Received: 28 March 2019 | Revised: 10 January 2020 | Accepted: 30 January 2020

DOI: 10.1002/nla.2293

RESEARCH ARTICLE

Inertia-based spectrum slicing for symmetric quadratic eigenvalue problems

Carmen Campos | **Jose E. Roman**

D. Sistemes Informàtics i Computació, Universitat Politècnica de València, Camí de Vera s/n, València, Spain

Correspondence
Jose E. Roman, D. Sistemes Informàtics i Computació, Universitat Politècnica de València, Camí de Vera s/n, València, Spain.
Email: jroman@dsic.upv.es

Funding information
Agencia Estatal de Investigación, Grant/Award Number: TIN2016-75985-P

Summary
In the quadratic eigenvalue problem (QEP) with all coefficient matrices symmetric, there can be complex eigenvalues. However, some applications need to compute real eigenvalues only. We propose a Lanczos-based method for computing all real eigenvalues contained in a given interval of large-scale symmetric QEPs. The method uses matrix inertias of the quadratic polynomial evaluated at different shift values. In this way, for hyperbolic problems, it is possible to make sure that all eigenvalues in the interval have been computed. We also discuss the general nonhyperbolic case. Our implementation is memory-efficient by representing the computed pseudo-Lanczos basis in a compact tensor product representation. We show results of computational experiments with a parallel implementation in the SLEPc library.

KEYWORDS
inertia, parallel computing, pseudo-Lanczos, quadratic eigenvalue problem, SLEPc, symmetric linearization

1 | INTRODUCTION

The quadratic eigenvalue problem (QEP) arises in many scientific computing applications. One such application is the analysis of damped oscillations of linear systems,[1,2] such as a spring-mass system, or systems that can be modeled by analogous equations, for example, electrical networks. The discretization of the governing equation leads to the eigenproblem

$$(\lambda^2 M + \lambda C + K)x = 0, \tag{1}$$

where λ is the eigenvalue and $x \neq 0$ is the eigenvector. In this article, we focus on the case where the coefficient matrices M, C and K are real symmetric (although the approach is also valid for Hermitian matrices). They are sometimes called mass, damping and stiffness matrices, respectively. Typically, M and K are positive definite.

In cases where damping effects can be neglected, the problem would be solved as a linear eigenproblem

$$Ax = \mu Bx. \tag{2}$$

This work was supported by the Spanish Agencia Estatal de Investigación under project SLEPc-HS (TIN2016-75985-P), which includes European Commission ERDF funds.

If the pencil (A, B) is symmetric-definite then all eigenvalues μ_i are real and if B is positive definite the eigenvectors x_i are mutually B-orthogonal. For large-scale problems where the full spectrum is not required, Lanczos-type methods can be used to obtain a few eigenpairs (x_i, μ_i). If the wanted eigenvalues lie in the interior of the spectrum, then one possible strategy is to combine Lanczos with the shift-and-invert spectral transformation, that operates with $(A - \sigma B)^{-1}B$ to compute eigenvalues closest to the shift σ. Furthermore, it is often necessary to compute all eigenvalues contained in a prescribed interval of the real line. The latter scenario is the main focus of this article. By performing several shift-and-invert Lanczos runs at different shift locations[3,4] it is possible to effectively sweep the interval and compute all eigenvalues efficiently. This type of strategy, often called spectrum slicing, relies on the matrix inertia of $(A - \sigma B)$ obtained at each shift σ, which provides the number of negative eigenvalues for this matrix and informs about the number of eigenvalues of Equation (2) lying on the left of that particular shift value, so that a bisection scheme can guarantee that no eigenvalues have been missed. Note that alternative spectrum slicing mechanisms that do not need inertia are being developed in recent times,[5] but we do not consider this kind of method here.

Our goal is to extend the inertia-based spectrum slicing scheme to the QEP Equation (1). A common way to solve the QEP is to linearize it first to obtain an equivalent linear eigenproblem Equation (2), then use a linear eigensolver to compute the wanted solutions, and finally extract the QEP eigenvectors from the eigenvectors of the linearization. In the case that all three coefficient matrices of the QEP are symmetric, it is convenient to use a symmetric linearization, where the resulting A and B matrices of the linearization are also symmetric. However, the pencil (A, B) is either indefinite, or it is definite but requires to compute a certain transformation so that B is positive definite.[6,7] In this scenario, the question is how to use the information of the matrix inertia of $A - \sigma B$ to determine the localization of eigenvalues.

The topic of inertia computation in indefinite pencils has been treated by Nakatsukasa and Noferini,[8] who show that it is not possible to get the exact eigenvalue count in this general case. Our problem is a particular case since the pencil comes from the linearization of a symmetric quadratic matrix polynomial. We will see below that for some problem types inertia still gives a complete information about the location of eigenvalues in this case.

In addition to computing the correct eigenvalue count inside an interval, another difficulty is that a Krylov solver that operates on a symmetric-indefinite pencil and wants to exploit symmetry has to rely on the pseudo-Lanczos process.[9] Furthermore, the linearization duplicates the size of the matrices, that is, A and B have $2n$ rows and columns, where n is the dimension of the matrices in Equation (1). A naive implementation of the solver will be inefficient in terms of computational and storage cost, because it has to operate with Lanczos vectors of length $2n$, and also especially because a matrix of order $2n$ must be factorized in the shift-and-invert technique. In a previous work,[10] we showed that it is possible to make an efficient implementation of the pseudo-Lanczos recurrence in the case of linearization of a symmetric QEP, and do the shift-and-invert solves implicitly by only requiring the factorization of $Q(\sigma)$, where Q is the matrix polynomial associated with the QEP,

$$Q(\lambda) = \lambda^2 M + \lambda C + K. \tag{3}$$

Moreover, we presented a memory-efficient variant, called STOAR,[10] that uses a compact representation of the Krylov basis and hence stores and operates only with vectors of length n.

In this article, we aim at extending STOAR with the capability to retrieve all eigenvalues contained in an interval, with inertia-based spectrum slicing. Since we are factorizing $Q(\sigma)$, rather than the linearization, we must consider how to perform bisection by using inertia information obtained from $Q(\sigma)$. This will be feasible in the particular case of hyperbolic QEPs, where all eigenvalues are real,[11] but also in some nonhyperbolic problems as shown by Li and Cai.[12] This latter article proposes a classical bisection algorithm to compute all real eigenvalues contained in a interval, using the inertia count. The advantage of our method, compared with Li and Cai's, is that the number of required factorizations is much smaller, as it is a subspace-based algorithm. Furthermore, our method computes both eigenvalues and eigenvectors, whereas Li and Cai return eigenvalues only, so one step of inverse iteration would be necessary for each wanted eigenvector.

The main contribution of this article is to show how all these ingredients must be combined to get a robust and efficient spectrum slicing solver for hyperbolic QEPs that can also be used to solve general symmetric QEPs under certain conditions in the selected interval. We also show results obtained with a practical implementation in SLEPc,[13,14] where we have combined our previous STOAR implementation[10] with the spectrum slicing solver for the linear case.[4] In particular, we have to consider practical issues such as how to perform deflation against basis vectors stored in a compact representation, which is required to avoid reconvergence of eigenvalues in a neighboring subinterval, or how to lock converged eigenvalues when doing the thick-restart step.

We conclude this section by mentioning that QEPs can be solved with alternative methods that are not based on linearization. Among these methods, we can cite a fixed-point iteration for symmetric QEPs,[15] Jacobi-Davidson,[16] SOAR,[17] as well as methods for the general nonlinear eigenvalue problem[18] such as Newton-trace iteration or residual inverse iteration.[19] However, only the first of these methods exploits the symmetry of the matrices, and none of them are specific for computing eigenvalues contained in an interval.

The article is organized as follows. Preliminaries are described in Section 2, including the description of inertia laws for linear and quadratic eigenproblems, as well as the restarted memory-efficient pseudo-Lanczos solver. The details of the spectrum slicing scheme for symmetric QEPs is presented in Section 3, including aspects related to parallel implementation. The performance and accuracy of the solver is assessed in Section 4, and the conclusions follow in Section 5.

2 | PREVIOUS WORK

In this section, we describe related works which include the main ingredients used in the spectrum slicing method proposed in this work.

2.1 | Spectrum slicing in linear eigenvalue problems

The technique of inertia-based spectrum slicing was developed in the 1980s, reaching its full potential in the article by Grimes et al.[3] Some years ago we implemented this technique in SLEPc, modernizing the algorithm by including features such as restart and parallel computing.[4] This implementation is being used to solve very challenging problems in computational chemistry.[20] We next give a brief overview of the method emphasizing three relevant elements: B-Lanczos, inertia and deflation. In this section, we assume a symmetric linear eigenproblem Equation (2) with positive definite B.

2.1.1 | Solver preserving the symmetric structure

The spectrum slicing technique uses the symmetric Lanczos method and the shift-and-invert transformation to obtain interior eigenvalues. It computes the largest magnitude eigenvalues of

$$Sx = \theta x, \quad S = (A - \sigma B)^{-1} B, \tag{4}$$

where the obtained eigenvalues $\theta = (\mu - \sigma)^{-1}$ correspond to eigenvalues μ closest to the target $\sigma \in \mathbb{R}$. The transformed matrix S is not symmetric but it is self-adjoint with respect to the inner product defined by B, $\langle u, v\rangle_B := u^* B v$. Hence, by using this nonstandard inner product, the Lanczos method can still be used to generate a Lanczos decomposition

$$SV_j = V_j T_j + \beta_j v_{j+1} e_j^*, \tag{5}$$

where the columns of V_j form a B-orthonormal basis of the Krylov subspace spanned by S and an initial vector v_1, $V_j^* B V_j = I$, and T_j is a real symmetric tridiagonal matrix.

The Krylov subspace contains increasingly accurate Ritz approximations $(\tilde{x}_i, \tilde{\theta}_i)$ of eigenpairs of Equation (4), where $T_j y_i = \tilde{\theta}_i y_i$ and $\tilde{x}_i = V_j y_i$. A cheap convergence criterion for the obtained approximations is $\beta_j |y_{ji}| \leq \varepsilon |\tilde{\theta}_i|$ for a given tolerance ε, which involves a bound for the B-norm of the residual

$$r_i = S\tilde{x}_i - \tilde{\theta}_i \tilde{x}_i. \tag{6}$$

In a spectrum slicing strategy we compute all eigenvalues in a given interval by chunks, using the shift-and-invert transformation at a sequence of shifts, σ_k, $k = 1, 2, \ldots$, and validating the number of eigenvalues obtained using the information supplied by the inertia of the matrices $(A - \sigma_k B)$.

2.1.2 | Counting eigenvalues using matrix inertias

We denote as $\nu(M)$ the number of negative eigenvalues of a symmetric matrix M, and we refer to it as the inertia of M. In virtue of Sylvester's inertia law,[21] this value remains unchanged under congruence transformations. In generalized linear eigenproblems Equation (2) with B positive definite, Grimes et al[3] consider the Cholesky decomposition $B = L_B L_B^*$ and the reduction to standard form $L_B^{-1} A L_B^{-*} x = \lambda x$ to see that $\nu(A - \sigma B) = \nu(L_B^{-1} A L_B^{-*} - \sigma I)$ gives the number of negative eigenvalues for the shifted eigenproblem. Thus, in this case the number of eigenvalues of $A - \lambda B$ to the left of a shift σ can be obtained as the inertia of the matrix $(A - \sigma B)$.

In the context of a Lanczos iteration for computing eigenvalues closest to a target value, to expand the Krylov subspace associated to S of Equation (4), we use an indefinite (block-)triangular factorization $A - \sigma_k B = L_k D_k L_k^*$ where the matrix D_k is block-diagonal with 1×1 or 2×2 diagonal blocks. As a byproduct of this factorization the number of eigenvalues on the left of σ_k can be obtained, $\nu_k := \nu(A - \sigma_k B) = \nu(D_k)$. In this way, running Lanczos on two different shifts $\sigma_k < \sigma_{k+1}$, we know that the number of eigenvalues, counted with their multiplicities, in the interval $[\sigma_k, \sigma_{k+1})$ is $\nu_{k+1} - \nu_k$.

2.1.3 | Avoiding reconvergence via deflation

Computing eigenvalues from different Lanczos runs may entail some difficulties that spectrum slicing has to address. One of them is ascertaining whether equal (in machine precision) values computed from different shifts are spurious duplicates or genuine multiples of a particular eigenvalue. Another problem faced by spectrum slicing is to ensure B-orthogonality between eigenvectors associated to a multiple eigenvalue (or a cluster of eigenvalues) when calculated from more than one shift. The tool used for solving these problems is deflation, which avoids reconvergence by forcing the Lanczos method to work in the orthogonal complement of a set of selected vectors. Since eigenvectors of Equation (2) are mutually B-orthogonal, in the context of the B-Lanczos method, such vectors can be easily deflated incorporating them directly in the Lanczos basis against which B-orthogonalization is performed. This fact makes the B-Lanczos procedure a suitable Krylov method when considering the deflation of eigenvector sets.

Although selected deflation reduces the possibility of recalculating eigenvalues already obtained, it also involves an additional cost since at each iteration of the Lanczos method one vector has to be orthogonalized against all vectors selected for deflation. Therefore, the amount of deflation to be applied for each shift must be adjusted. The strategy used in our slicing technique for linear eigenproblems[4] selects for deflation the set of computed eigenvectors associated to eigenvalues inside the smallest subinterval $[\sigma_\ell, \sigma_r]$ containing the current shift σ and having endpoints at the set of previous shifts. This scheme accepts any value computed from σ in $[\sigma_\ell, \sigma_r]$, where reconvergence is not possible, and discards any eigenvalue computed outside this interval.

Figure 1 depicts the spectrum slicing process for three consecutive shifts. First, it shows a *trust subinterval* (between a and σ_{k-1}) with no missing eigenvalues, obtained after $k-1$ shifts have been processed; second, it shows the subinterval selected for deflation at the kth Lanczos run, and the newly computed eigenvalues at σ_k; and third, it illustrates the deflation subinterval when backtracking to search for missing eigenvalues, and the extension of the trust subinterval (between a and σ_k) once the number of computed eigenvalues matches the inertia count.

2.2 | Symmetric quadratic eigenproblems

We now turn our attention to the QEP Equation (1), where we assume that all matrices are symmetric and M is nonsingular. These problems have special spectral properties, such as, for example, that eigenvalues λ are real or come in complex conjugate pairs λ and $\bar{\lambda}$, even if the matrices defining the QEP are complex Hermitian. In addition, in some cases we can ensure that all the eigenvalues are real, as discussed below. Under some assumptions,[22] real eigenvalues of symmetric QEPs present a variational characterization allowing results to be exploited in a spectrum slicing method relying on Sylvester's law of inertia, similar to those mentioned in Section 2.1.

A standard way of solving the QEP is via linearization, that is, via a linear eigenvalue problem whose eigenvalues coincide with those of the QEP. For the case of a symmetric QEP, the pencil

$$L(\lambda) = A - \lambda B, \quad \text{with} \quad A = \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix}, \quad B = \begin{bmatrix} -C & -M \\ -M & 0 \end{bmatrix}, \tag{7}$$

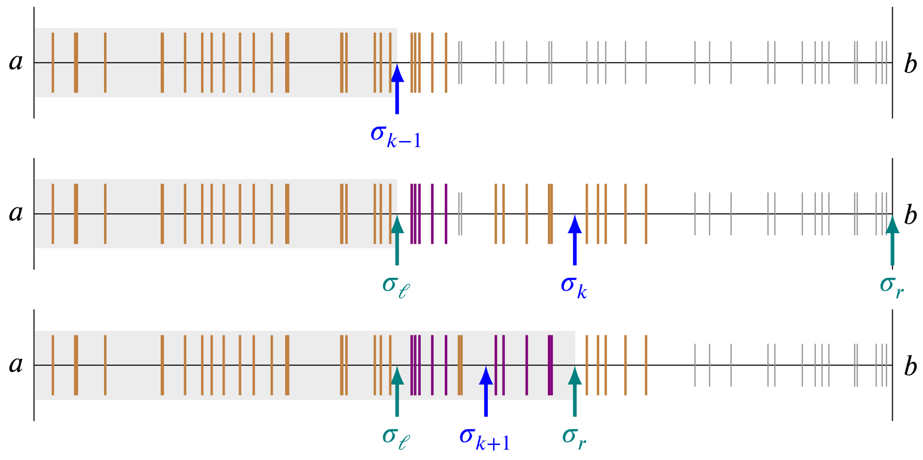

FIGURE 1 Three steps of a sample spectrum slicing computation, corresponding to three consecutive shift values σ_{k-1}, σ_k and σ_{k+1}. The picture illustrates a situation of backtracking, because σ_{k+1} is placed on the left of σ_k while normal movement is to the right. Vertical lines represent eigenvalues, with taller and thicker strokes for already computed eigenvalues (violet color for those being used for deflation) and smaller strokes for eigenvalues that have not been found by the algorithm yet. The shaded area is the trust subinterval, where computed inertia matches the number of eigenvalues found. The interval $[\sigma_\ell, \sigma_r]$ is used to choose eigenvectors for deflation

is a linearization that preserves symmetry. The eigenvectors of $L(\lambda)y = 0$ have the form $y = \begin{bmatrix} x \\ \lambda x \end{bmatrix}$, where (x, λ) are the eigenpairs of $Q(\lambda)x = 0$.

For our purpose, we will consider the general case of symmetric QEPs as well as the special subclass of hyperbolic quadratic eigenproblems. A symmetric QEP is *hyperbolic* if $M > 0$ and $(x^*Cx)^2 > 4(x^*Mx)(x^*Kx)$ for all $x \in \mathbb{C}^n \backslash \{0\}$.[1] In this case, all eigenvalues are guaranteed to be real, forming two disjoint groups of n eigenvalues, each having linearly independent eigenvectors. Hyperbolic QEPs are also characterized by the symmetric pencil Equation (7) being definite.[23] A particular case of hyperbolic QEP, often called overdamped, is when $C > 0$ and $K \geq 0$, in which case all its eigenvalues are nonpositive. On the other hand, a more general class of symmetric QEPs where all eigenvalues are real is the so-called definite QEP, characterized by the fact that a symmetric-definite linearization exists.[23,24] Symmetric QEPs of both hyperbolic and definite type can be transformed in a way that the B matrix of the symmetric linearization pencil Equation (7) is positive definite. A procedure for this has been proposed by Niendorf and Voss.[7] Then it is possible to apply the spectrum slicing technique of Section 2.1 without modification. However, we advocate for modifying the spectrum slicing method so that it can be used without the need of the Niendorf-Voss transformation, resulting in a cheaper computation as will be illustrated in Section 4. Furthermore, our approach may also be suitable for some general symmetric QEPs (not definite). For the modified spectrum slicing method, we are not exploiting the overdamped and definite structures, we will only distinguish between hyperbolic QEP and general symmetric QEP.

The spectrum slicing technique described in Section 2.1 relies on the eigenvalue count inside an interval, which is computed using the matrix inertia of $A - \sigma B$. To extend this scheme to the symmetric QEP we review similar results available for some particular QEP types, such as the hyperbolic case.

2.2.1 | Matrix inertia in quadratic symmetric matrix polynomials

In the sequel, we will use $n_\ell(\sigma)$ to denote the number of real eigenvalues to the left of $\sigma \in \mathbb{R}$ in a given symmetric eigenproblem (linear or quadratic).

In symmetric linear eigenvalue problems Equation (2), specific properties such as B being positive definite give us the possibility of computing $n_\ell(\sigma)$, which in this case is equal to the value of the matrix inertia $\nu(A - \sigma B)$. For the more general case of symmetric nonlinear eigenproblems, Kostić and Voss[25] give sufficient conditions that allow using the matrix inertia information to locate real eigenvalues. In the particular case of hyperbolic QEPs, they show that it is always possible to compute $n_\ell(\sigma)$ from the matrix inertia $\nu(Q(\sigma))$, in a similar way as in the linear case. Focusing particularly

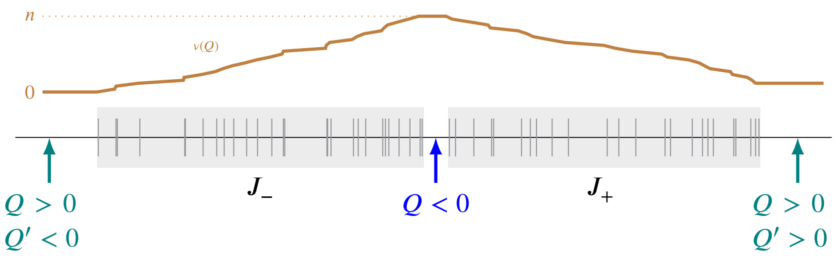

FIGURE 2 Pictorial representation of the two groups of eigenvalues of a hyperbolic QEP. The line $\nu(Q)$ illustrates how the value of the inertia of matrix $Q(\sigma)$ changes for varying σ

on the hyperbolic case, we now summarize results that have been used in the eigensolver implementation (Section 3) for computing $n_\ell(\sigma)$.

As mentioned before, the eigenvalues of a hyperbolic QEP form two disjoint groups of n real eigenvalues each,

$$\lambda_1^- \leq \lambda_2^- \leq \cdots \leq \lambda_n^- < \lambda_1^+ \leq \lambda_2^+ \leq \cdots \leq \lambda_n^+, \tag{8}$$

as shown in Figure 2. The intervals $J_- := [\lambda_1^-, \lambda_n^-]$ and $J_+ := [\lambda_1^+, \lambda_n^+]$ correspond to the two real solutions $\rho_-(x)$ and $\rho_+(x)$, respectively, of the scalar equation

$$x^*Q(\lambda)x = \lambda^2 x^*Mx + \lambda x^*Cx + x^*Kx = 0, \quad x \in \mathbb{C}^n \backslash \{0\}. \tag{9}$$

Another property of hyperbolic problems is that the matrix $Q(\sigma)$ is negative definite for all σ in the open interval $(\lambda_n^-, \lambda_1^+)$, and positive definite in the intervals $(-\infty, \lambda_1^-)$ and $(\lambda_n^+, +\infty)$.

To compute $n_\ell(\sigma)$, for a given shift $\sigma \in \mathbb{R}$, from the matrix inertia $\nu(Q(\sigma))$, we need to know the relative position of σ with respect to the intervals J_- and J_+. We distinguish three cases:[25]

- If $\nu(Q(\sigma)) = n$, then σ is located between the two intervals J_- and J_+, and hence there are n eigenvalues on the left of σ, $n_\ell(\sigma) = n$.
- If $\nu(Q(\sigma)) = 0$, then σ is either on the left of J_- or on the right of J_+. In that case, we compute $t := x^*Q'(\sigma)x$ for a random vector x to determine which side is σ on. It holds that $n_\ell(\sigma) = 2n$ if $t > 0$, and $n_\ell(\sigma) = 0$ otherwise.
- If $0 < \nu(Q(\sigma)) < n$, then σ is either inside J_+ or J_-. Similarly to the previous case, to discriminate between the two intervals we need to compute $t := x^*Q'(\sigma)x$, in this case for a vector x such that $x^*Q(\sigma)x > 0$. One such x is for example a Ritz vector approximating an eigenvector associated to a positive eigenvalue of $Q(\sigma)$. Then,

$$n_\ell(\sigma) = \begin{cases} \nu(Q(\sigma)), & \text{if } t < 0, \\ 2n - \nu(Q(\sigma)), & \text{if } t > 0. \end{cases} \tag{10}$$

The fact that $n_\ell(\sigma)$ can always be computed in the hyperbolic case makes the spectrum slicing technique applicable for these problems. The general case of symmetric QEPs is trickier. There may be both real and complex eigenvalues, and there is no general formula to compute $n_\ell(\sigma)$ from $\nu(Q(\sigma))$, even if all eigenvalues are real. As a consequence, the described technique cannot be applied in general for all symmetric QEPs. Still, we can find conditions that allow us to extend the spectrum slicing technique to certain nonhyperbolic problems. For that case, it is relevant to define eigenvalue types of symmetric matrix polynomials.[26] A real eigenvalue λ_i of $Q(\lambda)x = 0$ is of positive type if

$$x^*Q'(\lambda_i)x = 2\lambda_i x^*Mx + x^*Cx > 0, \quad \forall x \in \ker(Q(\lambda_i))\backslash\{0\}. \tag{11}$$

Likewise, it is of negative type if $x^*Q'(\lambda_i)x < 0$. We say that a real eigenvalue is of definite type if it is either of positive or negative type, and otherwise it is of mixed type. In the case of hyperbolic QEPs, all eigenvalues are of definite type, more precisely those contained in J_- are of negative type and those in J_+ of positive type.

Given an interval $[a, b]$, if all real eigenvalues contained in such interval are of definite type and, furthermore, all of them are of the same type (either positive or negative) and semisimple, then it is possible to show[12] that $n_\ell(b) - n_\ell(a) = |\nu(Q(b)) - \nu(Q(a))|$. Hence, under these assumptions, the spectrum slicing technique of Section 3 can be used to find all real eigenvalues contained in the interval in general symmetric QEPs. Unfortunately, it is not possible to know a priori if a given problem satisfies the assumptions or not.

2.3 | Memory-efficient Krylov solvers for the symmetric QEP

Our solver relies on a Krylov method operating on the pencil of the linearization Equation (7), which is not definite in general. As hinted in Section 2.1, we need to use an iterative solver that preserves the structure of the linearization, in a way that the computed eigenvectors can be readily used in the deflation mechanism required for spectrum slicing. This can be accomplished with pseudo-Lanczos.[9,10] Furthermore, due to the structure of the linearization it is very convenient to use specific memory-efficient methods such as STOAR.[10]

2.3.1 | The pseudo-Lanczos method to solve symmetric indefinite linear eigenproblems

The pseudo-Lanczos recurrence preserves the symmetric-indefinite structure of the problem by using an indefinite inner product,[27] $\langle u, v\rangle_B := u^*Bv$, for which $\langle z, z\rangle_B$ can be negative. The computed pseudo-Lanczos decomposition, analogous to Equation (5), is

$$SV_j = V_j\Omega_j^{-1}T_j + t_{j+1,j}\omega_{j+1}^{-1}v_{j+1}e_j^*, \tag{12}$$

where V_j is the Krylov basis satisfying $V_j^*BV_j = \Omega_j$, T_j is a real symmetric tridiagonal matrix, and $\Omega_j = \operatorname{diag}(\pm 1)$ is a signature matrix. Algorithm 1 shows the required steps to compute the pseudo-Lanczos decomposition.

Algorithm 1. Pseudo-Lanczos iteration

```
**Input:** Matrices $B$ and $S = (A - \sigma B)^{-1}B$, initial vector $v_1$, number of steps $k$
**Output:** Pseudo-Lanczos basis $V_{k+1}$, tridiagonal matrix $T_k$, signature matrix $\Omega_{k+1}$
    Normalize: $\omega_1 = \operatorname{sign}(v_1^*Bv_1)$,    $v_1 = v_1\omega_1/\sqrt{|v_1^*Bv_1|}$
    for $j = 1, 2, \dots, k$ do
        $z = Sv_j$
        $t_{1:j,j} = V_j^*Bz$
        $z = z - V_j\Omega_j^{-1}t_{1:j,j}$
        $t_{j+1,j} = \sqrt{|z^*Bz|}$
        $\omega_{j+1} = \operatorname{sign}(z^*Bz)$
        $v_{j+1} = z\omega_{j+1}/t_{j+1,j}$
    end for
```

Our implementation also incorporates a thick restart based on the eigenvectors of the projected matrix $\Omega_j^{-1}T_j$.[10]

The pseudo-Lanczos method can be seen as nonsymmetric Lanczos particularized for the case of a symmetric generalized eigenproblem. As such, it may suffer from breakdown (or near breakdown) in the case that an isotropic vector is generated, that is, if $\langle z, z\rangle_B = 0$ for $z \neq 0$. In addition, a solver based on pseudo-Lanczos may become unstable, either due to the pseudo-Lanczos iteration itself (involving indefinite Gram-Schmidt orthogonalization[28]) or the resolution of the symmetric-indefinite projected problem. A careful implementation of these two parts can reduce the risk of instability, and furthermore it is possible to detect instability by monitoring the loss of pseudo-symmetry in the projected problem.[10]

In the case of shift-and-invert transformation, the matrix used to expand the Krylov subspace is

$$S = (A - \sigma B)^{-1}B = \begin{bmatrix} -K - \sigma C & -\sigma M \\ -\sigma M & M \end{bmatrix}^{-1} \begin{bmatrix} C & M \\ M & 0 \end{bmatrix}. \tag{13}$$

A practical implementation should avoid the explicit computation of the inverse of $A - \sigma B$, and instead perform a linear solve whenever a matrix-vector product with S is required. Rather than factoring this $2n \times 2n$ matrix, a Schur complement approach is preferred, which involves operating with the (implicit) inverse of $Q(\sigma)$, that is, the quadratic polynomial Equation (3) evaluated at the current shift σ. Since $Q(\sigma)$ is symmetric (for a real σ), this can be performed with an indefinite (block-)triangular factorization $Q(\sigma) = LDL^*$, as described in Section 2.1 for the linear case. Then the inertia $\nu(Q(\sigma)) = \nu(D)$ is readily available to determine the number of eigenvalues in a subinterval.

2.3.2 | The STOAR method

The dimension of the matrices of the linearization Equation (7) is $2n$, where n is the size of the coefficient matrices of Q. This increased dimension not only implies double storage requirements for the Krylov basis, $V \in \mathbb{C}^{2n\times j}$ (for some dimension j), but also a high computational cost. Memory-efficient Krylov solvers[29] try to exploit the block structure of the linearization in a way that the memory requirements are restricted to a basis with vectors of dimension n only, $U \in \mathbb{C}^{n\times(j+1)}$, plus some additional coefficients, $G \in \mathbb{C}^{2(j+1)\times j}$. These bases are related as

$$V = (I_2 \otimes U)G. \tag{14}$$

These algorithms are also adapted to carry out much less operations than would be done in a naive implementation. Details of how these techniques have been implemented in SLEPc for general matrix polynomials of arbitrary degree are given in a previous article.30

Here, we focus on the particular case of symmetric matrix polynomials of degree 2. The details of the memory-efficient Krylov solver for this case have already been worked out in a previous work.[10] The STOAR method, that we will use in the spectrum slicing method of Section 3, is a variant of TOAR (two-level orthogonal Arnoldi)[29] that exploits symmetry by using the pseudo-Lanczos recurrence for the matrix

$$\check{S} = \check{B}^{-1}\check{A} = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix}, \tag{15}$$

associated with the linearization

$$\check{L}(\lambda) = \check{A} - \lambda\check{B}, \quad \text{with} \quad \check{A} = \begin{bmatrix} 0 & K \\ K & C \end{bmatrix}, \quad \check{B} = \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix}. \tag{16}$$

It generates a Krylov relation Equation (12) where the pseudo-Lanczos vectors are built in a compact form Equation (14),

$$V_j = \begin{bmatrix} V_j^0 \\ V_j^1 \end{bmatrix} = \begin{bmatrix} U_{j+1} & 0 \\ 0 & U_{j+1} \end{bmatrix} \begin{bmatrix} G_j^0 \\ G_j^1 \end{bmatrix}, \tag{17}$$

where G_j^0 and G_j^1 are the U_{j+1} coordinates of the blocks V_j^0 and V_j^1, respectively. We now summarize the main steps carried out by the STOAR method.[10]

The first step is the expansion of the Krylov subspace $z = \check{S}v_j$ (line 3 of Algorithm 1). Since the last pseudo-Lanczos vector $v_j = \begin{bmatrix} v_j^0 \\ v_j^1 \end{bmatrix}$ is expressed in the compact representation, $v_j^0 = U_{j+1}g_j^0, v_j^1 = U_{j+1}g_j^1$, a compact representation of $z = \begin{bmatrix} z^0 \\ z^1 \end{bmatrix}$, where

$$\begin{cases} z^0 = v_j^1 \\ z^1 = -M^{-1}(Kv_j^0 + Cv_j^1) = -M^{-1}(KU_{j+1}g_j^0 + CU_{j+1}g_j^1), \end{cases} \tag{18}$$

is computed taking $U_{j+2} = [U_{j+1}, u_{j+2}]$, with orthonormal columns, and coefficients $g = \begin{bmatrix} g^0 \\ g^1 \end{bmatrix}$ so that $z^0 = U_{j+2}g^0$, $z^1 = U_{j+2}g^1$. The vector u_{j+2} and coefficients g^1 are obtained via Gram-Schmidt orthonormalization of z^1 against the columns of U_{j+1}, whereas $g^0 = \begin{bmatrix} g_j^1 \\ 0 \end{bmatrix}$.

The orthogonalization and normalization steps (lines 4-8 of Algorithm 1) start with vector $z = (I_2 \otimes U_{j+2})g$. Using the fact that U_{j+2} has orthonormal columns, the Gram-Schmidt orthogonalization turn into the orthogonalization of g against the columns of G_j with the indefinite inner product defined by

$$\begin{bmatrix} U_{j+2}^*KU_{j+2} & 0 \\ 0 & -U_{j+2}^*MU_{j+2} \end{bmatrix}. \tag{19}$$

The normalization proceeds in the same way, by normalizing the resulting vector to obtain g_{j+1} and ω_{j+1}.

STOAR is a restarted method that compresses the Krylov basis when it reaches a specified maximum size. While reducing the Krylov basis from size m to p by a transformation of the form $\tilde{V}_p = V_m Q = (I_2 \otimes U_{m+1})G_m Q$, for $Q \in \mathbb{C}^{m\times p}$, this method performs (in the same way as TOAR), an additional step which involves an SVD decomposition of the matrix $\left[G_m^0, G_m^1\right]$ to adjust the size of the basis U_{m+1} to the new Krylov basis dimension.

3 | SPECTRUM SLICING FOR QUADRATIC EIGENPROBLEMS

In this section, we show how the spectrum slicing technique of Section 2.1 can be adapted to symmetric QEPs solved with the STOAR method of Section 2.3 using the information provided by inertia as discussed in Section 2.2.

3.1 | Basic scheme

Algorithm 2 shows a scheme of the spectrum slicing technique particularized for symmetric QEPs. The algorithm computes all real eigenvalues contained in a given interval $[a, b]$. The overall computation is similar to the case of linear eigenproblems, but with important differences that will be highlighted below.

Algorithm 2. Spectrum slicing for symmetric QEPs

```
Input: Matrices $A$ and $B$, interval $[a, b]$
Output: Computed eigenpairs $(x_i, \lambda_i)$ with $\lambda_i \in [a, b]$
    Set $k = 1$
    Compute $n_\ell(a), n_\ell(b)$
    $\Sigma = \{(a, a, b, n_\ell(a), n_\ell(b))\}$
    while $\Sigma \neq \emptyset$ do
        Extract $(\sigma_k, \sigma_\ell, \sigma_r, n_\ell(\sigma_\ell), n_\ell(\sigma_r))$ from $\Sigma$
        Compute $n_\ell(\sigma_k)$
        Select eigenvectors computed at $[\sigma_\ell, \sigma_r]$ as deflation vectors
        Run STOAR at $\sigma_k$ to obtain $m$ eigenpairs $(x_i, \lambda_i)$
        Discard $(x_i, \lambda_i)$ for $\lambda_i \notin [\sigma_\ell, \sigma_r]$
        Split eigenvalues in two groups $\{\lambda_i^\ell\}_{i=1}^{m_\ell}$ and $\{\lambda_i^r\}_{i=1}^{m_r}$ with $\lambda_i^\ell < \sigma_k < \lambda_i^r$
        if $m_\ell \neq n_\ell(\sigma_k) - n_\ell(\sigma_\ell)$ then
            Choose new shift $\sigma$ in $[\sigma_\ell, \sigma_k]$
            Insert $(\sigma, \sigma_\ell, \sigma_k, n_\ell(\sigma_\ell), n_\ell(\sigma_k))$ into $\Sigma$
        end if
        if $m_r \neq n_\ell(\sigma_r) - n_\ell(\sigma_k)$ then
            Choose new shift $\sigma$ in $[\sigma_k, \sigma_r]$
            Insert $(\sigma, \sigma_k, \sigma_r, n_\ell(\sigma_k), n_\ell(\sigma_r))$ into $\Sigma$
        end if
        $k = k + 1$
    end while
```

The computation proceeds by placing shifts σ_k at different points of the interval. Normally the method sweeps the interval either from left to right or right to left (depending on whether the interval is open in one end, for instance), but it does not mean that the values of the shifts increase (or decrease) monotonically, since in some cases the algorithm must get back to look for missing eigenvalues. There is a bag Σ of unprocessed shifts, where each shift is represented by a tuple $(\sigma_k, \sigma_\ell, \sigma_r, n_\ell(\sigma_\ell), n_\ell(\sigma_r))$ indicating the current shift value σ_k, the interval for deflation $[\sigma_\ell, \sigma_r]$ and the number of (real) eigenvalues on the left (inertia) of both ends of this interval. At the beginning of the algorithm, Σ must be filled with the initial shift, which is one of the ends of the interval, together with the whole interval $[a, b]$ (which can be open on one side) and the eigenvalue count at both a and b (see Section 3.4 for details about how $n_\ell(\cdot)$ is obtained). Next, the algorithm proceeds by extracting one shift at a time from the bag and processing it, until the bag is empty (at each shift, at most two new shifts may be inserted into the bag).

New shifts are added to the bag in the case that the number of computed eigenvalues is still less than the eigenvalue count provided by inertia (computed at line 6), either on the left or on the right of σ_k (or both). The values of the new shifts added to the bag (lines 12 and 16 of Algorithm 2) are determined in two ways. The normal case is when either σ_ℓ or σ_r coincide with one end of the global interval; in that case, the distance of the new shift with respect of the previous one takes into account the separation of computed eigenvalues.[4] In the case that the algorithm gets back to compute missing eigenvalues, then the new shift is just the midpoint of $[\sigma_\ell, \sigma_k]$ (or $[\sigma_k, \sigma_r]$).

Since eigenvalues are computed by chunks, from different pseudo-Lanczos factorizations at different shifts, it may happen that some of the eigenvalues are computed more than once. To avoid that, we perform explicit deflation, but only against a limited set of eigenvectors, as was mentioned in Section 2.1. Eigenvectors corresponding to computed eigenvalues located in $[\sigma_\ell, \sigma_r]$ are selected for deflation (line 7 of the algorithm), and this prevents reconvergence of these eigenvalues in the next pseudo-Lanczos run. In the event that an eigenvalue outside $[\sigma_\ell, \sigma_r]$ is computed we must discard it (line 9 of the algorithm) because we cannot tell if it is a genuine or spurious copy.

3.2 | Linearization

The computation associated with each shift consists essentially in a run of the STOAR method to compute a fixed number, m, of eigenpairs around σ_k. In our implementation, the employed linearization is a general one, that includes Equations (7) and (16) as particular cases. We consider symmetric pencils proposed by Higham et al[31] belonging to the space $\mathbb{H}(Q)$,

$$\hat{L}(\lambda) = \hat{A} - \lambda\hat{B}, \quad \text{with} \quad \hat{A} = \begin{bmatrix} \beta K & \alpha K \\ \alpha K & \alpha C - \beta M \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} \alpha K - \beta C & -\beta M \\ -\beta M & -\alpha M \end{bmatrix}, \tag{20}$$

with $\alpha, \beta \in \mathbb{R}$. A pencil of this type is a linearization for the matrix polynomial Equation (3) whenever the matrix

$$R_{\alpha,\beta} := \beta^2 M - \alpha\beta C + \alpha^2 K \tag{21}$$

is nonsingular.[31] The eigenvectors of the linearization have the form $y = \begin{bmatrix} x \\ \lambda x \end{bmatrix}$, where (x, λ) is an eigenpair of the QEP. The linearizations Equations (7) and (16) are particular cases of Equation (20) taking $(\alpha, \beta) = (1, 0)$ and $(0, 1)$, respectively. By leaving (α, β) as free parameters, we give the user the possibility to improve the properties of the indefinite inner product matrix $\hat{B}$ for a more stable Gram-Schmidt orthogonalization.[28]

For the spectrum slicing scheme, we need to perform the shift-and-invert transformation at each shift. Contrary to the approach used in our previous article,[10] where the transformation was done on the quadratic problem, the deflation technique, explained in Section 3.3, requires the shift-and-invert transformation to be performed on the linearization. The rationale is that in this latter case the transformed problems generated from different shifts have the same eigenvectors, which can be directly used from one shift to another for deflation.

To compute the action of $(\hat{A} - \sigma\hat{B})^{-1}\hat{B}$ on some vector $v \in \mathbb{C}^n$ without explicitly building and factorizing the matrix $(\hat{A} - \sigma\hat{B})$ of size $2n$, we use a block factorization. Assuming $\beta \neq 0$, the factorization is

$$\begin{aligned} \hat{A} - \sigma\hat{B} &= \begin{bmatrix} \beta K - \sigma\alpha K + \sigma\beta C & \alpha K + \sigma\beta M \\ \alpha K + \sigma\beta M & \alpha C - \beta M + \sigma\alpha M \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ \frac{\alpha}{\beta} I & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \frac{1}{\beta} R_{\alpha,\beta} \end{bmatrix} \begin{bmatrix} I & -(\alpha K + \sigma\beta M) \\ 0 & I \end{bmatrix} \begin{bmatrix} \beta Q(\sigma) & 0 \\ \sigma I & -I \end{bmatrix}, \end{aligned} \tag{22}$$

with $R_{\alpha,\beta}$ defined above. From decomposition Equation (22) we get

$$(\hat{A} - \sigma\hat{B})^{-1}\hat{B} = \begin{bmatrix} \beta Q(\sigma) & 0 \\ \sigma I & -I \end{bmatrix}^{-1} \begin{bmatrix} I & -(\alpha K + \sigma\beta M) \\ 0 & I \end{bmatrix}^{-1} \tilde{B}, \tag{23}$$

where

$$\tilde{B} := \begin{bmatrix} I & 0 \\ 0 & \beta R_{\alpha,\beta}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -\frac{\alpha}{\beta} I & I \end{bmatrix} \hat{B} = \begin{bmatrix} \alpha K - \beta C & -\beta M \\ -I & 0 \end{bmatrix}. \tag{24}$$

Expressions (23) and (24) allow us to make computations for $z = (\hat{A} - \sigma\hat{B})^{-1}\hat{B}v$ by blocks, operating with matrices and vectors of size n, with the formula

$$\begin{cases} z^0 = -Q(\sigma)^{-1}(\sigma M v^0 + C v^0 + M v^1) \\ z^1 = \sigma z^0 + v^0. \end{cases} \tag{25}$$

If $\beta = 0$, it is possible to derive the same formula assuming $\alpha \neq 0$. Considering that v is expressed in compact representation, vector z must also be computed in this representation as explained in Section 2.3.

As in the linear case, the computation of Equation (25) must not evaluate the inverse of $Q(\sigma)$ explicitly. Instead, a symmetric-indefinite triangular factorization of $Q(\sigma)$ is used to expand the basis. Furthermore, this factorization will provide the inertia of this matrix, which informs about the localization of eigenvalues, as discussed in Section 2.

3.3 | Deflating eigenvectors in compact representation

As mentioned in Section 2.1, the deflation mechanism for spectrum slicing consists in augmenting the Krylov subspace with already available eigenvectors. These eigenvectors have been computed in a previous shift σ and deflation prevents their re-computation. In the case of Algorithm 2, that uses the STOAR method, we augment the pseudo-Lanczos basis, which is expressed in the compact representation Equation (17), with eigenvectors of the linearization $\begin{bmatrix} x \\ \lambda x \end{bmatrix}$, which are also expressed in compact representation. This has several implications:

- During the spectrum slicing loop it is necessary to keep eigenvectors of the linearization $\begin{bmatrix} x \\ \lambda x \end{bmatrix}$, and only at the end of the computation we extract the relevant part x. Note that this does not mean an increase of memory storage, since $\begin{bmatrix} x \\ \lambda x \end{bmatrix}$ needs to store about n elements in compact representation.
- The management of the subspace basis is much more involved than in the linear case, because augmenting the basis is not as simple as appending new columns, due to the compact representation, and similarly for other operations such as extracting a number of columns from the subspace basis.

To illustrate the issues associated with management of the basis, let us consider several situations. The simplest case is when after step 7 of Algorithm 2 we have a basis of p deflation vectors expressed in compact representation $V_p = (I_2 \otimes U_p)G_p$. Deflation in pseudo-Lanczos is effected by including these vectors in the orthogonalization of steps 4-5 of Algorithm 1. A simple way to achieve this is to initialize the pseudo-Lanczos basis to V_p and then start the loop at $j = p$. This will extend the basis naturally to order k, and we just have to consider only the trailing principal submatrix of order $(k - p)$ of the projected matrices T_k and Ω_k (the leading part contains the already known eigenvalues).

The situation is somewhat more complicated when the set V of vectors selected for deflation is just a part of the eigenvectors obtained from a previous shift, or when such set includes vectors computed from two distinct shifts. In the first case, V would be stored in compact representation $V = (I_2 \otimes U)G$ with G having less columns than U, making it necessary to perform an operation to adjust and level off both dimensions. In the second case, where we have two sets of vectors $V_1 = (I_2 \otimes U_1)G_1$ and $V_2 = (I_2 \otimes U_2)G_2$, obtained from two different STOAR runs, we would need to orthogonally extend one of the two bases, for instance, $\tilde{U} = [U_1, \tilde{U}_2]$, in such a way that the new basis $\tilde{U}$ spans both U_1 and U_2, and we would get in this way a unified compact representation for the two groups of eigenvectors $[V_1, V_2] = (I_2 \otimes \tilde{U})[G_1, \tilde{G}_2]$.

The latter two cases get simplified if after every STOAR run the basis of computed eigenvectors (that includes those used for deflation) is divided in two groups (corresponding to eigenvalues lying on the left and right of the shift), each of them represented independently in compact form. In this way, the vectors associated with each of the subintervals generated at steps 11 and 15 of Algorithm 2, get stored directly in the appropriate way for their use in deflation of subsequent subintervals.

The basis split is accomplished by means of a small computation involving the SVD of a matrix as follows. The first $q < p$ vectors of the computed basis are $V_q = (I_2 \otimes U_p)G_q$ with $U_p \in \mathbb{C}^{n\times p}$ and $G_q \in \mathbb{C}^{2p\times q}$, so we need to adjust the number of columns of U_p and rows of G_q. The rank of $[V_q^0, V_q^1]$ is q and similarly for $[G_q^0, G_q^1] \in \mathbb{C}^{p\times 2q}$. We then compute the compact singular value decomposition

$$\begin{bmatrix} G_q^0 & G_q^1 \end{bmatrix} = \check{U}\check{\Sigma}\check{V}^*, \tag{26}$$

where $\check{\Sigma}$ is a $q \times q$ diagonal matrix of singular values, and $\check{U}$, $\check{V}$ have dimensions $p \times q$ and $q \times 2q$, respectively. This decomposition allows us to express the V_q vectors in the form

$$V_q^i = U_p G_q^i = U_p \check{U}\check{\Sigma}\check{V}^{i^*}, \quad \text{for } i = 0, 1, \tag{27}$$

where we have written $\check{V}^* = [\check{V}^{0^*}, \check{V}^{1^*}]$. Updating $U_p \leftarrow U_p\check{U}$ and $G_q^i \leftarrow \check{\Sigma}\check{V}^{i^*}$, for $i = 0, 1$, we obtain the representation for V_q we are looking for.

3.4 | Additional details

To complement the description, we provide some additional implementation details.

A fundamental value in Algorithm 2 is $n_\ell(\sigma)$, the number of real eigenvalues to the left of $\sigma \in \mathbb{R}$. In a practical implementation this value is explicitly computed only in the case of hyperbolic QEPs, but not in the general symmetric case that we will discuss below.

In the hyperbolic case, the computation of $n_\ell(\sigma)$ has been described in Section 2.2. Apart from computing the inertia of $Q(\sigma)$ (which implies a factorization), the case $0 < \nu(Q(\sigma)) < n$ involves an additional cost because it requires computing a value $t := x^*Q'(\sigma)x$ to determine the relative position of σ with respect to the intervals J_- and J_+. In our implementation, x is an approximate Ritz vector of $Q(\sigma)$ corresponding to a positive eigenvalue, and we compute it via a linear eigensolver from SLEPc's `EPS` module. This computation is required unconditionally at the beginning of Algorithm 2 (step 2), but inside the loop we can avoid it unless the endpoints of the interval satisfy $a \in J_-, b \in J_+$.

As we have mentioned in Section 2.2, in the general symmetric case there is no known way to explicitly compute $n_\ell(\sigma)$. Still, the spectrum slicing technique can be implemented if the interval $[a, b]$ contains only eigenvalues of the same definite type. The algorithm in this case works similarly to Algorithm 2, but storing the inertias $\nu(Q(\sigma))$ instead of $n_\ell(\sigma)$, and using $|\nu(Q(\sigma_2)) - \nu(Q(\sigma_1))|$ as a count of the number of eigenvalues in $[\sigma_1, \sigma_2]$. If the assumption that the interval contains only eigenvalues of the same type does not hold, there is no guarantee of a proper functioning and it may happen that the number of returned eigenvalues is smaller than the actual number of real eigenvalues in $[a, b]$. To minimize the occurrence of this latter scenario, we have implemented several mechanisms to check situations where the required assumption does not hold:

- The first check is done at the beginning of the algorithm on both endpoints of the initial interval, a and b. For each of them, we compute the eigenpair (x, λ) satisfying the QEP $Q(\lambda)x = 0$ for λ inside the interval $[a, b]$ and closest to the endpoint. The type of this eigenvalue is given by the sign of $x^*Q'(\lambda)x$. If the sign is different in both eigenvalues, then the solver aborts because the assumption does not hold.
- If the previous check did not fail, the eigenvalue type must also be checked for all eigenvalues as they are computed, because in principle eigenvalues of different type may be mixed arbitrarily. In this case, instead of explicitly computing the eigenvector x of the quadratic eigenproblem, the type of the eigenvalue λ can be determined by checking the type of λ directly in the linearization. From the relation

$$y^*\hat{L}'(\lambda)y = -y^*\hat{B}y = (\alpha\lambda + \beta)x^*Q'(\lambda)x, \tag{28}$$

where $y = \begin{bmatrix} x \\ \lambda x \end{bmatrix}$, we see that the type of λ in the QEP can be easily determined from the type of λ in the linearization $\hat{L}$. Furthermore, the sign of $y^*\hat{B}y$ is readily available at the end of each STOAR run. If the solver detects eigenvalues of different type within the interval, the execution is aborted with an informative error message.
- The algorithm will also abort the execution if it detects a mismatch between the inertia count and the number of found eigenvalues in a given subinterval.

Still, in the general case there could be cases in which the solver does not detect that the input interval does not satisfy the required conditions. In that case, the number of returned eigenvalues would be less than the actual number of eigenvalues contained in the interval.

We conclude this section with a brief description of the parallel implementation. SLEPc relies on PETSc[32] for linear algebra operations such as sparse matrix-vector products or linear solves. These operations are parallelized for distributed memory parallel computers with a message-passing paradigm (MPI). Oversimplifying, parallelizing an algorithm essentially amounts to deciding which objects (matrices and vectors) must be parallel (stored in a distributed way), and then PETSc (and SLEPc) internally manage the data exchange necessary to perform the different operations in parallel. Regarding Algorithm 2, the main cost lies in the computation of the inertia via a triangular factorization, as well as the STOAR runs for each shift. The inertia involves a triangular factorization, and parallel implementations of such factorizations are available in PETSc via third party packages such as MUMPS. For the STOAR runs we can distinguish two main building blocks: on one hand the operations in Equation (25), including triangular solves and matrix-vector products with the matrix polynomial coefficients, and on the other hand the orthogonalization of vectors. For the latter operation, we have implemented parallel subroutines in SLEPc that operate with vectors expressed in a compact representation Equation (14). Recall that, in this representation, the U basis consists of a set of long vectors (of length n) that will be distributed in the parallel code, whereas the G factor is in comparison very small and hence it must be stored redundantly in all MPI processes.

4 | EXPERIMENTAL RESULTS

We now show how our new spectrum slicing solver behaves in terms of accuracy as well as computational performance (including parallel efficiency). We first focus on hyperbolic problems and defer the general symmetric case until the end of this section.

The computer used for the executions is Tirant 3, consisting of 336 computing nodes, each of them with two Intel Xeon SandyBridge E5-2670 processors (16 cores each) running at 2.6 GHz with 32 GB of memory, interconnected with an Infiniband FDR10 network. All runs placed at most 4 MPI process per node. The presented results correspond to SLEPc version 3.10, together with PETSc 3.10 and MUMPS 5.1.2. All software has been compiled with Intel C and Fortran compilers (version 18) and Intel MPI.

For the hyperbolic QEPs, we make use of the NLEVP collection.[33] In particular, we consider two problems:

- spring is a QEP obtained from the discretization of a linearly damped mass-spring system. It is a monic polynomial $Q(\lambda) = \lambda^2 I + \lambda C + K$, where both C and K are tridiagonal. We consider the simplest case where all masses are equal, and all dampers and springs have the same damping constants. In particular, we use the default values $\mu = 1$ (masses), $\tau = 10$ (dampers) and $\kappa = 5$ (springs), which result in a hyperbolic (and overdamped) problem.
- loaded_string is a rational eigenvalue problem arising from the discretization of a loaded vibrating string, with a load of mass m attached by an elastic spring of stiffness k. Considering $\mu = k/m$, the problem is formulated as

$$\left(A - \lambda B + \frac{\lambda}{\lambda - \mu} C\right) x = 0, \tag{29}$$

  with A, B tridiagonal and positive definite, and $C = k e_n e_n^T$. This problem can be easily transformed to a QEP by multiplication with $\lambda - \mu$. We consider $m = 1$ and $k = 1$, in which case the resulting QEP is hyperbolic.

Table 1 shows information and results for these two problems. The spring problem has been solved for two different sizes and two different intervals, one of them is half-bounded and the other one intersects both J^- and J^+ and hence contains eigenvalues of both negative and positive type. In this way, we show that the solver is able to cope with all cases that may appear in hyperbolic problems.

The accuracy of a computed solution (x, λ) is assessed by means of its relative backward error

$$\eta(x, \lambda) = \frac{\|Q(\lambda)x\|}{\left(|\lambda^2|\|M\| + |\lambda|\|C\| + \|K\|\right)\|x\|}, \tag{30}$$

Problem	n	Interval	n_{eig}	$\#\sigma$	$\eta_{\max}$	t (s)
Spring	1.5 mill.	$(-\infty, -49.494891]$	411	29	$3 \cdot 10^{-14}$	181
Spring	20 000	$[-9.7, -0.5277]$	1423	99	$1 \cdot 10^{-11}$	41.3
Loaded_string	20 000	$[4, 100000]$	101	11	$1 \cdot 10^{-10}$	2.7

Note: For each problem, the table shows the matrix dimensions, n, the interval of interest, and the number of eigenvalues contained in the interval, n_{eig}. Results include the number of processed shifts, $\#\sigma$, the maximum backward error, $\eta_{\max}$, and the run time (in seconds) with 16 MPI processes.

TABLE 1 Computational results for the spectrum slicing solver with hyperbolic quadratic matrix polynomials

Problem	n	$\#\sigma$	$\eta_{\max}$	t_{slice} (s)	t_{transf} (s)
Spring	1.5 mill.	27	$5 \cdot 10^{-14}$	175	36.6
Spring	20 000	100	$4 \cdot 10^{-12}$	43.4	0.42

Note: Results include the number of processed shifts, $\#\sigma$, the maximum backward error, $\eta_{\max}$, the run time (in seconds) with 16 MPI processes for the initial transformation, t_{transf}, and for the spectrum slicing computation, t_{slice}.

TABLE 2 Computational results for the spectrum slicing solver with hyperbolic quadratic matrix polynomials, where the Niendorf-Voss transformation has been previously applied

where we use ∞-norms for practical computation of matrix norms. Note that the solver does not evaluate this backward error in the stopping criterion. Instead, the stopping criterion relies on the residual of the linearization, that is, readily available during the pseudo-Lanczos iteration. By default, we check $\|r_i\|/|\tilde{\theta}_i| < \texttt{tol}$, where $\tilde{\theta}_i$ is the approximate eigenvalue and r_i is the residual Equation (6). The tolerance used for the experiments in this section was 10^{-10}.

The results of Table 1 show that all computed eigenvalues satisfy the requested tolerance. Apart from the reported cases, we have tested the solver with many other problem sizes and intervals, serially and in parallel, and we can conclude that it is remarkably robust in all cases for hyperbolic problems. It is worth mentioning that even in the case that one of the STOAR runs quits earlier due to loss of symmetry (as mentioned in Section 2.3), the overall algorithm tolerates this situation because it can recover by computing missing eigenvalues from a different shift. In terms of performance, the run times largely depend on how many shifts are processed by the spectrum slicing algorithm, since each shift implies a sparse factorization. In any case, the total computation time is generally much smaller compared with a naive solution of the linearized problem, apart from the fact that this latter approach does not guarantee that all eigenpairs are returned.

Table 2 shows the results for the same problems as in Table 1 but solved by previously applying the Niendorf-Voss transformation (cf. Section 2.2), consisting in shifting the QEP with a value μ such that matrix $Q(\mu)$ is negative definite. Then, the symmetric linearization pencil has a definite B matrix, and the spectrum slicing method will operate always with a definite inner product. In the spring problem, both approaches provide a similar accuracy, and the computation time corresponding to the spectrum slicing method is also similar. However, the Niendorf-Voss transformation has an additional cost that in the case of $n = 1.500.000$ is nonnegligible. The loaded_string problem was not included in Table 2 since we could not carry out the Niendorf-Voss transformation because the required linear eigenproblems did not converge.

Results for parallel performance when the solver is applied to the spring problem of size 1.5 million are reported in Figure 4 (left), with up to 128 MPI processes. We can see that the cost associated with orthogonalization of vectors (expressed in compact representation) scales almost linearly. However, the overall scalability is hindered by the factorization and linear solves (red line) that have worse parallel efficiency. It is well known that parallel direct solvers do not scale well beyond a few tens of processes. For intervals containing many eigenvalues, it would be possible to implement a two-level parallelization scheme, where the interval is split in several subintervals, each of them being processed by a subset of MPI processes. In this way, the number of processes participating in a single factorization is limited. We have implemented this in SLEPc in the spectrum slicing algorithm for linear problems,[20] but not yet for quadratic eigenproblems.

We now turn our attention to symmetric QEPs that are not hyperbolic. We consider three problems, the first two belonging to the NLEVP collection. The information associated with these problems is shown in Table 3.

TABLE 3 Computational results for the spectrum slicing solver with symmetric quadratic matrix polynomials (nonhyperbolic)

Problem	n	Interval	n_{eig}	$\#\sigma$	$\eta_{\max}$	t (s)
Spring (n/o)	20 000	$[-2, -1.55]$	215	14	$6 \cdot 10^{-12}$	1.8
Sleeper	1.5 mill.	$[-0.99, -0.97]$	234	21	$9 \cdot 10^{-12}$	144
Atmos (0.5 Hz)	40 000	$[0.00628, 0.00973]$	230	18	$2 \cdot 10^{-13}$	3
Atmos (2 Hz)	40 000	$[0.02513, 0.03779]$	844	67	$2 \cdot 10^{-13}$	11

Note: See Table 1 for a description of the columns.

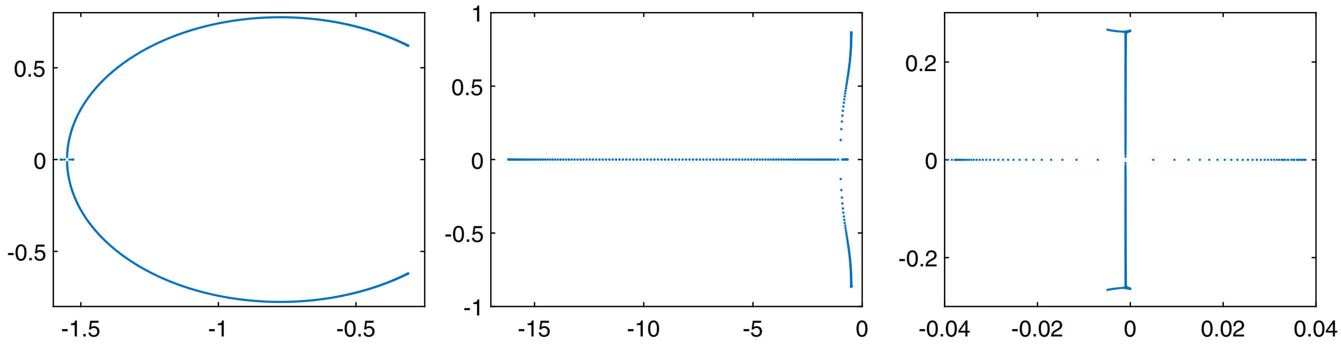

FIGURE 3 Full spectrum of the three nonhyperbolic problems: nonoverdamped spring (left), sleeper (center), and atmos (right). In all cases, the matrix size used to generate the plots is 400

- spring is the same problem considered before, but with modified parameters $\tau = 0.6202$ and $\kappa = 0.4807$ such that the QEP becomes nonhyperbolic. This problem is referred to as nonoverdamped mass-spring system by Li and Cai.[12]
- sleeper is a QEP modelling a railtrack resting on sleepers. It is a monic polynomial $Q(\lambda) = \lambda^2 I + \lambda C + K$, where both C and K are pentadiagonal.
- atmos is the name we use to refer to a QEP from modal analysis of atmospheric infrasound propagation.[34] It corresponds to a novel formulation valid for wide-angle propagation in atmospheres with high Mach number flow, summarized in this equation

$$\left[k_H^2\left(\frac{v_0^2}{c^2}-1\right)-k_H\frac{2\omega v_0}{c^2}+\frac{d^2}{dz^2}+\frac{\omega^2}{c^2}\right]\psi = 0. \tag{31}$$

A finite difference discretization is used along a vertical grid (z), resulting in a symmetric QEP with M, C diagonal, and K tridiagonal.

Figure 3 shows the full spectrum of these problems, corresponding to a problem of size $n = 400$. We can see that there are both real and complex eigenvalues. We have checked that, in these problems, real eigenvalues located on the left of a particular value are of negative type, and those on the right are of positive type. This implies that our spectrum slicing solver should work without difficulties, provided that the requested interval does not extend over eigenvalues of the two groups. The intervals shown in Table 3 satisfy this restriction*, and in the case of the atmos problem they represent the range of values with physical interest for the application at the given frequency.

Results in Table 3 show that our solver provides good accuracy also for the general symmetric case, with very fast computation times. Regarding the performance, the same comments in previous paragraphs also apply in this case. In particular, the scalability plot of Figure 4 for the nonhyperbolic case (sleeper, right plot) shows a very similar trend as before.

*We have also considered tests where the requested interval does not satisfy the assumption of having only eigenvalues of the same type, and our solver has detected this situation in all cases.

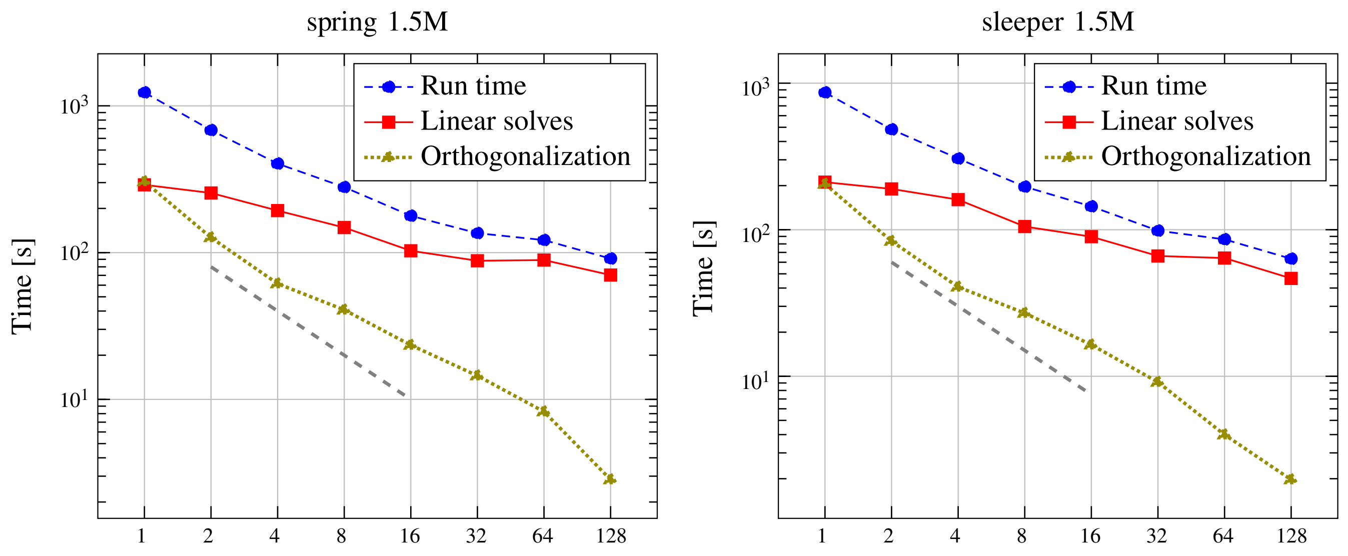

FIGURE 4 Execution times (in seconds) with up to 128 MPI processes for the spectrum slicing solver on the spring (left) and sleeper (right) problems of dimension 1.5 million. The problem parameters are shown in Tables 1 and 3, respectively

5 | CONCLUSIONS

We have designed a spectrum slicing method for symmetric (or Hermitian) QEPs that, under certain conditions, is able to compute all real eigenvalues contained in a given interval. This method extends the well-known inertia-based spectrum slicing technique for linear symmetric-definite eigenproblems, by relying on different ingredients such as the pseudo-Lanczos recurrence, the symmetric linearization of quadratic matrix polynomials and the inertia count. Our implementation is also memory-efficient as it stores the basis vectors in a compact representation, which enables significant memory savings at the cost of a more complicated implementation of deflation during the iteration.

Our solver is particularly effective for the special case of hyperbolic QEPs. In this case, the user can specify any interval of interest and the solver returns all the eigenvalues lying inside it. The more general case of symmetric QEPs can also be addressed, but only if the user-provided interval contains eigenvalues of the same definite type (positive or negative). Our solver is also compatible with the Niendorf-Voss transformation,[7] so if the QEP has been transformed with this method the spectrum slicing computation will work all the way using a definite inner product, and still use a memory-efficient representation of the basis.

Regarding the performance of the solver, we have presented results of both serial and parallel executions on various test cases. The parallel scalability is limited by the required factorizations at each shift. As a future work we will consider a two-level parallelism (based on subdivision of the interval) in such a way that the number of processes can be increased further without losing efficiency.

ACKNOWLEDGEMENTS

The computational experiments of Section 4 were carried out on the supercomputer Tirant 3 belonging to Universitat de València. The matrices from the atmospheric infrasound propagation problem were kindly provided by Assink et al.34

CONFLICTS OF INTEREST

This work does not have any conflicts of interest.

ORCID

Jose E. Roman https://orcid.org/0000-0003-1144-6772

REFERENCES

1. Tisseur F, Meerbergen K. The quadratic eigenvalue problem. SIAM Rev. 2001;43(2):235–286.
2. Veselić K. Damped oscillations of linear systems. Berlin Heidelberg / Germany: Springer-Verlag, 2011.

3. Grimes RG, Lewis JG, Simon HD. a shifted block lanczos algorithm for solving sparse symmetric generalized eigenproblems. SIAM J Matrix Anal Appl. 1994;15(1):228–272.
4. Campos C, Roman JE. Strategies for spectrum slicing based on restarted Lanczos methods. Numer Alg. 2012;60(2):279–295.
5. Li R, Xi Y, Vecharynski E, Yang C, Saad Y. A thick-restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems. SIAM J Sci Comput. 2016;38(4):A2512–A2534.
6. Guo CH, Higham NJ, Tisseur F. An improved arc algorithm for detecting definite Hermitian pairs. SIAM J Matrix Anal Appl. 2009;31(3):1131–1151.
7. Niendorf V, Voss H. Detecting hyperbolic and definite matrix polynomials. Linear Algebra Appl. 2010;432(4):1017–1035.
8. Nakatsukasa Y, Noferini V. Inertia laws and localization of real eigenvalues for generalized indefinite eigenvalue problems; 2017. Preprint arXiv:1711.00495.
9. Parlett BN, Chen HC. Use of indefinite pencils for computing damped natural modes. Linear Algebra Appl. 1990;140(1):53–88.
10. Campos C, Roman JE. Restarted Q-Arnoldi-type methods exploiting symmetry in quadratic eigenvalue problems. BIT. 2016;56(4):1213–1236.
11. Guo CH, Lancaster P. Algorithms for hyperbolic quadratic eigenvalue problems. Math Comp. 2005;74(252):1777–1791.
12. Li H, Cai Y. Solving the real eigenvalues of Hermitian quadratic eigenvalue problems via bisection. Electron J Linear Algebra. 2015;30(1):721–743.
13. Roman JE, Campos C, Romero E, and Tomas A. SLEPc users manual. DSIC-II/24/02–Revision 3.9. D. Sistemes Informàtics i Computació, Universitat Politècnica de València; 2018.
14. Hernandez V, Roman JE, Vidal V. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. ACM Trans Math Softw. 2005;31(3):351–362.
15. Guo JS, Lin WW, Wang CS. Numerical solutions for large sparse quadratic eigenvalue problems. Linear Algebra Appl. 1995;225:57–89.
16. Sleijpen GLG, Booten AGL, Fokkema DR, vander Vorst HA. Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems. BIT. 1996;36(3):595–633.
17. Bai Z, Su Y. SOAR: A second-order Arnoldi method for the solution of the quadratic eigenvalue problem. SIAM J Matrix Anal Appl. 2005;26(3):640–659.
18. Güttel S, Tisseur F. The nonlinear eigenvalue problem. Acta Numerica. 2017;26:1–94.
19. Yang L, Sun Y, Gong F. The inexact residual iteration method for quadratic eigenvalue problem and the analysis of convergence. J Comput Appl Math. 2018;332:45–55.
20. Keçeli M, Corsetti F, Campos C, et al. SIESTA-SIPs: Massively parallel spectrum-slicing eigensolver for an ab initio molecular dynamics package. J Comput Chem. 2018;39(22):1806–1814.
21. Horn RA, Johnson CR. Matrix analysis. 2nd ed. Cambridge, MA: Cambridge University Press, 2013.
22. Voss H, Werner B, Hadeler KP. A minimax principle for nonlinear eigenvalue problems with applications to nonoverdamped systems. Math Meth Appl Sci. 1982;4(1):415–424.
23. Higham NJ, Mackey DS, Tisseur F. Definite matrix polynomials and their linearization by definite pencils. SIAM J Matrix Anal Appl. 2009;31(2):478–502.
24. Al-Ammari M, Tisseur F. Hermitian matrix polynomials with real eigenvalues of definite type Part I: Classification. Linear Algebra Appl. 2012;436(10):3954–3973.
25. Kostić A, Voss H. On Sylvester's law of inertia for nonlinear eigenvalue problems. Electron Trans Numer Anal. 2013;40:82–93.
26. Gohberg I, Lancaster P, Rodman L. Spectral analysis of selfadjoint matrix polynomials. Ann Math. 1980;112(1):33–71.
27. Gohberg I, Lancaster P, Rodman L. Indefinite linear algebra and applications. Basel, Switzerland: Birkhaüser, 2005.
28. Rozložník M, Okulicka-Dluzewska F, Smoktunowicz A. Cholesky-like factorization of symmetric indefinite matrices and orthogonalization with respect to bilinear forms. SIAM J Matrix Anal Appl. 2015;36(2):727–751.
29. Lu D, Su Y, Bai Z. Stability analysis of the two-level orthogonal Arnoldi procedure. SIAM J Matrix Anal Appl. 2016;37(1):195–214.
30. Campos C, Roman JE. Parallel Krylov solvers for the polynomial eigenvalue problem in SLEPc. SIAM J Sci Comput. 2016;38(5):S385–S411.
31. Higham NJ, Mackey DS, Mackey N, Tisseur F. Symmetric linearizations for matrix polynomials. SIAM J Matrix Anal Appl. 2006;29(1):143–159.
32. Balay S, Abhyankar S, Adams M, et al. PETSc users manual. ANL-95/11 - Revision 3.10. Argonne National Laboratory; 2018.
33. Betcke T, Higham NJ, Mehrmann V, Schröder C, Tisseur F. NLEVP: A collection of nonlinear eigenvalue problems. ACM Trans Math Software. 2013;39(2):7:1–7:28.
34. Assink J, Waxler R, Velea D. A wide-angle high mach number modal expansion for infrasound propagation. J Acoust Soc Am. 2017;141(3):1781–1792.

How to cite this article: Campos C, Roman JE. Inertia-based spectrum slicing for symmetric quadratic eigenvalue problems. *Numer Linear Algebra Appl*. 2020;e2293. https://doi.org/10.1002/nla.2293