SIAM J. MATRIX ANAL. APPL.
Vol. 36, No. 2, pp. 820–838

COMPACT RATIONAL KRYLOV METHODS FOR NONLINEAR EIGENVALUE PROBLEMS*

ROEL VAN BEEUMEN†, KARL MEERBERGEN†, AND WIM MICHIELS†

Abstract. We propose a new uniform framework of compact rational Krylov (CORK) methods for solving large-scale nonlinear eigenvalue problems $A(\lambda)x = 0$. For many years, linearizations were used for solving polynomial and rational eigenvalue problems. On the other hand, for the general nonlinear case, $A(\lambda)$ can first be approximated by a (rational) matrix polynomial and then a convenient linearization is used. However, the major disadvantage of linearization-based methods is the growing memory and orthogonalization costs with the iteration count, i.e., in general they are proportional to the degree of the polynomial. Therefore, the CORK family of rational Krylov methods exploits the structure of the linearization pencils by using a generalization of the compact Arnoldi decomposition. In this way, the extra memory and orthogonalization costs due to the linearization of the original eigenvalue problem are negligible for large-scale problems. Furthermore, we prove that each CORK step breaks down into an orthogonalization step of the original problem dimension and a rational Krylov step on small matrices. We also briefly discuss implicit restarting of the CORK method and how to exploit low rank structure. The CORK method is illustrated with two large-scale examples.

Key words. linearization, matrix pencil, rational Krylov, nonlinear eigenvalue problem

AMS subject classifications. 65F15, 15A22

DOI. 10.1137/140976698

1. Introduction.

We present a new framework of compact rational Krylov (CORK) methods for solving the nonlinear eigenvalue problem (NLEP)

$$A(\lambda)x = 0, \tag{1.1}$$

where $\lambda \in \Omega \subseteq \mathbb{C}$ and $A : \Omega \to \mathbb{C}^{n\times n}$ is analytic on Ω. We call λ an eigenvalue and $x \in \mathbb{C}^n \setminus \{0\}$ the corresponding eigenvector. Linearizations have been for many years the classical and most widely used approach to solving polynomial eigenvalue problems [8, 15, 1]. The matrix polynomial $P(\lambda) = \sum_{i=0}^{d} \lambda^i P_i$ with $P_i \in \mathbb{C}^{n\times n}$, is transformed into a linear pencil $\mathbf{L}(\lambda) = \mathbf{X} - \lambda\mathbf{Y}$, with $\mathbf{X}, \mathbf{Y} \in \mathbb{C}^{dn\times dn}$, so that there is a one-to-one correspondence between the eigenvalues of $P(\lambda)x = 0$ and $\mathbf{L}(\lambda)\mathbf{x} = 0$.

For the general nonlinear case, i.e., nonpolynomial eigenvalue problem, $A(\lambda)$ is first approximated by a matrix polynomial [6, 11, 21] or rational matrix polynomial [9, 17] before a convenient linearization is applied. Most linearizations used in the literature can be written in a similar form, i.e., $\mathbf{L}(\lambda) = \mathbf{A} - \lambda\mathbf{B}$, where the parts below the first block rows of $\mathbf{A}$ and $\mathbf{B}$ have the Kronecker structures $M \otimes I_n$ and $N \otimes I_n$, respectively. Note that the pencil $(\mathbf{A}, \mathbf{B})$ also covers the dynamically growing linearization pencils used in [10, 11, 21, 9]. The construction of the polynomial or rational approximation of $A(\lambda)$ can be obtained using results on approximation

*Received by the editors July 9, 2014; accepted for publication (in revised form) March 31, 2015; published electronically June 18, 2015. The research of the authors was supported by the Programme of Interuniversity Attraction Poles of the Belgian Federal Science Policy Office (IAP P6-DYSCO), by OPTEC, the Optimization in Engineering Center of the KU Leuven, by the projects STRT1-09/33, OT/10/038, OT/14/074 of the KU Leuven Research Council, and the project G.0712.11N of the Research Foundation-Flanders (FWO).

http://www.siam.org/journals/simax/36-2/97669.html

†Department of Computer Science, KU Leuven, University of Leuven, 3001 Leuven, Belgium (Roel.VanBeeumen@kuleuven.be, Karl.Meerbergen@kuleuven.be, Wim.Michiels@kuleuven.be).

theory or can be constructed dynamically during the solution process [21, 9]. Also note that Fiedler linearizations do, in general, not satisfy this structure [7, 5]. Next to the classical companion linearizations, this structure can be found in linearizations for degree-graded polynomials, e.g., Newton and Chebyshev polynomials [1], for Lagrange polynomials [22], and also for rational approximations such as rational Newton polynomials [9], and the spectral discretization [10].

The major difficulty of (rational) Krylov methods using linearizations for large n is the growing memory and orthogonalization costs with the iteration count. In general, they are proportional to the degree of the polynomial. That is, at iteration j, the storage cost of the iteration vectors is of order $d \cdot j$ vectors of size n and the orthogonalization cost is of order $d \cdot j^2$ scalar products of size n. However, we can exploit the Kronecker structure, mentioned above, such that the memory cost is only of order $d+j$ vectors of size n and the orthogonalization cost is of order $(d+j)j$ scalar products of size n.

The CORK family of rational Krylov methods, presented in this paper, use a generalization of the compact Arnoldi decomposition, proposed in [20]. This family of methods constructs a subspace $\mathbf{V} \in \mathbb{C}^{dn \times j}$, represented in factored form

$$\mathbf{V} = (I_d \otimes Q)\mathbf{U},$$

where $Q \in \mathbb{C}^{n \times r}$ and $\mathbf{U} \in \mathbb{C}^{dr \times j}$ are matrices with orthonormal columns. The rank r is bounded from above by $d+j$, which is typically much lower than $d \cdot j$. Note that the idea of a compact representation of Arnoldi vectors was presented for a Chebyshev basis in [12, 23].

We present in this paper a generic but simple framework for solving eigenvalue problems represented by a linearization satisfying the Kronecker structure mentioned earlier. This includes locking, purging, and implicit restarting. We observe that, after implicit restarting, r is often smaller than $d + j$, which reduces the storage and orthogonalization costs even further. This interesting observation appears to be useful for the dynamical approaches where the degree d is not determined before the start of the algorithm. In addition, we show how to exploit low rank matrices in the polynomial expansion of $A(\lambda)$ in combination with small r.

This paper is organized as follows. Section 2 introduces a uniform framework for representing linearization pencils. Section 3 reviews the standard rational Krylov method for the generalized eigenvalue problem. Section 4 proposes the compact rational Krylov decomposition which is used to introduce the compact rational Krylov method in section 5. Section 6 discusses implicit restarting of the CORK method. Section 7 shows how to exploit low rank matrices. Section 8 illustrates the proposed CORK method with two large-scale numerical examples. Finally, the main conclusions are summarized in Section 9.

Throughout the paper, we denote vectors by lowercase Roman characters and matrices by capital Roman characters, e.g., v and A. For block vectors and block matrices we use $\mathbf{v}$ and $\mathbf{A}$, respectively, and a superscript as in $v^{[i]}$ denotes the ith block of the block vector $\mathbf{v}$. The conjugate transpose of a matrix A is denoted by A^*. $I_{i\times j}$ is the identity matrix of dimensions $i \times j$ and in the case $i = j$, we use I_i. V_j denotes a matrix with j columns and the ith column is denoted by v_i. We omit subscripts when the dimensions of the matrices are clear from the context. If not stated otherwise, we denote with $\|\cdot\|$ the 2-norm.

2. Approximation and linearization. Most methods for solving the NLEP consist of two steps. First, the matrix-valued function $A(\lambda)$ in (1.1) is approximated

by a function $P(\lambda)$, which is often a polynomial or a rational function. Second, linearization is used to transform the eigenvalue problem $P(\lambda)x = 0$ into a linear pencil with the same eigenvalues [8, 15, 1].

The starting point of this paper is a function $P(\lambda)$ having the structure defined by Definition 2.1. As we shall see, many polynomial and rational approximations of $A(\lambda)$ can be written in this form. We start with the following definitions.

DEFINITION 2.1. *Let $P(\lambda)$ with $P : \mathbb{C} \to \mathbb{C}^{n\times n}$ be defined as follows:*

$$P(\lambda) := \sum_{i=0}^{d-1} (A_i - \lambda B_i) f_i(\lambda), \tag{2.1}$$

where $A_i, B_i \in \mathbb{C}^{n\times n}$, and f_i are scalar functions of λ. We assume that $P(\lambda)$ is regular, i.e., $\det P(\lambda)$ does not vanish identically and that there exists a linear relation between the functions f_i,

$$(M - \lambda N) f(\lambda) = 0 \tag{2.2}$$

with $M, N \in \mathbb{C}^{(d-1)\times d}$, and $f(\lambda) := \begin{bmatrix} f_0(\lambda) & f_1(\lambda) & \cdots & f_{d-1}(\lambda) \end{bmatrix}^T \neq 0$ for all λ. We also assume that the matrix $M - \lambda N$ is of rank $d-1$ for all λ.

DEFINITION 2.2 (structured pencil). *Let $P(\lambda)$, A_i, B_i, f_i, M, and N be defined by Definition 2.1. Then, we define the $dn \times dn$ linear pencil $\mathbf{L}(\lambda)$ as follows:*

$$\mathbf{L}(\lambda) = \mathbf{A} - \lambda \mathbf{B}, \tag{2.3}$$

where

$$\mathbf{A} = \left[\begin{array}{ccccc} A_0 & A_1 & \cdots & A_{d-1} \\ \hline & & M \otimes I_n & & \end{array}\right], \qquad \mathbf{B} = \left[\begin{array}{ccccc} B_0 & B_1 & \cdots & B_{d-1} \\ \hline & & N \otimes I_n & & \end{array}\right]. \tag{2.4}$$

Note that Definition 2.2 covers many of the linearizations used in the literature. Table 1 gives an overview of polynomial bases such as the monomial basis [15], the orthogonal bases including the Chebyshev basis [1], the Lagrange basis [22], and the Newton basis [1]. Also linear rational bases are covered, e.g., the rational monomial basis [17], the rational Newton basis [9], etc. Moreover, the spectral discretization used in [10] also fits in the pencil (2.3)–(2.4).

The Kronecker structure of the linearization matrices $\mathbf{A}$ and $\mathbf{B}$, defined in (2.4), can be exploited for efficiently solving linear systems originating from the shift-and-invert step in Krylov methods. Therefore, we introduce the following block ULP factorization.

THEOREM 2.3 (block ULP decomposition). *Let $\mathbf{A}$ and $\mathbf{B}$ be defined by (2.4). Then, for every $\mu \in \mathbb{C}$ there exists a permutation matrix $\mathcal{P} \in \mathbb{C}^{d\times d}$ such that the matrix $(M_1 - \mu N_1) \in \mathbb{C}^{(d-1)\times(d-1)}$ is invertible with*

$$M =: \begin{bmatrix} m_0 & M_1 \end{bmatrix} \mathcal{P}, \qquad N =: \begin{bmatrix} n_0 & N_1 \end{bmatrix} \mathcal{P}.$$

Moreover, the pencil $\mathbf{L}(\mu)$ can be factorized as follows:

$$\mathbf{L}(\mu) = \mathbf{A} - \mu \mathbf{B} = \mathcal{U}\mathcal{L}(\mathcal{P} \otimes I_n),$$

where

$$\mathcal{L} = \begin{bmatrix} P(\mu) & 0 \\ (m_0 - \mu n_0) \otimes I_n & (M_1 - \mu N_1) \otimes I_n \end{bmatrix},$$
$$\mathcal{U} = \begin{bmatrix} \alpha^{-1} I_n & (\bar{\mathbf{A}}_1 - \mu \bar{\mathbf{B}}_1)\left((M_1 - \mu N_1)^{-1} \otimes I_n\right) \\ 0 & I_{(d-1)n} \end{bmatrix}$$

TABLE 1
Transformation of matrix polynomials of degree d into the form of Definition 2.1.

(a) Matrix polynomials of degree d

Basis	$P(\lambda)$	Basis functions
Monomial	$\sum_{i=0}^{d} P_i\lambda^i$	λ^i
Orthogonal	$\sum_{i=0}^{d} C_i p_i(\lambda)$	$\lambda p_i(\lambda) = \alpha_i p_{i+1}(\lambda) + \beta_i p_i(\lambda) + \gamma_i p_{i-1}(\lambda)$, with $\alpha_i \neq 0$, $\gamma_i > 0$
Newton	$\sum_{i=0}^{d} D_i n_i(\lambda)$	$n_0(\lambda) := 1$, $n_i(\lambda) := \prod_{k=0}^{i-1}(\lambda - \sigma_k)$ for $i > 0$
Lagrange	$\sum_{i=0}^{d} F_i \ell_i(\lambda)$	$\ell_i(\lambda) := \ell(\lambda)\frac{w_i}{\lambda - \sigma_i}$, with $\ell(\lambda) = (\lambda-\sigma_0)(\lambda-\sigma_1)\cdots(\lambda-\lambda_d)$

(b) A_i and B_i for matrix polynomials of degree d in the form of (2.1)

Basis	A_i	B_i
Monomial	$P_i \quad i = 0, 1, \ldots, d-1$	$\begin{cases} 0 & i < d-1 \\ -P_d & i = d-1 \end{cases}$
Orthogonal	$\begin{cases} C_i & i < d-2 \\ C_{d-2} - \frac{\gamma_{d-1}}{\alpha_{d-1}} C_d & i = d-2 \\ C_{d-1} - \frac{\beta_{d-1}}{\alpha_{d-1}} C_d & i = d-1 \end{cases}$	$\begin{cases} 0 & i < d-1 \\ -\frac{1}{\alpha_{d-1}} C_d & i = d-1 \end{cases}$
Newton	$\begin{cases} D_i & i < d-1 \\ D_{d-1} - \sigma_{d-1} D_d & i = d-1 \end{cases}$	$\begin{cases} 0 & i < d-1 \\ -D_d & i = d-1 \end{cases}$
Lagrange	$\begin{cases} \sigma_{i+1} F_i & i < d-1 \\ \sigma_d F_{d-1} + \sigma_{d-1} \frac{w_d}{w_{d-1}} F_d & i = d-1 \end{cases}$	$\begin{cases} F_i & i < d-1 \\ F_{d-1} + \frac{w_d}{w_{d-1}} F_d & i = d-1 \end{cases}$

(c) f_i and its linear relations for matrix polynomials of degree d in the form of (2.1)

Basis	$f_i(\lambda)$	Linear relations
Monomial	λ^i	$f_{i+1}(\lambda) = \lambda f_i(\lambda)$
Orthogonal	$p_i(\lambda)$	$\alpha_i f_{i+1}(\lambda) = (\lambda - \beta_i) f_i(\lambda) - \gamma_i f_{i-1}(\lambda)$
Newton	$n_i(\lambda)$	$f_{i+1}(\lambda) = (\lambda - \sigma_i) f_i(\lambda)$
Lagrange	$-\ell_i(\lambda)/(\lambda - \sigma_{i+1})$	$w_i(\lambda - \sigma_{i+2}) f_{i+1}(\lambda) = w_{i+1}(\lambda - \sigma_i) f_i(\lambda)$

with the scalar $\alpha = e_1^T \mathcal{P} f(\mu) \neq 0$ and

$$\begin{bmatrix} A_0 & A_1 & \cdots & A_{d-1} \end{bmatrix} =: \begin{bmatrix} \bar{A}_0 & \bar{\mathbf{A}}_1 \end{bmatrix} (\mathcal{P} \otimes I_n),$$
$$\begin{bmatrix} B_0 & B_1 & \cdots & B_{d-1} \end{bmatrix} =: \begin{bmatrix} \bar{B}_0 & \bar{\mathbf{B}}_1 \end{bmatrix} (\mathcal{P} \otimes I_n).$$

Proof. First, since $\operatorname{rank}(M - \mu N) = d - 1$ for all μ by Definition 2.1, we can always find a permutation matrix $\mathcal{P}$ such that $M_1 - \mu N_1$ is invertible. Next, except for the top left block, all blocks of $\mathbf{L}(\mu)(\mathcal{P}^T \otimes I_n) = \mathcal{U}\mathcal{L}$ follow immediately from Definition 2.2. Thus, we only need to prove that

$$\bar{A}_0 - \mu \bar{B}_0 = P(\mu)/\alpha + (\bar{\mathbf{A}}_1 - \mu \bar{\mathbf{B}}_1)\left((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0) \otimes I_n\right). \tag{2.5}$$

From Definition 2.2, we have that $\exists g \in \mathbb{C}^d \setminus \{0\} : (M - \lambda N)g = 0$. By using $\mathcal{P}\mathcal{P}^T = \mathcal{P}^T\mathcal{P} = I_d$ and multiplying from the left with $(M_1 - \mu N_1)^{-1}$ yields

$$\begin{bmatrix}(M_1 - \mu N_1)^{-1}(m_0 - \mu n_0) & I_{d-1}\end{bmatrix} \mathcal{P}g = 0.$$

Next, solving this linear system results in

$$\mathcal{P}g = \begin{bmatrix} 1 \\ -(M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)\end{bmatrix}, \tag{2.6}$$

where g is only defined up to a scalar. Hence, using (2.2) and (2.6) we obtain

$$\mathcal{P}f(\mu) = \alpha \begin{bmatrix} 1 \\ -(M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)\end{bmatrix} \tag{2.7}$$

with $\alpha = e_1^T \mathcal{P} f(\mu)$. By definition we have the identity

$$P(\lambda) = (e_1^T \otimes I_n) \cdot (\mathbf{A} - \lambda \mathbf{B}) \cdot (f(\lambda) \otimes I_n). \tag{2.8}$$

Now, substituting (2.7) into (2.8) proves the equality in (2.5). ∎

The eigenpair connections between $P(\lambda)$ and $\mathbf{L}(\lambda)$ follow now directly from the block ULP decomposition of $\mathbf{L}(\lambda)$.

COROLLARY 2.4 (structured eigenvectors). *Let $P(\lambda)$ and $\mathbf{L}(\lambda)$ be defined by Definitions 2.1 and 2.2, respectively.*

1. *If $P(\lambda)$ is regular, then $\mathbf{L}(\lambda)$ is also regular.*
2. *If the pair $(\lambda_\star, x)$ is an eigenpair of $P(\lambda)$, then the pair $(\lambda_\star, f(\lambda_\star) \otimes x)$ is an eigenpair of $\mathbf{L}(\lambda)$.*
3. *If the pair $(\lambda_\star, \mathbf{y})$ is an eigenpair of $\mathbf{L}(\lambda)$, then there exists a vector x such that $\mathbf{y} = f(\lambda_\star) \otimes x$ and the pair $(\lambda_\star, x)$ is an eigenpair of $P(\lambda)$.*

Proof. Part 1 of the proof follows immediately from the block ULP decomposition of $\mathbf{L}(\lambda)$. Next, we consider the following identity

$$(\mathbf{A} - \lambda \mathbf{B}) \cdot (f(\lambda) \otimes I_n) = e_1 \otimes P(\lambda). \tag{2.9}$$

Then, the proof of part 2 follows immediately from taking $\lambda = \lambda_\star$ and multiplying (2.9) from the right with $1 \otimes x$. For the proof of part 3 we start with

$$\mathbf{L}(\lambda_\star)\mathbf{y} = (\mathbf{A} - \lambda_\star \mathbf{B})\mathbf{y} = 0.$$

Next, from the second through the last block row we find that

$$((M - \lambda_\star N) \otimes I_n)\mathbf{y} = 0.$$

By choosing $x = (e_1^T \mathcal{P} \otimes I_n)\mathbf{y}/\alpha$, with $\alpha = e_1^T \mathcal{P} f(\lambda_\star) \neq 0$, and using (2.7) we obtain $\mathbf{y} = f(\lambda_\star) \otimes x$. Again evaluating (2.9) at $\lambda_\star$ and multiplying from the right with $1 \otimes x$ completes the proof. ∎

Remark 2.5. For almost all λ, the same permutation matrix $\mathcal{P}$ can be taken, since the proof only relies on the invertibility of $M_1 - \lambda N_1$. In all papers mentioned earlier [15, 1, 22, 17, 9, 10], $\mathcal{P}$ is chosen equal to the identity matrix. However, in general, it may happen that, by an unlucky choice of λ, permutation is necessary.

3. Rational Krylov method. The rational Krylov method [18, 19] is a generalization of the shift-and-invert Arnoldi method. There are two main differences between the two methods. First, instead of a fixed shift for the Arnoldi method, the rational Krylov method allows us to change the shift (or pole) at every iteration. Second, the rational Krylov method collects the information about the eigenvalues in a pair of Hessenberg matrices (K, H). The standard rational Krylov algorithm [19] is outlined in Algorithm 1.

ALGORITHM 1. RATIONAL KRYLOV METHOD.

1 Choose vector $\mathbf{v}_1$, where $\|\mathbf{v}_1\| = 1$.
for $j = 1, 2, \ldots$ **do**
2 Choose shift: σ_j.
3 Set continuation combination: t_j.
4 Compute: $\widehat{\mathbf{v}} := (\mathbf{A} - \sigma_j \mathbf{B})^{-1}\mathbf{B}\mathbf{w}_j$, where $\mathbf{w}_j = \mathbf{V}_j t_j$.
5 Orthogonalize: $\widetilde{\mathbf{v}} := \widehat{\mathbf{v}} - \mathbf{V}_j h_j$, where $h_j = \mathbf{V}_j^* \widehat{\mathbf{v}}$.
6 Get new vector: $\mathbf{v}_{j+1} = \widetilde{\mathbf{v}}/h_{j+1,j}$, where $h_{j+1,j} = \|\widetilde{\mathbf{v}}\|$.
7 Compute eigenpairs: (λ_i, s_i) and test for convergence.
end
8 Compute eigenvectors: $\mathbf{x}_i = \mathbf{V}_{j+1}\underline{H}_j s_i$.

The rational Krylov algorithm builds a subspace spanned by

$$\mathbf{v}_1,\ (\mathbf{A} - \sigma_1\mathbf{B})^{-1}\mathbf{B}\mathbf{w}_1,\ (\mathbf{A} - \sigma_2\mathbf{B})^{-1}\mathbf{B}\mathbf{w}_2, \ldots,$$

and by eliminating $\widehat{\mathbf{v}}$ and $\widetilde{\mathbf{v}}$ in the jth iteration in Algorithm 1 we get the relation

$$(\mathbf{A} - \sigma_j\mathbf{B})^{-1}\mathbf{B}\mathbf{w}_j = \mathbf{V}_{j+1}\underline{h}_j, \tag{3.1}$$

where $\underline{h}_j = \begin{bmatrix} h_j^* & h_{j+1,j}^* \end{bmatrix}^*$. Combining all the previous iterations, we arrive at the recurrence relation of the rational Krylov method,

$$\mathbf{A}\mathbf{V}_{j+1}\underline{H}_j = \mathbf{B}\mathbf{V}_{j+1}\underline{K}_j, \tag{3.2}$$

where $\underline{H}_j$ and $\underline{K}_j$ are two $(j+1) \times j$ upper Hessenberg matrices. The matrix $\underline{H}_j$ contains the coefficients of the Gram–Schmidt orthogonalization process and

$$\underline{K}_j = \underline{H}_j \operatorname{diag}(\sigma_1, \ldots, \sigma_j) + \underline{T}_j,$$

where the upper triangular matrix $\underline{T}_j$ is built up from the continuation combinations $t_1, \ldots, t_j$, selected in the way described in [19]. For the orthogonalization in step 5 of Algorithm 1, iterative Gram–Schmidt with reorthogonalization is used.

In each iteration step j, we assume that $h_{j+1,j} \neq 0$. Then, we call $\underline{H}_j$ *unreduced*. If $h_{j+1,j} = 0$, the Range$(\mathbf{V}_j)$ is an invariant subspace and

$$\mathbf{A}\mathbf{V}_j H_j = \mathbf{B}\mathbf{V}_j K_j,$$

where H_j and K_j are the $j \times j$ upper parts of $\underline{H}_j$ and $\underline{K}_j$, respectively. At this point, the Gram–Schmidt orthogonalization process fails.

Approximations for the eigenvalues and corresponding eigenvectors of the matrix pencil $(\mathbf{A}, \mathbf{B})$ can, in each iteration j of Algorithm 1, be obtained from the $j \times j$ upper parts of the two Hessenberg matrices $\underline{H}_j$ and $\underline{K}_j$,

$$K_j s_i = \lambda_i H_j s_i, \qquad s_i \neq 0.$$

Then, we call $(\lambda_i, \mathbf{x}_i := \mathbf{V}_{j+1}\underline{H}_j s_i)$ a Ritz pair of $(\mathbf{A}, \mathbf{B})$.

4. A compact rational Krylov decomposition. Consider the standard rational Krylov recurrence relation (3.2) with the matrices $\mathbf{A}$ and $\mathbf{B}$ as defined by (2.4). Then, similar to the compact Arnoldi decomposition [20], we can represent the subspace in a compact form.

We subdivide $\mathbf{V}_{j+1} \in \mathbb{C}^{dn\times(j+1)}$ as follows:

$$\mathbf{V}_{j+1} = \begin{bmatrix}\mathbf{V}_j & \mathbf{v}_{j+1}\end{bmatrix} = \begin{bmatrix} V_j^{[1]} & v_{j+1}^{[1]} \\ \vdots & \vdots \\ V_j^{[d]} & v_{j+1}^{[d]} \end{bmatrix},$$

where $V_j^{[i]} \in \mathbb{C}^{n\times j}$ and $v_{j+1}^{[i]} \in \mathbb{C}^n$ for $i = 1, \ldots, d$.

DEFINITION 4.1. *The matrix $Q_j \in \mathbb{C}^{n\times r_j}$ is defined so that its columns form an orthonormal basis for the column space of the matrix $[V_j^{[1]} \quad \cdots \quad V_j^{[d]}]$ with rank r_j.*

Using Definition 4.1, we can express $V_j^{[i]}$ as

$$V_j^{[i]} = Q_j U_j^{[i]}, \qquad i = 1, \ldots, d,$$

where $U_j^{[i]} \in \mathbb{C}^{r_j\times j}$. Thus, we have

$$\mathbf{V}_j = \begin{bmatrix} Q_j & & \\ & \ddots & \\ & & Q_j \end{bmatrix} \begin{bmatrix} U_j^{[1]} \\ \vdots \\ U_j^{[d]} \end{bmatrix} = (I_d \otimes Q_j)\mathbf{U}_j, \tag{4.1}$$

where

$$\mathbf{U}_j = \begin{bmatrix} U_j^{[1]} \\ \vdots \\ U_j^{[d]} \end{bmatrix} \in \mathbb{C}^{dr_j\times j}. \tag{4.2}$$

Since both $\mathbf{V}_j$ and $I_d \otimes Q_j$ are matrices with orthonormal columns, $\mathbf{U}_j$ also has orthonormal columns. Using the compact representation of $\mathbf{V}_j$ (4.1), the rational Krylov recurrence relation (3.2) yields the following CORK recurrence relation

$$\mathbf{A}(I_d \otimes Q_{j+1})\mathbf{U}_{j+1}\underline{H}_j = \mathbf{B}(I_d \otimes Q_{j+1})\mathbf{U}_{j+1}\underline{K}_j. \tag{4.3}$$

In order to refer to the CORK decomposition (4.3), we introduce the CORK quadruple.

DEFINITION 4.2 (CORK quadruple). *The quadruple $\left(Q_{j+1}, \mathbf{U}_{j+1}, \underline{H}_j, \underline{K}_j\right)$ with $Q_{j+1} \in \mathbb{C}^{n\times r_{j+1}}$, $\mathbf{U}_{j+1} \in \mathbb{C}^{dr_{j+1}\times(j+1)}$, and $\underline{H}_j, \underline{K}_j \in \mathbb{C}^{(j+1)\times j}$ is called a CORK quadruple of order j for $(\mathbf{A}, \mathbf{B})$, defined by (2.4), if*

1. *it satisfies the CORK recurrence relation (4.3),*
2. *Q_{j+1} has full rank and orthonormal columns and $\mathbf{U}_{j+1}$ has orthonormal columns,*
3. *$\underline{K}_j$ and $\underline{H}_j$ are upper Hessenberg matrices with $\underline{H}_j$ unreduced, and*
4. *none of the $\sigma_i = k_{i+1,i}/h_{i+1,i}$, $i = 1, \ldots, j$, is an eigenvalue of $(\mathbf{A}, \mathbf{B})$.*

LEMMA 4.3. *Let $\mathbf{A}$ and $\mathbf{B}$ be defined by (2.4). Then, by solving the linear system*

$$(\mathbf{A} - \sigma\mathbf{B})\mathbf{x} = \mathbf{B}\mathbf{y}, \tag{4.4}$$

one block of $\mathbf{x}$*, say* $x^{[p]}$*, is obtained from a system solve with* $P(\sigma)$*, while the other blocks of* $\mathbf{x}$ *are obtained as linear combinations of* $x^{[p]}$ *and the blocks of* $\mathbf{y}$*.*

Proof. The proof follows from the block ULP decomposition (Theorem 2.3) of $L(\sigma) = \mathbf{A} - \sigma\mathbf{B}$. The index p corresponds to the block column index of $\mathbf{A}$ and $\mathbf{B}$ that is permuted to the first block column of $\mathcal{U}(\sigma)\mathcal{L}(\sigma)$ by the permutation matrix $\mathcal{P} \otimes I_n$. From the UL factorization it can indeed be seen that the other blocks of $\mathbf{x}$ are linear combinations of blocks of $\mathbf{By}$ and $x^{[p]}$. This completes the proof. ∎

Lemma 4.3 now gives rise to Algorithm 2, where we use the block ULP decomposition in order to solve (4.4). This results in 1 matrix-vector operation followed by 2 block triangular system solves and a permutation.

ALGORITHM 2. SYSTEM SOLVE: $\mathbf{x} = (\mathbf{A} - \sigma\mathbf{B})^{-1}\mathbf{By}$.

1 Compute the right-hand side of (4.4): $\mathbf{z} = \mathbf{By}$.

2 Solve the block upper triangular system:

$$z^{[1]} = z^{[1]} - (\bar{\mathbf{A}}_1 - \sigma\bar{\mathbf{B}}_1)\left((M_1 - \sigma N_1)^{-1} \otimes I\right) z^{[2,\ldots,d]}.$$

3 Solve the block lower triangular system:

$$\text{(a)} \qquad z^{[1]} = P(\sigma)^{-1} z^{[1]},$$

$$\text{(b)} \qquad z^{[2,\ldots,d]} = \left((M_1 - \sigma N_1)^{-1} \otimes I\right)\left(z^{[2,\ldots,d]} - ((m_0 - \sigma n_0) \otimes I)\, z^{[1]}\right).$$

4 Permute the blocks of $\mathbf{z}$:

$$\mathbf{x} = (\mathcal{P}^T \otimes I)\mathbf{z}.$$

The following theorems summarize how the matrices Q_j and $\mathbf{U}_j$ can easily be extended into Q_{j+1} and $\mathbf{U}_{j+1}$, respectively. The Kronecker structure of the pencil $(\mathbf{A}, \mathbf{B})$, defined in (2.4), also provides an upper bound for the rank r_j of Q_j.

THEOREM 4.4. *Let* Q_j *be defined by Definition 4.1. Then,*

$$\operatorname{span}\{Q_{j+1}\} = \operatorname{span}\left\{Q_j, v_{j+1}^{[p]}\right\},$$

where $v_{j+1}^{[p]}$ *is the* p*th block in Lemma 4.3.*

Proof. By using Definition 4.1 and Lemma 4.3 with $\sigma = \sigma_j$, we have

$$\begin{aligned}
\operatorname{span}\{Q_{j+1}\} &= \operatorname{span}\left\{\begin{bmatrix} V_{j+1}^{[1]} & \cdots & V_{j+1}^{[d]} \end{bmatrix}\right\}, \\
&= \operatorname{span}\left\{Q_j, v_{j+1}^{[1]}, \ldots, v_{j+1}^{[d]}\right\}, \\
&= \operatorname{span}\left\{Q_j, v_{j+1}^{[p]}\right\},
\end{aligned}$$

which completes the proof. ∎

THEOREM 4.5. *Let* Q_j *be defined by Definition 4.1. Then, we have*

$$r_j < d + j.$$

Proof. By Definition 4.1, we have $\operatorname{span}\{Q_1\} = \operatorname{span}\{v_1^{[1]}, \ldots, v_1^{[d]}\}$. Then, from Theorem 4.4 it follows immediately that $r_j = \operatorname{rank}(Q_j) < d + j$, which concludes the proof. ∎

THEOREM 4.6. *Let* $\mathbf{U}_j \in \mathbb{C}^{dr_j \times j}$ *be defined by* (4.2). *Then,* $\mathbf{U}_{j+1} \in \mathbb{C}^{dr_{j+1} \times (j+1)}$ *takes the following form:*

$$\mathbf{U}_{j+1} = \begin{bmatrix} (I_d \otimes I_{r_{j+1} \times r_j}) \mathbf{U}_j & \mathbf{u}_{j+1} \end{bmatrix},$$

where $\mathbf{u}_{j+1} \in \mathbb{C}^{dr_{j+1}}$, *or, when* $r_{j+1} > r_j$,

$$U_{j+1}^{[i]} = \begin{bmatrix} U_j^{[i]} & u_{j+1}^{[i]} \\ 0_{1 \times j} & \end{bmatrix}, \qquad i = 1, \ldots, d.$$

Proof. The proof follows immediately from the definition. ∎

5. Compact rational Krylov method. In this section, we introduce the family of CORK methods applied to the linearization matrices $\mathbf{A}$ and $\mathbf{B}$ defined in (2.4).

In these methods, we use the standard rational Krylov process to generate $\mathbf{v}_{j+1}$ in its compact representation. We start with expressing the starting vector $\mathbf{v}_1$ in a compact form, i.e.,

$$\mathbf{v}_1 = (I_d \otimes Q_1)\mathbf{U}_1,$$

where $Q_1 \in \mathbb{C}^{n \times r_1}$ and $\mathbf{U}_1 \in \mathbb{C}^{dr_1}$, with $r_1 = \operatorname{rank}([v_1^{[1]} \quad \cdots \quad v_1^{[d]}])$. This results in a CORK quadruple of order 0. Next, given a CORK quadruple of order $j-1$, we compute the CORK quadruple of order j. This results in a two level orthogonalization process. First, Q_j is expanded into Q_{j+1} (*first level orthogonalization*). Second, $\mathbf{U}_j$ is expanded into $\mathbf{U}_{j+1}$ and the Hessenberg matrices are updated to $\underline{H}_j$ and $\underline{K}_j$ (*second level orthogonalization*).

Before presenting the algorithm in Section 5.3, we describe the two levels of orthogonalization which are needed to expand the matrices Q and $\mathbf{U}$. Section 5.1 discusses the first level orthogonalization and Section 5.2 the second level orthogonalization. Next, we discuss in Section 5.4 how the orthogonalization cost can significantly be reduced.

5.1. First level orthogonalization. From Theorem 4.4, we know that for expanding Q_j into Q_{j+1}, we need to compute $v_{j+1}^{[p]}$ and orthogonalize this vector against Q_j. Furthermore, the shift-and-invert step (Algorithm 1, step 4),

$$\widehat{\mathbf{v}} := (\mathbf{A} - \sigma_j \mathbf{B})^{-1} \mathbf{B} \mathbf{w}_j = (\mathbf{A} - \sigma_j \mathbf{B})^{-1} \mathbf{B} (I_d \otimes Q_j) \mathbf{U}_j t_j, \tag{5.1}$$

can easily be solved by Algorithm 2. However, since

$$\operatorname{span}\{Q_{j+1}\} = \operatorname{span}\left\{Q_j, v_{j+1}^{[p]}\right\} = \operatorname{span}\left\{Q_j, \widehat{v}^{[p]}\right\},$$

we only need to compute $\widehat{v}^{[p]}$. Therefore, step 3(b) in Algorithm 2 can be saved. Next, we orthogonalize $\widehat{v}^{[p]}$ against Q_j. Thus, let us denote

$$\widetilde{q} := \widehat{v}^{[p]} - Q_j Q_j^* \widehat{v}^{[p]} \qquad \text{and} \qquad \delta = \|\widetilde{q}\|.$$

Suppose first that $\delta \neq 0$, then we use $q_{j+1} = \widetilde{q}/\delta$ to expand Q_j into

$$Q_{j+1} = \begin{bmatrix} Q_j & q_{j+1} \end{bmatrix},$$

and $r_{j+1} = r_j + 1$. On the other hand, $\delta = 0$ implies that $\widehat{v}^{[p]}$ lies in the subspace spanned by Q_j. This situation is called *deflation* in SOAR [2]. In this case, we take $Q_{j+1} = Q_j$ and $r_{j+1} = r_j$.

5.2. Second level orthogonalization. Once Q_{j+1} is known, we still have to compute $\mathbf{u}_{j+1}$. We will show that $\mathbf{u}_{j+1}$, $\underline{H}_j$, and $\underline{K}_j$ can be computed from a rational Krylov step on small matrices. To see this, we need the following lemma.

LEMMA 5.1. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be defined by* (2.4) *and define* $\widetilde{\mathbf{A}}_{j+1}$ *and* $\widetilde{\mathbf{B}}_{j+1}$ *as follows:*

$$\widetilde{\mathbf{A}}_{j+1} = (I_d \otimes Q_{j+1}^*) \begin{bmatrix} P(\sigma_j)^{-1} & 0 \\ 0 & I_{(d-1)n} \end{bmatrix} \mathbf{A}(I_d \otimes Q_{j+1}),$$
$$\widetilde{\mathbf{B}}_{j+1} = (I_d \otimes Q_{j+1}^*) \begin{bmatrix} P(\sigma_j)^{-1} & 0 \\ 0 & I_{(d-1)n} \end{bmatrix} \mathbf{B}(I_d \otimes Q_{j+1}).$$

Then, the shift-and-invert rational Krylov relation (3.1) *is equivalent with*

$$\left(\widetilde{\mathbf{A}}_{j+1} - \sigma_j \widetilde{\mathbf{B}}_{j+1}\right)^{-1} \widetilde{\mathbf{B}}_{j+1} \underline{\mathbf{U}}_j t_j = \mathbf{U}_{j+1} \underline{h}_j,$$

where $\underline{\mathbf{U}}_j$ *is such that* $(I_d \otimes Q_j)\mathbf{U}_j = (I_d \otimes Q_{j+1})\underline{\mathbf{U}}_j$.

Proof. First, note that by using the block ULP decomposition of Theorem 2.3, we have that

$$\widetilde{\mathbf{A}}_{j+1} - \sigma_j \widetilde{\mathbf{B}}_{j+1} = \begin{bmatrix} \alpha I_{r_{j+1}} & Q_{j+1}^* P(\sigma_j)^{-1} (\bar{\mathbf{A}}_1 - \sigma_j \bar{\mathbf{B}}_1) \left((M_1 - \sigma_j N_1)^{-1} \otimes Q_{j+1}\right) \\ 0 & I_{(d-1)r_{j+1}} \end{bmatrix}$$
$$\times \begin{bmatrix} I_{r_{j+1}} & 0 \\ (m_0 - \sigma_j n_0) \otimes I_{r_{j+1}} & (M_1 - \sigma_j N_1) \otimes I_{r_{j+1}} \end{bmatrix} (\mathcal{P} \otimes I_{r_{j+1}}),$$

is always nonsingular. Next, rewriting relation (3.1) as follows,

$$(\mathbf{A} - \sigma_j \mathbf{B})(I_d \otimes Q_{j+1})\mathbf{U}_{j+1} \underline{h}_j = \mathbf{B}(I_d \otimes Q_j)\mathbf{U}_j t_j,$$

and multiplying on the left by $(I_d \otimes Q_{j+1}^*) \left[\begin{smallmatrix} P(\sigma_j)^{-1} & 0 \\ 0 & I_{(d-1)n} \end{smallmatrix}\right]$,

$$\left(\widetilde{\mathbf{A}}_{j+1} - \sigma_j \widetilde{\mathbf{B}}_{j+1}\right) \mathbf{U}_{j+1} \underline{h}_j = \widetilde{\mathbf{B}}_{j+1} \underline{\mathbf{U}}_j t_j.$$

Using that $\widetilde{\mathbf{A}}_{j+1} - \sigma_j \widetilde{\mathbf{B}}_{j+1}$ is invertible completes the proof. ∎

As a consequence of Lemma 5.1, $\mathbf{u}_{j+1}$ satisfies the following standard rational Krylov recurrence relation

$$\widetilde{\mathbf{A}}_{j+1} \mathbf{U}_{j+1} \underline{H}_j = \widetilde{\mathbf{B}}_{j+1} \mathbf{U}_{j+1} \underline{K}_j, \tag{5.2}$$

where the Hessenberg matrices $\underline{H}_j$ and $\underline{K}_j$ are the same as in the original CORK recurrence relation (4.3). Hence, the second level orthogonalization in each iteration of the CORK algorithm can be seen as a standard rational Krylov step (5.2) with the small matrices $\widetilde{\mathbf{A}}_{j+1}$ and $\widetilde{\mathbf{B}}_{j+1}$.

Note that, although $\widetilde{\mathbf{A}}_{j+1}$ and $\widetilde{\mathbf{B}}_{j+1}$ might change in every iteration, they always have the same Kronecker structure below the first block row as $\mathbf{A}$ and $\mathbf{B}$, respectively. Therefore, it will not be necessary to construct $\widetilde{\mathbf{A}}_{j+1}$ and $\widetilde{\mathbf{B}}_{j+1}$ explicitly, as we will explain in Section 5.3.

5.3. Algorithm. Based on Lemma 4.3 and Theorems 4.4–4.6, the two levels of orthogonalization, explained in Sections 5.1 and 5.2, can efficiently be implemented. The corresponding CORK algorithm is outlined in Algorithm 3.

ALGORITHM 3. COMPACT RATIONAL KRYLOV METHOD.

1 Choose Q_1 and $\mathbf{U}_1$, where $Q_1^*Q_1 = I_{r_1}$ and $\mathbf{U}_1^*\mathbf{U}_1 = 1$.
for $j = 1, 2, \ldots$ **do**
2 Choose shift: σ_j.
FIRST LEVEL ORTHOGONALIZATION:
3 Compute: $\widehat{v}^{[p]}$ via Algorithm 2.
4 Orthogonalize: $\widetilde{q} := \widehat{v}^{[p]} - Q_j Q_j^* \widehat{v}^{[p]}$.
5 Next vector: $q_{j+1} = \widetilde{q}/\|\widetilde{q}\|$.
SECOND LEVEL ORTHOGONALIZATION:
6 Update matrices: $U_j^{[i]} = \begin{bmatrix} U_j^{[i]} \\ 0 \end{bmatrix}$ for $i = 1, \ldots, d$.
7 Compute: $\widehat{\mathbf{u}}$ via Algorithm 2.
8 Orthogonalize: $\widetilde{\mathbf{u}} := \widehat{\mathbf{u}} - \mathbf{U}_j \mathbf{U}_j^* \widehat{\mathbf{u}}$.
9 Next vector: $\mathbf{u}_{j+1} = \widetilde{\mathbf{u}}/h_{j+1,j}$, where $h_{j+1,1} = \|\widetilde{\mathbf{u}}\|$.
10 Compute eigenpairs: (λ_i, s_i) and test for convergence.
end
11 Compute eigenvectors: $\mathbf{x}_i = (I_d \otimes Q_{j+1})\mathbf{U}_{j+1}\underline{H}_j s_i$.

Before starting the rational Krylov iteration in Algorithm 3, we need to choose a starting vector (step 1). A first possibility is taking a randomly generated vector $v_0 \in \mathbb{C}^{nd}$. Then, using the economy-size QR decomposition of $[v_0^{[1]} \quad \cdots \quad v_0^{[d]}] = \mathcal{Q}\mathcal{R}$ yields

$$Q_1 = \mathcal{Q} \in \mathbb{C}^{n \times d}, \qquad \mathbf{U}_1 = \operatorname{vec}(\mathcal{R}) \in \mathbb{C}^{d^2},$$

and $r_1 = d$. On the other hand, from Corollary 2.4 we know that the eigenvectors have a Kronecker structure. Therefore, we can also start Algorithm 3 with $v_0 = f \otimes q_0$, where $q_0 \in \mathbb{C}^n$ and $f \in \mathbb{C}^d$. Consequently, this results in

$$Q_1 = q_0/\|q_0\| \in \mathbb{C}^n, \qquad \mathbf{U}_1 = \|q_0\| f \in \mathbb{C}^d, \tag{5.3}$$

and $r_1 = 1$.

For methods with dynamically growing linearizations, such as the infinite Arnoldi method [11] and the Newton rational Krylov method [21], the structured starting vector (5.3) corresponds to taking $f := e_1$, with e_1 the first unit vector. Also in cases where it is inappropriate to choose f as a unit vector, i.e., in a Lagrange basis, the structured starting vector (5.3) is advantageous, since we only need to store one vector of dimension n instead of d vectors of dimension n.

In each iteration step j of Algorithm 3 we start with choosing a shift σ_j (step 2). Next, the two levels of orthogonalization are performed in steps 3–5 and steps 6–9, respectively. In the first level, we compute the pth block $\widehat{v}^{[p]}$ of the next rational Krylov vector $\widehat{\mathbf{v}}$ by Algorithm 2. Note that, for only computing $\widehat{v}^{[p]}$, we can skip step 3(b). Next, we orthogonalize this vector $\widehat{v}^{[p]}$ against Q_j in order to obtain q_{j+1}. Thereafter, in the second level, we perform a standard rational Krylov step with the projected matrices $\widetilde{\mathbf{A}}_{j+1}$ and $\widetilde{\mathbf{B}}_{j+1}$ in order to obtain $\mathbf{u}_{j+1}$ and to expand the

Hessenberg matrices. However, in practice, it is not necessary to form the matrices $\widetilde{\mathbf{A}}_{j+1}$ and $\widetilde{\mathbf{B}}_{j+1}$ since the pth block of $\widehat{\mathbf{u}}$ can be computed as follows:

$$\widehat{u}^{[p]} = Q_{j+1}^* \widehat{v}^{[p]},$$

where $Q_j^* \widehat{v}^{[p]}$ is already computed during the first level orthogonalization. Then, the other blocks of $\widehat{\mathbf{u}}$ can be computed by Algorithm 2 where we skip steps 2 and 3(a). Finally, in step 10 of Algorithm 3, we compute the Ritz pairs (λ_i, s_i) and test for convergence.

Remark also that, from the definition of $\mathbf{U}$ (4.2), it is natural to represent $\mathbf{U}$ as a tensor. Therefore, in the implementation of Algorithm 3, we have stacked the blocks $U^{[i]}$ for $i = 1, \ldots, d$ behind each other.

5.4. Orthogonalization cost. The CORK method not only results in a much lower memory cost, but also the orthogonalization cost can be significantly reduced. In particular, we do not need to explicitly compute blocks other than $\widehat{v}^{[p]}$ in (5.1), since the orthogonalization process in the second level only uses small matrices.

As mentioned before, the second level orthogonalization involves only 1 extra scalar product between vectors of size n for computing $\widehat{u}^{[p]}$. The other blocks of $\widehat{\mathbf{u}}$ can be computed as linear combinations of the blocks of $\mathbf{u}_j$ and $\widehat{u}^{[p]}$. This means, we only have to deal with vectors of length r_{j+1} in the second level orthogonalization. Therefore, the dominant orthogonalization cost takes place in the first level orthogonalization where the vector $\widehat{v}^{[p]} \in \mathbb{C}^n$ is orthogonalized against $q_1, \ldots, q_{r_j}$. In the second level orthogonalization we only have to deal with short vectors.

Table 2 gives an overview of the number of scalar products between vectors of size n in the orthogonalization process of the standard rational Krylov method and the CORK method. For dynamically growing linearization, such as in the infinite Arnoldi method [10] and the Newton rational Krylov method [21], this number is of order $O(j^3)$. However, by using the CORK method with $v_1 = 1 \otimes x$ as starting vector, it reduces to $O(j^2)$. On the other hand, this number in the standard rational Krylov method for fixed size linearizations is of order $O(dj^2)$. Using the CORK method with a full starting vector v_1 it reduces to $O(j(d+j))$ and by using a structured starting vector $v_1 = f \otimes q_0$ it further reduces to $O(j^2)$. Note that this is the same order of magnitude as for the CORK method for dynamically growing linearizations.

TABLE 2

Number of scalar products between vectors of size n in the standard rational Krylov method and the CORK methods.

Linearization	Rat. Krylov	CORK	
		full v_1	$v_1 = f \otimes q_0$
dynamically growing	$O(j^3)$	-	$O(j^2)$
fixed size	$O(dj^2)$	$O(j(d+j))$	$O(j^2)$

6. Implicit restarting. Since the CORK method is a special variant of the rational Krylov method, we can also perform implicit restarting [13, 16] on Algorithm 3. Therefore, we first apply a transformation on the Hessenberg matrices $\underline{H}$ and $\underline{K}$, which allows us to reorder and lock Ritz values. Next, representing the new subspace in its compact form, completes the restart of the CORK process. Note that the restarting techniques explained in this section are a kind of generalization of the ones in [12] to rational Krylov and also to structured linearization pencils in different bases.

Suppose that after m iterations of the CORK algorithm, we have the CORK quadruple $(Q_{m+1}, \mathbf{U}_{m+1}, \underline{H}_m, \underline{K}_m)$ which we want to reduce to a smaller CORK quadruple $(Q_{k+1}, \mathbf{U}_{k+1}, \underline{H}_k, \underline{K}_k)$ with $k < m$. Therefore, we start with defining the following matrices

$$\underline{H}^+ = Y^* \underline{H}_m Z,$$
$$\underline{K}^+ = Y^* \underline{K}_m Z,$$

where $Y \in \mathbb{C}^{(m+1)\times(k+1)}$ and $Z \in \mathbb{C}^{m\times k}$ have unitary columns. With a proper choice of Y and Z (see [4, 19]), we have that

$$\mathbf{A}(I_d \otimes Q_{m+1})\mathbf{W}\underline{H}^+ = \mathbf{B}(I_d \otimes Q_{m+1})\mathbf{W}\underline{K}^+, \tag{6.1}$$

where $\mathbf{W} := \mathbf{U}_{m+1}Y$.

Next, note that in (6.1) the matrix Q remains the same, although the unwanted Ritz values are removed from the pencil (K, H). However, from Definition 4.2 and Theorem 4.5 we know that the rank of Q is bounded, also after restarting. Therefore, suppose the economy size singular value decomposition of

$$\begin{bmatrix} W^{[1]} & \cdots & W^{[d]} \end{bmatrix} = \mathcal{U}\mathcal{S}\begin{bmatrix} \mathcal{V}^{[1]} & \cdots & \mathcal{V}^{[d]} \end{bmatrix}, \tag{6.2}$$

where $\mathcal{U} \in \mathbb{C}^{r_{m+1}\times r}$, $\mathcal{S} \in \mathbb{C}^{r\times r}$, and $\mathcal{V}^{[i]} \in \mathbb{C}^{r\times(k+1)}$ for $i = 1, \ldots, d$. By the definition of $\mathbf{W}$, we have $r \le d + k$. Then, by substituting (6.2) into (6.1), we obtain

$$\mathbf{A}(I_d \otimes Q^+)\mathbf{U}^+\underline{H}^+ = \mathbf{B}(I_d \otimes Q^+)\mathbf{U}^+\underline{K}^+, \tag{6.3}$$

where

$$Q^+ := Q_{m+1}\mathcal{U}, \qquad \mathbf{U}^+ := \begin{bmatrix} \mathcal{S}\mathcal{V}^{[1]} \\ \vdots \\ \mathcal{S}\mathcal{V}^{[d]} \end{bmatrix}.$$

Finally, note that the recurrence relation (6.3) is a standard CORK recurrence relation of order k with $Q_{k+1} := Q^+$, $\mathbf{U}_{k+1} := \mathbf{U}^+$, $\underline{H}_k := \underline{H}^+$, and $\underline{K}_k := \underline{K}^+$.

7. Low rank exploitation. In several applications, many of the blocks A_i and B_i in (2.4) are of low rank. Therefore, in this section we generalize the low rank structure exploitation proposed in [21].

Suppose that for $\widetilde{d} \le i < d$ the blocks A_i and B_i in (2.4) are of low rank. Furthermore, we assume that

$$A_i = \widetilde{A}_i\widetilde{Z}^*, \qquad B_i = \widetilde{B}_i\widetilde{Z}^*, \qquad i = \widetilde{d}, \ldots, d-1,$$

where $\widetilde{A}_i, \widetilde{B}_i, \widetilde{Z} \in \mathbb{C}^{n\times\widetilde{n}}$, $\widetilde{Z}$ has orthonormal columns, and $\widetilde{n} \ll n$. Then, we can transform (2.4) into a linear companion pencil of dimension $\widetilde{d}n + (d - \widetilde{d})\widetilde{n}$,

$$\widetilde{L}(\lambda) = \widetilde{\mathbf{A}} - \lambda\widetilde{\mathbf{B}}, \tag{7.1}$$

where

$$\widetilde{\mathbf{A}} = \left[\begin{array}{ccccc} A_0 & \cdots & A_{\widetilde{d}-1} & \widetilde{A}_{\widetilde{d}} & \cdots & \widetilde{A}_{d-1} \\ \hline \multicolumn{3}{c}{M_{11} \otimes I_n} & \multicolumn{3}{c}{0} \\ \multicolumn{3}{c}{M_{21} \otimes \widetilde{Z}^*} & \multicolumn{3}{c}{M_{22} \otimes I_{\widetilde{n}}} \end{array}\right], \tag{7.2}$$

$$\widetilde{\mathbf{B}} = \left[\begin{array}{ccccc} B_0 & \cdots & B_{\widetilde{d}-1} & \widetilde{B}_{\widetilde{d}} & \cdots & \widetilde{B}_{d-1} \\ \hline \multicolumn{3}{c}{N_{11} \otimes I_n} & \multicolumn{3}{c}{0} \\ \multicolumn{3}{c}{N_{21} \otimes \widetilde{Z}^*} & \multicolumn{3}{c}{N_{22} \otimes I_{\widetilde{n}}} \end{array}\right] \tag{7.3}$$

with the assumption that $M_{12} = N_{12} = 0$. This is the case for many of the bases used in the literature, e.g., degree-graded polynomial bases, Lagrange basis, rational Newton basis, etc. Similarly to Corollary 2.4, the linearization (7.1)–(7.3) yields the following structured eigenvector:

$$\widetilde{\mathbf{y}} = \left[\begin{array}{c} f_1 \otimes x \\ \hline f_2 \otimes \widetilde{Z}^* x \end{array}\right],$$

which is the motivation to subdivide $\mathbf{V}$ as follows:

$$\mathbf{V} = \left[\begin{array}{c} V^{[1]} \\ \vdots \\ V^{[\widetilde{d}]} \\ \hline \widetilde{V}^{[\widetilde{d}+1]} \\ \vdots \\ \widetilde{V}^{[d]} \end{array}\right] = \left[\begin{array}{ccc|ccc} Q & & & & & \\ & \ddots & & & & \\ & & Q & & & \\ \hline & & & \widetilde{Q} & & \\ & & & & \ddots & \\ & & & & & \widetilde{Q} \end{array}\right] \left[\begin{array}{c} U^{[1]} \\ \vdots \\ U^{[\widetilde{d}]} \\ \hline \widetilde{U}^{[\widetilde{d}+1]} \\ \vdots \\ \widetilde{U}^{[d]} \end{array}\right],$$

where

$$\begin{array}{lll} V^{[1,\ldots,\widetilde{d}]} \in \mathbb{C}^{n\times j}, & Q \in \mathbb{C}^{n\times r}, & \mathbf{U} \in \mathbb{C}^{r\times j}, \\ V^{[\widetilde{d}+1,\ldots,d]} \in \mathbb{C}^{\widetilde{n}\times j}, & \widetilde{Q} \in \mathbb{C}^{\widetilde{n}\times \widetilde{r}}, & \widetilde{\mathbf{U}} \in \mathbb{C}^{\widetilde{r}\times j}. \end{array}$$

In applications where $\widetilde{d}$ is relatively small compared to d, the memory cost can significantly be reduced since $r \leq \widetilde{d} + j$. Note that, due to the appearance of $\widetilde{Z}^*$ in (7.2) and (7.3), $\widetilde{r}$ is not bounded any more by Theorem 4.5. However, for large n the memory cost for storing $\widetilde{Q}$ (involving $\widetilde{n}$ and $\widetilde{r}$) is almost negligible compared to the one for Q (involving n and r). Furthermore, as we will illustrate in the numerical experiments in Section 8.2, $\widetilde{r}$ also remains bounded in practice.

8. Numerical examples. We now illustrate the CORK method (Algorithm 3) with two large-scale examples. In the first example, we consider a delay eigenvalue problem [10] for which we used a dynamically growing linearization based on spectral discretization. In this example, we compare the memory usage and the orthogonalization cost for the standard rational Krylov method with the CORK method. We show results for both with and without the implicit restarting technique explained in Section 6. In the second example, we consider the "gun" problem of the NLEVP collection [3] for which we used a rational Newton linearization of fixed degree. Here, we also used the CORK method with the low rank exploitation of Section 7.

All numerical experiments are performed in MATLAB version 7.14.0 (R2012a) on a Dell Latitude notebook running an Intel(R) Core(TM) i5-2540M CPU @ 2.60 GHz quad core processor with 8 GB RAM. Our experiments can be reproduced with the CORK code available from http://twr.cs.kuleuven.be/research/software/nleps/cork.html.

8.1. Delay problem. We start with a delay eigenvalue problem [10] corresponding to the delay differential equation

$$\frac{\partial v(x,t)}{\partial t} = \frac{\partial^2 v(x,t)}{\partial x^2} + a_0(x)v(x,t) + a_1(x)v(\pi - x, t-1), \tag{8.1}$$

where $a_0(x) = -2\sin(x)$, $a_1(x) = 2\sin(x)$, and $v_x(0,t) = v_x(\pi,t) = 0$. We discretize (8.1) by approximating the second derivative in space with central differences and obtain the following NLEP:

$$A(\lambda)x = (A_0 - \lambda I + A_1 e^{-\tau\lambda})x = 0, \tag{8.2}$$

where $A_0, A_1 \in \mathbb{R}^{5000\times 5000}$ and $\tau = 1$. By using a spectral dicretization, the linearization matrices have a companion-type structure [10, Theorem 2.1]. The goal in this experiment is to compute the 20 eigenvalues closest to the origin. For measuring the convergence of an approximate eigenpair (λ, x), we used the following relative residual norm

$$E(\lambda, x) = \frac{\|A(\lambda)x\|_2 / \|x\|_2}{\|A_0\|_1 + |\lambda| + |e^{-\tau\lambda}|\|A_1\|_1}.$$

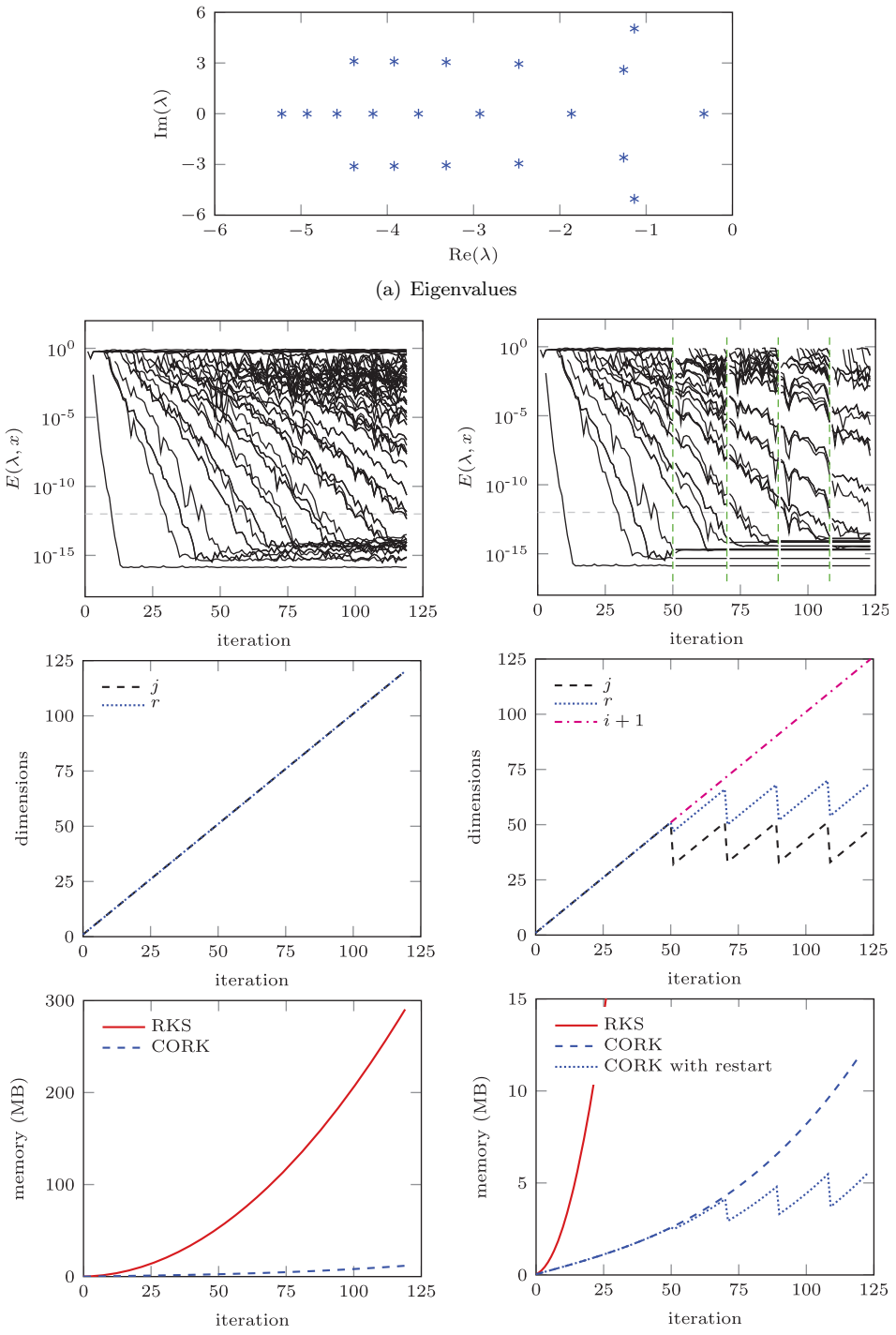
We first solved the NLEP (8.1) by Algorithm 3 without using restart and chose a *short* starting vector $v_0 \in \mathbb{R}^{5000}$ such that $r_1 = \operatorname{rank}(Q_1) = 1$. The eigenvalues and results of this experiment are shown in Figures 1(a) and 1(b), respectively. The convergence histories of the eigenpairs are given in the top figure, from which we see that after 119 iterations we found the 20 smallest eigenvalues in magnitude up to a tolerance of 10^{-12}. Since we did not allow restart, we see in the middle figure that the rank of Q, r, and the dimension of the the subspace, j, increase with the iteration count i. The bottom figure shows the memory usage for storing the subspace in the standard rational Krylov method (Algorithm 1) and in the CORK method (Algorithm 3). From this figure, we see that we get a gain factor of 25 since the standard rational Krylov method requires $O(n \cdot i^2/2)$ to store the subspace and the CORK method only $O(n \cdot r)$, where $r = i + 1$.

Next, we solved (8.1) by Algorithm 3 combined with the implicit restarting technique explained in Section 6. Here, we chose the maximum dimension of the subspace, $m = 50$, and the number of selected Ritz values, $p = 30$. The results are shown in Figure 1(c). This figure shows again the convergence histories of the eigenpairs at the top. We see that the method requires 4 restarts (indicated by the vertical green dashed lines) and 123 iterations before finding the 20 smallest eigenvalues in magnitude up to the tolerance. In theory the rank of Q is unbounded, since we used a dynamically growing linearization and thus $r \leq i + 1$ with i the iteration count, i.e., the number of rational Krylov steps which is larger than m in the situation with restarting. However, we notice in the middle figure that in practice r stagnates in the course of the algorithm. Consequently, it has also a positive effect on the memory requirements as illustrated in the bottom figure. By using a restarted CORK method, we were able to reduce the memory cost for this delay problem by a factor of 50.

8.2. Gun problem. We consider the "gun" problem of the NLEVP collection [3]. This is a large-scale problem that models a radio-frequency gun cavity and is of the form [14]

$$A(\lambda)x = \left(K - \lambda M + i\sqrt{\lambda - \sigma_1^2}\, W_1 + i\sqrt{\lambda - \sigma_2^2}\, W_2\right) x = 0, \tag{8.3}$$

where M, K, W_1, and W_2 are real symmetric matrices of size 9956×9956, K is positive semidefinite, M is positive definite, and $\operatorname{rank}(W_1) + \operatorname{rank}(W_2) = 84$. The complex square root $\sqrt{\cdot}$ corresponds to the principal branch and as in [3], we take $\sigma_1 = 0$ and



(a) Eigenvalues

(b) CORK method without restart

(c) CORK method with restart

FIG. 1. *Results for the delay problem.*

$\sigma_2 = 108.8774$. The goal in this experiment is to compute the 20 eigenvalues closest to 250^2. For measuring the convergence of an approximate eigenpair (λ, x), we used the relative residual norm $E(\lambda, x)$ defined in [14]. Similarly as in [9], we approximate $A(\lambda)$ in (8.3) by an interpolating rational Newton polynomial on the upper half-disk with center 250^2 and radius $300^2 - 200^2$. This results in a fixed linearization of degree $d = 36$. For more information on how to construct this linearization pencil, we refer to [9, section 7.1.2].

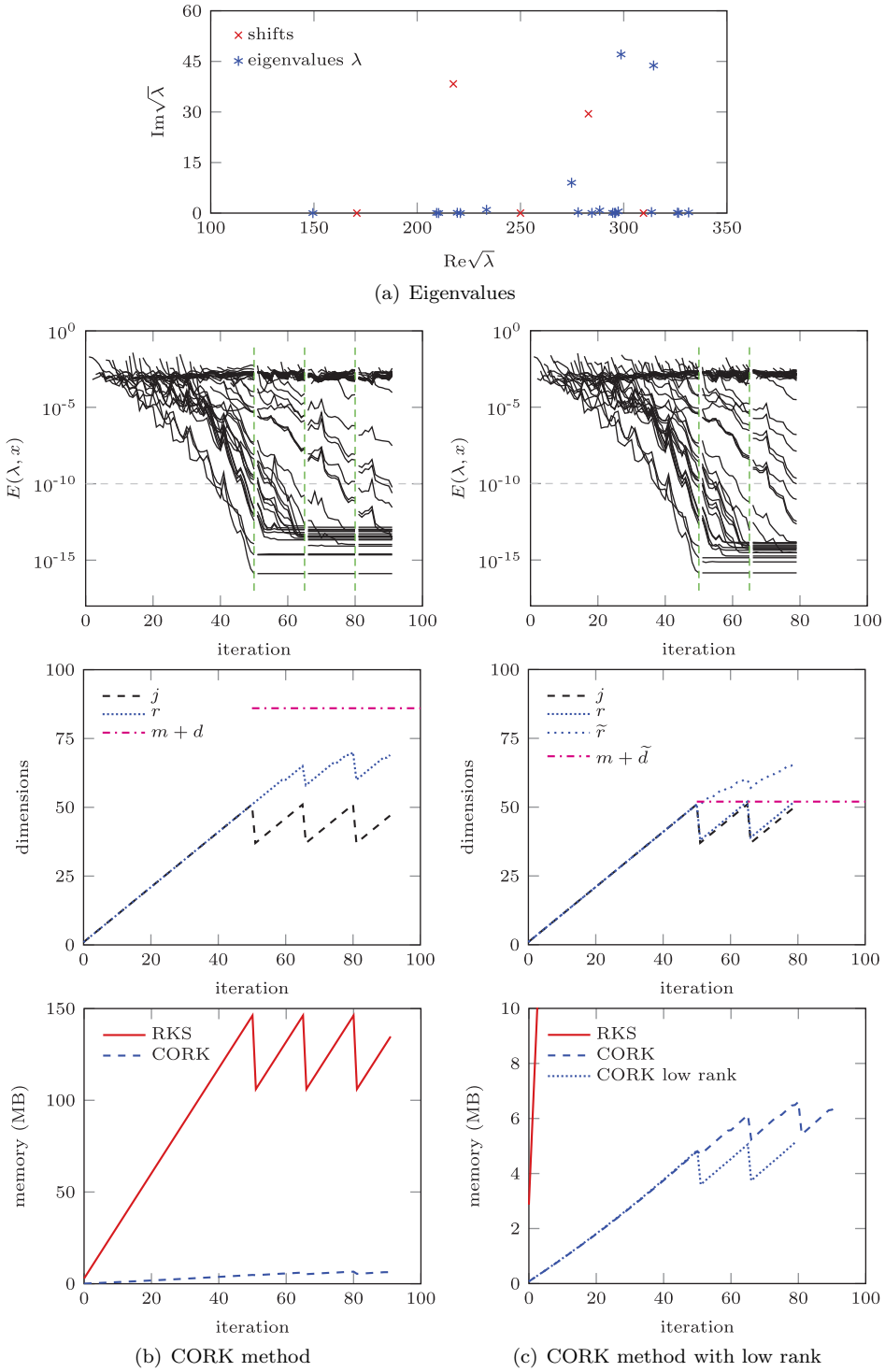
In a first experiment, the NLEP (8.3) is solved by Algorithm 3 with maximum subspace dimension $m = 50$ and $p = 35$ selected Ritz values in every restart. We also used cyclically repeated shifts in the rational Krylov steps, indicated by "$\times$" in Figure 2(a). The computed eigenvalues and the results of this experiment are shown in Figures 2(a) and 2(b), respectively. From the convergence histories of the eigenpairs in the top figure, we see that we needed 91 iterations to compute the 20 eigenvalues closest to 250^2 up to a tolerance of 10^{-10}. In the middle figure, we see that the rank of Q, r, stagnates again in practice and remains significantly below the theoretical upper bound $r \leq m + d$. Next, comparing the subspace memory cost of the CORK method with the one of the standard rational Krylov method results in a reduction with a factor of more than 20.

In a final experiment, we solve (8.3) by Algorithm 3 and use the low rank exploitation of Section 7. Note that in this case, the blocks A_i and B_i for $i \geq \widetilde{d} = 2$ in the linearization pencil have only $\widetilde{n} = 84$ columns. For this experiment, we chose all parameters equal to the ones in the previous experiment. The corresponding results are shown in Figure 2(c). In the top figure with the convergence histories of the eigenpairs, we see that now only 79 iterations are needed to compute the 20 eigenvalues closest to 250^2. In the middle figure, we see that the rank of Q, r, is now bounded by the upper bound $r \leq m + \widetilde{d}$ with $\widetilde{d} < d$. Note also that the rank of $\widetilde{Q}$, $\widetilde{r}$, is small. Consequently, since in the CORK method with low rank exploitation the memory cost is dominated by Q with $r \leq m + \widetilde{d}$ compared to $r \leq m + d$ for the standard CORK method, we notice in the bottom figure that the subspace memory cost is even further reduced.

9. Conclusions. In this paper, we have proposed a uniform framework of CORK methods for solving large-scale NLEPs. We also introduced a generic but simple representation of structured linearization pencils. The family of CORK methods is most applicable in cases where $d \ll n$ and $d \ll m$, with d the degree of the (rational) matrix polynomial, n the original problem dimension, and m the maximum dimension of the subspace.

By representing the subspace $\mathbf{V}$ in a compact form with $\mathbf{V} = (I \otimes Q)\mathbf{U}$, we are able to both reduce the memory cost as well as the orthogonalization cost. We also proved that the rank of Q, where Q forms an approximation of the eigenspace, is bounded by $m + d$. Therefore, the memory cost reduced from $O(dn \cdot m)$ to $O(n \cdot (d + m))$ and the orthogonalization process only involves $O((d + m)m)$ scalar products of size n instead of $O(dm^2)$. The numerical experiments showed that in practice we often get a further reduction when the upper bound $m + d$ on the rank of Q is not attained. We also briefly discussed implicit restarting of the CORK method and how to exploit low rank structure in the linearization pencil.

Acknowledgments. The authors are grateful to Laurent Sorber for the useful discussions about the tensor implementation of the MATLAB code. We also thank the referees for their suggestions which improved the presentation of the paper.



(a) Eigenvalues

(b) CORK method

(c) CORK method with low rank

FIG. 2. *Results for the "**gun**" problem.*

REFERENCES

[1] A. Amiraslani, R. M. Corless, and P. Lancaster, *Linearization of matrix polynomials expressed in polynomial bases*, IMA J. Numer. Anal., 29 (2009), pp. 141–157.

[2] Z. Bai and Y. Su, *SOAR: A second-order Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 640–659.

[3] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, *NLEVP: A collection of nonlinear eigenvalue problems*, ACM Trans. Math. Softw., 39 (2013), 7.

[4] G. De Samblanx, K. Meerbergen, and A. Bultheel, *The implicit application of a rational filter in the RKS method*, BIT, 37 (1997), pp. 925–947.

[5] F. De Terán, F. M. Dopico, and D. S. Mackey, *Fiedler companion linearizations and the recovery of minimal indices*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2181–2204.

[6] C. Effenberger and D. Kressner, *Chebyshev interpolation for nonlinear eigenvalue problems*, BIT, 52 (2012), pp. 933–951.

[7] M. Fiedler, *A note on companion matrices*, Linear Algebra Appl., 372 (2003), pp. 325–331.

[8] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, Academic Press, New York, 1982.

[9] S. Güttel, R. Van Beeumen, K. Meerbergen, and W. Michiels, *NLEIGS: A class of fully rational Krylov methods for nonlinear eigenvalue problems*, SIAM J. Sci. Comput., 36 (2014), pp. A2842–A2864.

[10] E. Jarlebring, K. Meerbergen, and W. Michiels, *A Krylov method for the delay eigenvalue problem*, SIAM J. Sci. Comput., 32 (2010), pp. 3278–3300.

[11] E. Jarlebring, W. Michiels, and K. Meerbergen, *A linear eigenvalue algorithm for the nonlinear eigenvalue problem*, Numer. Math., 122 (2012), pp. 169–195.

[12] D. Kressner and J. E. Roman, *Memory-efficient Arnoldi algorithms for linearizations of matrix polynomials in Chebyshev basis*, Numer. Linear Algebra Appl., 21 (2014), pp. 569–588.

[13] 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.

[14] B.-S. Liao, Z. Bai, L.-Q. Lee, and K. Ko, *Nonlinear Rayleigh-Ritz iterative method for solving large scale nonlinear eigenvalue problems*, Taiwanese J. Math., 14 (2010), pp. 869–883.

[15] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann, *Vector spaces of linearizations for matrix polynomials*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 971–1004.

[16] R. B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1231.

[17] Y. Nakatsukasa and F. Tisseur, *Eigenvector error bounds and perturbation for nonlinear eigenvalue problems*, in Manchester Workshop on Nonlinear Eigenvalue Problems (NEP14), Manchester, 2014.

[18] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computation*, Linear Algebra Appl., 58 (1984), pp. 391–405.

[19] A. Ruhe, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sci. Comput., 19 (1998), pp. 1535–1551.

[20] Y. Su, J. Zhang, and Z. Bai, *A compact Arnoldi algorithm for polynomial eigenvalue problems*, in Recent Advances in Numerical Methods for Eigenvalue Problems (RANMEP2008), Taiwan, 2008, Mathematical Society of the Republic of China (Taiwan), Taipei.

[21] R. Van Beeumen, K. Meerbergen, and W. Michiels, *A rational Krylov method based on Hermite interpolation for nonlinear eigenvalue problems*, SIAM J. Sci. Comput., 35 (2013), pp. A327–A350.

[22] R. Van Beeumen, W. Michiels, and K. Meerbergen, *Linearization of Lagrange and Hermite interpolating matrix polynomials*, IMA J. Numer. Anal., 35 (2015), pp. 909–930.

[23] Y. Zhang and Y. Su, *A memory-efficient model order reduction for time-delay systems*, BIT, 53 (2013), pp. 1047–1073.