A Sylvester–Arnoldi type method for the generalized eigenvalue problem with two-by-two operator determinants

Karl Meerbergen *Bor Plestenjak*

Report TW 653, August 2014



KU Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A Sylvester–Arnoldi type method for the generalized eigenvalue problem with two-by-two operator determinants

Karl Meerbergen *Bor Plestenjak*

Report TW 653, August 2014

Department of Computer Science, KU Leuven

Abstract

In various applications, for instance in the detection of a Hopf bifurcation or in solving separable boundary value problems using the two-parameter eigenvalue problem, one has to solve a generalized eigenvalue problem of the form

$$(B_1 \otimes A_2 - A_1 \otimes B_2)z = \mu(B_1 \otimes C_2 - C_1 \otimes B_2)z,$$

where matrices are 2×2 operator determinants. We present efficient methods that can be used to compute a small subset of the eigenvalues. For full matrices of moderate size we propose either the standard implicitly restarted Arnoldi or Krylov–Schur iteration with shift-and-invert transformation, performed efficiently by solving a Sylvester equation. For large problems, it is more efficient to use subspace iteration based on low-rank approximations of the solution of the Sylvester equation combined with a Krylov–Schur method for the projected problems.

Keywords : Generalized eigenvalue problem, Sylvester equation, Bartels-Stewart algorithm, inverse iteration, subspace iteration, Arnoldi method, two-parameter eigenvalue problem, Mathieu's system, Hopf bifurcation, low-rank approximation.
MSC : Primary : 15A18, 65F15

A Sylvester–Arnoldi type method for the generalized eigenvalue problem with two-by-two operator determinants

Karl Meerbergen * Bor Plestenjak †

August 13, 2014

Abstract

In various applications, for instance in the detection of a Hopf bifurcation or in solving separable boundary value problems using the two-parameter eigenvalue problem, one has to solve a generalized eigenvalue problem of the form

$$(B_1 \otimes A_2 - A_1 \otimes B_2)z = \mu(B_1 \otimes C_2 - C_1 \otimes B_2)z,$$

where matrices are 2×2 operator determinants. We present efficient methods that can be used to compute a small subset of the eigenvalues. For full matrices of moderate size we propose either the standard implicitly restarted Arnoldi or Krylov–Schur iteration with shift-and-invert transformation, performed efficiently by solving a Sylvester equation. For large problems, it is more efficient to use subspace iteration based on low-rank approximations of the solution of the Sylvester equation combined with a Krylov–Schur method for the projected problems.

Keywords: Generalized eigenvalue problem, Sylvester equation, Bartels-Stewart algorithm, inverse iteration, subspace iteration, Arnoldi method, two-parameter eigenvalue problem, Mathieu's system, Hopf bifurcation, low-rank approximation.

1 Introduction

In several applications, one can find a generalized eigenvalue problem of the form

$$(B_1 \otimes A_2 - A_1 \otimes B_2)z = \mu(B_1 \otimes C_2 - C_1 \otimes B_2)z, \tag{1}$$

where matrices A_i, B_i, and C_i are $n_i \times n_i$ matrices for $i = 1, 2$. If we define 2×2 operator determinants

$$\begin{aligned} M_1 &= B_1 \otimes A_2 - A_1 \otimes B_2 \\ M_0 &= B_1 \otimes C_2 - C_1 \otimes B_2, \end{aligned}$$

then we have a generalized eigenvalue problem

$$M_1 z = \mu M_0 z \tag{2}$$

*KULeuven, Department of Computer Science, 3001 Leuven, Belgium. Karl.Meerbergen@cs.kuleuven.be

†IMFM and Department of Mathematics, University of Ljubljana, Jadranska 19, SI-1000 Ljubljana, Slovenia. bor.plestenjak@fmf.uni-lj.si

with matrices of size $n_1 n_2 \times n_1 n_2$. If n_1 and n_2 are not too large, we can use one of the existing numerical methods for the generalized eigenvalue problem (2), for example, the QZ algorithm, and compute all of the eigenvalues. The complexity of this approach is $\mathcal{O}(n_1^3 n_2^3)$ flops.

If $n_1 n_2$ is too large for the QZ and other methods that compute all eigenvalues, then we are interested in a subspace method that computes a small number of eigenvalues close to a given target. One of the popular choices is implicitly restarted Arnoldi, where in each step one has to solve a linear system with matrix $M_1 - \sigma M_0$, where σ is an appropriate shift. Without any optimizations, this requires $\mathcal{O}(n_1^3 n_2^3)$ operations due to the size of matrices M_0 and M_1, but, as we show in Section 3, this can be done much faster in $\mathcal{O}(n_1^3 + n_2^3)$ operations using the Bartels-Stewart algorithm [2].

If matrices in (1) are large, even the Bartels-Stewart algorithm is too expensive. The eigenvector z in (1) is usually a tensor of low-rank, which means that it can be represented as

$$z = x_{11} \otimes x_{21} + \cdots + x_{k1} \otimes x_{k2}, \tag{3}$$

where k is very small. For instance, $k = 1$ for the two-parameter eigenvalue problems from Subsection 2.1 and $k = 2$ for problems in computing Hopf bifurcations from Subsection 2.2. For large matrices, we propose new methods in Section 5 and 6 that use low-rank approximations. These are based on the fact that the iteration vectors of inverse iteration converge to such low rank vectors and can therefore be more efficiently represented. Such low rank solutions can be computed using Krylov methods for solving Sylvester equations [3, 7]. This leads to a further reduction of the computational cost.

When (2) arises from a two-parameter eigenvalue problem, the Jacobi-Davidson method [5] is usually an efficient solution method, i.e., it computes several instances of pairs (λ, μ). The method is particularly reliable when pairs nearest to a target point (σ, τ) are looked for. Our methods, however, are suitable for applications, where τ is given, but a good value of σ is not available, since it is unknown where λ could be. The methods rely on the assumption that the solution of a Sylvester equation related to (2) can be well approximated by a low rank matrix and that such low rank solution can efficiently be computed numerically.

The paper is organized as follows. In Section 2, we present some applications that lead to problems of the form (1). We focus mainly on the two-parameter eigenvalue problem, but we also deal with Hopf bifurcations and all generic problems of the form (1). In Section 3, we apply Arnoldi's method to (2) where the shift-and-invert operator is computed using the Bartels-Stewart algorithm. In Section 4, we combine Arnoldi's method with locking with low-rank vectors. In Section 5, we extend the Lyapunov inverse iteration method [10] to the more general problem (1), update it first with subspace iteration and then with subspace iteration in Section 6. In Section 7, we give some numerical examples and show that the presented methods outperform existing approaches for Mathieu's system, presented in Section 2.

2 Motivating examples

2.1 Eigenmodes of an elliptic membrane

We would like to compute efficiently and accurately a couple of hundreds eigenmodes of an elliptic membrane Ω with a fixed boundary:

$$\left(\Delta + \omega^2\right) \psi\,(x, y) = 0, \quad (x, y) \in \Omega = \{(x/\alpha)^2 + (y/\beta)^2 \leq 1\}, \quad \psi|_{\partial\Omega} = 0.$$

Recently, a numerical method for this task that uses the two-parameter eigenvalue problem was proposed in [4]. If we apply separation of variables using elliptical coordinates ξ and η,

$$\begin{aligned} x &:= h\cosh\xi\cos\eta, \\ y &:= h\sinh\xi\sin\eta, \quad 0 \le \xi < \infty, \quad 0 \le \eta < 2\pi, \end{aligned}$$

then we obtain the coupled system of Mathieu's angular and radial equations (for details, see, e.g., [17])

$$\begin{aligned} G''(\eta) + (\lambda - 2\mu\cos 2\eta)\, G(\eta) &= 0 \\ F''(\xi) - (\lambda - 2\mu\cosh 2\xi)\, F(\xi) &= 0, \end{aligned} \tag{4}$$

with four different (π-even, 2π-even, π-odd, 2π-odd) boundary conditions. For example, π-even boundary conditions are $G'(0) = G'(\pi/2) = 0$ and $F'(0) = F(\xi_0) = 0$. Here $\xi_0 := \operatorname{arccosh}\frac{\alpha}{h}$, where $h = \sqrt{\alpha^2 - \beta^2}$. The parameter μ is related to the eigenfrequency ω by

$$\mu = \frac{h^2\omega^2}{4},$$

while λ is a result of the separation of variables.

Mathieu's system (4) is a classical example of a two-parameter eigenvalue problem. Similar two-parameter eigenvalue problems that consist of Lamé equations, spheroidal wave equations, and other second order differential equations, appear when separation of variables is applied to the Laplace equation, the Helmholtz equation, or the Schrödinger equation, see, e.g., [18]. If we linearize the differential equations, we obtain an algebraic two-parameter eigenvalue problem, which has the form

$$\begin{aligned} A_1 x_1 &= \lambda B_1 x_1 + \mu C_1 x_1 \\ A_2 x_2 &= \lambda B_2 x_2 + \mu C_2 x_2, \end{aligned} \tag{5}$$

where A_i, B_i, and C_i are given $n_i \times n_i$ complex matrices. A pair (λ, μ) is an eigenvalue if it satisfies (5) for nonzero vectors $x_1 \in \mathbb{C}^{n_1}$ and $x_2 \in \mathbb{C}^{n_2}$. The corresponding eigenvector is the tensor product $x_1 \otimes x_2$. Similarly, if

$$\begin{aligned} y_1^H A_1 &= \lambda y_1^H B_1 + \mu y_1^H C_1 \\ y_2^H A_2 &= \lambda y_2^H B_2 + \mu y_2^H C_2 \end{aligned} \tag{6}$$

for nonzero vectors $y_1 \in \mathbb{C}^{n_1}$ and $y_2 \in \mathbb{C}^{n_2}$, then $y_1 \otimes y_2$ is the left eigenvector.

We define the matrices

$$\begin{aligned} \Delta_0 &= B_1 \otimes C_2 - C_1 \otimes B_2, \\ \Delta_1 &= A_1 \otimes C_2 - C_1 \otimes A_2, \\ \Delta_2 &= B_1 \otimes A_2 - A_1 \otimes B_2, \end{aligned} \tag{7}$$

which are 2×2 operator determinants. If Δ_0 is nonsingular, then matrices $\Delta_0^{-1}\Delta_1$ and $\Delta_0^{-1}\Delta_2$ commute and (5) is equivalent (for details, see, e.g., [1]) to a coupled pair of generalized eigenvalue problems

$$\begin{aligned} \Delta_1 z &= \lambda \Delta_0 z, \\ \Delta_2 z &= \mu \Delta_0 z \end{aligned} \tag{8}$$

for decomposable tensors $z = x_1 \otimes x_2$, which means that we have $k = 1$ in (3).

There exist some numerical methods for two-parameter eigenvalue problems. If $n_1 n_2$ is small, we can apply the existing numerical methods for the generalized eigenvalue problem to solve the coupled pair (8). An algorithm of this kind, which is based on the QZ algorithm, is presented in [5].

For larger values of $n_1 n_2$ it is not possible to compute all eigenvalues, but there are some iterative methods that can be used to compute a small number of solutions. Most of them require good initial approximations in order to avoid misconvergence. One of the methods is the tensor Rayleigh quotient iteration from [12], which is a generalization of the standard Rayleigh quotient iteration and computes one eigenpair at a time.

When we are interested in more than just one eigenpair and we do not have any initial approximations, we can use a Jacobi–Davidson type method [5]. A sophisticated version uses harmonic Ritz values [6] and can compute a small number of eigenvalues close to a given target.

In [4], equation (4) is discretized by Chebyshev collocation. This gives an algebraic two-parameter eigenvalue problem of the form (5), where matrices A_1 and A_2 are dense, non-symmetric and have high condition numbers, $B_1 = I$, $B_2 = -I$, and matrices C_1 and C_2 are diagonal. We are interested in parameter μ only and are looking for the smallest $|\mu|$. When n_1 and n_2 are small, we can apply the existing numerical methods (for instance `eig` in Matlab) to the related eigenvalue problem

$$\Delta_0^{-1} \Delta_2 z = \mu z.$$

It turns out that for accurate results we need matrices of moderate size, where n_1 and n_2 are typically of order 10^2. Matrices Δ_0 and Δ_2 are then so large that, if we are interested only in the first several hundred eigenmodes, it is not efficient to compute all eigenvalues using one of the direct algorithms. Instead, we apply the implicitly restarted Arnoldi (`eigs` in Matlab) to matrix $\Gamma_2 = \Delta_0^{-1} \Delta_2$, which has full matrices in its diagonal blocks and diagonal matrices in non-diagonal blocks, represented as a sparse matrix. The L and U factors of the LU decomposition of matrix $\Gamma_2 - \sigma I$ are full triangular matrices and for too large values of $n_1 n_2$ implicitly restarted Arnoldi runs out of memory. We present a simple trick that can overcome this problem in the following section. The obtained method is faster and more competitive than the Jacobi–Davidson method used in [4].

2.2 Hopf bifurcations

The main idea comes from the method in [10] for the computation of the smallest $|\mu|$ such that the large, sparse generalized eigenvalue problem

$$(A + \mu B)x = \lambda M x$$

has a pair of purely imaginary eigenvalues λ, where A, B, and M are $n \times n$ real matrices and M is symmetric and positive definite. Possible applications include the detection of Hopf bifurcations.

If the pencil $(A + \mu B) - \lambda M$ has a pair of purely imaginary eigenvalues, then

$$(A + \mu B) \otimes M + M \otimes (A + \mu B)$$

has a double eigenvalue zero. Solutions are the eigenvalues of the $n^2 \times n^2$ eigenvalue problem

$$(A \otimes M + M \otimes A)z + \mu(B \otimes M + M \otimes B)z = 0 \tag{9}$$

which has the form (1). It turns out (see [10] for details) that we can restrict z to the symmetric eigenvector space, i.e., $z = \mathrm{vec}(Z)$, where $Z = Z^T$, and apply inverse iteration (this is sufficient for the application where we need only one solution). The vector z can be represented as a sum of two decomposable tensors, i.e., the value in (3) is $k = 2$.

In each step of inverse iteration we have to solve the linear system

$$(A \otimes M + M \otimes A)w_k = (B \otimes M + M \otimes B)z_k.$$

Without any optimization, this requires $\mathcal{O}(n^6)$ operations. But, if we use the well-known equality

$$(A \otimes B)\mathrm{vec}(X) = \mathrm{vec}(BXA^T),$$

then we can write the above as

$$MW_kA^T + AW_kM = MZ_kB^T + BZ_kM,$$

where $w_k = \mathrm{vec}(W_k)$ and $z_k = \mathrm{vec}(Z_k)$. We obtain the Lyapunov equation

$$W_kA^TM^{-1} + M^{-1}AW_k = Z_kB^TM^{-1} + M^{-1}BZ_k$$

that can be solved in $\mathcal{O}(n^3)$ operations using for instance the Bartels-Stewart algorithm [2]. If $Z_k = Z_k^T$, then $W_k = W_k^T$. So, if we start with a symmetric Z_0, then all approximations in inverse iteration remain in the symmetric eigenvector space.

3 Sylvester equation and implicitly restarted Arnoldi

In a two-parameter eigenvalue problem, we have to solve the generalized eigenvalue problem

$$\Delta_2 z = \mu \Delta_0 z, \tag{10}$$

where Δ_0 and Δ_2 are 2×2 operator determinants defined in (7). For each simple eigenvalue μ we can then compute the λ part of the eigenvalue (λ, μ) from the Rayleigh quotient

$$\lambda = \frac{z^H \Delta_1 z}{z^H \Delta_0 z}.$$

We want to find solutions of (10) by applying a method based on a Krylov subspace, like implicitly restarted Arnoldi [14] or Krylov–Schur [15], to matrix $\Gamma_2 := \Delta_0^{-1}\Delta_2$, combined with the shift-and-invert approach. This means that in each step of the method we have to solve the linear system

$$(B_1 \otimes A_2 - A_1 \otimes B_2)w = (B_1 \otimes C_2 - C_1 \otimes B_2)z.$$

Following the approach from Subsection 2.2, if we write $w = \mathrm{vec}(W)$ and $z = \mathrm{vec}(Z)$, then we have

$$A_2WB_1^T - B_2WA_1^T = C_2ZB_1^T - B_2ZC_1^T =: M. \tag{11}$$

Let us assume that matrices A_1 and A_2 are nonsingular. In this case, we obtain the Sylvester equation

$$A_2^{-1}B_2W - WB_1^TA_1^{-T} = -A_2^{-1}MA_1^{-T}$$

that can be solved in $\mathcal{O}(n_1^3 + n_2^3)$ operations using, e.g., the Bartels-Stewart algorithm. If any of the matrices A_1 or A_2 is singular, then we shift the parameter λ with constant σ as

$\lambda = \widetilde{\lambda} + \sigma$, where we select σ so that the new matrices $\widetilde{A}_1 = A_1 - \sigma B_1$ and $\widetilde{A}_2 = A_2 - \sigma B_2$ are both nonsingular. The following lemma shows that such σ always exists when operator determinant Δ_2 is nonsingular. As we are looking for the smallest eigenvalue of $\Delta_0^{-1}\Delta_2$, it is natural to assume that Δ_2 is nonsingular, otherwise the smallest eigenvalue is clearly $\mu = 0$.

Lemma 1 *If operator determinant Δ_2 is nonsingular, then there exists σ such that both matrices $A_1 - \sigma B_1$ and $A_2 - \sigma B_2$ are nonsingular.*

Proof. Suppose that such σ does not exist. Then at least one of the matrix pencils $A_1 - \lambda B_1$ or $A_2 - \lambda B_2$ has to be singular. Without loss of generality we can assume that $A_1 - \lambda B_1$ is singular. This means that for all $\sigma \in \mathbb{C}$, there is an $x \in \mathbb{C}^{n_1}$ so that $(A_1 - \sigma B_1)x = 0$. As a result, for any $y \in \mathbb{C}^{n_2}$, we have that

$$\begin{aligned} \Delta_2(x \otimes y) &= B_1 x \otimes A_2 y - A_1 x \otimes B_2 y \\ &= B_1 x \otimes (A_2 - \sigma B_1) y - (A_1 - \sigma B_1) x \otimes B_2 y \\ &= B_1 x \otimes (A_2 - \sigma B_1) y \end{aligned}$$

If we now take (σ, y) as an eigenpair of the pencil $A_2 - \lambda B_2$, then $\Delta_2(x \otimes y) = 0$. This is in contradiction with the nonsingularity of Δ_2. □

This simple trick with the Sylvester equation opens completely new perspectives in solving the two-parameter eigenvalue problems. Although there are some numerical methods available for these problems, see, e.g., [5] and the references therein, they are not so widespread and simple to use for a possible user faced with a two-parameter eigenvalue problem. It is not rare that the researchers report that a problem of type (5) cannot be solved because they are not aware of any available numerical methods, see, e.g., [9].

Now, at least for problems of moderate size, where $n_1, n_2 \leq 500$, say, this trouble should be solved. We show in Appendix how can one adapt function `eigs` in Matlab to work with the two-parameter eigenvalue problems. This approach uses full vectors of size $n_1 n_2$.

Let us remark that the approach in this section can be as well applied to a general problem with 2×2 operator determinants of the form

$$(A_{11} \otimes A_{22} - A_{12} \otimes A_{21})z = \mu(B_{11} \otimes B_{22} - B_{12} \otimes B_{21})z, \tag{12}$$

(the difference from (1) is that matrices B_1 and B_2 appear in both operator determinants in (1), while there are different matrices on the left and the right side in (12)). On the contrary to the eigenvectors of (1), which are low-rank vectors, the eigenvectors of (12) are in general full-rank vectors. A problem of form (12) appears for instance in a right definite two-parameter eigenvalue problem, where all matrices are real symmetric and $\Delta_0 = B_1 \otimes C_2 - C_1 \otimes B_2$ is symmetric and positive definite, when we are looking for the smallest eigenvalue of Δ_0 [16].

If we want to compute a large number of eigenvalues and $n_1 n_2$ is large, then the Arnoldi method might require too much memory for the storage of the iteration vectors of size $n_1 n_2$. In the following section, we show how to circumvent this problem by using low-rank vectors for the basis of the converged invariant subspace.

4 Locking with low-rank vectors

Suppose that $n_1 n_2$ is so large that the available memory puts a limit on the size of the subspace that we can use in the Arnoldi method. If the number of wanted eigenvalues is

larger than that, we can exploit the fact that all eigenvectors are low-rank vectors and apply locking. In the following, we present more details for the implicitly restarted Arnoldi iteration with locking [8], but the same principles can be applied to Krylov-Schur and other iterative methods.

Let as assume that all matrices A_1, B_1, C_1, A_2, B_2, and C_2 are real and that we are looking for the eigenvalues of $\Gamma_2 = \Delta_0^{-1}\Delta_2$ with the smallest absolute value. Suppose that we already computed and locked k eigenvalues using an approximate partial Schur form

$$\Gamma_2^{-1}Q - QS = E$$

where S is an upper quasi-triangular matrix, the columns of Q are orthonormal, and $\|E\|_F$ is much smaller than $\|S\|$. (We therefore set that $E = 0$, which is called locking.) This means that in the Arnoldi algorithm for matrix Γ_2^{-1} we have

$$\Gamma_2^{-1}\begin{bmatrix} Q & W \end{bmatrix} = \begin{bmatrix} Q & W \end{bmatrix}\begin{bmatrix} S & G \\ 0 & H \end{bmatrix}\begin{bmatrix} E & h_{m+1,m}w_{m+1}e_m^T \end{bmatrix},$$

where $\begin{bmatrix} Q & W \end{bmatrix}$ has $k+m$ orthogonal columns, S is an upper quasi-triangular matrix, and H is a Hessenberg matrix. The columns of W form an orthogonal basis for the active search subspace and the eigenvalues of H are Ritz values that correspond to this subspace.

Eigenvectors of (1) are low rank vectors and this also applies to columns of Q, as they are linear combinations of a small number of eigenvectors. In particular, each column in matrix Q can be represented as $q_j = \mathrm{vec}(UD_jV^H)$, where matrices U and V have k columns and D_j is a $k \times k$ matrix for $j = 1, \ldots, k$. In this way we save memory and we can still do computations with the vectors from the locked part efficiently. When we lock new vectors and extend Q with new columns, we extend matrices U and V and adjust the sizes of matrices D_j.

This approach enables us to use the maximum possible active subspace, which is limited only with the available memory. As the memory requirements for the locked part are relatively negligible, the number of the computed eigenvalues can be much larger than the maximum size of the active subspace.

If, however, n_1n_2 is so large, that we cannot keep enough vectors of size n_1n_2 in memory, then this approach cannot be applied. In such case we can use methods with low-rank vectors, which are presented in the next two sections.

5 Subspace iteration

In our application to two-parameter eigenvalue problems, the Sylvester equation

$$A_2WB_1^T - B_2WA_1^T = C_2ZB_1^T - B_2ZC_1^T$$

is related to system $\Delta_2 w = \Delta_0 z$ or

$$(B_1 \otimes A_2 - A_1 \otimes B_2)w = (B_1 \otimes C_2 - C_1 \otimes B_2)z. \tag{13}$$

Let us assume that matrices A_1, B_1, C_1, A_2, B_2, and C_2 are large and real. If n_1n_2 is so large that the Bartels-Stewart algorithm is unfeasible, or that we cannot hold vectors of size n_1n_2 in memory for the Arnoldi method, then we can perform inverse iteration with low-rank vectors in a similar way as in [10] and solve system (13) approximately. In general, it is

difficult to extract complex conjugate pairs of real matrices using (inexact) inverse iteration. For computing complex eigenvalues, and in order to speed up the convergence of inverse iteration, in particular when the absolute values of the two smallest eigenvalues do not differ much, we rather use a subspace iteration version. To make the presentation clearer, we will first write down a version of subspace iteration which still uses full vectors of size $n_1 n_2$.

In order to facilitate the computation of many eigenvalues, we use a form of deflation or locking. In this paper, we employ the unconventional Hotelling's deflation. Suppose that we have already computed eigenvalues $\mu_1, \ldots, \mu_m$ of (10) with the corresponding left and right eigenvectors $x^{(i)} = x_{i1} \otimes x_{i2}$ and $y^{(i)} = y_{i1} \otimes y_{i2}$ for $i = 1, \ldots, m$. In order to compute the next eigenvalues, we apply subspace iteration to matrix

$$S := \Gamma_2^{-1} - \sum_{i=1}^{m} \frac{1}{\mu_i} \cdot \frac{x^{(i)} y^{(i)H} \Delta_0}{y^{(i)H} \Delta_0 x^{(i)}} = \left(I - \sum_{i=1}^{m} \frac{x^{(i)} y^{(i)H}}{y^{(i)H} \Delta_0 x^{(i)}} \right) \Gamma_2^{-1}. \tag{14}$$

In the above formula, we use the fact that, if y is a left eigenvector of (10) for eigenvalue μ, then $\Delta_2^H y$ is a left eigenvector of matrix $\Gamma_2^{-1} = \Delta_2^{-1}\Delta_0$ for eigenvalue μ^{-1}. If we assume that all eigenvalues $\mu_1, \ldots, \mu_m$ are algebraically simple, then it is easy to see that $Sx^{(i)} = 0$ for $i = 1, \ldots, m$ and $Sz = \Gamma_2^{-1} z$ if z is an eigenvector of (10) for the eigenvalue $\mu \neq \mu_i$ for $i = 1, \ldots, m$.

We are aware that Hotelling's deflation is not considered to be a very accurate approach to compute many eigenvalues (see, e.g, [13]). However, in this specific application, its advantages overcome its weaknesses. The most important property is that Hotelling's deflation does not change the remaining left and right eigenvectors. The eigenvectors are still tensor decomposable and we will show later how this can be exploited by iterating low-rank vectors.

The initial version of subspace iteration is presented in Algorithm 1. This algorithm, which still uses full vectors and solves the Sylvester equation exactly, is just an intermediate solution approach that we give for clarity. The final versions do not use any explicit matrices or vectors of size $\mathcal{O}(n_1 n_2)$.

As all matrices A_1, B_1, C_1, A_2, B_2, and C_2 are real, we can use real arithmetic for all operations because complex eigenvalues and eigenvectors appear in conjugate pairs. Once we have a conjugate eigenpair, we can deflate in real arithmetic. We first show how this can be done for the standard eigenvalue problem.

Lemma 2 *Let μ be a complex eigenvalue of a real matrix A with the corresponding right eigenvector $x = x_1 + ix_2$ and left eigenvector $y = y_1 + iy_2$. Then matrix*

$$B = A - \mu \frac{xy^H}{y^H x} - \overline{\mu} \frac{\overline{xy}^H}{\overline{y}^H \overline{x}},$$

which corresponds to Hotelling's deflation, is real and can be written as

$$B = \left(I - 2 \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T \right) A,$$

where

$$\alpha + i\beta = \frac{1}{y^H x}.$$

Let $x^{(i)} = x_{i1} \otimes x_{i2}$ and $y^{(i)} = y_{i1} \otimes y_{i2}$ for $i = 1, \ldots, m$ be known left and right eigenvectors for the algebraically simple eigenvalues $\mu_1, \ldots, \mu_m$
Choose nonzero linearly independent vectors $z_1, \ldots, z_p$.
for $k = 0, 1, \ldots$ **do**
  for $j = 1, \ldots, p$ **do**
    Solve $\Delta_2 w_j = \Delta_0 z_j$.
    $\widetilde{w}_j = w_j - \sum_{i=1}^{m} \frac{{y^{(i)}}^T w_j}{{y^{(i)}}^T \Delta_0 x^{(i)}} x^{(i)}$ (Hotelling's deflation)
  end for
  Compute $R = Z_k^T \widetilde{W}$, where $\widetilde{W} = \begin{bmatrix} \widetilde{w}_1 & \cdots & \widetilde{w}_p \end{bmatrix}$.
  Compute eigenpairs (σ_i, q_i), $i = 1, \ldots, p$, of matrix R.
  for $j = 1, \ldots, p$ **do**
    if $(\sigma_j, Z_k q_j)$ is an eigenpair of Γ_2^{-1} **then** extract the eigenpair
  end for
  Let Z_{k+1} be the Q-factor of the QR factorization of $\widetilde{W}$.
end for

Algorithm 1: Subspace iteration with Hotelling's deflation for the generalized eigenvalue problem (10) related to the two-parameter eigenvalue problem (5) with real matrices.

We can now apply the same method and do Hotelling's deflation in Algorithm 1 for a conjugate pair of complex eigenvalues in real arithmetic. The details are in the following corollary.

Corollary 3 *Let μ be a complex eigenvalue for the generalized eigenvalue problem (10) related to the two-parameter eigenvalue problem (5) with the corresponding right eigenvector $x = x_1 \otimes x_2$ and left eigenvector $y = y_1 \otimes y_2$, where $x_j = x_{j1} + i x_{j2}$ and $y_j = y_{j1} + i y_{j2}$ for $j = 1, 2$. Then*

$$2\mathrm{Re}\left(\frac{x y^H \Delta_2}{y^H \Delta_2 x}\right) = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} \begin{bmatrix} b_1 & b_2 \end{bmatrix}^T,$$

where

$$\begin{aligned} a_1 &= x_{11} \otimes x_{21} - x_{12} \otimes x_{22}, \\ a_2 &= x_{11} \otimes x_{22} + x_{12} \otimes x_{21}, \\ b_1 &= B_1 y_{11} \otimes A_2 y_{21} - B_1 y_{12} \otimes A_2 y_{22} - A_1 y_{11} \otimes B_2 y_{21} + A_1 y_{12} \otimes B_2 y_{22}, \\ b_2 &= B_1 y_{11} \otimes A_2 y_{22} + B_1 y_{12} \otimes A_2 y_{21} - A_1 y_{11} \otimes B_2 y_{22} - A_1 y_{12} \otimes B_2 y_{21}, \end{aligned}$$

and

$$\alpha + i\beta = \frac{1}{y^H \Delta_2 x}.$$

The goal of Algorithm 1 is to develop the approximate partial Schur form

$$\Gamma_2^{-1} Z = Z R,$$

where $R \in \mathbb{R}^{p \times p}$ is quasi upper triangular and $Z \in \mathbb{R}^{n_1 n_2 \times p}$ has orthonormal columns. The subspace iteration method on $\Gamma_2^{-1} = \Delta_2^{-1} \Delta_0$ is implemented by solving linear systems with Δ_2 and right-hand sides $z_1, \ldots, z_p$, where z_j is the jth column of Z. Let us explain the details of Algorithm 1.

- In Line 8, we compute matrix $R = Z_k^T S Z_k$, where S from (14) is Γ_2^{-1} updated by Hotelling's deflation. The elements of matrix R are computed as the inner products of vectors $z_i^{(k)}$ and $\widetilde{w}_j$ for $i, j = 1, \ldots, p$. Recall that the columns of Z_k are approximate Schur vectors, so, matrix R should converge to a quasi upper triangular matrix as k goes to infinity.

- Hotelling's deflation enables us to use low-dimensional subspaces and still compute many eigenvalues. An alternative would be to lock the computed eigenvectors and keep them in the subspace, but, then the size of the subspace would have to grow. For each new eigenpair in Line 11 we compute the left eigenvector by inverse iteration directly on (5) and add it to the set of known eigenpairs that is used for Hotelling's deflation.

If we have very large and sparse matrices, then Algorithm 1 in its current form is unfeasible. A solution is to use low-rank vectors in a similar way as in [10]. If all eigenvalues are simple, the eigenvectors can be represented as rank-one tensors and the associated Schur vectors can be represented in low-rank form. We observed that the iterates of Algorithm 1 lead to z_j with low-rank tensor structures. We therefore represent the jth column of matrix Z_k as $z_j^{(k)} = \mathrm{vec}(U_k D_j^{(k)} V_k^T)$, where $U_k \in \mathrm{R}^{n_2 \times \ell}$, $V_k \in \mathrm{R}^{n_1 \times \ell}$, and $D_j^{(k)}$ is an $\ell \times \ell$ matrix. Note that U_k and V_k are the same for all $j = 1, \ldots, p$. So, instead of using Z_k, we can store and use the $\ell^2 \times p$ matrix $D_k = \begin{bmatrix} d_1^{(k)} & \cdots & d_p^{(k)} \end{bmatrix}$, where $d_j^{(k)} = \mathrm{vec}(D_j^{(k)})$ for $j = 1, \ldots, p$.

In each step of subspace iteration, we have to solve the equation $\Delta_2 w_j = \Delta_0 z_j^{(k)}$ for $j = 1, \ldots, p$. This can be written as a Sylvester equation as already explained before. (See the text around Eq. (11).) In order to avoid the expensive computations with full rank vectors, we compute a low-rank approximation to the solution of the Sylvester equation. We used the block Arnoldi version of Hu and Reichel from [7], but a rational Krylov method could also be used [3]. The bottomline is that the solution of the Sylvester equation is approximated by a matrix of low rank.

The Arnoldi type method for the Sylvester equation

$$AX - XB = C$$

from [7] works as follows. If X_0 is the initial approximation, we compute the initial residual $R_0 = C - AX_0 + X_0 B$ and take vectors f and g that minimize $\|R_0 - gf^T\|$. Then we build orthogonal bases for Krylov subspaces $\mathcal{K}_k(A, f)$ and $\mathcal{K}_k(B^T, g)$ by the standard Arnoldi process. Let columns of U_k and V_k span $\mathcal{K}_k(A, f)$ and $\mathcal{K}_k(B^T, g)$, respectively, and let $H_A = U_k^H A U_k$ and $H_B = V_k^H B^T V_k$ be the Hessenberg matrices obtained in the Arnoldi process. The approximate solution of rank k is then $X_k = U_k D_k V_k^H$, which satisfies the Galerkin condition

$$U_k^H (C - AX_k - X_k B) V_k = 0. \tag{15}$$

This leads to a small scale projected Sylvester equation

$$H_A D_k - D_k H_B = U_k^H C V_k$$

for matrix D_k.

In our application we have to solve the Sylvester equation

$$A_2^{-1} B_2 W - W B_1^T A_1^{-T} = -A_2^{-1} C_2 Z B_1^T A_1^{-T} + A_2^{-1} B_2 Z C_1^T A_1^{-T}, \tag{16}$$

which is related to system $\Delta_2 w = \Delta_0 z$. If z is an eigenvector of (1), i.e., $\Delta_2 z = \mu \Delta_0 z$, then $z = x_1 \otimes x_2$ is a decomposable tensor. It follows that $Z = x_2 x_1^T$ and the right-hand side of (16) has rank at most 2. In such case, the solution W has rank one and, if we start with $X_0 = 0$, then the algorithm should return $W = \mu^{-1} Z$ after one step. To ensure this we use the block Arnoldi version of [7]. In general, Z is not an eigenvector but an approximation which, in general, has full rank but can usually be well approximated by a matrix of small rank, say, r. In this case, the right-hand side of the Sylvester equation has rank at most $2r$.

Let $x^{(i)} = x_{i1} \otimes x_{i2}$ and $y^{(i)} = y_{i1} \otimes y_{i2}$ for $i = 1, \ldots, m$ be known left and right eigenvectors for the algebraically simple eigenvalues $\mu_1, \ldots, \mu_m$
Choose a matrix $Z_0 \in \mathbb{R}^{n_1 n_2 \times p}$ with orthogonal low-rank columns $z_1^{(0)}, \ldots, z_p^{(0)} \in \operatorname{span}(V_0 \otimes U_0)$ with $U_0 \in \mathbb{R}^{n_2 \times \ell}$ and $V_0 \in \mathbb{R}^{n_1 \times \ell}$.
for $k = 0, 1, \ldots$ **do**
  $F = \begin{bmatrix} A_2^{-1} C_2 U_k & A_2^{-1} B_2 U_k \end{bmatrix}$ and $G = \begin{bmatrix} A_1^{-1} B_1 V_k & A_1^{-1} C_1 V_k \end{bmatrix}$
  Compute orthonormal bases $V_{\text{exp}} \in \mathbb{R}^{n_1 \times r\ell}$ and $U_{\text{exp}} \in \mathbb{R}^{n_2 \times r\ell}$ for $\mathcal{K}_r(A_1^{-1} B_1, G)$ and $\mathcal{K}_r(A_2^{-1} B_2, F)$, respectively, using the block Arnoldi algorithm.
  $H_A = U_{\text{exp}}^T A_2^{-1} B_2^{-1} U_{\text{exp}}$ and $H_B = V_{\text{exp}}^T A_1^{-1} B_1^{-1} V_{\text{exp}}$
  for $j = 1, \ldots, p$ **do**
    $M_j = C_2 S_j B_1^T - B_2 S_j C_1^T$, where $z_j^{(k)} = \operatorname{vec}(S_j)$
    Solve the Sylvester equation $H_A Y_j - Y_j H_B = -U_{\text{exp}}^T A_2^{-1} M_j A_1^{-T} V_{\text{exp}}$.
    $w_j = \operatorname{vec}(U_{\text{exp}} Y_j V_{\text{exp}}^T)$
    $\widetilde{w}_j = w_j - \sum_{i=1}^{m} \frac{{y^{(i)}}^T w_j}{{y^{(i)}}^T \Delta_0 x^{(i)}} x^{(i)}$ (Hotelling's deflation)
  end for
  Compute $R = Z_k^T \widetilde{W}$, where $\widetilde{W} = \begin{bmatrix} \widetilde{w}_1 & \cdots & \widetilde{w}_p \end{bmatrix}$, exploiting the low-rank structures of Z_k and $\widetilde{W}$.
  Compute eigenpairs (σ_i, q_i), $i = 1, \ldots, p$, of matrix R.
  for $j = 1, \ldots, p$ **do**
    if $(\sigma_j, Z_k q_j)$ is an eigenpair of Γ_2^{-1} **then** extract the eigenpair
  end for
  Orthogonalize vectors $\widetilde{w}_1, \ldots, \widetilde{w}_p$.
  Compute matrices $U_{k+1} \in \mathbb{R}^{n_2 \times \ell}$ and $V_{k+1} \in \mathbb{R}^{n_1 \times \ell}$ with orthogonal columns such that $\operatorname{span}(U_{k+1}) \subset \operatorname{span}(U_{\text{exp}})$, $\operatorname{span}(V_{k+1}) \subset \operatorname{span}(V_{\text{exp}})$ and $\widetilde{w}_1, \ldots, \widetilde{w}_p$ can be well approximated in $\operatorname{span}(V_{k+1} \otimes U_{k+1})$.
  $z_i^{(k+1)} = (V_{k+1} \otimes U_{k+1})(V_{k+1} \otimes U_{k+1})^T \widetilde{w}_i$ for $i = 1, \ldots, p$
  Orthogonalize vectors $z_1^{(k+1)}, \ldots, z_p^{(k+1)}$.
end for

Algorithm 2: Subspace iteration with Hotelling's deflation and low-rank vectors for the generalized eigenvalue problem (10) related to the two-parameter eigenvalue problem (5) with real matrices. In the algorithm ℓ denotes the rank of the vectors that we use, $p \leq \ell$ is the size of the subspace for subspace iteration, and r is the number of block Arnoldi steps used to solve the Sylvester equations approximately.

The proposed method is presented in Algorithm 2. From the structure of the eigenvectors, the Δ-matrices from (7) and the low-rank vectors $z_j^{(k)}$, it follows that it is possible

to implement the algorithm without involving any vectors of size $\mathcal{O}(n_1 n_2)$. Here, we give some additional explanations and show that all steps in the algorithm can be performed with matrices of size $n_i \times n_i$ and vectors of length n_i for $i = 1, 2$.

- The formation and solution of the p Sylvester equations in Lines 4–6, and Lines 8–9, that correspond to Line 5 in Algorithm 1, goes as follows. The right-hand side of the system in Line 5 in Algorithm 1 is computed in matrix form. This matrix is assembled from F and G, computed in Line 4, and M_j in Line 8. Because all vectors $z_1^{(k)}, \ldots, z_p^{(k)}$ lie in $\mathrm{span}(V_k \otimes U_k)$, we use the same two Krylov subspaces, spanned by the columns of U_{exp} and V_{exp}, for solving the p Sylvester equations. As approximate solutions we take vectors from $\mathrm{span}(V_{\mathrm{exp}} \otimes U_{\mathrm{exp}})$ that satisfy the Galerkin condition (15).

- In Line 11, we express the vector $\widetilde{w}_j$ after Hotelling's deflation as

$$\widetilde{w}_j = \mathrm{vec}\left(\widetilde{U}_{\mathrm{exp}} \begin{bmatrix} Y_j & & & \\ & \alpha_{j1} & & \\ & & \ddots & \\ & & & \alpha_{jm} \end{bmatrix} \widetilde{V}_{\mathrm{exp}}^T \right), \tag{17}$$

  where $\widetilde{U}_{\mathrm{exp}} = \begin{bmatrix} U_{\mathrm{exp}} & x_{12} & \cdots & x_{m2} \end{bmatrix}$, $\widetilde{V}_{\mathrm{exp}} = \begin{bmatrix} V_{\mathrm{exp}} & x_{11} & \cdots & x_{m1} \end{bmatrix}$, and

$$\alpha_{ji} = -\frac{{y^{(i)}}^T w_j}{{y^{(i)}}^T \Delta_2 x^{(i)}} \tag{18}$$

  for $i = 1, \ldots, m$. Note that (17) has 2×2 blocks on the main diagonal when the deflated eigenpairs are complex conjugate pairs and instead of eigenvectors we then use the real and imaginary parts of the complex conjugate pair to extend matrices $\widetilde{U}_{\mathrm{exp}}$ and $\widetilde{V}_{\mathrm{exp}}$. We compute the denominator in (18) efficiently as

$${y^{(i)}}^T \Delta_0 x^{(i)} = (y_{i1}^T B_1 x_{i1})(y_{i2}^T C_2 x_{i2}) - (y_{i1}^T C_1 x_{i1})(y_{i2}^T B_2 x_{i2}),$$

  and the numerator as

$${y^{(i)}}^T w_j = y_{i2}^T U_{\mathrm{exp}} Y_j V_{\mathrm{exp}}^T y_{i1} - y_{i2}^T U_{\mathrm{exp}} Y_j D V_{\mathrm{exp}}^T y_{i1}.$$

  In the subsequent lines we continue to use the notation $\widetilde{w}_j = \mathrm{vec}(U_{\mathrm{exp}} Y_j V_{\mathrm{exp}}^T)$, where we take $U_{\mathrm{exp}} = \widetilde{U}_{\mathrm{exp}}, V_{\mathrm{exp}} = \widetilde{U}_{\mathrm{exp}}$, and expand Y_j as in (17).

- In Line 13, the inner product of $z_i^{(k)}$ and w_j for $i, j = 1, \ldots, p$ can be formed efficiently as

$${z_i^{(k)}}^T w_j = \mathrm{vec}(D_i^{(k)})^T \mathrm{vec}(U_k^T U_{\mathrm{exp}} Y_j V_{\mathrm{exp}}^T V_k).$$

- In Line 19, a new subspace $\mathrm{span}(V_{k+1} \otimes U_{k+1})$ for low-rank approximations for the next step is determined as follows. First, we consider vector $\widetilde{w}_1 = \mathrm{vec}(U_{\mathrm{exp}} Y_1 V_{\mathrm{exp}}^T)$. Let r_1 be the rank of matrix Y_1 and let $Y_1 = Q \Sigma P$ be its singular value decomposition. We take $U_{k+1} = U_{\mathrm{exp}} Q(:, 1 : r_1)$ and $V_{k+1} = V_{\mathrm{exp}} P(:, 1 : r_1)$ for the first part of matrices U_{k+1} and V_{k+1}. If $r_1 = \ell$, then we have the subspace for the next step, otherwise we continue with vector w_2. We consider the part of w_2 that is not included in the new current subspace $\mathrm{span}(V_{k+1} \otimes U_{k+1})$, i.e., we take $w_2 - (V_{k+1} \otimes U_{k+1})(V_{k+1} \otimes U_{k+1})^T w_2$ and select new vectors using the corresponding singular value decomposition as before. By continuing this process, we eventually obtain ℓ vectors for U_{k+1} and V_{k+1}.

- As k increases, vectors $w_1, \ldots, w_p$ should converge to Schur vectors and matrix R in Line 13 should converge to a partial Schur matrix. In this situation (if we assume that all eigenvalues have different absolute values), vector w_j should have rank j as it is a linear combination of j decomposable eigenvectors for $j = 1, \ldots, p$ and the above procedure would select one vector for U_{k+1} and V_{k+1} from each vector $w_1, \ldots, w_p$.
- We noticed that the algorithm converges very slowly, but is pretty reliable in finding the desired eigenvalues. Therefore, when, in Line 16, an eigenpair of (5) is found with a relatively small residual norm, several steps of the tensor Rayleigh quotient iteration from [12] are performed to improve the quality of the solution even further, which is then finally accepted if the residual after the iterative improvement is small enough.

Let us remark that the approach in Algorithm 2 cannot be applied to a general problem with 2×2 operator determinants of the form (12). Since eigenvectors of (12) are in general full-rank vectors, the low-rank approximation approach does not work.

6 Subspace iteration with projection

If we observe Algorithm 2 closely, then we see that behind the iteration of matrices Z_k with orthogonal columns, there is an iteration on subspaces $\operatorname{span}(V_k \otimes U_k)$ that contain the low-rank vectors from Z_k. So, in one way we are iterating on subspaces, but, we only consider p vectors from the subspace of size ℓ^2. Together with the observation that the subspace $\operatorname{span}(V_{\text{exp}} \otimes U_{\text{exp}})$, which we obtain from the block Arnoldi solver for the Sylvester equation in Algorithm 2, contains many good approximations to the eigenvectors, this leads us to the following algorithm, presented in Algorithm 3.

Similarly to Algorithm 2, we iterate on subspaces $\operatorname{span}(V_k \otimes U_k)$, but the next subspace $\operatorname{span}(V_{k+1} \otimes U_{k+1})$ is chosen from Ritz vectors obtained from the projection of the two-parameter eigenvalue problem on $\operatorname{span}(V_{\text{exp}} \otimes U_{\text{exp}})$. For solving this small two-parameter eigenvalue problem, the algorithm from Section 3 can be used, since we do not have to compute all solutions of the projected problem.

In Line 10, we have to select ℓ Ritz pairs for the next subspace. It seems natural to select the ℓ pairs with the smallest values of $|\tau|$, but this can be dangerous because of possible spurious values. Namely, if we take ℓ eigenvectors $x_{11} \otimes x_{12}, \ldots, x_{\ell 1} \otimes x_{\ell 2}$ and form the subspaces $\mathcal{V} = \operatorname{span}(x_{11}, \ldots, x_{\ell 1})$ and $\mathcal{U} = \operatorname{span}(x_{12}, \ldots, x_{\ell 2})$, then the projected two-parameter eigenvalue problem has ℓ^2 Ritz values, but, in general, only ℓ among them correspond to the eigenvalues of the selected ℓ eigenvectors. So, in particular if we are computing interior eigenvalues, there can be up to $\ell^2 - \ell$ spurious Ritz values with small values of $|\tau|$. We therefore compute and consider $q \geq \ell$ Ritz vectors with smallest $|\tau|$ and then beside the ℓ smallest also take all Ritz vectors with a sufficiently small residual norm. This results in slightly larger subspaces, but improves the convergence. As a rule of thumb, our suggestion is to use subspaces of size $\ell = 2m$ and consider $q = 2\ell = 4m$ Ritz values if we want to compute m eigenvalues. If eigenvalues are close to the exterior, then we can save time and use smaller values of q and ℓ.

As we are not working with individual vectors as in Algorithm 2, Hotelling's deflation can no longer be applied. Therefore, the subspace has to be large enough for all eigenvalues that we would like to compute. In most practical cases this is not an obstacle, but, in some particular examples, where we want to compute many eigenvalues, Algorithm 2 can be more

Start with matrices $U_0 \in \mathbb{C}^{n_2 \times \ell}$, $V_0 \in \mathbb{C}^{n_1 \times \ell}$ with orthogonal columns.
for $k = 0, 1, \ldots$ **do**
  $F = \begin{bmatrix} A_2^{-1}C_2U_k & A_2^{-1}B_2U_k \end{bmatrix}$ and $G = \begin{bmatrix} A_1^{-1}B_1V_k & A_1^{-1}C_1V_k \end{bmatrix}$
  Compute $V_{\exp}$ and $U_{\exp}$ with orthogonal basis for $\mathcal{K}_r(A_1^{-1}B_1, G)$ and $\mathcal{K}_r(A_2^{-1}B_2, F)$, respectively, with a block Arnoldi algorithm.
  Compute Ritz values (σ_j, τ_j) and vectors $V_{\exp}c_j \otimes U_{\exp}d_j$, $j = 1, \ldots, q$, from the projected problem

$$\begin{aligned} V_{\exp}^H A_1 V_{\exp} c &= \sigma V_{\exp}^H B_1 V_{\exp} + \tau V_{\exp}^H C_1 V_{\exp} \\ U_{\exp}^H A_2 U_{\exp} d &= \sigma U_{\exp}^H B_2 U_{\exp} + \tau U_{\exp}^H C_2 U_{\exp} \end{aligned}$$

  with smallest $|\tau|$.
  For each Ritz pair compute the corresponding residuals and test convergence, i.e., if $\|(A_1 - \sigma_k B_1 - \tau_k C_1)V_{\exp}c_k\| < \epsilon$ and $\|(A_2 - \sigma_k B_2 - \tau_k C_2)U_{\exp}d_k\| < \epsilon$.
  if m Ritz pairs have converged **then**
    Extract the corresponding eigenpairs.
  else
    Select ℓ Ritz vectors $V_{\exp}c_j \otimes U_{\exp}d_j$ for $j = 1, \ldots, \ell$.
    Compute $V_{k+1} \in \mathbb{C}^{n_1 \times \ell}$ and $U_{k+1} \in \mathbb{C}^{n_2 \times \ell}$ with orthonormal columns such that $\operatorname{span}(V_{k+1}) = \operatorname{span}(V_{\exp}c_1, \ldots, V_{\exp}c_\ell)$ and $\operatorname{span}(U_{k+1}) = \operatorname{span}(U_{\exp}d_1, \ldots, V_{\exp}d_\ell)$.
  end if
end for

Algorithm 3: Subspace iteration with Arnoldi expansion and restart based on selected Ritz vectors for the generalized eigenvalue problem (10) related to the two-parameter eigenvalue problem (5). In the algorithm m denotes the number of wanted eigenvalues, $\ell \geq m$ is the size of the subspaces, $q \geq \ell$ is the number of Ritz vectors that we compute, and r is the number of block Arnoldi steps used to solve the Sylvester equations approximately.

efficient than Algorithm 3. Finally, note that Rayleigh Quotient refinement is not required in this algorithm, as convergence is fast.

7 Numerical results

The following numerical examples were obtained on 64-bit Windows version of Matlab R2012b running on Intel 8700 processor and 8 GB of RAM.

Example 4 In the first example we compute several hundred eigenvalues with the smallest $|\mu|$ of Mathieu's system (4) that corresponds to the problem of computing the eigenfrequencies of an elliptical membrane with a fixed boundary. The differential equations are discretized by Chebyshev collocation, for more details, see [4]. The problems in this example are small enough for using the implicitly restarted Arnoldi method directly on (2). The Matlab implementation `EigElip` from [4] uses `eigs` applied to a sparse representation of matrices Δ_0 and Δ_2. We first compare this code to the new algorithm using the Bartels-Stewart algorithm as proposed in Section 3 and in Appendix (with the only difference that we use the real Schur form because all matrices are real). See columns 5 and 6 in Table 1. For a comparison, we include the times required by another available Matlab implementation `runelip` [19]

for eigenfrequencies of an elliptical membrane, which uses expansions into Bessel functions to solve Mathieu's system (for details, see [20]). One can see in Table 1 that the speedup obtained by using the Sylvester equation relation is significant.

			`EigElip`				`runelip`	
α	β	m	(n_1, n_2)	new time	time	error	time	error
2	1	100	(54,25)	0.8	2.5	3e-11	14.8	3e-11
2	1	300	(80,36)	7.7	23.2	3e-11	37.7	6e-11
2	1	500	(93,45)	27.5	78.4	3e-11	70.0	3e-01
4	1	100	(68,24)	1.0	3.5	3e-11	12.5	2e-11
4	1	200	(86,26)	3.2	10.2	5e-11	22.1	3e-11
4	1	250	(94,28)	5.7	16.1	3e-11	29.0	3e-06
4	1	300	(100,30)	8.2	24.8	5e-11	36.6	3e-03
8	1	100	(84,18)	1.0	3.1	2e-11	11.1	2e-11
8	1	125	(94,20)	1.7	5.3	2e-11	17.2	3e-05
8	1	150	(100,20)	2.0	6.9	1e-11	19.4	1e-02

Table 1: Computational times and errors for computing m lowest eigenfrequencies of an ellipse with major radius α and minor radius β using `EigElip` (old and new implementation) and `runelip`. Column (n_1, n_2) contains the sizes of the matrices used in `EigElip`.

Example 5 In the second example we take a two-parameter eigenvalue problem with matrices of size $n_1 = n_2 = 500$ that correspond to a discretization of equation (4) for the ellipse with $\alpha = 2$ and $\beta = 1$ by Chebyshev collocation. This is the same problem as in Example 4, but now with matrices of larger sizes so that the methods that use low-rank vectors can show their potential.

As a reference, note that `eigs` combined with the Bartels-Stewart algorithm now requires 59.7s to compute the 10 eigenvalues with smallest $|\mu|$. We can do better with the low-rank methods and in Table 2 we compare results obtained by implementations of Algorithms 2 and 3. As we are computing exterior eigenvalues, we set $p = 10$ and $\ell = 11$ for Algorithm 2 and $q = \ell = m + 1$, where m is the number of wanted eigenvalues, for Algorithm 3. These values are hand picked from several numerical experiments. In Algorithm 2, we perform two steps of block Arnoldi, while in Algorithm 3, we need to use more Arnoldi steps (column r in the table) if we want to compute more eigenvalues. A large number of steps of block Arnoldi increases the subspace size and, therefore, the computation time as well. For both methods we use an absolute stopping criterion $\epsilon = 10^{-6}$ for the norm of the residual.

As expected, Algorithm 2 gives better results if we want to compute many eigenvalues, while Algorithm 3 is better for a small number of eigenvalues. Let us also remark that all results for Algorithm 2 were obtained in one run, where in the end, 200 eigenvalues were computed in 684 steps and 162.1s. For Algorithm 3, the subspace dimension was adjusted and the algorithm reran for each number of wanted eigenvalues. This explains why Algorithm 3 computed 100 eigenvalues faster than 90 eigenvalues. As can be seen in the table, it required 4 steps for 100 eigenvalues and 5 steps for 90 eigenvalues where a smaller subspace is used.

Example 6 Lamé's system is another example of two-parameter eigenvalue problems that appears when separation of variables is applied to a separable boundary-value problem. We

eigenvalues	Algorithm 2		Algorithm 3		
m	time	steps	time	steps	r
10	2.6	20	0.8	5	2
20	4.7	34	1.8	5	2
30	7.5	55	3.6	5	2
40	10.5	76	7.7	6	2
50	13.3	92	9.6	5	2
60	17.9	128	13.2	5	2
70	21.9	156	33.8	5	3
80	27.0	191	50.3	5	3
90	30.4	209	59.3	5	3
100	34.4	231	50.4	4	3
110	40.3	265	180.0	4	4
120	46.6	301	214.0	4	4

Table 2: Computational times for computing the m lowest eigenfrequencies of an ellipse with $\alpha = 2$ and $\beta = 1$ using Chebyshev collocation discretization with $n_1 = n_2 = 500$ and Algorithms 2 and 3.

consider the trigonometric form of Lamé's system [11]

$$\begin{aligned}(1-k^2\cos^2\varphi)L''(\varphi)+k^2(\sin\varphi)(\cos\varphi)L'(\varphi)+(k^2\rho(\rho+1)\sin^2\varphi+\delta]L(\varphi) &= 0\\ (1-k'^2\cos^2\theta)N''(\theta)+k'^2(\sin\theta)(\cos\theta)N'(\theta)+(k'^2\rho(\rho+1)\sin^2\theta-\delta]N(\theta) &= 0,\end{aligned} \tag{19}$$

where $\varphi, \theta \in [0, \pi]$, $k, k' \in (0, 1)$, $k^2 + k'^2 = 1$, and δ is a separation constant. Solution $N(\theta)$ is either odd or even, so it suffices to consider $\theta \in [0, \pi/2]$. The boundary conditions, related to the problem of computing the strength of a charge singularity of a flat plate for corner angle $0 < \chi < \pi$ in [11], are $L(0) = L'(\pi) = 0$ and $N'(0) = N'(\pi/2) = 0$. The goal is to compute the smallest $\rho > 0$ for $k = \sin((\pi - \chi)/2)$.

We set $\lambda = \delta$, $\mu = \rho(\rho + 1)$, and discretize the equations (19) using standard finite differences. As finite differences do not converge as fast as Chebyshev collocation in Example 5, much larger matrices have to be used for accurate results. In the discretized problem

$$\begin{aligned}A_1x_1 &= \lambda B_1x_1 + \mu C_1x_1\\ A_2x_2 &= \lambda B_2x_2 + \mu C_2x_2\end{aligned} \tag{20}$$

matrices A_1 and A_2 are tridiagonal, matrices C_1 and C_2 are diagonal, and $-B_1$ and B_2 are identity matrices. All matrices can thus be efficiently represented in Matlab.

Due to the boundary conditions, matrices A_2, C_1, and C_2 are singular and the corresponding Δ-matrices Δ_0 and Δ_1 are singular as well. However, the most important point is that Δ_2 is nonsingular. Following Lemma 1, a shift σ exists so that $A_1 - \sigma B_1$ and $A_2 - \sigma B_2$ are non-singular. In this case, we use $\sigma = -10$. We now numerically solve the shifted system

$$\begin{aligned}(A_1 + 10B_1)x_1 &= \widetilde{\lambda}B_1x_1 + \mu C_1x_1\\ (A_2 + 10B_2)x_2 &= \widetilde{\lambda}B_2x_2 + \mu C_2x_2.\end{aligned} \tag{21}$$

We discretize (19) to (21) with matrices of size 40000×40000, which are clearly too large for methods from Section 3 that use full vectors. This does, however, not pose any problem

for Algorithms 2 and 3. The settings were $p = 5$ and $\ell = 10$ for Algorithm 2 and $\ell = 4$ and $q = 8$ for Algorithm 3. In both algorithms, we apply two steps of block Arnoldi and use $\epsilon = 10^{-6}$ for the absolute stopping criteria. For $\chi = \pi/2$ we obtain the smallest three eigenvalues

$$\begin{aligned} \mu_1 &= 0.3845467 \\ \mu_2 &= 3.4614507 \\ \mu_3 &= 6.1994403 \end{aligned}$$

in 1.4s with Algorithm 3 and in 20.2s with Algorithm 2. The smallest eigenvalue μ_1 gives $\rho_1 = 0.2965844$, which agrees perfectly with the results in Table 1 in [11].

8 Conclusions

We presented new numerical methods for two-parameter eigenvalue problems. The first method uses the implicitly restarted Arnoldi or Krylov–Schur method and is very efficient for problems of moderate size. The same approach can be applied to any generalized eigenvalue problem of the form (12), where matrices are 2×2 operator determinants.

The other two methods use low-rank vectors and solve the related Sylvester equation only approximately. The method with subspace iteration and Hotelling's deflation is recommended when we want to compute many eigenvalues, while the method that iterates on a subspace based on Ritz vectors is more efficient for a small number of eigenvalues.

Acknowledgements

This research was supported in part by ARRS and FWO in the bilateral project BI-BE/11-12-F-011 between Slovenia and Flanders. The work by K. Meerbergen is supported by the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State Science Policy Office, and the KU Leuven Research Council grants PFV/10/002, OT/10/038 and OT/14/074. The research was performed in part while the second author was visiting the CASA group at the TU Eindhoven. The author wishes to thank the NWO for the visitor grant and the CASA group for its hospitality.

References

[1] F. V. Atkinson, *Multiparameter Eigenvalue Problems*, Academic Press, New York, 1972.

[2] R. H. Bartels and G. W. Stewart, *Algorithm 432: Solution of matrix equation $AX + XB = C$*, Comm. ACM, 15 (1972), pp. 820–826.

[3] B. Beckermann, *An error analysis for rational Galerkin projection applied to the Sylvester equation*, SIAM J. Numer. Anal., 49 (2011), pp. 2430–2450.

[4] C. I. Gheorghiu, M. E. Hochstenbach, B. Plestenjak, and J. Rommes, *Spectral collocation solutions to multiparameter Mathieu's system*, Appl. Math. Comp., 218 (2012), pp. 11990–12000.

[5] M. E. Hochstenbach, T. Košir, and B. Plestenjak, *A Jacobi–Davidson type method for the nonsingular two-parameter eigenvalue problem*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 477–497.

[6] M. E. Hochstenbach and B. Plestenjak, *Harmonic Rayleigh-Ritz for the multiparameter eigenvalue problem*, Electron. Trans. Numer. Anal., 29 (2008), pp. 81–96.

[7] D. Y. Hu and L. Reichel, *Krylov-subspace methods for the Sylvester equation*, Linear Algebra Appl., 172 (1992), pp. 283–313.

[8] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.

[9] B. C. Lesieutre, A. V. Mamishev, Y. Du, E. Keskiner, M. Zahn, and G. C. Verghese, *Forward and inverse parameter estimation algorithms of interdigital dielectrometry sensors*, IEEE T. Dielect. El. In., 8 (2001), pp. 577–588.

[10] K. Meerbergen and A. Spence, *Shift-and-invert iteration for purely imaginary eigenvalues with application to the detection of Hopf bifurcations in large scale problems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1463–1482.

[11] J. A. Morrison and J. A. Lewis, *Charge singularity at the corner of a flat plate*, SIAM J. Appl. Math., 31 (1976), pp. 233–250.

[12] B. Plestenjak, *A continuation method for a right definite two-parameter eigenvalue problem*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1163–1184.

[13] Y. Saad, *Numerical Methods for Large Eigenvalue Problems. Revised Edition*, SIAM, Philadelphia, 2011.

[14] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[15] G. W. Stewart, *A Krylov-Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601–614.

[16] H. Volkmer, *On the minimal eigenvalue of a positive definite operator determinant*, Proc. Roy. Soc. Edinburgh Sect. A, 103 (1986), pp. 201–208.

[17] H. Volkmer, *Multiparameter Problems and Expansion Theorems*, Lecture Notes in Math. 1356, Springer-Verlag, New York, 1988.

[18] M. Willatzen and L. C. Lew Yan Voon, *Separable Boundary-Value Problems in Physics*, Willey-VCH, Weinheim, 2011.

[19] H. B. Wilson, *Vibration modes of an elliptic membrane*. Available from http://www.mathworks.com/matlabcentral/fileexchange. MATLAB File Exchange, The MathWorks, Natick, 2004.

[20] H. B. Wilson and R. W. Scharstein, *Computing elliptic membrane high frequencies by Mathieu and Galerkin methods*, J. Eng. Math., 57 (2007), pp. 41–55.

A Matlab implementation

We give some details on the implementation of the Sylvester equation approach from Section 3 in Matlab. Suppose that we would like to compute k eigenvalues (λ, μ) of a two-parameter eigenvalue problem (5) with smallest $|\mu|$. We can first compute matrices Δ_2 and Δ_0 in (7) as

```
Delta2 = kron(A1,B2) - kron(B1,A2)
Delta0 = kron(B1,C2) - kron(C1,B2)
```

and then apply `eigs` as

```
mu = eigs(Delta2,Delta0,k,'SM')
```

We work with matrices of dimension $n_1 n_2 \times n_1 n_2$ and the complexity of the above approach is $\mathcal{O}(n_1^3 n_2^3)$. If we apply the Sylvester equation relation, then we reduce the complexity to $\mathcal{O}(n_1^3 + n_2^3)$. We just have to force `eigs` to use a Sylvester equation solver to multiply by matrix $\Delta_0^{-1}\Delta_2$. This is implemented in the following function `MEPeigs` that returns k eigenvalues with smallest $|\mu|$ of a generalized eigenvalue problem $\Delta_2 z = \mu \Delta_0 z$ (we assume that matrices A_1 and A_2 are nonsingular).

```
function mu = MEPeigs(A1,B1,C1,A2,B2,C2,k)
    n1 = size(A1,1); n2 = size(A2,1);
    [U1,R1] = schur(A2\B2,'complex');
    [U2,R2] = schur(-transpose(B1)/transpose(A1),'complex');
    opts.isreal = false;
    mu = eigs(@multGamma,n1*n2,k,'SM',opts);
    function y = multGamma(x)
        W = reshape(x,n2,n1);
        F = C2*W*transpose(B1) - B2*W*transpose(C1);
        y = reshape(SylvBSUT(U1,R1,U2,R2,-A2\F/transpose(A1)),n1*n2,1);
    end
end
```

In the above, `SylvBSUT` is an auxiliary function, such as, for instance, function `lyap` from *Control Toolbox*, that solves the Sylvester equation $AX + XB = C$, where $A = QR$ and $B = SU$, Q and U are unitary matrices, and R and S are upper triangular matrices. An alternative Matlab implementation is as follows.

```
function X = SylvBSUT(Q,R,U,S,C)
    m = size(R,1); n = size(S,1);
    X = zeros(m,n);
    F = Q'*C*U;
    X(:,1) = (R + S(1,1)*eye(m))\F(:,1);
    for k = 2:n
        X(:,k) = (R + S(k,k)*eye(m))\(F(:,k) - X(:,1:k-1)*S(1:k-1,k));
    end
    X = Q*X*U';
end
```