SIAM J. SCI. COMPUT.
Vol. 35, No. 1, pp. A327–A350

A RATIONAL KRYLOV METHOD BASED ON HERMITE INTERPOLATION FOR NONLINEAR EIGENVALUE PROBLEMS*

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

Abstract. This paper proposes a new rational Krylov method for solving the nonlinear eigenvalue problem: $A(\lambda)x = 0$. The method approximates $A(\lambda)$ by Hermite interpolation where the degree of the interpolating polynomial and the interpolation points are not fixed in advance. It uses a companion-type reformulation to obtain a linear generalized eigenvalue problem (GEP). To this GEP we apply a rational Krylov method that preserves the structure. The companion form grows in each iteration and the interpolation points are dynamically chosen. Each iteration requires a linear system solve with $A(\sigma)$, where σ is the last interpolation point. The method is illustrated by small- and large-scale numerical examples. In particular, we illustrate that the method is fully dynamic and can be used as a global search method as well as a local refinement method. In the last case, we compare the method to Newton's method and illustrate that we can achieve an even faster convergence rate.

Key words. rational Krylov, Newton polynomials, Hermite interpolation, nonlinear eigenvalue problem

AMS subject classifications. 47J10, 65F15

DOI. 10.1137/120877556

1. Introduction.

Consider the nonlinear eigenvalue problem (NLEP)

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

where $\lambda \in \Omega \subseteq \mathbb{C}$, $A : \Omega \to \mathbb{C}^{n\times n}$, and $x \in \mathbb{C}^n\backslash\{0\}$. We assume that A is analytic in Ω. This eigenvalue problem has been extensively studied in the literature. See, e.g., [11, 18]. There are specialized methods for different types of structures of $A(\lambda)$ [16, 25], but the goal of this paper is to address the solution of the more general NLEP (1.1). We present a general algorithmic framework, applicable to a large class of NLEPs allowing us to find eigenvalues and eigenvectors of (1.1) close to given targets. The proposed method is applicable as a global search method, in order to find eigenvalues in a region, as well as a local refinement method, in order to improve the accuracy of a few eigenpairs.

A first possible approach is to use a Newton type method. In the literature, we find examples as the residual inverse iteration method [19], the Jacobi–Davidson type projection method [5], and the block Newton method [13]. A second approach is first to derive a polynomial or rational approximation of $A(\lambda)$ and then solve the approximate eigenvalue problem by a standard technique, e.g., [9]. A third approach is based on contour integration techniques, which project the nonlinear problem onto a small dimensional one for the interesting eigenvalues; see, e.g., [7, 6, 10].

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 16, 2012; accepted for publication (in revised form) November 19, 2012; published electronically January 15, 2013. This work 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 projects STRT1-09/33 and OT/10/038 of the KU Leuven Research Council, and by project G.0712.11N of the Research Foundation-Flanders (FWO).

http://www.siam.org/journals/sisc/35-1/87755.html

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

The method proposed in the current paper is inspired by the infinite Arnoldi method for the NLEP [12]. However, there are several main differences. First, the infinite Arnoldi method builds a Krylov space on an equivalent linear *infinite dimensional* eigenvalue problem. Second, implementing this method requires the coefficients of the Taylor series of $A(\lambda)$ developed around a given fixed shift σ. In this paper, we use a (Hermite) interpolating polynomial of low degree to approximate $A(\lambda)$. The expectations are that a better approximation of (1.1) can be obtained than a truncated Taylor expansion of the same degree.

Polynomial interpolation of $A(\lambda)$ in the interpolation points $\sigma_0 \leq \sigma_1 \leq \cdots \leq \sigma_N$ results in a linearization of (1.1), which can be reformulated as a generalized eigenvalue problem (GEP)

$$\mathcal{A}y = \lambda \mathcal{B}y, \tag{1.2}$$

where $\mathcal{A}, \mathcal{B} \in \mathbb{C}^{(N+1)n \times (N+1)n}$ and $y \in \mathbb{C}^{(N+1)n}$. For solving (1.2) we use the rational Krylov method [22]. One of the advantages of the rational Krylov method, compared to the Arnoldi method, is that the shift can be changed at every iteration.

We use Newton polynomial bases. The advantage is that adding a new interpolation point just adds a new polynomial to the basis. This allows for iteratively adding new points in a flexible way, which implies that the linearization grows in every iteration. When, in addition, we choose the shifts of the rational Krylov method equal to the interpolation points, the rational Krylov expansion on the linearized problem takes advantage of specific structure, so that the rational Krylov method can be interpreted as a rational Krylov method applied to a fixed size matrix (that does not grow during the iterations). This property makes the process dynamic and has the important consequence that the interpolation points need not be fixed in advance. In each iteration we can choose a new interpolation point based on the results of the previous ones.

Matrices $\mathcal{A}$ and $\mathcal{B}$ have special sparse structure which can efficiently be exploited. Our method only involves linear algebra operations with matrices of dimension $n \times n$, which makes it suitable for large and sparse matrix operations $A(\lambda)$. More precisely, in each iteration we only need to solve one matrix-vector equation of dimension n and therefore one LU-decomposition should be computed. This is the LU-decomposition of $A(\sigma_j)$, which is the nonlinear matrix function evaluated in the last interpolation point. In the case when $\sigma_j = \sigma_{j-1} = \cdots = \sigma_{j-l+1}$ (Hermite interpolation) we can reuse the LU-decomposition for l successive iterations, which significantly reduces the computational cost. Also the possible low rank structure of the coefficient matrices of the interpolating polynomial can be exploited to reduce the storage significantly.

This paper is organized as follows. Section 2 discusses the linearization of $A(\lambda)$, which results in a companion-type reformulation of the NLEP. Section 3 reviews the standard rational Krylov method for the GEP. Section 4 introduces the new rational Krylov method for solving the NLEP. Section 5 illustrates the proposed algorithm with some numerical examples and compares it to other methods. Finally, the main conclusions are summarized in section 6.

Throughout the paper, we denote by A^* the conjugate transpose of the matrix A. V_j denotes a matrix with j columns and $A_{j,k}$ is a matrix of dimensions $j \times k$. We omit subscripts when the dimensions of the matrices are clear from the context. Column j of the matrix V is denoted by v_j and row k by v_k^*. A superscript such as $\lambda^{(j)}$ is used to distinguish the value at step j of a quantity that changes from iteration to iteration. A superscript as in $v^{[j]}$ denotes the jth block of the block vector v. With $\|\cdot\|$ we denote the 2-norm.

2. Linearization and companion-type formulation. In order to solve the NLEP (1.1), we first approximate $A(\lambda)$ by a matrix polynomial. Therefore, we use Newton polynomials in the current paper. This results in a linearization which can be expressed in a companion-type matrix form. Next, these matrices are further used in the rational Krylov method.

We can always write a matrix function $A(\lambda) \in \mathbb{C}^{n\times n}$ as follows:

$$A(\lambda) = \sum_{i=1}^{m} B_i f_i(\lambda), \tag{2.1}$$

where $B_i \in \mathbb{C}^{n\times n}$ are constant matrices, $f_i(\lambda)$ are scalar functions of λ, and $m \leq n^2$. However, in many applications, such as time-delay eigenvalue problems, $m \ll n^2$. Other applications are mathematical models that are linear in λ but adopt nonlinear boundary conditions, which leads to low rank B_i and $m \ll n$; see section 5.2 for an example.

In this case, we can greatly reduce the storage and computational cost because we only have to compute m scalar functions and store m constant matrices instead of an $n \times n$ matrix function. We now review polynomial interpolation in Newton and Hermite form for the scalar case in sections 2.1 and 2.2, respectively.

2.1. Newton polynomial basis. The interpolating polynomial in Newton form of the function $f(\lambda)$ is a linear combination of the Newton polynomials

$$p_N(\lambda) := \sum_{i=0}^{N} \alpha_i n_i(\lambda), \tag{2.2}$$

where the Newton polynomials are defined as

$$\begin{aligned} n_0(\lambda) &:= 1, \\ n_i(\lambda) &:= \prod_{j=0}^{i-1} (\lambda - \sigma_j), \qquad i = 1, 2, \ldots, \end{aligned} \tag{2.3}$$

and the coefficients α_i are the divided differences

$$\begin{aligned} \alpha_0 := f[\sigma_0] &= f(\sigma_0), \\ \alpha_i := f[\sigma_0, \ldots, \sigma_i] &= \frac{f[\sigma_1, \ldots, \sigma_i] - f[\sigma_0, \ldots, \sigma_{i-1}]}{\sigma_i - \sigma_0}, \end{aligned} \tag{2.4}$$

where $\sigma_0, \sigma_1, \ldots$ are distinct interpolation points. These divided differences can be computed efficiently from a divided differences table.

2.2. Hermite interpolation. The interpolating polynomial in Hermite form is again a linear combination of the Newton polynomials (2.2), but it also interpolates higher order derivatives. For computing the divided differences (2.4), a small modification has to be made to avoid division by zero. More precisely, assuming that the same interpolation points can only be used in a successive way, we can compute

$$f[\underbrace{\sigma_i, \ldots, \sigma_i}_{j+1 \text{ times}}] = \frac{f^{(j)}(\sigma_i)}{(j!)}.$$

In this way, we can again use a divided differences table to calculate the α_i in (2.2).

2.3. A companion-type reformulation. Let the scalar functions $f_i(\lambda)$ of (2.1) be approximated by interpolating Newton polynomials (2.3). Then,

$$P_N(\lambda) := \sum_{j=1}^{m} B_j \sum_{i=0}^{N} \alpha_{ij} n_i(\lambda) = \sum_{i=0}^{N} \left(\sum_{j=1}^{m} \alpha_{ij} B_j \right) n_i(\lambda) =: \sum_{i=0}^{N} A_i n_i(\lambda), \tag{2.5}$$

where α_{ij} are scalars and $A_i \in \mathbb{C}^{n\times n}$ is the matrix polynomial which (Hermite) interpolates $A(\lambda)$ in the interpolation points $\sigma_0, \sigma_1, \ldots, \sigma_N$. Using (2.5), the polynomial eigenvalue problem (PEP)

$$P_N(\lambda)x = 0 \tag{2.6}$$

can now be linearized. By linearization, we mean here the transformation of (2.6) into a GEP $Ax = \lambda Bx$ by a suitable choice of the matrices A and B such that there is a one-to-one correspondence between the eigenpairs of (2.6) and the eigenpairs of the $A - \lambda B$ pencil [2, 16]. The results of this linearization are now summarized in the following theorem.

THEOREM 2.1. *The pair* $(\lambda, x \neq 0)$ *is an eigenpair of the PEP* (2.6) *if and only if*

$$\mathcal{A}_N y_N = \lambda \mathcal{B}_N y_N, \tag{2.7}$$

where

$$\mathcal{A}_N = \begin{bmatrix} A_0 & A_1 & A_2 & \ldots & A_N \\ \sigma_0 I & I & & & \\ & \sigma_1 I & I & & \\ & & \ddots & \ddots & \\ & & & \sigma_{N-1} I & I \end{bmatrix}, \quad \mathcal{B}_N = \begin{bmatrix} 0 & & & & \\ I & 0 & & & \\ & I & 0 & & \\ & & \ddots & \ddots & \\ & & & I & 0 \end{bmatrix}, \tag{2.8}$$

and

$$y_N = \mathrm{vec}(x,\, n_1(\lambda)x,\, n_2(\lambda)x,\, \cdots,\, n_N(\lambda)x).$$

COROLLARY 2.2. *If we want to include an additional interpolation point,* $\mathcal{A}_{N+1}$ *and* $\mathcal{B}_{N+1}$ *are obtained by adding one block column to the right and one block row at the bottom of the matrices* $\mathcal{A}_N$ *and* $\mathcal{B}_N$*, respectively.*

This property will be very useful in iterative methods, as we will see in section 4.

3. Rational Krylov method. The rational Krylov method [21, 22] is a generalization of the shifted and inverted 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).

3.1. Algorithm. We now review the rational Krylov method and derive its recurrence relation. The standard rational Krylov algorithm [22] is outlined in Algorithm 1.

By eliminating w at the jth iteration we get the relation

$$V_{j+1} h_j = (A - \sigma_j B)^{-1} B V_j t_j,$$

ALGORITHM 1. Rational Krylov method.

1 Choose vector v_1, where $\|v_1\| = 1$.
for $j = 1, 2, \ldots$ **do**
2 Choose shift: σ_j.
3 Set continuation combination following (3.3): t_j.
4 Set continuation vector: $w = V_j t_j$.
5 Compute: $w := (A - \sigma_j B)^{-1} B w$.
6 Orthogonalize: $w := w - V_j h_j$, where $h_j = V_j^* w$.
7 Get new vector: $v_{j+1} = w/h_{j+1,j}$, where $h_{j+1,j} = \|w\|$.
8 Compute eigenpairs: $\left(\lambda_i^{(j)}, s_i^{(j)}\right)$ and test for convergence.
end
9 Compute eigenvectors: $x_i = V_{j+1} H_{j+1,j} s_i$.

where h_j is a vector of length $j+1$. This is equivalent to

$$AV_{j+1}h_j = BV_{j+1}(h_j\sigma_j + t_j),$$

where at the bottom of the vector t_j a zero is added to give it length $j+1$. Combining now all the previous iterations, we arrive at the basic recurrence relation of the rational Krylov method

$$AV_{j+1}H_{j+1,j} = BV_{j+1}K_{j+1,j}, \tag{3.1}$$

where $H_{j+1,j}$ and $K_{j+1,j}$ are two $(j+1) \times j$ upper Hessenberg matrices. $H_{j+1,j}$ contains the coefficients of the Gram–Schmidt orthogonalization process and

$$K_{j+1,j} = H_{j+1,j}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_j) + T_{j+1,j}, \tag{3.2}$$

where the upper triangular matrix $T_{j+1,j}$ is built up from the continuation combinations $t_1, \ldots, t_j$. Note that from the definition of $K_{j+1,j}$ in (3.2), we can easily find that

$$\sigma_j = \frac{k_{j+1,j}}{h_{j+1,j}}.$$

DEFINITION 3.1 (rational Krylov subspace). *A rational Krylov subspace is a subspace built by the rational Krylov method and is spanned by*

$$v_1,\ (A - \sigma_1 B)^{-1} B w_1,\ (A - \sigma_2 B)^{-1} B w_2, \ldots,$$

where $w_j = V_j t_j$.

We select the continuation combination t_j (step 3 of Algorithm 1) as suggested in [22],

$$t_j = \begin{cases} e_j, & \sigma_j = \sigma_{j-1}, \\ q_j = Q_j e_j, & \sigma_j \neq \sigma_{j-1}, \end{cases} \tag{3.3}$$

where $t_1 := e_1$ and Q_j is obtained from the QR factorization of

$$Q_j R_{j,j-1} = (K_{j,j-1} - \sigma_j H_{j,j-1}).$$

3.2. Orthogonalization. For the orthogonalization in step 6 of Algorithm 1, classical iterative Gram–Schmidt with reorthogonalization is used. In each iteration step j, we assume that $h_{j+1,j} \neq 0$. Then, we call $H_{j+1,j}$ *unreduced*. If $h_{j+1,j} = 0$, the Range(V_j) is an invariant subspace and

$$AV_jH_{j,j} = BV_jK_{j,j},$$

where $H_{j,j}$ and $K_{j,j}$ are the $j \times j$ upper parts of $H_{j+1,j}$ and $K_{j+1,j}$, respectively. At this point, the Gram–Schmidt orthogonalization process fails.

3.3. Computing approximate eigenpairs. Approximations for the eigenvalues and corresponding eigenvectors of the matrix pencil (A, B) can, in each iteration j of Algorithm 1, be obtained from the $j \times j$ upper parts of the two Hessenberg matrices $H_{j+1,j}$ and $K_{j+1,j}$.

DEFINITION 3.2. *Let* $(\lambda_i^{(j)}, s_i^{(j)})$ *satisfy*

$$K_{j,j}s_i^{(j)} = \lambda_i^{(j)}H_{j,j}s_i^{(j)}, \qquad s_i^{(j)} \neq 0. \tag{3.4}$$

Then we call $(\lambda_i^{(j)}, x_i^{(j)})$*, where*

$$x_i^{(j)} := V_{j+1}H_{j+1,j}s_i^{(j)}, \tag{3.5}$$

a Ritz pair of (A, B)*.*

For the rational Krylov method there are various ways to extract eigenvalues, but this is not the main purpose of this paper. Therefore, we consider here the standard Ritz values, although the theory can easily be extended to harmonic Ritz values [22].

3.4. Stopping criterion. The accuracy of a Ritz pair (λ, x) is typically estimated by the residual norm $\|Ax - \lambda Bx\|$. Let

$$r_i^{(j)} := Ax_i^{(j)} - \lambda_i^{(j)}Bx_i^{(j)}.$$

Using the substitution (3.5), the recurrence relation (3.1) and (3.4), respectively, yields

$$\begin{aligned} r_i^{(j)} &= AV_{j+1}H_{j+1,j}s_i^{(j)} - \lambda_i^{(j)}BV_{j+1}H_{j+1,j}s_i^{(j)} \\ &= BV_{j+1}\left(K_{j+1,j}s_i^{(j)} - \lambda_i^{(j)}H_{j+1,j}s_i^{(j)}\right) \\ &= B\begin{bmatrix} V_j & v_{j+1} \end{bmatrix}\begin{bmatrix} K_{j,j}s_i^{(j)} - \lambda_j^{(j)}H_{j,j}s_i^{(j)} \\ k_{j+1,j}e_j^*s_i^{(j)} - \lambda_j^{(j)}h_{j+1,j}e_j^*s_i^{(j)} \end{bmatrix} \\ &= Bv_{j+1}g_i^{(j)}, \end{aligned}$$

where $g_i^{(j)} := (k_{j+1,j} - \lambda_j^{(j)}h_{j+1,j})e_j^*s_i^{(j)}$. Thus, a simple check for convergence of a Ritz pair in step 8 of Algorithm 1 results in

$$\left|\frac{g_i^{(j)}}{\lambda_i^{(j)}}\right| \leq \varepsilon_{\text{tol}},$$

where ε_{tol} is defined by the user.

4. A rational Krylov method for the NLEP. In this section, we introduce a new rational Krylov method for the NLEP (1.1). This method uses the companion-type matrix formulation (2.7) which was obtained from the linearization by Hermite interpolation of the NLEP.

In order to maximally exploit the structure of the matrices of the GEP (2.7), we choose the shifts in the rational Krylov algorithm equal to the interpolation points of the interpolating polynomial $P_N(\lambda)$ (2.5) and use a specific set of starting vectors. Hence, we end up with an algorithm whose interpolation points can be chosen in a dynamical way.

Before we give the algorithm (section 4.2) and discuss its implementation (section 4.3), we introduce some definitions and properties for building the rational Krylov subspace (section 4.1) in order to support the elaboration in the remaining sections.

4.1. Building the rational Krylov subspace. Let us start with the following lemma.

LEMMA 4.1. *Let $\mathcal{A}_N$ and $\mathcal{B}_N$ be defined by* (2.8) *and*

$$y_j = \operatorname{vec}\left(y_j^{[1]}, y_j^{[2]}, \ldots, y_j^{[j+1]}, 0, \ldots, 0\right),$$

where $y_j \in \mathbb{C}^{(N+1)n}$ and $y_j^{[i]} \in \mathbb{C}^n$ for $i = 1, \ldots, j+1$. Then for all j, $0 \le j < N$, the solution x_j of the system

$$(\mathcal{A}_N - \sigma_j \mathcal{B}_N) x_j = y_j \tag{4.1}$$

has the following structure:

$$x_j = \operatorname{vec}\left(x_j^{[1]}, x_j^{[2]}, \ldots, x_j^{[j+1]}, 0, \ldots, 0\right),$$

where again $x_j \in \mathbb{C}^{(N+1)n}$ and $x_j^{[i]} \in \mathbb{C}^n$ for $i = 1, \ldots, j+1$.

Proof. We can expand (4.1) in the following block form:

$$\left[\begin{array}{cccc|cccc}
A_0 & A_1 & \ldots & A_j & A_{j+1} & A_{j+2} & \ldots & A_N \\
-\mu_0^{(j)} I & I & & & & & & \\
 & \ddots & \ddots & & & & & \\
 & & -\mu_{j-1}^{(j)} I & I & & & & \\
\hline
 & & & \mathbf{0} & I & & & \\
 & & & & -\mu_{j+1}^{(j)} I & I & & \\
 & & & & & \ddots & \ddots & \\
 & & & & & & -\mu_{N-1}^{(j)} I & I
\end{array}\right] x_j = \left[\begin{array}{c} y_j^{[1]} \\ y_j^{[2]} \\ \vdots \\ y_j^{[j+1]} \\ \hline 0 \\ 0 \\ \vdots \\ 0 \end{array}\right],$$

where

$$\mu_i^{(j)} = \sigma_j - \sigma_i, \qquad i = 0, 1, \ldots, N-1.$$

The zero block at the $(j+1)$st subdiagonal position yields a decoupling of the system (4.1). Because the right-bottom part of the matrix $\mathcal{A}_N - \sigma_j \mathcal{B}_N$ and the bottom part of y_j results in a zero solution, only the top part of x_j is nonzero. This proves the lemma. ∎

The main difference with the standard definition of the rational Krylov method is that the shifts are not free parameters, but they are already implicitly defined by the matrices $\mathcal{A}_N$ and $\mathcal{B}_N$ in (2.8). We now introduce the following definition.

DEFINITION 4.2. *Let $\mathcal{A}_N$ and $\mathcal{B}_N$ be defined by* (2.8)*. Then we define by*

$$\begin{aligned}\mathcal{RK}_k &:= \operatorname{span}\left\{v_1,\, (\mathcal{A}_N-\sigma_1\mathcal{B}_N)^{-1}\mathcal{B}_N w_1,\, (\mathcal{A}_N-\sigma_2\mathcal{B}_N)^{-1}\mathcal{B}_N w_2,\right.\\ &\qquad\qquad \left.\ldots,\, (\mathcal{A}_N-\sigma_{k-1}\mathcal{B}_N)^{-1}\mathcal{B}_N w_{k-1}\right\}\\ &:= \operatorname{span}\left\{v_1,\, v_2,\, v_3,\, \ldots,\, v_k\right\}\end{aligned} \tag{4.2}$$

the rational Krylov subspace of dimension $k \le N+1$ constructed by the matrices $\mathcal{A}_N$, $\mathcal{B}_N$ and the starting vector $v_1 \in \mathbb{C}^{(N+1)n}$, where $w_j = V_j t_j$ with $V_j = [v_1, v_2, \ldots, v_j]$ and t_j the continuation combination.

This definition holds for all starting vectors v_1, but the special structure of the matrices $\mathcal{A}_N$ and $\mathcal{B}_N$ can be exploited by choosing a particular starting vector of the following form:

$$v_1 := \operatorname{vec}\left(v_1^{[1]},\, 0,\, 0,\, \ldots,\, 0\right), \tag{4.3}$$

where $v_1 \in \mathbb{C}^{(N+1)n}$ and $v_1^{[1]} \in \mathbb{C}^n$. The results for choosing (4.3) as starting vector for (4.2) are now summarized in the following lemmas.

LEMMA 4.3. *Suppose that a starting vector v_1 of the form* (4.3) *is used for the rational Krylov method. Then for all j, $0 < j \le N$,*

$$v_{j+1} = (\mathcal{A}_N - \sigma_j \mathcal{B}_N)^{-1}\mathcal{B}_N w_j, \tag{4.4}$$

where $w_j = V_j t_j$, has the structure

$$v_{j+1} = \operatorname{vec}\left(v_{j+1}^{[1]},\, v_{j+1}^{[2]},\, \ldots,\, v_{j+1}^{[j+1]},\, 0,\, \ldots,\, 0\right),$$

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

Proof. We prove this lemma by induction and start with $j = 1$. Since $w_1 = v_1$, applying $\mathcal{B}_N$ to w_1 results in a downshift of the block $v_1^{[1]}$. The result of $(\mathcal{A}_N - \sigma_1\mathcal{B}_N)^{-1}\mathcal{B}_N w_1$ is the solution of the system

$$\begin{bmatrix} A_0 & A_1 & A_2 & A_3 & \cdots & A_N \\ (\sigma_0-\sigma_1)I & I & & & & \\ & \mathbf{0} & I & & & \\ & & (\sigma_2-\sigma_1)I & I & & \\ & & & \ddots & \ddots & \\ & & & & (\sigma_{N-1}-\sigma_1)I & I \end{bmatrix} v_2 = \begin{bmatrix} 0 \\ v_1^{[1]} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

and, based on Lemma 4.1, it has the form

$$v_2 = \operatorname{vec}\left(v_2^{[1]},\, v_2^{[2]},\, 0,\, 0,\, \ldots,\, 0\right).$$

Thus, only the first two blocks of v_2 are nonzero.

Now suppose that the lemma holds for $j-1$; then only the first j blocks of v_j are nonzero and thus so are the first j blocks of the continuation vector w_j. Applying $\mathcal{B}_N$ to w_j results again in a downshift of the first j nonzero blocks of w_j. The result

of $(\mathcal{A}_N - \sigma_j \mathcal{B}_N)^{-1}\mathcal{B}_N w_j$ is the solution of the system

$$\begin{bmatrix} A_0 & A_1 & \dots & A_j & A_{j+1} & A_{j+2} & \dots & A_N \\ -\mu_0^{(j)} I & I & & & & & & \\ & \ddots & \ddots & & & & & \\ & & -\mu_{j-1}^{(j)} I & I & & & & \\ & & & \mathbf{0} & I & & & \\ & & & & -\mu_{j+1}^{(j)} I & I & & \\ & & & & & \ddots & \ddots & \\ & & & & & & -\mu_{N-1}^{(j)} I & I \end{bmatrix} v_{j+1} = \begin{bmatrix} 0 \\ w_j^{[1]} \\ \vdots \\ w_j^{[j]} \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

where $\mu_i^{(j)} = \sigma_j - \sigma_i$, $i = 0, 1, \dots, N-1$, and, based on Lemma 4.1, v_{j+1} has the form

$$v_{j+1} = \operatorname{vec}\left(v_{j+1}^{[1]}, \dots, v_{j+1}^{[j+1]}, 0, 0, \dots, 0\right).$$

Thus, only the first $j+1$ blocks of v_{j+1} are nonzero, which proves the lemma. ∎

LEMMA 4.4. *Let the rational Krylov subspace $\mathcal{RK}_k$ be constructed as in Definition* 4.2 *and Lemma* 4.3. *Then, at each iteration j of the rational Krylov method, only the top-left parts of the matrices $\mathcal{A}_N - \sigma_j \mathcal{B}_N$ are used to compute the nonzero top parts of the vectors v_{j+1}, i.e.,*

$$(\mathcal{A}_j - \sigma_j \mathcal{B}_j)\tilde{v}_{j+1} = \mathcal{B}_j \tilde{w}_j, \tag{4.5}$$

where

$$\tilde{v}_{j+1} = \operatorname{vec}\left(v_{j+1}^{[1]}, v_{j+1}^{[2]}, \dots, v_{j+1}^{[j+1]}\right)$$

and

$$\tilde{w}_j = \operatorname{vec}\left(w_j^{[1]}, w_j^{[2]}, \dots, w_j^{[j]}, 0\right).$$

Proof. The proof of this lemma follows as a direct consequence of Lemmas 4.1 and 4.3. ∎

In each iteration j of the rational Krylov method, following Lemma 4.4, we only have to solve system (4.5) of dimension $(j+1)n \times (j+1)n$, instead of (4.4), which is of dimension $(N+1)n \times (N+1)n$. This results already in a significant reduction of the computation time. Also, the companion-type form of the matrix $\mathcal{A}_j - \sigma_j \mathcal{B}_j$ can be exploited to solve (4.5) efficiently with only operations on $n \times n$ sparse matrices. This results in the following lemma.

LEMMA 4.5. *The linear system* (4.5) *can be efficiently solved using the following equations:*

$$A(\sigma_j) v_{j+1}^{[1]} = y_0^{(j)}, \tag{4.6}$$

where

$$y_0^{(j)} = -\sum_{i=1}^{j} A_i \left(w_j^{[i]} + \sum_{k=1}^{i-1} \left(\prod_{l=k}^{i-1} \mu_l^{(j)} \right) w_j^{[k]} \right), \tag{4.7}$$

and

$$
\begin{aligned}
v_{j+1}^{[2]} &= w_j^{[1]} + \mu_0^{(j)} v_{j+1}^{[1]}, \\
v_{j+1}^{[3]} &= w_j^{[2]} + \mu_1^{(j)} v_{j+1}^{[2]}, \\
&\vdots \\
v_{j+1}^{[j+1]} &= w_j^{[j]} + \mu_{j-1}^{(j)} v_{j+1}^{[j]}.
\end{aligned} \tag{4.8}
$$

Proof. We start from (4.5),

$$
\begin{bmatrix}
A_0 & A_1 & A_2 & \dots & A_j \\
-\mu_0^{(j)} I & I & & & \\
& -\mu_1^{(j)} I & I & & \\
& & \ddots & \ddots & \\
& & & -\mu_{j-1}^{(j)} I & I
\end{bmatrix}
\begin{bmatrix} v_{j+1}^{[1]} \\ v_{j+1}^{[2]} \\ v_{j+1}^{[3]} \\ \vdots \\ v_{j+1}^{[j+1]} \end{bmatrix}
=
\begin{bmatrix} 0 \\ w_j^{[1]} \\ w_j^{[2]} \\ \vdots \\ w_j^{[j]} \end{bmatrix}.
$$

From the second block row we find

$$
v_{j+1}^{[2]} = w_j^{[1]} + \mu_0^{(j)} v_{j+1}^{[1]}. \tag{4.9}
$$

Substituting (4.9) in (4.5) and removing the second equation results in

$$
\begin{bmatrix}
A_0 + \mu_0^{(j)} A_1 & A_2 & A_3 & \dots & A_j \\
-\mu_0^{(j)} \mu_1^{(j)} I & I & & & \\
& -\mu_2^{(j)} I & I & & \\
& & \ddots & \ddots & \\
& & & -\mu_{j-1}^{(j)} I & I
\end{bmatrix}
\begin{bmatrix} v_{j+1}^{[1]} \\ v_{j+1}^{[3]} \\ v_{j+1}^{[4]} \\ \vdots \\ v_{j+1}^{[j+1]} \end{bmatrix}
=
\begin{bmatrix} -A_1 w_j^{[1]} \\ w_j^{[2]} + \mu_1^{(j)} w_j^{[1]} \\ w_j^{[3]} \\ \vdots \\ w_j^{[j]} \end{bmatrix}.
$$

By repeating this procedure j times we obtain

$$
\begin{aligned}
&\left(A_0 + \mu_0^{(j)} A_1 + \mu_0^{(j)} \mu_1^{(j)} A_2 + \mu_0^{(j)} \mu_1^{(j)} \mu_2^{(j)} A_3 + \ \cdots \ + \mu_0^{(j)} \mu_1^{(j)} \dots \mu_{j-1}^{(j)} A_j\right) v_{j+1}^{[1]} \\
&= -A_1 w_j^{[1]} - A_2 \left(w_j^{[2]} + \mu_1^{(j)} w_j^{[1]}\right) - A_3 \left(w_j^{[3]} + \mu_2^{(j)} w_j^{[2]} + \mu_1^{(j)} \mu_2^{(j)} w_j^{[1]}\right) - \cdots \\
&\quad - A_j \left(w_j^{[j]} + \mu_{j-1}^{(j)} w_j^{[j-1]} + \mu_{j-2}^{(j)} \mu_{j-1}^{(j)} w_j^{[j-2]} + \ \cdots \ + \mu_1^{(j)} \mu_2^{(j)} \dots \mu_{j-1}^{(j)} w_j^{[1]}\right).
\end{aligned} \tag{4.10}
$$

Note that the left-hand side of (4.10) is just the evaluation of (2.5) in the interpolation point σ_j and hence equal to $A(\sigma_j)$. The right-hand side of (4.10) is equal to (4.7). Thus (4.6) is proved.

For the remainder of the proof, we substitute $v_{j+1}^{[1]}$ into (4.5). Then, from the second block row of (4.5) we now obtain $v_{j+1}^{[2]}$. Next, $v_{j+1}^{[2]}$ is substituted in the third block row of (4.5), and so on. These subsequent substitutions are continued until we obtain $v_{j+1}^{[j+1]}$. ∎

COROLLARY 4.6. *In each iteration j of the rational Krylov method where we construct the rational Krylov subspace $\mathcal{RK}_k$, as defined in Definition* 4.2*, we only have to perform the LU-factorization of $A(\sigma_j) \in \mathbb{C}^{n \times n}$, instead of an LU-factorization of*

$\mathcal{A}_N - \sigma_j \mathcal{B}_N \in \mathbb{C}^{(N+1)n \times (N+1)n}$. *For Hermite interpolation, we can reuse the same LU-decomposition for successive iterations.*

PROPOSITION 4.7. *The Ritz values $\lambda_i^{(j)}$ computed in iteration j of the rational Krylov method are independent of N as long as $j < N$. These Ritz values are also independent of $\sigma_{j+1}, \ldots, \sigma_N$.*

Proof. In iteration j, the Ritz values are computed from the $j \times j$ upper parts of the two matrices $H_{j+1,j}$ and $K_{j+1,j}$. These Hessenberg matrices are obtained from the orthogonalization process of only $v_1, v_2, \ldots, v_{j+1}$. Following Lemmas 4.3–4.5, only the first $j+1$ interpolation points, $\sigma_0, \ldots, \sigma_j$, are used for the construction of the rational Krylov vectors V_{j+1}. Therefore, the approximated eigenvalues are independent of the interpolation points $\sigma_{j+1}, \ldots, \sigma_N$. Hence they are also independent of N, which proves the proposition. ∎

COROLLARY 4.8. *It is not necessary to choose either the interpolation points in advance or the degree of the interpolating polynomial. Instead, in each iteration we can choose the next interpolation point based on the results of the previous iterations. Therefore, the rational Krylov method can be implemented in an adaptive and an incremental way. The rational Krylov method is started with two interpolation points and can go on until convergence by adding an additional interpolation point in each iteration.*

Remark 4.9. Performing j steps of our method, with an appropriated starting vector, produces the same Krylov vectors as j steps of the standard rational Krylov method with the matrices $\mathcal{A}_N$ and $\mathcal{B}_N$ for any $N > j$. Taking a limit argument, $N \to \infty$, our method can be interpreted as a rational Krylov method directly applied to an infinite dimensional linear problem equivalent to the original nonlinear problem. See [12], where this connection is fully worked out for the Arnoldi method ($\sigma_j \equiv \sigma$). However, only finite arithmetic is used, i.e., standard linear algebra operations applied to matrices of finite size.

Remark 4.10. Another consequence of the independence of N is that we can restart the rational Krylov algorithm without any adaptation. Since at any iteration j, the method is independent of $\sigma_{j+1}, \sigma_{j+2}, \ldots$, we can return back to iteration $k < j$ and continue from this iteration with other interpolation points $\sigma_{k+1}, \sigma_{k+2}, \ldots$.

4.2. Algorithm. Based on Lemmas 4.3 – 4.5 and the important Corollary 4.8, the algorithm for solving the NLEP (1.1) can be implemented efficiently. Algorithm 2 gives the outline.

Each iteration step j of the algorithm can be subdivided into two main parts. First, the rational Krylov matrices $\mathcal{A}_{j-1}$ and $\mathcal{B}_{j-1}$ are extended with the next divided difference A_j to obtain $\mathcal{A}_j$ and $\mathcal{B}_j$, respectively. Also, the matrix V_j is extended with a zero block at the bottom. Second, a rational Krylov step with the extended matrices $\mathcal{A}_j$ and $\mathcal{B}_j$ is performed. This is graphically illustrated by Figure 4.1. Finally, in step 11 of Algorithm 2, the Ritz vectors s_i are multiplied on the left by $V_{j+1} H_{j+1,j}$ to obtain the approximate eigenvectors u_i.

4.3. Implementation details. We now discuss how to efficiently and reliably implement the operations in Algorithm 2 at lines 3, 7, and 11.

As mentioned in sections 2.1 and 2.2, the coefficients of the Newton and Hermite interpolating polynomials can be cheaply computed by using a divided differences table. However, this way of computing the coefficients can be numerically unstable. An alternative computation relies on a semianalytical operation.

Let m_i be the multiplicities of, respectively, the interpolation points σ_i, $i = 0, 1, \ldots,$ and $f(\lambda)$ the nonlinear function we want to interpolate. Then, we can expand

ALGORITHM 2. A rational Krylov method for the NLEP.

Choose shift σ_0 and starting vector v_1.
for $j = 1, 2, \ldots$ **do**
  EXPANSION PHASE:
    Choose shift: σ_j.
    Compute next divided difference: A_j.
    Expand $\mathcal{A}_j$, $\mathcal{B}_j$ and V_j.
  RATIONAL KRYLOV STEP:
    Set continuation combination following (3.3): t_j.
    Set continuation vector: $w = V_j t_j$.
    Apply: $w := (\mathcal{A}_j - \sigma_j \mathcal{B}_j)^{-1} \mathcal{B}_j w$.
    Orthogonalize: $w := w - V_j h_j$, where $h_j = V_j^* w$.
    Get new vector: $v_{j+1} = w/h_{j+1,j}$, where $h_{j+1,j} = \|w\|$.
    Compute eigenpairs: $\left(\lambda_i^{(j)}, s_i^{(j)}\right)$ and test for convergence.
end
Compute eigenvectors: $u_i = V_{j+1} H_{j+1,j} s_i$.

$f(\lambda)$ as

$$\begin{aligned} f(\lambda) = a_0 &+ a_1(\lambda-\sigma_0) + \cdots + a_{m_0}(\lambda-\sigma_0)^{m_0} \\ &+ a_{m_0+1}(\lambda-\sigma_0)^{m_0}(\lambda-\sigma_1) + \cdots + a_{m_0+m_1}(\lambda-\sigma_0)^{m_0}(\lambda-\sigma_1)^{m_1} \\ &+ a_{m_0+m_1+1}(\lambda-\sigma_0)^{m_0}(\lambda-\sigma_1)^{m_1}(\lambda-\sigma_2) + \cdots . \end{aligned} \tag{4.11}$$

For computing the coefficients a_i of (4.11) in a semianalytical way, we start from the Taylor series of $f(\lambda)$ of order $m_0 - 1$ around σ_0,

$$t_0(\lambda) = f(\sigma_0) + f'(\sigma_0)(\lambda-\sigma_0) + \ \cdots \ + \frac{f^{(m_0-1)}(\sigma_0)}{(m_0-1)!}(\lambda-\sigma_0)^{m_0-1}. \tag{4.12}$$

We now obtain from (4.12) the first m_0 coefficients of (4.11):

$$a_i = \frac{f^{(i)}(\sigma_0)}{i!}, \qquad i = 0, \ldots, m_0 - 1.$$

Next, we subtract $t_0(\lambda)$ from the original function $f(\lambda)$ and divide by $(\lambda - \sigma_0)^{m_0}$. This results in the new function

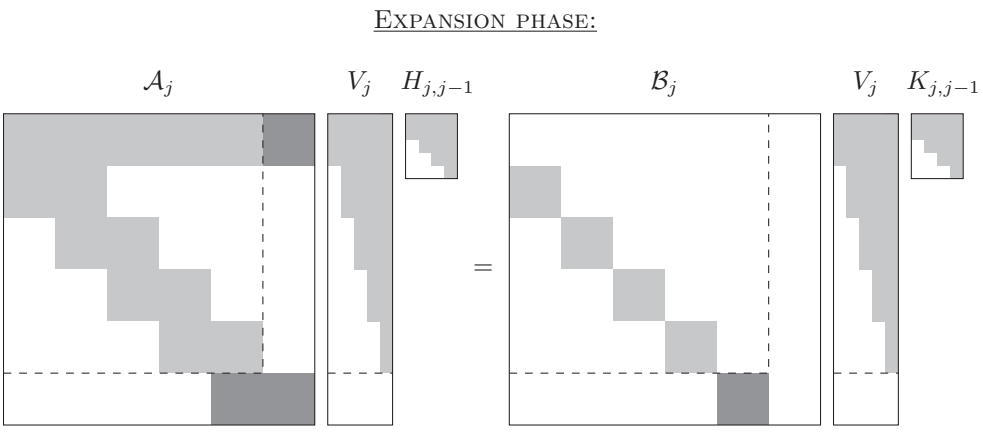
$$f_0(\lambda) = \frac{f(\lambda) - t_0(\lambda)}{(\lambda-\sigma_0)^{m_0}},$$

which is used to obtain the next m_1 coefficients of (4.11). Therefore, we compute the truncated Taylor series of $f_0(\lambda)$ of degree $m_1 - 1$ around σ_1,

$$t_1(\lambda) = f_0(\sigma_1) + f_0'(\sigma_1)(\lambda-\sigma_1) + \ \cdots \ + \frac{f_0^{(m_1-1)}(\sigma_1)}{(m_1-1)!}(\lambda-\sigma_1)^{m_1-1}.$$

We now have

$$a_{m_0+i} = \frac{f_0^{(i)}(\sigma_1)}{i!}, \qquad i = 0, \ldots, m_1 - 1.$$



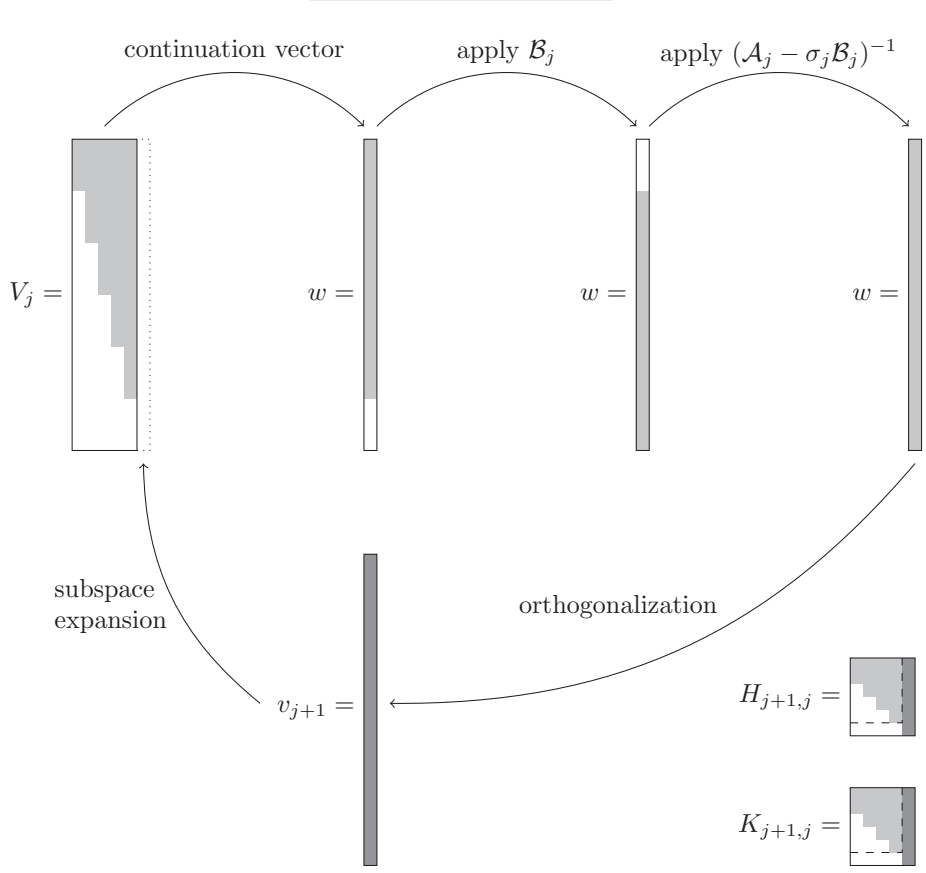


FIG. 4.1. *Visualization of Algorithm* 2.

Next, $t_1(\lambda)$ is subtracted from $f_0(\lambda)$ and divided by $(\lambda-\sigma_1)^{m_1}$, which yields the new function

$$f_1(\lambda) = \frac{f_0(\lambda) - t_1(\lambda)}{(\lambda-\sigma_1)^{m_1}} = \frac{\frac{f(\lambda)-t_0(\lambda)}{(\lambda-\sigma_0)^{m_0}} - t_1(\lambda)}{(\lambda-\sigma_1)^{m_1}}.$$

This process is repeated for all the interpolation points in order to obtain the remaining coefficients a_i of (4.11). The functions $f_0(\lambda), f_1(\lambda), \ldots$ and respective Taylor polynomials $t_0(\lambda), t_1(\lambda), \ldots$ can be computed by symbolic software tools, which we illustrate in the next example.

Example 4.11. In this example we consider the nonlinear function $f(\lambda) = \exp(-\lambda)$, which we want to interpolate in the interpolation points $\sigma_i = 0.1, 0.2, \ldots, 0.5$ with multiplicities $m_i = 4$. Figure 4.2 illustrates the evolution of the absolute values of the coefficients a_i obtained by the divided differences table and obtained in the semianalytical way by using the symbolic toolbox of MATLAB. This figure shows that for an interpolating polynomial of small degree, the two methods yield the same coefficients. However, at a certain point, the divided difference method start to suffer from numerical instabilities and the calculated coefficients diverge to infinity.

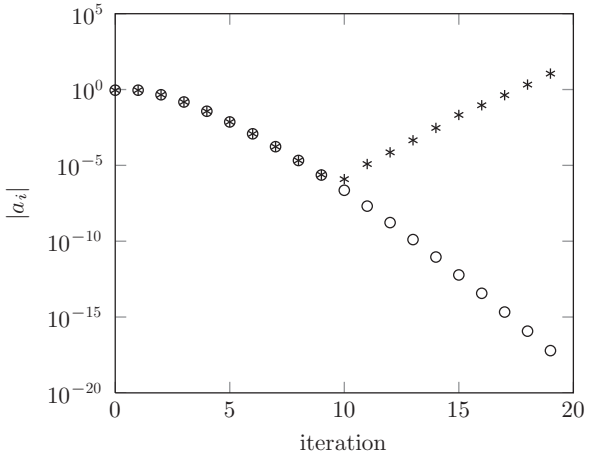


FIG. 4.2. *Graphical illustration of the evolution of the absolute values of the coefficients a_i of the Hermite interpolating polynomial of $f(\lambda) = \exp(-\lambda)$ in $\sigma_i = 0.1, 0.2, \ldots, 0.5$ all of multiplicity $m_i = 4$. The coefficients calculated by using divided differences are indicated by $*$, while those obtained in the semianalytical way are indicated by $\circ$.*

Step 7 of Algorithm 2 can be efficiently implemented. Based on Corollary 4.6, we first calculate the LU-decomposition of $A(\sigma_j)$. Next, $(\mathcal{A}_j - \sigma_j\mathcal{B}_j)^{-1}\mathcal{B}_j$ is applied as explained in Lemma 4.5. Note that in this case only one matrix vector solve of dimension n is needed in each iteration of Algorithm 2. Furthermore, since the matrices $\mathcal{A}_j$ and $\mathcal{B}_j$ are only used in step 7 and this step is efficiently performed, it is not necessary to build these matrices explicitly. We only have to store the coefficient matrices A_i. In the case the nonlinear matrix function $A(\lambda)$ is expressed in the form of (2.1), we only need to store the scalar coefficients α_{ij} of (2.5).

Breakdown of the rational Krylov method leads to an invariant subspace. However, possible breakdown of Algorithm 2 needs some attention and differs from breakdown in Algorithm 1.

PROPOSITION 4.12. *If $v_1 \neq 0$, then the shifts in Algorithm 2 can always be chosen such that the Gram–Schmidt orthogonalization process never causes a breakdown.*

Proof. A breakdown of the orthogonalization process in iteration j of Algorithm 2 occurs when the Range(V_j) is an invariant subspace. In other words, this can only be the case when $v_{j+1}^{[j+1]} = 0$. From (4.8) and $w_j^{(j)} \neq 0$, it follows that using $\sigma_j = \sigma_{j-1}$ always results in $v_{j+1}^{[j+1]} \neq 0$, which proves the proposition. ∎

However, when we choose the continuation vector as defined in (3.3) breakdown does not happen in practice. During all the numerical experiments we performed, no breakdown occurred.

4.4. Exploiting low rank structure of coefficient matrices. In several applications many of the matrices B_i in (2.1) are of low rank. Therefore, we now discuss how to exploit this low rank structure by using a different type of linearization.

Suppose that we can write $A(\lambda) \in \mathbb{C}^{n\times n}$ as follows:

$$A(\lambda) = \sum_{i=0}^{p} B_i \lambda^i + \sum_{i=1}^{m} C_i f_i(\lambda), \tag{4.13}$$

where $B_i, C_i \in \mathbb{C}^{n\times n}$ are constant matrices, $f_i(\lambda)$ are scalar functions of λ, $p \ll n^2$, and $m \ll n^2$. Furthermore, we assume that the matrices C_i have the rank-revealing decompositions

$$C_i = L_i U_i^*,$$

where $L_i, U_i \in \mathbb{C}^{n\times r_i}$ are of full column rank $r_i \ll n$.

Approximating the scalar functions $f_i(\lambda)$ of (4.13) by interpolating Newton polynomials results in the following matrix polynomial which interpolates $A(\lambda)$ in the interpolation points $\sigma_0, \sigma_1, \ldots, \sigma_N$:

$$\tilde{P}_N(\lambda) = \sum_{i=0}^{N} \tilde{A}_i n_i(\lambda) = \sum_{i=0}^{p} \left(\tilde{B}_i + \tilde{C}_i\right) n_i(\lambda) + \sum_{i=p+1}^{N} \tilde{C}_i n_i(\lambda), \tag{4.14}$$

where

$$\tilde{B}_i = \sum_{j=0}^{p} \beta_{ij} B_j,$$
$$\tilde{C}_i = \sum_{j=1}^{m} \gamma_{ij} C_j = \sum_{j=1}^{m} \gamma_{ij} L_j U_j^*,$$

with β_{ij} and γ_{ij} scalars. Define

$$\tilde{L}_i = \begin{bmatrix} \gamma_{i1}L_1 & \gamma_{i2}L_2 & \cdots & \gamma_{im}L_m \end{bmatrix},$$
$$\tilde{U} = \begin{bmatrix} U_1 & U_2 & \cdots & U_m \end{bmatrix},$$

where the size of $\tilde{L}_i$ and $\tilde{U}$ is $n \times r$ and $r = r_1 + r_2 + \cdots + r_m$.

Similar to Theorem 2.1, we obtain a companion-type reformulation where the pair $(\lambda, x \neq 0)$ is an eigenpair of the PEP (4.14) if and only if

$$\tilde{\mathcal{A}}_N \tilde{y}_N = \lambda \tilde{\mathcal{B}}_N \tilde{y}_N,$$

where

$$\tilde{\mathcal{A}}_N = \begin{bmatrix} \tilde{B}_0+\tilde{C}_0 & \tilde{B}_1+\tilde{C}_1 & \dots & \tilde{B}_p+\tilde{C}_p & \tilde{L}_{p+1} & \tilde{L}_{p+2} & \dots & \tilde{L}_N \\ \sigma_0 I & I & & & & & & \\ & \ddots & \ddots & & & & & \\ & & \sigma_{p-1}I & I & & & & \\ & & & \sigma_p\tilde{U}^* & I & & & \\ & & & & \sigma_{p+1}I & I & & \\ & & & & & \ddots & \ddots & \\ & & & & & & \sigma_{N-1}I & I \end{bmatrix}$$

and

$$\tilde{\mathcal{B}}_N = \begin{bmatrix} 0 & & & & & & & \\ I & 0 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & I & 0 & & & & \\ & & & \tilde{U}^* & 0 & & & \\ & & & & I & 0 & & \\ & & & & & \ddots & \ddots & \\ & & & & & & I & 0 \end{bmatrix}, \qquad \tilde{y}_N = \begin{bmatrix} x \\ n_1(\lambda)x \\ \vdots \\ n_p(\lambda)x \\ n_{p+1}(\lambda)\tilde{U}^*x \\ n_{p+2}(\lambda)\tilde{U}^*x \\ \vdots \\ n_N(\lambda)\tilde{U}^*x \end{bmatrix}.$$

For this type of linearization we can also prove Lemmas 4.1–4.5.

4.5. Connection with Newton's method. We are now ready to discuss a connection of Algorithm 2 with Newton's method. In each iteration of Newton's method, the NLEP (1.1) can be written as follows:

$$A(\lambda_{j+1})x_{j+1} = A(\lambda_j + \Delta\lambda_j)(x_j + \Delta x_j) \approx 0,$$

where $\Delta\lambda_j = \lambda_{j+1} - \lambda_j$ and $\Delta x_j = x_{j+1} - x_j$. Using a first order approximation of $A(\lambda_j + \Delta\lambda_j)$ results in

$$[A(\lambda_j) + A'(\lambda_j)\Delta\lambda_j](x_j + \Delta x_j) \approx 0, \tag{4.15}$$

and by omitting the higher order term, $O(\Delta\lambda_j \Delta x_j)$, we deduce

$$A(\lambda_j)(x_j + \Delta x_j) + A'(\lambda_j)x_j\Delta\lambda_j \approx 0. \tag{4.16}$$

Using (4.16) we define

$$x_{j+1} := -A(\lambda_j)^{-1}A'(\lambda_j)x_j,$$

where we have omitted the factor $\Delta\lambda_j$, since we normalize x_{j+1} in each iteration. By multiplying (4.15) on the left with x_{j+1}^*, we find the Newton update for the approximate eigenvalue

$$\lambda_{j+1} := \lambda_j - \frac{x_{j+1}^* A(\lambda_j)x_{j+1}}{x_{j+1}^* A'(\lambda_j)x_{j+1}}.$$

The connection between the linear rational Krylov algorithm and the Jacobi–Davidson algorithm [23] is illustrated in [22]. From [24], we also know that each step

of the Jacobi–Davidson iteration method can be interpreted as a Newton update. Since the rational Krylov method is a subspace method, it is generally accepted that using Ritz values as shifts reaches asymptotically (super) quadratic convergence.

As mentioned in Remark 4.9, Algorithm 2 can also be interpreted as a standard linear rational Krylov method applied to the matrices $\mathcal{A}_N$ and $\mathcal{B}_N$, which are obtained from a linearization with the shifts as interpolation points. Using Ritz values as shifts, $A'(\lambda_j)$ is approximated better and better in each iteration. In the real case, using the mean value theorem for divided differences, the approximation error vanishes exponentially, since

$$A'(\lambda_j) - P_j'(\lambda_j) = O\big(n_j(\lambda_j)\big) = O\big((\lambda_j - \lambda_0)(\lambda_j - \lambda_1)\cdots(\lambda_j - \lambda_{j-1})\big).$$

Therefore, we also expect an asymptotically (super) quadratic convergence for Algorithm 2. This is illustrated in section 5.3.

5. Numerical examples. We first illustrate the method as a root finder of a scalar equation. We show the difference between interpolation in Leja points [14] and Hermite interpolation. Then, we apply the rational Krylov method to two applications. For the `'gun'` problem, we first perform a global eigenvalue search and compare with the single-shift variant, which is, in fact, the shift-and-invert Arnoldi method; we then use the algorithm for local correction and compare with Newton's method. For a mechanical application, we use the algorithm for computing the eigenvalues in a specific region by global eigenvalue search. All numerical experiments are performed in MATLAB version 7.11.0 (R2010b) on a Dell Latitude with an Intel Core i5-2540M CPU @ 2.60 GHz quad core processor with 4 GB RAM.

5.1. Interpolation in Leja points versus Hermite interpolation. We start with the scalar NLEP

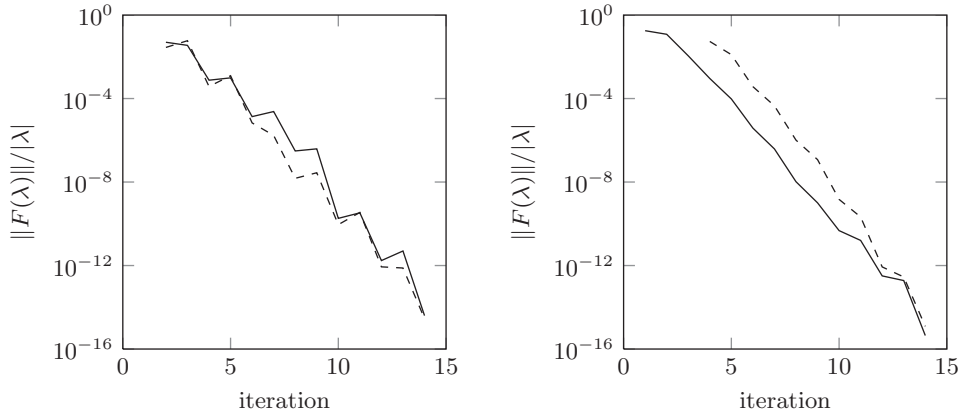
$$F(\lambda) = 3 + e - 3\lambda + \lambda^2 - e^{\lambda-1} - e^{2-\lambda} = 0$$

and suppose we are interested in the real roots in the interval $[0, 3]$, which are $\lambda_1 = 1$ and $\lambda_2 = 2$.

A first possible technique for selecting the shifts is choosing the shifts in Leja fashion [8, 14, 3] in the interval of interest. Leja points have the property that their limit distribution on an interval is the same as the limit distribution of the zeros of shifted and scaled Chebyshev polynomials for the same interval. The convergence history of λ_1 and λ_2, computed by Algorithm 2 with Leja points in the interval $[0, 3]$, is shown in Figure 5.1(a). This figure illustrates that after some iterations the eigenvalues start to converge. This happens when the approximation by the interpolating polynomials is improving in the neighborhood of the eigenvalues.

Another possibility is using Hermite interpolation in a few points chosen in the interval of interest. Here, we chose the interpolation points 0.5, 1.5, and 2.5, all with multiplicity 5. The corresponding convergence history for λ_1 and λ_2, computed by Algorithm 2, is shown in Figure 5.1(b). This figure shows that the eigenvalue $\lambda_1 = 1$ almost immediately starts to converge, since this eigenvalue is closest to the first interpolation points. When the algorithm proceeds and the interpolation points move toward the right, the eigenvalue $\lambda_2 = 2$ also starts converging.

Remark 5.1. All the eigenvectors are equal to 1 in the scalar case. Since $n = 1$, the matrix V, which is constructed in Algorithm 2, is the identity matrix. Therefore, it is only necessary to store the Hessenberg matrices H and K from which the approximate eigenvalues are computed.



(a) Newton interpolation in Leja points (b) Hermite interpolation

FIG. 5.1. *Convergence history for λ_1 (solid line) and λ_2 (dashed line) of the scalar NLEP with* (a) *Newton interpolation in Leja points in the interval* $[0, 3]$ *and* (b) *Hermite interpolation in the points* $0.5, 1.5$*, and* 2.5*, all with multiplicity* 5.

Remark 5.2. If Algorithm 2 is used to find the roots of a polynomial of degree k, then after k iterations the k roots are found exactly (in exact arithmetic). Moreover, since breakdown can always be avoided (see Proposition 4.12), additional iterations produce additional roots, which are all infinite.

5.2. Gun problem: Global eigenvalue search. We consider the `'gun'` problem of the Manchester collection of NLEPs [4]. This is a large-scale NLEP that models a radio frequency gun cavity and is of the form

$$F(\lambda)x = \left(K - \lambda M + i\sqrt{\lambda - \sigma_1^2}W_1 + i\sqrt{\lambda - \sigma_2^2}W_2\right)x = 0, \tag{5.1}$$

where M, K, W_1, and W_2 are real symmetric matrices of size 9956×9956, K is positive semidefinite, and M is positive definite. As in [4], we take $\sigma_1 = 0$ and $\sigma_2 = 108.8774$. The notation of the complex square root, $\sqrt{\cdot}$, denotes the principal branch. The domain of interest is such that $\mathrm{Im}(\lambda) \geq 0$ and $\mathrm{Re}(\lambda)$ is bounded away from the branch points $\lambda = 0$ and $\lambda = \sigma_2^2$ [15].

As in [12], we first shifted and scaled the original problem (5.1) by $\lambda = \gamma\hat{\lambda} + \mu$, such that the region of interest was transformed to be roughly within the unit circle. Therefore, we chose $\gamma = 300^2 - 200^2$ and $\mu = 250^2$. We obtained the transformed NLEP

$$\hat{F}(\hat{\lambda})x = \left(K - (\gamma\hat{\lambda} + \mu)M + i\sqrt{\gamma\hat{\lambda} + \mu - \sigma_1^2}W_1 + i\sqrt{\gamma\hat{\lambda} + \mu - \sigma_2^2}W_2\right)x = 0, \tag{5.2}$$

which was solved by Algorithm 2.

For measuring the convergence of an approximate eigenpair (λ, x), we used, as defined in [15], the relative residual norm

$$E(\lambda, x) = \frac{\|F(\lambda)x\|_2}{(\|K\|_1 + |\lambda|\|M\|_1 + \sqrt{|\lambda - \sigma_1^2|}\|W_1\|_1 + \sqrt{|\lambda - \sigma_2^2|}\|W_2\|_1)\|x\|_2}.$$

Since the domain of interest of the transformed NLEP (5.2) is roughly the top half of the unit circle, we took five interpolation points, each of them with multiplicity 12, almost uniformly distributed in this half circle. The interpolation points are shown by

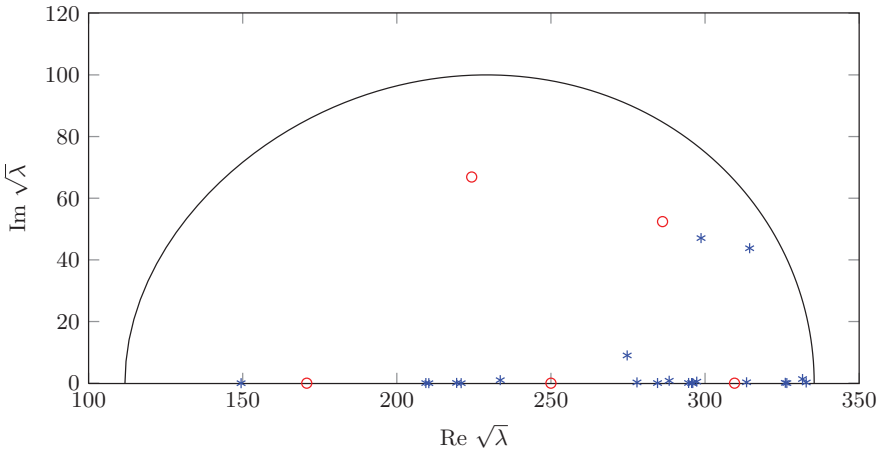


FIG. 5.2. *Visualization of the results of the simulation of the 'gun' problem. The approximate eigenvalues ∗ and the interpolation points ∘ are shown for the original NLEP.*

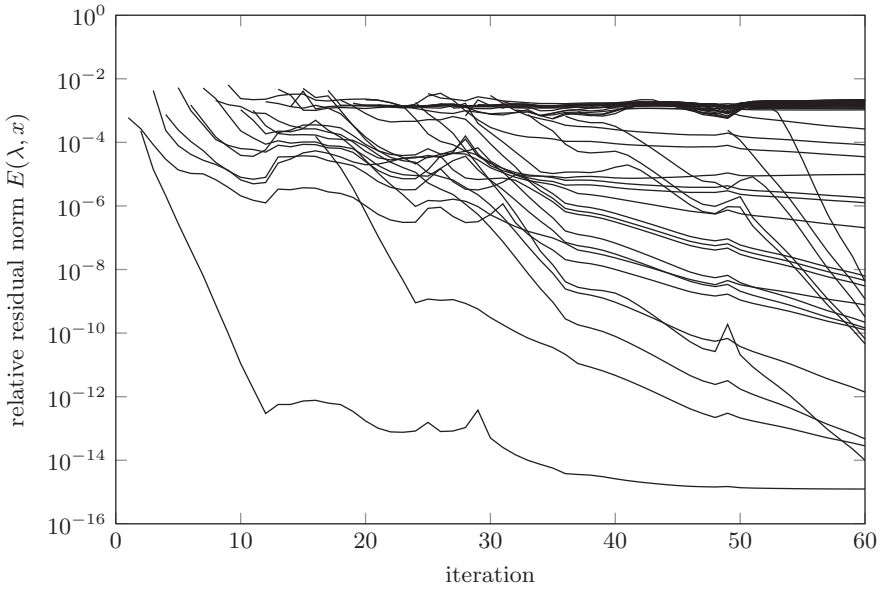


FIG. 5.3. *Convergence history for the 'gun' problem solved with Algorithm* 2 *with the shifts as indicated in Figure* 5.2.

∘ in Figure 5.2. The convergence history of Algorithm 2 is given in Figure 5.3. From this figure, we see that 24 eigenvalues show decreasing residual curves. The square roots of the corresponding 21 approximate eigenvalues, which are lying in the domain of interest, are shown in Figure 5.2.

The 60 iterations of Algorithm 2 required 40.7 seconds all together. From this time, 73 % was used for the Gram–Schmidt orthogonalization process with reorthogonalization, 13 % for the LU decompositions, and 6 % for the system solves described in Lemma 4.5.

We repeated this experiment with the low rank exploiting version of Algorithm 2, discussed in section 4.4. The corresponding convergence history is shown in Figure 5.4

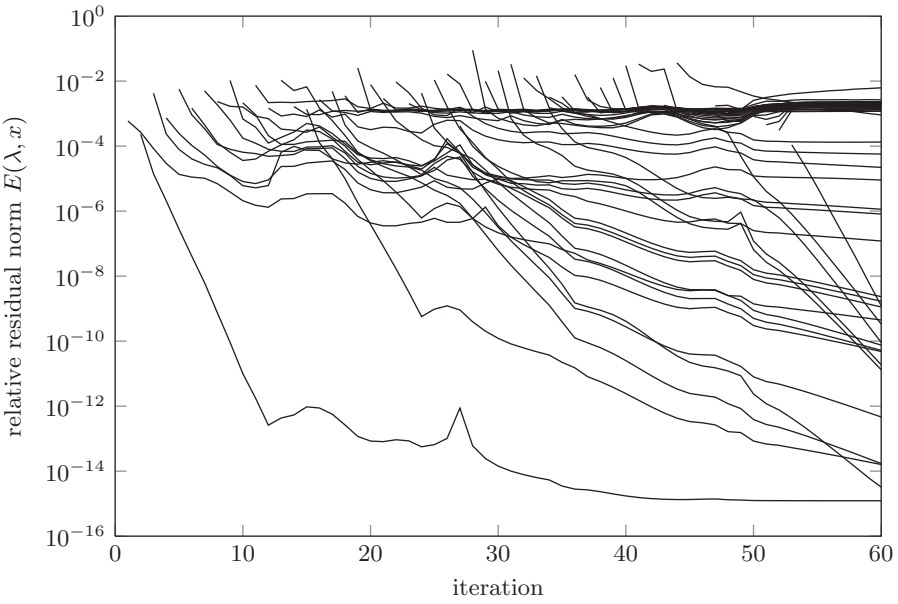


FIG. 5.4. *Convergence history for the 'gun' problem solved with the low rank exploiting version of Algorithm* 2 *with the shifts as indicated in Figure* 5.2.

and is very similar to the convergence history of the standard implementation of Algorithm 2. However, the total computation time is now only 8.9 seconds, which is remarkably less than without low rank exploiting.

A comparison of Algorithm 2 and the low rank exploiting version of Algorithm 2 is given in Table 5.1. From this table, it follows that the significant reduction in computation time is due to the orthogonalization process. Indeed, in each iteration $j > p$ of the low rank exploiting version, the vectors are of dimension $(p+1)n+(j-p)r$. On the other hand, in the version without low rank exploiting, the vectors are of dimension $(j+1)n$. In this problem $n = 9956$, $p = 1$, and $r = 84$, which explains the low computational cost of the orthogonalization process in the low rank exploiting version of Algorithm 2.

TABLE 5.1
Computation times for the 'gun' problem solved with Algorithm 2 *and with the low rank exploiting version of Algorithm* 2.

	Algorithm 2	Algorithm 2 + low rank
LU decompositions	5.3 s	5.3 s
Linear system solves	2.4 s	2.4 s
Gram–Schmidt orth. + reorth.	29.7 s	1.2 s
Total	40.7 s	8.9 s

To illustrate the importance of using multiple interpolation points in the global eigenvalue search strategy, we solved the transformed NLEP (5.2) with Algorithm 2 with the same constant shift, i.e., we used the shift and inverted Arnoldi method. Table 5.2 shows the number of eigenvalues with relative residual norm $E(\lambda, x) \leq 10^{-4}$. This table indicates that the number of accurately computed eigenvalues strongly depends on the choice of shift σ and is always smaller than the number of eigenvalues computed by Algorithm 2 with multiple shifts (see Figure 5.2).

TABLE 5.2

The number of accurate eigenvalues for the 'gun' problem for five runs of Algorithm 2 with different single constant shift σ. The number of iterations is fixed at 60 and $\#\lambda$ denotes the number of eigenvalues with relative residual norm $E(\lambda, x) < 10^{-4}$. The number between parentheses is the number of accurate eigenvalues with $|\lambda| > 1$. The last line shows the results obtained with Hermite interpolation in the points indicated in Figure 5.2.

σ	$\#\lambda : E(\lambda, x) \leq 10^{-4}$
$-2/3$	6
$-1/3 + i3/5$	9(+1)
0	18
$1/3 + i3/5$	15(+3)
$2/3$	16(+8)
Hermite	21(+2)

Thus, we can conclude that the proposed rational Krylov method with multiple interpolation points is really suitable for finding eigenvalues in a specified region of interest. Furthermore, since the rational Krylov method uses an approximation of the nonlinear matrix function $A(\lambda)$ based on multiple interpolation points, this method is even more suitable for global eigenvalue search than the Taylor–Arnoldi method, introduced in [12], which uses only one interpolation point.

5.3. Gun problem: Local eigenvalue correction. The numerical experiments of the previous section have shown that the proposed algorithm performs well as a global method. In this section, we now illustrate that the method can also be used as a local method and we compare the proposed method to Newton's method.

Therefore, we return to the square root 'gun' problem (5.1). As suggested in [4], we consider the eigenvalue λ for which $\sqrt{\lambda}$ is nearest to 146.71. We first performed Newton's method, outlined in section 4.5, with $\lambda_0 = 146.71^2$ and x_0 a random vector. During the first iteration we took $\lambda_1 = \lambda_0$, since otherwise the algorithm would converge to an eigenvalue outside the region of interest. This could be explained by the fact that the random starting vector x_0 was not yet a good approximation of the eigenvector corresponding to the eigenvalue nearest to 146.71^2. The resulting convergence history toward the eigenvalue nearest to λ_0 is shown in Table 5.3(a).

TABLE 5.3

Convergence history of the approximate eigenvalue of the 'gun' problem nearest to 146.71^2 *with* (a) *Newton's method and with* (b) *the rational Krylov method.*

(a) Newton's method

i	$\sqrt{\lambda}$	$E(\lambda, x)$
0	146.71	4.8544e-02
1	146.71	5.7677e-04
2	$149.10 + 0.012i$	3.5515e-05
3	$149.48 + 0.002i$	1.8332e-07
4	$149.48 + 0.002i$	6.0417e-14
5	$149.48 + 0.002i$	1.3171e-17

(b) Rational Krylov method

i	$\sqrt{\sigma}$	$\sqrt{\lambda}$	$E(\lambda, x)$
0	146.71		4.8544e-02
1	146.71	$153.13 + 0.007i$	2.7855e-05
2	$153.13 + 0.007i$	$149.48 + 0.002i$	1.5795e-07
3	$149.48 + 0.002i$	$149.48 + 0.002i$	2.6212e-12
4	$149.48 + 0.002i$	$149.48 + 0.002i$	5.0740e-16

Now, we compare the rational Krylov method to Newton's method. To this end, we started Algorithm 2 with $\sigma_0 = \sigma_1 = 146.71^2$. For the other shifts, we chose, in each iteration, the Ritz value of the previous iteration which resulted in the smallest relative residual norm. The corresponding convergence history of the rational Krylov method is shown in Table 5.3(b). From this table, we can conclude that the rational Krylov

method converges in fewer iterations than Newton's method. This is expected because the rational Krylov method is a subspace method which builds an expanding subspace.

Recall a second difference between the proposed method and Newton's method. The rational Krylov method can converge at the same time to more than one eigenvalue.

5.4. Clamped sandwich beam with viscoelastic core. We consider a symmetric 210-mm long constrained-layer damping [20] cantilever beam composed of two cold rolled DC04 steel layers and an ethylene-propylene-diene adhesive core [17] (see Figure 5.5).

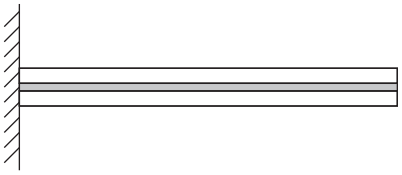


FIG. 5.5. *Clamped sandwich beam with viscoelastic core.*

The beam is discretized, using the finite element formulation proposed in [1], into 42 5-mm-long finite elements defined by 43 nodes, which yields 168 degrees of freedom after applying the clamped boundary condition. This results in the following NLEP:

$$F(\lambda)x = \left(K_e - \lambda^2 M + \frac{G_0 + G_\infty (i\lambda\tau)^\alpha}{1 + (i\lambda\tau)^\alpha} K_v\right) x = 0, \tag{5.3}$$

where K_e, M, and K_v are constant matrices, and the shear modulus of the sandwich beam is described by the four-parameter fractional derivative model, also known as the generalized Zener model. The static shear modulus $G_0 = 350.4\,\text{kPa}$, the asymptotic shear modulus $G_\infty = 3.062\,\text{MPa}$, the relaxation time $\tau = 8.230\,\text{ns}$, and the fractional parameter $\alpha = 0.675$.

We first scaled and transformed the original problem (5.3) by $\lambda = \exp(10\hat{\lambda})$. Then, we chose the interpolation points 0.2, 0.6, 0.8, 0.9, and 1, all with multiplicity 8, and solved the transformed NLEP by Algorithm 2 in less than 5 s. The 10 smallest eigenvalues and their corresponding relative residuals are given in Table 5.4. Figure 5.6 shows the transversal displacement of the clamped sandwich beam with viscoelastic core in function of x for the four smallest natural frequencies, $f_n = \text{Re}(\lambda)/(2\pi)$.

TABLE 5.4
The 10 *smallest eigenvalues of the clamped sandwich beam with viscoelastic core and the corresponding relative residuals.*

λ	$\|F(\lambda)x\|/\|\lambda\|$
1.3089e+02 + 3.9759e+00i	6.6637e-10
7.2337e+02 + 8.2940e+01i	7.1686e-13
1.9207e+03 + 2.9849e+02i	1.3876e-12
3.5800e+03 + 6.5778e+02i	4.9937e-13
5.6749e+03 + 1.1327e+03i	3.0197e-13
8.1832e+03 + 1.7015e+03i	3.5288e-13
1.1097e+04 + 2.3423e+03i	2.0809e-11
1.4415e+04 + 3.0390e+03i	9.2233e-10
1.8141e+04 + 3.7793e+03i	1.4533e-09
2.2280e+04 + 4.5536e+03i	6.6331e-09

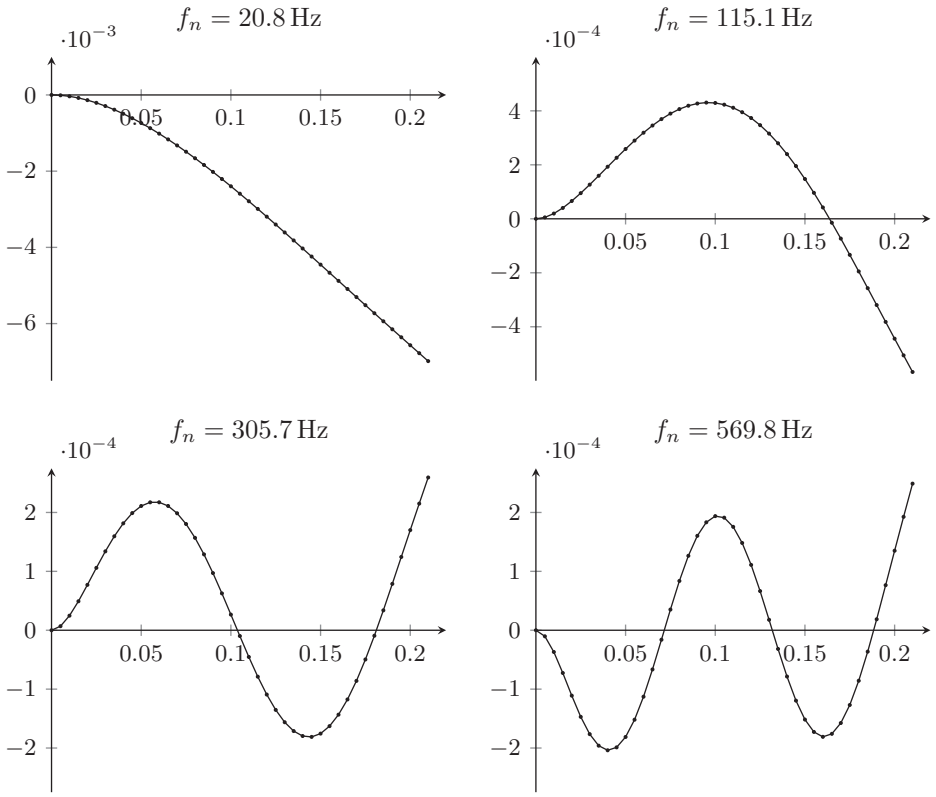


FIG. 5.6. *Transversal displacement* [m] *in function of* x [m] *of the clamped sandwich beam with viscoelastic core for the four smallest natural frequencies.*

6. Conclusions. In this paper, we have presented a new rational Krylov method for solving the NLEP. This method is fully dynamic and does not rely on a discrete-first approach. Instead, $A(\lambda)$ is approximated by a low-degree Hermite interpolating polynomial whose interpolation points are not fixed in advance.

The linearization of $A(\lambda)$ resulted in a GEP which is expressed in a companion-type matrix reformulation. Consequently by exploiting the specific structure, this GEP is efficiently solved by a rational Krylov method, where we choose the shifts or poles equal to the interpolation points. As a consequence, the interpolation points need not be fixed in advance and we can choose in each iteration a new interpolation point based on the results of the previous ones.

The algorithm can be efficiently implemented, such that it involves only matrix operations of the size of the original NLEP. Although applicable to NLEPs, the proposed method inherits properties of the rational Krylov method for linear eigenvalue problems, such as simultaneous convergence to several eigenvalues.

The numerical experiments have shown that the proposed method can be used as well as a global and a local method. The former can compute eigenvalues and corresponding eigenvectors in a large range of interest, while the latter computes local corrections of approximate eigenpairs as in Newton's method. We have illustrated that the proposed method converges faster than Newton's method.

In the numerical examples, we used scaling and/or zooming in on the region of interest. We experienced that this was vital for good convergence of the algorithm. In

addition, special care is needed in the close vicinity of branch points of the f_i's, since f_i cannot be well approximated by a low-degree polynomial in the neighborhood of a branch point.

Acknowledgments. The authors are grateful to Natalia Navarrete and Wim Desmet for providing the matrices used in section 5.4.

REFERENCES

[1] K. Amichi and N. Atalla, *A new 3D finite element for sandwich beams with a viscoelastic core*, J. Vib. Acoustics, 131 (2009), 21010.
[2] A. Amiraslani, R. M. Corless, and P. Lancaster, *Linearization of matrix polynomials expressed in polynomial bases*, IMA J. Numer. Anal., 29 (2008), pp. 141–157.
[3] J. Baglama, D. Calvetti, and L. Reichel, *Fast Leja points*, Electron. Trans. Numer. Anal., 7 (1998), pp. 124–140.
[4] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, *NLEVP: A Collection of Nonlinear Eigenvalue Problems*, Technical report 2011.116, Manchester Institute for Mathematical Sciences, University of Manchester, Manchester, UK, 2011.
[5] T. Betcke and H. Voss, *A Jacobi–Davidson-type projection method for nonlinear eigenvalue problems*, Future Generation Computer Systems, 20 (2004), pp. 363–372.
[6] W.-J. Beyn, C. Effenberger, and D. Kressner, *Continuation of eigenvalues and invariant pairs for parameterized nonlinear eigenvalue problems*, Numer. Math., 119 (2011), pp. 489–516.
[7] W.-J. Beyn, *An integral method for solving nonlinear eigenvalue problems*, Linear Algebra Appl., 436 (2012), pp. 3839–3863.
[8] A. Edrei, *Sur les déterminants récurrents et les singularités d'une fonction donnée par son développement de Taylor*, Compos. Math., 7 (1940), pp. 20–88.
[9] C. Effenberger and D. Kressner, *Chebyshev interpolation for nonlinear eigenvalue problems*, BIT, 52 (2012), pp. 933–951.
[10] C. Effenberger, *Robust Successive Computation of Eigenpairs for Nonlinear Eigenvalue Problems*, Technical Report 27, Ecole Polytechnique Federale de Lausanne, Lausanne, 2012.
[11] J. Huitfeldt and A. Ruhe, *A new algorithm for numerical path following applied to an example from hydrodynamical flow*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 1181–1192.
[12] E. Jarlebring, W. Michiels, and K. Meerbergen, *A linear eigenvalue algorithm for the nonlinear eigenvalue problem*, Numer. Math., 122 (2012), pp. 169–195.
[13] D. Kressner, *A block Newton method for nonlinear eigenvalue problems*, Numer. Math., 114 (2009), pp. 355–372.
[14] F. Leja, *Sur certaines suites liees aux ensembles plans et leur application a la representation conforme*, Ann. Polon. Math., 4 (1957), pp. 8–13.
[15] 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.
[16] 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.
[17] M. Martinez Aguirre, *Experimental and Numerical Dynamic Analysis of Press-Formed Viscoelastic Sandwich Structures*, Ph.D. thesis, University of Mondragón, Mondragón, Spain, 2011.
[18] V. Mehrmann and H. Voss, *Nonlinear eigenvalue problems: A challenge for modern eigenvalue methods*, GAMM Mitt. Ges. Angew. Math. Mech., 27 (2004), pp. 121–152.
[19] A. Neumaier, *Residual inverse iteration for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal., 22 (1985), pp. 914–923.
[20] M. D. Rao, *Recent applications of viscoelastic damping for noise control in automobiles and commercial airplanes*, J. Sound Vib., 262 (2003), pp. 457–474.
[21] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computation*, Linear Algebra Appl., 58 (1984), pp. 391–405.
[22] A. Ruhe, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sci. Comput., 19 (1998), pp. 1535–1551.
[23] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.
[24] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM Rev., 42 (2000), pp. 267–293.
[25] Y. Su and Z. Bai, *Solving rational eigenvalue problems via linearization*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 201–216.