A}_g$. We have then

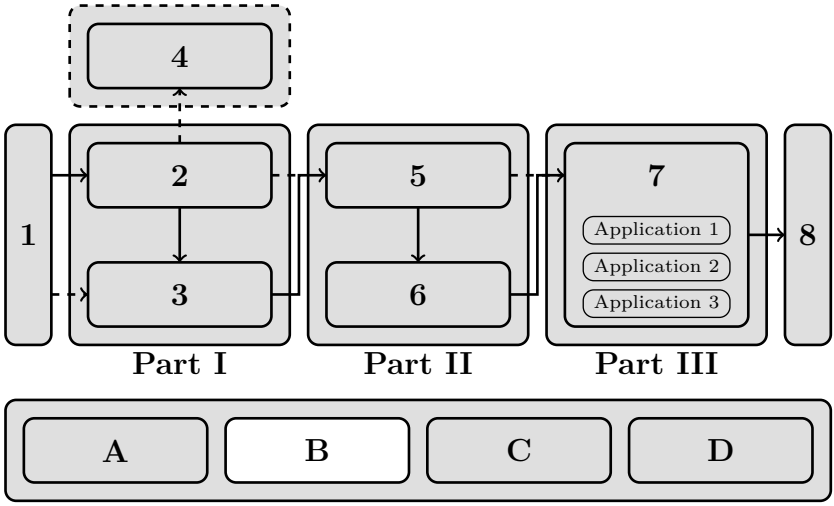
$$\boldsymbol{A}_g \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_1 x_3 \end{bmatrix} = g(\boldsymbol{x}) \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \\ x_1^2 \\ x_1 x_3 \end{bmatrix}. \tag{A.49}$$

The eigenvectors of $\boldsymbol{A}_g$ are re-scaled such that the first entry equals one, from which we can then read off the solutions (Table A.1).

Numerical Linear Algebra

In this appendix, we review some concepts and methods of (numerical) linear algebra. The focus is mainly on the computation of the different matrix decompositions that we use in the dissertation. The numerical algorithms to compute these matrix decompositions are at the heart of the solutions approaches that we have developed. Note that the decades of advancements in numerical linear algebra algorithms have given us state-of-the-art implementations that are able to solve problems in an computational efficient and a backward stable way.

Some good reference works on (numerical) linear algebra are [29, 74, 97, 207, 235, 249]. A historical overview of the field can be found in [46]. For more information about eigenvalue problems, [123, 270] are useful.



Outline. First, we review some essential properties about vectors and matrices in Appendix B.1. The conditioning of a problem and the stability of an algorithm are discussed in Appendix B.2. Next, in Appendix B.3, we consider the rank, determinant, and inverse of a matrix. Appendix B.4 contains more information about eigenvalues and eigenvectors. Finally, in Appendix B.5, we summarize the necessary background information about the four different matrix decompositions that appear in this text.

B.1 Vectors and matrices

In linear algebra, vectors and matrices in finite-dimensional vector spaces are omnipresent. A vector is one-dimensional array of (scalar) numbers, while a matrix is a two-dimensional array of scalar numbers. These numbers belong to a certain field K. We denote scalars by italic lowercase letters, e.g., $a \in K$, and vectors by boldface lowercase letters, e.g., $\boldsymbol{a} \in K^{k\times 1}$. By default, vectors are assumed to be column vectors. Matrices are denoted by bold-face upper case letters, e.g., $\boldsymbol{A} \in K^{k\times l}$ is a matrix with k rows and l columns consisting of numbers from K. In this text, we mainly considered complex numbers; hence, we restrict ourselves in the remainder of this section to the field of complex numbers ($K = \mathbb{C}$). The operations $\boldsymbol{A}^{-1}$, $\boldsymbol{A}^{\mathrm{T}}$, and $\boldsymbol{A}^{\mathrm{H}}$ denote the inverse, transpose, and Hermitian transpose of a matrix $\boldsymbol{A}$, respectively.

A (complex) vector $\boldsymbol{a} \in \mathbb{C}^{k\times 1}$ in this dissertation is thus a one-dimensional array, whose entries are denoted by

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ \vdots \\ a_k \end{bmatrix} = (\boldsymbol{a}_i)_{1\leq i\leq k}. \tag{B.1}$$

Similarly, a (complex) matrix $\boldsymbol{A} \in \mathbb{C}^{k\times l}$ is a two-dimensional array, whose entries are denoted by

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & \cdots & a_{1l} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kl} \end{bmatrix} = (\boldsymbol{A}_{ij})_{1\leq i\leq k, 1\leq j\leq l}. \tag{B.2}$$

The transpose $\boldsymbol{A}^{\mathrm{T}}$ of a matrix $\boldsymbol{A}$ is an operator that flips a matrix over its diagonal. Formally, the ijth element of $\boldsymbol{A}$ becomes the jith element of $\boldsymbol{A}^{\mathrm{T}}$:

$$\left[\boldsymbol{A}^{\mathrm{T}}\right]_{ij} = \left[\boldsymbol{A}\right]_{ji}. \tag{B.3}$$

The Hermitian transpose $\boldsymbol{A}^{\mathrm{H}}$, also known as conjugate transpose, of a matrix $\boldsymbol{A}$ is obtained by taking the transpose of that matrix and applying complex conjugation on each entry:

$$\left[\boldsymbol{A}^{\mathrm{H}}\right]_{ij} = \left[\bar{\boldsymbol{A}}\right]_{ji}. \tag{B.4}$$

A vector norm is a function of a vector that assigns a (positive) length to that vector. Two vector norms are used in this dissertation:

- The Euclidean norm or 2-norm of a vector is defined as

$$\|\boldsymbol{a}\|_2 = \sqrt{\overline{a_1}a_1 + \cdots \overline{a_n}a_n}, \tag{B.5}$$

  where $\overline{\cdot}$ denotes the complex conjugation. Notice that $\|\boldsymbol{a}\|_2 = \boldsymbol{a}^{\mathrm{H}}\boldsymbol{a}$.

- The Manhattan norm or 1-norm of a vector is defined as

$$\|\boldsymbol{a}\|_1 = \sum_{i=1}^{n} |a_i|. \tag{B.6}$$

The notion of a norm can be naturally extended to matrices. For an n-dimensional vector space V and an m-dimensional vectors space W with norms $\|\cdot\|_V$ and $\|\cdot\|_W$ on V and W, respectively, the induced matrix norm is

$$\|\boldsymbol{A}\|_{V,W} = \sup_{\boldsymbol{x}\in V\setminus\{\boldsymbol{0}\}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_W}{\|\boldsymbol{x}\|_V}. \tag{B.7}$$

We encounter the following specific matrix norms in this text:

- The 2-norm of a matrix is defined as

$$\|\boldsymbol{A}\|_2 = \max\{\|\boldsymbol{A}\boldsymbol{z}\|_2 : \|\boldsymbol{z}\|_2 = 1\} = \sigma_1(\boldsymbol{A}), \tag{B.8}$$

  where $\sigma_1(\boldsymbol{A})$ denotes the largest singular value of $\boldsymbol{A}$.

- The Frobenius norm of a matrix is defined as

$$\|\boldsymbol{a}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|a_{ij}|^2} = \sqrt{\sum_{i=1}^{\min(m,n)}\sigma_i^2(\boldsymbol{A})}, \tag{B.9}$$

  where $\sigma_i(\boldsymbol{A})$ denotes the ith largest singular value of $\boldsymbol{A}$.

B.2 Conditioning and stability

Two important concepts in numerical linear algebra, when solving problems with numerical algorithms, are the condition and stability. Consider the following example.

Example B.1. If we have a nonsingular, square matrix $\boldsymbol{A}$ and a column vector $\boldsymbol{b}$, then solving the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ yields exactly one solution, which is given by $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. In exact arithmetic, this is the end of the story. However, on computers, if we want to compute the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ in floating-point arithmetic, the intermediate results are replaced by nearby machine numbers, causing rounding errors in the computed solution $\widetilde{\boldsymbol{x}}$. Moreover, the values of $\boldsymbol{A}$ and $\boldsymbol{b}$ can also not be represented exactly on the computer, so their entries are already replace by machine numbers before the computations start. So, we hope that our numerical algorithm computes a good approximation $\widetilde{x}$ of the true solution, in a sense that $\boldsymbol{A}\widetilde{\boldsymbol{x}} \approx \boldsymbol{b}$, or even better $\widetilde{\boldsymbol{x}} \approx \boldsymbol{x}$.

We want to known whether our numerical algorithm did a good job, we want to measure how good that job was, and want a way of deciding whether the obtained errors are satisfactory. A good criterion for deciding this takes into account that the computer treats our problem as a slightly perturbed version of the problem, and the solution may be very sensitive to such perturbations. These questions are captured in the fundamental concepts condition and stability from numerical analysis. While the condition has to do with the problem, the stability is a property of the numerical algorithm used to solve that problem, or in the words of [249],

> *Conditioning pertains to the perturbation behavior of a mathematical problem. Stability pertains to the perturbation behavior of an algorithm used to solve that problem on a computer.*

These two concepts are often confused. Hence, we summarize their meanings in the following two sections (Appendices B.2.1 and B.2.2). Afterwards, we explain how we quantify the obtained errors in this text (Appendix B.2.3).

B.2.1 Conditioning and condition numbers

In general, we can view a problem as a function $f : X \to Y$ from a normed vector space X of data to a normed vector space Y of solutions. This function is usually nonlinear, even in linear algebra, but most of the time at least continuous. Typically, we shall be concerned with the behavior of a problem f at a particular point $x \in X$, because the behavior can greatly vary from one point to another.

A well-conditioned problem is one with the property that all small perturbations of x lead to only small changes in $f(x)$. An ill-conditioned problem is one with the property that some small perturbation of x leads to a large change in $f(x)$. Of course, the meaning of "small" and "large" in these statements depends on the application.

One way of quantifying the perturbation behavior is via condition numbers. Let δx denote a small perturbation of x and write $\delta f = f(x + \delta x) - f(x)$. The absolute condition number $\kappa = \kappa(x)$ of the problem f at x is defined as

$$\kappa = \lim_{\delta \to 0} \sup_{\|\delta x\| \leq \delta} \frac{\|\delta f\|}{\|\delta x\|}. \tag{B.10}$$

If f is differentiable, we can evaluate the condition number by means of the derivative of f. Let $J(x)$ be the matrix whose i, j-entry is the partial derivative $\partial f_i / \partial x_j$ evaluated at x, known as the Jacobian of f at x. The definition of the derivative gives us, to the first order, in the limit

$$\kappa = \|J(x)\|, \tag{B.11}$$

where $\|J(x)\|$ represents the norm of $J(x)$ induced by the norms on X and Y.

When we are concerned with relative changes, we need the notion of relative condition. The relative condition number $\kappa^{(r)} = \kappa^{(r)}(x)$ is defined by

$$\kappa^{(r)} = \lim_{\delta \to 0} \sup_{\|\delta x\| \leq \delta} \left(\frac{\|\delta f\|}{\|f(x)\|} \Big/ \frac{\|\delta x\|}{\|x\|} \right). \tag{B.12}$$

If f is again differentiable, then we can express this quantity in terms of the Jacobian:

$$\kappa^{(r)} = \frac{\|J(x)\|}{\|f(x)\| / \|x\|}. \tag{B.13}$$

Both absolute and relative condition numbers have their uses, but the latter are more important in numerical analysis because the floating-point arithmetic used by computers introduces relative errors rather than absolute ones. The

condition number may depend strongly on x, meaning that some instances of the problem are more sensitive to perturbations than others. A problem is now well-conditioned if $\kappa^{(r)}$ is small (e.g., $\kappa^{(r)} = 10^0$ or $\kappa^{(r)} = 10^2$) and ill-conditioned when $\kappa^{(r)}$ is large (e.g., $\kappa^{(r)} = 10^6$ or $\kappa^{(r)} = 10^{10}$).

Example B.2. To illustrate, we consider an example from polynomial root-finding. The determination of the roots of a univariate polynomial is a classic example of an ill-conditioned problem. Consider $x^2-2x+1 = (x-1)^2$, which has clearly a double root at $x = 1$. A small perturbation in the coefficients may lead to a larger change in the roots. For example, $x^2 - 2x + 0.9999 = (x-0.99)(x-1.01)$. In fact the roots can change in proportion to the square root of the change in the coefficients, so in this case the Jacobian is infinite (the problem is not differentiable), and $\kappa^{(r)} = \infty$.

Polynomial root-finding is typically ill-conditioned even in cases that do not involve multiple roots. If the ith coefficient c_i of a polynomial $p(x)$ is perturbed by an infinitesimal quantity δc_i, the perturbation of the jth root x_j is $\delta x_j = -(\delta c_i)x_j^i/p'(x_j)$, where $p'(x)$ is the derivative of $p(x)$. The condition number of x_j with respect to perturbations of the single coefficient c_i is

$$\kappa^{(r)} = \frac{|\delta x_j|}{|x_j|} \Big/ \frac{|\delta c_i|}{|c_i|} = \frac{\left|c_i x_j^{i-1}\right|}{|p'(x_j)|}. \tag{B.14}$$

This number is often very large, for example, in the famous Wilkinson polynomial

$$p(x) = \prod_{i=1}^{20}(x-i) = c_0 + c_1 x + \cdots + c_{19}x^{19} + x^{20}, \tag{B.15}$$

where the condition number of the most sensitive coefficient, c_{15}, is $\approx 5.1 \times 10^{13}$.

Example B.3. Fix an invertible matrix $\boldsymbol{A} \in \mathbb{C}^{k\times k}$ and let the operation be the matrix-vector product

$$f : \mathbb{C}^k \to \mathbb{C}^k : \boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}. \tag{B.16}$$

For any norm on $\mathbb{C}^k$, we find that

$$\kappa^{(r)} \leq \|\boldsymbol{A}\|\left\|\boldsymbol{A}^{-1}\right\|. \tag{B.17}$$

This relative condition number is an very important constant for an invertible matrix, called the condition number of $\boldsymbol{A}$. It is also used to quantify, for example, the relative condition number of solving a linear system.

B.2.2 Numerical stability

Previously, we defined a problem as a function $f : X \to Y$ from a vector space X of data to a vector space Y of solutions. An algorithm can be viewed as another

map $\widetilde{f} : X \to Y$ between the same two spaces. Given data $x \in X$ rounded to floating-point machine data, let us supply this data to the algorithm. The result is again a collection of floating-point numbers that belong to the vector space Y, namely $\widetilde{f}(x)$. The result $\widetilde{f}(x)$ will be affected by rounding errors and, depending on the circumstances, other complications like convergence tolerances or other jobs of the computer. The function $\widetilde{f}(x)$ may even take different values from run to run; it may be multivalued. Yet, despite these complications, we can make some statements about $\widetilde{f}(x)$.

First, however, we need a way of measuring the error of $\widetilde{f}(x)$ as an approximation for $f(x)$. We can consider the absolute forward error of a computation,

$$\left\| \widetilde{f}(x) - f(x) \right\|, \tag{B.18}$$

or the relative forward error of a computation,

$$\frac{\left\| \widetilde{f}(x) - f(x) \right\|}{\|f(x)\|}. \tag{B.19}$$

If $\widetilde{f}$ is a good algorithm, one might expect the relative forward error to be small, of order ϵ_{mach} (machine precision). One may say that an algorithm $\widetilde{f}$ for a problem f is accurate if for each $x \in X$

$$\frac{\left\| \widetilde{f}(x) - f(x) \right\|}{\|f(x)\|} = \mathcal{O}(\epsilon_{mach}). \tag{B.20}$$

The relative forward error is small if the approximate solution is

If the problem f is ill-conditioned, however, the goal of accuracy is unreasonably ambitious. Rounding the input data is unavoidable on a computer, and even if all the subsequent computations could be carried out perfectly, this perturbation alone might lead to a significant change in the result. Instead of aiming for accuracy in all cases, the most that is appropriate is to aim for forward stability. We say that an algorithm $\widetilde{f}$ for a problem f is forward stable if for each $x \in X$

$$\frac{\left\| \widetilde{f}(x) - f(\widetilde{x}) \right\|}{\|f(\widetilde{x})\|} = \mathcal{O}(\epsilon_{mach}), \tag{B.21}$$

for some $\widetilde{x}$ with

$$\frac{\|\widetilde{x} - x\|}{\|x\|} = \mathcal{O}(\epsilon_{mach}). \tag{B.22}$$

In words, this means that a forward stable algorithm gives nearly the right answer to nearly the right question.

Many algorithms of numerical linear algebra satisfy a condition that is both stronger and simpler than forward stability. We say that an algorithm $\widetilde{f}$ for a problem f is backward stable if for each $x \in X$

$$\widetilde{f}(x) = f(\widetilde{x}) \tag{B.23}$$

for some $\widetilde{x}$ with

$$\frac{\|\widetilde{x} - x\|}{\|x\|} = \mathcal{O}(\epsilon_{mach}). \tag{B.24}$$

This tightening of the definition of stability means that a backward stable algorithm gives exactly the right answer to nearly the right question. Unlike conditioning, stability is a property of the algorithm, not of the problem. Usually, the conditioning of a problem is out of our hands, but (backward) stability is what we aim for when designing algorithms.

B.2.3 Error measures

In this dissertation, two types of error measures are used to describe the numerical correctness of the results: the reconstruction error and residual error.

Reconstruction error. Given a matrix $\boldsymbol{A}$ and a computed basis matrix of its right null space $\boldsymbol{Z}$, the absolute reconstruction error is defined as

$$\widetilde{e} = \|\boldsymbol{AZ}\|_2. \tag{B.25}$$

In the context of floating-point arithmetic, the relative reconstruction error is often more meaningful:

$$\widetilde{e}^{(r)} = \frac{\|\boldsymbol{AZ}\|_2}{\|\boldsymbol{A}\|_2}. \tag{B.26}$$

For both the absolute and relative reconstruction error, an index i is sometimes added as a subscript, i.e., $\widetilde{e}_i$ and $\widetilde{e}_i^{(r)}$, to denote, for example, the corresponding iteration.

Residual error. To describe the numerical correctness of the computed solutions of a system of multivariate polynomial equations or rectangular multiparameter eigenvalue problem (MEP), we use the residual error. The absolute residual error of a solution $\boldsymbol{x}^*$ for a polynomial system with polynomials $p_i(\boldsymbol{x})$, $i = 1, \dots s$ is given by

$$\|\boldsymbol{e}\|_2 = \sum_{i=1}^{s} \|p_i(\boldsymbol{x}^*)\|_2, \tag{B.27}$$

while the same error for an eigenvalue $\boldsymbol{\lambda}^*$ and associated eigenvector $\boldsymbol{z}^*$ of a rectangular MEP with matrix pencil $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})$ is given by

$$\|\boldsymbol{e}\|_2 = \|\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^*)\boldsymbol{z}^*\|_2. \tag{B.28}$$

It is also possible to define a relative residual error for both problems, namely

$$\|\boldsymbol{e}\|_2^{(r)} = \frac{\sum_{i=1}^{s} \|p_i(\boldsymbol{x}^*)\|_2}{\sum_{i=1}^{s} \sum_{\mathcal{A}} \left\| c_{i,\boldsymbol{\alpha}} (\boldsymbol{x}^*)^{\boldsymbol{\alpha}} \right\|_2} \tag{B.29}$$

and

$$\|\boldsymbol{e}\|_2^{(r)} = \frac{\|\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda}^*)\boldsymbol{z}^*\|_2}{\sum_{i=1}^{k} \sum_{j=1}^{l} \sum_{\mathcal{W}} \left\| [\boldsymbol{A}_{\boldsymbol{\omega}}]_{ij} (\boldsymbol{\lambda}^*)^{\boldsymbol{\omega}} \right\|_2}. \tag{B.30}$$

In experiments, often the maximum (absolute/relative) residual error for all obtained solutions is reported.

B.3 Rank, determinant, and inverse of a matrix

Essential when working with matrices are the concepts of the rank, determinant, and inverse of a matrix.

Definition B.1. A set of vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ is said to be **linearly independent** if

$$\sum_{i=1}^{n} c_i \boldsymbol{a}_i = \boldsymbol{0} \tag{B.31}$$

implies that $c_1 = \cdots = c_n = 0$. If, on the other hand, a non-trivial linear combination of the vectors $\boldsymbol{a}_i$ exists that is equal to zero, then we say that the set of vectors is **linearly dependent**.

Definition B.2. A **subspace** $\mathcal{S}$ of $\mathbb{C}^k$ is a subset that is also a vector space. The set of all linear combinations of a given collection of vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \in \mathbb{C}^{k\times 1}$ is a subspace, also referred to as the **span** of $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$, i.e.,

$$\langle \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \rangle = \left\{ \sum_{i=1}^{n} c_i \boldsymbol{a}_i : c_i \in \mathbb{C} \right\}. \tag{B.32}$$

Definition B.3. A **basis** $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m\}$ for a subspace $\mathcal{S}$ has two properties:

- It is linearly independent.
- It spans the subspace, i.e., $\forall \boldsymbol{s} \in \mathcal{S}$ we have that $\boldsymbol{s} = \sum_{i=1}^{k} c_i \boldsymbol{b}_i$.

All bases for a subspace $\mathcal{S}$ have the same number of elements, which is called the **dimension** of the subspace and is denoted $\dim(\mathcal{S})$.

Suppose that we consider a matrix $\boldsymbol{A} \in \mathbb{C}^{p\times q}$, which is a linear transformation from $\mathbb{C}^p$ to $\mathbb{C}^q$. Then we can associate four fundamental subspaces with that matrix, two in $\mathbb{C}^p$ and two in $\mathbb{C}^q$: the row space $\mathcal{R}(\boldsymbol{A})$, right null space $\mathcal{N}(\boldsymbol{A})$, column space $\mathcal{R}(\boldsymbol{A})$, and left null space $\mathcal{L}(\boldsymbol{A})$. These four fundamental subspaces, applied to the Macaulay matrix, are the topic of Chapter 4. The rank of a matrix $\boldsymbol{A}$ is defined as the dimension of its column space, or formally

Definition B.4. Let $\boldsymbol{A} \in \mathbb{C}^{k\times l}$ be a matrix. The **rank** of $\boldsymbol{A}$ is defined as

$$\operatorname{rank}(\boldsymbol{A}) = \dim(\mathcal{C}(\boldsymbol{A})). \tag{B.33}$$

A well-known property of matrices is that the rank of $\boldsymbol{A}$ is equal to the rank of $\boldsymbol{A}^{\mathrm{T}}$, i.e., the row rank and the column rank of a matrix are equal. The dimension of the null space (which is the rank of a basis matrix of the null space) is called the nullity, i.e.,

$$\operatorname{nullity}(\boldsymbol{A}) = \dim(\mathcal{N}(\boldsymbol{A})). \tag{B.34}$$

The relation between the rank and nullity is captured in the rank-nullity theorem (Theorem 4.2):

$$\operatorname{rank}(\boldsymbol{A}) + \operatorname{nullity}(\boldsymbol{A}) = l, \tag{B.35}$$

for any $k \times l$ matrix $\boldsymbol{A}$.

A determinant of a square matrix is a number that is a function of the entries of the matrix. Determinants play a central role in understanding the concepts of linear algebra, e.g., when studying the inverse of a matrix or its eigenvalues. Although they are often cumbersome in a numerical setting, we can not avoid to define them for the sake of completeness, while we try to avoid them in numerical computations[1]. The determinant can be defined in several equivalent ways. Laplace's formula allows to compute the determinant using the so-called minors.

Definition B.5. Let $\boldsymbol{A}$ be an $m \times m$ square matrix, of which the elements are denoted by a_{ij}. The minor M_{ij} is defined as the determinant of the $(m-1) \times (m-1)$ matrix that results from $\boldsymbol{A}$ by removing the $i$the row and jth column. The expression $C_{ij} = (-1)^{i+j} M_{ij}$ is called the cofactor. The **determinant** of $\boldsymbol{A}$ is then given by the formula

$$\det(\boldsymbol{A}) = \sum_{i=1}^{m} (-1)^{i+j} M_{ij}, \tag{B.36}$$

with the determinant of a scalar equal to that scalar, i.e., $\det(a) = a$.

Another important operation for a square matrix $\boldsymbol{A}$ is its inverse $\boldsymbol{A}^{-1}$. Note that the inverse of a matrix exists if and only if its determinant is nonzero.

Definition B.6. Let $\boldsymbol{A}$ be a square matrix. If we can find a matrix $\boldsymbol{X}$ for which $\boldsymbol{AX} = \boldsymbol{XA} = \boldsymbol{I}_n$, where $\boldsymbol{I}_n$ is the $n \times n$ identity matrix, we call $\boldsymbol{X}$ the **inverse** of $\boldsymbol{A}$, denoted as $\boldsymbol{A}^{-1}$.

If $\boldsymbol{A}^{-1}$ exists, $\boldsymbol{A}$ is said to be nonsingular (sometimes called regular), otherwise we say that $\boldsymbol{A}$ is singular.

When looking at invertible matrices, unitary matrices are very interesting. An invertible complex square matrix is called unitary when its Hermitian transpose $\boldsymbol{Q}^{\mathrm{H}}$ is also its inverse, i.e.,

$$\boldsymbol{Q}^{\mathrm{H}}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{H}} = \boldsymbol{Q}\boldsymbol{Q}^{-1} = \boldsymbol{I}_k, \tag{B.37}$$

where $\boldsymbol{I}_k$ is the identity matrix. Unitary matrices have some interesting properties: the columns/rows form an orthonormal basis of $\mathbb{C}^k$ with respect to the usual inner product, it is a normal matrix (i.e., the matrix commutes with

[1] Although determinants (via characteristic polynomials) played an important role in computing the eigenvalues of a matrix back in the history (and even in elementary linear algebra courses students use them to solve the standard eigenvalue problems), they are a poor method in general, and are almost never used in scientific computing. Not only determinants are inefficient ("How do we compute a determinant efficiently?"), they can be notoriously inaccurate in floating-point arithmetics.

its Hermitian transpose), and multiplication by $\boldsymbol{Q}$ preserves the inner product, that is $\langle \boldsymbol{Qx}, \boldsymbol{Qy} \rangle = \langle \boldsymbol{x}, \boldsymbol{y} \rangle$. For real numbers, the analogue of the unitary matrix is an orthogonal matrix:

$$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^{\mathrm{T}} = \boldsymbol{Q}\boldsymbol{Q}^{-1} = \boldsymbol{I}_k. \tag{B.38}$$

B.4 Eigenvalues and eigenvectors

When we consider a square $l \times l$ matrix with complex entries, $\boldsymbol{A} \in \mathbb{C}^{l \times l}$, then the vector $\boldsymbol{z} \in \mathbb{C}^{l \times 1}$ is called an eigenvector of $\boldsymbol{A}$ if $\boldsymbol{z}$ is a non-zero vector and $\boldsymbol{Az}$ is a multiple of $\boldsymbol{z}$. That is, there exists a scalar $\lambda \in \mathbb{C}$, so that

$$\boldsymbol{Az} = \lambda \boldsymbol{z}. \tag{B.39}$$

The problem of finding all pairs[2] $(\lambda, \boldsymbol{z})$ that satisfy this condition is the standard eigenvalue problem.

Definition B.7. Given a coefficient matrix $\boldsymbol{A} \in \mathbb{C}^{l \times l}$, the **standard eigenvalue problem (SEP)** consists in finding all scalars $\lambda \in \mathbb{C}$ and corresponding vectors $\boldsymbol{z} \in \mathbb{C}^{l \times 1} \setminus \{\boldsymbol{0}\}$, so that

$$\boldsymbol{\mathcal{M}}(\lambda)\boldsymbol{z} = (\boldsymbol{A} - \boldsymbol{I}\lambda) = \boldsymbol{0}. \tag{B.40}$$

The scalars λ and (non-zero) vectors $\boldsymbol{z}$ are the eigenvalues and eigenvectors of the SEP, respectively.

The eigenvalue associated with a given eigenvector is unique, while each eigenvalue has many eigenvectors associated with it (if λ is an eigenvalue associated with an eigenvector $\boldsymbol{z}$, then any non-zero multiple of $\boldsymbol{z}$ is also an eigenvector with associated eigenvalue λ). Indeed, the set of all vectors (including the zero vector) satisfying (B.40) for a fixed eigenvalue λ is a subspace of $\mathbb{C}^l$, which we call the eigenspace of the matrix $\boldsymbol{A}$ associated with λ. The set of all eigenvalues of a matrix $\boldsymbol{A}$ is called the spectrum of that matrix and is denoted by $\Lambda(\boldsymbol{A})$.

Clearly, (B.39) and (B.40) are equivalent. Thus, λ is an eigenvalue of $\boldsymbol{A}$ if and only if the matrix equation $(\boldsymbol{A} - \boldsymbol{I}_l\lambda)\boldsymbol{z} = \boldsymbol{0}$ has a non-zero solution:

Proposition B.1. For a coefficient matrix $\boldsymbol{A} \in \mathbb{C}^{l \times l}$, the following statements are equivalent:

- λ is an eigenvalue of $\boldsymbol{A}$.
- $\boldsymbol{A} - \boldsymbol{I}_l\lambda$ is a singular matrix.
- $\mathcal{N}(\boldsymbol{A} - \boldsymbol{I}_l\lambda) \neq \{\boldsymbol{0}\}$.

Note that $\mathcal{N}(\boldsymbol{A} - \boldsymbol{I}_l\lambda)$ is exactly the eigenspace associated with λ.

[2] The pair $(\lambda, \boldsymbol{z})$ is often called an *eigenpair* of the matrix $\boldsymbol{A}$.

Proof. The proof is left as an exercise to the reader. □

Corollary. The matrix $\boldsymbol{A}$ is non-singular if and only if 0 is not an eigenvalue of $\boldsymbol{A}$.

An important question immediately arises: "Does every square matrix have eigenvalues?" The answer on this crucial question is given by the following theorem:

Theorem B.1. Every square matrix $\boldsymbol{A} \in \mathbb{C}^{l \times l}$ has l (affine) eigenvalues $\lambda \in \mathbb{C}$.

Proof. This theorem follows directly from the fundamental theorem of algebra (Theorem A.1). □

This theorem is a consequence of Proposition B.1: a scalar λ is an eigenvalue of a matrix $\boldsymbol{A}$ if and only if $\boldsymbol{A} - \boldsymbol{I}_l\lambda$ is singular. A classical way to asses if $\boldsymbol{A} - \boldsymbol{I}_l\lambda$ is singular is to check whether its determinant is equal to zero. Hence, λ is an eigenvalue of the matrix $\boldsymbol{A}$ if and only if the characteristic polynomial[3] $\chi(\lambda)$ is equal to zero:

$$\chi(\lambda) = \det(\boldsymbol{A} - \boldsymbol{I}_l\lambda) = 0. \tag{B.41}$$

One can easily check that the characteristic polynomial is a univariate polynomial in λ of degree l. By the fundamental theorem of algebra (Theorem A.1), every complex polynomial of degree l has exactly l roots, and subsequently the characteristic polynomial equation of a matrix $\boldsymbol{A}$ yields always l eigenvalues. These eigenvalues need not to be distinct, they can appear with a certain multiplicity. The multiplicity of an eigenvalue λ as a root of the characteristic polynomial is called the algebraic multiplicity of an eigenvalue.

Proposition B.2. Let $\boldsymbol{A} \in \mathbb{C}^{l \times l}$ have m distinct eigenvalues $\lambda|_{(1)}, \ldots, \lambda|_{(m)}$, and let $\boldsymbol{z}|_{(1)}, \ldots, \boldsymbol{z}|_{(m)}$ be eigenvectors associated with $\lambda|_{(1)}, \ldots, \lambda|_{(m)}$, respectively. Then the eigenvectors $\boldsymbol{z}|_{(1)}, \ldots, \boldsymbol{z}|_{(m)}$ are linearly independent.

Proof. The proof is left as an exercise to the reader. □

[3] In the literature, one mostly finds a characteristic polynomial defined $\chi(\lambda) = \det(\boldsymbol{I}_l\lambda - \boldsymbol{A}) = 0$, which of course results in the same eigenvalues but is monic. We do not consider the monic definition in order to keep our notation consistent.

Corollary. If $\boldsymbol{A} \in \mathbb{C}^{l\times l}$ has l distinct eigenvalues, then $\boldsymbol{A}$ has a set of l linearly independent eigenvectors. In other words, there is a basis of $\mathbb{C}^l$ consisting of the eigenvectors of $\boldsymbol{A}$.

A matrix $\boldsymbol{A} \in \mathbb{C}^{l\times l}$ that has l linearly independent eigenvectors is called diagonalizable (other word: *semisimple*). The previous corollary states that every matrix that has only distinct eigenvalues is diagonalizable. The converse is not true: a matrix can have repeated eigenvalues and still be diagonalizable. An eigenvalue can have multiple linearly independent associated eigenvectors. The number of linearly independent eigenvectors associated with an eigenvalue is called the geometric multiplicity of an eigenvalue. Stated in other word: the geometric multiplicity of an eigenvalue λ is the dimension of $\mathcal{N}(\boldsymbol{A} - \boldsymbol{I}_l\lambda)$. A matrix that is not diagonalizable is called defective, not to be confused with a derogatory matrix.

Definition B.8 ([123, p. 77]). Let $\boldsymbol{A} \in \mathbb{C}^{l\times l}$. We say that $\boldsymbol{A}$ is **defective** if the geometric multiplicity of some eigenvalue of $\boldsymbol{A}$ is strictly less than its algebraic multiplicity. If the geometric multiplicity of each eigenvalue of $\boldsymbol{A}$ is the same as its algebraic multiplicity, we say that $\boldsymbol{A}$ is non-defective. If each eigenvalue of $\boldsymbol{A}$ has geometric multiplicity equal to 1, we say that $\boldsymbol{A}$ is non-derogatory, otherwise it is **derogatory**.

Finally, we summarize the difference between algebraic and geometric multiplicity and between defective, derogatory, and diagonalizable matrices:

- The algebraic multiplicity of an eigenvalue is the number of times the eigenvalue appears as a root of the characteristic polynomial, while the geometric multiplicity is the dimension of the eigenspace spanned by that eigenvalue. In general, the algebraic multiplicity and geometric multiplicity of an eigenvalue can differ, but the geometric multiplicity can never exceed the algebraic multiplicity [123].

- Both defective and derogatory imply the algebraic multiplicity of at least one eigenvalue, but they are in fact independent concepts [100]. A matrix is diagonalizable if and only if it is non-defective; it has l distinct eigenvalues if and only if it is non-derogatory and non-defective [123].

Example B.4. A good example of a non-defective, derogatory matrix is the identity matrix

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{B.42}$$

It has only one eigenvalue, namely 1. The algebraic multiplicity of this eigenvalue is obviously 2, since the characteristic equation is $\chi(\lambda) = (1-\lambda)^2$. Since every non-zero eigenvector is an eigenvector of $\boldsymbol{I}$, any basis of $\mathbb{C}^2$ is a set of 2 linearly independent eigenvectors of $\boldsymbol{I}$. The geometric multiplicity is thus also equal to 2.

Example B.5. On the other hand, the matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{B.43}$$

has an eigenvalue 0 with algebraic multiplicity equal to 2 (the characteristic equation is $\chi(\lambda) = (-\lambda)^2$), while the geometric multiplicity of that eigenvalue is equal to 1 (the eigenspace associated with the eigenvalue 0 is one-dimensional). The matrix $\boldsymbol{A}$ is thus defective, and hence not diagonalizable.

An important observation related to algorithms for computing the eigenvalues of a general matrix is that any such algorithm must be of an iterative nature. By this, we mean that the algorithm may iteratively compute better and better approximations, but can never, even in exact arithmetic, compute the eigenvalues in finite time. This is prohibited by the Abel–Ruffini theorem, which states that there is no general expression in radicals for the roots of a univariate polynomial of degree 5 or higher. By the fact that the univariate root-finding problem can be translated directly in an eigenvalue problem, the existence of a direct algorithm for computing eigenvalues would contradict this theorem. The key idea of many of the successful eigenvalue solvers is to apply a sequence of similarity transformations to the matrix, such that it converges to a structured matrix from which we can read off the eigenvalues (for example, a diagonal matrix).

Many eigenvalue problems that arise in applications can be formulated as a generalized eigenvalue problem (GEP),

$$\boldsymbol{A}\boldsymbol{z} = \lambda \boldsymbol{B}\boldsymbol{z}, \tag{B.44}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are both $l \times l$ complex matrices. Clearly, the GEP is a direct extension of the SEP, where the diagonal matrix in (B.39) is replaced by the matrix $\boldsymbol{B}$.

Definition B.9. Given two coefficient matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{l \times l}$, the **generalized eigenvalue problem (GEP)** consists in finding all scalars $\lambda \in \mathbb{C}$ and corresponding vectors $\boldsymbol{z} \in \mathbb{C}^{l \times 1} \setminus \{\boldsymbol{0}\}$, so that

$$\boldsymbol{\mathcal{M}}(\lambda)\boldsymbol{z} = (\boldsymbol{A} - \boldsymbol{B}\lambda)\boldsymbol{z} = \boldsymbol{0}. \tag{B.45}$$

The scalar λ and (non-zero) vectors $\boldsymbol{z}$ are the eigenvalues and eigenvectors of the GEP, respectively.

The expression $\boldsymbol{A} - \boldsymbol{B}\lambda$ with indeterminate λ is commonly called a *matrix pencil* (other word: *matrix pair*). A backward stable algorithm to compute the generalized eigenvalue is given by the QZ algorithm, which computes a generalization of the Schur decomposition.

An eigenvalue λ can be split into two scalars μ and ν to improve the interpretation. If $\mu \neq 0$, then (B.44) is equivalent to

$$\mu \boldsymbol{A}\boldsymbol{z} = \nu \boldsymbol{B}\boldsymbol{z}. \tag{B.46}$$

Note that scalars μ and ν are not uniquely determined by an eigenvector $\boldsymbol{z}$, but the ratio $\lambda = \frac{\nu}{\mu}$ is (except when $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{B}\boldsymbol{z} = 0$, which we briefly discuss below). When $\mu = 0$ and $\nu \neq 0$, then the GEP has an infinite eigenvalue $\lambda = \infty$. The next proposition considers some well-known facts about GEPs:

Proposition B.3. Let us consider two complex matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{l \times l}$ and a non-zero scalar $\lambda \in \mathbb{C} \setminus \{0\}$:

- λ is an eigenvalue of $(\boldsymbol{A}, \boldsymbol{B})$ if and only if $\frac{1}{\lambda}$ is an eigenvalue of $(\boldsymbol{B}, \boldsymbol{A})$.
- ∞ is an eigenvalue of $(\boldsymbol{A}, \boldsymbol{B})$ if and only if 0 is an eigenvalue of $(\boldsymbol{B}, \boldsymbol{A})$.
- ∞ is an eigenvalue of $(\boldsymbol{A}, \boldsymbol{B})$ if and only if $\boldsymbol{B}$ is a singular matrix.
- If $\boldsymbol{B}$ is a non-singular matrix, then the eigenvalues of $(\boldsymbol{A}, \boldsymbol{B})$ are exactly the eigenvalues of $\boldsymbol{B}^{-1}\boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{B}^{-1}$. If $\boldsymbol{z}$ is an eigenvector of $(\boldsymbol{A}, \boldsymbol{B})$ with associated eigenvalue λ, then $\boldsymbol{z}$ is an eigenvector of $\boldsymbol{B}^{-1}\boldsymbol{A}$ with eigenvalue λ and $\boldsymbol{B}\boldsymbol{z}$ is an eigenvector of $\boldsymbol{A}\boldsymbol{B}^{-1}$ with eigenvalue λ.

Also for GEPs, a scalar λ is an eigenvalue of the problem if and only if it is a solution of the characteristic equation $\chi(\lambda)$. The characteristic equation for a GEP is defined as

$$\chi(\lambda) = \det(\boldsymbol{A} - \lambda \boldsymbol{B}) = 0. \tag{B.47}$$

The characteristic polynomial is a polynomial in λ of degree l or less:

- If $\boldsymbol{B}$ is a non-singular matrix, then the GEP is formally equivalent with a SEP and the degree of the characteristic polynomial is equal to l.
- If $\boldsymbol{B}$ is a singular matrix, then the degree of the characteristic polynomial is less than l.

In fact, when the characteristic polynomial is of degree less than l, there is not a complete set of eigenvalues for the problem [174]. In some cases the missing eigenvalues may be regarded as "infinite". The term infinite eigenvalue is justified by the fact that if $\boldsymbol{B}$ is perturbed slightly so that it is no longer singular, there may appear a number of large eigenvalues that grow unboundedly as the perturbation is reduced to zero. It can happen that the characteristic polynomial is identically zero: $\chi(\lambda) \equiv 0$. If there is a non-zero vector $\boldsymbol{z}$ such that $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{B}\boldsymbol{z} = \boldsymbol{0}$, then $(\boldsymbol{A} - \lambda\boldsymbol{B})\boldsymbol{z} = \boldsymbol{0}$ for all λ and $\det(\boldsymbol{A} - \lambda\boldsymbol{B})$ is identically zero, i.e., $\boldsymbol{A}$ and $\boldsymbol{B}$ have a common null space and any λ may be regarded as an eigenvalue. We call the matrix pencil $(\boldsymbol{A}, \boldsymbol{B})$ in that case a singular pencil, while we focus in this text on regular pencils. Such problems have usually pathological features, and they are often referred to as ill-posed problems.

Theorem B.2. Every pair of matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{C}^{l \times l}$ that forms a regular pencil (i.e., $\chi(\lambda) \not\equiv 0$) has l (affine and infinite) eigenvalues $\lambda \in \mathbb{C} \cup \infty$.

Proof. This theorem follows directly from the fundamental theorem of algebra (Theorem A.1). □

B.5 Four important matrix decompositions

We can now review the four matrix decompositions used in this dissertation: the eigenvalue decomposition (Appendix B.5.1), Schur decomposition (Appendix B.5.2), QR decomposition (Appendix B.5.3), and singular value decomposition (Appendix B.5.4).

B.5.1 Eigenvalue decomposition

When a matrix is diagonalizable, it can be transformed into a diagonal matrix via a similarity transformation:

Theorem B.3. Let $\boldsymbol{A} \in \mathbb{C}^{l \times l}$. Then $\boldsymbol{A}$ is diagonalizable if and only if there exists a non-singular matrix $\boldsymbol{X} \in \mathbb{C}^{l \times l}$ and a diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{C}^{l \times l}$, so that

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1}. \tag{B.48}$$

The columns of $\boldsymbol{X}$ constitute a basis of $\mathbb{C}^l$ consisting of the eigenvectors of $\boldsymbol{A}$. This factorization is also called the **eigendecomposition** of the matrix $\boldsymbol{A}$.

Proof. A proof of this theorem can be found in [270]. □

Corollary. A matrix $\boldsymbol{A}$ is called **similar** to a matrix $\boldsymbol{B}$ if there is an invertible matrix $\boldsymbol{X}$ such that $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{B}\boldsymbol{X}^{-1}$. If $\boldsymbol{A}$ is similar to $\boldsymbol{B}$, then they have the same eigenvalues. Moreover, these eigenvalues occur with the same algebraic and geometric multiplicities.

Not every matrix can be diagonalized by a similarity transformation, only diagonalizable ones can. The theory of the Jordan canonical form exists because matrices that are defective exist. How close to diagonal form can a defective matrix be brought by a similarity transformation? We know from Schur's theorem [270] that every matrix is similar to an upper-triangular matrix. How much additional progress toward a diagonal form can we make?

Theorem B.4. Let $\boldsymbol{A} \in \mathbb{C}^{l\times l}$. Then $\boldsymbol{A}$ is similar to a matrix $\boldsymbol{J}$ that is a direct sum of Jordan blocks. $\boldsymbol{J}$ is uniquely determined up to the order of the blocks (i.e., the number and size of the blocks associated with each eigenvalue are uniquely determined, but the blocks can appear in any order on the main diagonal). $\boldsymbol{J}$ is called the **Jordan canonical form** of $\boldsymbol{A}$ and we can write

$$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{J}\boldsymbol{P}^{-1}, \tag{B.49}$$

where $\boldsymbol{P} \in \mathbb{C}^{l\times l}$ is a non-singular matrix.

Proof. A proof of this theorem can be found in [270]. □

A Jordan block associated with the eigenvalue λ is a matrix with λ's on the main diagonal, ones on the super-diagonal and zeros elsewhere. For example, a 4×4 Jordan block has the form

$$\boldsymbol{J}_\lambda = \begin{bmatrix} \lambda & 1 & 0 & 0 \\ 0 & \lambda & 1 & 0 \\ 0 & 0 & \lambda & 1 \\ 0 & 0 & 0 & \lambda \end{bmatrix}. \tag{B.50}$$

If the eigenvalue $\lambda = 0$, then we obtain a nilpotent Jordan block, which looks like

$$\boldsymbol{J}_0 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{B.51}$$

A diagonal matrix itself is a Jordan canonical form with n Jordan blocks $\boldsymbol{J}_\lambda$ of size 1×1. A matrix is similar to a diagonal matrix if and only if its Jordan canonical form is diagonal. If a matrix has a non-diagonal Jordan form, then it must necessarily have at least one repeated eigenvalue. Consequently, a matrix is not diagonalizable unless its Jordan form is diagonal.

We do not elaborate further on the Jordan canonical form, since they (almost) never occur in numerical computation in floating-point arithmetics. If a matrix has a non-diagonal Jordan form (i.e., if a matrix is defective), then any small perturbation of the matrix (for example, by the first round-off error of the similarity transformation) is likely to split the repeated eigenvalue into a cluster of nearby eigenvalues. The perturbed matrix is now non-defective; its Jordan form is diagonal.

B.5.2 Schur decomposition

Another decomposition that reveals the eigenvalues of a matrix is the Schur decomposition. Its advantage is that it can be computed (approximately) by only applying unitary similarity transformations.

Definition B.10. For a matrix $\boldsymbol{A}$, a decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{T}\boldsymbol{U}^{\mathrm{H}}$ is called a Schur decomposition if $\boldsymbol{T}$ is upper triangular and $\boldsymbol{U}$ is unitary.

It is clear that, if $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{T}\boldsymbol{U}^{\mathrm{H}} = \boldsymbol{U}\boldsymbol{T}\boldsymbol{U}^{-1}$ is a Schur decomposition, then $\boldsymbol{A}$ is similar to $\boldsymbol{T}$ and $\boldsymbol{T}$ has the eigenvalues of $\boldsymbol{A}$ on its diagonal. Every matrix has a Schur decomposition and it can be computed in a backward stable way [249].

B.5.3 QR decomposition

In general, any $m \times n$ matrix can be decomposed into the product of an unitary matrix $\boldsymbol{Q}$ and an upper-triangular matrix $\boldsymbol{R}$. This is called the (forward) QR decomposition.

Theorem B.5. Any rectangular complex matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ with $m \geq n$ can be decomposed as

$$\boldsymbol{A} = \underbrace{\begin{bmatrix}\boldsymbol{Q}_1 & \boldsymbol{Q}_2\end{bmatrix}}_{\boldsymbol{Q}} \underbrace{\begin{bmatrix}\boldsymbol{R}_1 \\ \boldsymbol{0}\end{bmatrix}}_{\boldsymbol{R}}, \tag{B.52}$$

where $\boldsymbol{Q} \in \mathbb{C}^{m\times n}$ is a unitary matrix and $\boldsymbol{R}_1 \in \mathbb{C}^{n\times n}$ is an upper-triangular matrix.

Every matrix has a QR factorization. The QR decomposition is usually computed using well-conditioned backward stable algorithms, like Householder reflections or Givens rotations to systematically create zeros under the diagonal of the matrix $\boldsymbol{A}$ [97, 249].

If we assume that $k \geq l$ and $\boldsymbol{A}$ has rank r, then a QR decomposition of $\boldsymbol{A}$ can be written as

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix}\boldsymbol{Q}_1 & \boldsymbol{Q}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{R}_1 \\ \boldsymbol{0}\end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R}_1. \tag{B.53}$$

The diagonal entries of $\boldsymbol{R}$ can be chosen real and positive, which makes the factorization $\boldsymbol{A} = \boldsymbol{Q}_1\boldsymbol{R}_1$, which is called the reduced QR factorization, unique.

The QR decomposition can be used to do several matrix operations, e.g., computing the rank of a matrix or constructing an orthonormal basis for the column space or right null space of a matrix. For example, for a matrix $\boldsymbol{A}$ of rank l, the columns of the matrix $\boldsymbol{Q}_1$ form an orthonormal basis for the column space of $\boldsymbol{A}$. Unfortunately, the full column rank assumption can not be dropped. A solution for this is given by a generalization of the QR decomposition, in which it is allowed to permute columns of $\boldsymbol{A}$.

Definition B.11. For a matrix $\boldsymbol{A} \in \mathbb{C}^{k\times l}$, a decomposition $\boldsymbol{A}\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{R}$ of $\boldsymbol{A}\boldsymbol{P}$ is called a **column pivoted QR factorization** of $\boldsymbol{A}$ if

- $\boldsymbol{P}$ is a column permutation matrix.
- $\boldsymbol{Q}$ is unitary.
- $\boldsymbol{R}$ is upper triangular.

B.5.4 Singular value decomposition

The singular value decomposition (SVD) of a matrix is sometimes called the "swiss army knife of numerical analysis" or "work horse of numerical linear algebra". It shows up in a multitude of applied mathematics techniques and is the corner stone of many mathematical modeling and analysis approaches.

Theorem B.6. The **singular value decomposition (SVD)** of a matrix $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ with $m \geq n$ is

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}}, \tag{B.54}$$

where $\boldsymbol{U} \in \mathbb{C}^{m \times m}$ and $\boldsymbol{V} \in \mathbb{C}^{n \times n}$ are unitary matrices. The matrix $\boldsymbol{\Sigma}$ is an $m \times n$ real matrix, having the following form:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \tag{B.55}$$

where $\boldsymbol{\Sigma}_r$ is an $r \times r$ diagonal matrix when $\operatorname{rank}(\boldsymbol{A}) = r$. The elements on the diagonal of $\boldsymbol{\Sigma}_r$ are called the singular values of $\boldsymbol{A}$ and are denoted by $\sigma_1 \geq \cdots \geq \sigma_r > 0$. The are ordered in descending order on the diagonal.

The SVD exists for any $m \times n$ matrix. Moreover, the singular values σ_i are uniquely determined. This is not true for the eigenvalue decompositions (which only exists for square matrices), although there exists a strong connection between both decompositions: the columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{H}}$ and the columns of $\boldsymbol{V}$ are the eigenvectors of $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{A}$. The r singular values on the diagonal of $\boldsymbol{\Sigma}_r$ are the square roots of the non-zero eigenvalues of both $\boldsymbol{A}\boldsymbol{A}^{\mathrm{H}}$ and $\boldsymbol{A}^{\mathrm{H}}\boldsymbol{A}$.

Some problems for which we use the SVD in this dissertation are calculating the rank of a matrix (rank checks), computing numerical basis matrix of a subspace of a matrix, determining the pseudo-inverse of a matrix, and performing row and column compressions.

Rank of a matrix. The rank of a matrix can be read off as the number of nonzero singular values of that matrix. The SVD is the most reliable method for determining the (numerical) rank of a matrix, which is in practice done by counting the number of singular values that are greater than a certain user-defined threshold value ϵ. This also highlight the difficulty of numerical rank checks. Suppose that the gap between σ_r and σ_{r+1} is small, then it is clear that the matrix is nearly as close to being rank $r-1$ as to being rank r, and the partitioning can be very sensitive to the threshold value ϵ. In Matlab, the default threshold is

$$\max(m, n)\operatorname{eps}(\sigma_1), \tag{B.56}$$

where $\operatorname{eps}(\sigma_1)$ is the distance from σ_1 to the next larger in magnitude floating point number of the same precision as σ_1. We sometimes also use an absolute threshold in MacaulayLab, for example $\epsilon = 1 \times 10^{-10}$.

Numerical basis matrix for subspaces. The SVD also has an interesting geometric interpretation. Let us consider the SVD of a real matrix $\boldsymbol{A} \in \mathbb{C}^{m \times n}$. If we partition $\boldsymbol{U}$ and $\boldsymbol{V}$ in accordance to $\boldsymbol{\Sigma}$, then we obtain

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{U}_1 & \boldsymbol{U}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_1^{\mathrm{H}} \\ \boldsymbol{V}_2^{\mathrm{H}} \end{bmatrix}. \tag{B.57}$$

The two different factorizations of the matrix $\boldsymbol{A}$,

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}} \quad \text{and} \quad \boldsymbol{A} = \boldsymbol{U}_1\boldsymbol{\Sigma}_r\boldsymbol{V}_1^{\mathrm{H}}, \tag{B.58}$$

are sometimes also called the full SVD and reduced (or thin/economy) SVD of the matrix. The SVD provides numerical basis matrices for the four fundamental subspaces of a matrix. We have that

- $\mathrm{range}\big(\boldsymbol{U}_1\big) = \mathrm{range}\big(\boldsymbol{A}\big)$,
- $\mathrm{range}\big(\boldsymbol{U}_2\big) = \mathrm{null}\big(\boldsymbol{A}^{\mathrm{T}}\big)$,
- $\mathrm{range}\big(\boldsymbol{V}_1\big) = \mathrm{range}\big(\boldsymbol{A}^{\mathrm{T}}\big)$,
- $\mathrm{range}\big(\boldsymbol{V}_2\big) = \mathrm{null}\big(\boldsymbol{A}\big)$.

Hence, the SVD creates the following numerical basis matrices:

- the last $q - r$ columns of $\boldsymbol{V}$ are a basis matrix for the right null space,
- the first r columns of $\boldsymbol{U}$ are a basis matrix for the column space,
- the first r columns of $\boldsymbol{V}$ are a basis matrix for the row space,
- and the last $p - r$ columns of $\boldsymbol{U}$ are a basis matrix for the left null space.

Pseudo-inverse of a matrix. The inverse of a rectangular matrix or square singular matrix is not defined. However, there exists a generalization of the inversion operation, called the pseudo-inverse of a matrix. For $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ with $m \geq n$ and its SVD $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}}$, the Moore–Penrose pseudo-inverse of a matrix, denoted by $\boldsymbol{A}^{\dagger}$, is defined as

$$\boldsymbol{A}^{\dagger} = \boldsymbol{V}\boldsymbol{\Sigma}^{\dagger}\boldsymbol{U}^{\mathrm{H}}, \tag{B.59}$$

where $\boldsymbol{\Sigma}^{\dagger} = \mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_r, 0, \ldots, 0)$.

Row and column compression. Because it uses orthogonal transformations, the SVD can be used to compress a matrix in a numerically reliable way, which means that it can transform a matrix into a smaller matrix with vectors that span that matrix.

Theorem B.7. Let the SVD of $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ be given by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}}$. Then, we have that

$$\begin{aligned} \boldsymbol{U}^{\mathrm{H}}\boldsymbol{A} &= \boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}} && \text{(B.60)} \\ &= \begin{bmatrix} \boldsymbol{\Sigma}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}_1^{\mathrm{H}} \\ \boldsymbol{V}_2^{\mathrm{H}} \end{bmatrix} && \text{(B.61)} \\ &= \begin{bmatrix} \boldsymbol{\Sigma}_r \boldsymbol{V}_1^{\mathrm{H}} \\ \boldsymbol{0} \end{bmatrix}, && \text{(B.62)} \end{aligned}$$

with $\mathrm{null}(\boldsymbol{A}) = \mathrm{null}(\boldsymbol{U}^{\mathrm{H}}\boldsymbol{A}) = \mathrm{null}(\boldsymbol{\Sigma}_r\boldsymbol{V}_1^{\mathrm{H}})$, which means that the pre-multiplication of $\boldsymbol{A}$ by $\boldsymbol{U}^{\mathrm{H}}$ is the **row compression** of $\boldsymbol{A}$. The matrix $\boldsymbol{\Sigma}_r\boldsymbol{V}_1^{\mathrm{H}} \in \mathbb{C}^{r\times n}$ has full row rank.

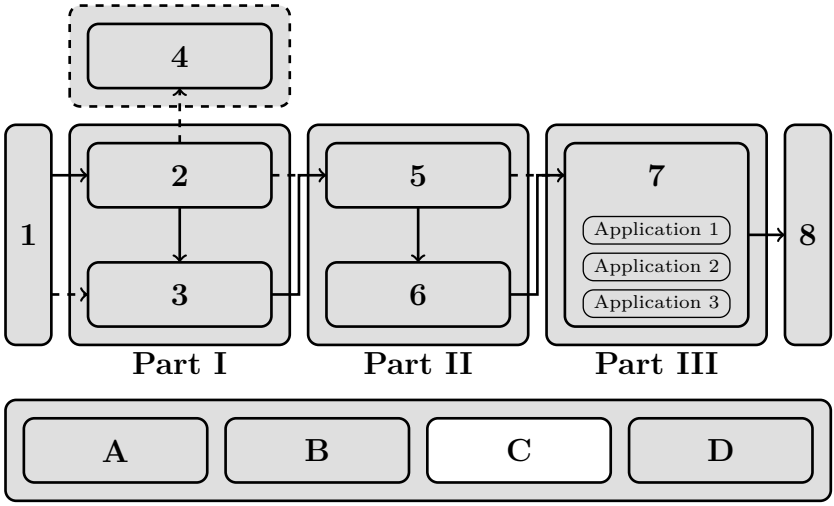
Theorem B.8. Let the SVD of $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ be given by $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{H}}$. Then, we have that

$$\begin{aligned} \boldsymbol{A}\boldsymbol{V} &= \boldsymbol{U}\boldsymbol{\Sigma} && \text{(B.63)} \\ &= \begin{bmatrix} \boldsymbol{U}_1 & \boldsymbol{U}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Sigma}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} && \text{(B.64)} \\ &= \begin{bmatrix} \boldsymbol{U}_1\boldsymbol{\Sigma}_r & \boldsymbol{0} \end{bmatrix}, && \text{(B.65)} \end{aligned}$$

with $\mathrm{range}(\boldsymbol{A}) = \mathrm{range}(\boldsymbol{A}\boldsymbol{V}) = \mathrm{range}(\boldsymbol{U}_1\boldsymbol{\Sigma}_r)$, which means that the post-multiplication of $\boldsymbol{A}$ by $\boldsymbol{V}$ is the **row compression** of $\boldsymbol{A}$. The matrix $\boldsymbol{U}_1\boldsymbol{\Sigma}_r \in \mathbb{C}^{m\times r}$ has full column rank.

Systems Theory and Shift-Invariant Subspaces

The algorithms in Chapters 2 and 3 strongly rely on the concept of (backward) shift-invariance. Shift-invariance of a subspace is usually defined for infinite matrices, i.e., operators [93]. De Cock and De Moor [69] have adapted in their paper the definition of backward scalar/block single/multi-shift-invariance to finite dimensional subspaces. It is this adaptation that we use in our algorithms and present in this appendix.



Outline. Firstly, Appendix C.1 contains a summary of the system theoretical concepts used in this dissertation. Next, in Appendix C.2, we consider the four different cases of shift-invariance that are encountered throughout this dissertation. Finally, we interpret the solution approaches of this dissertation as applications of multidimensional realization theory in Appendix C.3.

C.1 Systems theory

We give a summary of some important concepts from systems theory, which we need in Appendix C.3 to interpret the solution algorithms of this dissertation in terms of multidimensional realization theory. A more elaborate introduction to the subject of systems theory can be found in textbooks like [91, 128, 192].

C.1.1 Description of a dynamical system

A dynamic linear time-invariant (LTI) discrete time system can be described by its state space representation $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}_k, \tag{C.1}$$
$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}_k, \tag{C.2}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n\times 1}$ is called the state vector (with initial value $\boldsymbol{x}_0$) at time instant k, $\boldsymbol{u}_k \in \mathbb{R}^{p\times 1}$ denotes the input at time instant k, and $\boldsymbol{y}_k \in \mathbb{R}^{q\times 1}$ denotes the output at time instant k. Starting from a given initial state, we can compute the output sequence for a given input sequence,

$$\boldsymbol{x}_1 = \boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{B}\boldsymbol{u}_0, \tag{C.3}$$
$$\boldsymbol{y}_0 = \boldsymbol{C}\boldsymbol{x}_0 + \boldsymbol{D}\boldsymbol{u}_0, \tag{C.4}$$
$$\Downarrow$$
$$\boldsymbol{x}_2 = \boldsymbol{A}^2\boldsymbol{x}_0 + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{B}\boldsymbol{u}_1, \tag{C.5}$$
$$\boldsymbol{y}_1 = \boldsymbol{C}\boldsymbol{A}\boldsymbol{x}_0 + \boldsymbol{C}\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{D}\boldsymbol{u}_1, \tag{C.6}$$
$$\Downarrow$$
$$\boldsymbol{x}_3 = \boldsymbol{A}^3\boldsymbol{x}_0 + \boldsymbol{A}^2\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_1 + \boldsymbol{B}\boldsymbol{u}_2, \tag{C.7}$$
$$\boldsymbol{y}_2 = \boldsymbol{C}\boldsymbol{A}^2\boldsymbol{x}_0 + \boldsymbol{C}\boldsymbol{A}\boldsymbol{B}\boldsymbol{u}_0 + \boldsymbol{C}\boldsymbol{B}\boldsymbol{u}_1 + \boldsymbol{D}\boldsymbol{u}_2, \tag{C.8}$$
$$\vdots$$

which shows us how the dynamical system evolves over time. In general, we obtain the following expression:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}^{k+1}\boldsymbol{x}_0 + \sum_{i=0}^{k} \boldsymbol{A}^{k-i}\boldsymbol{B}\boldsymbol{u}_i, \tag{C.9}$$

$$\boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{x}_0 + \sum_{i=0}^{k-1} \boldsymbol{C}\boldsymbol{A}^{k-i-1}\boldsymbol{B}\boldsymbol{u}_i + \boldsymbol{D}\boldsymbol{u}_k. \tag{C.10}$$

A fundamental result in systems theory is that the behavior of a dynamical LTI system can completely by characterized by its impulse response. Let us explain this by starting with a single-input/single-output (SISO) system. When dealing with a SISO system, the input and output are scalar signals, i.e., $p = q = 1$. The impulse response of a SISO system is the sequence of outputs y_k

obtained by applying the impulse input signal $u_k = \delta_k$, defined as

$$\delta_k = \begin{cases} 1 & \text{for } k = 0, \\ 0 & \text{for } k \neq 0. \end{cases} \tag{C.11}$$

It can be seen from (C.10) that the output of the dynamical system, which is called the impulse response h_k, is then given by

$$h_k = \begin{cases} d & \text{for } k = 0, \\ \boldsymbol{c}^{\mathrm{T}} \boldsymbol{A}^{k-1} \boldsymbol{b} & \text{for } k \neq 0. \end{cases} \tag{C.12}$$

The output of a SISO system to any given input is simply the convolution of the input with the system's impulse response. For multiple-input/multiple-output (MIMO) systems the same ideas remain valid, but now with a impulse response matrix $\boldsymbol{H}_k$,

$$\boldsymbol{H}_k = \begin{cases} \boldsymbol{D} & \text{for } k = 0, \\ \boldsymbol{C}\boldsymbol{A}^{k-1}\boldsymbol{B} & \text{for } k \neq 0, \end{cases} \tag{C.13}$$

where every ij-th element corresponds to the impulse response of the ith output on an impulse on the jth input. These matrices are sometimes also called the Markov parameters of the dynamical system.

C.1.2 Ho–Kalman's realization algorithm

An important problem in systems theory (and system identification) is the so-called realization problem.

> **Definition C.1.** Given a set of impulse response matrices $\boldsymbol{H}_k$, for $k = 0, \ldots, N-1$, the **realization problem** is the problem of finding the state dimension n and a system realization $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$.

By embedding the observed impulse response information in an appropriately sized Hankel matrix, the essential information about the underlying dynamical system is revealed: the rank of the constructed Hankel matrix corresponds to the McMillan degree of the underlying dynamical LTI system and a state-space model (the so-called realization) can be estimated from a decomposition of this Hankel matrix. Note that the state space realization of a dynamical system is not unique.

Let us first introduce three more matrices, before considering the realization algorithm of Ho and Kalman [116] in Algorithm C.1:

- The extended observability matrix $\mathcal{O}_i$ with $i > n$ is defined as

$$\mathcal{O}_i = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\boldsymbol{A} \\ \vdots \\ \boldsymbol{C}\boldsymbol{A}^{i-1} \end{bmatrix}. \tag{C.14}$$

Algorithm C.1 Ho–Kalman's realization algorithm

Require: $\boldsymbol{H}_k$ for $k = 0, \ldots, K$
1: The matrix $\boldsymbol{D}$ can be found easily as $\boldsymbol{D} = \boldsymbol{H}_0$
2: Construct the block Hankel matrix $\mathcal{H}_{ij}$ as in (C.16)
3: Take the SVD of $\mathcal{H}_{ij}$: $\mathcal{H}_{ij} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$
4: $\mathcal{O}_i = \boldsymbol{U}\boldsymbol{\Sigma}^{\frac{1}{2}}$ and $\mathcal{C}_j = \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{V}^{\mathrm{T}}$
5: n is the rank of $\mathcal{H}_{ij}$
6: $\boldsymbol{C}$ is formed from the first p rows of $\mathcal{O}_i$, while $\boldsymbol{B}$ is formed from the first q columns of $\mathcal{C}_j$
7: $\boldsymbol{A} = \underline{\mathcal{O}_i}^{\dagger}\overline{\mathcal{O}_i}$
8: **return** n and $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$

- The extended controllability matrix $\mathcal{C}_j$ with $j > n$ is defined as

$$\mathcal{C}_j = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A}\boldsymbol{B} & \cdots & \boldsymbol{A}^{j-1}\boldsymbol{B} \end{bmatrix}. \tag{C.15}$$

  The $(pi) \times (qj)$ block Hankel matrix $\mathcal{H}_{ij}$ with block dimensions i and j $(i + j - 1 \leq N)$ is defined as

$$\mathcal{H}_{ij} = \begin{bmatrix} \boldsymbol{H}_1 & \boldsymbol{H}_2 & \boldsymbol{H}_3 & \cdots & \boldsymbol{H}_j \\ \boldsymbol{H}_2 & \boldsymbol{H}_3 & \cdots & \boldsymbol{H}_j & \boldsymbol{H}_{j+1} \\ \boldsymbol{H}_3 & \cdots & \boldsymbol{H}_j & \boldsymbol{H}_{j+1} & \boldsymbol{H}_{j+2} \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ \boldsymbol{H}_i & \boldsymbol{H}_{i+1} & \boldsymbol{H}_{i+2} & \cdots & \boldsymbol{H}_{i+j-1} \end{bmatrix}. \tag{C.16}$$

A consequence from the definition of $\boldsymbol{H}_k$ is that the block Hankel matrix $\mathcal{H}_{ij}$ can be factorized into the product of the extended observability matrix and the extended controllability matrix, i.e.,

$$\mathcal{H}_{ij} = \mathcal{O}_i \mathcal{C}_j. \tag{C.17}$$

If i and j are sufficiently large, this block Hankel matrix is rank deficient and its rank is equal to the McMillan degree n of the dynamical system. This insight leads to the realization algorithm of Ho and Kalman [116], outlined in Algorithm C.1. By using the so-called shift trick (cf., Appendix C.3), it is possible to obtain the system matrix $\boldsymbol{A}$, while the other matrices of the state space representation can be retrieved from the definitions of the Hankel matrix, extended observability matrix, and extended controllability matrix.

C.2 Shift-invariant subspaces

We define the (backward[1]) scalar single-shift-invariance of a subspace via the column space (i.e., range) of a matrix that represents that subspace.

[1]Note that sometimes ambiguity arises when considering the shift operator. In this text, we adopt the convention of Garcia et al. [93], and we define the backward shift operator as $\mathcal{S}\{f(z)\} = \frac{f(z)-f(0)}{z}$, or, in terms of Taylor coefficients $\{a_i\}_{i\geq 0}$ of $f(z)$, as $\mathcal{S}\{(a_0, a_1, \ldots)\} = (a_1, a_2, \ldots)$.

Definition C.2. Let $\mathcal{C}(\boldsymbol{G})$ be the column space of a matrix $\boldsymbol{G} \in \mathbb{C}^{m \times n}$ with full column rank. $\mathcal{C}(\boldsymbol{G})$ is **(backward) scalar single-shift-invariant** if and only if

$$\mathcal{C}\left(\overline{\boldsymbol{G}}\right) \subseteq \mathcal{C}(\underline{\boldsymbol{G}}), \tag{C.18}$$

where $\overline{\boldsymbol{G}}$ and $\underline{\boldsymbol{G}}$ (with full column rank) are the matrix $\boldsymbol{G}$ without its first and last row, respectively.

The backward scalar single-shift-invariance of $\mathcal{C}(\boldsymbol{G})$ can also be expressed as

$$\exists \boldsymbol{\Gamma} \in \mathbb{C}^{n \times n} : \underline{\boldsymbol{G}}\boldsymbol{\Gamma} = \overline{\boldsymbol{G}}, \tag{C.19}$$

where $\mathcal{C}\left(\overline{\boldsymbol{G}}\right) = \mathcal{C}(\underline{\boldsymbol{G}})$ if $\boldsymbol{\Gamma}$ is non-singular (and otherwise $\mathcal{C}\left(\overline{\boldsymbol{G}}\right) \subsetneq \mathcal{C}(\underline{\boldsymbol{G}})$). These row selection operations can be written via row selection matrices $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, similar to the mathematical formulation in other parts of this dissertation. The scalar single-shift-invariance property in Definition C.2 can then be expressed as

$$\exists \boldsymbol{\Gamma} \in \mathbb{C}^{n \times n} : (\boldsymbol{S}_1\boldsymbol{G})\boldsymbol{\Gamma} = (\boldsymbol{S}_2\boldsymbol{G}), \tag{C.20}$$

where $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ select all the rows of the matrix $\boldsymbol{G}$ except the last one and the first one, respectively. Note that shift-invariance is a property of the vector space, and not of the specific basis matrix of this vector space. Consider a second basis matrix $\boldsymbol{H} \in \mathbb{C}^{m \times n}$, then there exists a basis transformation via a non-singular matrix $\boldsymbol{T} \in \mathbb{C}^{n \times n}$, such that $\boldsymbol{G} = \boldsymbol{H}\boldsymbol{T}$. Consequently, we can rewrite (C.20) as

$$(\boldsymbol{S}_1\boldsymbol{H})\boldsymbol{\Lambda} = (\boldsymbol{S}_2\boldsymbol{H}), \quad \text{with } \boldsymbol{\Lambda} = \boldsymbol{T}\boldsymbol{\Gamma}\boldsymbol{T}^{-1}. \tag{C.21}$$

Clearly, the matrices $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$ are similar, and hence have the same spectrum. The spectrum is an invariant property of a (backward) shift-invariance subspace. This observation is one of the key observations of the (block) Macaulay matrix approach in this dissertation.

Example C.1. Consider a univariate Vandermonde matrix $\boldsymbol{V}_U \in \mathbb{C}^{(d+1) \times n}$ (with n distinct columns and degree d). The column space of this matrix clearly exhibits scalar single-shift invariance:

$$\underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_n \\ \vdots & & \vdots \\ \alpha_1^{d-1} & \cdots & \alpha_n^{d-1} \end{bmatrix}}_{\boldsymbol{S}_1\boldsymbol{V}_U} \underbrace{\begin{bmatrix} \alpha_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha_n \end{bmatrix}}_{\boldsymbol{\Gamma}} = \underbrace{\begin{bmatrix} \alpha_1 & \cdots & \alpha_n \\ \vdots & & \vdots \\ \alpha_1^{d-1} & \cdots & \alpha_n^{d-1} \\ \alpha_1^{d} & \cdots & \alpha_n^{d} \end{bmatrix}}_{\boldsymbol{S}_2\boldsymbol{V}_U}. \tag{C.22}$$

Definition C.2 covers scalar single-shift-invariance: *scalar* meaning that we shift row-wise and *single-shift* meaning that only one shift direction is possible in the subspace. We can extend this definition and introduce an additional shift direction (or variable in that Vandermonde example), which leads to the concept of scalar multi-shift-invariance.

Example C.2. An example of scalar multi-shift-invariance is given by the column space of the bivariate Vandermonde matrix $\boldsymbol{V}_M \in \mathbb{C}^{(d^2+3d+2)/2\times n}$ (with n distinct columns and degree d),

$$\boldsymbol{V}_M = \begin{bmatrix} 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_n \\ \beta_1 & \cdots & \beta_n \\ \alpha_1^2 & \cdots & \alpha_n^2 \\ \vdots & & \vdots \\ \beta_1^d & \cdots & \beta_n^d \end{bmatrix}, \tag{C.23}$$

which can be shifted by both variables α and β. As an example, we shift the first three rows of $\boldsymbol{V}_M$:

$$\begin{aligned} \underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_n \\ \beta_1 & \cdots & \beta_n \end{bmatrix}}_{\boldsymbol{S}_1\boldsymbol{V}_M} \underbrace{\begin{bmatrix} \alpha_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha_n \end{bmatrix}}_{\boldsymbol{\Gamma}_\alpha} &= \underbrace{\begin{bmatrix} \alpha_1 & \cdots & \alpha_n \\ \alpha_1^2 & \cdots & \alpha_n^2 \\ \alpha_1\beta_1 & \cdots & \alpha_n\beta_n \end{bmatrix}}_{\boldsymbol{S}_2\boldsymbol{V}_M} \\ \underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ \alpha_1 & \cdots & \alpha_n \\ \beta_1 & \cdots & \beta_n \end{bmatrix}}_{\boldsymbol{S}_1\boldsymbol{V}_M} \underbrace{\begin{bmatrix} \beta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \beta_n \end{bmatrix}}_{\boldsymbol{\Gamma}_\beta} &= \underbrace{\begin{bmatrix} \beta_1 & \cdots & \beta_n \\ \alpha_1\beta_1 & \cdots & \alpha_n\beta_n \\ \beta_1^2 & \cdots & \beta_n^2 \end{bmatrix}}_{\boldsymbol{S}_3\boldsymbol{V}_M}, \end{aligned} \tag{C.24}$$

where the row selection matrices $\boldsymbol{S}_2$ and $\boldsymbol{S}_3$ select the rows of the bivariate Vandermonde matrix after a shift of the first three rows (the matrix $\boldsymbol{S}_1$) by α (the matrix $\boldsymbol{\Gamma}_\alpha$) and by β (the matrix $\boldsymbol{\Gamma}_\beta$), respectively.

While the column space of the matrix $\boldsymbol{V}_U$ is scalar single-shift-invariant, some subspaces are block single-shift-invariant, which means that we can shift entire block rows of the basis matrix, i.e., $\mathcal{C}(\boldsymbol{S}_1\boldsymbol{G}) \subseteq \mathcal{C}(\boldsymbol{S}_2\boldsymbol{G})$, where the row selection matrices $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ select the entire matrix $\boldsymbol{G}$ without the first and last block row, respectively.

Example C.3. A block univariate Vandermonde matrix $\boldsymbol{V}_B \in \mathbb{C}^{(d+1)s\times n}$ (with a vector $\boldsymbol{z}_i \in \mathbb{C}^{s\times 1}, i = 1,\ldots,n$) exhibits this property:

$$\underbrace{\begin{bmatrix} \boldsymbol{z}_1 & \cdots & \boldsymbol{z}_n \\ \alpha_1\boldsymbol{z}_1 & \cdots & \alpha_n\boldsymbol{z}_n \\ \vdots & & \vdots \\ \alpha_1^{d-1}\boldsymbol{z}_1 & \cdots & \alpha_n^{d-1}\boldsymbol{z}_n \end{bmatrix}}_{\boldsymbol{S}_1\boldsymbol{V}_B} \underbrace{\begin{bmatrix} \alpha_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha_n \end{bmatrix}}_{\boldsymbol{\Gamma}} = \underbrace{\begin{bmatrix} \alpha_1\boldsymbol{z}_1 & \cdots & \alpha_n\boldsymbol{z}_n \\ \vdots & & \vdots \\ \alpha_1^{d-1}\boldsymbol{z}_1 & \cdots & \alpha_n^{d-1}\boldsymbol{z}_n \\ \alpha_1^{d}\boldsymbol{z}_1 & \cdots & \alpha_n^{d}\boldsymbol{z}_n \end{bmatrix}}_{\boldsymbol{S}_2\boldsymbol{V}_B}. \tag{C.25}$$

The block multi-shift-invariant subspace is a natural extension of the previous types of shift-invariant subspaces. This property appears multiple times

in this dissertation, since the (affine) right null space of the block Macaulay matrix is block multi-shift-invariant.

Example C.4. As an example, we shift the first three block rows of the block multivariate Vandermonde matrix (compare to (3.54) with $\boldsymbol{\lambda} = (\alpha, \beta)$):

$$\begin{aligned}
&\underbrace{\begin{bmatrix} \boldsymbol{z}_1 & \cdots & \boldsymbol{z}_n \\ \alpha_1\boldsymbol{z}_1 & \cdots & \alpha_n\boldsymbol{z}_n \\ \beta_1\boldsymbol{z}_1 & \cdots & \beta_n\boldsymbol{z}_n \end{bmatrix}}_{\boldsymbol{S}_1\boldsymbol{V}} \underbrace{\begin{bmatrix} \alpha_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha_n \end{bmatrix}}_{\boldsymbol{\Gamma}_\alpha} = \underbrace{\begin{bmatrix} \alpha_1\boldsymbol{z}_1 & \cdots & \alpha_n\boldsymbol{z}_1 \\ \alpha_1^2\boldsymbol{z}_1 & \cdots & \alpha_n^2\boldsymbol{z}_n \\ \alpha_1\beta_1\boldsymbol{z}_1 & \cdots & \alpha_n\beta_n\boldsymbol{z}_n \end{bmatrix}}_{\boldsymbol{S}_2\boldsymbol{V}} \\
&\underbrace{\begin{bmatrix} \boldsymbol{z}_1 & \cdots & \boldsymbol{z}_n \\ \alpha_1\boldsymbol{z}_1 & \cdots & \alpha_n\boldsymbol{z}_n \\ \beta_1\boldsymbol{z}_1 & \cdots & \beta_n\boldsymbol{z}_n \end{bmatrix}}_{\boldsymbol{S}_1\boldsymbol{V}} \underbrace{\begin{bmatrix} \beta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \beta_n \end{bmatrix}}_{\boldsymbol{\Gamma}_\beta} = \underbrace{\begin{bmatrix} \beta_1\boldsymbol{z}_1 & \cdots & \beta_n\boldsymbol{z}_n \\ \alpha_1\beta_1\boldsymbol{z}_1 & \cdots & \alpha_n\beta_n\boldsymbol{z}_n \\ \beta_1^2\boldsymbol{z}_1 & \cdots & \beta_n^2\boldsymbol{z}_n \end{bmatrix}}_{\boldsymbol{S}_3\boldsymbol{V}}.
\end{aligned} \tag{C.26}$$

C.3 Multidimensional realization theory

For multidimensional dynamical systems, the realization problem of Definition C.1 corresponds to the following question: "How can we obtain a state-space realization from a given set of multivariate difference equations?" The equivalence between (C.20),

$$(\boldsymbol{S}_1\boldsymbol{G})\boldsymbol{\Gamma} = (\boldsymbol{S}_2\boldsymbol{G}), \tag{C.27}$$

and (C.21),

$$(\boldsymbol{S}_1\boldsymbol{H})\boldsymbol{\Lambda} = (\boldsymbol{S}_2\boldsymbol{H}), \quad \text{with } \boldsymbol{\Lambda} = \boldsymbol{T}\boldsymbol{\Gamma}\boldsymbol{T}^{-1}. \tag{C.28}$$

show that the matrices $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$ are similar, and hence, have the same spectrum. The spectrum is thus an invariant property of a (backward) shift-invariant subspace. This property allows us to interpret the basis matrices of (backward) shift-invariant column spaces as observability matrices of multidimensional dynamical systems. In that regard, we can see the null space based solution approach, where we search for the spectrum of $\boldsymbol{\Gamma}$ (or $\boldsymbol{\Lambda}$), as a multidimensional realization problem: a basis matrix of the right null space of the (block) Macaulay matrix can, after stabilization of the degree, be interpreted as a multidimensional observability matrix. This multidimensional observability matrix is generated by a multidimensional descriptor system [80, 259]. In this basis matrix, we find a multidimensional realization problem when we select two blocks and use the multiplicative relationship of the shift-invariant subspace. The eigenvalues that we obtain are the eigenvalues of the system matrices of the generating multidimensional descriptor system. Then, the full rank condition of $\boldsymbol{G}$ is equivalent with the dynamical system being observable and the full rank condition of $\underline{\boldsymbol{G}}$ is the partial realization condition, required for a unique solution [69, 80]. This observation has been made in [80] for the scalar case and in [259] for the block case.

Example C.5. Consider the (block) column echelon basis matrix $\boldsymbol{H}$ of the right null space of a (block) Macaulay matrix with two parameters α and β (i.e., two variables or two eigenvalue parameters):

$$\boldsymbol{H} = \left[\begin{array}{cc} \boldsymbol{C}_{\mathrm{R}} & \boldsymbol{0} \\ \hline \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\alpha} & \boldsymbol{0} \\ \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\beta} & \boldsymbol{0} \\ \hline \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\alpha}^{2} & \boldsymbol{0} \\ \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\alpha}\boldsymbol{A}_{\beta} & \boldsymbol{0} \\ \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\beta}^{2} & \boldsymbol{0} \\ \hline \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\alpha}^{3} & \boldsymbol{0} \\ \vdots & \vdots \\ \hline \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\alpha}^{n} & \boldsymbol{0} \\ \vdots & \vdots \\ \hline \times & \boldsymbol{C}_{\mathrm{S}}\boldsymbol{E}_{\alpha}^{m-1} \\ \vdots & \vdots \end{array}\right] \tag{C.29}$$

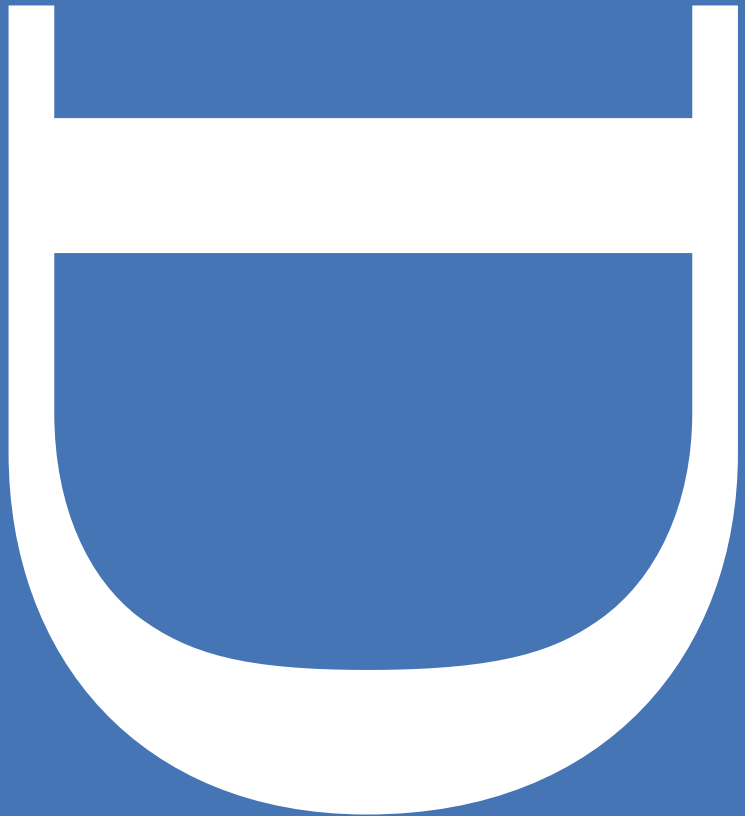
The regular columns of this block column echelon basis matrix $\boldsymbol{H}$, i.e., the m_a left-most columns corresponding to the affine solutions, and the singular columns, i.e., the remaining $(m_b - m_a)$ right-most columns corresponding to solutions at infinity, determine two observability matrices $\boldsymbol{\Gamma}_{\mathrm{R}}$ and $\boldsymbol{\Gamma}_{\mathrm{S}}$, which are generated by a multidimensional descriptor system. In this multidimensional observability matrix, we find the multidimensional realization problem that yield us the GEP(s) to solve the seed equation(s). Indeed, if one selects two blocks of $\boldsymbol{H}$, for example $\boldsymbol{S}_{\alpha}\boldsymbol{H} = \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\alpha}$ and $\boldsymbol{S}_{\alpha,\beta}\boldsymbol{H} = \boldsymbol{C}_{\mathrm{R}}\boldsymbol{A}_{\alpha}\boldsymbol{A}_{\beta}$ ($\boldsymbol{S}_{\alpha}$ and $\boldsymbol{S}_{\alpha,\beta}$ operate as row selection matrices), we recognize the solution approach in the right null space:

$$\boldsymbol{S}_{\alpha}\boldsymbol{H}\boldsymbol{A}_{\beta} = \boldsymbol{S}_{\alpha,\beta}\boldsymbol{H}, \tag{C.30}$$

or

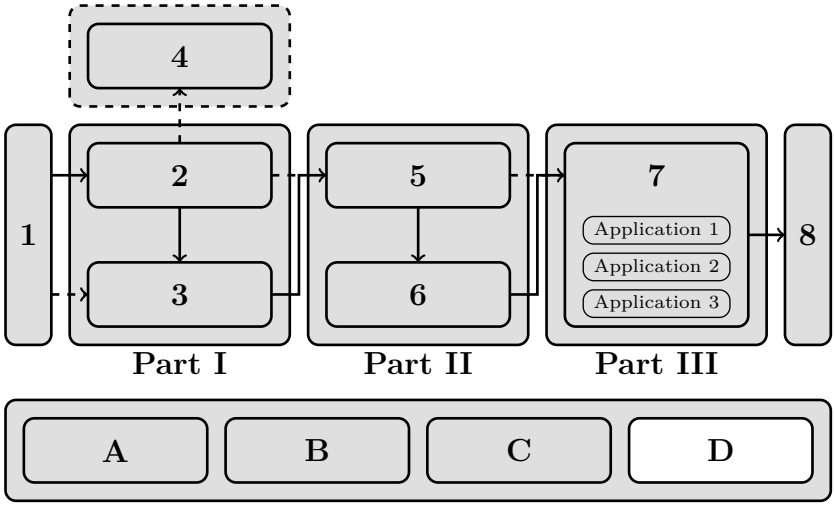
$$\boldsymbol{A}_{\beta} = (\boldsymbol{S}_{\alpha}\boldsymbol{H})^{\dagger}(\boldsymbol{S}_{\alpha,\beta}\boldsymbol{H}). \tag{C.31}$$

The eigenvalues of all the system matrices $\boldsymbol{A}_{\alpha}$ and $\boldsymbol{A}_{\beta}$ constitute the solutions of the seed equation(s).



User-Manual of the MacaulayLab Toolbox

This appendix contains the user-manual of the MacaulayLab toolbox (version `1.0`) developed alongside this dissertation. Via several examples, the user is guided through the available functions of the software. The user-manual explains how to solve systems of multivariate polynomial equations and rectangular multiparameter eigenvalue problems.



Outline. The user-manual helps the user getting started with the toolbox in Appendix D.1. Next, in Appendix D.2, we explain how to represent a problem, being a system of multivariate polynomial equations or a rectangular multiparameter eigenvalue problems. Solving these problems via the (block) Macaulay matrix is the topic of Appendix D.3, while the user-manual shows how to change the monomial ordering or polynomial basis in Appendix D.4. Finally, some other useful functions of the toolbox are highlighted in Appendix D.5

Quick start. For those users that are not interested in the full user-manual of MacaulayLab, but just want to solve problems as soon as possible, we recommend jumping to Appendix D.1.3 for a minimal introduction to the software. This part of the text should provide you with all the necessary information (and nothing more) to solve your problem.

D.1 Getting started

MacaulayLab is a Matlab toolbox that features algorithms to solve systems of multivariate polynomial equations and rectangular multiparameter eigenvalue problems (MEPs). It also contains a database with many test problems. Before using MacaulayLab, you need to download the zip archive of the toolbox from the website `www.macaulaylab.net` and unzip MacaulayLab to any directory. You could also clone the latest version of the stable repository. This user-manual is based on version `1.0` of the software. You can check the current version of MacaulayLab that you use via `ver MacaulayLab`.

D.1.1 Installation

Afterward unzipping MacaulayLab to a directory, you can browse to that location in Matlab and add the path.

Code D.1. It is easy to add MacaulayLab to your path:

```
>> addpath(genpath(pwd));
>> savepath;
```

D.1.2 Help and documentation

The different functions of MacaulayLab are well-documented. Using the function `help function` in the command line displays more information about the functionality and interface of that function.

Code D.2. The documentation of the `nbmonomials` function:

```
>> help nbmonomials

nbmonomials calculates the number of monomials.

    [s] = nbmonomials(d,n) calculates the number of
    monomials s in the monomial basis for n variables and
    maximum total degree d.

Input/output variables:

    s: [int] number of monomials in the monomial basis.
    d: [int] maximum total degree of the monomials.
    n: [int] number of variables of the monomials.

See also monomialsmatrix.
```

D.1.3 Quick start

The easiest way to represent a system of multivariate polynomials is by considering a matrix for every polynomial of the system, where the first column corresponds to the coefficients of the polynomials and the remaining columns represent the powers of the variables in the corresponding monomials. These matrices are combined in a cell array and given to the `systemstruct` constructor.

Code D.3. We start by constructing a system of two bivariate polynomial equations.

```
>> p1 = [2 2 0; -3 0 1; 1 0 0];
>> p2 = [1 2 0; 1 0 2; 16 0 0];
>> eqs = {p1, p2};
>> problem = systemstruct(eqs);
```

Similarly, a rectangular MEP can be represented by a cell array that contains all the coefficient matrices in the correct monomial ordering (including zero matrices). This cell array, together with the total degree and number of variables of the problem, is given to the `mepstruct` constructor.

Code D.4. Similarly, we can construct a linear two-parameter eigenvalue problem. Contrary to `systemstruct`, `mepstruct` requires information about the maximum total degree `dmax` and number of eigenvalue `n`.

```
>> dmax = 1; n = 2;
>> A00 = randn(4,3); A10 = randn(4,3); A01 = randn(4,3);
>> mat = {A00, A10, A01};
>> problem = mepstruct(mat,dmax,n);
```

Solving the problem requires only one additional line of code.

Code D.5. Given the problem, it is possible to obtain its solutions without any additional information.

```
>> solutions = macaulaylab(problem);
```

D.1.4 Tests to check all functionality

MacaulayLab contains a set of tests to check the different functions. This allows the user to change the code and experiment with the different functions, but at

all times the user can check whether everything still works. You can find the test suite in the folder `Tests`. In order to run all the tests, simply use Matlab's function `runtests`.

Code D.6. After moving to the correct folder, you can run all the tests:

```
>> cd Tests/
>> results = runtests(`outputdetail',0);
```

D.2 Representation of a problem

In order to keep the toolbox user-friendly, describing a problem is kept very simple. MacaulayLab revolves around two different types of problems: systems of multivariate polynomial equations and rectangular MEPs. Both problem types are internally represented by the same class `problemstruct`: all necessary information is stored in the cell arrays `coef` and `supp`, where each cell of `coef` and `supp` contains the coefficients/coefficient matrices and support of one polynomial (matrix) equation, respectively. Although it is possible to submit the problem directly in its internal representation, the sub-classes `systemstruct` and `mepstruct` provide constructors to set-up the specific problems more easily (Figure D.1).

D.2.1 Systems of multivariate polynomial equations

The natural way of representing a single polynomial is via a row vector with its coefficients. The coefficients in that row vector are ordered according to a particular monomial ordering. For example, the polynomial $p(x, y, z) = 1x^2 + 2yz + 3$ as a row vector corresponds to

$$\begin{bmatrix} 3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 \end{bmatrix}, \tag{D.1}$$

in the graded inverse lexicographic (GRINVLEX) ordering. Of course, this representation contains many zeros when the polynomial is sparse, especially for high degrees and many variables. A more efficient approach to represent a polynomial is by considering a matrix, where the first column corresponds to the coefficients of the polynomial and the remaining columns represent the powers of the variables in the corresponding monomials, i.e.,

$$p_i(\boldsymbol{x}) \leftrightarrow \begin{bmatrix} c_1 & \alpha_{11} & \cdots & \alpha_{1n} \\ c_2 & \alpha_{21} & \cdots & \alpha_{2n} \\ \vdots & \vdots & & \vdots \\ c_N & \alpha_{N1} & \cdots & \alpha_{Nn} \end{bmatrix}, \tag{D.2}$$

with c_i the coefficients of the polynomial and α_{ij} the power of the variable x_j for that ith coefficient. The polynomial $p(x, y, z) = 1x^2+2yz+3$ is represented by a

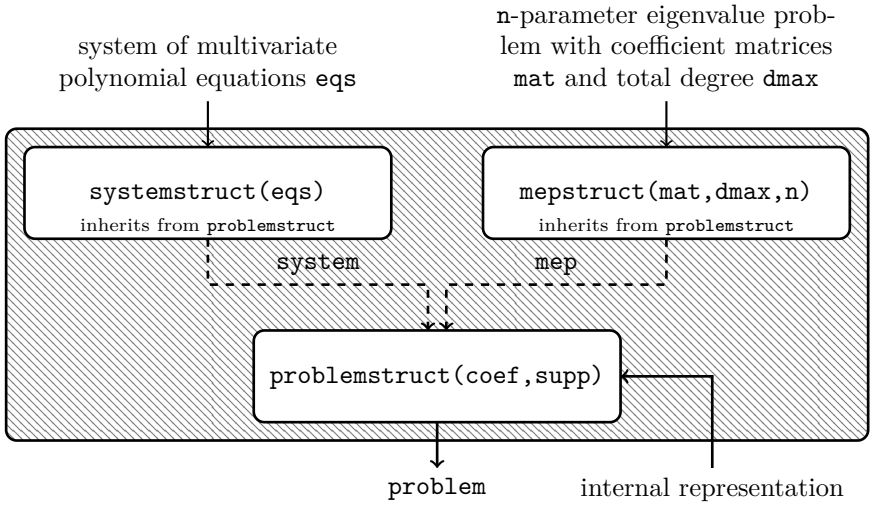


Figure D.1. Representation of a system of multivariate polynomial equations or rectangular MEP in MacaulayLab. Both problem types are internally represented by the same `problemstruct`: all necessary information is stored in the cells `coef` and `supp`. The sub-classes `systemstruct` and `mepstruct` provide constructors to set-up the problems more easily, but it is also possible to submit the problem directly in the internal representation.

matrix with three rows and four columns (three variables and the coefficients):

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 2 & 0 & 1 & 1 \\ 3 & 0 & 0 & 0 \end{bmatrix}. \tag{D.3}$$

When the monomial ordering is set, it is possible to switch between these two representations

Code D.7. A polynomial in its matrix representation can be expanded into a row vector via `expandedpoly(poly)`

```
>> p = [1 2 0 0; 2 0 1 1; 3 0 0 0];
>> pexpanded = expandedpoly(p)

pexpanded =
    3     0     0     0     1     0     0     0     2     0
```

The function `contractedpoly(poly,d,n)` goes in the other direction:

```
>> contractedpoly(pexpanded,3,2)

ans =
    1     2     0     0
    2     0     1     1
    3     0     0     0
```

A system of (multivariate) polynomials is represented internally by two cell arrays `coef` and `supp`. However, you do not need to worry about this internal representation. Via a cell array and the `systemstruct` constructor, multiple polynomials can be combined into that problem representation.

Code D.8. By combining different polynomials in `systemstruct`, a system can be constructed in MacaulayLab.

```
>> p1 = [2 2 0 0; -3 0 1 3; 1 0 0 0];
>> p2 = [1 2 0 0; 1 0 2 0; 1 0 0 2; 16 0 0 0];
>> p3 = [1 1 1 1; 2 0 0 0];
>> eqs = {p1, p2, p3};
>> system = systemstruct(eqs);
```

Suppose that you want to keep the coefficients and support in two separate cell arrays, then you could use a column vector with the coefficients and a matrix with the support per polynomial. You could avoid using the `systemstruct` constructor and submit the system directly using the internal representation of the toolbox. The cell array `coef` contains per equation a cell with a column vector of the coefficients, while each cell of `supp` has a matrix with the corresponding support for these coefficients.

Code D.9. A system can also be constructed directly by giving the internal representation to `problemstruct`.

```
>> coef1 = [2; -3; 1];
>> supp1 = [2 0 0; 0 1 3; 0 0 0];
>> coef2 = [1; 1; 1; 16];
>> supp2 = [2 0 0; 0 2 0; 0 0 2; 0 0 0];
>> coef3 = [1; 2];
>> supp3 = [1 1 1; 0 0 0];
>> coef = {coef1, coef2, coef3};
>> supp = {supp1, supp2, supp3};
>> system = problemstruct(coef,supp);
```

After defining a system, it is possible to retrieve information about the system via the overloaded `disp` and `probleminfo`, or directly via dot indexing.

Code D.10. You can get more information about a problem via `disp`:

```
>> disp(system)

    system of multivariate polynomial equations in the
    standard monomial basis:
        - number of equations = 3
        - number of variables = 3
        - maximum total degree = 4
```

The information can be accessed via `probleminfo` dot indexing:

```
>> [n,s,di,dmax,nnze] = probleminfo(system);
>> eqs = system.coef;
```

In order to create a random dense system, `randomsystem(s,dmax,n)` can be used, which generates a system of `s` multivariate polynomials in `n` variables that has random coefficients for every monomial up to total degree `dmax`.

D.2.2 Rectangular multiparameter eigenvalue problems

Similarly, a rectangular MEP can be represented by a cell array that contains all the coefficient matrices in the correct monomial ordering (including zero matrices). Now, every entry corresponds to one coefficient matrix, ordered in a particular monomial ordering. For example, the linear two-parameter eigenvalue problem $\boldsymbol{\mathcal{M}}(\boldsymbol{\lambda})\boldsymbol{z} = (A_{00} + A_{10}\lambda_1 + A_{01}\lambda_2)\boldsymbol{z} = \mathbf{0}$ has three coefficient matrices. This cell array, together with the total degree and number of variables of the problem, is then given to the `mepstruct` constructor.

Code D.11. Contrary to `systemstruct`, `mepstruct` requires information about the maximum total degree `dmax` and number of eigenvalue `n`.

```
>> dmax = 1; n = 2;
>> A00 = randn(4,3); A10 = randn(4,3); A01 = randn(4,3);
>> mat = {A00, A10, A01};
>> mep = mepstruct(mat,dmax,n);
```

Again, you could decide to enter the problem directly in its internal representation. A single rectangular MEP consists of one cell in `coef` and one cell

in `supp`. The cell in `coef` is a three-dimensional array where the coefficient matrices are stacked along the first dimension, while the cell in `supp` contains a two-dimensional array with the support (each row corresponds to the monomials that belong to that coefficient matrix).

Code D.12. A MEP can also be constructed directly by giving the internal representation to `problemstruct`.

```
>> coef = {mepshape(mat)};
>> supp = {[0 0; 1 0; 0 1]};
>> mep = problemstruct(coef,supp);
```

It is again possible to retrieve information about the MEP via the overloaded `disp` and `probleminfo`, or directly via dot indexing. Quickly constructing a rectangular MEP with random coefficient matrices is also very easy with `randommep(dmax,n,k,l)`, where `dmax` is the maximum total degree, `n` is the number of eigenvalues, `k` is the number of rows, and `l` is the number of columns of the MEP.

D.2.3 Database with test problems

The accompanying database contains many test problems that can be used directly with the functions in MacaulayLab. More information about a system of multivariate polynomials or rectangular MEP can be obtained via `help problem` or `disp(problem)`. Most problems in the database already contain information about the number of affine solutions, total number of solutions, required time to solve the system, etc.

Code D.13. `help(problem)` and `disp(problem)` give more information about the `problem`.

```
>> help noon3

noon3 contains a system of multivariate polynomial
equations.

    [system] = noon3() returns the system of multivariate
    polynomial equations.
```

Some problems have one or multiple additional parameters.

Code D.14. The MEP `arma11` requires a vector as parameter.

```
>> y = randn(10,1);
>> example = arma11(y);
```

D.3 Solutions via the (block) Macaulay matrix

MacaulayLab uses a similar approach to solve both problem types. Many of the functions are, therefore, re-used when solving the different problems. We give a step-by-step solution approach (Appendix D.3.1), but the user is more likely to use the direct solution approach (Appendix D.3.2).

D.3.1 Step-by-step solution approach

Building the (block) Macaulay matrix generated by the problem is probably the first step you take after defining the problem. Since both problems are represented internally by the same data structure, the same function can be used to build the Macaulay matrix for a system of multivariate polynomial equations or block Macaulay matrix for a rectangular MEP. The function `macaulay(problem,d)` builds the (block) Macaulay matrix of degree `d` that incorporates the `problem`.

Code D.15. A Macaulay matrix of degree d can easily be constructed via `macaulay(system,d)`.

```
>> M = macaulay(redeco6,10);
```

With the same function, a block Macaulay matrix of degree d can be constructed: `macaulay(mep,d)`.

```
>> N = macaulay(hkp2,5);
```

The default (block) Macaulay matrix solution approach in this toolbox uses a basis matrix of the right null space the (block) Macaulay matrix. This matrix can be computed directly via the standard approach `null`, via the recursive approach `macaulayupdate` and `nullrecrmacualay`, or via the sparse approach `nullsparsemacaulay`.

Code D.16. The standard approach to compute a basis matrix of the right null space is via `null(Z)`:

```
>> M = macaulay(system,5);
>> Z = null(M);
```

Alternatively, the basis matrix can also be built recursively via `macaulayupdate(M,problem,d)` and `nullrecrmacaulay(Z,Y)`, where `Y` is the difference between the two (block) Macaulay matrices:

```
>> M = macaulay(system,2);
>> Z = null(M);
>> for d = 3:5
       rows = size(M,1);
       M = macaulayupdate(M,system,d);
       Z = nullrecrmacaulay(Z,M(rows+1:end,:));
   end
```

It is also possible to use `nullsparsemacaulay(Z,problem,d)`, which avoids the construction of the (block) Macaulay matrix:

```
>> M = macaulay(system,2);
>> Z = null(M);
>> for d = 3:5
       Z = nullsparsemacaulay(Z,system,d);
   end
```

Once you have a basis matrix of the right null space, you want to look for the standard monomials and determine the gap zone. Of course, you can also determine the gap directly via `gap`.

Code D.17. To determine the standard monomials and the degree of the gap zone:

```
>> c = stdmonomials(Z);
>> [dgap, ma] = gapstdmonomials(c,d,n);
```

Or, directly, via `gap` (which also gives the number of affine solutions):

```
>> [dgap, ma] = gap(Z,d,n);
```

With this information, the column compression to remove the solutions at infinity is quite straightforward:

Code D.18. You can perform a column compression via:

```
>> W11 = columncompression(Z,dgap,n);
```

Solving the problem can be done via performing shifts in the right null space. `shiftnullspace` constructs multiple shift problems in the right null space that yield the solutions of the problem.

Code D.19. The function `shiftnullspace(W,shift,rows,l)` considers $n+1$ shift problems in the basis matrix `W`. `rows` indicates the rows that are shifted and `l` is the length of the eigenvector.

```
>> D = shiftnullspace(W,shift,rows,l);
```

When `rows = NaN`, all degree blocks up to the gap zone are shifted.

D.3.2 Direct solution approach

Of course, there is a solver implemented that takes care of all these intermediate steps and checks whether the right null space of the (block) Macaulay matrix can accommodate the shift polynomial. Furthermore, it is also possible to consider the column space instead of the right null space of the (block) Macaulay matrix. You can use `macaulaylab(problem)` to solve a problem via the default approach (which is currently a block-wise sparse null space based approach).

Code D.20. Two examples of using `macaulaylabproblem`:

```
>> commonroots = macaulaylab(redeco6);
>> eigenvalues = macaulaylab(hkp2);
```

To avoid unpleasant surprises, it is recommended to set a maximum degree for the (block) Macaulay matrix.

Code D.21. By using `macaulaylab(problem,dend)`, the maximum degree of the (block) Macaulay matrix is set to `dend = 20`.

```
>> roots = macaulaylab(redeco6,20);
```

The solver has many options and outputs a lot of information. Table D.1 contains an overview of the different options, but the default options should result in an acceptable trade-off between computation speed and numerical stability. The options can be set via supplementing a `struct` to the solver. For example, asking the solver for a verbose output of its execution can be achieved by setting `options.verbose = true`.

Code D.22. A structure can be used to set the options of the algorithms.

```
>> options = struct;
>> options.verbose = true;
```

These options are added to `macaulaylab(problem,dend,options)`:

```
>> [X, output] = macaulaylab(noon5,15,options);

                        MacaulayLab
-------------------------------------------------------------
The system has 5 equations in 5 variables (maximum total
degree is 3). The selected polynomial basis is the standard
monomial basis.

The solvers tackles the system via the null space of the
Macaulay matrix:
    * Building a basis matrix of the null space (sparse -
    block row-wise)
        ⋮
The solver results in 233 affine solution candidates in
3.5888 seconds.
        max. abs. res. error before clustering = 8.8622e-11
        max. abs. res. error after clustering = 8.8622e-11
        cluster tolerance = 1.0000e-03
        rank tolerance = 1.0000e-10
-------------------------------------------------------------
```

The output `struct` contains a lot of additional information:

```
>> output

output =
    struct with fields:
        maxresidual: 8.8622e-11
            ⋮
        shiftvalues: [233x1 double]
```

Table D.1. Overview of the different options that can be used by the MacaulayLab solver. Appendix D.4 gives more information about the options to set the monomial ordering and polynomial basis. For the full list of available options and possible inputs, which change when other techniques become available, we recommend to consult the documentation of `macaulaylab`.

option	type	default	explanation
`subspace`	`string`	`` `null' ``	choice of subspace
`blocked`	`boolean`	`true`	blocked version of the algorithm
`algorithm`	`string`	`` `sparse' ``	algorithm for the subspace
`tol`	`double`	`10e-10`	tolerance for rank checks
`clustertol`	`double`	`10e-10`	tolerance for clustering
`cluster`	`boolean`	`true`	flag for clustering
`posdim`	`boolean`	`true`	flag for positive-dimensional solution sets
`shiftpoly`	`double(m,n+1)`	`[randn(n,1) eye(n)]`	shift polynomial
`basis`	`function`	`basismon`	definition of the polynomial basis
`position`	`function`	`posgrinvlex`	definition of the monomial ordering

D.4 Monomial ordering and polynomial basis

MacaulayLab is implemented independently from the monomial ordering and polynomial basis, which means that you can supply a certain monomial ordering or polynomial basis and use all the functions without any adaptations. Choosing a certain monomial ordering is done by giving a function that determines the position of a monomial to the solver, e.g., `posgrinvlex` and `posgrevlex`. The definition of a basis requires the user to supply (or use) two functions:

- Definition of the shift property, e.g., `basismon` and `basischeb`.
- Evaluation of a problem, e.g., `evalmon` and `evalcheb`.

These functions are given to MacaulayLab as function handles.

Code D.23. It is possible to construct the Macaulay matrix in any polynomial basis or monomial ordering. `basis` and `order` should be two functions that implement the basis multiplication and the position of a monomial in the monomial ordering.

```
>> basis = @basischeb; % Chebyshev polynomial basis
>> order = @ordergrevlex; % grevlex monomial ordering
>> M = macaulay(problem,d,basis,order);
```

The toolbox has some pre-implemented functions for the monomial ordering and polynomial basis. For the former, `posgrinvlex` and `posgrevlex` allow the user to work in the GRINVLEX or graded reverse lexicographic (GREVLEX) ordering. The standard monomial basis (`basismon` and `evalmon`) and Chebyshev basis (`basischeb` and `evalcheb`) are implemented for the latter. The user can also use its own functions and give them to `macaulaylab`:

Code D.24. You can give a different polynomial basis to the solver:

```
>> options.basis = @basisdefinition;
>> options.eval = @evaldefinition;
>> solutions = macaulaylab(problem,d,options);
```

D.5 Other useful functions

When working with systems of multivariate polynomial equations and rectangular MEPs, some other functions might come in handy:

- `nbmonomials(d,n)` gives the number of multivariate monomials in `n` variables up to degree `d`.

- `monomialsmatrix(d,n)` creates a matrix with as its rows all the different monomial vectors in `n` variables up to degree `d`.
- `bezout(system)` determines the Bézout number of a system of multivariate polynomial equations `system`.
- `kushnirenko(system)` determines the Kushnirenko bound on the number of affine solutions a `system` of multivariate polynomial equations.
- `bkk(system)` determines, for a `system` of multivariate polynomial equations, the Bernstein–Khovanskii–Kushnirenko (BKK) bound on the number of affine solutions.
- `hkp(mep)` gives the Hochstenbach–Košir–Plestenjak (HKP) bound on the number of solutions for a rectangular `mep`.

Bibliography

[1] Tülay Adali and Peter J. Schreier. "Optimization and Estimation of Complex-Valued Signals". In: *IEEE Signal Processing Magazine* 31.5 (2014), pp. 112–128.

[2] George B. Adams, Michael F. Griffin, and Gilbert W. Stewart. "Direction-of-Arrival Estimation Using the Rank-Revealing URV Decomposition". In: *Proc. of the 1991 International Conference on Acoustics, Speech, and Signal Processing (ICASSP 91)*. Toronto, Canada, 1991, pp. 1385–1388.

[3] Oscar M. Agudelo, Christof Vermeersch, and Bart De Moor. "Globally Optimal $\mathcal{H}_2$-Norm Model Reduction: A Numerical Linear Algebra Approach". In: *IFAC-PapersOnLine* 54.9 (2021). Part of special issue: 24th International Symposium on Mathematical Theory of Networks and Systems (MTNS), pp. 564–571.

[4] Mian I. Ahmad, Michalis Frangos, and Imad M. Jaimoukha. "Second Order $\mathcal{H}_2$ Optimal Approximation of Linear Dynamical Systems". In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 12767–12771.

[5] Mian I. Ahmad, Imad Jaimoukha, and Michalis Frangos. "$\mathcal{H}_2$ Optimal Model Reduction of Linear Dynamical Systems". In: *Proc. of the 49th IEEE Conference on Decision and Control (CDC)*. Atlanta, GA, USA, 2010, pp. 5368–5371.

[6] Fawwaz F. F. Alsubaie. "$\mathcal{H}_2$ Optimal Model Reduction for Linear Dynamic Systems and the Solution of Multiparameter Matrix Pencil Problems". PhD thesis. London, UK: Imperial College London, 2019.

[7] Branimir Anić, Christopher A Beattie, Serkan Gugercin, and Athanasios C. Antoulas. "Interpolatory Weighted-$\mathcal{H}_2$ Model Reduction". In: *Automatica* 49.5 (2013), pp. 1275–1280.

[8] Athanasios C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Advances in Design and Control. Philadelphia, PA, USA: SIAM, 2005.

[9] Athanasios C. Antoulas. "Introduction". In: *Mathematical System Theory*. Ed. by Athanasios C. Antoulas. Berlin, Germany: Springer, 1991, pp. 1–8.

[10] Athanasios C. Antoulas, Christopher A. Beattie, and Serkan Gugercin. "Interpolatory Model Reduction of Large-Scale Dynamical Systems". In: *Efficient Modeling and Control of Large-Scale Systems*. Ed. by Javad Mohammadpour and Karolos M. Grigoriadis. Boston, MA, USA: Springer, 2010, pp. 3–58.

[11] Frederick V. Atkinson. *Multiparameter Eigenvalue Problems*. Vol. 82. Mathematics in Science and Engineering. New York, NY, USA: Academic Press, 1972.

[12] Frederick V. Atkinson. "Multiparameter Spectral Theory". In: *Bulletin of the American Mathematical Society* 74.1 (1968), pp. 1–27.

[13] Frederick V. Atkinson and Angelo B. Mingarelli. *Multiparameter Eigenvalue Problems: Sturm–Liouville Theory*. Boca Raton, FL, USA: CRC Press, 2010.

[14] Winfried Auzinger and Hans J. Stetter. "An Elimination Algorithm for the Computation of all Zeros of a System of Multivariate Polynomial Equations". In: *Proc. of the International Conference on Numerical Mathematics*. Singapore, 1988, pp. 11–30.

[15] Raymond G. Ayoub. "Paolo Ruffini's Contributions to the Quintic". In: *Archive for History of Exact Sciences* 23.3 (1980), pp. 253–277.

[16] Philip Ball. "Paper Trail Reveals References Go Unread by Citing Authors". In: *Nature* 420 (2002), p. 594.

[17] Magali Bardet, Jean-Charles Faugère, and Salvy Bruno. "On the Complexity of the F_5 Gröbner Basis Algorithm". In: *Journal of Symbolic Computation* 70 (2015), pp. 49–70.

[18] Stephen Barnett. "A Companion Matrix Analogue for Orthogonal Polynomials". In: *Linear Algebra and its Applications* 12.3 (1975), pp. 197–202.

[19] Daniel J. Bates, Jonathan D. Hauenstein, Sommese Andrew J., and Charles W. Wampler II. *Bertini: Software for Numerical Algebraic Geometry*. `https://bertini.nd.edu`. 2013.

[20] Daniel J. Bates, Jonathan D. Hauenstein, Sommese Andrew J., and Charles W. Wampler II. *Numerically Solving Polynomial Systems with Bertini*. Vol. 25. Software, Environments, and Tools. SIAM, 2013.

[21] Kim Batselier. "A Numerical Linear Algebra Framework for Solving Problems with Multivariate Polynomials". PhD thesis. Leuven, Belgium: KU Leuven, 2013.

[22] Kim Batselier, Philippe Dreesen, and Bart De Moor. "A Fast Recursive Orthogonalization Scheme for the Macaulay Matrix". In: *Journal of Computational and Applied Mathematics* 267 (2014), pp. 20–32.

[23] Kim Batselier, Philippe Dreesen, and Bart De Moor. "A Geometrical Approach to Finding Multivairate Approximate LCMs and GCDs". In: *Linear Algebra and its Applications* 438.9 (2013), pp. 3618–3628.

[24] Kim Batselier, Philippe Dreesen, and Bart De Moor. "On the Null Spaces of the Macaulay Matrix". In: *Linear Algebra and its Applications* 460 (2014), pp. 259–289.

[25] Kim Batselier, Philippe Dreesen, and Bart De Moor. "Prediction Error Method Identification is an Eigenvalue Problem". In: *Proc. of the 16th IFAC Symposium on System Identification*. Brussels, Belgium, 2012, pp. 221–226.

[26] Kim Batselier, Philippe Dreesen, and Bart De Moor. "The Canonical Decomposition of $\mathcal{C}_d^n$ and Numerical Gröbner and Border Bases". In: *Siam Journal on Matrix Analysis and Applications* 35.4 (2014), pp. 1242–1264.

[27] Kim Batselier, Philippe Dreesen, and Bart De Moor. "The Geometry of Multivariate Polynomial Division and Elimination". In: *SIAM Journal on Matrix Analysis and Applications* 34.1 (2013), pp. 102–125.

[28] Christopher A. Beattie and Serkan Gugercin. "Model Reduction by Rational Interpolation". In: *Model Reduction and Approximation*. Ed. by Peter Benner, Albert Cohen, Mario Ohlberger, and Karen Willcox. Philadelphia, PA, USA: SIAM, 2017, pp. 297–334.

[29] Larisa Beilina, Evgenii Karchevskii, and Mikhail Karchevskii. *Numerical Linear Algebra: Theory and Applications*. Cham, Switzerland: Springer, 2017.

[30] Matías R. Bender. "Solving Sparse Polynomial Systems Using Gröbner Bases and Resultants". In: *Proc. of the 2022 International Symposium on Symbolic and Algebraic Computation (ISSAC)*. Villeneuve-d'Ascq, France, 2022, pp. 21–30.

[31] Matías R. Bender and Simon Telen. "Toric Eigenvalue Methods for Solving Sparse Polynomial Systems". In: *Mathematics of Computation* 91.337 (2022), pp. 2397–2429.

[32] Matías R. Bender and Simon Telen. *Yet another Eigenvalue Algorithm for Solving Polynomial Systems*. Tech. rep. Berlin, Germany: Technische Universität Berlin, 2021.

[33] David N. Bernstein. "The Number of Roots of a System of Equations". In: *Funktsional'nyi Analiz i ego Prilozheniya* 9.3 (1975). [citation only], pp. 183–185.

[34] Barbara Betti, Marta Panizzut, and Simon Telen. *Solving Equations Using Khovanskii Bases*. Tech. rep. Leipzig, Germany: Max Planck Institute for Mathematics in the Sciences, 2023.

[35] Edward K. Blum and Ai-Fu Chang. "A Numerical Method for the Solution of the Double Eigenvalue Problem". In: *IMA Journal of Applied Mathematics* 22.1 (1978), pp. 29–42.

[36] Edward K. Blum and Andrew R. Curtis. "A Convergent Gradient Method for Matrix Eigenvector-Eigentuple Problems". In: *Numerische Mathematik* 31.3 (1978), pp. 247–263.

[37] Edward K. Blum and Peter B. Geltner. “Numerical Solution of Eigentuple-Eigenvector Problems in Hilbert Spaces by a Gradient Method”. In: *Numerische Mathematik* 31.3 (1978), pp. 231–246.

[38] Jacek Bochnak, Michel Coste, and Marie-Françoise Roy. *Real Algebraic Geometry*. Modern Surveys in Mathematics. Berlin, Germany: Springer, 1998.

[39] Zvonimir Bohte. “Numerical Solution of Some Two-Parameter Eigenvalue Problems”. In: *Anton Kuhelj Memorial Volume*. [citation only]. Ljubljana, Slovenia: Slovenian Academy of Science and Art, 1982, pp. 17–28.

[40] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control*. Revised. Holden-Day Series in Time Series Analysis. Oakland, CA, USA: Holden Day, 1976.

[41] Petter Brändén. “A Generalization of the Heine–Stieltjes Theorem”. In: *Constructive Approximation* 34 (2011), pp. 135–148.

[42] David H. Brandwood. “A Complex Gradient Operator and its Application in Adaptive Array Theory”. In: *IEE Proceedings H (Microwaves, Optics and Antennas)* 130.1 (1983), pp. 11–16.

[43] Franklin H. Branin jr. “Widely Convergent Method of Finding Multiple Solutions of Simultaneous Nonlinear Equations”. In: *IBM Journal of Research and Development* 16.5 (1972), pp. 504–522.

[44] Frédéric Brechenmacher. “Lagrange and the Secular Equation”. In: *Lettera Matematica* 2 (2014), pp. 79–91.

[45] Paul Breiding and Sascha Timme. “HomotopyContinuation.jl: A Package for Homotopy Continuation in Julia”. In: *Proc. of the 6th International Conference on Mathematical Software*. Ed. by James H. Davenport, Manuel Kauers, George Labahn, and Josef Urban. South Bend, IN, USA, 2018, pp. 458–465.

[46] Claude Brezinski, Gérard Meurant, and Michela Redivo-Zaglia. *A Journey through the History of Numerical Linear Algebra*. Philadelphia, PA, USA: SIAM, 2023.

[47] Peter J. Brockwell and Richard A. Davis. *Time Series: Theory and Methods*. 2nd. Springer Series in Statistics. New York, NY, USA: Springer, 1991.

[48] Patrick J. Browne and Brian D. Sleeman. “A Numerical Technique for Multiparameter Eigenvalue Problems”. In: *IMA Journal of Numerical Analysis* 2.4 (1982), pp. 451–457.

[49] Bruno Buchberger. “Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem Nulldimensionalen Polynomideal”. [citation only]. PhD thesis. Innsbruck, Austria: Universitat Innsbruck, 1965.

[50] James R. Bunch and Christopher P. Nielsen. “Updating the Singular Value Decomposition”. In: *Numerische Mathematik* 31.2 (1978), pp. 111–129.

[51] Angelika Bunse-Gerstner, Dorota Kubalińska, Georg Vossen, and Diane Wilczek. "h_2-norm Optimal Model Reduction for Large Scale Discrete Dynamical MIMO Systems". In: *Journal of Computational and Applied Mathematics* 233.5 (2010), pp. 1202–1216.

[52] Laurent Busé, Houssam Khalil, and Bernard Mourrain. "Resultant-Based Methods for Plane Curves Intersection Problems". In: *Proc. of the International Workshop on Computer Algebra in Scientific Computing (CASC)*. Kalamata, Greece, 2005, pp. 75–92.

[53] Çağatay Candan. "Properly Handling Complex Differentiation in Optimization and Approximation Problems". In: *IEEE Signal Processing Magazine* 36.2 (2019), pp. 117–124.

[54] Robert D. Carmichael. "Boundary Value and Expansion Problems: Algebraic Basis of the Theory". In: *American Journal of Mathematics* 43.2 (1921), pp. 69–101.

[55] Robert D. Carmichael. "Boundary Value and Expansion Problems: Formulation of Various Transcedental Problems". In: *American Journal of Mathematics* 43.4 (1921), pp. 232–270.

[56] Robert D. Carmichael. "Boundary Value and Expansion Problems: Oscillation, Comparison and Expansion Theorems". In: *American Journal of Mathematics* 44.2 (1922), pp. 129–152.

[57] Alessandro Castagnotto, Maria Cruz Varona, Lisa Jeschek, and Boris Lohmann. "sss & sssMOR: Analysis and Reduction of Large-Scale Dynamic Systems in MATLAB". In: *Automatisierungstechnik* 65.2 (2017), pp. 134–150.

[58] Tian-ran Chen, Tsung-Lin Lee, and Tien-Yien Li. "Hom4PS-3: A Parallel Numerical Solver for Systems of Polynomial Equations Based on Polyhedral Homotopy Continuation Methods". In: *Proc. of the 4th International Congress on Mathematical Software (ICMS)*. Seoul, South Korea, 2014, pp. 183–190.

[59] ByoungSeon Choi. *ARMA Model Identification*. Springer Series in Statistics. New York, NY, USA: Springer, 1992.

[60] Carsten Conradi, Elisenda Feliu, Maya Mincheva, and Carsten Wiuf. "Identifying Parameter Regions for Multistationarity". In: *PLOS Computational Biology* 13.10 (2017), pp. 1–25.

[61] Robert M. Corless, Patrizia M. Gianni, Barry M. Trager, and Stephen M. Watt. "The Singular Value Decomposition for Polynomial Systems". In: *Proc. of the 1995 International Symposium on Symbolic and Algebraic Computation (ISSAC)*. Montreal, Canada, 1995, pp. 195–207.

[62] Robert M. Corless, Gianni M. Patrizia, and Barry M. Trager. "A Reordered Schur Factorization Method for Zero-Dimensional Polynomial Systems with Multiple Roots". In: *Proc. of the 1997 International Symposium on Symbolic and Algebraic Computation (ISSAC)*. Maui, HI, USA, 1997, pp. 133–140.

[63] David A. Cox. *Applications of Polynomial Systems*. Vol. 134. Regional Conference Series in Mathematics. Providence, RI, USA: Conference Board of the Mathematical Sciences, 2020.

[64] David A. Cox. *Stickelberger and the Eigenvalue Theorem*. Tech. rep. Amherst, MA, USA: Amherst College, 2020.

[65] David A. Cox, John B. Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms*. 4th. Undergraduate Texts in Mathematics. New York, NY, USA: Springer, 2015.

[66] David A. Cox, John B. Little, and Donal O'Shea. *Using Algebraic Geometry*. 2nd. Vol. 185. Graduate Texts in Mathematics. New York, NY, USA: Springer, 2005.

[67] Ruchira S. Datta. "Using Computer Algebra to find Nash Equilibria". In: *Proc. of the 2003 International Symposium on Symbolic and Algebraic Computation (ISSAC)*. Philadelphia, PA, USA, 2003, pp. 74–79.

[68] Barry H. Dayton, Tien-Yien Li, and Zhonggang Zeng. "Multiple Zeros of Nonlinear Systems". In: *Mathematics of Computation* 80.276 (2011), pp. 2143–2168.

[69] Katrien De Cock and Bart De Moor. "Multiparameter Eigenvalue Problems and Shift-Invariance". In: *IFAC-PapersOnLine* 54.9 (2021). Part of special issue: 24th International Symposium on Mathematical Theory of Networks and Systems (MTNS), pp. 159–165.

[70] Bart De Moor. "Least Squares Optimal Realisation of Autonomous LTI Systems Is an Eigenvalue Problem". In: *Communications in Information and Systems* 20.2 (2020), pp. 163–207.

[71] Bart De Moor. "Least Squares Realization of LTI Models Is an Eigenvalue Problem". In: *Proc. of the 18th European Control Conference (ECC)*. Naples, Italy, 2019, pp. 2270–2275.

[72] Bart De Moor. "Structured Total Least Squares and L_2 Approximation Problems". In: *Linear Algebra and its Applications* 188–189 (1993), pp. 163–205.

[73] Bart De Moor. "Total Least Squares for Affinely Structured Matrices and the Noisy Realization Problem". In: *IEEE Transactions on Signal Processing* 42.11 (1994), pp. 3104–3113.

[74] James W. Demmel. *Applied Numerical Linear Algebra*. Philadelphia, PA, USA: SIAM, 1997.

[75] René Descartes. "La Géométrie". In: *Discours de la Méthode pour Bien Conduire sa Raison, et Chercher la Vérité dans les Sciences*. [citation only]. Leiden, The Netherlands: Ian Maire, 1637, pp. 297–413.

[76] Alicia Dickenstein. "Biochemical Reaction Networks: An Invitation for Algebraic Geometers". In: *Contemporary Mathematics* 656 (2016). Part of special issue: 1st Mathematical Congress of the Americas (MCA), pp. 69–97.

[77] Bo Dong, Bo Yu, and Yan Yu. "A Homotopy Method for Finding All Solutions of a Multiparameter Eigenvalue Problem". In: *SIAM Journal on Matrix Analysis and Applications* 37.2 (2016), pp. 550–571.

[78] Philippe Dreesen. "Back to the Roots: Polynomial System Solving Using Linear Algebra". PhD thesis. Leuven, Belgium: KU Leuven, 2013.

[79] Philippe Dreesen, Kim Batselier, and Bart De Moor. "Back to the Roots: Polynomial System Solving, Linear Algebra, Systems Theory". In: *Proc. of the 16th IFAC Symposium on System Identification*. Brussels, Belgium, 2012, pp. 1203–1208.

[80] Philippe Dreesen, Kim Batselier, and Bart De Moor. "Multidimensional Realisation Theory and Polynomial System Solving". In: *International Journal of Control* 91.12 (2018), pp. 2692–2704.

[81] Philippe Dreesen and Bart De Moor. "Polynomial Optimization Problems Are Eigenvalue Problems". In: *Model-Based Control*. Ed. by Paul M. J. Hof, Carsten Scherer, and Peter S. C. Heuberger. Boston, MA, USA: Springer, 2009, pp. 49–68.

[82] David Eisenbud and Joe Harris. *The Geometry of Schemes*. Vol. 197. Graduate Texts in Mathematics. New York, NY, USA: Springer, 2006.

[83] Mohamed Elkadi and Bernard Mourrain. *Introduction à la Résolution des Systèmes Polynomiaux*. Mathématiques et Applications. Berlin, Germany: Springer, 2007.

[84] Howard C. Elman, Karl Meerbergen, Alastair Spence, and Minghao Wu. "Lyapunov Inverse Iteration for Identifying Hopf Bifurcations in Models of Incpressible Flow". In: *SIAM Journal on Scientific Computing* 34.3 (2012), A1584–A1606.

[85] Ioannis Z. Emiris. "Sparse Elimination and Applications in Kinematics". PhD thesis. Berkeley, CA, USA: University of California, 1994.

[86] Ioannis Z. Emiris and Bernard Mourrain. "Computer Algebra Methods for Studying and Computing Molecular Conformations". In: *Algorithmica* 25.2–3 (1999), pp. 372–402.

[87] Ioannis Z. Emiris and Bernard Mourrain. "Matrices in Elimination Theory". In: *Journal of Symbolic Computation* 28.1–2 (1999), pp. 3–44.

[88] Jean-Charles Faugère. "A New Efficient Algorithm for Computing Gröbner Bases (F4)". In: *Journal of Pure and Applied Algebra* 139.1–3 (1999), pp. 61–88.

[89] Jean-Charles Faugère. "A New Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (F5)". In: *Proc. of the 2002 International Symposium on Symbolic and Algebraic Computation (ISSAC)*. Lille, France, 2002, pp. 75–83.

[90] Jean-Charles Faugère, Patrizia Gianni, Daniel Lazard, and Teo Mora. "Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering". In: *Journal of Symbolic Computation* 16.4 (1993), pp. 329–344.

[91] Gene F. Franklin, David J. Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. 8th. New York, NY, USA: Pearson, 2018.

[92] Shuhong Gao, Yinhua Guan, and Frank Volny. "A New Incremental Algorithm for Computing Groebner Bases". In: *Proc. of the 2010 International Symposium on Symbolic and Algebraic Computation (ISSAC10)*. Munich, Germany, 2010, pp. 13–19.

[93] Stephan R. Garcia, Javad Mashreghi, and William T. Ross. *Introduction to Model Spaces and their Operators*. Cambridge Studies in Advanced Mathematics. Cambridge, UK: Cambridge University Press, 2016.

[94] Vladimir P. Gerdt and Amir Hashemi. "On the Use of Buchberger Criteria in G^2V Algorithm for Calculating Gröbner Bases". In: *Programming and Computer Software* 39 (2013), pp. 81–90.

[95] Michel Gevers. "Identification for Control: From the Early Achievements to the Revival of Experiment Design". In: *European Journal of Control* 11.4–5 (2005), pp. 335–352.

[96] Călin I. Gheorghiu, Michiel E. Hochstenbach, Bor Plestenjak, and Joost Rommes. "Spectral Collocation Solutions to Multiparameter Mathieu's system". In: *Applied Mathematics and Computation* 218.24 (2012), pp. 11990–12000.

[97] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. 4th. Baltimore, MD, USA: Johns Hopkins University Press, 2013.

[98] Irving J. Good. "The Colleague Matrix, A Chebyshev Analogue of the Companion Matrix". In: *The Quarterly Journal of Mathematics* 12.1 (1961), pp. 61–68.

[99] Nithin Govindarajan, Raphaël Widdershoven, Shivkumar Chandrasekaran, and Lieven De Lathauwer. *A Fast Algorithm for Computing Macaulay Null Spaces of Bivariate Polynomial Systems*. Tech. rep. Leuven, Belgium: KU Leuven, 2023.

[100] Robert T. Gregory. "Defective and Derogatory Matrices". In: *SIAM Review* 2.2 (1960), pp. 134–139.

[101] Serkan Gugercin, Athanasios C. Antoulas, and Christopher A. Beattie. "$\mathcal{H}_2$ Model Reduction for Large-Scale Linear Dynamical Systems". In: *SIAM Journal on Matrix Analysis and Applications* 30.2 (2008), pp. 609–638.

[102] Serkan Gugercin, Athanasios C. Antoulas, and Christopher A. Beattie. "A Rational Krylov Iteration for Optimal $\mathcal{H}_2$ Model Reduction". In: *Proc. of the 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS)*. Kyoto, Japan, 2006, pp. 1665–1667.

[103] Takayuki Gunji, Sunyoung Kim, Katsuki Fujisawa, and Masakuzu Kojima. "PHoMpara – Parallel Implementation of the Polyhedral Homotopy Continuation Method for Polynomials Systems". In: *Computing* 77 (2006), pp. 387–411.

[104] Takayuki Gunji, Sunyoung Kim, Masakuzu Kojima, Akiko Takeda, Katsuki Fujisawa, and Tomohiko Mizutani. "PHoM – a Polyhedral Homotopy Continuation Method for Polynomial Systems". In: *Computing* 73 (2004), pp. 57–77.

[105] Paul R. Halmos. *A Hilbert Space Problem Book*. 1st. Princeton, NJ, USA: Van Nostrand, 1967.

[106] Bernard Hanzon and Dorina Jibetean. "Global Minimization of a Multivariate Polynomial Using Matrix Methods". In: *Journal of Global Optimization* 27.1 (2003), pp. 1–23.

[107] Bernard Hanzon, Jan M. Maciejowski, and Chun Tung Chou. *Model Reduction in H2 Using Matrix Solutions of Polynomial Equations*. Tech. rep. `http://www-control.eng.cam.ac.uk/Homepage/papers/cued_control_1142.pdf`. Cambridge, UK: Cambridge University, 1998.

[108] Bernard Hanzon, Jan M. Maciejowski, and Chun Tung Chou. *Optimal H2 Order-One Reduction by Solving Eigenproblems for Polynomial Equations*. Tech. rep. `https://arxiv.org/pdf/0706.1862.pdf`. Cambridge, UK: Cambridge University, 2007.

[109] Wenrui Hao, Andrew J. Sommese, and Zhonggang Zeng. "Algorithm 931: An Algorithm and Software for Computing Multiplicity Structures at Zeros of Nonlinear Systems". In: *ACM Transactions on Mathematical Software* 40.1 (2013), 5:1–5:16.

[110] Behnam Hashemi, Yuji Nakatsukasa, and Llyod N. Trefethen. "Rectangular Eigenvalue Problems". In: *Advances in Computational Mathematics* 48.80 (2022), pp. 1–16.

[111] Didier Henrion, Milan Korda, and Jean-Bernard Lasserre. *The Moment-SOS Hierarchy*. Lectures in Probability, Statistics, Computational Geometry, Control and Nonlinear PDEs. London, UK: World Scientific, 2020.

[112] Didier Henrion and Jean-Bernard Lasserre. "Detecting Global Optimality and Extracting Solutions in GloptiPoly". In: *Positive Polynomials in Control*. Ed. by Didier Henrion and A. Garulli. Vol. 312. Lecture Notes in Control and Information Science. Berlin, Germany: Springer, 2005, pp. 293–310.

[113] Nicholas J. Higham, Steven D. Mackey, and Françoise Tisseur. "The Conditioning of Linearizations of Matrix Polynomials". In: *SIAM Journal of Matrix Analysis and Applications* 28.4 (2006), pp. 1005–1028.

[114] David Hilbert. "Grundzüge einer Allgemeinen Theorie der Linearen Integralgleichungen (Erste Mitteilung)". In: *Nachrichten von der Königlichen Gesellschaft der Wissenschaften zu Göttingen* 1904 (1904). [citation only], pp. 49–91.

[115] Heisuke Hironaka. "Resolution of Singularities of an Algebraic Variety over a Field of Characteristic Zero: I". In: *Annals of Mathematics* 79.1 (1964), pp. 109–203.

[116] Bin-Lun Ho and Rudolf E. Kalman. “Effective Construction of Linear State-Variable Models from Input/Output Functions”. In: *Reglungstechnik* 14.12 (1966), pp. 545–548.

[117] Michiel E. Hochstenbach, Tomaž Košir, and Bor Plestenjak. “A Jacobi–Davidson Type Method for the Two-Parameter Eigenvalue Problem”. In: *SIAM Journal on Matrix Analysis and Applications* 26.2 (2004), pp. 477–497.

[118] Michiel E. Hochstenbach, Tomaž Košir, and Bor Plestenjak. “Numerical Methods for Rectangular Multiparameter eigenvalue Problems, with Applications to Finding Optimal ARMA and LTI Models”. In: *Numerical Linear Algebra with Applications* (2023), e2540.

[119] Michiel E. Hochstenbach, Karl Meerbergen, Emre Mengi, and Bor Plestenjak. “Subspace Methods for Three-Parameter Eigenvalue Problems”. In: *Numerical Linear Algebra with Applications* 26.4 (2019), pp. 1–22.

[120] Michiel E. Hochstenbach, Andrej Muhič, and Bor Plestenjak. “On Linearizations of the Quadratic Two-Parameter Eigenvalue Problem”. In: *Linear Algebra and its Applications* 436 (2012), pp. 2725–2743.

[121] Michiel E. Hochstenbach and Bor Plestenjak. “A Jacobi–Davidson Type Method for a Right Definite Two-Parameter Eigenvalue Problem”. In: *SIAM Journal on Matrix Analysis and Applications* 24.2 (2002), pp. 392–410.

[122] Michiel E. Hochstenbach and Bor Plestenjak. “Harmonic Rayleigh–Ritz Extraction for the Multiparameter Eigenvalue Problem”. In: *Electronic Transactions on Numerical Analysis* 29 (2008), pp. 81–96.

[123] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. 2nd. Cambridge, UK: Cambridge University Press, 2012.

[124] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1994.

[125] Birkett Huber and Bernd Sturmfels. “Bernstein’s Theorem in Affine Space”. In: *Discrete & Computational Geometry* 17.2 (1997), pp. 137–141.

[126] Xingzhi Ji. “Numerical Solution of Joint Eigenpairs of a Family of Commutative Matrices”. In: *Applied Mathematics Letters* 4.3 (1991), pp. 57–60.

[127] Gułdbjörn F. Jónsson and Stephen A. Vavasis. “Accurate Solution of Polynomial Equations using Macaulay Resultant Matrices”. In: *Mathematics of Computation* 74.249 (2004), pp. 221–262.

[128] Thomas Kailath. *Linear Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1980.

[129] Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear Estimation*. Prentice Hall Information and System Sciences Series. Upper Saddle River, NJ, USA: Prentice Hall, 2000.

[130] Rudolf E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45.

[131] Shigetoshi Katsura, Wataru Fukuda, Sakari Inawashiro, Nahomi M. Fujiki, and Rüdiger Gebauer. “Distribution of Effective Field in the Ising Spin Glass of the $\pm J$ Model at $T = 0$: Solutions as Superpositions of Δ-Functions”. In: *Cell Biophysics* 11 (1987), pp. 309–319.

[132] Vladimir B. Khazanov. “Generating Eigenvectors of a Multiparameter Polynomial Matrix”. In: *Journal of Mathematical Sciences* 101.4 (2000), pp. 3326–3337.

[133] Vladimir B. Khazanov. “Methods for Solving Spectral Problems for Multiparameter Matrix Pencils”. In: *Journal of Mathematical Sciences* 127 (2005), pp. 2033–2050.

[134] Vladimir B. Khazanov. “On Spectral Properties of Multiparameter Polynomial Matrices”. In: *Journal of Mathematical Sciences* 89.6 (1998), pp. 1775–1800.

[135] Vladimir B. Khazanov. “Spectral Properties of λ-Matrices”. In: *Journal of Soviet Mathematics* 24 (1984), pp. 121–132.

[136] Vladimir B. Khazanov. “The Multiparameter Eigenvalue Problem: Jordan Vector Semilattices”. In: *Journal of Mathematical Sciences* 86.4 (1997), pp. 2944–2949.

[137] Vladimir B. Khazanov. “To Solving Spectral Problems for Multiparameter Polynomial Matrices”. In: *Journal of Mathematical Sciences* 141.6 (2007), pp. 1690–1700.

[138] Aleksey Kondratyev. “Numerical Computation of Groebner Bases”. [citation only]. PhD thesis. Linz, Austria: Johannes Kepler University, 2003.

[139] Tomaž Košir and Bor Plestenjak. “On the Singular Two-Parameter Eigenvalue Problem II”. In: *Linear Algebra and its Applications* 649 (2022), pp. 433–451.

[140] David M. Kreps. “Game Theory”. In: ed. by John Eatwell, Murray Milgate, and Peter Newman. London, UK: The Macmillan Press Limited, 1989. Chap. Nash Equilibrium, pp. 167–177.

[141] Ken Kreutz-Delgado. *The Complex Gradient Operator and the $\mathbb{CR}$-Calculus*. Lecture notes. San Diego, CA, USA: University of California, 2009.

[142] Gopi T. Krishna, Istdeo Singh, and Krishnamurthy Giridhar. “Null-Space of Block Convolution Matrix”. In: *Proc. of the 2013 National Conference on Communications (NCC)*. New Delhi, India, 2013, pp. 1–5.

[143] Shira Kritchman and Boaz Nadler. “Non-Parametric Detection of the Number of Signals: Hypothesis Testing and Random Matrix Theory”. In: *IEEE Transactions on Signal Processing* 57.10 (2009), pp. 3930–3941.

[144] Harold W. Kuhn, John C. Harsanyi, Reinhard Selten, Jörgen W. Weibull, Eric van Damme, John F. Nash jr., and Peter Hammerstein. "The Work of John Nash in Game Theory". In: *Journal of Economic Theory* 69.1 (1996), pp. 153–185.

[145] Zuzana Kúkelová. "Algebraic Methods in Computer Vision". PhD thesis. Prague, Czech Republic: Czech Technical University, 2013.

[146] Zuzana Kúkelová, Martin Bujnak, and Tomáš Pajdla. "Automatic Generator of Minimal Problem Solvers". In: *Proc. of the 10th European Conference on Computer Vision (ECCV)*. Marseille, France, 2008, pp. 302–315.

[147] Anatoliĭ G. Kushnirenko. "Newton Polytopes and the Bézout Theorem". In: *Funktsional'nyi Analiz i ego Prilozheniya* 10.3 (1976). [citation only], pp. 233–235.

[148] Sibren Lagauw, Oscar M. Agudelo, and Bart De Moor. *Globally Optimal SISO H_2-Norm Model Reduction Using Walsh's Theorem*. Tech. rep. Leuven, Belgium: KU Leuven, 2023.

[149] Peter Lancaster and Ion Zaballa. "Diagonalizable Quadratic Eigenvalue Problems". In: *Mechanical Systems and Signal Processing* 23.4 (2009), pp. 1134–1144.

[150] Jean-Bernard Lasserre. *An Introduction to Polynomial and Semi-Algebraic Optimization*. Vol. 52. Cambridge Texts in Applied Mathematics. Cambridge, UK: Cambridge University Press, 2015.

[151] Monique Laurent. "Sums of Squares, Moment Matrices and Optimization Over Polynomials". In: *Emerging Applications of Algebraic Geometry*. Ed. by Mihai Putinar and Seth Sullivant. Vol. 149. The IMA Volumes in mathematics and its Applications. New York, NY, USA: Springer, 2008.

[152] Oliver Lauwers, Christof Vermeersch, and Bart De Moor. "Cepstral Identification of Autoregressive Systems". In: *Automatica* 139 (2022), 110214:1–14.

[153] Daniel Lazard. "Gröbner Bases, Gaussian Elimination and Resolution of Systems of Algebraic Equations". In: *Proc. of the 1983 European Conference on Computer Algebra (EUROCAL)*. London, UK, 1983, pp. 146–156.

[154] Daniel Lazard. "Résolution des Systèmes d' Équations Algébriques". In: *Theoretical Computer Science* 15 (1981). [citation only], pp. 77–110.

[155] Philippe Lemmerling and Bart De Moor. "Misfit Versus Latency". In: *Automatica* 37.12 (2001), pp. 2057–2067.

[156] Anton Leykin. "Numerical Algebraic Geometry". In: *Journal of Software for Algebra and Geometry* 3 (2011), pp. 5–10.

[157] Tien-Yien Li. "Numerical Solution of Multivariate Polynomial Systems by Homotopy Continuation Methods". In: *Acta Numerica* 6 (1997), pp. 399–436.

[158] Lennart Ljung. *System Identification: Theory for the User*. 2nd. Prentice Hall Information and System Science Series. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[159] Francis S. Macaulay. "Some Formulae in Elimination". In: *Proc. of the London Mathematical Society* 1.1 (1902), pp. 3–27.

[160] Francis S. Macaulay. *The Algebraic Theory of Modular Systems*. Vol. 19. Cambridge Tracts in Mathematics and Mathematical Physics. London, UK: Cambridge University Press, 1916.

[161] Steven D. Mackey, Niloufer Mackey, Christian Mehl, and Volker Mehrmann. "Structured Polynomial Eigenvalue Problems: Good Vibrations From Good Linearizations". In: *SIAM Journal on Matrix Analysis and Applications* 28.4 (2006), pp. 1029–1051.

[162] Steven D. Mackey, Niloufer Mackey, and Françoise Tisseur. "Polynomial Eigenvalue Problems: Theory, Computation, and Structure". In: *Numerical Algebra, Matrix Theory, Differential-Algebraic Equations and Control Theory*. Ed. by Peter Benner, Matthias Bollhöfer, Daniel Kressner, Christian Mehl, and Tatjana Stykel. Cham, Switzerland: Springer, 2015, pp. 319–348.

[163] Dinesh Manocha. "MultiPolynomial Resultant Algorithms". In: *Journal on Symbolic Computation* 15 (1993), pp. 99–122.

[164] Dinesh Manocha. "Solving Systems of Polynomial Equations". In: *IEEE Computer Graphics and Applications* 14.2 (1994), pp. 46–55.

[165] Dinesh Manocha and James W. Demmel. "Algorithms for Intersecting Parameteric and Algebraic Curves II; Multiple Intersections". In: *Graphical Models and Image Processing* 57.2 (1995), pp. 81–100.

[166] Ivan Markovsky, Jan C. Willems, Sabine Van Huffel, and Bart De Moor. *Exact and Approximate Modeling of Linear Systems: A Behavioral Approach*. Mathematical Modeling and Computation. Philadelphia, PA, USA: SIAM, 2006.

[167] Nicola Mastronardi, Marc Van Barel, and Raf Vandebril. "On the Computation of the Null Space of Toeplitz-like Matrices". In: *Electronic Transactions on Numerical Analysis* 33 (2009), pp. 151–162.

[168] Timothy W. McKeithan. "Kinetic Proofreading in T-cell Receptor Signal Transduction". In: *Proc. of the National Academy of Sciences* 92.11 (1995), pp. 5042–5046.

[169] Karl Meerbergen and Bor Plestenjak. "A Sylvester–Arnoldi Type Method for the Generalized Eigenvalue Problem with Two-by-Two Operator Derminants". In: *Numerical Linear Algebra with Applications* 432 (2015), pp. 2529–2542.

[170] Karl Meerbergen and Alastair Spence. "Shift-and-Invert Iteration for Purely Imaginary Eigenvalues with Application to the Detection of Hopf Bifurcations in Large Scale Problems". In: *SIAM Journal on Matrix Analysis and Applications* 31.4 (2010), pp. 11990–120000.

[171] Lewis Meier and David G. Luenberger. “Approximation of Linear Constant Systems”. In: *IEEE Transactions on Automatic Control* 12.5 (1967), pp. 585–588.

[172] William Miller. “Death of a Genius: His Fourth Dimension, Time, Overtakes Einstein”. In: *LIFE Magazine* 38.18 (1955), pp. 61–64.

[173] Cleve B. Moler. *Four Fundamental Subspaces of Linear Algebra*. `https://blogs.mathworks.com/cleve/2016/11/28/four-fundamental-subspaces-of-linear-algebra/`. 2016.

[174] Cleve B. Moler and Gilbert W. Stewart. “An Algorithm for Generalized Matrix Eigenvalue Problems”. In: *SIAM Journal on Numerical Analysis* 10.2 (1973), pp. 241–256.

[175] Michael H. Möller and Ferdinando Mora. “Upper and Lower Bounds for the Degree of Groebner Bases”. In: *Proc. of the 3rd International Symposium on Symbolic and Algebraic Manipulation (EUROSAM)*. Cambridge, UK, 1984, pp. 172–183.

[176] Michael H. Möller and Hans J. Stetter. “Multivariate Polynomial Equations with Multiple Zeros Solved by Matrix Eigenproblems”. In: *Numerische Mathematik* 70.3 (1995), pp. 311–329.

[177] Jamil Momin and Xin-She Yang. “A Literature Survey of Benchmark Functions for Global Optimisation Problems”. In: *International Journal of Mathematical Modelling and Numerical Optimisation* 4.2 (2013), pp. 150–194.

[178] Marc Moonen, Bart De Moor, Lieven Vandenberghe, and Joos Vandewalle. “On- and Off-line Identification of Linear State Space Models”. In: *International Journal of Control* 49.1 (1989), pp. 219–232.

[179] Marc Moonen, Paul M. Van Dooren, and Joos Vandewalle. “A Singular Value Decomposition Updating Algorithm for Subspace Tracking”. In: *SIAM Journal on Matrix Analysis and Applications* 13.4 (1992), pp. 1015–1038.

[180] Alexander P. Morgan. *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1987.

[181] Bernard Mourrain. “A New Criterion for Normal Form Algorithms”. In: *Proc. of the 13th International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes (AAECC99)*. Honolulu, HI, USA, 1999, pp. 430–442.

[182] Bernard Mourrain. “Computing the Isolated Roots by Matrix Methods”. In: *Journal of Symbolic Computation* 26 (1998), pp. 715–738.

[183] Bernard Mourrain and Victor Y. Pan. “Multivariate Polynomials, Duality, and Structured Matrices”. In: *Journal of Complexity* 16.1 (2000), pp. 110–180.

[184] Bernard Mourrain and Jean-Pascal Pavone. “Subdivision Methods for Solving Polynomial Equations”. In: *Journal of Symbolic Computation* 44.3 (2009), pp. 292–306.

[185] Bernard Mourrain, Simon Telen, and Marc Van Barel. “Truncated Normal Forms for Solving Polynomial Systems: Generalized and Efficient Algorithms”. In: *Journal of Symbolic Computation* 102 (2021), pp. 63–85.

[186] Andrej Muhič and Bor Plestenjak. “On the Quadratic Two-Parameter Eigenvalue Problem and its Linearization”. In: *Linear Algebra and its Applications* 432 (2010), pp. 2529–2542.

[187] Andrej Muhič and Bor Plestenjak. “On the Singular Two-Parameter Eigenvalue Problem”. In: *The Electronic Journal of Linear Algebra* 18 (2009), pp. 420–437.

[188] Yuki Nakatsukasa, Vanni Noferini, and Alex Townsend. “Computing the common zeros of two bivariate functions via Bézout resultants”. In: *Numerische Mathematik* 129.1 (2015), pp. 181–209.

[189] Jiawang Nie, James W. Demmel, and Bernd Sturmfels. “Minimizing Polynomials via Sum of Squares Over the Gradient Ideal”. In: *Mathematical Programming* 106 (2006), pp. 587–606.

[190] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. 2nd. Springer Series in Operations Research and Financial Engineering. New York, NY, USA: Springer, 2006.

[191] Vanni Noferini and Alex Townsend. “Numerical Instability of Resultant Methods for Multidimensional Rootfinding”. In: *SIAM Journal on Numerical Analysis* 54.2 (2016), pp. 719–743.

[192] Katsuhiko Ogata. *Modern Control Engineering*. 5th. Upper Saddle River, NJ, USA: Pearson, 2010.

[193] Suely Oliveira and David E. Stewart. *Writing Scientific Software*. Cambridge, UK: Cambridge University Press, 2006.

[194] Victor Y. Pan. “Solving a Polynomial Equation: Some History and Recent Progress”. In: *SIAM Review* 39.2 (1997), pp. 187–220.

[195] Patrick O. Perry and Patrick J. Wolfe. “Minimax Rank Estimation for Subspace Tracking”. In: *IEEE Journal of Selected Topics in Signal Processing* 4.3 (2010), pp. 504–513.

[196] Bor Plestenjak. “A Continuation Method for a Right Definite Two-Parameter Eigenvalue Problem”. In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1163–1184.

[197] Bor Plestenjak. “Minimal Determinantal Representations of Bivariate Polynomials”. In: *Linear Algebra and its Applications* 532 (2017), pp. 550–569.

[198] Bor Plestenjak. *MultiParEig: Toolbox for Multiparameter Eigenvalue Problems*. `https://mathworks.com/matlabcentral/fileexchange/47844-multipareig`. (Visited on July 8, 2023). 2023.

[199] Bor Plestenjak. “Numerical Methods for Nonlinear Two-Parameter Eigenvalue Problems”. In: *BIT Numerical Mathematics* 56 (2016), pp. 241–262.

[200] Bor Plestenjak, Călin I. Gheorghiu, and Michiel E. Hochstenbach. "Spectral Collocation for Multiparameter Eigenvalue Problems Arising from Separable Boundary Value Problems". In: *Journal of Computational Physics* 298 (2015), pp. 585–601.

[201] Bor Plestenjak and Michiel E. Hochstenbach. "Roots of Bivariate Polynomial Systems via Determinantal Representations". In: *SIAM Journal on Scientific Computing* 38.2 (2016), pp. 765–788.

[202] Henri Poincaré. "Sur les Propriétés du Potentiel et sur les Fonctions Abéliennes". In: *Acta Mathematica* 22.1 (1899). [citation only], pp. 89–178.

[203] Arion Pons and Stefanie Gutschmidt. "Multiparameter Solution Methods for Semistructured Aeroelastic Flutter Problems". In: *AIAA Journal* 55.10 (2017), pp. 3530–3538.

[204] Greg Reid, Jianliang Tang, and Lihong Zhi. "A Complete Symbolic-Numeric Linear Method for Camera Pose Determination". In: *Proc. of the 2003 International Symposium on Symbolic and Algebraic Computation (ISSAC)*. Philadelphia, PA, USA, 2003, pp. 215–223.

[205] Miles Reid. *Undergraduate Commutative Algebra*. Vol. 29. London Mathematical Society Student Texts. Cambridge, UK: Cambridge University Press, 1995.

[206] Jose I. Rodriguez, Jin-Hong Du, Yiling You, and Lek-Heng Lim. "Fiber Product Homotopy Method for Multiparameter Eigenvalue Problems". In: *Numerische Mathematik* 148.4 (2021), pp. 853–888.

[207] Steven Roman. *Advanced Linear Algebra*. 3rd. Vol. 135. Graduate Texts in Mathematics. New York, NY, USA: Springer, 2008.

[208] Joost Rommes. "Modal Approximation and Computation of Dominant Poles". In: *Model Order Reduction: Theory, Research Aspects and Applications*. Ed. by Wilhelmus H. A. Schilders, Henk A. van der Vorst, and Joost Rommes. Berlin, Germany: Springer, 2008, pp. 177–193.

[209] Michael I. Rosen. "Niels Hendrik Abel and Equations of the Fifth Degree". In: *The American Mathematical Monthly* 102.6 (1995), pp. 495–505.

[210] Mohammed M. Rounaghi and Farzaneh N. Zadeh. "Investigation of Market Efficiency and Financial Stability between S&P 500 and London Stock Exchange: Monthly and Yearly Forecasting of Time Series Stock Returns Using ARMA Model". In: *Physica A: Statistical Mechanics and its Applications* 456 (2016), pp. 10–21.

[211] Gary Rubinstein. *Descartes' Method for Constructing Roots of Polynomials with 'Simple' Curves*. `http://www.maa.org/press/periodicals/convergence/descartes-method-for-constructing-roots-of-polynomials-with-simple-curves`. 2016.

[212] Koen Ruymbeek, Karl Meerbergen, and Wim Michiels. “Subspace Method for Multiparameter-Eigenvalue Problems based on Tensor-Train Representations”. In: *Numerical Linear Algebra with Applications* 29.5 (2022), 1–20 (e2439).

[213] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. Revised. Vol. 66. Classics In Applied Mathematics. Philadelphia, PA, USA: SIAM, 2011.

[214] Massimiliano Sala, Teo Mora, Ludovic Perret, Shojiro Sakata, and Carlo Traverso, eds. *Gröbner Bases, Coding, and Cryptography*. Berlin, Germany: Springer, 2009.

[215] Boris Shapiro. “Algebro-Geometric Aspects of Heine–Stieltjes Theory”. In: *Journal of the London Mathematical Society* 83.1 (2010), pp. 36–56.

[216] Boris Shapiro and Michael Shapiro. “On Eigenvalues of Rectangular Matrices”. In: *Proc. of the Steklov Institute of Mathematics* 267.1 (2009), pp. 248–255.

[217] Brian D. Sleeman. “Multiparameter Spectral Theory and Separation of Variables”. In: *Journal of Physics A: Mathematical and Theoretical* 41.1 (2007), pp. 1–20.

[218] Tomaž Slivnik and Gabrijel Tomšič. “A Numerical Method for the Solution of Two-Parameter Eigenvalue Problems”. In: *Journal of Computational and Applied Mathematics* 15.1 (1986), pp. 109–115.

[219] Eugen Slutzky. “The Summation of Random Causes as the Source of Cyclic Processes”. In: *Econometrica* 5.2 (1937), pp. 105–146.

[220] Steve Smale. “The Fundamental Theorem of Algebra and Complexity Theory”. In: *Bulletin of the American Mathematical Society* 4.1 (1981), pp. 1–36.

[221] Andrew J. Sommese, Jan Verschelde, and Charles W. Wampler II. “Numerical Decomposition of the Solution Sets of Polynomial Systems into Irreducible Components”. In: *SIAM Journal on Numerical Analysis* 38.6 (2001), pp. 2022–2046.

[222] Andrew J. Sommese and Charles W. Wampler II. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. Singapore: World Scientific Publishing Company, 2005.

[223] Laurent Sorber, Ignat Domanov, Marc Van Barel, and Lieven De Lathauwer. “Exact Line and Plane Search for Tensor Optimization”. In: *Computational Optimization and Applications* 63.1 (2016), pp. 121–142.

[224] Laurent Sorber, Marc Van Barel, and Lieven De Lathauwer. “Numerical Solution of Bivariate and Polyanalytic Polynomial Systems”. In: *SIAM Journal on Numerical Analysis* 52.4 (2014), pp. 1551–1572.

[225] Laurent Sorber, Marc Van Barel, and Lieven De Lathauwer. “Unconstrained Optimization of Real Functions in Complex Variables”. In: *SIAM Journal on Optimization* 22.3 (2012), pp. 879–898.

[226] Frank Sottile. *Real Solutions to Equations from Geometry*. University Lecture Series. Providence, RI, USA: American Mathematical Society, 2011.

[227] John T. Spanos, Mark H. Milman, and Lewis D. Mingori. "A New Algorithm for L_2 Optimal Model Reduction". In: *Automatica* 28.5 (1992), pp. 897–909.

[228] Wilhelm Specht. "Die Lage der Nullstellen eines Polynoms III". In: *Mathematische Nachrichten* 16.5–6 (1957). [citation only], pp. 369–389.

[229] Hans J. Stetter. "Matrix Eigenproblems Are at the Heart of Polynomial System Solving". In: *ACM SIGSAM Bulletin* 30.4 (1996), pp. 22–25.

[230] Hans J. Stetter. *Numerical Polynomial Algebra*. Philadelphia, PA, USA: SIAM, 2004.

[231] Gilbert W. Stewart. "An Updating Algorithm for Subspace Tracking". In: *IEEE Transactions on Signal Processing* 40.6 (1992), pp. 1535–1541.

[232] Gilbert W. Stewart. "Updating a Rank-Revealing ULV Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 14.2 (1993), pp. 494–499.

[233] Stephen M. Stigler. "Stigler's Law of Eponymy". In: *Transactions of the New York Academy of Sciences* 39.1 (1980), pp. 147–157.

[234] Gilbert Strang. "Graphs, Matrices, and Subspaces". In: *The College Mathematics Journal* 24.1 (1993), pp. 20–28.

[235] Gilbert Strang. *Introduction to Linear Algebra*. 6th. Wellesley, MA, USA: Cambridge University Press, 2023.

[236] Gilbert Strang. "The Fundamental Theorem of Linear Algebra". In: *The American Mathematical Monthly* 100.9 (1993), pp. 848–855.

[237] Bernd Sturmfels. *Solving Systems of Polynomial Equations*. Vol. 97. Regional Conference Series in Mathematics. Providence, RI, USA: American Mathematical Sciences, 2002.

[238] Bernd Sturmfels. "WHAT IS ...a Gröbner Basis". In: *Notices of the American Mathematical Society* 52.10 (2005), pp. 1199–1200.

[239] James J. Sylvester. "On a Theory of the Syzygetic Relations of Two Rational Integral Functions, Comprising an Application to the Theory of Sturm's Functions, and That of the Greatest Algebraical Common Measure". In: *Philosophical Transactions of the Royal Society of London* 143 (1853), pp. 407–548.

[240] Huanze Tang. "Stock Prices Prediction Based on ARMA Model". In: *Proc. of the 2021 International Conference on Computer, Blockchain and Financial Development (CBFD)*. Nanjing, China, 2021, pp. 201–204.

[241] Simon Telen. *Solving Polynomial Equations and Applications*. Tech. rep. Lecture notes. Amsterdam, The Netherlands: Centrum Wiskunde & Informatica, 2022.

[242] Simon Telen. “Solving Systems of Polynomial Equations”. PhD thesis. Leuven, Belgium: KU Leuven, 2020.

[243] Simon Telen, Bernard Mourrain, and Marc Van Barel. “Solving Polynomial Systems via Truncated Normal Forms”. In: *SIAM Journal on Matrix Analysis and Applications* 39.3 (2018), pp. 1421–1447.

[244] Simon Telen and Marc Van Barel. “A Stabilized Normal Form Algorithm for Generic Systems of Polynomial Equations”. In: *Journal of Computational and Applied Mathematics* 342 (2018), pp. 119–132.

[245] Françoise Tisseur and Karl Meerbergen. “The Quadratic Eigenvalue Problem”. In: *SIAM Review* 43.2 (2001), pp. 235–286.

[246] Erik R. Tou. *Math Origins: Eigenvectors and Eigenvalues*. `https://www.maa.org/press/periodicals/convergence/math-origins-eigenvectors-and-eigenvalues`. 2018.

[247] Lloyd N. Trefethen. *Approximation Theory and Approximation Practice*. extended. Philadelphia, PA, USA: SIAM, 2019.

[248] Llyod N. Trefethen. “Six Myths of Polynomial Interpolation and Quadrature”. In: *Mathematics Today* (2011), pp. 184–188.

[249] Llyod N. Trefethen and David Bau III. *Numerical Linear Algebra*. Philadelphia, PA, USA: SIAM, 1997.

[250] Paul M. Van Dooren. “A Generalized Eigenvalue Approach for Solving Riccati Equations”. In: *SIAM Journal on Scientific and Statistical Computing* 2.2 (1981), pp. 121–135.

[251] Paul M. Van Dooren. “The Generalized Eigenstructure Problem in Linear System Theory”. In: *IEEE Transactions on Automatic Control* 26.1 (1981), pp. 111–129.

[252] Paul M. Van Dooren, Kyle A. Gallivan, and Pierre-Antoine Absil. “$\mathcal{H}_2$-Optimal Model Reduction of MIMO Systems”. In: *Applied Mathematics Letters* 21.12 (2008), pp. 1267–1273.

[253] Peter Van Overschee and Bart De Moor. “N4SID: Subspace Algorithms for the Identification of Combined Deterministic-Stochastic Systems”. In: *Automatica* 30.1 (1994), pp. 75–93.

[254] Peter Van Overschee and Bart De Moor. *Subspace Identification for Linear Systems: Theory — Implementation — Applications*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1996.

[255] Jeroen Vanderstukken and Lieven De Lathauwer. “Systems of Polynomial Equations, Higher-Order Tensor Decompositions and Multidimensional Harmonic Retrieval: An Unifying Framework. Part I: The Canonical Polyadic Decomposition”. In: *SIAM Journal on Matrix Analysis and Applications* 42.2 (2021), pp. 883–912.

[256] Jeroen Vanderstukken, Patrick Kürschner, Ignat Domanov, and Lieven De Lathauwer. “Systems of Polynomial Equations, Higher-Order Tensor Decompositions and Multidimensional Harmonic Retrieval: An Unifying Framework. Part II: The Block-Term Decomposition”. In: *SIAM Journal on Matrix Analysis and Applications* 42.2 (2021), pp. 913–953.

[257] Christof Vermeersch. *MacaulayLab: A Numerical Linear Algebra Solver for System of Multivariate Polynomials and Multiparameter Eigenvalue Problems*. Tech. rep. `https://www.macaulaylab.net`. Leuven, Belgium: KU Leuven, 2023.

[258] Christof Vermeersch and Bart De Moor. "A Column Space Based Approach to Solve Systems of Multivariate Polynomial Equations". In: *IFAC-PapersOnLine* 54.9 (2021). Part of special issue: 24th International Symposium on Mathematical Theory of Networks and Systems (MTNS), pp. 137–144.

[259] Christof Vermeersch and Bart De Moor. "Globally Optimal Least-Squares ARMA Model Identification Is an Eigenvalue Problem". In: *IEEE Control Systems Letters* 3.4 (2019), pp. 1062–1067.

[260] Christof Vermeersch and Bart De Moor. "Recursive Algorithms to Update a Numerical Basis Matrix of the Null Space of the Block Row, (Banded) Block Toeplitz, and Block Macaulay Matrix". In: *SIAM Journal on Scientific Computing (SISC)* 45.2 (2023), A596–A620.

[261] Christof Vermeersch and Bart De Moor. "Two Complementary Block Macaulay Matrix Algorithms to Solve Multiparameter Eigenvalue Problems". In: *Linear Algebra and its Applications* 654 (2022), pp. 177–209.

[262] Christof Vermeersch and Bart De Moor. "Two Double Recursive Block Macaulay Matrix Algorithms to Solve Multiparameter Eigenvalue Problems". In: *IEEE Control Systems Letters* 7 (2023), pp. 319–324.

[263] Christof Vermeersch, Sibren Lagauw, and Bart De Moor. *Multivariate Polynomial Optimization in Complex Variables Is a (Rectangular) Multiparameter Eigenvalue Problem*. Tech. rep. Accepted for publication. Leuven, Belgium: KU Leuven, 2023.

[264] Jan Verschelde. "Algorithm 795: PHCpack: A General-Purpose Solver for Polynomial Systems by Homotopy Continuation". In: *ACM Transactions on Mathematical Software* 25.2 (1999), pp. 251–276.

[265] Jan Verschelde. "Homotopy Continuation Methods for Solving Polynomial Systems". PhD thesis. Leuven, Belgium: KU Leuven, 1996.

[266] Jan Verschelde. *The Test Database of Polynomial Systems*. `http://homepages.math.uic.edu/~jan/PHCpack/node10.html`. (Visited on April 1, 2021). 1999.

[267] Hans Volkmer. *Multiparameter Eigenvalue Problems and Expansion Theorems*. Vol. 1356. Lecture Notes in Mathematics. Berlin, Germany: Springer, 1988.

[268] Morris A. Walker. "Note on a Generalization of the Large Sample Goodness of Fit Test for Linear Autoregressive Schemes". In: *Journal of the Royal Statistical Society* 12.1 (1950), pp. 102–107.

[269] Charles W. Wampler II and Andrew J. Sommese. "Numerical Algebraic Geometry and Algebraic Kinematics". In: *Acta Numerica* 20 (2011), pp. 469–567.

[270] David S. Watkins. *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*. Other Titles in Applied Mathematics. Philadelphia, PA, USA: SIAM, 2007.

[271] Layne T. Watson, Stephen C. Billups, and Alexander P. Morgan. "Algorithm 652: HOMPACK: A Suite of Codes for Globally Convergent Homotopy Algorithms". In: *ACM Transactions on Mathematical Software* 13.3 (1987), pp. 281–310.

[272] Peter Whittle. "Hypothesis Testing in Time Series Analysis". [citation only]. PhD thesis. Uppsala, Sweden: Uppsala University, 1951.

[273] Michael A. Wicks and Raymond A. DeCarlo. "Computing Most Nearly Rank-Reducing Structured Matrix Perturbations". In: *SIAM Journal on Matrix Analyis and Applications* 16.1 (1995), pp. 123–137.

[274] Raphaël Widdershoven, Nithin Govindarajan, and Lieven De Lathauwer. "Overdetermined Systems of Polynomial Equations: Tensor-Based Solution and Application". In: *Proc. of the 31st European Signal Processing Conference (EUSIPCO)*. Helsinki, Finland, 2023, pp. 650–654.

[275] James H. Wilkinson. "The Perfidious Polynomial". In: *Studies in Numerical Analysis*. Ed. by Gene H. Golub. Vol. 24. Washington, D.C., USA: Mathematical Association of America, 1984, pp. 1–28.

[276] Morten Willatzen and Lew Yan L. C. Voon. *Separable Boundary-Value Problems in Physics*. Weinheim, Germany: Wiley-VCH, 2011.

[277] Jan C. Willems. "From Time Series to Linear Systems — Part I. Finite Dimensional Linear Time Invariant Systems". In: *Automatica* 22.5 (1986), pp. 561–580.

[278] Jan C. Willems. "From Time Series to Linear Systems — Part II. Exact Modelling". In: *Automatica* 22.6 (1986), pp. 675–694.

[279] Jan C. Willems. "From Time Series to Linear Systems — Part III. Approximate Modelling". In: *Automatica* 23.1 (1987), pp. 87–115.

[280] Wilhelm Wirtinger. "Zur Formalen Theorie der Funktionen von mehr Komplexen Veränderlichen". In: *Mathematische Annalen* 97.1 (1927). [citation only], pp. 357–375.

[281] Thomas G. Wright and Lloyd N. Trefethen. "Pseudospectra of Rectangular Matrices". In: *IMA Journal on Matrix Analysis* 22.4 (2002), pp. 501–519.

[282] Wei-Yong Yan and James Lam. "An Approximate Approach to H_2 Optimal Model Reduction". In: *IEEE Transactions on Automatic Control* 44.7 (1999), pp. 1341–1358.

[283] Udny G. Yule. "On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers". In: *Philosophical Transactions of the Royal Society of London* 226.636-646 (1927), pp. 267–298.

[284] Zhonggang Zeng. “The Closedness Subspace Method for Computing the Multiplicity Structure of a Polynomial System”. In: *Contemporary Mathematics: Interactions of Classical and Numerical Algebraic Geometry*. Ed. by Daniel J. Bates, Gian M. Besana, Sandra Di Rocco, and Charles W. Wampler II. Vol. 496. 2009, pp. 347–361.

[285] Dragan Žigić, Layne T. Watson, and Christopher A. Beattie. “Contragredient Transformations Applied to the Optimal Projection Equations”. In: *Linear Algebra and its Applications* 188–189 (1993), pp. 665–676.

List of Publications

Articles in internationally reviewed academic journals.

Christof Vermeersch and Bart De Moor. "Globally Optimal Least-Squares ARMA Model Identification Is an Eigenvalue Problem". In: *IEEE Control Systems Letters* 3.4 (2019), pp. 1062–1067

Christof Vermeersch and Bart De Moor. "A Column Space Based Approach to Solve Systems of Multivariate Polynomial Equations". In: *IFAC-PapersOnLine* 54.9 (2021). Part of special issue: 24th International Symposium on Mathematical Theory of Networks and Systems (MTNS), pp. 137–144

Oscar M. Agudelo, Christof Vermeersch, and Bart De Moor. "Globally Optimal $\mathcal{H}_2$-Norm Model Reduction: A Numerical Linear Algebra Approach". In: *IFAC-PapersOnLine* 54.9 (2021). Part of special issue: 24th International Symposium on Mathematical Theory of Networks and Systems (MTNS), pp. 564–571

Christof Vermeersch and Bart De Moor. "Two Complementary Block Macaulay Matrix Algorithms to Solve Multiparameter Eigenvalue Problems". In: *Linear Algebra and its Applications* 654 (2022), pp. 177–209

Oliver Lauwers, Christof Vermeersch, and Bart De Moor. "Cepstral Identification of Autoregressive Systems". In: *Automatica* 139 (2022), 110214:1–14

Christof Vermeersch and Bart De Moor. "Recursive Algorithms to Update a Numerical Basis Matrix of the Null Space of the Block Row, (Banded) Block Toeplitz, and Block Macaulay Matrix". In: *SIAM Journal on Scientific Computing (SISC)* 45.2 (2023), A596–A620

Christof Vermeersch and Bart De Moor. "Two Double Recursive Block Macaulay Matrix Algorithms to Solve Multiparameter Eigenvalue Problems". In: *IEEE Control Systems Letters* 7 (2023), pp. 319–324

Articles accepted for publication.

Christof Vermeersch, Sibren Lagauw, and Bart De Moor. *Multivariate Polynomial Optimization in Complex Variables Is a (Rectangular) Multiparameter Eigenvalue Problem.* Tech. rep. Accepted for publication. Leuven, Belgium: KU Leuven, 2023

Invited talks and seminar presentations.

Christof Vermeersch, Oscar M. Agudelo, Katrien De Cock, and Bart De Moor. *Multiparameter Eigenvalue Problems in Systems and Control.* Seminar presentation at the Structured Low-Rank Matrix/Tensor Approximation (SeLMA) Seminar. Leuven, Belgium, 2019

Christof Vermeersch and Bart De Moor. *System Identification Problems as Multiparameter Eigenvalue Problems.* Invited talk at the 30th ERNSI Workshop in System Identification. Leuven, Belgium, 2022

Presentations and posters at international conferences.

Christof Vermeersch, Oscar M. Agudelo, and Bart De Moor. *Solving Multivariate Polynomial Optimization Problems via Numerical Linear Algebra.* Presentation at the 37th Benelux Meeting on Systems and Control. Soesterberg, The Netherlands, 2018

Christof Vermeersch and Bart De Moor. *Globally Optimal Least-Squares ARMA Model Identification Is an Eigenvalue Problem.* Poster at the 27th ERNSI Workshop in System Identification. Cambridge, UK, 2018

Christof Vermeersch and Bart De Moor. *Globally Optimal Least-Squares ARMA Model Identification Is an Eigenvalue Problem.* Presentation at the 38th Benelux Meeting on Systems and Control. Lommel, Belgium, 2019

Christof Vermeersch and Bart De Moor. *Linear Multiparameter Eigenvalue Problems.* Poster at the 28th ERNSI Workshop in System Identification. Maastricht, The Netherlands, 2019

Christof Vermeersch and Bart De Moor. *Globally Optimal Least-Squares ARMA Model Identification Is an Eigenvalue Problem.* Presentation at the 58th Conference on Decision and Control (CDC). Nice, France, 2019

Christof Vermeersch and Bart De Moor. *A Column Space Based Approach to Solve Multiparameter Eigenvalue Problems.* Presentation at the 39th Benelux Meeting on Systems and Control. Elspeet, The Netherlands, 2020

Christof Vermeersch and Bart De Moor. *A Recursive Algorithm to Compute a Numerical Basis Matrix of the Null Space of the Block Macaulay Matrix.* Poster at the 29th ERNSI Workshop in System Identification. Rennes, France, 2021

Christof Vermeersch and Bart De Moor. *Two Double Recursive Block Macaulay Matrix Algorithms to Solve Multiparameter Eigenvalue Problems.* Presentation at the 61st Conference on Decision and Control (CDC). Cancun, Mexico, 2022

Christof Vermeersch, Katrien De Cock, and Bart De Moor. *Exploiting Shift-Invariant Subspaces.* Poster at the SeLMA Closing Event. Leuven, Belgium, 2022

Christof Vermeersch, Sarthak De, and Bart De Moor. *About (Rectangular) Multiparameter Eigenvalue Problems*. Presentation at the 42nd Benelux Meeting on Systems and Control. Elspeet, The Netherlands, 2023

Christof Vermeersch and Bart De Moor. *About the Fundamental Subspaces of the Macaulay Matrix*. Presentation at the 6th SIAM Conference on Applied Algebraic Geometry (SIAM AG). Eindhoven, The Netherlands, 2023