Received: 19 May 2017 | Revised: 5 June 2018 | Accepted: 29 August 2018

DOI: 10.1002/nla.2214

RESEARCH ARTICLE

WILEY

A compact rational Krylov method for large-scale rational eigenvalue problems

Froilán M. Dopico | Javier González-Pizarro

Departamento de Matemáticas, Universidad Carlos III de Madrid, Leganés, Spain

Correspondence
Froilán M. Dopico, Departamento de Matemáticas, Universidad Carlos III de Madrid, Avenida Universidad 30, 28911 Leganés, Spain.
Email: dopico@math.uc3m.es

Present Address
Froilán M. Dopico, Avenida Universidad 30, 28911 Leganés, Spain

Funding information
Comisión Nacional de Investigación Científica y Tecnológica (CONICYT) de Chile, Grant/Award Number: BCH 72160331; Ministerio de Ciencia, Innovación y Universidades of Spain; Fondo Europeo de Desarrollo Regional (FEDER) of EU, Grant/Award Number: MTM2012-32542, MTM2015-65798-P, MTM2015-68805-REDT, and MTM2017-90682-REDT

Summary

In this work, we propose a new method, termed as R-CORK, for the numerical solution of large-scale rational eigenvalue problems, which is based on a linearization and on a compact decomposition of the rational Krylov subspaces corresponding to this linearization. R-CORK is an extension of the compact rational Krylov method (CORK) introduced very recently in the literature to solve a family of nonlinear eigenvalue problems that can be expressed and linearized in certain particular ways and which include arbitrary polynomial eigenvalue problems, but not arbitrary rational eigenvalue problems. The R-CORK method exploits the structure of the linearized problem by representing the Krylov vectors in a compact form in order to reduce the cost of storage, resulting in a method with two levels of orthogonalization. The first level of orthogonalization works with vectors of the same size as the original problem, and the second level works with vectors of size much smaller than the original problem. Since vectors of the size of the linearization are never stored or orthogonalized, R-CORK is more efficient from the point of view of memory and orthogonalization than the classical rational Krylov method applied directly to the linearization. Taking into account that the R-CORK method is based on a classical rational Krylov method, to implement implicit restarting is also possible, and we show how to do it in a memory-efficient way. Finally, some numerical examples are included in order to show that the R-CORK method performs satisfactorily in practice.

KEYWORDS

large-scale, linearization, rational eigenvalue problem, rational Krylov method

1 | INTRODUCTION

In this work, we consider the rational eigenvalue problem (REP)

$$R(\lambda)x = 0, \tag{1}$$

where $R(\lambda) \in \mathbb{C}(\lambda)^{n\times n}$ is a nonsingular rational matrix, that is, the entries of $R(\lambda)$ are scalar rational functions in the variable λ with complex coefficients and $\det(R(\lambda)) \not\equiv 0$ is not identically zero, and $x \in \mathbb{C}^n$ is a nonzero vector. More

precisely, we consider that $R(\lambda)$ is given as

$$R(\lambda) = P(\lambda) - \sum_{i=1}^{k} \frac{f_i(\lambda)}{g_i(\lambda)} E_i, \tag{2}$$

where $P(\lambda) \in \mathbb{C}[\lambda]^{n\times n}$ is a matrix polynomial of degree d in the variable λ; $f_i(\lambda)$ and $g_i(\lambda)$ are coprime scalar polynomials of degrees m_i and n_i, respectively; and $m_i < n_i$ and $E_i \in \mathbb{C}^{n\times n}$ are constant matrices for $i = 1, \ldots, k$. We emphasize that it is well known that every rational matrix can be written in the form given in (2)[1,2] (see also section 2 in the work of Amparan et al.[3]) and that such form appears naturally in many applications.[4]

The REP has attracted considerable interest in recent years since it arises in different applications in some fields such as vibration of fluid–solid structures,[5] optimization of acoustic emissions of high-speed trains,[6] free vibration of plates with elastically attached masses,[7] free vibrations of a structure with a viscoelastic constitutive relation describing the behavior of a material,[8,9] and electronic structure calculations of quantum dots.[10,11]

A first idea to solve REPs is a brute-force approach, since one can multiply by $\prod_{i=1}^{k} g_i(\lambda)$ to turn the rational matrix (2) into a matrix polynomial of degree $d + n_1 + \cdots + n_k$. The common approach to solve a polynomial eigenvalue problem (PEP) is via linearization (see, for instance, the works of Mehrmann and Voss,[8] De Terán et al.,[12] and Mackey et al.[13]), that is, by transforming the PEP into a generalized eigenvalue problem (GEP) and then applying a well-established algorithm to this GEP, as, for instance, the QZ algorithm in the case of dense medium-sized problems[14] or some Krylov subspace method for large-scale problems. However, this brute-force approach is only useful when $n_1 + n_2 + \cdots + n_k$ is small compared with d. Hence, if k or some n_i are big, then the degree of the matrix polynomial associated to the problem is also big, and this makes the size of the linearization too large, which is impractical for medium- to large-scale problems. This drawback has motivated the idea of linearizing directly the REP.[4] The linearization for $R(\lambda)$ in (2) constructed in the work of Su and Bai[4] has a size much smaller than the size of the linearization obtained by the brute-force approach. Nonetheless, the increase in the size of the problem is still considerable; hence, for large-scale REPs, a direct application of this approach, that is, without taking into account the structure of the linearization, is also impractical. This idea of taking advantage of the structure of the linearization for solving large-scale REPs is closely connected to the intense research effort developed in the last years by different authors for solving large-scale PEPs via linearizations, and that is briefly discussed in the next paragraph.

Several methods have been developed to solve large-scale PEPs numerically by applying Krylov methods to the associated GEPs obtained through linearizations. In this approach, the key issues to be solved for using Krylov methods for large-scale PEPs are the increase in the memory cost and the increase in the orthogonalization cost at each step, as a consequence of the increase in the size of the linearization with respect to the size of the original problem. In order to reduce these costs, different representations of the Krylov vectors of the linearizations have been developed. First, the second-order Arnoldi method[15] and the quadratic Arnoldi method[16] were developed to solve quadratic eigenvalue problems (QEPs), introducing a new representation of the Krylov vectors. However, both methods are potentially unstable as a consequence of performing implicitly the orthogonalization. To cure this instability, the two-level orthogonal Arnoldi process (TOAR)[17,18] for the QEP proposed a different compact representation for the Krylov vectors of the linearization and, combining this representation with the linearization and the Arnoldi recurrence relation, resulted in a memory-saving and numerically stable method. Extending the ideas of a compact representation of the Krylov vectors and of the two levels of orthogonalization from polynomials of degree 2 (TOAR) to polynomials of any degree, Kressner and Roman[19] developed a memory-efficient and stable Arnoldi process for the linearizations of matrix polynomials expressed in the Chebyshev basis. In 2015, the compact rational Krylov (CORK) method for nonlinear eigenvalue problems (NLEPs) was introduced in the work of Van Beeumen et al.[20] CORK considers particular NLEPs that can be expressed and linearized in certain ways, which are solved by applying a CORK method to such linearizations. A key feature of the CORK method is that it works for many kinds of linearizations involving a Kronecker structure, as the Frobenius companion form or linearizations of matrix polynomials in different bases (as Newton or Chebyshev, among others[21]). CORK reduces both the costs of memory and orthogonalization by using a generalization of the compact Arnoldi representation of the Krylov vectors of the linearizations used in TOAR[17,18] and gets stability through two levels of orthogonalization as in TOAR.

In this paper, we develop a rational Krylov (RK) method that works on the linearization of REPs introduced in the work of Su and Bai[4] to solve large-scale and sparse $n \times n$ REPs. To this aim, we introduce a CORK method for REPs (R-CORK). In the spirit of TOAR and CORK, we will work with two levels of orthogonalization, and, as in CORK, we adapt the classical RK method[20,22,23] on the linearization to a compact representation of the Krylov vectors and to the two levels of orthogonalization. We can perform the shift-and-invert step by solving linear systems of size n. To this purpose, the linearization introduced in the work of Su and Bai[4] is preprocessed in a convenient way, and then, a UL decomposition is

used. This decomposition is similar to the one employed in the work of Van Beeumen et al.[20] directly on the linearizations of the NLEPs considered there. Once this step is performed, we start with the two levels of orthogonalization. The first level involves an orthogonalization process with vectors of size n, and in the second level of orthogonalization, we work with vectors of size much smaller than n; hence, this level is cheap compared with the first level. As a result, we develop a stable method that allows us to reduce the orthogonalization cost and the memory cost by exploiting the structure of the matrix pencil that linearizes the REP and by using the RK recurrence relation.

The rest of this paper is organized as follows. Section 2 introduces some preliminary concepts: a summarized background on PEPs and REPs, the classical RK method for the GEP, and the CORK method particularized to PEPs. Section 3 proposes the CORK decomposition that we use to develop the R-CORK method and presents the detailed algorithm with the two levels of orthogonalization. Section 4 discusses the implementation of implicit restarting for the R-CORK method. Section 5 presents numerical examples that show that the R-CORK method works satisfactorily in practice, and finally, the main conclusions and some lines of future research are discussed in Section 6.

Notation. We denote vectors by lowercase characters, u, and matrices by capital characters, A. Block vectors and block matrices are denoted by boldface fonts, $\mathbf{u}$ and $\mathbf{A}$, respectively, and the ith block of $\mathbf{u}$ is represented by $u^{(i)}$. The conjugate transpose of A is denoted as A^*. The $i \times j$ matrix with the main diagonal entries equal to 1 and the rest of the entries equal to zero is represented by $I_{i\times j}$. In the particular case of $i = j$, this matrix is the identity matrix and is denoted by I_i. The vector e_j represents the canonical vector associated to the jth column of the identity matrix, and $0_{i\times j}$ represents the zero matrix of size $i \times j$, which, in the particular case $i = j$, is denoted simply by 0_j. The matrix U_j represents a matrix with j columns, and u_i represents the ith column of U_j. The RK subspace of order m associated with the matrices A and $B \in \mathbb{C}^{n\times n}$, the initial vector $u_1 \in \mathbb{C}^n$, and the shifts $\theta_1, \theta_2, \ldots, \theta_{m-1} \in \mathbb{C}$ is denoted by

$$\mathcal{K}_m(A, B, u_1, \theta_{1,\ldots,m-1}) = \text{span}\left\{u_1, (A-\theta_1 B)^{-1}Bu_1, (A-\theta_2 B)^{-1}Bu_2, \ldots, (A-\theta_{m-1}B)^{-1}Bu_{m-1}\right\}, \tag{3}$$

where $u_{i+1} = (A - \theta_i B)^{-1}Bu_i$, $i = 1, \ldots, m-1$. We omit subscripts when the dimensions of the matrices are clear from the context. The norm $\|\cdot\|_2$ represents the 2-norm, and $\|\cdot\|_F$ represents the Frobenius norm (see chapter 5 in the work of Horn and Johnson[24]). The Kronecker product of two matrices is denoted by $A \otimes B$. The set of $n \times n$ rational matrices is denoted by $\mathbb{C}(\lambda)^{n\times n}$, and the set of $n \times n$ polynomial matrices (or, equivalently, matrix polynomials) is denoted by $\mathbb{C}[\lambda]^{n\times n}$.

2 | PRELIMINARIES

2.1 | Basics on PEPs and linearizations

The classical approach to solve a regular PEP is as follows:

$$P(\lambda)x = 0, \tag{4}$$

where $P(\lambda) = \sum_{i=0}^{d} \lambda^i P_i$ with $P_i \in \mathbb{C}^{n\times n}$ and $\det(P(\lambda)) \not\equiv 0$ is via linearization. In this process, the matrix polynomials are mapped into matrix pencils with the same eigenvalues and multiplicities.[13,25] More precisely, a pencil $L(\lambda) = \mathbf{A} - \lambda\mathbf{B}$ is called a linearization of $P(\lambda)$ if there exist unimodular matrix polynomials* $E_1(\lambda)$, $E_2(\lambda)$ such that

$$\begin{bmatrix} P(\lambda) & 0 \\ 0 & I_{(d-1)n} \end{bmatrix} = E_1(\lambda)(\mathbf{A} - \lambda\mathbf{B})E_2(\lambda).$$

Some linearizations of matrix polynomials of degree d and size $n \times n$, very useful in practice, are of the form as the pencils in Definition 1.

Definition 1. (See definition 2.2 in the work of Van Beeumen et al.[20])
Let $P(\lambda) \in \mathbb{C}[\lambda]^{n\times n}$ be a regular matrix polynomial, that is, $\det(P(\lambda))$ does not vanish identically, of degree $d \geq 2$ and size $n \times n$. A $dn \times dn$ matrix pencil $L(\lambda)$ of the form

$$L(\lambda) = \mathbf{A} - \lambda\mathbf{B}, \tag{5}$$

*Unimodular matrix polynomials are matrix polynomials whose determinant is a nonzero constant, that is, it does not depend on λ. Most of the linearizations considered in this work are, in fact, strong linearizations[12,13]; hence, they preserve also the eigenvalues at infinity of $P(\lambda)$ and their multiplicities, if they are present. Nevertheless, in this work, we do not intend to compute infinite eigenvalues since their existence is not generic, and so, we do not need to use the concept of strong linearization.

where

$$\mathbf{A} = \left[\begin{array}{c|ccc} A_0 & A_1 & \cdots & A_{d-1} \\ \hline M \otimes I_n \end{array}\right], \quad \mathbf{B} = \left[\begin{array}{c|ccc} B_0 & B_1 & \cdots & B_{d-1} \\ \hline N \otimes I_n \end{array}\right], \tag{6}$$

where $A_i, B_i \in \mathbb{C}^{n\times n}$, $i = 0, 1, \ldots, d-1$, and $M, N \in \mathbb{C}^{(d-1)\times d}$, is called a structured linearization pencil of $P(\lambda)$ if the following conditions hold:

1. $L(\lambda)$ is a linearization of $P(\lambda)$;
2. $M - \lambda N$ has rank $d - 1$ for all $\lambda \in \mathbb{C}$;
3. $(\mathbf{A} - \lambda\mathbf{B})(f(\lambda) \otimes I_n) = e_1 \otimes P(\lambda)$ for some polynomial function $f : \mathbb{C} \to \mathbb{C}[\lambda]^d, f(\lambda) \neq 0$ for all $\lambda \in \mathbb{C}$, where $e_1 \in \mathbb{C}^d$ is the first vector of the canonical basis of $\mathbb{C}^d$.

The matrices A_i and B_i that appear in the first block rows in (6) are related to the matrix polynomial $P(\lambda)$, and matrices M and N correspond to the linear relations between the basis functions $f_i(\lambda)$, where $f(\lambda) := [f_1(\lambda), \ldots, f_d(\lambda)]^T$, used in the representation of the matrix polynomial. The interested reader can find some examples in the work of Van Beeumen et al.[20] The identity $(\mathbf{A} - \lambda\mathbf{B})(f(\lambda) \otimes I_n) = e_1 \otimes P(\lambda)$ generalizes the identity used in the work of Mackey et al.[13] to define certain vector spaces of linearizations of matrix polynomials.

An important property of structured linearization pencils is that their eigenvectors are closely related to the eigenvectors of the matrix polynomial, as we can see in Theorem 1.

Theorem 1. (See corollary 2.4 in the work of Van Beeumen et al.[20])
Let $L(\lambda)$ be a structured linearization pencil of $P(\lambda)$ as in Definition 1, and let $(\lambda_\star, \mathbf{x})$ be an eigenpair of $L(\lambda)$. Then, the eigenvector $\mathbf{x}$ has the following structure:

$$\mathbf{x} = f(\lambda_\star) \otimes x,$$

where $x \in \mathbb{C}^n$ is an eigenvector of $P(\lambda)$ corresponding to $\lambda_\star$.

The block ULP decomposition in Theorem 2 for the structured linearization pencils of matrix polynomials is important for the CORK method introduced in the work of Van Beeumen et al.[20] because it allows to perform the shift-and-invert step in CORK efficiently. We emphasize that the structure of the linearization presented in Section 2.2 for solving the REP allows us to develop a similar UL decomposition to perform efficiently the shift-and-invert step in the R-CORK method introduced in Section 3. This UL decomposition will be presented in Lemma 1 and, in contrast to that in Theorem 2, does not include any permutation and is applied after preprocessing the linearization.

Theorem 2. (See theorem 2.3 in the work of Van Beeumen et al.[20])
Let $\mathbf{A}$ and $\mathbf{B}$ be defined by (6). 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

$$M =: [m_0 \quad M_1]\mathcal{P}, \quad N =: [n_0 \quad N_1]\mathcal{P}.$$

Moreover, the matrix $L(\mu)$, that is, the pencil $L(\lambda)$ in (5) evaluated in μ, can be factorized as follows:

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

where

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

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

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

2.2 | A linearization for REPs

In this subsection, we present some results and notations related to the REP. Interested readers can find more information in the summaries presented in sections 1 and 2 in the work of Alam and Behera[26] and section 2 in the work of Amparan et al.[3] as well as in the classical works of Kailath[1] and Rosenbrock.[2]

In this work, we assume that the rational matrix $R(\lambda)$ in (2) is regular, that is, $\det(R(\lambda)) \not\equiv 0$. With a slight lack of rigor, we can say that if the matrices E_i in (2) are linearly independent, then the roots of the denominators $g_i(\lambda)$ are the poles of $R(\lambda)$ and that $R(\lambda)$ is not defined in these poles. A scalar $\lambda \in \mathbb{C}$, which is not a pole, is called an eigenvalue of $R(\lambda)$ if $\det(R(\lambda)) = 0$, and a nonzero vector $x \in \mathbb{C}^n$ is called an eigenvector of $R(\lambda)$ associated to the eigenvalue λ if condition (1) holds. The pair (λ, x) constitutes an eigenpair of $R(\lambda)$, and our goal is to compute a subset of such eigenpairs.

We express the matrix polynomial $P(\lambda)$ of degree d in (2) as follows:

$$P(\lambda) = \lambda^d P_d + \lambda^{d-1} P_{d-1} + \cdots + \lambda P_1 + P_0, \tag{7}$$

where $P_i \in \mathbb{C}^{n\times n}$ for $i = 0, \ldots, d$. From now on, we assume the generic condition that the leading coefficient matrix P_d is nonsingular in (7). As explained in the Introduction, we assume that $f_i(\lambda)$ and $g_i(\lambda)$ in (2) are coprime, that is, they do not have common factors, and that the rational functions $\frac{f_i(\lambda)}{g_i(\lambda)}$ are strictly proper, that is, the degree, m_i, of $f_i(\lambda)$ is smaller than the degree, n_i, of $g_i(\lambda)$. Under these assumptions, Su and Bai[4] proposed a linearization to solve the REP. With this aim, they first showed that one can find matrices E and F of size $n \times s$ and matrices C and D of size $s \times s$, with $s = r_1 n_1 + r_2 n_2 + \cdots + r_k n_k$ with r_i =rank(E_i) in (2), such that

$$R(\lambda) = P(\lambda) - E(C - \lambda D)^{-1} F^T. \tag{8}$$

In fact, it is a classical result (much older than the work of Su and Bai[4]) that any rational matrix can be written as in (8) by expressing $R(\lambda)$ as the sum of its unique polynomial and strictly proper parts and then constructing a state-space realization of the strictly proper part[2] (see also section 2 in the work of Amparan et al.[3]). However, we emphasize that, as far as we know, the work of Su and Bai[4] is the first one available in the literature that uses (8) with the purpose of computing the eigenvalues of an REP and also the first to point out that representation (8) is immediately available from the data in many practical REPs without any computational cost.

Once representation (8) for the REP is available, the REP $R(\lambda)x = 0$ can be linearized according to the work of Su and Bai[4] as follows:[†]

$$(\mathcal{A} - \lambda\mathcal{B})\mathbf{z} = 0, \tag{9}$$

where

$$\mathcal{A} = \left[\begin{array}{cccc|c} P_0 & P_1 & \cdots & P_{d-1} & E \\ 0 & -I_n & & & \\ & \ddots & \ddots & & \\ & & 0 & -I_n & \\ \hline F^T & & & & C \end{array}\right], \quad \mathcal{B} = -\left[\begin{array}{cccc|c} & & & P_d & \\ I_n & 0 & & & \\ & \ddots & \ddots & & \\ & & I_n & 0 & \\ \hline & & & & -D \end{array}\right] \tag{10}$$

and

$$\mathbf{z} = \left[\begin{array}{c} x \\ \lambda x \\ \vdots \\ \lambda^{d-1} x \\ \hline y \end{array}\right], \qquad \text{with } y = -(C - \lambda D)^{-1} F^T x. \tag{11}$$

Denoting by $\mathbf{A}$ and $\mathbf{B}$ the upper left $nd \times nd$ submatrices of $\mathcal{A}$ and $\mathcal{B}$, we can write $\mathcal{A} - \lambda\mathcal{B}$ as follows:

$$\mathcal{A} - \lambda\mathcal{B} = \left[\begin{array}{c|c} \mathbf{A} - \lambda\mathbf{B} & e_1 \otimes E \\ \hline e_1^T \otimes F^T & C - \lambda D \end{array}\right], \tag{12}$$

where e_1 is the first column of I_d. Observe that $\mathbf{A} - \lambda\mathbf{B}$ is a permutation of the famous first Frobenius companion linearization of the matrix polynomial $P(\lambda)$ in (7).[25]

[†]We remark that Su and Bai[4] considered the permutations of the matrices $\mathcal{A}$ and $\mathcal{B}$ in (10) for the linearized problem (9). More precisely, they ordered the matrices P_i in reverse order and interchanged the identity blocks in $\mathcal{A}$ and $\mathcal{B}$. Since we are following, in this paper, the spirit of CORK, then $\mathcal{A}$ and $\mathcal{B}$ are written in (10) in the same way as in table 1 in the work of Van Beeumen et al.[20]

As mentioned above, it is important to remark that in many applications of REPs,[4,8,11] the first step in the process above, that is, to construct representation (8), does not involve any computational effort, since the matrices $P_0, P_1, \ldots, P_d, E, C, D$, and F can be obtained directly from the data. As a consequence, the linearization $\mathcal{A} - \lambda\mathcal{B}$ above can be constructed also without any computational effort, and all the computational effort is attached to the solution of the GEP (9). Another important remark that has a deep computational impact is that in many applications of REPs,[4,8,11] the size $s \times s$ of matrices C and D is much smaller than the size $n \times n$ of the original REP, that is, $s \ll n$, or, in plain words, the rank of the strictly proper part of $R(\lambda)$ is much smaller than the size of $R(\lambda)$. Therefore, if $s \ll n$, then the size $(nd + s) \times (nd + s)$ of the linearization $\mathcal{A} - \lambda\mathcal{B}$ is approximately equal to the size $(nd) \times (nd)$ of the linearization $\mathbf{A} - \lambda\mathbf{B}$ of the matrix polynomial $P(\lambda)$, and the costs of solving the GEPs $\mathcal{A} - \lambda\mathcal{B}$ and $\mathbf{A} - \lambda\mathbf{B}$ are expected to be similar. Thus, it is not surprising that the R-CORK algorithm developed in Section 3 for large-scale REPs is particularly efficient in terms of storage and orthogonalization costs when $s \ll n$. However, we emphasize in this context that R-CORK also improves significantly these costs when $s \approx n$ (and $d \geq 2$) with respect to a direct application of large-scale eigensolvers to $\mathcal{A} - \lambda\mathcal{B}$. We will often make comments about the advantages of considering $s \ll n$ throughout this paper.

A formal definition of linearization of a rational matrix can be found in the work of Alam and Behera[26] and another one that includes the concept of strong linearization in the work of Amparan et al.[3] In fact, it is proved in the aforementioned work[3] that $\mathcal{A} - \lambda\mathcal{B}$ in (9) is a strong linearization of $R(\lambda)$ in (8) whenever $-E(C - \lambda D)^{-1}F^T$ is a *minimal* state-space realization[2] of the strictly proper part of $R(\lambda)$ (see section 8 in the work of Amparan et al.[3]). We emphasize that the requirement that $-E(C - \lambda D)^{-1}F^T$ is a minimal state-space realization is very mild (see section 8 in the work of Amparan et al.[3]) and that it is fully necessary to guarantee that for every eigenvalue λ of $R(\lambda)$, the matrix $C - \lambda D$ is nonsingular (see example 3.2 in the work of Amparan et al.[3]). In the rest of this paper, we implicitly assume that $E(C - \lambda D)^{-1}F^T$ is a minimal realization, although the only result we will use explicitly is Theorem 3, which remains valid even when $E(C - \lambda D)^{-1}F^T$ is not minimal. The subtle point is that Theorem 3 assumes that λ is a number such that the matrix $(C - \lambda D)$ is invertible, but if $E(C - \lambda D)^{-1}F^T$ is not minimal, then there may be eigenvalues of $R(\lambda)$ that do not satisfy such assumption.

Theorem 3. (See theorem 3.1 in the work of Su and Bai[4])
Let $\lambda \in \mathbb{C}$ be such that $det(C - \lambda D) \neq 0$. Then, the following statements hold.

(a) If λ is an eigenvalue of the REP (8), then it is an eigenvalue of the GEP (9).
(b) Let λ be an eigenvalue of the GEP (9) and $z = [z_1^T, z_2^T, \ldots, z_d^T, y^T]^T$ be a corresponding eigenvector, where z_i are vectors of length n for $i = 1, 2, \ldots, d$ and y is a vector of length s. Then, $z_1 \neq 0$ and $R(\lambda)z_1 = 0$, namely, λ is an eigenvalue of the REP (8) and z_1 is a corresponding eigenvector. Moreover, the algebraic and geometric multiplicities of λ for the REP (8) and GEP (9) are the same.

Part (b) of Theorem 3 is the key result that allows us to get the eigenvalues and eigenvectors of $R(\lambda)$ from those of the GEP $\mathcal{A} - \lambda\mathcal{B}$ in (9). In fact, observe that part (b) of Theorem 3 can be improved if $\lambda \neq 0$, since in this case, according to (11), every z_i is an eigenvector of $R(\lambda)$. Therefore, we have d degrees of freedom for the recovery of the eigenvector of $R(\lambda)$. The most sensible option from the point of view of rounding errors is to choose the z_i with the largest 2-norm, that is, z_d when $|\lambda| > 1$ and z_1 when $|\lambda| \leq 1$.

The final comment of this section is that in contrast to the CORK method developed in the work of Van Beeumen et al.[20] for PEPs, which is valid for many linearizations, the rational CORK method, R-CORK, introduced in this paper uses only the linearization $\mathcal{A} - \lambda\mathcal{B}$ in (9). The reason for this restriction is that the theory of linearizations of REPs is far less developed than the theory of linearizations of PEPs. Thus, although many linearizations of REPs have been introduced very recently in the works of Amparan et al.[3] and Alam and Behera,[26] their properties are not yet fully understood.

2.3 | The classical RK method for GEPs

We revise in this subsection the RK method for GEPs since the R-CORK algorithm presented in this paper is based on this method. The RK method[22,23] is a generalization for computing the eigenvalues of matrices and of matrix pencils of the shift-and-invert Arnoldi method. The main differences between these methods are basically two: In the RK methods, we can change the shift θ_j at each iteration instead of fixing the shift as in the shift-and-invert Arnoldi method. Also, the information of the approximate eigenvalues is contained in two upper Hessenberg matrices $\underline{H}_j$ and $\underline{K}_j$ instead of in only one matrix. In Algorithm 1, we present a basic pseudocode of the RK method that summarizes its main steps and guides the developments in the rest of this subsection, which are a very brief sketch of the RK method. The reader can find more details in the works of Van Beeumen et al.[20] and Ruhe.[22,23]

Algorithm 1. Rational Krylov method

Input: $\mathcal{A}$ and $\mathcal{B}$ square matrices and an initial vector $\mathbf{u}_1$ with $\|\mathbf{u}_1\|_2 = 1$.

Output: The matrix $\mathbf{U}_{m+1}$ whose columns are an orthonormal basis of $\mathcal{K}_{m+1}(\mathcal{A}, \mathcal{B}, \mathbf{u}_1, \theta_{1,\dots,m})$, and the Ritz pairs $(\lambda, \mathbf{x})$ of $\mathcal{A} - \lambda\mathcal{B}$, corresponding to the rational Krylov subspace $\mathcal{K}_m(\mathcal{A}, \mathcal{B}, \mathbf{u}_1, \theta_{1,\dots,m-1})$.

Initialize $\mathbf{U}_1 = [\mathbf{u}_1]$.

for $j = 1, 2, \dots, m$ **do**

1. Choose the shift θ_j.
2. $\hat{\mathbf{u}} = (\mathcal{A} - \theta_j\mathcal{B})^{-1}\mathcal{B}\mathbf{u}_j$.
3. $h_j = \mathbf{U}_j^*\hat{\mathbf{u}}$.
4. $\tilde{\mathbf{u}} = \hat{\mathbf{u}} - \mathbf{U}_j h_j$.
5. Compute the new vector $\mathbf{u}_{j+1} = \tilde{\mathbf{u}}/h_{j+1,j}$ with $h_{j+1,j} = \|\tilde{\mathbf{u}}\|_2$.
6. Update $\mathbf{U}_{j+1} = [\mathbf{U}_j \quad \mathbf{u}_{j+1}]$.
7. Compute the eigenpairs (λ_i, t_i) of (15) and test for convergence.

end for

8. Compute the eigenvectors $\mathbf{x}_i = \mathbf{U}_{j+1}\underline{H}_j t_i$, $i = 1, \dots, j$.

This method produces an orthonormal basis for the subspace $\mathcal{K}_{m+1}(\mathcal{A}, \mathcal{B}, \mathbf{u}_1, \theta_{1,\dots,m})$. By using the equalities for $\hat{\mathbf{u}}$ and $\tilde{\mathbf{u}}$ from Steps 2 and 4 in Algorithm 1, at the ith iteration, we obtain

$$(\mathcal{A} - \theta_i\mathcal{B})^{-1}\mathcal{B}\mathbf{u}_i = \mathbf{U}_{i+1}\underline{h}_i,$$

with $\underline{h}_i = [h_i^* \quad h_{i+1,i}]^*$. After j steps of the RK method, we obtain the following classic RK recurrence relation[23]:

$$\mathcal{A}\mathbf{U}_{j+1}\underline{H}_j = \mathcal{B}\mathbf{U}_{j+1}\underline{K}_j, \tag{13}$$

where $\underline{H}_j, \underline{K}_j \in \mathbb{C}^{(j+1)\times j}$ are upper Hessenberg matrices and

$$\underline{K}_j = \underline{H}_j \operatorname{diag}(\theta_1, \theta_2, \dots, \theta_j) + I_{(j+1)\times j}. \tag{14}$$

For simplicity, we assume that breakdowns do not occur in the RK method, that is, $h_{i+1,i} \neq 0$ for all $i = 1, \dots, m$, and in this case, the upper Hessenberg matrix $\underline{H}_j$ is unreduced. We can approximate in each iteration of Algorithm 1 the corresponding j eigenvalues and eigenvectors of the pencil $\mathcal{A} - \lambda\mathcal{B}$ by solving the small GEP, as follows:

$$K_j t_i = \lambda_i H_j t_i, \quad t_i \neq 0, \tag{15}$$

where H_j and K_j are the $j \times j$ upper Hessenberg matrices obtained by removing the last rows of $\underline{H}_j$ and $\underline{K}_j$, respectively. Then, we call $(\lambda_i, \mathbf{x}_i = \mathbf{U}_{j+1}\underline{H}_j t_i)$ a Ritz pair of $(\mathcal{A}, \mathcal{B})$. We emphasize that the approximate eigenvectors $\mathbf{x}_i$ are not computed in each iteration, since this would be very expensive, and that the test for convergence in Step 7 of Algorithm 1 can be performed in an inexpensive way by using only the small vectors t_i, as it is done in most Krylov methods.

2.4 | The CORK method for PEPs

Van Beeumen et al.[20] proposed a method based on CORK decomposition, extending the two levels of orthogonalization idea of TOAR from the QEP[17,18] to arbitrary-degree PEPs and to other NLEPs, including many other linearizations apart from the Frobenius one used in the works of Su et al.[17] and Lu et al.[18] and using the RK method instead of the Arnoldi method. This method was baptized as CORK in the work of Van Beeumen et al.,[20] and for simplicity, we described it to be particularized to PEPs of degree d. The key idea in the work of Van Beeumen et al.[20] is to apply the RK method in Algorithm 1 to a structured linearization pencil of a matrix polynomial $P(\lambda)$ of degree d (recall Definition 1), taking into account that the special structure of these pencils imposes a special structure on the bases of the corresponding RK subspaces. By using this structure, Van Beeumen et al.[20] reduced both the memory cost and the orthogonalization cost of

the classical RK method applied to an arbitrary pencil of the same size. Considering the matrices $\mathbf{A}$ and $\mathbf{B}$ in (6) and the RK recurrence relation (13) for $\mathbf{A}$ and $\mathbf{B}$, Van Beeumen et al.[20] partitioned conformably the matrix $\mathbf{U}_{j+1}$ as follows:

$$\mathbf{U}_{j+1} = [\mathbf{U}_j \quad \mathbf{u}_{j+1}] = \begin{bmatrix} U_j^{(1)} & u_{j+1}^{(1)} \\ U_j^{(2)} & u_{j+1}^{(2)} \\ \vdots & \vdots \\ U_j^{(d)} & u_{j+1}^{(d)} \end{bmatrix},$$

and then, they constructed a matrix $Q_j \in \mathbb{C}^{n\times r_j}$ with orthonormal columns such that

$$\operatorname{span}\{Q_j\} = \operatorname{span}\left\{U_j^{(1)}, U_j^{(2)}, \ldots, U_j^{(d)}\right\} \tag{16}$$

and $\operatorname{rank}(Q_j) = r_j$. By using matrix Q_j, the blocks $U_j^{(i)}$ for $i = 1, 2, \ldots, d$ can be represented as follows:

$$U_j^{(i)} = Q_j R_j^{(i)}, \quad i = 1, 2, \ldots, d,$$

for some matrices $R_j^{(i)} \in \mathbb{C}^{r_j\times j}$. Then, we have

$$\mathbf{U}_j = \begin{bmatrix} Q_j R_j^{(1)} \\ Q_j R_j^{(2)} \\ \vdots \\ Q_j R_j^{(d)} \end{bmatrix} = \begin{bmatrix} Q_j & & & \\ & Q_j & & \\ & & \ddots & \\ & & & Q_j \end{bmatrix} \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \end{bmatrix} = (I_d \otimes Q_j)\mathbf{R}_j, \tag{17}$$

where

$$\mathbf{R}_j := \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \end{bmatrix}.$$

By using this representation, the RK recurrence relation (13) can be written as follows (see equation (4.3) in the work of Van Beeumen et al.[20]):

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

Observe that $\mathbf{U}_j$ has ndj entries, whereas the representation in (17) involves $(n + jd)r_j$ parameters. Therefore, taking into account that in the solution of large-scale PEPs, the dimension j of the RK subspaces is much smaller than the dimension n of the problem and that the degree d of applied PEPs is a low number (for sure smaller than 30 [see the work of Kressner and Roman[19]] and often much smaller than 30 [see the work of Betcke et al.[27]]), we get that $jd \ll n$ and that the representation (17) of $\mathbf{U}_j$ stores approximately nr_j numbers. The fundamental reason why the representation of $\mathbf{U}_j$ in (17) is of interest and is, indeed, compact is that r_j is considerably much smaller than jd for the matrices $\mathbf{A}$ and $\mathbf{B}$ in (6). More precisely, the following result is proved in the work of Van Beeumen et al.[20]

Theorem 4. (See theorems 4.4 and 4.5 in the work of Van Beeumen et al.[20])
Let Q_j be defined as in (16). Then, we have

$$\operatorname{span}\{Q_{j+1}\} = \operatorname{span}\left\{Q_j, u_{j+1}^{(p)}\right\}, \tag{18}$$

where $u_{j+1}^{(p)}$ represents the block of vector $\mathbf{u}_{j+1}$ in a certain pth position determined in the work of Van Beeumen et al.[20] Also, we have

$$r_j < j + d. \tag{19}$$

Note that Theorem 4 shows that Q_j can be expanded to Q_{j+1} by orthogonalizing only one vector of size n at each iteration. Also, $\mathbf{R}_{j+1}$ can be expanded in an easy way; if $u_{j+1}^{(p)} \notin \operatorname{span}\{Q_j\}$, then the blocks $R_{j+1}^{(i)}$, $i = 1, \ldots, d$, can be written as

$$R_{j+1}^{(i)} = \left[\begin{array}{c|c} R_j^{(i)} & r_{j+1}^{(i)} \\ 0_{1\times j} & \end{array}\right], \quad i = 1, \ldots, d,$$

and if $u_{j+1}^{(p)} \in \operatorname{span}\{Q_j\}$, then $R_{j+1}^{(i)} = \begin{bmatrix} R_j^{(i)} & r_{j+1}^{(i)} \end{bmatrix}$, $i = 1, \ldots, d$. On the basis of these ideas, Van Beeumen et al.[20] developed CORK, splitting the method into two levels of orthogonalization: The first level is to expand Q_j into Q_{j+1}, and the second

level is to expand $\mathbf{R}_j$ into $\mathbf{R}_{j+1}$. We can see a basic pseudocode for the CORK method in Algorithm 2, whose complete explanation can be found in the work of Van Beeumen et al.[20] For simplicity, we assume that breakdown does not occur in Algorithm 2, that is, $h_{j+1,j} \neq 0$ for all j.

Algorithm 2. Compact rational Krylov method (CORK)

Input: $Q_1 \in \mathbb{C}^{n\times r_1}$ and $\mathbf{R}_1 \in \mathbb{C}^{dr_1\times 1}$ with $Q_1^*Q_1 = I_{r_1}$ and $\mathbf{R}_1^*\mathbf{R}_1 = 1$, where $r_1 \leq d$.
Output: Approximate eigenpairs $(\lambda, \mathbf{x})$ associated to $\mathbf{A} - \lambda\mathbf{B}$, with $\mathbf{A}, \mathbf{B}$ as in (6).
for $j = 1, 2, \ldots$ **do**
1. Choose shift θ_j.
First level of orthogonalization:
2. Compute $\hat{u}^{(p)}$ by using the ULP decomposition in Theorem 2 with $\mu = \theta_j$ (see the work of Van Beeumen et al.[20] for details).
3. Orthogonalize: $\tilde{q} = \hat{u}^{(p)} - Q_j Q_j^* \hat{u}^{(p)}$.
4. If $\tilde{q} \neq 0$ then compute next vector: $q_{j+1} = \tilde{q}/\|\tilde{q}\|_2$ and $Q_{j+1} = [Q_j \quad q_{j+1}]$. Otherwise $Q_{j+1} = Q_j$.
Second level of orthogonalization:
5. If $r_{j+1} > r_j$ then update matrices: $R_j^{(i)} = \begin{bmatrix} R_j^{(i)} \\ 0_{1\times j} \end{bmatrix}$ for $i = 1, \ldots, d$.
6. Compute: $\hat{\mathbf{r}}$ by using the ULP decomposition in Theorem 2 (see the work of Van Beeumen et al.[20] for details).
7. Compute: $\tilde{\mathbf{r}} = \hat{\mathbf{r}} - \mathbf{R}_j h_j$, where $h_j = \mathbf{R}_j^*\hat{\mathbf{r}}$.
8. Next vector: $\mathbf{r}_{j+1} = \tilde{\mathbf{r}}/h_{j+1,j}$, where $h_{j+1,j} = \|\tilde{r}\|_2$ and $\mathbf{R}_{j+1} = [\mathbf{R}_j \quad \mathbf{r}_{j+1}]$.
9. Compute eigenpairs: (λ_i, t_i) of (15) and test for convergence.
end for
10. Compute eigenvectors: $\mathbf{x}_i = (I_d \otimes Q_{j+1})\mathbf{R}_{j+1}\underline{H}_j t_i$.

From the discussion above, it is clear that CORK reduces significantly the storage requirements with respect to a direct application of the RK method to the $(nd) \times (nd)$ GEP $\mathbf{A} - \lambda\mathbf{B}$, since, essentially, CORK represents $\mathbf{U}_j$ in terms of $n(j + d)$ parameters and, in addition, $n(j + d) \approx nj$ for moderate values of d. Therefore, the memory cost of CORK is approximately the cost of any Krylov method applied to an $n \times n$ GEP. Moreover, it can be seen in section 5.4 in the work of Van Beeumen et al.[20] that the orthogonalization cost of CORK is essentially independent of d for moderate values of d and, hence, much lower than the orthogonalization cost of a direct application of RK to $\mathbf{A} - \lambda\mathbf{B}$. With respect to the comparison of the costs of the shift-and-invert steps in CORK (included in Step 2 of Algorithm 2) and in RK (Step 2 in Algorithm 1), we can say that in CORK, the particular structure of the pencil $\mathbf{A} - \lambda\mathbf{B}$, together with the fact that only one block of vector $\hat{\mathbf{u}}$, is needed allow us to perform this step very efficiently by essentially solving just one "difficult" $n \times n$ linear system (see algorithm 2 in the work of Van Beeumen et al.[20]). In contrast, in RK, the whole vector $\hat{\mathbf{u}}$ must be computed, and there is some extra cost with respect to CORK even in the case where the structure of $\mathbf{A} - \lambda\mathbf{B}$ is taken into account for solving the linear system $(\mathbf{A} - \theta_j\mathbf{B})\hat{\mathbf{u}} = \mathbf{B}\mathbf{u}_j$. On the other hand, there is some overhead cost involved in Step 2 of Algorithm 2, since, in CORK, the actual vector $\mathbf{u}_j$ has to be constructed before solving the linear system associated to the shift-and-invert step. Fortunately, according to (17), this computation can be arranged as the single matrix–matrix product $Q_j[r_j^{(1)} \cdots r_j^{(d)}]$, where $r_j^{(1)}, \ldots, r_j^{(d)}$ are the blocks of the last column of $\mathbf{R}_j$, which allows optimal efficiency and cache usage on modern computers (see the work of Kressner and Roman[19(p.577)]).

Inspired by CORK, we will develop in Section 3 the new algorithm R-CORK to solve large-scale and sparse REPs by using a decomposition similar to (17) for the bases of the RK subspaces associated to the linearization (9) of the REP and by working in the spirit of the two levels of orthogonalization originally introduced in TOAR.[17,18] We will see that R-CORK has memory and computational advantages similar to those discussed for CORK in the previous paragraph.

3 | A NEW METHOD FOR SOLVING LARGE-SCALE AND SPARSE REPs

3.1 | A compact decomposition for RK subspaces of $\mathcal{A} - \lambda\mathcal{B}$

Consider the matrices $\mathcal{A}$ and $\mathcal{B}$ in (10) and the RK recurrence relation (13), which is valid for arbitrary pencils. Our goal is to particularize such relation to the matrices $\mathcal{A}$ and $\mathcal{B}$ in (10) in order to save memory and orthogonalization costs. For

this purpose, we will partition $\mathbf{U}_{j+1}$ conformably into $\mathcal{A}$ and $\mathcal{B}$ as follows:

$$\mathbf{U}_{j+1} = [\mathbf{U}_j \quad \mathbf{u}_{j+1}] = \begin{bmatrix} U_j^{(1)} & u_{j+1}^{(1)} \\ U_j^{(2)} & u_{j+1}^{(2)} \\ \vdots & \vdots \\ U_j^{(d)} & u_{j+1}^{(d)} \\ V_j & v_{j+1} \end{bmatrix}, \tag{20}$$

where $U_j^{(i)} \in \mathbb{C}^{n\times j}$, $u_{j+1}^{(i)} \in \mathbb{C}^n$, for $i = 1, \ldots, d$, $V_j \in \mathbb{C}^{s\times j}$, and $v_{j+1} \in \mathbb{C}^s$. Next, following CORK for the first d blocks, we define the matrix $Q_j \in \mathbb{C}^{n\times r_j}$ such that the columns of Q_j are orthonormal with

$$\text{span}\{Q_j\} = \text{span}\left\{ U_j^{(1)}, U_j^{(2)}, \ldots, U_j^{(d)} \right\} \tag{21}$$

and $\text{rank}(Q_j) = r_j$. Using (21), we can express

$$U_j^{(i)} = Q_j R_j^{(i)}, \quad i = 1, 2, \ldots, d, \tag{22}$$

where $R_j^{(i)} \in \mathbb{C}^{r_j\times j}$ for $i = 1, 2, \ldots, d$. Then, by using (22), we have

$$\mathbf{U}_j = \begin{bmatrix} Q_j R_j^{(1)} \\ Q_j R_j^{(2)} \\ \vdots \\ Q_j R_j^{(d)} \\ V_j \end{bmatrix} = \begin{bmatrix} Q_j & & & & \\ & Q_j & & & \\ & & \ddots & & \\ & & & Q_j & \\ & & & & I_s \end{bmatrix} \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \\ V_j \end{bmatrix}. \tag{23}$$

By introducing the notation

$$\mathbf{Q}_j := \left[\begin{array}{c|c} (I_d \otimes Q_j) & 0_{dn\times s} \\ \hline 0_{s\times dr_j} & I_s \end{array}\right] \in \mathbb{C}^{(dn+s)\times(dr_j+s)} \quad \text{and} \quad \mathbf{R}_j := \begin{bmatrix} R_j^{(1)} \\ R_j^{(2)} \\ \vdots \\ R_j^{(d)} \\ V_j \end{bmatrix} \in \mathbb{C}^{(dr_j+s)\times j}, \tag{24}$$

we have $\mathbf{U}_j = \mathbf{Q}_j\mathbf{R}_j$. Note that the columns of $\mathbf{U}_j$ and $\mathbf{Q}_j$ are orthonormal; hence, matrix $\mathbf{R}_j$ also has orthonormal columns. With this notation, we can rewrite (13) as the following CORK recurrence relation:

$$\mathcal{A}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{H}_j = \mathcal{B}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{K}_j. \tag{25}$$

In order to prove that, as in CORK, we need only one vector to expand Q_j into Q_{j+1} and that, as a consequence, r_j is considerably smaller than jd, that is, that $\mathbf{Q}_j\mathbf{R}_j$ is, indeed, a compact representation of $\mathbf{U}_j$, we will prove first Lemmas 1 and 2. We emphasize the relationship between Lemma 1 and Theorem 2, which also has two differences: The first difference comes from the presence of the strictly proper part $E(C - \lambda D)^{-1}F^T$ of the rational matrix $R(\lambda)$, which motivates the definition of the rational matrix $\mathbf{A}(\lambda)$ in Lemma 1, and the second difference is related with the matrices $\mathcal{A}$ and $\mathcal{B}$ in (10). Since the matrices P_i are ordered in increasing index order in (10), it occurs that the permutation $\mathcal{P}$ in Theorem 2 is not needed in Lemma 1; therefore, we developed a UL decomposition instead of a ULP decomposition. Apart from these differences, we have stated Lemma 1 in an analogous way to Theorem 2, with the purpose of stressing the relation with CORK, but note that the simple particular structures of $M, N \in \mathbb{C}^{(d-1)\times d}$, and $\mathbf{B}$ inherited from (10)–(12) imply that in Lemma 1

$$M := [\, m_0 \,|\, M_1 \,] = \left[\begin{array}{c|ccc} 0 & -1 & & \\ \hline & \ddots & \ddots & \\ & & 0 & -1 \end{array}\right], \quad N := [\, n_0 \,|\, N_1 \,] = \left[\begin{array}{c|ccc} -1 & 0 & & \\ \hline & \ddots & \ddots & \\ & & -1 & 0 \end{array}\right] \tag{26}$$

and that $\bar{\mathbf{B}}_1$ has only one nonzero block. Therefore, the factors $\mathcal{L}(\mu)$ and $\mathcal{U}(\mu)$ in Lemma 1 are simpler than the general ones in Theorem 2. Note also that Lemma 2 is related to lemma 4.3 in the work of Van Beeumen et al.[20]; although, again, the strictly proper part of the rational matrix introduces relevant differences.

Lemma 1. *Consider a rational matrix*

$$R(\lambda) = P(\lambda) - E(C - \lambda D)^{-1}F^T \in \mathbb{C}(\lambda)^{n\times n},$$

where $P(\lambda) = \sum_{i=0}^{d} \lambda^i P_i$, $P_i \in \mathbb{C}^{n\times n}$ *for* $i = 0, \ldots, d$, $E, F \in \mathbb{C}^{n\times s}$, $C, D \in \mathbb{C}^{s\times s}$, D *is nonsingular, and* $E(C - \lambda D)^{-1}F^T$ *is a minimal realization. Define the rational matrix*

$$\mathbf{A}(\lambda) = \left[\begin{array}{c|cccc} P_0 - E(C-\lambda D)^{-1}F^T & P_1 & \cdots & P_{d-2} & P_{d-1} \\ \hline M \otimes I_n & & & & \end{array}\right]$$

and the constant matrix

$$\mathbf{B} = \left[\begin{array}{c|ccc} 0_n & 0_n & \cdots & 0_n - P_d \\ \hline N \otimes I_n & & & \end{array}\right],$$

with M *and* N *defined as in (26). Then, for every* $\mu \in \mathbb{C}$, *which is not a pole of* $R(\mu)$, *that is, such that* $(C - \mu D)$ *is nonsingular, we can factorize* $\mathbf{A}(\mu) - \mu\mathbf{B}$ *as follows:*

$$\mathbf{A}(\mu) - \mu\mathbf{B} = \mathcal{U}(\mu)\mathcal{L}(\mu), \tag{27}$$

where

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

and

$$\begin{bmatrix} P_0 - E(C-\mu D)^{-1}F^T & P_1 & \cdots & P_{d-1} \end{bmatrix} =: \begin{bmatrix} P_0 - E(C-\mu D)^{-1}F^T & \bar{\mathbf{A}}_1 \end{bmatrix},$$
$$[0_n \quad 0_n \quad \cdots \quad -P_d] =: [0_n \quad \bar{\mathbf{B}}_1].$$

Proof. Observe first that the definitions of M and N imply trivially that $M_1 - \mu N_1$ is nonsingular for every $\mu \in \mathbb{C}$. By a direct matrix multiplication, we obtain

$$\begin{aligned} \mathcal{U}(\mu)\mathcal{L}(\mu) &= \begin{bmatrix} R(\mu) + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)\left(\left((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)\right) \otimes I_n\right) & (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1) \\ (m_0 - \mu n_0) \otimes I_n & (M_1 - \mu N_1) \otimes I_n \end{bmatrix} \\ &= \left[\begin{array}{c} R(\mu) + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)\left(\left((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)\right) \otimes I_n\right) \quad (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1) \\ \hline (M - \mu N) \otimes I_n \end{array}\right]. \end{aligned}$$

Therefore, we only need to prove that

$$R(\mu) + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)\left(\left((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)\right) \otimes I_n\right) = P_0 - E(C - \mu D)^{-1}F^T,$$

which is equivalent to proving that

$$P(\mu) + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)\left(\left((M_1 - \mu N_1)^{-1}(m_0 - \mu n_0)\right) \otimes I_n\right) = P_0. \tag{28}$$

The proof of (28) is a very simple algebraic manipulation as a consequence of the extremely simple structures of m_0 and n_0, that is, M_1 and N_1 in this case. Another proof comes from the observation that (28) holds because it is proved for proving the ULP decomposition in Theorem 2 (see the work of Van Beeumen et al.[20(pp.823–824)]). □

Remark 1. Observe that Theorem 2 involves the constant $\alpha = e_1^T\mathcal{P}f(\mu)$, which is not present in Lemma 1. The reason is that in Lemma 1, this constant is equal to 1 as a consequence of the structure of (11) and that $\mathcal{P} = I_d$.

Lemma 2. *Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be the matrices defined in (10). Consider the linear system*

$$(\mathcal{A} - \mu\mathcal{B})\mathbf{x} = \mathcal{B}\mathbf{w}, \tag{29}$$

where $\mathbf{x} = [x^{(1)^T}, x^{(2)^T}, \ldots, x^{(d)^T}, y^T]^T$ *and* $\mathbf{w} = [w^{(1)^T}, w^{(2)^T}, \ldots, w^{(d)^T}, z^T]^T$, *the blocks* $x^{(i)}, w^{(i)} \in \mathbb{C}^n$, $i = 1, 2, \ldots, d$, $y, z \in \mathbb{C}^s$, *and* μ *is not a pole of* $R(\mu)$, *that is,* $(C - \mu D)$ *is nonsingular. Then, the block* $x^{(1)}$ *of* $\boldsymbol{x}$ *can be computed by solving the following* $n \times n$ *linear system whose coefficient matrix is* $R(\mu)$ *in (8):*

$$R(\mu)x^{(1)} = -P_d w^{(d)} - E(C - \mu D)^{-1}Dz + (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)\left((M_1 - \mu N_1)^{-1} \otimes I_n\right) w^{(1,\ldots,d-1)},$$

where the matrices introduced in Lemma 1 are used and $w^{(1,\dots,d-11)} = [w^{(1)^T}, \dots, w^{(d-1)^T}]^T$. *The remaining blocks* $x^{(i)}$ *for* $i = 2, \dots, d$ *of* $\mathbf{x}$ *can be obtained as linear combinations of* $x^{(1)}$ *and* $w^{(i)}$, $i = 1, \dots, d-1$. *Also,* $x^{(2,\dots,d)} = [x^{(2)^T}, \dots, x^{(d)^T}]^T$ *satisfies the linear system*

$$x^{(2,\dots,d)} = -\left((M_1 - \mu N_1)^{-1} \otimes I_n\right)\left(w^{(1,\dots,d-1)} + ((m_0 - \mu n_0) \otimes I_n)x^{(1)}\right).$$

In addition, y *can be computed by solving the* $s \times s$ *linear system*

$$(C - \mu D)y = Dz - F^T x^{(1)}.$$

Proof. Rewrite the matrix pencil (9) as in (12), that is,

$$\mathcal{A} - \mu\mathcal{B} = \left[\begin{array}{c|c} \mathbf{A} - \mu\mathbf{B} & e_1 \otimes E \\ \hline e_1^T \otimes F^T & C - \mu D \end{array}\right]$$

with

$$\mathbf{A} = \left[\begin{array}{ccccc} P_0 & \cdots & P_{d-2} & P_{d-1} \\ \hline & & M \otimes I_n & \end{array}\right], \quad \mathbf{B} = \left[\begin{array}{ccccc} 0_n & \cdots & 0_n - P_d \\ \hline & N \otimes I_n & \end{array}\right],$$

and M and N defined as in (26). Then, we can solve system (29) by solving

$$(\mathbf{A} - \mu\mathbf{B})x^{(1,2,\dots,d)} + (e_1 \otimes E)y = \mathbf{B}w^{(1,2,\dots,d)}, \tag{30}$$

$$\left(e_1^T \otimes F^T\right)x^{(1,2,\dots,d)} + (C - \mu D)y = Dz, \tag{31}$$

where $x^{(1,2,\dots,d)} = [x^{(1)^T}, x^{(2)^T}, \dots, x^{(d)^T}]^T$ and $w^{(1,2,\dots,d)} = [w^{(1)^T}, w^{(2)^T}, \dots, w^{(d)^T}]^T$. The second equation is the equation for y in the statement. By replacing $y = (C - \mu D)^{-1}(Dz - F^T x^{(1)})$ from (31) in (30) and using the notation of Lemma 1, we obtain

$$\begin{aligned}
(\mathbf{A} - \mu\mathbf{B})x^{(1,2,\dots,d)} - (e_1 \otimes E)(C - \mu D)^{-1}F^T x^{(1)} &= \mathbf{B}w^{(1,2,\dots,d)} - (e_1 \otimes E)(C - \mu D)^{-1}Dz, \\
\left[\begin{array}{ccccc} P_0 - E(C - \mu D)^{-1}F^T & \cdots & P_{d-2} & P_{d-1} + \mu P_d \\ \hline & & (M - \mu N) \otimes I_n & \end{array}\right]x^{(1,2,\dots,d)} &= -\left[\begin{array}{c} P_d w^{(d)} + E(C - \mu D)^{-1}Dz \\ \hline w^{(1,\dots,d-1)} \end{array}\right], \\
(\mathbf{A}(\mu) - \mu\mathbf{B})x^{(1,2,\dots,d)} &= -\left[\begin{array}{c} P_d w^{(d)} + E(C - \mu D)^{-1}Dz \\ \hline w^{(1,\dots,d-1)} \end{array}\right].
\end{aligned}$$

By combining factorization (27) in Lemma 1 and the equation above, it is immediate to see that the blocks $x^{(i)}$ for $i = 2, \dots, d$ of $\mathbf{x}$ are linear combinations of $x^{(1)}$ and the blocks $w^{(i)}$, $i = 1, \dots, d-1$. In addition, some elementary matrix manipulations with the matrices $\mathcal{U}(\mu)$ and $\mathcal{L}(\mu)$ in (27) lead to the equations for $x^{(2,\dots,d)}$ and $x^{(1)}$ in the statement. This finishes the proof. □

As announced before, Lemma 2 is the key result that allows us to prove through Theorems 5 and 6 that only one vector is needed to expand Q_j into Q_{j+1} and, hence, that the representation (23) for $\mathbf{U}_j$ is indeed compact. Moreover, the equations for $x^{(1)}$, $x^{(2,\dots,d)}$, and y deduced in Lemma 2 lead to the efficient Algorithm 3 for solving the linear system (29), which is fundamental for performing efficiently the shift-and-invert step in the R-CORK method developed in Section 3.2. Observe that in Algorithm 3, a notation similar to that in Lemma 2 is used.

Algorithm 3. Solve for the linear system $(\mathcal{A} - \mu\mathcal{B})\mathbf{x} = \mathcal{B}\mathbf{w}$, with $\mathcal{A}$ and $\mathcal{B}$ as in (10)

Input: $\mathcal{A}, \mathcal{B} \in \mathbb{C}^{(nd+s)\times(nd+s)}$ as in (10), $\mu \in \mathbb{C}$ such that $(C - \mu D)^{-1}$ exists and $\mathbf{w} \in \mathbb{C}^{nd+s}$.

Output: The solution $\mathbf{x}$ of the linear system.

1. Compute $x = \mathbf{B}w^{(1,2,\dots,d)} - (e_1 \otimes E)(C - \mu D)^{-1}Dz$ as $x = -\left[\frac{P_d w^{(d)} + E(C-\mu D)^{-1}Dz}{w^{(1,\dots,d-1)}}\right]$.

Solve the block upper triangular system associated to $\mathcal{U}(\mu)$ in (27):

2. $x^{(1)} = x^{(1)} - (\bar{\mathbf{A}}_1 - \mu\bar{\mathbf{B}}_1)((M_1 - \mu N_1)^{-1} \otimes I_n)x^{(2,\dots,d)}$.

Solve the block lower triangular system associated to $\mathcal{L}(\mu)$ in (27):

3. $x^{(1)} = (R(\mu))^{-1}x^{(1)}$.
4. $x^{(2,\dots,d)} = ((M_1 - \mu N_1)^{-1} \otimes I_n)(x^{(2,\dots,d)} - ((m_0 - \mu n_0) \otimes I_n)x^{(1)})$.

Compute the block y of $\mathbf{x}$

5. $y = (C - \mu D)^{-1}(Dz - F^T x^{(1)})$.

Remark 2. The multiplications by inverses in Algorithm 3 have to be understood, in principle, as solutions of linear systems, and the key observation in Algorithm 3 is that all the involved linear systems have sizes smaller than the size $(nd + s) \times (nd + s)$ of $(\mathcal{A} - \mu\mathcal{B})$, as we discuss in this remark. The only linear system that is always large is the one in Line 3 involving $R(\mu)$, which has the size $n \times n$ of the original REP. Solving the system in Line 3 may require just the ability of multiplying by $R(\mu)$, if an iterative Krylov method is used, which might be done through the coefficients of $P(\lambda)$ and the matrices E, C, D, F in (8) without computing $R(\mu)$, or may require to compute $R(\mu)$, if a direct method is used. In the case where $(C - \mu D)$ is large and complicated, the computation of $R(\mu)$ might be performed more efficiently through (2) than through (8), although this depends on each particular problem. However, we emphasize once again that the matrix $(C - \mu D) \in \mathbb{C}^{s\times s}$ is, in many applications,[4,8] very small, since $s \ll n$, and has, in addition, a very simple structure, which implies that it is often possible just to compute $(C - \mu D)^{-1}$ and to perform the corresponding matrix multiplications to construct $R(\mu)$ through (8). These comments on the size $s \ll n$ also apply to the linear systems involving $(C - \mu D) \in \mathbb{C}^{s\times s}$ in Lines 1 and 5, which are very often, in practice, very small. Finally, the linear systems involving $(M_1 - \mu N_1) \otimes I_n$ have size $(d - 1)n \times (d - 1)n$ and look very large, but they are block linear systems very easy to solve with cost $2n(d - 2)$ flops by using a simple two-term recurrence relation. More precisely, the solution of $((M_1 - \mu N_1) \otimes I_n)\mathbf{x} = \mathbf{b}$, taking into account that

$$M_1 - \mu N_1 = \begin{bmatrix} -1 & & & \\ \mu & -1 & & \\ & \ddots & \ddots & \\ & & \mu & -1 \end{bmatrix} \in \mathbb{C}^{(d-1)\times(d-1)}$$

and partitioning the vectors in $(d - 1)$ blocks of size $n \times 1$, can be obtained as $x^{(1)} = -b^{(1)}$ and $x^{(i)} = \mu x^{(i-1)} - b^{(i)}$ for $i = 2, 3, \ldots, d - 1$.

The following theorems are similar to the results obtained in theorems 4.4 and 4.5 in the work of Van Beeumen et al.[20]

Theorem 5. *Let Q_j be defined as in (21). Then, we have*

$$\operatorname{span}\{Q_{j+1}\} = \operatorname{span}\left\{ Q_j, u_{j+1}^{(1)} \right\}. \tag{32}$$

Proof. The proof is immediate from definition (21) and Lemma 2 with $\mu = \theta_j$ and $\mathbf{w} = \mathbf{u}_j$ (see the proof of theorem 4.4 in the work of Van Beeumen et al.[20]). □

Theorem 6. *Let Q_j be defined as in (21). Then, we have*

$$r_j < d + j. \tag{33}$$

Proof. We will prove this theorem by induction. From the definition of Q_j in (21), we have that

$$\operatorname{span}\{Q_1\} = \operatorname{span}\left\{ u_1^{(1)}, u_1^{(2)}, \ldots, u_1^{(d)} \right\};$$

thus, $r_1 \leq d$. Assuming that inequality (33) is satisfied until $j - 1$, then we have, by Theorem 5, that $r_j \leq r_{j-1} + 1 < d + j$. □

By considering inequality (33) and the fact that r_j increases, at most, by 1 in each iteration, we will show the possible structures of the expansion of the first d blocks of the matrix $\mathbf{R}_j$ defined in (24).

Lemma 3. *Let $\mathbf{R}_j \in \mathbb{C}^{(dr_j+s)\times j}$ be defined as in (24). Then, the first d blocks of the matrix $\mathbf{R}_{j+1} \in \mathbb{C}^{(dr_{j+1}+s)\times(j+1)}$ can take the following forms:*

- *if $r_{j+1} > r_j$, then*

$$R_{j+1}^{(i)} = \begin{bmatrix} R_j^{(i)} & r_{j+1}^{(i)} \\ 0_{1\times j} & \end{bmatrix}, \quad i = 1, 2, \ldots, d,$$

  where $r_{j+1}^{(i)} \in \mathbb{C}^{r_{j+1}}$, or
- *if $r_{j+1} = r_j$, then*

$$R_{j+1}^{(i)} = \begin{bmatrix} R_j^{(i)} & r_{j+1}^{(i)} \end{bmatrix}, \quad i = 1, 2, \ldots, d,$$

  with $r_{j+1}^{(i)} \in \mathbb{C}^{r_{j+1}}$.

3.2 | The R-CORK method

In this section, we will introduce the method to solve large-scale and sparse REPs based on the compact representation presented in Section 3.1 of the orthonormal bases of the RK subspaces of the linearization $\mathcal{A}-\lambda\mathcal{B}$ in (9). First, we consider an initial vector $\mathbf{u}_1 \in \mathbb{C}^{nd+s}$ with $\|\mathbf{u}_1\|_2 = 1$ partitioned as in (20), and then, we express that vector in a compact form as follows:

$$\mathbf{u}_1 = \begin{bmatrix} u_1^{(1)} \\ \vdots \\ u_1^{(d)} \\ v_1 \end{bmatrix} = \begin{bmatrix} Q_1 R_1^{(1)} \\ \vdots \\ Q_1 R_1^{(d)} \\ v_1 \end{bmatrix},$$

where $Q_1 \in \mathbb{C}^{n\times r_1}$ has orthonormal columns such that

$$\operatorname{span}\{Q_1\} = \operatorname{span}\left\{u_1^{(1)}, \ldots, u_1^{(d)}\right\}, \quad r_1 = \operatorname{rank}\left(\left[u_1^{(1)} \ldots u_1^{(d)}\right]\right).$$

Observe that $r_1 = 1$ if and only if $\mathbf{u}_1$ is chosen to have collinear nonzero blocks $u_1^{(1)}, \ldots, u_1^{(d)}$. Now, taking into account the definition of $\mathbf{R}_j$ in (24), after j steps, we want to expand Q_j into Q_{j+1} and $\mathbf{R}_j$ into $\mathbf{R}_{j+1}$, which results in the so-called two levels of orthogonalization.

First level of orthogonalization. In Theorem 5, we have proved that we need to orthogonalize $u_{j+1}^{(1)}$ with respect to Q_j to obtain the last orthonormal column of Q_{j+1}. In addition, it can be easily seen that

$$\operatorname{span}\{Q_{j+1}\} = \operatorname{span}\left\{Q_j, u_{j+1}^{(1)}\right\} = \operatorname{span}\left\{Q_j, \hat{u}^{(1)}\right\}, \tag{34}$$

where $\hat{u}^{(1)}$ is the first block of size n of the vector $\hat{\mathbf{u}}$ obtained by applying the shift-and-invert step to $\mathbf{u}_j$ (Step 2 in Algorithm 1) when $\hat{\mathbf{u}}$ is partitioned as in (20). Therefore, we only need to compute the block $\hat{u}^{(1)}$ of $\hat{\mathbf{u}}$ to compute Q_{j+1}. Thus, we can run Algorithm 3 with $\mathbf{w} = \mathbf{u}_j$ and $\mu = \theta_j$ until Step 3, saving the resulting vector $x^{(1)} = \hat{u}^{(1)}$. It is important to observe that the first d blocks of $\mathbf{u}_j$ have to be constructed, since the variables in R-CORK are Q_j and $\mathbf{R}_j$, and $\mathbf{u}_j$ is not stored. As in CORK, they are computed as the single matrix–matrix product $Q_j\,[r_j^{(1)}\cdots r_j^{(d)}]$, where $r_j^{(1)}, \ldots, r_j^{(d)}$ are the first d blocks of the last column $\mathbf{r}_j$ of $\mathbf{R}_j$, which is a very efficient computation in terms of cache utilization on modern computers. Once $\hat{u}^{(1)}$ is available, by (34), we can decompose

$$\hat{u}^{(1)} = Q_j x_j + \alpha_j q_{j+1}, \tag{35}$$

where q_{j+1} is a unit vector orthogonal to Q_j and $x_j = Q_j^* \hat{u}^{(1)}$. Observe also that since $\hat{u}^{(1)}$ has been already computed, we can compute the last s entries of $\hat{\mathbf{u}}$, denoted by $\hat{v}$, from Step 5 in Algorithm 3 without the need of performing Step 4. Vector $\hat{v}$ will be used in the second level of orthogonalization. Now, if $\hat{u}^{(1)}$ does not lie in the subspace spanned by the columns of Q_j, that is, if $\hat{u}^{(1)} - Q_j x_j \neq 0$, we can expand Q_j into Q_{j+1} as follows:

$$Q_{j+1} = [Q_j \quad q_{j+1}], \quad r_{j+1} = r_j + 1.$$

On the other hand, if $\hat{u}^{(1)}$ lies in the subspace spanned by the columns of Q_j, we have $Q_{j+1} = Q_j$ and $r_{j+1} = r_j$. We summarize the first level of orthogonalization in Algorithm 4. In Step 2, if it is necessary, we can reorthogonalize $\tilde{q}$ to ensure orthogonality. In fact, in our MATLAB code, we perform the classical Gram–Schmidt method twice.

Algorithm 4. First level of orthogonalization in R-CORK

Input: The matrix $Q_j \in \mathbb{C}^{n\times r_j}$ and the vector $\hat{u}^{(1)} \in \mathbb{C}^n$ (the first block of $\hat{\mathbf{u}} = (\mathcal{A}-\theta_j\mathcal{B})^{-1}\mathcal{B}\mathbf{u}_j$).
Output: The matrix $Q_{j+1} \in \mathbb{C}^{n\times r_{j+1}}$, the vector x_j, and the scalar α_j.

Expanding Q_j into Q_{j+1}.
1. $x_j = Q_j^* \hat{u}^{(1)}$.
2. $\tilde{q} = \hat{u}^{(1)} - Q_j x_j$.
3. $\alpha_j = \|\tilde{q}\|_2$.
if $\alpha_j \neq 0$ **then**
  4a. $Q_{j+1} = [Q_j \quad \tilde{q}/\alpha_j]$.
  5a. $r_{j+1} = r_j + 1$.
else
  4b. $Q_{j+1} = Q_j$.
  5b. $r_{j+1} = r_j$.
end if

Second level of orthogonalization. In Algorithm 1, after choosing the shift and performing the shift-and-invert step, we need to compute the entries of the jth column of $\underline{H}_j$ in Step 3. Let us see how to do it efficiently in R-CORK. By using the compact representation of $\mathbf{U}_j$ in (23) and (24), we have

$$\begin{aligned} h_j &= \mathbf{U}_j^* \hat{\mathbf{u}}, \\ &= \left(R_j^{(1)}\right)^* Q_j^* \hat{u}^{(1)} + \cdots + \left(R_j^{(d)}\right)^* Q_j^* \hat{u}^{(d)} + V_j^* \hat{v}, \end{aligned} \tag{36}$$

where $\hat{\mathbf{u}}$ has been partitioned in an analogous way to (20). Since $(\mathcal{A} - \theta_j \mathcal{B})\hat{\mathbf{u}} = \mathcal{B}\mathbf{u}_j$ and $\mathcal{A}$ and $\mathcal{B}$ have the structures in (10), we obtain the following relation between the blocks of size n of $\hat{\mathbf{u}}$ and the blocks of size n of $\mathbf{u}_j$:

$$\hat{u}^{(i)} = \theta_j \hat{u}^{(i-1)} + u_j^{(i-1)}, \quad \text{for} \quad i = 2, \ldots, d. \tag{37}$$

Motivated by (37), we consider the vectors $x_j \in \mathbb{C}^{r_j}$ obtained in Step 1 of Algorithm 4 and $\hat{v} \in \mathbb{C}^s$ obtained in Step 5 of Algorithm 4, with $x^{(1)} = \hat{u}^{(1)}$ and $\mu = \theta_j$, and a vector $\hat{\mathbf{p}} \in \mathbb{C}^{dr_j+s}$ partitioned as follows:

$$\hat{\mathbf{p}} = \begin{bmatrix} \hat{p}^{(1)} \\ \vdots \\ \hat{p}^{(d)} \\ \hat{v} \end{bmatrix}, \quad \hat{p}^{(i)} \in \mathbb{C}^{r_j}, \quad i = 1, \ldots, d, \tag{38}$$

with the blocks defined by the recurrence relation

$$\begin{aligned} \hat{p}^{(1)} &= x_j, \\ \hat{p}^{(i)} &= \theta_j \hat{p}^{(i-1)} + r_j^{(i-1)}, \quad i = 2, \ldots, d, \end{aligned} \tag{39}$$

where $r_j^{(i)}$ represents the jth column of the block $R_j^{(i)}$ in (23). If $\alpha_j \neq 0$ in Step 3 of Algorithm 4, by using $\hat{\mathbf{p}}$, the decomposition (35), and the recurrence relation (37), the vectors $\hat{u}^{(i)}$, $i = 1, \ldots, d$, corresponding to the partition of $\hat{\mathbf{u}}$ as in (20) can be represented as follows:

$$\hat{u}^{(i)} = Q_{j+1} \begin{bmatrix} \hat{p}^{(i)} \\ \theta_j^{i-1} \alpha_j \end{bmatrix}, \quad i = 1, \ldots, d, \tag{40}$$

whereas if $\alpha_j = 0$, we can represent the blocks $\hat{u}^{(i)}$, $i = 1, \ldots, d$, as follows:

$$\hat{u}^{(i)} = Q_j \hat{p}^{(i)}. \tag{41}$$

Then, by using either (40) or (41) (depending on the value of α_j) in (36) and recalling that the columns of Q_{j+1} are orthonormal, we have

$$h_j = \begin{bmatrix} R_j^{(1)} \\ \vdots \\ R_j^{(d)} \\ V_j \end{bmatrix}^* \begin{bmatrix} \hat{p}^{(1)} \\ \vdots \\ \hat{p}^{(d)} \\ \hat{v} \end{bmatrix} = \mathbf{R}_j^* \hat{\mathbf{p}}. \tag{42}$$

Thus, after computing $\hat{\mathbf{p}}$ with the recurrence relation (39), we can compute h_j by performing a matrix–vector multiplication of size $dr_j + s$, which, according to (33), is much smaller than $dn + s$ in large-scale problems and, even more, much smaller than n whenever $s \ll n$, as often happens in applications.[4,8]

Next, in Step 4 of Algorithm 1, we need to compute vector $\tilde{\mathbf{u}}$, which means that in R-CORK, we need its compact representation. By using the compact representation of $\mathbf{U}_j$ and (40), if $\alpha_j \neq 0$, we have

$$\begin{aligned}\tilde{\mathbf{u}} &= \hat{\mathbf{u}} - \mathbf{U}_j h_j,\\ &= \begin{bmatrix} Q_{j+1}\begin{bmatrix}\hat{p}^{(1)}\\ \alpha_j\end{bmatrix}\\ \vdots\\ Q_{j+1}\begin{bmatrix}\hat{p}^{(d)}\\ \theta_j^{d-1}\alpha_j\end{bmatrix}\\ \hat{v}\end{bmatrix} - \begin{bmatrix} Q_j R_j^{(1)}\\ \vdots\\ Q_j R_j^{(d)}\\ V_j\end{bmatrix} h_j,\\ &= \begin{bmatrix} Q_{j+1} & & & \\ & \ddots & & \\ & & Q_{j+1} & \\ & & & I_s\end{bmatrix}\begin{bmatrix}\begin{bmatrix}\hat{p}^{(1)} - R_j^{(1)} h_j\\ \alpha_j\end{bmatrix}\\ \vdots\\ \begin{bmatrix}\hat{p}^{(d)} - R_j^{(d)} h_j\\ \theta_j^{d-1}\alpha_j\end{bmatrix}\\ \hat{v} - V_j h_j\end{bmatrix},\end{aligned}$$

and in a similar way, if $\alpha_j = 0$, we obtain

$$\tilde{\mathbf{u}} = \begin{bmatrix} Q_j & & & \\ & \ddots & & \\ & & Q_j & \\ & & & I_s\end{bmatrix}\begin{bmatrix}\hat{p}^{(1)} - R_j^{(1)} h_j\\ \vdots\\ \hat{p}^{(d)} - R_j^{(d)} h_j\\ \hat{v} - V_j h_j\end{bmatrix}.$$

Defining

$$\tilde{\mathbf{p}} := \begin{bmatrix}\tilde{p}^{(1)}\\ \vdots\\ \tilde{p}^{(d)}\\ \tilde{v}\end{bmatrix}, \quad \tilde{p}^{(i)} := \hat{p}^{(i)} - R_j^{(i)} h_j \in \mathbb{C}^{r_j},\ i = 1, \ldots, d, \quad \tilde{v} := \hat{v} - V_j h_j \in \mathbb{C}^s, \tag{43}$$

and taking into account that the columns of Q_{j+1} and Q_j are orthonormal, we can express Step 5 in Algorithm 1 as follows: If $\alpha_j \neq 0$, then

$$h_{j+1,j} = \|\tilde{\mathbf{u}}\|_2 = \left\|\begin{bmatrix}\tilde{p}^{(1)}\\ \alpha_j\\ \vdots\\ \tilde{p}^{(d)}\\ \theta_j^{d-1}\alpha_j\\ \tilde{v}\end{bmatrix}\right\|_2 \text{ and } \quad \mathbf{u}_{j+1} = \mathbf{Q}_{j+1}\cdot\frac{1}{h_{j+1,j}}\begin{bmatrix}\tilde{p}^{(1)}\\ \alpha_j\\ \vdots\\ \tilde{p}^{(d)}\\ \theta_j^{d-1}\alpha_j\\ \tilde{v}\end{bmatrix}, \tag{44}$$

where the notation in (24) is used, whereas if $\alpha_j = 0$, we proceed as in (44) by removing all the entries involving α_j and with $\mathbf{Q}_{j+1} = \mathbf{Q}_j$. From the previous equations, we can conclude that the first d blocks of size r_{j+1} of the last column of $\mathbf{R}_{j+1}$ in (24) are given, if $\alpha_j \neq 0$, by

$$r_{j+1}^{(i)} = \frac{1}{h_{j+1,j}}\begin{bmatrix}\tilde{p}^{(i)}\\ \theta_j^{i-1}\alpha_j\end{bmatrix}, \quad i = 1, \ldots, d, \tag{45}$$

and, if $\alpha_j = 0$, by

$$r_{j+1}^{(i)} = \frac{1}{h_{j+1,j}}\tilde{p}^{(i)}, \quad i = 1, \ldots, d. \tag{46}$$

In addition, the last block of size s of the last column of $\mathbf{R}_{j+1}$ is

$$v_{j+1} = \frac{1}{h_{j+1,j}}\tilde{v}. \tag{47}$$

Since $\mathbf{R}_j$ has orthonormal columns, from (42) and the definitions in (38), (39), and (43), we have that h_j and $\tilde{\mathbf{p}}$ satisfy

$$\tilde{\mathbf{p}} = \hat{\mathbf{p}} - \mathbf{R}_j h_j, \tag{48}$$

where $\tilde{\mathbf{p}}$ is orthogonal to $\mathbf{R}_j$. This process is the Gram–Schmidt process without the normalization step, and it is summarized in Algorithm 5.

Algorithm 5. Second level of orthogonalization in R-CORK

Input: The matrix $\mathbf{R}_j \in \mathbb{C}^{(dr_j+s)\times j}$ and the vector $\hat{\mathbf{p}} \in \mathbb{C}^{dr_j+s}$ from (38) and (39).
Output: Vectors $h_j \in \mathbb{C}^j$ and $\tilde{\mathbf{p}} \in \mathbb{C}^{dr_j+s}$.
1. $h_j = \mathbf{R}_j^* \hat{\mathbf{p}}$.
2. $\tilde{\mathbf{p}} = \hat{\mathbf{p}} - \mathbf{R}_j h_j$.

Remark 3. In order to improve orthogonality, a reorthogonalization method can be included in Algorithm 5. In our MATLAB code, we use the classical Gram–Schmidt process twice.

The whole procedure of this new method to solve large-scale and sparse REPs requires the use of the two levels of orthogonalization described in this section, where the first level is to expand Q_j into Q_{j+1} and the second level is to expand $\mathbf{R}_j$ into $\mathbf{R}_{j+1}$. The complete R-CORK method is summarized in Algorithm 6. Note that R-CORK has, as inputs, the matrix Q_1 and the vector $\mathbf{R}_1$, which have to be computed. As in CORK (see the work of Van Beeumen et al.[20(p.830)]), there are two possible ways of computing these inputs: either starting with a random vector $\mathbf{u}_1 \in \mathbb{C}^{nd+s}$ and using an economy-size QR factorization or emulating the structure of the eigenvectors (11) of the linearization in (9). The details are very similar to the ones in the work of Van Beeumen et al.[20(p.830)] and are omitted. Recall in Algorithm 6 that $\mathbf{r}_j$ denotes the last column of the matrix $\mathbf{R}_j$ in (24).

Algorithm 6. Compact rational Krylov method for REP (R-CORK)

Input: $Q_1 \in \mathbb{C}^{n\times r_1}$, $\mathbf{R}_1 \in \mathbb{C}^{(dr_1+s)\times 1}$ with $Q_1^* Q_1 = I_{r_1}$ and $\mathbf{R}_1^* \mathbf{R}_1 = 1$.
Output: Approximate eigenpairs $(\lambda, \mathbf{x})$ of $\mathcal{A} - \lambda\mathcal{B}$ with $\mathcal{A}$ and $\mathcal{B}$ as in (10).
for $j = 1, 2, \ldots$ **do**
1. Choose shift θ_j.
2. Compute $\mathbf{u}_j = \mathbf{Q}_j \mathbf{r}_j$, obtaining the first d blocks as matrix–matrix product $Q_j\,[r_j^{(1)} \cdots r_j^{(d)}]$.
3. Compute $\hat{u}^{(1)}$ by using Algorithm 3 until Step 3 applied to $\mathbf{w} = \mathbf{u}_j$ and $\mu = \theta_j$.
4. Compute $\hat{v}$ from Step 5 in Algorithm 3.

First level of orthogonalization

5. Run Algorithm 4 obtaining Q_{j+1}, the scalar α_j and the vector x_j.

Second level of orthogonalization:

6. Compute $\hat{\mathbf{p}}$ in (38) via the recurrence relation in (39).
7. Run Algorithm 5 obtaining $\tilde{\mathbf{p}}$ and h_j.
8. Compute $h_{j+1,j}$ and $\mathbf{r}_{j+1}$ using (44)–(47) and get $\mathbf{R}_{j+1}$ with Lemma 3.
9. Compute eigenpairs: (λ_i, t_i) of (15) and test for convergence.

end for
10. Compute eigenvectors: $\mathbf{x}_i = \mathbf{Q}_{j+1} \mathbf{R}_{j+1} \underline{H}_j t_i$.

3.3 | Memory and computational costs

In this section, we discuss the memory and computational costs of R-CORK and compare these costs with those of the classical RK method, that is, Algorithm 1, applied directly to the linearization $\mathcal{A} - \lambda\mathcal{B}$ of the REP in (9). In order to simplify the results, we will take $r_j = j + d$ in R-CORK, which is the upper bound in Theorem 6 and essentially corresponds to start the R-CORK iteration with $Q_1 \in \mathbb{C}^{n\times d}$ ($r_1 = d$) or, equivalently, with a random initial vector $\mathbf{u}_1$ whose first d blocks in partition (20) are linearly independent. If the first d blocks of $\mathbf{u}_1$ are taken to be collinear, then one can take $r_j = j$ and to improve even more the costs of R-CORK. In addition, note that we estimate the costs for any value of s, where $s \times s$ is the size of the lower-right block $C - \lambda D$ of $\mathcal{A} - \lambda\mathcal{B}$ appearing in the strictly proper part of the REP (8). This way, it will be seen that even if $s \approx n$, R-CORK has considerable advantages with respect to RK in terms of memory and computational costs. However, we emphasize that such advantages are still much more relevant when $s \ll n$, as happens very often in applications.[4,8]

For the memory costs, after j iterations, R-CORK stores $Q_j \in \mathbb{C}^{n\times r_j}$ and $\mathbf{R}_j \in \mathbb{C}^{(dr_j+s)\times j}$, which amounts to $(n + dj)(j + d) + sj \approx n(j + d) + sj$ numbers. Note that the approximation $n + dj \approx n$ holds in any reasonable large-scale REP. In

TABLE 1 Orthogonalization and memory costs for the classical rational Krylov method and the R-CORK method after j iterations

	Classical rational Krylov method	R-CORK method
Orthogonalization cost	$\mathcal{O}(j^2nd + j^2s)$	$\mathcal{O}(j^2n + jdn + j^2s)$
Memory cost	$ndj + sj$	$n(j + d) + sj$

contrast, RK stores $\mathbf{U}_j$, which amounts to $(nd + s)j = ndj + sj$ numbers. Since $(j + d) < dj$ for most reasonable choices of j and degrees d appearing in practice, we see that R-CORK is much more memory efficient than RK. These memory costs are shown in Table 1.

With respect to the computational costs, observe that for both R-CORK and RK, the cost is the sum of (a) the shift-and-invert step and (b) the orthogonalization steps. Let us analyze first the shift-and-invert steps. If the shift-and-invert step in RK, that is, Step 2 in Algorithm 1, is performed by applying an unstructured solver to the $(nd + s) \times (nd + s)$ linear system $(\mathcal{A}-\theta_j\mathcal{B})\hat{\mathbf{u}} = \mathcal{B}\mathbf{u}_j$, then the cost of RK is much larger than the cost of R-CORK, since R-CORK solves this system with Algorithm 3 (removing Step 4), which is much more efficient because it requires the solution of smaller linear systems (essentially, see Remark 2, one of size $n \times n$ and two of size $s \times s$, which are very often extremely small). However, one can consider to perform the shift-and-invert step in RK with Algorithm 3, but this is still somewhat more expensive than R-CORK, because for RK, it is needed to perform Step 4 of Algorithm 3, with an additional cost of $2n(d - 2)$ flops in each iteration (see Remark 2). A final important remark on the shift-and-invert step is that R-CORK involves the overhead cost of constructing $\mathbf{u}_j$ in Step 2 of Algorithm 6, which, in RK, is not needed. However, note that, as explained in the previous sections, this construction can be performed as in CORK via a single matrix–matrix product, which allows for optimal efficiency and cache utilization on modern computers (see the work of Kressner and Roman[19(p.577)]). Moreover, we emphasize that a traditional construction of $\mathbf{u}_j$ in R-CORK costs $\mathcal{O}(dnr_j) = \mathcal{O}(dn(j + d)) \approx \mathcal{O}(dnj)$ flops at iteration j, which, added to the orthogonalization cost of R-CORK discussed below, would give a cost of the same order of the orthogonalization cost of RK.

Finally, we discuss the orthogonalization costs of RK and R-CORK. In RK, the orthogonalization is performed in Steps 3–5 of Algorithm 1 and its cost is well known to be $\mathcal{O}(j(nd + s)) = \mathcal{O}(jnd + js)$ flops at iteration j, which amounts to $\mathcal{O}(j^2nd + j^2s)$ flops in the first j iterations (see Table 1). In R-CORK, the orthogonalization is performed in Steps 5–8 of Algorithm 6. At iteration j, the cost of Step 5 is $\mathcal{O}(r_jn) = \mathcal{O}((j + d)n)$ flops; the cost of Step 6 is $\mathcal{O}(r_jd) = \mathcal{O}((j + d)d)$ flops, which is negligible with respect to the cost of Step 5; the cost of Step 7 is $\mathcal{O}(j(dr_j + s)) = \mathcal{O}(jd(j + d) + js)$ flops; and the cost of Step 8 is $\mathcal{O}(dr_j + s) = \mathcal{O}(d(j + d) + s)$ flops. Therefore, the total cost at iteration j of the orthogonalization in R-CORK is $\mathcal{O}((n + jd)(j + d) + js) \approx \mathcal{O}(n(j + d) + js)$, where we have used again the approximation $n + jd \approx n$, which gives $\mathcal{O}(j^2n + jdn + j^2s)$ flops in the first j iterations (see Table 1). Observe that the orthogonalization cost of RK includes the large term j^2nd, which is not present in the cost of R-CORK. Therefore, the orthogonalization cost of R-CORK is considerably smaller than the one of RK.

In Table 1, we summarize the comparison of the costs between R-CORK and RK.

4 | IMPLICIT RESTARTING IN R-CORK

Practical implementations of any Krylov-type method for computing the eigenvalues of large-scale problems require effective restarting strategies. The goal of this section is to develop an implicit restarting strategy for R-CORK that restarts both Q_j and $\mathbf{R}_j$ in the compact representation of $\mathbf{U}_j$ in (23) and (24). Since R-CORK shares many of the properties of CORK, the results of this section are similar to those in section 6 in the work of Van Beeumen et al.,[20] which, in turn, are based on implicit restarting procedures for classical RK methods[28] and on the Krylov–Schur restart developed for TOAR in section 4.2 in the work of Kressner and Roman.[19]

Following the Krylov–Schur spirit[29] (see also section 5.2 in the work of Stewart[30]), the restarting technique we propose transforms first the matrices $\underline{H}_j$ and $\underline{K}_j$ in (25) into (quasi)triangular form, in order to reorder the Ritz values and to preserve the desired ones with an RK subspace of smaller dimension. Second, by representing the new smaller Krylov subspace in its compact form in an efficient way, the implicit restart of R-CORK is completed. The main difference of the process described below with respect to the implicit restarting in section 6 in the work of Van Beeumen et al.[20] is that, here, we need to add a new block of size $s \times s$ corresponding to the rational part of $R(\lambda)$ in (8).

Suppose that after j iterations, we have the RK recurrence relation in its compact form as in (25), that is,

$$\mathcal{A}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{H}_j = \mathcal{B}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\underline{K}_j, \tag{49}$$

and we want to reduce this representation to a smaller compact rational decomposition of size p, $p < j$, that is,

$$\mathcal{A}\mathbf{Q}^+_{p+1}\mathbf{R}^+_{p+1}\underline{H}^+_p = \mathcal{B}\mathbf{Q}^+_{p+1}\mathbf{R}^+_{p+1}\underline{K}^+_p, \quad p < j.$$

For this purpose, we consider the generalized Schur decomposition, as follows:

$$H_j = \begin{bmatrix} Y_p & Y_{j-p} \end{bmatrix} \begin{bmatrix} T^{(H)}_{p\times p} & * \\ 0 & T^{(H)}_{(j-p)\times(j-p)} \end{bmatrix} \begin{bmatrix} Z^*_p \\ Z^*_{j-p} \end{bmatrix}, \tag{50}$$

$$K_j = \begin{bmatrix} Y_p & Y_{j-p} \end{bmatrix} \begin{bmatrix} T^{(K)}_{p\times p} & * \\ 0 & T^{(K)}_{(j-p)\times(j-p)} \end{bmatrix} \begin{bmatrix} Z^*_p \\ Z^*_{j-p} \end{bmatrix}, \tag{51}$$

where H_j and K_j are the $j \times j$ upper Hessenberg matrices obtained by removing the last row of $\underline{H}_j$ and $\underline{K}_j$, respectively, $Y := \begin{bmatrix} Y_p & Y_{j-p} \end{bmatrix}$ and $Z := \begin{bmatrix} Z_p & Z_{j-p} \end{bmatrix} \in \mathbb{C}^{j\times j}$ are unitary matrices with Y_p, $Z_p \in \mathbb{C}^{j\times p}$, Y_{j-p}, $Z_{j-p} \in \mathbb{C}^{j\times(j-p)}$, and $T^{(H)} := \begin{bmatrix} T^{(H)}_{p\times p} & * \\ 0 & T^{(H)}_{(j-p)\times(j-p)} \end{bmatrix}$ and $T^{(K)} := \begin{bmatrix} T^{(K)}_{p\times p} & * \\ 0 & T^{(K)}_{(j-p)\times(j-p)} \end{bmatrix} \in \mathbb{C}^{j\times j}$ are upper (quasi)triangular matrices with $T^{(H)}_{p\times p}$, $T^{(K)}_{p\times p} \in \mathbb{C}^{p\times p}$ and $T^{(H)}_{(j-p)\times(j-p)}$, $T^{(K)}_{(j-p)\times(j-p)} \in \mathbb{C}^{(j-p)\times(j-p)}$. The $p < j$ Ritz values of interest are the eigenvalues of the pencil $T^{(K)}_{p\times p} - \lambda T^{(H)}_{p\times p}$. By multiplying by Z on the right the recurrence relation (49) and using (50) and (51), and considering the first p columns, we obtain

$$\mathcal{A}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\begin{bmatrix} Y_p & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} T^{(H)}_{p\times p} \\ h_{j+1,j}\tilde{z}^* \end{bmatrix} = \mathcal{B}\mathbf{Q}_{j+1}\mathbf{R}_{j+1}\begin{bmatrix} Y_p & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} T^{(K)}_{p\times p} \\ k_{j+1,j}\tilde{z}^* \end{bmatrix}, \tag{52}$$

where $\tilde{z}^*$ represents the first p entries of the last row of Z. By introducing the notation

$$Y_1 := \begin{bmatrix} Y_p & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{C}^{(j+1)\times(p+1)}, \quad \underline{H}^+_p := \begin{bmatrix} T^{(H)}_{p\times p} \\ h_{j+1,j}\tilde{z}^* \end{bmatrix}, \quad \underline{K}^+_p = \begin{bmatrix} T^{(K)}_{p\times p} \\ k_{j+1,j}\tilde{z}^* \end{bmatrix} \in \mathbb{C}^{(p+1)\times p} \tag{53}$$

and defining $\mathbf{W}_{p+1} = \mathbf{R}_{j+1}Y_1$, we obtain

$$\mathcal{A}\mathbf{Q}_{j+1}\mathbf{W}_{p+1}\underline{H}^+_p = \mathcal{B}\mathbf{Q}_{j+1}\mathbf{W}_{p+1}\underline{K}^+_p. \tag{54}$$

Note that with this transformation, we reduce the size of the matrices $\underline{H}^+_p, \underline{K}^+_p$, and $\mathbf{W}_{p+1}$ with respect to $\underline{H}_j, \underline{K}_j$, and $\mathbf{R}_{j+1}$ and remove the Ritz values that are not of interest. However, observe that the large factor $\mathbf{Q}_{j+1}$ remains unchanged. In order to reduce the size of $\mathbf{Q}_{j+1}$, consider

$$\mathbf{W}_{p+1} = \begin{bmatrix} W^{(1)}_{p+1} \\ \vdots \\ W^{(d)}_{p+1} \\ V_{j+1}Y_1 \end{bmatrix}, \quad W^{(i)}_{p+1} \in \mathbb{C}^{r_{j+1}\times(p+1)}, \quad i = 1, \ldots, d,$$

and let ω be the rank of $[W^{(1)}_{p+1} \quad \cdots \quad W^{(d)}_{p+1}]$. The key observation is that although the matrices $\underline{H}^+_p, \underline{K}^+_p$ are no longer in Hessenberg form, the subspace spanned by the columns of $(\mathbf{Q}_{j+1}\mathbf{W}_{p+1})$ is still an RK subspace corresponding to $\mathcal{A}-\lambda\mathcal{B}$.[28] Therefore, we can apply Theorem 6 to span $\left\{ Q_{j+1}W^{(1)}_{p+1}, \ldots, Q_{j+1}W^{(d)}_{p+1} \right\} = Q_{j+1}\,\text{span}\left\{ W^{(1)}_{p+1}, \ldots, W^{(d)}_{p+1} \right\}$ to obtain that $\omega \le d + p < d + j$. Then, we compute the economy singular value decomposition of

$$\begin{bmatrix} W^{(1)}_{p+1} & \cdots & W^{(d)}_{p+1} \end{bmatrix} = \mathcal{U}\mathcal{S}[\mathcal{V}^{(1)} \quad \cdots \quad \mathcal{V}^{(d)}],$$

where $\mathcal{U} \in \mathbb{C}^{r_{j+1}\times\omega}$, $\mathcal{S} \in \mathbb{C}^{\omega\times\omega}$, and $\mathcal{V}^{(i)} \in \mathbb{C}^{\omega\times(p+1)}$ for $i = 1, \ldots, d$. Thus, by defining

$$Q^+_{p+1} = Q_{j+1}\mathcal{U}, \quad \mathbf{R}^+_{p+1} = \begin{bmatrix} \mathcal{S}\mathcal{V}^{(1)} \\ \vdots \\ \mathcal{S}\mathcal{V}^{(d)} \\ V_{j+1}Y_1 \end{bmatrix}, \quad \mathbf{Q}^+_{p+1} = \begin{bmatrix} Q^+_{p+1} & & \\ & \ddots & \\ & & Q^+_{p+1} \\ & & & I_s \end{bmatrix},$$

we get from (54) the CORK recurrence relation

$$\mathcal{A}\mathbf{Q}_{p+1}^{+}\mathbf{R}_{p+1}^{+}\underline{H}_{p}^{+} = \mathcal{B}\mathbf{Q}_{p+1}^{+}\mathbf{R}_{p+1}^{+}\underline{K}_{p}^{+}, \tag{55}$$

with $p < j$. It is important to emphasize again that the matrices $\underline{H}_p^+$ and $\underline{K}_p^+$ are no longer upper Hessenberg matrices; however, they contain the required Ritz values, and the columns of $\mathbf{Q}_{p+1}^{+}\mathbf{R}_{p+1}^{+}$ span a corresponding RK subspace. We continue the process by expanding (55) with Algorithm 6 until we get an RK subspace of dimension j. The matrices $\underline{H}_j^+$ and $\underline{K}_j^+$ obtained in this expansion are not in Hessenberg form, although their columns $p+1, \ldots, j$ have a Hessenberg structure (see the work of Stewart[30(p.329)]). Then, the restarting process described in this section is applied again to get a new compact relation (55) of "size p." This expansion–restarting procedure is cyclicly repeated until the prescribed stopping criterion is satisfied for a certain desired number, less than or equal to p, of Ritz pairs.

5 | NUMERICAL EXPERIMENTS

In this section, we present two large-scale and sparse numerical examples to illustrate the efficiency of the R-CORK method. All reported experiments were performed using MATLAB R2013a on a PC with a 2,2 GHz Intel Core i7 processor, with 16 GB of RAM and DDR3 memory type, and with the operating system macOS Sierra, version 10.12.1.

By following section 8 in the work of Van Beeumen et al.,[20] in the numerical experiments, we plot the residuals at each iteration, with and without restarts, obtained by using the R-CORK method, the dimension of the subspace at each iteration for R-CORK, and the comparison of the memory storage of R-CORK and of the classical RK method applied directly to the linearization (9). We also report on the number of iterations until convergence.

Inspired by the applications in section 4 in the work of Su and Bai,[4] we construct numerical experiments with prescribed eigenvalues and poles of a rational matrix $R(\lambda)$ represented as in (8). In order to measure the convergence of an approximate eigenpair (λ, x) of $R(\lambda)$, we consider the relative norm residual

$$E(\lambda, x) = \frac{\|R(\lambda)x\|_2}{\left(\sum_{i=0}^{d} |\lambda|^i \|P_i\|_F + \|E(C-\lambda D)^{-1}F^T\|_F\right)\|x\|_2}. \tag{56}$$

Observe that the computation of $E(\lambda, x)$ involves matrices and vectors of size n and, hence, is expensive. Therefore, in actual practice, we recommend to test first the convergence through a cheap estimation of the residual of the linearized problem, that is, $\|(\mathcal{A}-\lambda\mathcal{B})\mathbf{z}\|_2$, involving only the small projected problem (15), and once such residual is sufficiently small to compute the residual (56) every $q > 1$ iterations instead of at each iteration. However, in our examples, we performed the computation of $E(\lambda, x)$ at each iteration for the purpose of illustration.

The computation of (56) deserves some comments. Note first that it requires to recover the approximated eigenvector x of $R(\lambda)$ from the approximated eigenvector $\mathbf{z}$ of the linearization $\mathcal{A}-\lambda\mathcal{B}$ in (9) computed in Step 10 of Algorithm 6. This recovery, according to the first equation in (11), can be done by taking any of the first d blocks of $\mathbf{z}$ if $\lambda \neq 0$. Since in our numerical examples the moduli of the approximate eigenvalues are larger than 1, we have chosen the dth block of $\mathbf{z}$ as approximate x. However, we recommend choosing the first block if the moduli of the approximate eigenvalues are smaller than 1. The calculation of the quantities $\|P_i\|_F, i = 0, \ldots, d$, needs to be performed only once, and it is inexpensive since the matrices P_i are sparse in practice. Finally, to compute the expression $\|E(C-\lambda D)^{-1}F^T\|_F$ on the denominator in (56), we use

$$\begin{aligned}\|E(C-\lambda D)^{-1}F^T\|_F^2 &= \operatorname{trace}\left(\left(E(C-\lambda D)^{-1}F^T\right)^* E(C-\lambda D)^{-1}F^T\right),\\ &= \operatorname{trace}\left((E^*E)(C-\lambda D)^{-1}(F^T\bar{F})(C-\lambda D)^{-*}\right),\end{aligned}$$

which only involves the matrices E^*E, $F^T\bar{F}$, $(C-\lambda D)^{-1}$, and $(C-\lambda D)^{-*}$ of size $s \times s$. Since in many applications, $s \ll n$, this computation is usually inexpensive.

Numerical experiment 1. We construct an REP of the type arising from the free vibrations of a structure if one uses a viscoelastic constitutive relation to describe the behavior of a material.[4,8] The REPs of this type have the following structure:

$$R(\lambda)x = \left(\lambda^2 M + K - \sum_{i=1}^{k} \frac{1}{1+b_i\lambda}\Delta G_i\right)x = 0, \tag{57}$$

where the mass and stiffness matrices M and K are real symmetric and positive definite, b_j are relaxation parameters over k regions, and ΔG_j is an assemblage of element stiffness matrices over the region with the distinct relaxation parameters.

As in the work of Su and Bai,[4] we consider the case where $\Delta G_i = E_i E_i^T$ and $E_i \in \mathbb{R}^{n\times s_i}$. By defining

$$E = [E_1, E_2, \ldots, E_k], \quad D = \operatorname{diag}(b_1 I_{s_1}, b_2 I_{s_2}, \ldots, b_k I_{s_k}),$$

the REP (57) can be written in the form (8), as follows:

$$\left(\lambda^2 M + K - E(I + \lambda D)^{-1} E^T\right) x = 0.$$

In our particular example, we consider the case with one region and one relaxation parameter $b_1 = -1$. The construction of the matrices M and K in our example proceeds as follows: Construct first $R_1(\lambda) = \lambda^2 A_2 + A_0 - e_{10000}(1-\lambda)^{-1}(e_{10000})^T$, with $A_2, A_0 \in \mathbb{R}^{10000\times 10000}$ diagonal and positive definite matrices and e_{10000} the last column of I_{10000}. This structure allows to prescribe easily the eigenvalues for $R_1(\lambda)$. Then, we consider the following invertible tridiagonal matrix P:

$$P = \begin{bmatrix} 1 & \frac{1}{2} & & \\ \frac{1}{3} & 1 & \ddots & \\ & \ddots & \ddots & \frac{1}{2} \\ & & \frac{1}{3} & 1 \end{bmatrix},$$

and, finally, construct $R(\lambda) = PR_1(\lambda)P^T$. Since P is invertible, the eigenvalues of $R(\lambda)$ and $R_1(\lambda)$ are the same. By using this procedure, we have constructed the REP

$$R(\lambda)x = \left(\lambda^2 M + K - p_{10000}(1-\lambda)^{-1}(p_{10000})^T\right) x = 0, \tag{58}$$

where $M := PA_2P^T$, $K := PA_0P^T \in \mathbb{R}^{10000\times 10000}$ are symmetric, positive definite, and pentadiagonal matrices, and $p_{10000} \in \mathbb{R}^{10000}$ represents the last column of matrix P.

In this example, we are interested in computing the 20 eigenvalues of (58) with a negative imaginary part and with the largest absolute value of the negative imaginary part. To aim our goal, we use three cyclically repeated shifts in the RK steps and a random unit real vector as an initial vector. The reader can see the approximate eigenvalues computed by R-CORK and the chosen shifts in Figure 1a. We first solve the REP (58) by using Algorithm 6 without restart, and after 85 iterations, we find the required eigenvalues with a tolerance (56) of 10^{-10}. The convergence history is shown in Figure 1b. In Figure 1d, we plot r_j, the rank of Q_j at iteration j, and j, the dimension of the Krylov subspace. Since we did not perform restart, we can see that both r_j and j increase with the iteration count j and that $r_j = j + 1$, as expected since the degree of the polynomial part of (58) is $d = 2$. Figure 1f displays the comparison between the cost of memory storage of both the R-CORK method, by using Algorithm 6, and the classical RK method, by using Algorithm 1. From this Figure, we can see that the R-CORK method requires approximately half of the memory storage of the classical RK method, which is consistent with the degree 2 of the polynomial part of (58).

Next, we apply Algorithm 6 to the REP (58) combined with the implicit restarting introduced in Section 4. We choose the maximum dimension of the subspace $m = 45$, which is reduced after each restart to dimension $p = 30$ to compute the 20 required eigenvalues. The convergence history of the eigenpairs (λ, x) computed by this restarted R-CORK method is shown in Figure 1c. After 3 restarts and 81 iterations, the 20 required eigenvalues have been found with a tolerance (56) of 10^{-10}. In Figure 1e, the reader can see the rank of Q_j at the jth iteration, and it can be seen that with restart, the relation between j and r_j continues the same. Finally, in Figure 1g, we plot the memory storage for R-CORK and classical RK, and it can be observed that the memory cost for the R-CORK method is a factor close to 2 smaller than the memory cost obtained by the classical RK method.

Numerical experiment 2. For this numerical example, we consider an academic REP of size 5000×5000 and with the degree of its polynomial part equal to 3, that is, an REP of the form

$$R(\lambda) = \lambda^3 A_3 + \lambda^2 A_2 + \lambda A_1 + A_0 - E(C - \lambda D)^{-1} F^T. \tag{59}$$

The coefficient matrices of $R(\lambda)$ in (59) were constructed in a similar way as in numerical experiment 1: First, we consider a rational matrix $R_2(\lambda) = \lambda^3 P_3 + \lambda^2 P_2 + \lambda P_1 + P_0 - E_0(C - \lambda D)^{-1} F_0^T$ with the prescribed eigenvalues, where $P_i \in \mathbb{R}^{5000\times 5000}$ are diagonal matrices, $E_0 = [e_1 + e_2, \quad e_5 + e_6]$, $F_0 = [e_{4997} + e_{4998}, \quad e_{4999} + e_{5000}] \in \mathbb{R}^{5000\times 2}$, with e_i as the ith canonical vector of size 5000, and

$$C = \begin{bmatrix} 105 & 0 \\ 0 & -105 \end{bmatrix}, \quad D = I_2,$$

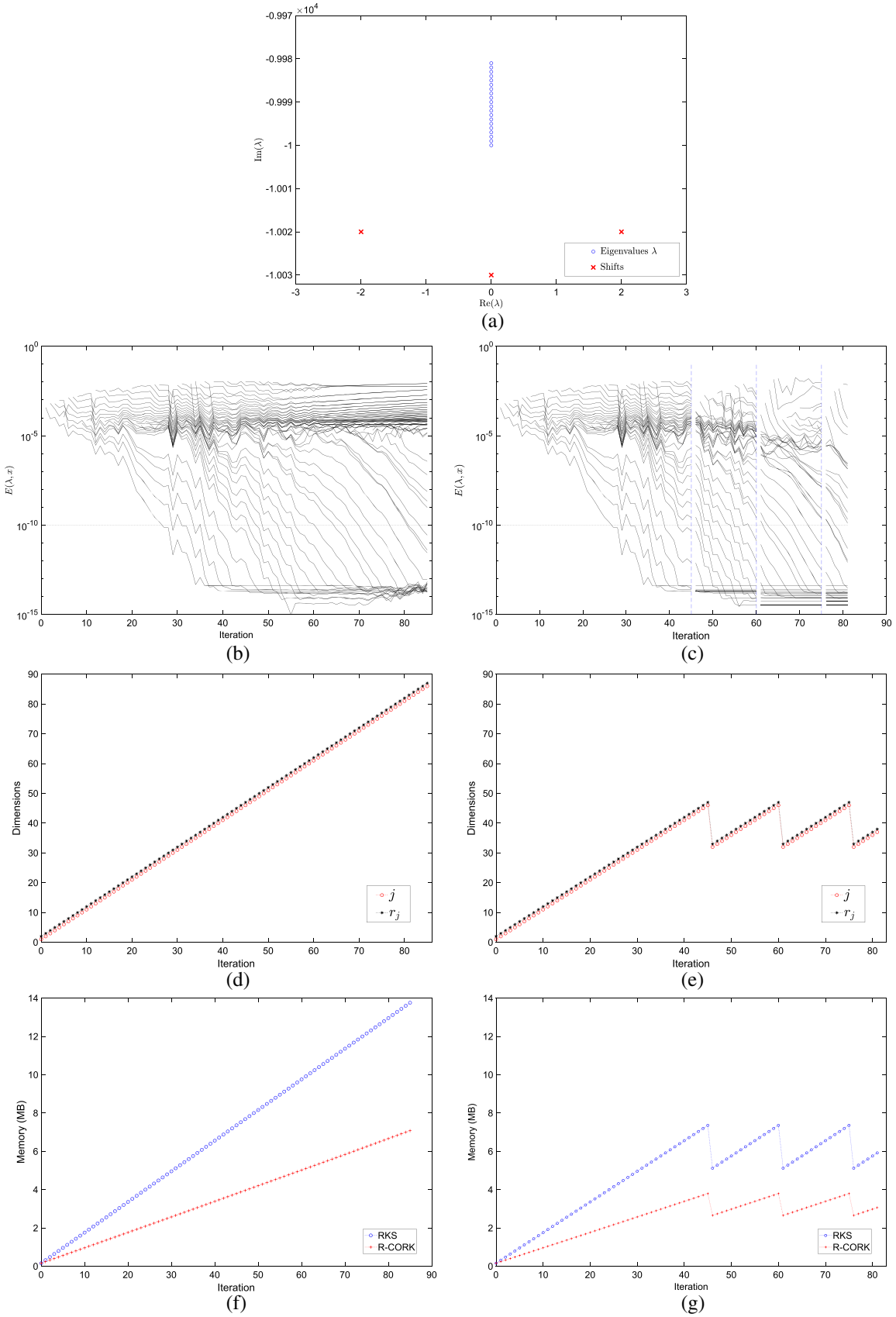

FIGURE 1 Numerical experiment 5.1. (a) Eigenvalues and shifts. (b) Convergence history without restart. (c) Convergence history with restart. (d) Dimension of the subspace without restart. (e) Dimension of the subspace with restart. (f) Memory cost without restart. (g) Memory cost with restart

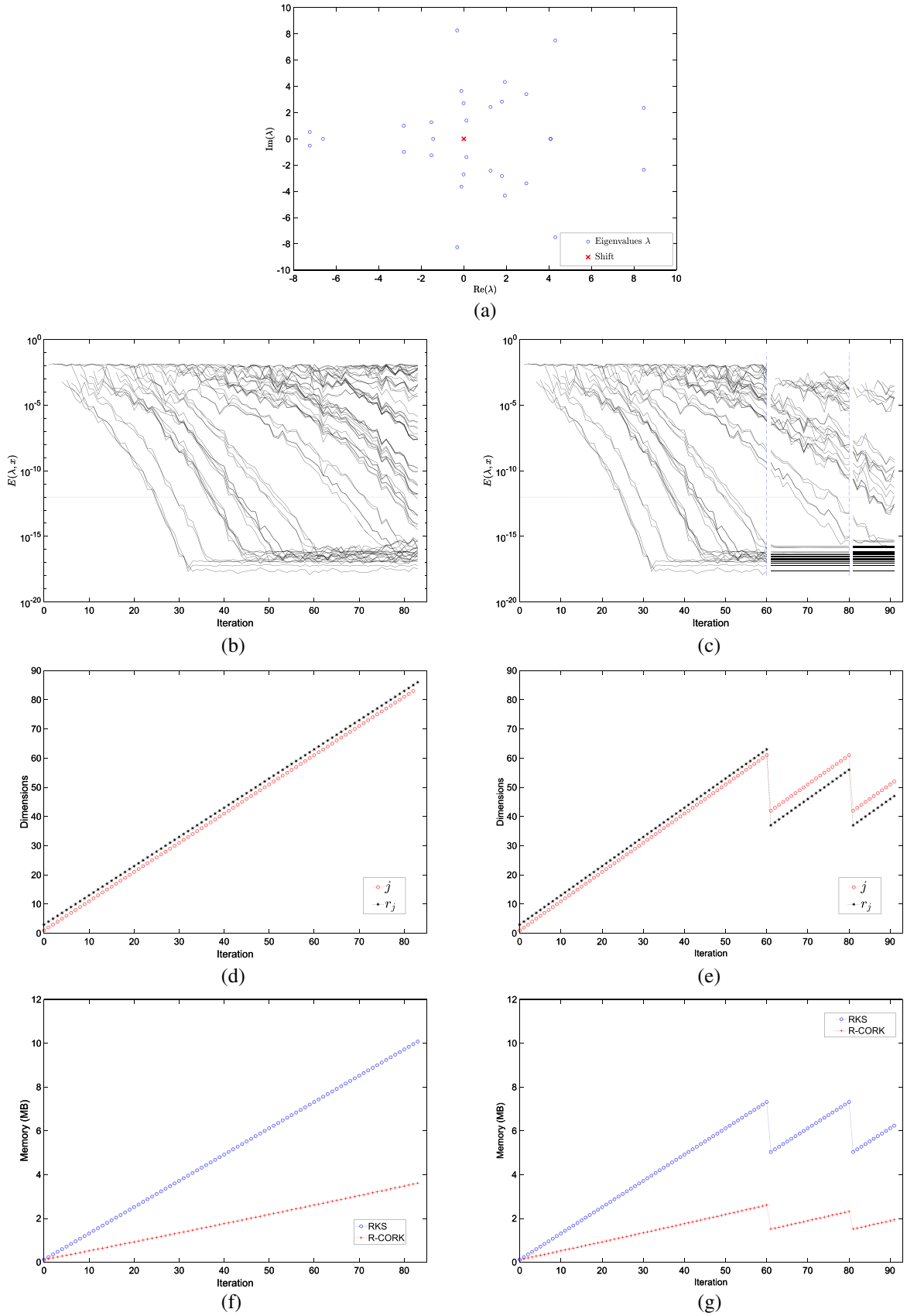

FIGURE 2 Numerical experiment 5.2. (a) Eigenvalues and shifts. (b) Convergence history without restart. (c) Convergence history with restart. (d) Dimension of the subspace without restart. (e) Dimension of the subspace with restart. (f) Memory cost without restart. (g) Memory cost with restart

and then, we define $R(\lambda) = PR_2(\lambda)Q$, where

$$P = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & & \\ -\frac{1}{4} & \ddots & \ddots & \ddots & \\ -\frac{1}{5} & \ddots & \ddots & \ddots & \frac{1}{3} \\ & \ddots & \ddots & \ddots & \frac{1}{2} \\ & & -\frac{1}{5} & -\frac{1}{4} & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} -1 & -\frac{1}{3} & & \\ \frac{1}{2} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{1}{3} \\ & & \frac{1}{2} & -1 \end{bmatrix} \in \mathbb{R}^{5000\times 5000}.$$

The goal of this example is to compute the 30 eigenvalues closest to zero. In this situation, it is natural to choose the origin as a fixed shift. In Figure 2a, the approximate eigenvalues computed by R-CORK are displayed. By starting with a random unit complex vector, first, we apply R-CORK without restarting, and after 83 iterations, the desired eigenvalues are obtained with a tolerance (56) of 10^{-12}. The convergence history can be seen in Figure 2b. In Figure 2d, we see that the relation $r_j < j + d$, with j as the number of iterations, also holds in this example, although, in this case, with $d = 3$ since this is the degree of the polynomial part in (59). Figure 2f shows the memory costs of R-CORK and classical RK. It is observed that the reduction in the cost of R-CORK is approximately a factor of 3, that is, the degree of the polynomial part of (59).

As a final example, we solve (59) by using R-CORK combined with restarting and taking a maximum subspace dimension $m = 60$, which is reduced to $p = 40$ after every restart. The convergence history is shown in Figure 2c, where it is observed that after 91 iterations and 2 restarts, the 30 eigenvalues closest to the origin have been found with a tolerance (56) of 10^{-12}. Despite the fact that a few more iterations are needed with restart than without restart, we see in Figure 2e that we are using a subspace of much smaller dimension to compute the eigenpairs and, particularly for this example, $r_j < j$ after the restart. Finally, the comparison of the memory costs for the R-CORK and classical RK methods is plotted in Figure 2g, where we see again that the cost of R-CORK is a factor $d = 3$ smaller.

6 | CONCLUSIONS AND LINES OF FUTURE RESEARCH

In this paper, we have introduced the R-CORK method for solving large-scale REPs that are represented as the sum of their polynomial and strictly proper parts as in (8). The first key idea is that R-CORK solves the GEP associated to the Frobenius companion-like linearization (9) previously introduced in the work of Su and Bai.[4] The second key idea is that R-CORK is a structured version of the classical RK method for solving GEPs that takes advantage of the particular structure of (9). This structure allows us to represent the orthonormal bases of the RK subspaces of (9) in a compact form, involving less parameters than the bases of RK subspaces of the same dimension corresponding to unstructured GEPs of the same size as the considered linearization. In addition, this compact form can be efficiently and stably updated in each RK iteration by the use of two levels of orthogonalization in the spirit of the TOAR[17,19] and CORK[20] methods for large-scale PEPs.

The combined use of the compact representation of RK subspaces and the two levels of orthogonalization in R-CORK reduces significantly the orthogonalization and the memory costs with respect to a direct application of the classical RK method to the linearization (9). If $n \times n$ is the size of the REP, j is the maximum dimension of the considered Krylov subspaces of the linearization, d is the degree of the matrix polynomial $P(\lambda)$ in (8), and $s \times s$ is the size of the pencil $(C - \lambda D)$ appearing in (8) (note that if $s \leq n$, then s is essentially the rank of the strictly proper part of the rational matrix $R(\lambda)$), then the reduction in costs of R-CORK is appreciable whenever $jd \ll n$ and very considerable if, moreover, $s \ll n$ and $d < j$. In this situation, after j iterations, the orthogonalization cost of R-CORK is $\mathcal{O}(j^2 n)$, whereas that of classical RK is $\mathcal{O}(j^2 nd)$; the memory cost of R-CORK is approximately nj numbers, whereas that of classical RK is ndj. These reductions can be combined with an structured implementation of a Krylov–Schur implicit restarting adapted to the compact representation used by R-CORK, which allows us to keep the dimension of the Krylov subspaces moderate without essentially increasing the number of iterations until convergence. The performed numerical experiments confirm all these good properties of R-CORK.

Since many linearizations of rational matrices different from (9) have been developed very recently[3,26] and some of them include the option of considering that the matrix polynomial $P(\lambda)$ in (8) is expressed in nonmonomial bases, an interesting

line of future research on the numerical solution of REPs is to investigate the potential extension of the R-CORK strategy to other linearizations.

ACKNOWLEDGEMENTS

This work was partially supported by "Ministerio de Ciencia, Innovación y Universidades of Spain" and "Fondo Europeo de Desarrollo Regional (FEDER) of EU" under Grants MTM2012-32542, MTM2015-65798-P, MTM2017-90682-REDT, and MTM2015-68805-REDT. The research of Javier González-Pizarro was funded by "Comisión Nacional de Investigación Científica y Tecnológica (CONICYT) de Chile" under Grant BCH 72160331. The authors sincerely thank Roel Van Beeumen and Karl Meerbergen for answering patiently and very carefully many questions on the CORK method they developed.[20] Their help has been very important for improving some parts of this paper.

ORCID


Froilán M. Dopico http://orcid.org/0000-0001-7779-8960

REFERENCES

1. Kailath T. Linear systems. Englewood Cliffs, NJ: Prentice-Hall; 1980.
2. Rosenbrock HH. State-space and multivariable theory. London, UK: Thomas Nelson & Sons; 1970.
3. Amparan A, Dopico FM, Marcaida S, Zaballa I. Strong linearizations of rational matrices. Manchester, UK: Manchester Institute for Mathematical Sciences, The University of Manchester; 2016. MIMS EPrint 2016.51.
4. Su Y, Bai Z. Solving rational eigenvalue problems via linearization. SIAM J Matrix Anal Appl. 2011;32(1):201–216.
5. Voss H. A rational spectral problem in fluid–solid vibration. Electron Trans Numer Anal. 2003;16:94–106.
6. Mackey DS, Mackey N, Mehl C, Mehrmann V. Structured polynomial eigenvalue problems: good vibrations from good linearizations. SIAM J Matrix Anal Appl. 2006;28:1029–1051.
7. Solov'ëv S. Preconditioned iterative methods for a class of nonlinear eigenvalue problems. Linear Algebra Appl. 2006;415:210–229.
8. Mehrmann V, Voss H. Nonlinear eigenvalue problems: a challenge for modern eigenvalue methods. GAMM-Reports. 2004;27:121–152.
9. Mohammadi SA, Voss H. Variational characterization of real eigenvalues in linear viscoelastic oscillators. Math Mech Solids. 2017. Prepublished.
10. Hwang T-M, Lin W-W, Wang W-C, Wang W. Numerical simulation of three dimensional pyramid quantum dot. J Comput Phys. 2004;196:208–232.
11. Voss H. Iterative projection methods for computing relevant energy states of a quantum dot. BIT Numer Math. 2004;44:387–401.
12. De Terán F, Dopico FM, Mackey DS. Fiedler companion linearizations and the recovery of minimal indices. SIAM J Matrix Anal Appl. 2010;31:2181–2204.
13. Mackey DS, Mackey N, Mehl C, Mehrmann V. Vector spaces of linearizations for matrix polynomials. SIAM J Matrix Anal Appl. 2006;28:971–1004.
14. Golub GH, Van Loan CF. Matrix computations. 4th ed. Baltimore, MD: Johns Hopkins University Press; 2013.
15. Bai Z, Su Y. A second-order Arnoldi method for the solution of the quadratic eigenvalue problem. SIAM J Matrix Anal Appl. 2005;26:640–659.
16. Meerbergen K. The quadratic Arnoldi method for the solution of the quadratic eigenvalue problem. SIAM J Matrix Anal Appl. 2008;30(4):1463–1482.
17. Su Y, Zhang J, Bai Z. A compact Arnoldi algorithm for polynomial eigenvalue problems. Paper presented at: Recent Advances in Numerical Methods for Eigenvalue Problems (RANMEP2008); 2008 Jan 4–8; Hsinchu, Taiwan.
18. Lu D, Su Y, Bai Z. Stability analysis of the two-level orthogonal Arnoldi procedure. SIAM J Matrix Anal Appl. 2016;37:195–214.
19. Kressner D, Roman JE. Memory-efficient Arnoldi algorithms for linearizations of matrix polynomials in Chebyshev basis. Numer Linear Algebra Appl. 2014;21(4):569–588.
20. Van Beeumen R, Meerbergen K, Michiels W. Compact rational Krylov methods for nonlinear eigenvalue problems. SIAM J Matrix Anal Appl. 2015;36(2):820–838.
21. Amiraslani A, Corless RM, Lancaster P. Linearization of matrix polynomials expressed in polynomial bases. IMA J Numer Anal. 2009;29:141–157.
22. Ruhe A. Rational Krylov sequence methods for eigenvalue computation. Linear Algebra Appl. 1984;58:391–405.
23. Ruhe A. Rational Krylov: a practical algorithm for large sparse nonsymmetric matrix pencils. SIAM J Sci Comput. 1998;19:1535–1551.
24. Horn R, Johnson C. Matrix analysis. 2nd ed. Cambridge, UK: Cambridge University Press; 2013.
25. Gohberg I, Lancaster P, Rodman L. Matrix polynomials. New York, NY: Academic Press; 1982.
26. Alam R, Behera N. Linearizations for rational matrix functions and Rosenbrock system polynomials. SIAM J Matrix Anal Appl. 2016;37(1):354–380.

27. Betcke T, Higham NJ, Mehrmann V, Schröder C, Tisseur F. NLEVP: a collection of nonlinear eigenvalue problems. ACM Trans Math Software. 2013;39(2):28. Article No. 7.
28. De Samblanx G, Meerbergen K, Bultheel A. The implicit application of a rational filter in the RKS method. BIT Numer Math. 1997;37(4):925–947.
29. Stewart GW. A Krylov–Schur algorithm for large eigenproblems. SIAM J Matrix Anal Appl. 2001;23(3):601–614.
30. Stewart GW. Matrix algorithms. Volume II: Eigensystems. Philadelphia, PA: Society for Industrial and Applied Mathematics; 2001.

How to cite this article: Dopico FM, González-Pizarro J. A compact rational Krylov method for large-scale rational eigenvalue problems. *Numer Linear Algebra Appl*. 2019;26:e2214. https://doi.org/10.1002/nla.2214