BIT manuscript No.
(will be inserted by the editor)

Restarted Q-Arnoldi-type methods exploiting symmetry in quadratic eigenvalue problems

Carmen Campos · **Jose E. Roman**

Received: date / Accepted: date

Abstract We investigate how to adapt the Q-Arnoldi method for the case of symmetric quadratic eigenvalue problems, that is, we are interested in computing a few eigenpairs (λ,x) of $(\lambda^2M+\lambda C+K)x=0$ with M,C,K symmetric $n\times n$ matrices. This problem has no particular structure, in the sense that eigenvalues can be complex or even defective. Still, symmetry of the matrices can be exploited to some extent. For this, we perform a symmetric linearization $Ay=\lambda By$, where A,B are symmetric $2n\times 2n$ matrices but the pair (A,B) is indefinite and hence standard Lanczos methods are not applicable. We implement a symmetric-indefinite Lanczos method and enrich it with a thick-restart technique. This method uses pseudo inner products induced by matrix B for the orthogonalization of vectors (indefinite Gram-Schmidt). The projected problem is also an indefinite matrix pair. The next step is to write a specialized, memory-efficient version that exploits the block structure of A and B, referring only to the original problem matrices M,C,K as in the Q-Arnoldi method. This results in what we have called the Q-Lanczos method. Furthermore, we define a stabilized variant analog of the TOAR method. We show results obtained with parallel implementations in SLEPc.

Keywords Quadratic eigenvalue problem · Pseudo-Lanczos · Q-Arnoldi · TOAR · Thick-restart · SLEPc

Mathematics Subject Classification (2000) 65F15 · 15A18 · 65F50

This work was supported by the Spanish Ministry of Economy and Competitiveness under grant TIN2013-41049-P. Carmen Campos was supported by the Spanish Ministry of Education, Culture and Sport through an FPU grant with reference AP2012-0608.

C. Campos
Departament de Sistemes Informàtics i Computació, Universitat Politècnica de València, Camí de Vera s/n, 46022 València (Spain), E-mail: mccampos@dsic.upv.es.

J. E. Roman
Departament de Sistemes Informàtics i Computació, Universitat Politècnica de València, Camí de Vera s/n, 46022 València (Spain), E-mail: jroman@dsic.upv.es.

1 Introduction

We address the question of how to exploit symmetry of coefficient matrices when solving the quadratic eigenvalue problem. This was already analyzed in a 1990 paper by Parlett and Chen [21], where the authors demonstrate the use of a pseudo-Lanczos recurrence operating on a symmetric linearization of the quadratic eigenproblem. We want to extend this methodology by incorporating several new ingredients to make its practical use more appealing, such as an effective restarting mechanism and memory-efficient variants that avoid storage of double-sized bases. The ultimate goal is to have symmetric eigensolvers that are as robust as possible, on a par with Arnoldi-based counterparts.

Although the methods discussed in this paper are relevant for many application areas, the paradigmatic example arises in the analysis of damped vibrating structures, where the discretization of the equations of motion results in the quadratic eigenvalue problem (QEP) of the form

$$(\lambda^2 M + \lambda C + K)x = 0. \tag{1.1}$$

The $n \times n$ coefficient matrices M, C and K (mass, damping and stiffness matrices, respectively) are all real and symmetric. The eigenvector x is related to displacements of the structure being analyzed, and one is normally interested in computing a few modes of vibration corresponding to eigenvalues λ located in a certain region of the complex plane. Results in this paper also apply to the case of complex Hermitian matrices. In both cases, the eigenvalues are real or come in complex conjugate pairs $(\lambda, \bar{\lambda})$.

In the simplest case, the eigenproblem (1.1) has $2n$ finite solutions, i.e., there exist $2n$ complex scalars λ, not necessarily distinct, that, together with the corresponding x, satisfy (1.1). More generally, the problem can have infinite eigenvalues if M is singular. In the previous discussion and throughout the paper, we will assume that the matrix polynomial $Q(\lambda) = \lambda^2 M + \lambda C + K$ is regular, that is, $\det Q(\lambda)$ is not identically zero for all values of λ. For further details regarding the properties of the QEP, the reader is referred to [28].

Iterative methods for computing a few eigenpairs of large-scale, sparse quadratic eigenproblems are based on subspace projection. Some of these methods (e.g., SOAR [1] or Jacobi-Davidson [24]) directly project the quadratic eigenproblem, resulting in a small-sized QEP, whereas other approaches consist in projecting an equivalent linear eigenvalue problem. We focus on the latter category of methods, where it is possible to apply techniques that are well-known in the realm of linear eigenproblems, such as locking to deflate converged eigenpairs.

Consider the $\mathbb{L}_1$ symmetric linearization $L(\lambda) = A - \lambda B$, with

$$A = \begin{bmatrix} 0 & -K \\ -K & -C \end{bmatrix}, \qquad B = \begin{bmatrix} -K & 0 \\ 0 & M \end{bmatrix}, \tag{1.2}$$

where K is assumed to be non-singular. The linear eigenproblem $L(\lambda)z = 0$ is equivalent to (1.1) in the sense that its eigenvalues coincide and their partial multiplicities are preserved [14]. The eigenvector of the linearization has the form $z = \left[\begin{smallmatrix} x \\ \lambda x \end{smallmatrix}\right]$, from which the eigenvector x of (1.1) can be retrieved. We note that A and B are $2n \times 2n$ symmetric matrices, but the pencil $A - \lambda B$ is not necessarily definite.

Assuming B is non-singular, i.e., M is non-singular, to solve the linear eigenproblem one can apply a standard Krylov method such as Arnoldi to matrix

$$S := B^{-1}A = \begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix}, \qquad (1.3)$$

to compute an orthonormal basis V of the Krylov subspace $\mathscr{K}_k(S, Ve_1)$ and an upper Hessenberg matrix $H = V^*SV$, from which Rayleigh-Ritz approximations to the eigenpairs can be extracted. Alternatively, instead of applying the Arnoldi algorithm directly on S, it is possible to exploit the block structure shown in (1.3) to perform the computation in a memory-efficient way, where the computed basis will consist of n-vectors rather than explicitly storing the whole basis V of length $2n$. This is the idea of Meerbergen's Q-Arnoldi method [18]. The outcome of the two methods may differ in finite-precision arithmetic under some circumstances, and Q-Arnoldi may even get unstable. Later, a stabilized reformulation called TOAR was proposed [26, 16].

Both A and B in (1.2) are symmetric, and we would like to take advantage of this fact. Potential benefits are a reduction of computational effort and the preservation of the symmetry in the projected problem, which may be a good thing from the numerical point of view. If the pencil $A - \lambda B$ was (semi-)definite, the way to go would be B-Lanczos, a reformulation of the Lanczos recurrence where the standard Hermitian inner product is replaced by the inner product induced by matrix B. In this way, since S is self-adjoint with respect to this inner product, symmetry is preserved throughout the computation and the projected matrix is real, symmetric and tridiagonal, with real eigenvalues. This scheme entails Gram-Schmidt B-orthogonalization to obtain basis vectors satisfying $V^*BV = I$. For the case of an indefinite pencil, we can adapt this scheme, but with important differences. Now we have a pseudo-inner product defined by B, where it is possible to find vectors $w \neq 0$ such that $w^*Bw \leq 0$. The pseudo-Lanczos method [21, 3, 15, 17] collects these negative (or positive) signs in a signature matrix $\Omega = \mathrm{diag}(\pm 1)$, in such a way that the computed Lanczos basis now satisfies $V^*BV = \Omega$, and the projected problem is a symmetric-indefinite pencil. We will see that this is just a particularization of non-symmetric Lanczos where the left basis is assumed to be $W = BV\Omega^{-1}$ and not computed explicitly. As in any flavour of non-symmetric Lanczos, pitfalls such as serious breakdown must be addressed.

In this paper, we build upon pseudo-Lanczos and cast it to the schemes of the Q-Arnoldi and TOAR methods mentioned above. The derivation of the new algorithms, that we call Q-Lanczos and STOAR, follows the original methods closely, but introduce the pseudo-inner product with B. We will also pay special attention to the solution of the projected problem, trying to exploit its symmetric-indefinite structure. Another important aspect to take into account in a practical implementation is restarting, that can be done with a Krylov-Schur scheme [25] or its equivalent for symmetric problems called thick restart [30]. We will show how to adapt this type of restart to the newly proposed methods.

Often, the sought-after eigenvalues are interior and it is convenient to apply a spectral transformation analog of shift-and-invert in the linear case. The transformed quadratic eigenproblem $(\mu^2 M_\sigma + \mu C_\sigma + K_\sigma)x = 0$, where

$$M_\sigma = Q(\sigma), \tag{1.4a}$$
$$C_\sigma = C + 2\sigma M, \tag{1.4b}$$
$$K_\sigma = M, \tag{1.4c}$$

has the same eigenvectors x and re-mapped eigenvalues $\mu = (\lambda - \sigma)^{-1}$, for a certain $\sigma \in \mathbb{R}$. In this case, (1.2) and (1.3) require M_σ and K_σ to be non-singular, i.e., M must be non-singular and σ must not be equal to an eigenvalue of (1.1).

We have implemented the new methods as full-fledged solvers within the SLEPc parallel library of eigensolvers [9]. In this framework, we are able to conduct a number of numerical experiments to compare them with solvers that disregard symmetry.

A related work is the use of symplectic Lanczos for the case of quadratic eigenvalue problems with Hamiltonian structure [5] or more generally the SHIRA method based on Arnoldi [19, 11]. Other authors have considered adapting other methods (LOBPCG) to the indefinite B case [13].

The rest of the paper is organized as follows. We start by giving some definitions that will be used throughout the paper and describing the methods we will use to solve small-sized symmetric indefinite eigenproblems (projected eigenproblems). Section 3 describes the pseudo-Lanczos method, including details about restart. Sections 4 and 5 present the new Q-Lanczos and STOAR methods, respectively. A few implementation details are discussed in section 6. Section 7 shows results of numerical experiments as well as parallel performance. We wrap up with some concluding remarks.

2 Pseudo-symmetric matrices

For any Hermitian matrix $M \in \mathbb{C}^{n\times n}$, we use the quadratic form given by

$$\langle x,y\rangle_M := y^* M x \quad \text{for} \quad x,y \in \mathbb{C}^n, \tag{2.1}$$

as the inner product defined by M, even when M is not positive definite and hence (2.1) is not a proper inner product. We refer to it as the pseudo (or indefinite) inner product. Also for Hermitian matrices, we call pseudo-norm defined by M the real function $\|x\|_M := \mathrm{sgn}(\langle x,x\rangle_M)\sqrt{|\langle x,x\rangle_M|}$, which is a norm if M is a Hermitian positive definite matrix. A vector u is said to be M-normalized if $|\,\|u\|_M\,| = 1$.

In this context, we say two vectors x and y, $x \neq y$, are M-orthogonal if $\langle x,y\rangle_M = 0$. More generally, the column vectors of a matrix $V = [v_1,\dots,v_k]$ are said to be mutually M-orthogonal if V^*MV is a non-singular diagonal matrix. When we want to stress the fact that, in addition to being M-orthogonal, the columns of V are M-normalized, we will say they are M-orthonormal. In this case, $\Omega := V^*MV$ is a signature matrix (a diagonal matrix whose diagonal elements are either 1 or -1), and we will equivalently say that the columns of V are (M,Ω)-orthogonal.

For a Hermitian matrix M we state that a matrix S is M-symmetric if it is self-adjoint with respect to the (pseudo) inner product defined by M, i.e., if $S^*M = MS$. In the particular case that M is a signature matrix, this condition is equivalent to S

being the product of a signature matrix (M) by a symmetric (or Hermitian) matrix. In this latter case, when we do not want to specify the involved signature matrix, we will just say that S is pseudo-symmetric.

2.1 Block diagonalization

The pseudo-Lanczos method described in §3 generates a projected eigenproblem whose matrix is real pseudo-symmetric. The restart technique we will see in §3.2 requires that this projected eigenproblem is solved preserving this structural property. Next we describe different ways we have considered to compute the solution of these small-sized eigenproblems.

Solving an eigenproblem defined by a pseudo-symmetric matrix $\Omega^{-1}T$, where Ω is a signature and T is symmetric, is equivalent to solving the generalized eigenproblem defined by the symmetric-indefinite pencil (T,Ω). We are interested in computing real matrices Q, $\tilde{\Omega}$ and D such that D is a block diagonal matrix with maximum block size equal to two and $\tilde{\Omega}$ is a signature matrix satisfying

$$Q^*\Omega Q = \tilde{\Omega}, \tag{2.2a}$$

$$Q^*TQ = D. \tag{2.2b}$$

Relations (2.2) imply that Q is $(\Omega,\tilde{\Omega})$-orthogonal and D is symmetric. The relation

$$\Omega^{-1}TQ = Q\tilde{\Omega}^{-1}D \tag{2.3}$$

is also satisfied. We will refer to this decomposition as a real pseudo-symmetric diagonalization of the pencil (T,Ω).

To compute the matrices Q, $\tilde{\Omega}$ and D in (2.2) we have considered two approaches: (i) preserve the symmetric-indefinite structure of the matrix pair throughout the computation, or (ii) use standard non-symmetric methods and recover the symmetric-indefinite structure at the end.

Structure-preserving diagonalization We start by describing the structure-preserving method described in [29]. This approach is based on first reducing to tridiagonal form and then applying the HR iteration.

The tridiagonalization process is done by making zeros in the matrix in a similar way as in the symmetric case, although in order to preserve pseudo-symmetry, it is necessary that the used similarity transformations are Ω-orthonormal. So, in this case we use Householder reflectors combined with hyperbolic rotations,

$$H_r = \begin{bmatrix} c & s \\ s & c \end{bmatrix}, \qquad c^2 - s^2 = \pm 1. \tag{2.4}$$

These transformations are not unitary and can have arbitrarily high condition numbers that could result in inaccurate solutions. Also, it is not always possible to build a hyperbolic rotation, H_r, such that $H_r x = \alpha e_1$ for a nonzero vector, x, and hence the tridiagonalization process may suffer from breakdown. That is why, to minimize both effects, it is necessary that the tridiagonalization be done very carefully. Tisseur

[27] proposes a method to reduce a symmetric-diagonal matrix pair to tridiagonal-diagonal form by means of a combination of orthogonal and hyperbolic transformations, reducing the number of the latter as much as possible and controlling their condition number. To carry out the first step of reducing to tridiagonal pseudo-symmetric form, we have an implementation based on [27] that takes into account the particular structure of the matrix generated in the restarted Lanczos process (see §3.2),

$$T = \begin{bmatrix} D_1 & & & \Gamma_1 & & & \\ & \ddots & & \vdots & & & \\ & & D_m & \Gamma_m & & & \\ \Gamma_1^T & \cdots & \Gamma_m^T & \alpha_{p+1} & \ddots & & \\ & & & \ddots & \ddots & \beta_{k-1} \\ & & & & \beta_{k-1} & \alpha_k \end{bmatrix}, \tag{2.5}$$

where D_i, $i = 1,\ldots,m$, are 1×1 or 2×2 blocks, and Γ_i are vectors with dimension consistent with D_i. In order to minimize the number of required hyperbolic transformations, we are adapting [27] to be used only on the arrowhead part of (2.5).

The next step uses the HR method [29], a GR-type iteration (similar to the QR method) that maintains pseudo-symmetry along the computation. If no breakdown occurs, this method (together with the tridiagonalization process) computes matrices Q, $\tilde{\Omega}$ and D satisfying (2.2)-(2.3). To produce the $(\Omega,\tilde{\Omega})$-orthogonal transformation Q, the HR iteration uses hyperbolic rotations (2.4) in the *bulge-chasing* phase. That is why this method has risk of breakdown, and potential instability, similarly to the tridiagonalization process. No implementations of the HR method are available in libraries such as LAPACK, that is why we have included in our code a basic implementation based on [29].

Alternative diagonalization Now we consider an alternative procedure to obtain matrices Q, $\tilde{\Omega}$ and D of (2.2)-(2.3) from the eigendecomposition of the pair (T,Ω). First, such decomposition, $TY = \Omega Y\Theta$, is computed, for example, using the QR method. Then, by representing the real and imaginary parts of the complex eigenpairs separately, we obtain an equivalent real decomposition, $TY_r = \Omega Y_r\Theta_r$, in which Θ_r is block diagonal. Later, the obtained real vectors, Y_r, are Ω-orthogonalized. A general property of Hermitian indefinite pencils (A,B) is that if B is nonsingular and $B^{-1}A$ is diagonalizable, then there exists a complete set of eigenvectors, W, such that W^TBW is a signature matrix [3]. So, although the columns of Y_r associated with simple real eigenvalues of $\Omega^{-1}T$ are already a set of Ω-orthogonal vectors, and it would only be needed to perform Ω-orthogonalization between pairs associated with a complex eigenvalue (real and imaginary part), or between vectors associated with a multiple eigenvalue, for the sake of accuracy, we prefer to Ω-orthogonalize all vectors Y_r, and compute an HR decomposition $Y_r = QR$ such that $Q^*\Omega Q = \tilde{\Omega}$, and R is upper triangular. Finally, the diagonal blocks of D are obtained from $D = Q^*TQ = \tilde{\Omega}R\Theta_rR^{-1}$. We remark that in our implementation we just compute the diagonal blocks of D, hence implicitly zeroing the rest of elements that could be nonzero in finite precision arithmetic.

The HR decomposition can be carried out by the Gram-Schmidt process with the indefinite inner product defined by Ω, or, alternatively, working with a combination of Householder reflectors and hyperbolic rotations, similarly to the tridiagonalization described in [27]. This latter option is the one we use in our implementation, since it provides information about the condition number of the applied transformations that can be used to improve the overall numerical robustness in the following cases:

- In the case of complex conjugate eigenpairs, the order in which the two corresponding columns are orthogonalized may have an influence on the global accuracy. In order to minimize errors, we use the condition number of the transformations to decide about the order of orthogonalization.
- As we will see in §3.2, when restarting, the Krylov relation is truncated and therefore only a part of the computed decomposition will be kept. That is, for computed matrices of dimension $(p+q)$,

$$D = \begin{array}{c|cc} & p & q \\ \hline p & D_p & \\ q & & D_q \end{array}, \quad Q = \begin{bmatrix} Q_p & Q_q \end{bmatrix}, \text{ and } \quad \tilde{\Omega} = \begin{bmatrix} \tilde{\Omega}_p & \\ & \tilde{\Omega}_q \end{bmatrix},$$

  with $\tilde{\Omega}_p^{-1} D_p$ containing the most desirable eigenvalues of $\Omega^{-1}T$, only matrices D_p, Q_p and $\tilde{\Omega}_p$ will be used to update the Krylov relation. Hence, when computing the HR decomposition we perform a column permutation so that we move to the group, Q_q, to be eliminated those columns whose orthogonalization results in breakdown or in an ill-conditioned transformation (it is not necessary to maintain Ω-orthogonality in Q_q). Matrices $\tilde{\Omega}$ and D are permuted accordingly.

3 Lanczos for symmetric-indefinite pencils

The pseudo-Lanczos method [21] is a variant of Lanczos to solve symmetric (not necessarily definite) generalized eigenvalue problems, $Ax = \lambda Bx$, where A and B are real symmetric or complex Hermitian matrices. It uses the indefinite inner product defined by the Hermitian matrix B to preserve the symmetric structure in the projected eigenproblem that it generates. This method can be derived from non-symmetric Lanczos [8] applied to symmetric-indefinite pencils.

3.1 Pseudo-Lanczos method

Given a matrix $S \in \mathbb{C}^{n\times n}$ and a vector $v \in \mathbb{C}^n$, the Krylov subspace of order k associated with S and v is defined as $\mathcal{K}_k(S,v) := \operatorname{span}\{v, Sv, S^2v, \ldots, S^{k-1}v\}$. Non-symmetric Lanczos is an oblique projection method, for solving non-symmetric eigenvalue problems, that computes biorthogonal bases V_k and W_k of the Krylov subspaces $\mathcal{K}_k(S, V_k e_1)$ and $\mathcal{K}_k(S^*, W_k e_1)$, respectively, by using a two-sided Gram-Schmidt process for each step of recurrences

$$p_k = Sv_k - V_k W_k^* S v_k = Sv_k - \hat{\gamma}_{k-1} v_{k-1} - \hat{\alpha}_k v_k, \tag{3.1a}$$

$$q_k = S^* w_k - W_k V_k^* S^* w_k = S^* w_k - \bar{\hat{\beta}}_{k-1} w_{k-1} - \bar{\hat{\alpha}}_k w_k, \tag{3.1b}$$

where, $p_k = \hat{\beta}_k v_{k+1}$, $q_k = \bar{\hat{\gamma}}_k w_{k+1}$, $q_k^* p_k = \hat{\gamma}_k \hat{\beta}_k$ and $\hat{\alpha}_k = w_k^* S v_k$. The short recurrences (3.1) produce the oblique projection matrix of S onto the right Krylov subspace,

$$\hat{T}_k := W_k^* S V_k = \begin{bmatrix} \hat{\alpha}_1 & \hat{\gamma}_1 & & & \\ \hat{\beta}_1 & \hat{\alpha}_2 & \hat{\gamma}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \hat{\beta}_{k-2} & \hat{\alpha}_{k-1} & \hat{\gamma}_{k-1} \\ & & & \hat{\beta}_{k-1} & \hat{\alpha}_k \end{bmatrix},$$

and the decompositions

$$SV_k = V_k \hat{T}_k + \hat{\beta}_k v_{k+1} e_k^*,$$
$$S^* W_k = W_k \hat{T}_k^* + \bar{\hat{\gamma}}_k w_{k+1} e_k^*,$$

from which non-symmetric Lanczos provides approximations for both right and left eigenvectors.

The pseudo-Lanczos method is a particular form of non-symmetric Lanczos that arises in the case that S is B-symmetric, for example when $S = B^{-1}A$, or $S = (A - \sigma B)^{-1}B$ for some $\sigma \in \mathbb{R}$. In this case, if initial vectors v_1 and $w_1 := Bv_1(v_1^* B v_1)^{-1}$ are used, recurrence (3.1b) produces $W = BV\Omega^{-1}$, where $\Omega = V^*BV$ is diagonal. The biorthogonality of the bases V and W generated by non-symmetric Lanczos results in the B-orthogonality of vectors V generated by pseudo-Lanczos. Hence, this variant does not need to form w_k vectors explicitly, and it only computes right vectors with recurrence (3.1a).

Both symmetric and non-symmetric Lanczos (and pseudo-Lanczos) can hit a happy breakdown if a basis of an invariant subspace is computed (if and only if $Sv_k \in \operatorname{span}(V_k)$ or $S^* w_k \in \operatorname{span}(W_k)$ for some k when expanding Krylov subspaces). On the other hand, as a result of working with biorthogonal bases, non-symmetric Lanczos can also lead to a serious breakdown, which appears when Lanczos recurrences (3.1) yield non-null vectors, p_k and q_k satisfying $q_k^* p_k = 0$. Various strategies have been proposed to deal with serious breakdown in non-symmetric Lanczos, such as look-ahead variants [22], adaptive block [2] or implicit restart [23]. Next, we give details of a restarted version of the pseudo-Lanczos method that aims at reducing the effects of working with an indefinite inner product.

Algorithm 3.1 shows how pseudo-Lanczos is used to compute a basis V_k of Lanczos vectors, a symmetric tridiagonal matrix and a signature matrix,

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix} \quad \text{and} \quad \Omega_k = \begin{bmatrix} \omega_1 & & & \\ & \omega_2 & & \\ & & \ddots & \\ & & & \omega_k \end{bmatrix},$$

such that

$$SV_k = V_k \Omega_k^{-1} T_k + \beta_k \omega_{k+1}^{-1} v_{k+1} e_k^*, \tag{3.3}$$

where $\omega_i = \pm 1, i = 1, \ldots, k+1$, $V_k^* B V_k = \Omega_k$ and $V_k^* B v_{k+1} = 0$. Note that the non-symmetric tridiagonal matrix of the projected problem, $\hat{T}_k = \Omega_k^{-1} T_k$, generated by

Algorithm 3.1 Pseudo-Lanczos

```
Input: Hermitian matrices $A, B \in \mathbb{C}^{n\times n}$ defining $S$ as in (1.3), initial vector $v_1 \in \mathbb{C}^n$
Output: Matrices $T_k, \Omega_k \in \mathbb{C}^{k\times k}$, $V_{k+1} \in \mathbb{C}^{n\times(k+1)}$ and $\beta_k, \omega_{k+1} \in \mathbb{R}$ satisfying (3.3)
 1: $V_1 \leftarrow v_1 = v_1/\|v_1\|_B$
 2: $\Omega_1 \leftarrow \omega_1 = v_1^* B v_1$
 3: for $j = 1,2,\ldots,k$ do
 4:     $u = Sv_j$
 5:     $h = V_j^* Bu$
 6:     $u = u - V_j \Omega_j^{-1} h$
 7:     $\alpha_j = e_j^* h$
 8:     $v_{j+1} = u/\|u\|_B$
 9:     $\omega_{j+1} = v_{j+1}^* B v_{j+1}$
10:     $\beta_j = \omega_{j+1}\|u\|_B$
11:     $V_{j+1} \leftarrow [V_j \; v_{j+1}]$,   $\Omega_{j+1} \leftarrow \begin{bmatrix} \Omega_j & \\ & \omega_{j+1} \end{bmatrix}$
12: end for
```

Algorithm 3.1, is tridiagonal, pseudo-symmetric and real (even with A and B being complex). Note also that in Algorithm 3.1, Lanczos vectors are B-normalized so that ω_i is always ± 1. However, we keep the notation Ω_k^{-1}, because it is possible to use other normalizations so that Ω_k is no longer a signature matrix. In that case the tridiagonal matrix $\hat{T}_k$ would not be pseudo-symmetric, but the projected problem (Ω_k, T_k) would still be a symmetric-indefinite pair.

There are several aspects to consider when executing Algorithm 3.1 in finite precision arithmetic. First, in order to preserve B-orthogonality of vectors generated by the Lanczos short recurrence (3.1a), we opt for storing previously computed vectors and orthogonalizing against all of them (lines 5 and 6 of Algorithm 3.1). For more information about loss of orthogonality in Lanczos see, e.g., [20]. Second, for efficiency reasons, Algorithm 3.1 (lines 5 and 6) uses the classical Gram-Schmidt procedure instead of modified Gram-Schmidt, but in a practical implementation this process must be repeated twice to enhance numerical stability. See [10] for details on parallel implementation of iterated Gram-Schmidt. Finally, although full reorthogonalization is performed, all Gram-Schmidt coefficients are discarded except the diagonal element α_j, as is done in practical Lanczos implementations, and similarly we only store the diagonal elements of Ω_k, even if $V_k^* B V_k$ is not really diagonal in finite precision arithmetic (furthermore, ω_j's are rounded to ± 1 when Lanczos vectors are B-normalized).

The Rayleigh-Ritz procedure can be used to compute approximations to the eigenpairs $(\tilde{\theta}_i, \tilde{x}_i)$ in a similar way as symmetric Lanczos. The eigenpairs $(\tilde{\theta}_i, y_i)$ from the projected problem $\Omega_k^{-1} T_k$ supply the approximations on $\mathrm{span}(V_k)$, verifying $S\tilde{x}_i - \tilde{\theta}_i \tilde{x}_i \perp \mathrm{span}(W_k)$ (Petrov-Galerkin condition), for $W_k := BV_k\Omega_k^{-1}$. Multiplying decomposition (3.3) by y_i on the right, it gives the residual for each right Petrov vector computed,

$$r_i = SV_k y_i - \tilde{\theta}_i V_k y_i = \beta_k \omega_{k+1}^{-1} v_{k+1} e_k^* y_i. \tag{3.4}$$

We consider a computed approximation as converged if the norm of the residual, (3.4), is smaller than a certain tolerance $\mathtt{tol}_{\mathrm{conv}}$,

$$\|r_i\| = |\beta_k \omega_{k+1}^{-1} e_k^* y_i| \|v_{k+1}\| < \mathtt{tol}_{\mathrm{conv}}. \tag{3.5}$$

The pseudo-Lanczos recurrence can suffer from breakdown when it produces a vector, u, with B-norm equal to zero. This breakdown is a serious one if u is not identically zero. In finite precision, exact breakdowns will not likely appear, but near breakdowns can introduce instability in the process, so it is necessary to detect them and take care of them too. To this end, at each iteration of Algorithm 3.1 the angle between vector u computed in line 6 and Bu must be checked to test if they are nearly orthogonal, and it is considered a breakdown if its cosine is smaller than a fixed tolerance ($\texttt{tol}_{\text{break}}$),

$$\cos\angle(u,Bu) = \frac{|u^*Bu|}{\|u\|\|Bu\|} < \texttt{tol}_{\text{break}}.$$

When a serious breakdown is detected it is not possible to extend the factorization with Algorithm 3.1. One possible workaround is to force a restart, either explicit with a new initial vector, throwing away the computed vectors, or implicit such as the one proposed in [23]. In this situation, we opt for performing an implicit restart in the way it is explained below. This strategy works correctly, as we have checked it with a synthetic case (we have not seen serious breakdowns in any of our practical runs).

3.2 Thick restart in pseudo-Lanczos

It is well known that convergence in Lanczos methods can be slow, resulting in an increasingly high cost for each iteration in the case of full reorthogonalization. That is why methods of this type are often restarted. To control the storage requirements and the orthogonalization cost, we propose to adapt the thick-restart technique [30], or the more general Krylov-Schur algorithm [25], to the pseudo-Lanczos method described above.

Implicit restart procedures aim at reducing the size of the working basis while preserving the most relevant information. Thick-restart techniques are based on Krylov relations [25],

$$SU_k = U_kC_k + u_{k+1}b^*, \tag{3.6}$$

which generalize the Arnoldi and Lanczos relations in the sense that C_k and b have no particular structure. Bases U_k in Krylov relations are different from those in Lanczos or Arnoldi, but also span Krylov subspaces.

We will consider that a Krylov relation (3.6) is symmetric if S is symmetric with respect to a Hermitian matrix B, and C_k is symmetric with respect to $\Omega_k := U_k^*BU_k$. For this type of Krylov relations there exists another implicit Krylov relation that holds for $W_k := BU_k\Omega_k^{-1}$. Indeed, if we multiply (3.6) on the left by B and on the right by Ω_k^{-1}, due to the symmetry of S and C_k, we obtain

$$S^*W_k = W_kC_k^* + w_{k+1}d^*,$$

where $w_{k+1} = Bu_{k+1}$ and $d = \Omega_k^{-1}b$. Thus, the columns of U_k and W_k generate Krylov subspaces for S and S^*, respectively.

To restart the pseudo-Lanczos method we consider symmetric Krylov relations in which U_k has B-orthogonal columns and the Rayleigh quotient C_k is real and has the form $C_k = \Omega_k^{-1}T_k$ with T_k symmetric and $\Omega_k := U_k^* B U_k$ (diagonal),

$$SU_k = U_k \Omega_k^{-1} T_k + u_{k+1} b^*. \tag{3.7}$$

We will refer to this type of relations as pseudo-orthogonal Krylov relations. One instance of them is the relation (3.3), generated by Algorithm 3.1. When multiplying a pseudo-orthogonal Krylov relation (3.7) on the right by an Ω_k-orthogonal matrix, Q_k, we obtain another pseudo-orthogonal Krylov relation,

$$S\tilde{U}_k = \tilde{U}_k \tilde{\Omega}_k^{-1} \tilde{T}_k + u_{k+1} \tilde{b}^*, \tag{3.8}$$

where $\tilde{U}_k = U_k Q_k$, $\tilde{\Omega}_k = Q_k^* \Omega_k Q_k$, $\tilde{T}_k = Q_k^* T_k Q_k$, and $\tilde{b} = Q_k^* b$. In this situation we say that both relations (3.7) and (3.8) are equivalent.

The strategy for restarting a pseudo-orthogonal Krylov relation (3.7), is to compute an equivalent one (3.8) such that $\tilde{\Omega}_k^{-1}\tilde{T}_k$ has a block diagonal form

$$\tilde{\Omega}_k^{-1}\tilde{T}_k = \begin{array}{c c} & \begin{array}{cc} p & k-p \end{array} \\ \begin{array}{c} p \\ k-p \end{array} & \begin{bmatrix} D_{11} & \\ & D_{22} \end{bmatrix} \end{array}.$$

The idea is to keep the decomposition corresponding to the leading part only, which is a pseudo-orthogonal Krylov relation of order p, and then use Algorithm 3.1 to extend it again to order k.

Assuming Algorithm 3.1 has generated a relation (3.3), thick restart for pseudo-Lanczos proceeds as follows. First, we compute matrices Q_k, $\tilde{\Omega}_k$ and D_k of a real pseudo-symmetric diagonalization (2.3) of the symmetric-indefinite pair (T_k, Ω_k), in such a way that the sought-after eigenvalues (or the associated blocks) remain in the leading principal submatrix of D_k of order p. Then, post-multiplying the decomposition (3.3) by $Q_k = [Q_p, Q_{p+1:k}]$ results in the equivalent relation

$$SV_kQ_k = V_kQ_k(Q_k^*\Omega_kQ_k)^{-1}Q_k^*T_kQ_k + \beta_k\omega_{k+1}^{-1}v_{k+1}e_k^*Q_k,$$

that can be truncated (for any p that does not break a 2×2 block), resulting in

$$S\tilde{V}_p = \tilde{V}_p\tilde{\Omega}_p^{-1}D_p + v_{k+1}b^*, \tag{3.9}$$

where $b = \beta_k\omega_{k+1}^{-1}Q_p^*e_k$ and $\tilde{V}_p = V_kQ_p$.

Note that this update (together with the indefinite orthogonalization in pseudo-Lanczos) is one of the sources of instability in the generated Krylov relation. An update with a very high condition number must be avoided, and this is the aim of discarding the last part of the factorization when using the alternative method for the pseudo-symmetric diagonalization as discussed in §2.

The new equation (3.9) satisfies the definition of pseudo-orthogonal Krylov relation, and hence it is possible to take it as a starting point to resume the pseudo-Lanczos iteration. The new iteration will extend the basis $\tilde{V}_p$ as well as the Rayleigh quotient matrix, which will get the shape characteristic of this type of restart with an arrowhead part plus a tridiagonal,

$$\tilde{T}_k = \begin{bmatrix} D_p & b e_1^* \\ e_1 b^* & \hat{T}_k \end{bmatrix}, \quad \text{with} \quad \hat{T}_k = \begin{bmatrix} \alpha_{p+1} & \beta_{p+1} & & \\ \beta_{p+1} & \alpha_{p+2} & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}. \tag{3.10}$$

The thick-restart technique allows reducing the dimension of the Krylov subspace purging unwanted directions from it. Another chance for reducing the computational effort is when the criterion (3.5) indicates that certain Petrov pairs are sufficiently converged. Then, they can be deflated and locked so that from that point on they remain unchanged and the Rayleigh quotient gets the form,

$$C_k = \Omega_k^{-1} \begin{bmatrix} D_\ell & 0 \\ 0 & G \end{bmatrix},$$

where D_ℓ is symmetric block diagonal (with block size 2 at most) and G is symmetric. This results in an effective size reduction of the eigendecomposition to be computed (only needed for G). Nevertheless, to maintain B-orthogonality of vectors computed by Algorithm 3.1, each new one must be orthogonalized also against locked vectors.

4 Q-Lanczos

The Quadratic Arnoldi method, or Q-Arnoldi [18], solves the quadratic eigenvalue problem (1.1) by exploiting the structure of the Krylov vectors when applying the Arnoldi method to the linearization

$$A - \lambda B = \begin{bmatrix} 0 & N \\ -K & -C \end{bmatrix} - \lambda \begin{bmatrix} N & 0 \\ 0 & M \end{bmatrix}, \tag{4.1}$$

where $N = I$ or $N = -K$. Q-Arnoldi computes the Arnoldi relation for $S = B^{-1}A$,

$$\begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} V_k^0 \\ V_k^1 \end{bmatrix} = \begin{bmatrix} V_k^0 & v^0 \\ V_k^1 & v^1 \end{bmatrix} \underline{H}_k, \tag{4.2}$$

where $\underline{H}_k$ is the extended $(k+1) \times k$ upper Hessenberg matrix. The goal is to represent the Arnoldi decomposition (4.2) without storing the full Arnoldi basis $\begin{bmatrix} V_k^0 \\ V_k^1 \end{bmatrix}$, but only the upper part V_k^0. To this end, Q-Arnoldi takes advantage of the relation,

$$V_k^1 = [V_k^0, v^0] \underline{H}_k, \tag{4.3}$$

that comes from the first block row of (4.2). Algorithm 4.1 shows how Q-Arnoldi performs the Arnoldi iteration to compute vectors V_k^0 and the Rayleigh quotient H_k. In the sequel, V_j^0 and V_j^1 denote, for $j \in \mathbb{N}$, the top and bottom halves of the Arnoldi basis V_j, respectively. In general, when a vector x is given, we use x^0 and x^1 to denote the two halves of x.

Algorithm 4.1 Q-Arnoldi

```
Input: Hermitian $M, C, K \in \mathbb{C}^{n\times n}$ defining the QEP, initial vectors $w^0$ and $w^1 \in \mathbb{C}^n$
Output: Matrix $\underline{H}_k \in \mathbb{C}^{(k+1)\times k}$ and vectors $V_k^0 \in \mathbb{C}^{n\times k}$ and $v^0, v^1 \in \mathbb{C}^n$ satisfying (4.2) for $V_k^1 = V_{k+1}\underline{H}_k$
1: /* Normalize initial vector $w$ */
   $\beta = \sqrt{\|w^0\|^2 + \|w^1\|^2}$
   $v^0 = w^0/\beta, \quad v^1 = w^1/\beta,$        /* $v = \begin{bmatrix} v^0 \\ v^1 \end{bmatrix} = \frac{w}{\|w\|}$ */
2: $V_1^0 \leftarrow \left[v^0\right]$
3: for $j = 1, 2, \ldots, k$ do
4:     /* Expand $w = Sv$ */
       $w^0 = v^1$
       $w^1 = -M^{-1}(Kv^0 + Cv^1)$
5:     /* Gram-Schmidt coefficients $h_j = V_j^* w$ */
       $t = (V_{j-1}^1)^* w^1 = \underline{H}_{j-1}^* \left[V_{j-1}^0 \; v^0\right]^* w^1$
       $h_j = \begin{bmatrix} V_{j-1}^0 & v^0 \\ V_{j-1}^1 & v^1 \end{bmatrix}^* w = \begin{bmatrix} (V_{j-1}^0)^* w^0 + t \\ v^* w \end{bmatrix}$
6:     /* Gram-Schmidt update $\tilde{w} = w - V_j h_j$ */
       $\tilde{w}^0 = w^0 - \left[V_{j-1}^0 \; v^0\right] h_j$
       $\tilde{w}^1 = w^1 - \left[\left[V_{j-1}^0 \; v^0\right] \underline{H}_{j-1} \; v^1\right] h_j$
7:     /* Normalize */
       $h_{j+1,j} = \sqrt{\|\tilde{w}^0\|^2 + \|\tilde{w}^1\|^2}$
       $v^0 = \tilde{w}^0/h_{j+1,j}, \quad v^1 = \tilde{w}^1/h_{j+1,j}$
8:     /* Append new Arnoldi vector (top part)*/
       $V_{j+1}^0 \leftarrow [V_j^0 \; v^0]$
9: end for
```

To solve a symmetric quadratic eigenvalue problem (1.1) by symmetric linearization ($N = -K$ in (4.1)), the pseudo-Lanczos method can be applied to $S = B^{-1}A$ to produce a relation (4.2) in which H_k is a pseudo-symmetric tridiagonal matrix and the vectors V_k are B-orthogonal. In this case, it is also possible to use the relation (4.3) to reduce the storage requirements in the same way as Q-Arnoldi does. We refer to this adaptation to the symmetric quadratic eigenvalue problem as the Q-Lanczos method.

The Q-Lanczos method works in a similar way as Q-Arnoldi except that the former must take into account the pseudo inner product defined by B in the orthogonalization stages. Hence, to normalize vectors (at steps 1 and 7 of Algorithm 4.1), the Q-Lanczos algorithm uses the pseudo-norm defined by B, resulting in:

```
1: $\delta = \mathrm{sgn}(-w^{0*}Kw^0 + w^{1*}Mw^1)\sqrt{|-w^{0*}Kw^0 + w^{1*}Mw^1|}$     /* $\delta = \|w\|_B$ */
   $v^0 = w^0/\delta, \quad v^1 = w^1/\delta, \quad \omega_1 = \mathrm{sgn}(\delta)$     /* $\omega_1 = v^* B v$ */
7: $h_{j+1,j} = \|\tilde{w}\|_B, \quad v = \tilde{w}/h_{j+1,j}, \quad \omega_{j+1} = \mathrm{sgn}(\|\tilde{w}\|_B)$
```

Also, step 5 of Algorithm 4.1, that computes the Gram-Schmidt coefficients, is substituted in the Q-Lanczos algorithm by:

```
5: /* Gram-Schmidt coefficients $h_j = (V_j^* B V_j)^{-1} V_j^* B w$ */
   $t = \underline{H}_{j-1}^* \left[V_{j-1}^0 \; v^0\right]^* M w^1$
   $h_j = \Omega_j^{-1} \begin{bmatrix} -(V_{j-1}^0)^* K w^0 + t \\ -v^{0*} K w^0 + v^{1*} M w^1 \end{bmatrix}$
```

As in the pseudo-Lanczos iteration (Algorithm 3.1), and as a result of working with finite precision arithmetic, the generated matrix H_j is not tridiagonal. However, to recover the vectors V_j^1 when needed, Q-Lanczos, contrarily to pseudo-Lanczos, needs to store all Gram-Schmidt coefficients that result from full orthogonalization. After generating relation (4.2) by means of Q-Lanczos, elements outside the H_k tridiagonal are discarded and the pseudo-Lanczos process continues as explained in §3.

Restart in Q-Arnoldi and Q-Lanczos The Krylov-Schur restart can be easily adapted to Q-Arnoldi. Given the order-k decomposition (4.2), we compute the (sorted) Schur decomposition $H_kY_k = Y_kS_k$ and perform a similarity transformation with Y_k. The resulting decomposition is truncated to

$$\begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} \tilde{V}_p^0 \\ \tilde{V}_p^1 \end{bmatrix} = \begin{bmatrix} \tilde{V}_p^0 & v^0 \\ \tilde{V}_p^1 & v^1 \end{bmatrix} \underline{S}_p,$$

where $\tilde{V}_p = V_kY_p$, being Y_p the first p columns of Y_k, $\underline{S}_p = \begin{bmatrix} S_p \\ e_{k+1}^*\underline{H}_kY_p \end{bmatrix}$ and S_p the leading $p \times p$ submatrix of S_k containing the wanted eigenvalues. It is possible to extend this compressed factorization to order k again with the Q-Arnoldi algorithm, because a relation analog to (4.3) also holds,

$$\tilde{V}_p^1 = \begin{bmatrix} \tilde{V}_p^0 & v^0 \end{bmatrix} \underline{S}_p. \tag{4.4}$$

In Q-Lanczos, the restarting procedure is very similar, replacing the Schur decomposition by the real pseudo-symmetric diagonalization described in §2.

5 STOAR

As already pointed out by Meerbergen [18], the Q-Arnoldi method is not numerically stable. The instability may appear in the case that $\|H_k\|$ becomes large. This is often the case when a transformed QEP (1.4) is being solved with a shift σ close to an eigenvalue. The source of the problem comes from representing the vectors V_k^1 in (4.3) with two matrices, $\begin{bmatrix} V_k^0, v^0 \end{bmatrix}$ and $\underline{H}_k$, whose columns are not orthogonal.

The main idea of TOAR (Two-level Orthogonal Arnoldi) [26, 16, 7] is to compute an orthonormal basis, U_{k+1}, of span$([V_k^0, V_k^1, v^0])$, from which to recover both V_k^0 and V_k^1 when needed. The Krylov basis is represented as the product of two matrices with orthogonal columns,

$$V_k = \begin{bmatrix} U_{k+1} & \\ & U_{k+1} \end{bmatrix} \begin{bmatrix} G_k^0 \\ G_k^1 \end{bmatrix}, \tag{5.1}$$

where G_k^0 and G_k^1 are the U_{k+1} coordinates of V_k^0 and V_k^1, respectively. TOAR rewrites the Arnoldi relations using this representation of the Krylov basis,

$$\begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} U_{k+1}G_k^0 \\ U_{k+1}G_k^1 \end{bmatrix} = \begin{bmatrix} U_{k+2}G_{k+1}^0 \\ U_{k+2}G_{k+1}^1 \end{bmatrix} \underline{H}_k. \tag{5.2}$$

Algorithm 5.1 TOAR

```
Input: Hermitian $M, C, K \in \mathbb{C}^{n\times n}$ defining the QEP, initial vectors $w^0$ and $w^1 \in \mathbb{C}^n$
Output: $\underline{H}_k \in \mathbb{C}^{(k+1)\times k}$, $U_{k+2} \in \mathbb{C}^{n\times(k+2)}$, $G^0_{k+1}, G^1_{k+1} \in \mathbb{C}^{(k+2)\times(k+1)}$ satisfying (5.2)
 1: $u_1 = w^0/\|w^0\|, \quad g^0 = \begin{bmatrix} \|w^0\| \\ 0 \end{bmatrix}$
    $\tilde{u} = w^1 - u_1 u_1^* w_1, \quad u_2 = \tilde{u}/\|\tilde{u}\|, \quad g^1 = \begin{bmatrix} u_1^* w_1 \\ \|\tilde{u}\| \end{bmatrix}$
    $U_2 \leftarrow [u_1\ u_2]$
 2: $\beta = \|g\|, \quad g^0_1 = g^0/\beta, \quad g^1_1 = g^1/\beta, \quad G_1 \leftarrow \begin{bmatrix} g^0_1 \\ g^1_1 \end{bmatrix}$
 3: for $j = 1,2,\dots,k$ do
 4:     /* Expand $w = Sv_j$ */
        $w^1 = -M^{-1}(KU_{j+1}g^0_j + CU_{j+1}g^1_j)$
 5:     /* Expand $U_{j+1}$ */
        $g^0 = \begin{bmatrix} g^1_j \\ 0 \end{bmatrix}$            /* $w^0 = v^1_j$ */
        $\hat{w} = U^*_{j+1}w^1, \quad \tilde{u} = w^1 - U_{j+1}\hat{w}, \quad \alpha = \|\tilde{u}\|$
        $u_{j+2} = \tilde{u}/\alpha, \quad g^1 = \begin{bmatrix} \hat{w} \\ \alpha \end{bmatrix}$
 6:     /* Gram-Schmidt coefficients */
        $h_j = G^*_j \begin{bmatrix} g^1_j \\ \hat{w} \end{bmatrix}$
 7:     /* Gram-Schmidt update */
        $y^0 = g^1_j - G^0_j h_j, \quad \tilde{g}^0 = \begin{bmatrix} y^0 \\ 0 \end{bmatrix}$
        $y^1 = \hat{w} - G^1_j h_j, \quad \tilde{g}^1 = \begin{bmatrix} y^1 \\ \alpha \end{bmatrix}, \quad \tilde{g} = \begin{bmatrix} \tilde{g}^0 \\ \tilde{g}^1 \end{bmatrix}$
 8:     /* Normalize with $\|\tilde{w}\| = \|\tilde{g}\|$ */
        $h_{j+1,j} = \|\tilde{g}\|, \quad g_{j+1} = \tilde{g}/h_{j+1,j}$
 9:     /* Append new Arnoldi vector */
        $U_{j+2} \leftarrow [U_{j+1}\ u_{j+2}]$
        $\underline{G}^0_j = \begin{bmatrix} G^0_j \\ 0 \end{bmatrix}, \quad \underline{G}^1_j = \begin{bmatrix} G^1_j \\ 0 \end{bmatrix}, \quad G_{j+1} \leftarrow \begin{bmatrix} \underline{G}^0_j & g^0_{j+1} \\ \underline{G}^1_j & g^1_{j+1} \end{bmatrix}$
10: end for
```

Algorithm 5.1 shows the TOAR version of the Arnoldi iteration. Given an initial vector w, the algorithm starts by computing a representation, in the form (5.1), of $w/\|w\|$. For this, step 1 uses the Gram-Schmidt procedure to compute orthonormal vectors u_1, u_2 and coefficients g^0, g^1 such that $w^i = [u_1, u_2]g^i$, $i = 1,2$. In step 2, w is normalized by normalizing g, since $\|g\| = \|w\|$. For $j = 1,\dots,k$ the Arnoldi iteration continues expanding the Krylov subspace, applying the operator matrix to the last Arnoldi vector computed, which is expressed in the form (5.1). In step 5, the bottom part of the new computed vector, w^1, is used to expand U_{j+1} to obtain a basis of $\operatorname{span}([V^0_j, V^1_j, w^0, w^1])$. The new basis vector $u_{j+2} \in U^{\perp}_{j+1}$ is obtained via Gram-Schmidt orthogonalization, together with $g = \begin{bmatrix} g^0 \\ g^1 \end{bmatrix}$ so that $w = \begin{bmatrix} [U_{j+1}\ u_{j+2}]g^0 \\ [U_{j+1}\ u_{j+2}]g^1 \end{bmatrix}$. Note that u_{j+2} cannot be computed if $\alpha = 0$, which implies that $\tilde{u} = 0$ or, equivalently, $w^1 \in \operatorname{span}(U_{j+1})$. In this case, only the matrix G_j will be extended and not the basis U_{j+1}.

The Gram-Schmidt coefficients are computed in step 6, taking into account that

$$h_j = V^*_j w = \begin{bmatrix} G^0_j \\ G^1_j \end{bmatrix}^* \begin{bmatrix} U^*_{j+1} & \\ & U^*_{j+1} \end{bmatrix} \begin{bmatrix} U_{j+1}g^1_j \\ U_{j+1}\hat{w} + u_{j+2}\alpha \end{bmatrix} = G^*_j \begin{bmatrix} g^1_j \\ \hat{w} \end{bmatrix}.$$

The orthogonalization of w can be expressed as

$$\tilde{w} = w - V_j h_j = \begin{bmatrix} [U_{j+1}\ u_{j+2}] & \\ & [U_{j+1}\ u_{j+2}] \end{bmatrix} \left(\begin{bmatrix} g^0 \\ g^1 \end{bmatrix} - \begin{bmatrix} \underline{G}_j^0 \\ \underline{G}_j^1 \end{bmatrix} h_j \right),$$

where $\underline{G}_j^i = \left[\begin{smallmatrix} G_j^i \\ 0 \end{smallmatrix}\right]$, for $i = 0,1$. That is why the Gram-Schmidt update (step 7) is carried out by orthogonalizing g, computed in step 5, against the columns of $\underline{G}_j$. Finally, by normalizing in step 8, we obtain a representation of the new Arnoldi vector in the form (5.1),

$$v_{j+1} = \frac{\tilde{w}}{\|\tilde{w}\|} = \begin{bmatrix} [U_{j+1}\ u_{j+2}] & \\ & [U_{j+1}\ u_{j+2}] \end{bmatrix} \frac{\tilde{g}}{\|\tilde{g}\|} = \begin{bmatrix} U_{j+2} & \\ & U_{j+2} \end{bmatrix} g_{j+1}.$$

The STOAR method is an adaptation of TOAR for solving the symmetric QEP. In a similar way as TOAR relates to Q-Arnoldi, STOAR expresses the Lanczos vectors in the form (5.1) and rewrites the Q-Lanczos algorithm using this decomposition. Since Lanczos vectors in Q-Lanczos are B-orthogonal, the matrices of the STOAR decomposition are not orthogonal. Expressing the condition of B-orthogonality using (5.1),

$$\Omega_k = V_j^* B V_j = G_j^* \begin{bmatrix} U_{j+1}^* & \\ & U_{j+1}^* \end{bmatrix} \begin{bmatrix} -K & \\ & M \end{bmatrix} \begin{bmatrix} U_{j+1} & \\ & U_{j+1} \end{bmatrix} G_j,$$

and defining the Hermitian matrix

$$\hat{B}_j = \begin{bmatrix} -\hat{K}_j & 0 \\ 0 & \hat{M}_j \end{bmatrix}, \quad \text{with} \quad \hat{K}_j = U_j^* K U_j, \quad \hat{M}_j = U_j^* M U_j,$$

we obtain that the columns of matrix G_j are $\hat{B}_{j+1}$-orthogonal.

Algorithm 5.2 (STOAR) has the same input and output as TOAR, but the projected matrix in (5.2) for STOAR is pseudo-symmetric. Thus, STOAR generates a pseudo-Lanczos relation of the form

$$\begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} U_{k+1} G_k^0 \\ U_{k+1} G_k^1 \end{bmatrix} = \begin{bmatrix} U_{k+2} G_{k+1}^0 \\ U_{k+2} G_{k+1}^1 \end{bmatrix} \underline{\hat{T}}_k, \tag{5.3}$$

where $\hat{T}_k$ is tridiagonal pseudo-symmetric, U_{k+2} has orthogonal columns and G_{k+1} has $\hat{B}_{k+2}$-orthogonal columns. To generate the columns of U_j, both TOAR and STOAR use a Gram-Schmidt process (steps 1 and 6 of Algorithms 5.1 and 5.2). The B-normalization of a vector in the form $\tilde{w} = \left[\begin{smallmatrix} U_j & \\ & U_j \end{smallmatrix}\right] g$ (steps 2 and 8 of Algorithm 5.2), is performed in STOAR by $\hat{B}_j$-normalizing g, using the fact that $\|\tilde{w}\|_B = \|g\|_{\hat{B}_j}$.

The Gram-Schmidt coefficients are computed, in step 6 of Algorithm 5.2, as

$$\begin{aligned} h = \Omega_j^{-1} V_j^* B w &= \Omega_j^{-1} \begin{bmatrix} G_j^0 \\ G_j^1 \end{bmatrix}^* \begin{bmatrix} U_{j+1}^* & \\ & U_{j+1}^* \end{bmatrix} \begin{bmatrix} -K U_{j+1} g_j^1 \\ M(U_{j+1}\hat{w} + u_{j+2}\alpha) \end{bmatrix} \\ &= \Omega_j^{-1} \begin{bmatrix} G_j^0 \\ \underline{0} \\ G_j^1 \\ 0 \end{bmatrix}^* \begin{bmatrix} U_{j+2}^* & \\ & U_{j+2}^* \end{bmatrix} \begin{bmatrix} -K U_{j+2} \left[\begin{smallmatrix} g_j^1 \\ 0 \end{smallmatrix}\right] \\ M U_{j+2} \left[\begin{smallmatrix} \hat{w} \\ \alpha \end{smallmatrix}\right] \end{bmatrix}. \end{aligned}$$

Algorithm 5.2 STOAR

```
Input: Hermitian $M, C, K \in \mathbb{C}^{n\times n}$ defining the QEP, initial vectors $w^0$ and $w^1 \in \mathbb{C}^n$
Output: $\underline{\hat{T}}_k \in \mathbb{C}^{(k+1)\times k}$, $U_{k+2} \in \mathbb{C}^{n\times(k+2)}$, $G^0_{k+1}, G^1_{k+1} \in \mathbb{C}^{(k+2)\times(k+1)}$ satisfying (5.3)
 1: $u_1 = w^0/\|w^0\|, \quad g^0 = \begin{bmatrix} \|w^0\| \\ 0 \end{bmatrix}$
    $\hat{M}_1 = u_1^* M u_1, \quad \hat{K}_1 = u_1^* K u_1$        /* Initialize $\hat{M}_1, \hat{K}_1$ */
    $\tilde{u} = w^1 - u_1 u_1^* w^1, \quad u_2 = \tilde{u}/\|\tilde{u}\|, \quad g^1 = \begin{bmatrix} u_1^* w^1 \\ \|\tilde{u}\| \end{bmatrix}$
    $U_2 \leftarrow \begin{bmatrix} u_1 & u_2 \end{bmatrix}$
    /* Update $\hat{M}_2$ and $\hat{K}_2$ */
    $\hat{M}_2 \leftarrow \begin{bmatrix} \hat{M}_1 & u_2^* M U_1 \\ U_1^* M u_2 & u_2^* M u_2 \end{bmatrix}, \quad \hat{K}_2 \leftarrow \begin{bmatrix} \hat{K}_1 & u_2^* K U_1 \\ U_1^* K u_2 & u_2^* K u_2 \end{bmatrix}$
 2: $\beta = \mathrm{sgn}(-g^{0*}\hat{K}_2 g^0 + g^{1*}\hat{M}_2 g^1)\sqrt{-g^{0*}\hat{K}_2 g^0 + g^{1*}\hat{M}_2 g^1}$,        /* $\beta = \|g\|_{\hat{B}_2}$ */
    $g_1^0 = g^0/\beta, \quad g_1^1 = g^1/\beta, \quad G_1 \leftarrow \begin{bmatrix} g_1^0 \\ g_1^1 \end{bmatrix}$
    $\omega_1 = \mathrm{sgn}(\beta), \quad \beta = \omega_1^{-1}\beta, \quad \Omega_1 \leftarrow \omega_1$        /* $\omega_1 = g_1^* \hat{B}_2 g_1$ */
 3: for $j = 1, 2, \ldots, k$ do
 4:     /* Expand $w = S v_j$ */
        $w^1 = -M^{-1}(K U_{j+1} g_j^0 + C U_{j+1} g_j^1)$
 5:     /* Expand $U_{j+1}$ */
        $g^0 = \begin{bmatrix} g_j^1 \\ 0 \end{bmatrix}$        /* $w^0 = v_j^1$ */
        $\hat{w} = U_{j+1}^* w^1, \quad \tilde{u} = w^1 - U_{j+1}\hat{w}, \quad \alpha = \|\tilde{u}\|$
        $u_{j+2} = \tilde{u}/\alpha, \quad g^1 = \begin{bmatrix} \hat{w} \\ \alpha \end{bmatrix}$
        /* Update $\hat{M}_{j+2}$ and $\hat{K}_{j+2}$ */
        $\hat{M}_{j+2} \leftarrow \begin{bmatrix} \hat{M}_{j+1} & u_{j+2}^* M U_{j+1} \\ U_{j+1}^* M u_{j+2} & u_{j+2}^* M u_{j+2} \end{bmatrix}, \quad \hat{K}_{j+2} \leftarrow \begin{bmatrix} \hat{K}_{j+1} & u_{j+2}^* K U_{j+1} \\ U_{j+1}^* K u_{j+2} & u_{j+2}^* K u_{j+2} \end{bmatrix}$
 6:     /* Gram-Schmidt coefficients */
        $\underline{G}_j^0 = \begin{bmatrix} G_j^0 \\ 0 \end{bmatrix}, \quad \underline{G}_j^1 = \begin{bmatrix} G_j^1 \\ 0 \end{bmatrix}$,
        $h = \Omega_j^{-1} \underline{G}_j^* \hat{B}_{j+2} \begin{bmatrix} g^0 \\ g^1 \end{bmatrix}$
        $\alpha_j = e_j^* \Omega_j h$
 7:     /* Gram-Schmidt update */
        $\tilde{g}^i = g^i - \underline{G}_j h, \quad i = 0, 1$
 8:     /* Normalize with $\|\tilde{w}\|_B = \|\tilde{g}\|_{\hat{B}_{j+2}}$ */
        $\beta_j = \|\tilde{g}\|_{\hat{B}_{j+2}}, \quad g_{j+1} = \tilde{g}/\beta_j, \quad \omega_{j+1} = \mathrm{sgn}(\beta_j)$
        $\beta_j = \omega_{j+1}\beta_j, \quad \Omega_{j+1} = \begin{bmatrix} \Omega_j & \\ & \omega_{j+1} \end{bmatrix}$
 9:     /* Append new Arnoldi vector */
        $U_{j+2} \leftarrow \begin{bmatrix} U_{j+1} & u_{j+2} \end{bmatrix}$
        $G_{j+1} \leftarrow \begin{bmatrix} \underline{G}_j^0 & g_{j+1}^0 \\ \underline{G}_j^1 & g_{j+1}^1 \end{bmatrix}$
10: end for
```

There are no relevant differences between TOAR and STOAR in the rest of stages.

As in the case of TOAR, happy breakdown occurs when a null vector $\tilde{g}$ is obtained at step 8. A non-null $\tilde{g}$ giving a zero value of β_j in step 8 indicates that a serious breakdown has taken place in the pseudo-Lanczos process.

Restart in TOAR and STOAR After k steps, the TOAR method computes an Arnoldi decomposition (5.2), that can be compressed to order $p < k$ and then restarted [12, 7]. For this, (5.2) is reduced to a Krylov-Schur decomposition by means of the similarity transformation defined by Y_k, from the (sorted) Schur decomposition of H_k, $H_k Y_k =$

$Y_k S_k$. This decomposition can then be truncated to order p,

$$\begin{bmatrix} 0 & I \\ -M^{-1}K & -M^{-1}C \end{bmatrix} \begin{bmatrix} U_{k+1}\tilde{G}_p^0 \\ U_{k+1}\tilde{G}_p^1 \end{bmatrix} = \begin{bmatrix} U_{k+1}\tilde{G}_p^0 \\ U_{k+1}\tilde{G}_p^1 \end{bmatrix} S_p + \begin{bmatrix} U_{k+2}g_{k+1}^0 \\ U_{k+2}g_{k+1}^1 \end{bmatrix} b^*,$$

where $\tilde{G}_p = G_k Y_p$ (Y_p denotes the first p columns of Y_k), S_p the leading $p \times p$ submatrix of S_k containing the wanted eigenvalues, and $b = Y_p^* \underline{H}_k^* e_{k+1}$. The next step is to reduce the number of columns of U_{k+1} (and rows of $\tilde{G}_p$), otherwise there is no effective reduction of memory needs. For this, we use the fact that $\operatorname{rank}([\underline{\tilde{G}}_p^0, g_{k+1}^0, \underline{\tilde{G}}_p^1, g_{k+1}^1]) = \operatorname{rank}([\tilde{V}_p^0, v_{k+1}^0, \tilde{V}_p^1, v_{k+1}^1]) = p+2$ (at most), since $\tilde{V}_p^1 = [\tilde{V}_p^0, v_{k+1}^0] S_p$ in a similar way as was shown in (4.4) for Q-Arnoldi. We then compute the compact singular value decomposition

$$\begin{bmatrix} \underline{\tilde{G}}_p^0 & g_{k+1}^0 & \underline{\tilde{G}}_p^1 & g_{k+1}^1 \end{bmatrix} = \check{U}\check{\Sigma}\check{V}^*, \tag{5.4}$$

where $\check{\Sigma}$ is a $(p+2) \times (p+2)$ diagonal matrix of singular values, and $\check{U}$, $\check{V}$ have dimensions $(k+2) \times (p+2)$ and $(2p+2) \times (p+2)$, respectively.

This decomposition allows us to express the updated Arnoldi vectors in the form (5.1),

$$\begin{bmatrix} \tilde{V}_p^i & v_{k+1}^i \end{bmatrix} = U_{k+2} \begin{bmatrix} \underline{\tilde{G}}_p^i & g_{k+1}^i \end{bmatrix} = U_{k+2}\check{U}\check{\Sigma}\check{V}^{i^*}, \quad \text{for } i = 0, 1,$$

where we have written $\check{V}^* = [\check{V}^{0^*}, \check{V}^{1^*}]$. Updating $U_{p+2} \leftarrow U_{k+2}\check{U}$ and $G_{p+1}^i \leftarrow \check{\Sigma}\check{V}^{i^*}$, for $i = 0, 1$, we obtain the representation for $[\tilde{V}_p, v_{k+1}]$ we are looking for.

6 Implementation Details

We have implemented all the methods described in this paper as eigensolvers in the SLEPc library [9]. SLEPc provides a collection of parallel solvers for computing a few eigenpairs of large scale eigenvalue problems, both linear and nonlinear, in either real or complex arithmetic. Our solvers for symmetric QEP's are available in SLEPc 3.6, except Q-Lanczos that was taken out of the release version due to its poorer performance (see section 7).

Matrix inverses are not computed explicitly. Instead, the eigensolvers compute an LDL^T factorization of M (or $A - \sigma B$) and perform triangular solves whenever the inverse is required. In our implementation, we delegate this task to the underlying PETSc library [4], which provides direct linear solvers, both sequential and parallel (the latter via third-party libraries such as MUMPS).

For measuring the quality of the computed eigenpairs, e.g. in the experiments of section 7, we use the relative backward error [28],

$$\eta_Q(x,\lambda) = \frac{\|Q(\lambda)x\|_2}{(|\lambda^2|\|M\|_2 + |\lambda|\|C\|_2 + \|K\|_2)\,\|x\|_2}. \tag{6.1}$$

In the implementation, we replace the 2-norms in the above expression by ∞-norms, which can be easily computed for a parallel matrix distributed by rows.

On the other hand, during the iteration, the quantities used in the convergence criterion are based on the residual of the linearization, that is readily available during the pseudo-Lanczos iteration, (3.4). By default, we check $\|r_i\|/|\tilde{\theta}_i| < \texttt{tol}_{\text{conv}}$.

As discussed in previous sections, all methods that use an indefinite inner product can potentially suffer from instability, either due to the pseudo-Lanczos process or the resolution of the symmetric-indefinite projected problem. Although there is no known way of making these methods as stable as the methods based on orthogonal transformations, we have noticed that the first immediate consequence of instability is the loss of symmetry, that is, that the computed Krylov relation (3.7) is no longer symmetric. In this way, to avoid returning wrong results, we have devised a simple mechanism that tracks symmetry of matrix T_k at each pseudo-Lanczos step, and exits the run (keeping the eigenpairs converged so far) whenever $\|T_k - T_k^*\|_F/\|T_k\|_F$ is larger than a given tolerance (this symmetry test is done before the Gram-Schmidt coefficients corresponding to values outside the tridiagonal are discarded).

7 Numerical Experiments

In this section, we conduct a number of numerical tests in order to analyze the behaviour of the methods when computing a few eigenpairs of symmetric (Hermitian) QEP's from the NLEVP collection [6]. All problems are solved in real arithmetic, except *sign1* that has complex coefficient matrices. Most of our experiments use the shift-and-invert transformation (1.4) to compute eigenvalues closest to a given σ (except *sign1* that computes eigenvalues with real part in $[-0.9, 0.9]$). We remark that, in the case of Krylov-Schur and pseudo-Lanczos, the spectral transformation is performed on the linearized problem, i.e., the iteration operates on $(A - \sigma B)^{-1}Bx = \theta x$, where A and B are the matrices of the linearization (1.2).

The computer used for the executions is Tirant, an IBM cluster consisting of 512 JS21 blade computing nodes, each of them with two 64-bit PowerPC 970MP dual core processors running at 2.2 GHz with 4 GB of memory, interconnected with a low latency Myrinet network. All runs placed a single MPI process per node. The software consists of SLEPc 3.6 and PETSc 3.6, together with MUMPS 5.0 that is used where a parallel direct linear solver is required.

Table 7.1 shows results for 6 NLEVP test problems of varying dimensions, solved using 1 processor. The results include the maximum backward error obtained for a tolerance of 10^{-8} when computing 10 eigenpairs with a maximum basis size $k = 25$. Also, the total computation time is shown (including the initial matrix factorization). In all runs, the projected problem was solved with the second option discussed in section 2, that is, the QR method followed by pseudo-orthogonalization.

From the results of the experiments, we conclude that the three proposed indefinite methods have strengths and weaknesses. In some problems, these methods have much faster convergence compared to the non-symmetric counterparts, as in *shaft*. Also, these methods may sometimes provide better accuracy. In the *sleeper* problem, that has some real eigenvalues with multiplicity equal to 2, it is noteworthy that the methods that exploit symmetry do not perturb real eigenvalues out of the real axis, while the non-symmetric solvers may sometimes return a complex conjugate pair of

Table 7.1 Results for various test problems in the NLEVP collection, when computing 10 eigenpairs with the different methods using a basis of size $k = 25$. The table shows the number of converged eigenvalues (nconv), number of restarts (its), maximum backward error $\eta_Q(x,\lambda)$ and execution time (for one process).

name	method	nconv	its	η_Q	time
	Krylov-Schur	11	2	1.7×10^{-11}	16
gen_hyper2	Pseudo-Lanczos	11	2	1.2×10^{-9}	18
$n = 1000$	Q-Arnoldi	11	2	7.3×10^{-11}	2.3
$\sigma = -1$	Q-Lanczos	11	2	6.7×10^{-9}	3.6
	TOAR	11	2	8.7×10^{-11}	2.3
	STOAR	11	2	5.9×10^{-10}	2.6
	Krylov-Schur	20	1	8.1×10^{-13}	0.66
schrodinger	Pseudo-Lanczos	24	1	5.9×10^{-12}	0.32
$n = 1998$	Q-Arnoldi	20	1	2.2×10^{-13}	0.21
$\sigma = -0.3$	Q-Lanczos	16	1	5.6×10^{-7}	0.24
$k = 50$	TOAR	20	1	3.2×10^{-12}	0.21
	STOAR	16	1	2.7×10^{-10}	0.34
	Krylov-Schur	12	81	1.4×10^{-9}	1.5
shaft	Pseudo-Lanczos	12	2	4.4×10^{-10}	0.20
$n = 400$	Q-Arnoldi	12	83	2.3×10^{-6}	0.99
$\sigma = -10$	Q-Lanczos	12	2	2.9×10^{-9}	0.17
	TOAR	10	186	2.9×10^{-9}	1.7
	STOAR	10	2	6.0×10^{-11}	0.19
	Krylov-Schur	10	3	5.7×10^{-10}	0.67
sign1	Pseudo-Lanczos	10	6	1.4×10^{-10}	1.0
$n = 1000$	Q-Arnoldi	10	4	1.6×10^{-11}	0.028
shift	Q-Lanczos	0	6	-	-
	TOAR	10	4	5.0×10^{-10}	0.58
	STOAR	10	6	1.0×10^{-11}	0.85
	Krylov-Schur	11	3	2.9×10^{-12}	185
sleeper	Pseudo-Lanczos	10	3	2.8×10^{-13}	202
$n = 1000000$	Q-Arnoldi	10	3	1.9×10^{-10}	130
$\sigma = -0.9$	Q-Lanczos	2	3	4.0×10^{-9}	86
	TOAR	10	3	5.7×10^{-14}	123
	STOAR	10	3	4.3×10^{-13}	138
	Krylov-Schur	10	2	6.9×10^{-12}	151
spring	Pseudo-Lanczos	10	2	4.7×10^{-13}	162
$n = 1000000$	Q-Arnoldi	10	2	2.2×10^{-7}	111
$\sigma = -10$	Q-Lanczos	6	1	8.8×10^{-8}	91
	TOAR	10	2	9.6×10^{-12}	107
	STOAR	10	2	4.1×10^{-13}	116

eigenvalues with small imaginary part. Furthermore, when solving complex Hermitian problems in complex arithmetic, there is the guarantee that indefinite solvers will always get complex conjugate eigenvalues $(\lambda,\bar{\lambda})$ together, while it may happen that non-symmetric solvers miss one of them depending on convergence.

The downside of the symmetric-indefinite methods is that they occasionally fail to converge, as they rely on the non-symmetric Lanczos process. The good news is that the guard for detecting loss of symmetry discussed in section 6 confers the meth-

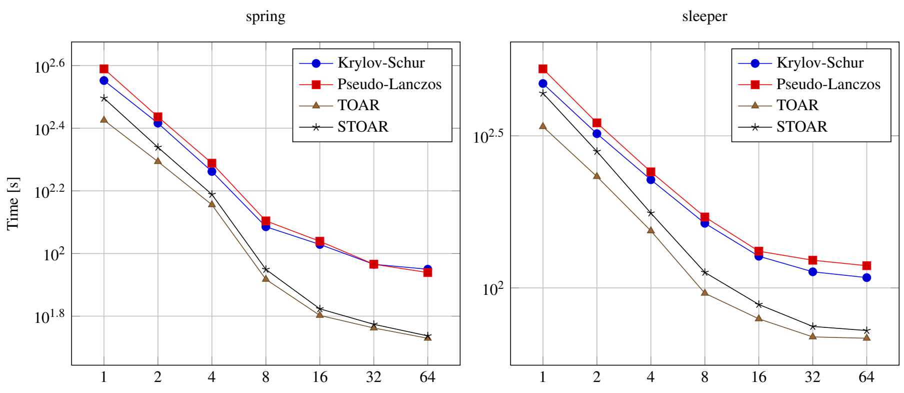

Fig. 7.1 Comparison of parallel execution time of the *spring* (left) and *sleeper* (right) test problems for the different solvers with increasing number of processes. The execution parameters are the same as in Table 7.1, but requesting 30 eigenpairs.

ods guaranteed robustness in the sense that the computed solution is always correct. That is, when the pseudo-Lanczos process becomes unstable, the solver aborts and returns less than the number of requested eigenpairs, but all of them with a small backward error. From Table 7.1, we notice that failure is more often encountered in Q-Lanczos (as in *sign1*, *sleeper* and *spring*), while pseudo-Lanczos and STOAR perform remarkably well except in some cases (e.g. for some values of σ).

Figure 7.1 displays the parallel execution times of the different methods for increasing number of processes in the *spring* and *sleeper* problems, both of them for problem size 1 million. In this case, 30 eigenpairs were requested with a maximum basis size of 60 vectors. The scalability is far from ideal, and this can be attributed to the fact that linear systems are solved with a parallel direct solver (MUMPS), whose overhead grows with the number of processes. The times for TOAR and STOAR are well below those of solvers that operate on the explicit linearization, since in the latter case the matrix to be factored is twice as large. We can also see that STOAR has a small performance penalty compared to TOAR, but both of them scale similarly.

8 Conclusions

We have proposed several methods for exploiting symmetry when solving the symmetric QEP. The first method extends the pseudo-Lanczos recurrence with a thick-restart technique. Using this kind of restart, which has also been incorporated in the other methods, is of paramount importance when solving real applications. The other two methods, Q-Lanczos and STOAR, are memory-efficient schemes that avoid storing the complete Krylov basis. This issue may be quite important in very large problems. Anyway, the reduction of memory requirements will be much more significant in symmetric polynomial eigenvalue problems of higher degree. We will consider as future work how the proposed methods could be extended for this setting.

We must note that the indefinite methods that we have proposed are not as robust as the orthogonal counterparts, as expected. Still, they work reasonably well, especially pseudo-Lanczos and STOAR. Our implementations incorporate a mechanism to avoid returning wrong solutions in case of instability, by aborting execution if symmetry loss is detected. The symmetric-indefinite solvers may have advantage in some situations, with faster convergence or better accuracy, and our implemented solvers are publicly available in SLEPc for anyone interested in trying for applications that could benefit from them.

Acknowledgements The authors are grateful to the reviewers for their constructive comments that helped improve the presentation. The computational experiments of section 7 were carried out on the supercomputer Tirant at Universitat de València.

References

1. Bai Z, Su Y (2005) SOAR: a second-order Arnoldi method for the solution of the quadratic eigenvalue problem. SIAM J Matrix Anal Appl 26(3):640–659
2. Bai Z, Day D, Ye Q (1999) ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems. SIAM J Matrix Anal Appl 20(4):1060–1082
3. Bai Z, Ericsson T, Kowalski T (2000) Symmetric indefinite Lanczos method. In: Bai Z, Demmel J, Dongarra J, Ruhe A, van der Vorst H (eds) Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Society for Industrial and Applied Mathematics, Philadelphia, pp 249–260
4. Balay S, Abhyankar S, Adams M, Brown J, Brune P, Buschelman K, Dalcin L, Eijkhout V, Gropp W, Kaushik D, Knepley M, McInnes LC, Rupp K, Smith B, Zampini S, Zhang H (2015) PETSc users manual. Tech. Rep. ANL-95/11 - Revision 3.6, Argonne National Laboratory
5. Benner P, Faßbender H, Stoll M (2008) Solving large-scale quadratic eigenvalue problems with Hamiltonian eigenstructure using a structure-preserving Krylov subspace method. Electron Trans Numer Anal 29:212–229
6. Betcke T, Higham NJ, Mehrmann V, Schröder C, Tisseur F (2013) NLEVP: a collection of nonlinear eigenvalue problems. ACM Trans Math Software 39(2):7:1–7:28
7. Campos C, Roman JE (2015) Parallel Krylov solvers for the polynomial eigenvalue problem in SLEPc, submitted
8. Day D (1997) An efficient implementation of the nonsymmetric Lanczos algorithm. SIAM J Matrix Anal Appl 18(3):566–589
9. Hernandez V, Roman JE, Vidal V (2005) SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. ACM Trans Math Software 31(3):351–362
10. Hernandez V, Roman JE, Tomas A (2007) Parallel Arnoldi eigensolvers with enhanced scalability via global communications rearrangement. Parallel Comput 33(7–8):521–540
11. Jia Z, Sun Y (2013) A refined variant of SHIRA for the skew-Hamiltonian/Hamiltonian (SHH) pencil eigenvalue problem. Taiwan J Math 17(1):259–274

12. Kressner D, Roman JE (2014) Memory-efficient Arnoldi algorithms for linearizations of matrix polynomials in Chebyshev basis. Numer Linear Algebra Appl 21(4):569–588
13. Kressner D, Pandur MM, Shao M (2014) An indefinite variant of LOBPCG for definite matrix pencils. Numer Algorithms 66(4):681–703
14. Lancaster P (2008) Linearization of regular matrix polynomials. Electron J Linear Algebra 17:21–27
15. Lancaster P, Ye Q (1993) Rayleigh-Ritz and Lanczos methods for symmetric matrix pencils. Linear Algebra Appl 185:173–201
16. Lu D, Su Y (2012) Two-level orthogonal Arnoldi process for the solution of quadratic eigenvalue problems, manuscript
17. Meerbergen K (2001) The Lanczos method with semi-definite inner product. BIT 41(5):1069–1078
18. Meerbergen K (2008) The Quadratic Arnoldi method for the solution of the quadratic eigenvalue problem. SIAM J Matrix Anal Appl 30(4):1463–1482
19. Mehrmann V, Watkins D (2001) Structure-preserving methods for computing eigenpairs of large sparse skew-Hamiltonian/Hamiltonian pencils. SIAM J Sci Comput 22(6):1905–1925
20. Parlett BN (1980) The Symmetric Eigenvalue Problem. Prentice-Hall, Englewood Cliffs, NJ, reissued with revisions by SIAM, Philadelphia, 1998.
21. Parlett BN, Chen HC (1990) Use of indefinite pencils for computing damped natural modes. Linear Algebra Appl 140(1):53–88
22. Parlett BN, Taylor DR, Liu ZA (1985) A look-ahead Lánczos algorithm for unsymmetric matrices. Math Comp 44(169):105–124
23. de Samblanx G, Bultheel A (1998) Nested Lanczos: Implicitly restarting an unsymmetric Lanczos algorithm. Numer Algorithms 18(1):31–50
24. Sleijpen GLG, Booten AGL, Fokkema DR, van der Vorst HA (1996) Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems. BIT 36(3):595–633
25. Stewart GW (2001) A Krylov–Schur algorithm for large eigenproblems. SIAM J Matrix Anal Appl 23(3):601–614
26. Su Y, Zhang J, Bai Z (2008) A compact Arnoldi algorithm for polynomial eigenvalue problems, talk presented at RANMEP
27. Tisseur F (2004) Tridiagonal-diagonal reduction of symmetric indefinite pairs. SIAM J Matrix Anal Appl 26(1):215–232
28. Tisseur F, Meerbergen K (2001) The quadratic eigenvalue problem. SIAM Rev 43(2):235–286
29. Watkins DS (2007) The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods. Society for Industrial and Applied Mathematics
30. Wu K, Simon H (2000) Thick-restart Lanczos method for large symmetric eigenvalue problems. SIAM J Matrix Anal Appl 22(2):602–616