Strategies for spectrum slicing based on restarted Lanczos methods*

Carmen Campos, Jose E. Roman

DSIC, Universitat Politècnica de València, Camí de Vera s/n, 46022 València (Spain)

Tel.: +34-963877356, Fax: +34-963877359

carcamgo@upv.es, jroman@dsic.upv.es

Abstract

In the context of symmetric-definite generalized eigenvalue problems, it is often required to compute all eigenvalues contained in a prescribed interval. For large-scale problems, the method of choice is the so-called spectrum slicing technique: a shift-and-invert Lanczos method combined with a dynamic shift selection that sweeps the interval in a smart way. This kind of strategies were proposed initially in the context of unrestarted Lanczos methods, back in the 1990's. We propose variations that try to incorporate recent developments in the field of Krylov methods, including thick restarting in the Lanczos solver and a rational Krylov update when moving from one shift to the next. We discuss a parallel implementation in the SLEPc library and provide performance results.

1 Introduction

Eigenvalue problems lie at the core of many computationally intensive applications in science and engineering. In this work, we focus on the symmetric-definite generalized eigenvalue problem,

$$Ax = \lambda Bx, \tag{1}$$

where A and B are real symmetric (or complex Hermitian) matrices of order n, and without loss of generality we assume that B is positive (semi-)definite. The above equation is satisfied exactly by n eigenvalue-eigenvector pairs, (λ_i, x_i), where all the eigenvalues are real, $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, and the eigenvectors are mutually B-orthogonal, $x_i^* B x_j = 0$ if $i \neq j$. The case of singular B, where some of the eigenvalues become infinite, will also be treated in this paper. Note that we consider only regular pencils, that is, A and B do not share a common null space, so $A - \sigma B$ is nonsingular unless σ is an eigenvalue. Another aspect that we will take into consideration is when some eigenvalues are multiple, $\lambda_i = \lambda_j, i \neq j$.

In large-scale applications it is prohibitive to compute the whole spectrum, and only a subset of eigenvalues are sought. Depending on the application, the number of wanted eigenvalues can range from 1 to a substantial percentage of n. The part of the spectrum of interest is also application-dependent. In this paper, we are concerned with the case when all eigenvalues contained in a given

*This work was supported by the Spanish Ministerio de Ciencia e Innovación under grant TIN2009-07519.

interval $[a, b]$ are required, where either a or b may be infinite. The classical scenario for this is when the eigenvalue is directly related to a frequency, and the objective is to analyze a dynamical system in a given frequency range, *e.g.*, in undamped vibrations of structures. Also, many other applications in electromagnetism or electronic structure calculations require the use of computational intervals. In these applications, the matrices are very large and sparse, and eigenvalues may have very high multiplicity, thus making the accurate computation of all eigenvalues in the interval a challenging task.

The wanted interval typically lies in the interior of the spectrum, or in a region far away from the largest eigenvalues in magnitude. Therefore, Krylov subspace methods alone cannot be used for this purpose. However, the situation changes when used in combination with a shift-and-invert technique that operates with matrix $(A-\sigma B)^{-1}B$ and maps eigenvalues λ to $\theta = (\lambda-\sigma)^{-1}$, since eigenvalues that are closest to the shift σ become dominant in the transformed spectrum. The spectral transformation Lanczos method [3, 10] has been used successfully for this task, often with several shifts within an interval. The motivation for using several shifts is that eigenvalues close to σ converge very fast because they are mapped to well separated θ's, but for eigenvalues relatively far away from the shift the effectiveness of the spectral transformation decays significantly.

A multi-shift spectral transformation Lanczos method must not be implemented naively. The main reason is that changing to a new shift may imply a high computational cost. Dealing with the inverse $(A-\sigma B)^{-1}$ requires solving a linear system of equations every time the Lanczos basis must be expanded. Typically, this matrix is factored at the beginning and triangular solves are carried out during the iteration. Therefore, a change of σ forces recomputation of the factorization, and there is a trade-off between this cost and the savings due to improved convergence. Also, a smart multi-shift strategy must choose the new shift in a judicious way, not very far away from the current position, but not too close either. The goal is to discover all eigenvalues, without any miss (including multiplicities), and avoid wastefully computing the same eigenvalues from different shifts. For this, a very valuable tool is matrix inertia, that allows the validation of subintervals where all eigenvalues have been computed, extending them until the whole interval has been covered. This type of elaborated strategies are usually referred to as *spectrum slicing*.

The 1994 paper by Grimes, Lewis and Simon [4] provides a detailed description of a robust and efficient spectrum slicing procedure. Their method is based on an unrestarted block Lanczos method with partial reorthogonalization, together with various heuristics for dynamic shift selection. At about the same time, the area of Krylov eigensolvers experienced a big leap with the development of effective restarting mechanisms, beginning with Sorensen's implicit restart [14], which was employed by many authors in the context of Lanczos methods, and later with the thick-restart Lanczos method [17], which is the symmetric equivalent to Stewart's Krylov-Schur method [15]. These methods can effectively compute a given number of eigenvalues using a subspace of bounded dimension. The main idea of these advanced restarting mechanisms is to compress the built subspace into a subspace of smaller dimension retaining as much useful spectral information as possible. The compressed subspace must satisfy the Krylov relation so that it can be used as a starting point to extend the Lanczos factorization. We will see that it is possible to take the compressed subspace computed from a given shift σ_1 and translate it so that it becomes the seed for the Lanczos recurrence of a different shift σ_2. This is not a new idea: it is related to the well-known rational Krylov method [13], that has been employed, *e.g.*, as a multi-shift Arnoldi method for model reduction [11], and also in a professional (sequential) implementation of block Lanczos with variable shifts in the HSL library [9, 8] (`EA16` subroutine).

A weakness of the Grimes *et al.* approach is that they ground their heuristics on a cost model

that may no longer be valid. The authors present results on moderately sized problems (up to 15,000) that have very small multiplicities (3 at most). We are now interested in computing thousands of eigenvalues of problems that can reach millions of degrees of freedom with clusters of eigenvalues as large as 200. Furthermore, these problem sizes require the use of parallel computing, and in this context we must reconsider certain decisions such as which operations must be avoided in favour of others.

We propose an updated spectrum slicing methodology that relies on a thick-restart Lanczos method, combined with a rational Lanczos update, with a number of strategies for shift selection, deflation, etc. that aim at optimizing the utilization of resources when solving large problems on supercomputers. Our intention is to provide an industrial-strength parallel implementation in SLEPc, the Scalable Library for Eigenvalue Problem Computations [6]. A similar endeavor was pursued by Zhang *et al.* [18], also with SLEPc, proposing a strategy for shift selection and solution bookkeeping mostly tailored for a specific application and focusing mainly on a multi-communicator parallelization. Another parallel implementation of spectrum slicing customized for Toeplitz matrices can be found in [16]. Our goal is to provide a general solver that can be robust enough to be used in any scenario. There is a need for such a tool, since the tests in [18] revealed that neither the Grimes *et al.* solution (implemented in the commercial sequential library BCSLIB-EXT) nor the similar BLZPACK package [7] are robust enough.

The plan of the paper is the following. In §2, we describe the main ingredients required in a spectrum slicing method, as well as the particular approach advocated by Grimes *et al.* Section 3 provides a detailed report of the proposed methodology. In §4, we discuss some aspects of our particular implementation. The evaluation of the implemented algorithm is treated in §5. We wrap up with some conclusions.

2 Preliminaries and related work

In this section we set up the notation and provide basic facts about Lanczos methods. We also describe the main features of the method by Grimes *et al.*

2.1 Spectral transformation Lanczos

The shift-and-invert spectral transformation consists in replacing (1) by

$$Sx = \theta x, \qquad S = (A - \sigma B)^{-1} B, \tag{2}$$

for a given shift $\sigma \in \mathbb{R}$, where $\theta = (\lambda - \sigma)^{-1}$. Matrix S is not symmetric but it is self-adjoint with respect to the inner product induced by B, $\langle u, v \rangle_B := u^* B v$ (the case of singular B is deferred for later discussion). With this inner product, and the related norm $\|x\|_B := \sqrt{\langle x, x \rangle_B}$, the Lanczos recurrence

$$\begin{aligned} \beta_1 v_2 &= S v_1 - \alpha_1 v_1, \\ \beta_j v_{j+1} &= S v_j - \alpha_j v_j - \beta_{j-1} v_{j-1}, \qquad j = 2, 3, \ldots \end{aligned} \tag{3}$$

where $\alpha_j = \langle v_j, S v_j \rangle_B$ and $\beta_j = \|S v_j - \alpha_j v_j - \beta_{j-1} v_{j-1}\|_B$, generates a B-orthonormal basis of the Krylov subspace spanned by S and an initial vector v_1. That is, $V_j^* B V_j = I$ where $V_j =$

$[v_1, v_2, \ldots, v_j]$ and we assume that v_1 has been normalized so that $\|v_1\|_B = 1$. The Lanczos decomposition

$$SV_j = V_j T_j + \beta_j v_{j+1} e_j^* \tag{4}$$

relates the Lanczos basis V_j to the symmetric tridiagonal matrix

$$T_j = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{j-2} & \alpha_{j-1} & \beta_{j-1} \\ & & & \beta_{j-1} & \alpha_j \end{bmatrix}, \tag{5}$$

where e_j is the jth coordinate vector. The Krylov subspace contains increasingly accurate approximations of eigenvectors of (2), which can be retrieved by means of a Rayleigh-Ritz procedure. Let $(\tilde{\theta}_i, y_i)$ be an eigenpair of (5), $T_j y_i = \tilde{\theta}_i y_i$, then postmultiplying (4) by y_i results in

$$SV_j y_i = \tilde{\theta}_i V_j y_i + \beta_j v_{j+1} y_{ji}, \tag{6}$$

where $y_{ji} = e_j^* y_i$. Since $V_j^* B v_{j+1} = 0$ and $T_j = V_j^* B S V_j$, $\tilde{x}_i = V_j y_i$ is the Ritz vector satisfying the Galerkin condition that the residual $S\tilde{x}_i - \tilde{\theta}_i \tilde{x}_i$ is B-orthogonal to the Krylov subspace. An alternative eigenvector approximation is the purified vector,

$$\hat{x}_i = \tilde{x}_i + \frac{\beta_j y_{ji}}{\tilde{\theta}_i} v_{j+1}, \tag{7}$$

whose name is motivated by the case of singular B, see §3.3.

A cheap convergence criterion for an eigenpair is $\beta_j |y_{ji}| \le \varepsilon |\tilde{\theta}_i|$ for a given tolerance ε, which involves a bound for the B-norm of the residual $S\tilde{x}_i - \tilde{\theta}_i \tilde{x}_i$.

The Lanczos vectors v_j start to lose their mutual B-orthogonality as soon as a Ritz value stabilizes. Therefore, practical implementations must explicitly enforce B-orthogonality by either full reorthogonalization in each iteration or the cheaper partial reorthogonalization. Another variation of the method is to use a block Lanczos strategy wherein the recurrence operates on blocks of p vectors and T_j is a banded matrix, see [4] for details.

In a spectrum slicing method, we have a sequence of shifts, σ_k, $k = 1, 2, \ldots$, and for each shift we must compute an indefinite (block-)triangular factorization

$$A - \sigma_k B = L_k D_k L_k^*, \tag{8}$$

whose factors will be used during the Lanczos iteration to effect the action of $(A - \sigma_k B)^{-1}$ on a vector (D_k is block-diagonal with 1×1 or 2×2 diagonal blocks). By Sylvester's law of inertia, we get as a byproduct the number of eigenvalues on the left of σ_k

$$\nu_k := \nu(A - \sigma_k B) = \nu(D_k), \tag{9}$$

so if $\sigma_k < \sigma_{k+1}$ then in the interval $[\sigma_k, \sigma_{k+1}]$ there are $\nu_{k+1} - \nu_k$ eigenvalues, counted with their multiplicities.

2.2 Restarted Lanczos methods

In practical Lanczos implementations, the number of basis vectors v_j must be bounded, due to memory limitations but also because the computational cost increases with j, especially if full reorthogonalization is employed. Restarting a Lanczos decomposition (4) consists in rebuilding the decomposition in a way that it contains better approximations compared to the previous one. To build the decomposition from scratch one should choose an initial vector v_1 that contains as many useful spectral components as possible taken from the first decomposition, but this vector is very difficult to compute explicitly. Thick-restart Lanczos [17] achieves the same effect in an implicit way, by keeping a low-dimensional subspace rather than a single vector.

Suppose a Lanczos decomposition of order m has been computed,

$$SV_m = \left[\begin{array}{cc} V_m & v_{m+1} \end{array} \right] \left[\begin{array}{c} T_m \\ b_m^* \end{array} \right], \tag{10}$$

where $b_m = \beta_m e_m$, then the goal is to compress the m-dimensional subspace to a smaller subspace, of dimension $\hat{m} = m/2$, say, and keep the directions associated to the most relevant approximations (in our case, those corresponding to the largest Ritz values $\tilde{\theta}_i$ in magnitude). For this, compute a sorted eigendecomposition $Y_m^* T_m Y_m = \tilde{\Theta}_m$ where the Ritz values appear on the diagonal of $\tilde{\Theta}_m$ in decreasing order of magnitude. The decomposition can be truncated by taking the first $\hat{m}$ columns of the decomposition resulting from the similarity transformation with Y_m,

$$SV_m \hat{Y}_m = \left[\begin{array}{cc} V_m \hat{Y}_m & v_{m+1} \end{array} \right] \left[\begin{array}{c} \hat{\Theta}_m \\ \hat{b}_m^* \end{array} \right], \tag{11}$$

where $\hat{b}_m = \hat{Y}_m^* b_m$, $\hat{Y}_m$ represents the first $\hat{m}$ columns of Y_m and $\hat{\Theta}_m$ is the $\hat{m} \times \hat{m}$ leading principal submatrix of $\tilde{\Theta}_m$. Equation (11) is not strictly a Lanczos decomposition, because $\hat{b}_m$ is full and breaks the tridiagonal shape. It is the more general Krylov decomposition [15], that can be extended by orthogonalizing Sv_{m+1} with respect to the new basis vectors $\hat{V}_m = V_m \hat{Y}_m$ and continuing the Lanczos recurrence.

Restarted methods are effective even for moderate values of m, and this has implications on different implementation details. For instance, it is sufficient to check for convergence only at the time of a restart, not at every Lanczos iteration. In the case that one or more eigenpairs have converged to the requested precision, these eigenvectors are *locked*, that is, the corresponding columns of $\hat{V}_m$ are extracted from the active basis, so the dimension is effectively reduced even further, but subsequent Lanczos iterations enforce orthogonality against these locked vectors, to avoid the appearance of spurious duplicates. The number of locked vectors may become large compared to the maximum basis size m, and explicit orthogonalization against the first $\hat{m}$ vectors of the active basis is also required. For these reasons, restarted methods do not need to bother about sophisticated schemes such as partial reorthogonalization when expanding the Lanczos basis.

2.3 The Grimes-Lewis-Simon approach

The Grimes-Lewis-Simon (GLS) method [4] performs spectrum slicing of the requested interval $[a, b]$ by a multi-shift strategy that sweeps the interval computing eigenvalues by chunks, and using inertia to validate sub-intervals. The GLS approach is based on a block Lanczos method, where ideally the blocksize p must be chosen to accommodate the maximum eigenvalue multiplicity of the problem at hand (but no larger than 10), together with an efficient partial and external selective

reorthogonalization technique. The Lanczos method used in GLS is unrestarted, *i.e.*, does not incorporate any of the techniques discussed in §2.2.

The general strategy of GLS is to create an initial trust sub-interval (where it is possible to assure that all eigenvalues contained therein have been computed), and extend it until the whole $[a, b]$ interval has been completed. Next we summarize the main distinguishing features of their strategy.

- At each Lanczos step, convergence of eigenvalues is monitored. When convergence is slower than desired, or the cost per iteration has grown too much, the Lanczos iteration is interrupted and the analysis moves to a new shift.
- The selection of the new shift σ_{k+1} tries to leave as many to-be-computed eigenvalues on its left as eigenvalues were computed to the right of σ_k. This decision is based on non-converged Ritz values, if enough of them are available. Otherwise, an alternative criterion is used based on the distance from σ_k to the furthest computed eigenvalue to the right.
- The shift selection often leaves gaps, so in order to close the trust interval it is necessary to place additional shifts in between σ_k and σ_{k+1}.
- A deflation strategy is used to avoid recomputation of eigenvalues already computed from the previous shift, by enforcing orthogonality against some of the eigenvectors already computed (see a detailed discussion in §3.1).
- At a new shift, the block Lanczos recurrence is started by a block of vectors obtained by adding unconverged Ritz vectors available from the previous shift.

3 New strategies for spectrum slicing

The GLS method is effective for many situations but it may have pitfalls in some cases, for instance when multiplicity is high. The main weakness is probably the assumption that all multiplicities are of the same order, and using a block Lanczos method that exploits *a priori* knowledge of the multiplicity. Also, the block size cannot be arbitrarily large, and difficulties may arise with high multiplicities.

Another drawback is that GLS tries to avoid orthogonalization as much as possible, even if this implies more Lanczos iterations due to many repeated eigenvalues being discarded. This assumes that triangular solves are much cheaper than orthogonalization. But this is no longer true for very large matrices in the context of parallel computation.

Our method, that uses a restarted Lanczos solver, is based on new assumptions. First, Lanczos convergence is not a problem because the cost per iteration does not grow too much. Second, the orthogonalization is relatively cheap and scales very well in parallel. In contrast, performance of the factorization degrades with n and triangular solves are not scalable in parallel. With these assumptions, we devise new strategies aiming at obtaining a spectrum slicing technique that can be robust enough for irregular spectra with high multiplicity, and scalable to 100's of processors. These strategies try to reduce the number of factorizations (avoid creating unnecessary shifts) as well as Lanczos iterations (containing costly triangular solves) by orthogonalizing more.

The basic idea of our method is that, at each shift σ_i, a fixed number of eigenvalues (`nev`) is requested, with a limited number of restarts (`maxit`). Then, the shift is moved and the factorization is recomputed.

3.1 Shift selection, backtracking and deflation

The first shift σ_1 will always be one of the ends of interval (the left one if it is finite, or the right one otherwise). For choosing each new shift, σ_{i+1}, the average distance, δ, between eigenvalues computed from the two previous shifts (σ_i and σ_{i-1}) is used. Based on some experiments, we have noticed that for matrices with medium or high multiplicity, this distance seems to be more appropriate than the average distance calculated from the last shift only, σ_i. Therefore, when all eigenvalues between σ_1 and σ_i have been computed, we choose a new $\sigma_{i+1} = \tilde{\lambda} + \xi\frac{\mathtt{nev}}{2}\delta$, where $\tilde{\lambda}$ is the eigenvalue computed from σ_i furthest away from σ_1, the direction ξ is 1 or -1 depending on the initial shift, and `nev` is the number of eigenvalues expected for the new shift. To simplify the description, in the following we assume that $\xi = 1$ and we have started from the left end of the interval.

If an irregular distribution of the spectrum results in the chosen shift being too far away, it might not be possible or appropriate to complete the trust subinterval from such shift. In this case there will be a number of eigenvalues inside subinterval $[\sigma_{i-1}, \sigma_i]$ that would not be calculated. Also, even in the case of a shift located at the appropriate distance, data from inertia might reveal that some eigenvalues within $[\sigma_{i-1}, \sigma_i]$ are missing, most probably due to incompletely computed multiplicities. In any case, it will be necessary to place a shift backwards in order to search for those missing values (*backtracking*). This new shift will be chosen halfway between σ_{i-1} and σ_i. Backtracking will be needed whenever inertia informs about missing eigenvalues.

Since Lanczos methods do not guarantee obtaining the complete multiplicity of computed eigenvalues, one of the difficulties faced by spectrum slicing is ascertaining whether some particular values are spurious duplicates or genuine multiples of eigenvalues already computed from previous shifts. Another problem that spectrum slicing has to address is to ensure 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 allows working in the orthogonal complement of a set of selected vectors.

Deflation involves an additional cost, because one vector has to be orthogonalized against all vectors selected for deflation for each iteration of the Lanczos method. On the other hand, deflation can also have a positive effect on cost, since it reduces the possibility of recalculating eigenvalues already obtained. Those two opposing effects of deflation have been studied in this work in order to choose the appropriate amount of deflation to be applied for improving performance of our parallel code. We consider three strategies for selecting the set of vectors for deflation, described below.

Basic strategy The minimum deflation necessary for spectrum slicing is the minimum deflation that ensures orthogonality between eigenvectors calculated from different shifts. In this first strategy, the set of eigenvectors selected for deflation correspond to the rightmost eigenvalue and all eigenvalues to its left until two consecutive values are separated by a distance greater than $\tilde{\varepsilon}$, where $\tilde{\varepsilon}$ is a fixed tolerance needed for assessing whether two eigenvalues belong to a cluster. When calculating eigenpairs from the new shift, any value found to the left of the smallest eigenvalue considered for deflation (plus the defined tolerance) is discarded.

GLS strategy During computation for a given shift σ_i, it is quite common to leave several unconverged Ritz values to the left of accepted ones, but with the previous strategy these values cannot be obtained when computing from σ_{i+1} since they will be discarded if they appear. The goal of this second strategy, used in the GLS method, is to allow the computation of these values from the new shift, in this way avoiding backtracking. The set of eigenvectors selected for deflation are those associated to the largest converged eigenvalue lying on the left of any unconverged Ritz value, $\lambda^\dagger$,

together with all converged eigenvalues to its right. As in the basic strategy, it is necessary to take into account possible eigenvalue clusters around $\lambda^\dagger$ (with a tolerance $\tilde{\varepsilon}$). Again, any computed eigenvalue found to the left of the first eigenvalue in the deflation set is discarded.

Augmented deflation strategy The previous strategy does not account for missing eigenvalues that do not have an associated unconverged Ritz value (located to the left of $\lambda^\dagger$). This happens, for instance, in multiple eigenvalues, because it is quite common that not all copies are computed in a given shift σ_i and therefore the remaining copies are computed from σ_{i+1} but discarded if the GLS strategy is used. So for high multiplicity it is necessary to recourse to backtracking often. The third criterion tries to avoid pending eigenvalues being discarded. We define the left neighbour σ_ℓ of a given shift σ_i to be the largest used shift which is smaller than σ_i (not necessarily the previous one σ_{i-1}), and equivalently the right neighbour σ_r. When computing from σ_i, the subinterval $[\sigma_1, \sigma_\ell]$ is a trust interval (all eigenvalues have been found). The set of deflation vectors for this criterion consists of all eigenvectors associated to converged eigenvalues between σ_ℓ and σ_i. In this way, in the event of discarding an eigenvalue it would belong to the trust interval $[\sigma_1, \sigma_\ell]$, and any value in $[\sigma_\ell, \sigma_i]$ will be accepted. Even in this case, backtracking may be necessary if the run at σ_i terminates (either `nev` eigenvalues found or `maxit` restarts done) without having computed all eigenvalues in $[\sigma_\ell, \sigma_i]$. A new factorization to compute a few eigenvalues may be wasteful, so if the number of remaining eigenvalues is very small (*e.g.*, `nev`/4), we advocate for continuing the computation with the same shift, rather than forcing backtracking, at least for a few restarts more (`maxit`/4 in our implementation). This optimization is possible only in the third strategy, not in the others.

Irrespective of which deflation strategy is being used, when the current shift σ_i has been created for backtracking, we force to use the augmented deflation strategy taking all converged eigenvectors of values between σ_ℓ and σ_r. In this situation, the Lanczos run can only generate eigenvalues in the interval $[\sigma_\ell, \sigma_r]$ that were not found before, or eigenvalues outside $[\sigma_\ell, \sigma_r]$. We assume that eigenvalues previously computed were the most favourable in terms of convergence, so it is important to purge them, otherwise the difficult ones have no chance to appear. Also, eigenvalues with high multiplicity can be computed assuring orthogonality of the associated eigenvectors even if more that two shifts are required to discover all of them.

3.2 Recycling subspaces during the change of shift

As shown in [12, 8, 11], it is feasible to transform the basis obtained from one shift σ_1, into another one that generates the same Krylov subspace and that can be seen as obtained from another shift σ_2. We intend to test if such procedure could also be of interest in the context of spectrum slicing. In other words, we wonder if it is beneficial to reuse part of the Krylov relation obtained in the previous shift to build the new subspace when starting to compute from a new shift, thus avoiding creating it all anew.

Suppose we have a Krylov-Schur relation (11) obtained by truncating the Lanczos decomposition computed with $S_1 = (A - \sigma_1 B)^{-1} B$,

$$S_1 \hat{V}_m = \hat{V}_{m+1} \begin{bmatrix} \hat{\Theta}_m \\ \hat{b}_m^* \end{bmatrix}, \tag{12}$$

where $\hat{V}_{m+1} = \begin{bmatrix} \hat{V}_m & v_{m+1} \end{bmatrix}$, and we intend to derive another one for $S_2 = (A - \sigma_2 B)^{-1} B$, for the same space $\mathcal{R}(\hat{V}_m)$.

From (12), and introducing the new shift, we obtain

$$B\hat{V}_{m+1}L_m = (A-\sigma_2 B)\hat{V}_{m+1}\begin{bmatrix}\hat{\Theta}_m\\ \hat{b}_m^*\end{bmatrix}, \tag{13}$$

with

$$L_m = \begin{bmatrix} I_m \\ 0 \end{bmatrix} + (\sigma_1-\sigma_2)\begin{bmatrix}\hat{\Theta}_m\\ \hat{b}_m^*\end{bmatrix}. \tag{14}$$

The Krylov relation sought,

$$S_2 W_m = \begin{bmatrix} W_m & w_{m+1}\end{bmatrix}\begin{bmatrix} G_m \\ t_m^*\end{bmatrix}, \tag{15}$$

is obtained using the QR factorization of L_m, $L_m = Q_{m+1}\begin{bmatrix} R_m \\ 0\end{bmatrix}$, updating the basis $W_{m+1} = \hat{V}_{m+1}Q_{m+1}$ of the Krylov subspace and the projected matrix,

$$\begin{bmatrix} G_m \\ t_m^*\end{bmatrix} = Q_{m+1}^*\begin{bmatrix}\hat{\Theta}_m\\ \hat{b}_m^*\end{bmatrix}R_m^{-1} = \frac{1}{\sigma_1-\sigma_2}\left(I - Q_{m+1}^*\begin{bmatrix} R_m^{-1} \\ 0\end{bmatrix}\right). \tag{16}$$

Note that the resulting matrix G_m has no particular nonzero structure.

In our implementation, we employ this technique whenever the Krylov subspace available from the previous shift is useful for the next one.

3.3 The case of singular B

If B is singular, *i.e.*, semi-definite, the eigenproblem (1) has infinite eigenvalues. We are interested in computing eigenvectors corresponding to finite eigenvalues, which belong to the range (column space) of S. It can be shown [10] that the Lanczos vectors belong to $\mathcal{R}(S)$ provided that the initial vector v_1 also lies in this subspace, *e.g.*, computing it as $v_1 = S\hat{v}_1$ for some random vector $\hat{v}_1$ and then normalizing it. In finite precision arithmetic this is not sufficient for maintaining all the computation within $\mathcal{R}(S)$, because rounding errors introduce components in the nullspace of B that do not affect the Lanczos recurrence but contaminate the computed eigenvectors and must therefore be removed. This purification of eigenvectors can be done in two ways: with the explicit application of S as in the case of the initial vector, or by using the purified vector (7), which is equal to $S\tilde{x}_i/\tilde{\theta}_i$. In our implementation we use the latter and compute it at the moment of locking converged Ritz vectors.

4 Implementation details

The spectrum slicing method described in the previous section has been implemented in SLEPc, the Scalable Library for Eigenvalue Problem Computations [6], and it has been included in version 3.2 (and later). SLEPc is a freely available software package for the solution of large-scale eigenvalue problems on parallel computers. It provides solvers for linear eigenvalue problems (standard or generalized), as well as quadratic eigenproblems and the singular value decomposition. Most of the linear eigensolvers can address general non-symmetric problems, but internally the algorithm

is specialized to possibly exploit symmetry. For instance, there is a Krylov-Schur solver in SLEPc, that will behave as a thick-restart Lanczos if the problem is known to be symmetric-definite.

SLEPc is built on top of PETSc (Portable, Extensible Toolkit for Scientific Computation [2]), a parallel framework for the numerical solution of partial differential equations, that provides object-oriented abstractions of mathematical objects such as matrices and vectors (the underlying data structures are mostly hidden to the application programmer), together with various classes of solver objects, including linear, non-linear and time-stepping solvers. For some functionality such as direct linear solvers and preconditioners, PETSc offers the possibility to interface with third-party software libraries in a straightforward way, as has been done with MUMPS [1], for example.

The user can specify many run-time parameters such as the number of eigenvalues to compute, the convergence tolerance, or the maximum dimension of the built subspace. The thick-restart Lanczos method implemented in SLEPc is a single-vector version, where in the case of generalized symmetric-definite problems:

- Each new Lanczos vector is explicitly B-orthogonalized against all previous Lanczos vectors and any other deflation vectors given by the user. Orthogonalization is done with an iterative classical Gram-Schmidt method that is both numerically robust and efficient in terms of memory overhead and parallel communication [5].

- The shift-and-invert spectral transformation (2) is implemented by means of factorizations and triangular solves via PETSc's direct linear solvers.

For the implementation of spectrum slicing, we use MUMPS to compute the indefinite (block-)triangular factorization (8) and get the inertia information. The slicing solver maintains a parallel data structure containing all eigenvectors computed from the different shifts, and a subset of these vectors is used for deflation during the Lanczos basis expansion. Our code uses the augmented deflation strategy, together with rational update and interval completion (we have also implemented the other alternatives for the experiments in §5). The parameter `nev` can be selected by the user, typically in the range 40-120, and may have a significant impact on performance.

Parallelization is based on performing collective operations on distributed matrices and vectors. The factorization and linear solves are done in parallel (with MUMPS) as well as all operations required for the Lanczos run. The different shifts are processed one after the other.

5 Evaluation

In this section we provide performance results of the proposed method with a set of test matrices. We analyze the impact of the different choices of deflation, as well as the use of rational Krylov updates and the completion of intervals. At the end of the section, we also show results of parallel executions to analyze scalability of the implementation.

Table 1 shows the test problems that have been used for the experiments. The first five problems come from the analysis of mechanical structures: *fuse5k*, *fuse1m*, *fuse2m*, and *v1v2-31* have been provided by LMS-SAMTECH company, and *bcsst_12* is an ore car from the Harwell-Boeing matrix collection. The three *fuse* test cases correspond to a simplified but realistic model of a fuselage, consisting of a cylinder with skin, frames and stringers, where the frequency range of interest is [0-60] Hz. This is a parametric model that can scale to an arbitrary dimension, and we use it for scalability studies. The rest are problems related to materials science: *graphene_m* was provided

Table 1: List of problems for the tests, showing the dimension of the matrices, the number of nonzero entries and whether B is singular or not, as well as the requested computational interval with the number of contained eigenvalues. The last column shows the maximum eigenvalue multiplicity in the considered interval (*: cluster size with radius 10^{-6}).

name	dimension	nonzeros	sing. B	interval	# evals	multipl.
fuse5k	5,406	138,477	yes	$[99, 1.5 \cdot 10^7]$	1,145	2
fuse1m	1,036,698	∼29 mill.	yes	$[0, 1.4 \cdot 10^5]$	1,989	2
fuse2m	2,141,646	∼59 mill.	yes	$[0, 1.4 \cdot 10^5]$	2,039	2
v1v2-31	6,732	177,966	no	$[0, 10^8]$	1,444	1
bcsst_12	1,473	17,868	no	$[100, 10^8]$	578	1
graphene_m	1,600	67,200	no	$[-1, 0]$	921	12 (*)
benzenes-2592	23,328	900,720	no	$]-\infty, -0.27]$	4,536	200 (*)

by the authors of [18], and *benzenes-2592* was generated with the SIESTA code[1] for self-consistent DFT computations.

The executions were carried out on CaesarAugusta, an IBM BladeCenter cluster consisting of 256 JS20 blade nodes, each of them with two 64-bit PowerPC 970+ processors running at 2.2 GHz, with 4 GB memory per node, interconnected with a low latency Myrinet network. We have used up to 128 processors, due to account limitations. All tests have been done with a tolerance of 10^{-10} for acceptance of eigenvalues, a tolerance of 10^{-5} for cluster detection (used in the basic and GLS deflation strategies), and a value of 10 for the parameter `maxit` (maximum number of restarts).

Table 2 shows the results with small test matrices (with 1 processor) for different deflation strategies. The most remarkable observation is that we can see a significant reduction in the overall number of Lanczos iterations (directly related to the number of triangular solves) for the augmented deflation compared to the basic and GLS strategies, and this is due to the fact that less duplicate eigenvalues are rejected. Thus we foresee a more significant gain in parallel. We also observe that the option of completing the interval when only a few eigenvalues are missing often helps in further reducing the number of rejections, and in any case it does not penalize. The last line in each group of the table also shows how the rational Lanczos update can represent a further improvement, with a time reduction up to 20% in some cases.

Table 3 shows also a comparison of deflation strategies, but in parallel (with 16 processes, which is the minimum number required for *fuse1m*). The results in terms of reduction of iterations by augmented deflation and rational update are very similar to Table 2. Here we also show split times for the main operations, where we can see that the time of the numerical factorizations and triangular solves is proportional to the number of shifts and iterations, respectively. Regarding the time of orthogonalization, although one would expect an increase with the augmented deflation strategy, we cannot observe wide variations. This is due to the fact that most of this cost is associated to full orthogonalization and locking within a Lanczos run, not to deflation against vectors from other shifts.

In Table 4 we present results with varying number of processes, starting with 16 for *fuse1m* and 32 for *fuse2m*. We can observe a sustained decrease in execution time, up to 128 processes. The time breakdown reveals that, as expected, the part of the computation that hinders scalability

[1] `http://www.icmab.es/siesta`

Table 2: Results for small test matrices: def=deflation strategy (1=basic, 2=GLS, 3=augmented), comp=interval completion activated, rt=rational update, time=total execution time in seconds, nshift=number of shifts (in parenthesis the number of backtracks), rest=number of restarts, its=total number of Lanczos iterations, rej=number of rejected duplicates. These runs were done with `nev`=40.

name	rt	def	comp	time	nshift	rest	its	rej
	0	1	0	166	86 (40)	141	8276	874
	0	2	0	139	66 (20)	118	6602	430
fuse5k	0	3	0	137	60 (16)	115	6128	280
	0	3	1	125	51 (8)	116	5548	157
	1	3	1	115	52 (8)	111	4867	141
	0	1	0	120	23 (2)	36	4212	488
	0	2	0	117	22 (1)	35	4049	429
v1v2-31	0	3	0	120	18 (0)	30	3340	90
	0	3	1	121	18 (0)	30	3340	90
	1	3	1	114	18 (0)	31	2825	78
	0	1	0	36.7	30 (17)	51	3517	392
	0	2	0	38.2	30 (17)	50	3464	278
graphene_m	0	3	0	32.6	27 (13)	46	3157	228
	0	3	1	30.6	22 (9)	48	2967	173
	1	3	1	24.0	19 (6)	42	2364	158
	0	1	0	15.6	49 (11)	90	4901	468
	0	2	0	17.0	49 (11)	90	4885	385
bcsst_12	0	3	0	15.7	41 (5)	81	4222	119
	0	3	1	15.0	36 (1)	85	4001	73
	1	3	1	13.6	36 (2)	80	3307	54

Table 3: Results for the *fuse1m* test case with 16 processes, using `nev`=80: [rt, def, comp, time, nshift, rest, its]=same meaning as in Table 2, tNF=time of numerical factorization, tTS=time of triangular solves, tOrt=time of orthogonalization.

rt	def	comp	time	nshift	rest	its	tNF	tTS	tOrt
0	1	0	12286	34 (14)	51	6149	4012	5656	2179
0	2	0	10778	29 (9)	45	5283	3475	4863	2012
0	3	0	10339	27 (7)	43	4989	3181	4594	2144
0	3	1	9928	25 (6)	47	4892	2937	4508	2071
1	3	1	8691	24 (4)	39	3876	2841	3572	1830

Table 4: Parallel results for varying number of processes (np) with *fuse1m* (`nev`=80) and *fuse2m* (`nev`=120). These tests were run with augmented deflation strategy, interval completion and rational update.

name	np	time	nshift	rest	its	tNF	tTS	tOrt
fuse1m	16	8691	24 (4)	39	3876	2841	3572	1830
	32	5181	24 (4)	42	4046	1147	2757	845
	64	3951	25 (6)	48	4340	725	2430	376
	128	3333	25 (5)	41	4084	524	2228	166
fuse2m	32	11744	17 (4)	27	4080	2378	5323	2872
	64	8060	18 (4)	25	4111	1275	4423	1211
	128	6931	18 (4)	26	4228	981	4407	535

Table 5: Results for test case *benzenes-2592* to illustrate the capability of the code to compute eigenvalues with high multiplicity (using the same options as Table 4 and `nev`=100).

np	time	nshift	rest	its	tNF	tTS	tOrt
2	3328	51 (31)	301	33978	15.6	641	2433
4	1720	52 (28)	302	34292	9.1	493	1093
8	1038	54 (30)	305	34679	6.0	408	551
16	698	52 (28)	304	34412	3.8	360	291

the most is the application of $(A-\sigma B)^{-1}$, that is, triangular solves done by MUMPS. This is the main motivation in our proposed method for trying to minimize this operation by reducing the number of Lanczos iterations as much as possible. The remaining time (total time minus the three operation times shown in the table) corresponds to the symbolic factorization, which is reused from one shift to the next. We have performed this factorization sequentially (since the option of doing it in parallel resulted in worse time for these particular matrices), so the time is constant for all numbers of processes: around 400 seconds for *fuse1m* and 1000 seconds for *fuse2m*.

To complete our analysis, we present parallel execution times for a test problem with large eigenvalue clusters, the largest one containing 200 eigenvalues (considering a radius of 10^{-6}). In such cases, any of the deflation strategies involves orthogonalization against many vectors to guarantee eigenvector orthogonality in the clusters, but in the basic and GLS strategies there is the added difficulty of selecting an appropriate value for the $\tilde{\varepsilon}$ tolerance, since a bad choice can imply unnecessary deflation (for instance, $\tilde{\varepsilon} = 10^{-5}$ would yield clusters of 400 values). Table 5 shows executions with the augmented deflation, using a value of `nev` equal to 100. This problem is very favorable for parallel execution because the factorization is extremely cheap (the nonzero pattern is concentrated in a very narrow band), and therefore most of the cost is in the orthogonalization, which scales well.

6 Conclusions

We have developed a parallel spectrum slicing method, that has been inspired by the work by Grimes *et al.*, but incorporates recent algorithm techniques such as thick-restart Lanczos and rational Krylov update, on one hand, and modifications aiming at improving parallel performance, on the other. Our approach provides the robustness and flexibility necessary to be used in general situations without any restriction associated with concrete types of problems. In particular, our solver can be used in problems with high multiplicity or large eigenvalue clusters. We have devised a deflation strategy that aims at reducing the number of factorizations and triangular solves, which typically scale very badly, even if this requires orthogonalizing more.

Our numerical experiments show that our code is able to solve problems with millions of unknowns, with good parallel behaviour at least up to 128 processes. We have also checked that our proposed strategy of augmented deflation combined with interval completion and rational update provides a significant time reduction in all cases.

As a future work, we are interested in exploring strategies with two levels of parallelism, where the interval is divided in several subintervals, each of them being assigned to a different subgroup of processes, similarly to [18].

Acknowledgements

We are grateful to Masha Sosonkina and Hong Zhang for helpful suggestions of an early draft of the paper, and for providing test matrices. We acknowledge the computer resources provided by the Barcelona Supercomputing Center (BSC).

References

[1] P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods Appl. Mech. Engrg.*, 184(2–4):501–520, 2000.

[2] Satish Balay, Jed Brown, Kris Buschelman, Victor Eijkhout, William Gropp, Dinesh Kaushik, Matt Knepley, Lois Curfman McInnes, Barry Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.2, Argonne National Laboratory, 2011.

[3] T. Ericsson and A. Ruhe. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Math. Comp.*, 35(152):1251–1268, 1980.

[4] Roger G. Grimes, John G. Lewis, and Horst D. Simon. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM J. Matrix Anal. Appl.*, 15(1):228–272, 1994.

[5] V. Hernandez, J. E. Roman, and A. Tomas. Parallel Arnoldi eigensolvers with enhanced scalability via global communications rearrangement. *Parallel Comput.*, 33(7–8):521–540, 2007.

[6] V. Hernandez, J. E. Roman, and V. Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Trans. Math. Softw.*, 31(3):351–362, 2005.

[7] O. A. Marques. BLZPACK: Description and user's guide. Technical Report TR/PA/95/30, CERFACS, Toulouse, France, 1995.

[8] K. Meerbergen. Changing poles in the rational Lanczos method for the Hermitian eigenvalue problem. *Numer. Linear Algebra Appl.*, 8(1):33–52, 2001.

[9] K. Meerbergen and J. Scott. The design of a block rational Lanczos code with partial reorthogonalization and implicit restarting. Technical Report RAL-TR-2000-011, Rutherford Appleton Laboratory, 2000.

[10] Bahram Nour-Omid, Beresford N. Parlett, Thomas Ericsson, and Paul S. Jensen. How to implement the spectral transformation. *Math. Comp.*, 48(178):663–673, 1987.

[11] K. H. A. Olsson and A. Ruhe. Rational Krylov for eigenvalue computation and model order reduction. *BIT*, 46:99–111, 2006.

[12] A. Ruhe. Rational Krylov subspace method. In Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, pages 246–249. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

[13] Axel Ruhe. Rational Krylov sequence methods for eigenvalue computation. *Linear Algebra Appl.*, 58:391–405, 1984.

[14] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[15] G. W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614, 2001.

[16] A. M. Vidal, V. M. Garcia, P. Alonso, and M. O. Bernabeu. Parallel computation of the eigenvalues of symmetric Toeplitz matrices through iterative methods. *J. Parallel and Distrib. Comput.*, 68(8):1113–1121, 2008.

[17] Kesheng Wu and Horst Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 22(2):602–616, 2000.

[18] Hong Zhang, Barry Smith, Michael Sternberg, and Peter Zapol. SIPs: Shift-and-invert parallel spectral transformations. *ACM Trans. Math. Softw.*, 33(2):1–19, 2007.