SIAM J. SCI. COMPUT.
Vol. 38, No. 1, pp. A500–A527

GENERALIZED PRECONDITIONED LOCALLY HARMONIC RESIDUAL METHOD FOR NON-HERMITIAN EIGENPROBLEMS*

EUGENE VECHARYNSKI†, CHAO YANG†, AND FEI XUE‡

Abstract. We introduce the generalized preconditioned locally harmonic residual (GPLHR) method for solving standard and generalized non-Hermitian eigenproblems. The method is particularly useful for computing a subset of eigenvalues, and their eigen- or Schur vectors, closest to a given shift. The proposed method is based on block iterations and can take advantage of a preconditioner if it is available. It does not need to perform exact shift-and-invert transformation. Standard and generalized eigenproblems are handled in a unified framework. Our numerical experiments demonstrate that GPLHR is generally more robust and efficient than existing methods, especially if the available memory is limited.

Key words. eigenvalue, eigenvector, non-Hermitian, preconditioned eigensolver

AMS subject classifications. 65F15, 65F08, 65F50

DOI. 10.1137/15M1027413

1. Introduction.

Large non-Hermitian eigenproblems arise in a variety of important applications, including resonant state calculation [3, 15, 33] or excited state analysis for equation-of-motion coupled-cluster (EOM-CC) approaches [13, 14] in quantum chemistry, linear stability analysis of the Navier–Stokes equation in fluid dynamics [5, 7], crystal growth simulation [21, 22], magnetohydrodynamics [18], power systems design [17], and many others; see, e.g., [45] for more examples.

In their most general form, these problems can be written as

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

where A and B are general square matrices. We are particularly interested in the case in which both A and B are very large and sparse, or available only implicitly through a matrix-vector multiplication procedure. If B is the identity matrix, then (1) becomes a standard eigenproblem.

The spectrum of (1), denoted by $\Lambda(A, B)$, is given by a set of numbers $\lambda \in \mathbb{C}$ that make $A - \lambda B$ singular. The value of λ can be infinity in the case of a singular B. Given a scalar $\sigma \in \mathbb{C}$, which we refer to as a *shift*, we seek to find a subset of eigenvalues $\lambda \in \Lambda(A, B)$ that are closest to σ and their associated (right) eigenvectors x. These eigenvalues can be either extreme eigenvalues (e.g., eigenvalues with the largest magnitude) or interior eigenvalues that are inside the convex hull of the spectrum.

Our focus in this paper is on algorithms for computing interior eigenvalues and their corresponding eigenvectors of a non-Hermitian pencil. It is well known that these eigenpairs are often difficult to compute in practice. Traditional methods, such

*Submitted to the journal's Methods and Algorithms for Scientific Computing section June 23, 2015; accepted for publication (in revised form) December 21, 2015; published electronically February 18, 2016.

http://www.siam.org/journals/sisc/38-1/M102741.html

†Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (evecharynski@lbl.gov, cyang@lbl.gov). The work of these authors was supported by the Scientific Discovery through Advanced Computing (SciDAC) program funded by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research and Basic Energy Sciences.

‡Department of Mathematics, University of Louisiana at Lafayette, Lafayette, LA 70504-1010 (fxue@louisiana.edu). This author's work was supported by the National Science Foundation under grant DMS-1419100.

as the inverse subspace iteration or variants of the shift-invert Arnoldi algorithm (see, e.g., [2, 36]), rely on using spectral transformations that require performing LU factorizations of $A - \sigma B$ and computing solutions to triangular linear systems. Such an approach can be prohibitively expensive, especially when the problem size is large and the LU factors of $A - \sigma B$ are not sparse.

There has been some effort in recent years to develop methods that are factorization free. Examples of such methods include the inexact inverse subspace iteration [34, 48] and inexact shift-invert Arnoldi methods [9, 49] in which linear systems of the form $(A - \sigma B)w = r$ are solved iteratively. While these schemes can be considerably less expensive per iteration, the overall convergence of these methods is less predictable.

Another class of factorization-free methods includes the generalized Davidson (GD) method [27, 28] and the Jacobi–Davidson (JD) methods [8]. These methods generally do not rely on solution of linear systems. The JD-style QR and QZ algorithms (JDQR and JDQZ) presented in [8] can be viewed as *preconditioned eigensolvers* in which a Newton-like correction equation is solved approximately by a preconditioned iterative solver.

The GD method can easily be extended into a block method in which several eigenpairs can be approximated simultaneously [38, 50]. Block GD is widely used in quantum chemistry. A block GD method is particularly well suited for modern high performance computers with a large number of processing units. This is because in a block method several (sparse) matrix vector multiplications, which often constitute the major cost of the algorithm, can be carried out simultaneously. Furthermore, one can easily implement data blocking, exploit data reuse, and take advantage of BLAS3 in a block method. These techniques can lead to significant speedups on modern high performance computers as reported in [1]. However, the convergence of the block GD method depends to a large degree on the effectiveness of the preconditioner and size of the search subspace.

The JDQR and JDQZ methods tend to be more robust with respect to the choice of preconditioners. However, extending JDQR and JDQZ to block methods is not straightforward. Because JDQR and JDQZ methods compute one eigenpair at a time, concurrency can only be exploited within a single matrix-vector multiplication procedure, which limits its parallel scalability to a large number of processors.

In this paper, we present a new family of methods for solving non-Hermitian eigenproblems, called the generalized preconditioned locally harmonic residual (GPLHR) methods. The proposed scheme is a block method that constructs approximations to a partial (generalized) Schur form of the matrix pair (A, B) associated with (1) iteratively. It does not require performing an LU factorization of $A - \sigma B$. It allows a preconditioner to be used to construct a search space from which approximations to the Schur vectors can be extracted. We demonstrate that the GPLHR methods are generally more robust and efficient than JDQR/JDQZ as well as the block GD method when the same preconditioner is used and when dimensions of the search spaces are comparable. In addition, because GPLHR is a block method, it is well suited for high performance computers that consist of many cores or processors.

The GPLHR algorithm is a generalization of the recently introduced preconditioned locally harmonic residual (PLHR) method for computing interior eigenpairs of Hermitian eigenproblems [46] to the non-Hermitian case (hence the name). The GPLHR scheme can also be viewed as an extension of the well-known LOBPCG [19] method and block inverse-free preconditioned (BIFP) Krylov subspace methods [10, 32] for computing extreme eigenpairs of Hermitian problems. At each step, all these

methods construct trial subspaces of a fixed dimension and use them to extract the desired approximations by means of a subspace projection. The key difference is that GPLHR performs iterations using the approximations to Schur vectors instead of potentially ill-conditioned eigenvectors and develops the trial subspaces that are based on the idea of an approximate solution of a generalized Sylvester equation. The proposed method explicitly utilizes properly chosen projectors to stabilize convergence and suggests a thick restart-based definition of additional, or "conjugate," search directions. An important specificity of GPLHR is that it systematically uses a *harmonic Rayleigh–Ritz* procedure [8, 16, 26, 29], which allows for a robust eigen- or Schur pair extraction independent of the location of the targeted eigenvalues.

This paper is organized as follows. In section 2, we review standard eigenvalue-revealing decompositions for non-Hermitian eigenproblems and propose a new generalized Schur form which emphasizes only the right Schur vectors. This form is then used to derive the GPLHR algorithms for standard and generalized eigenproblems in section 3. The preconditioning options are discussed in section 4. Section 5 presents numerical results. Our conclusions can be found in section 6. Appendix A briefly describes the approximate eigenbasis-based GPLHR iteration (GPLHR-EIG).

2. Eigenvalue-revealing decompositions. For non-Hermitian eigenproblems, it is common (see, e.g., [24, 36, 44]) to replace (1) by an equivalent problem

$$\beta A x = \alpha B x, \tag{2}$$

where the pair (α, β), called a *generalized eigenvalue*, defines an eigenvalue $\lambda = \alpha/\beta$ of the matrix pair (A, B). This formulation is advantageous from the numerical perspective. For example, it allows revealing the infinite or indeterminate eigenvalues of (1). The former corresponds to the case where B is singular and $\beta = 0$, $\alpha \neq 0$, whereas the latter is indicated by $\alpha = \beta = 0$ and takes place if both A and B are singular with a nontrivial intersection of their null spaces. Additionally, the separate treatment of α and β allows one to handle situations where either of the quantities is close to zero, which leads to underflow or overflow for $\lambda = \alpha/\beta$. Note that for any nonzero $t \in \mathbb{C}$, $(t\alpha, t\beta)$ is also a generalized eigenvalue. In order to fix a representative pair, a normalizing condition should be imposed, e.g., $|\alpha|^2 + |\beta|^2 = 1$.

In this paper, we assume that the pair (A, B) is *regular*, i.e., $\det(\beta A + \alpha B)$ is not identically zero for all α and β. The violation of this assumption gives a *singular* pair, which corresponds to an ill-posed problem. In this case, a regularization procedure should be applied to obtain a nearby regular pair (see, e.g., [2, Chapter 8.7]) for which the eigenproblem is solved. In most cases, however, (A, B) is regular, even when A or B or both are singular as long as the intersection of their null spaces is zero. Thus, regular pairs can have infinite eigenvalues, whereas the possibility of indeterminate eigenvalues is excluded.

2.1. Partial eigenvalue decomposition and generalized Schur form. Let us assume that the targeted eigenvalues of (1) are nondeficient, i.e., their algebraic and geometric multiplicities coincide. Then, given formulation (2), the *partial eigenvalue decomposition* of (A, B) can be written in the form

$$AX\Lambda_B = BX\Lambda_A, \tag{3}$$

where $X \in \mathbb{C}^{n\times k}$ is a matrix whose columns x_j correspond to the k eigenvectors associated with the wanted eigenvalues λ_j. The k-by-k matrices Λ_A and Λ_B are diagonal, with the diagonal entries given by α_j and β_j, respectively. In our context, it is assumed that $\lambda_j = \alpha_j/\beta_j$ are the eigenvalues closest to σ.

Since, by our assumption, λ_j's are nondeficient, there exist k associated linearly independent eigenvectors so that X is full-rank and (3) is well defined. For a vast majority of applications, the partial eigenvalue decomposition (3) does exist; i.e., the desired eigenvalues are nondeficient. Nevertheless, pursuing decomposition (3) is generally not recommended, as it aims at computing a basis of eigenvectors which can potentially be ill-conditioned.

In order to circumvent the difficulties related to ill-conditioning or deficiency of the eigenbasis, we consider instead a *partial generalized Schur decomposition*

$$\left\{ \begin{array}{rcl} AV & = & QR_A, \\ BV & = & QR_B; \end{array} \right. \tag{4}$$

see, e.g., [8, 36, 43]. Here, V and Q are the n-by-k matrices of the *right and left generalized Schur vectors*, respectively. The factors R_A and R_B are k-by-k upper triangular matrices such that the ratios $\lambda_j = R_A(j,j)/R_B(j,j)$ of their diagonal entries correspond to the desired eigenvalues.

The advantage of the partial Schur form (4) is that it is defined for any (A, B) and both V and Q are orthonormal so that they can be computed in a numerically stable manner. The matrix V then gives an orthonormal basis of the invariant subspace associated with the eigenvalues of interest. If individual eigenvectors are needed, the right generalized Schur vectors can be transformed into eigenvector block X by means of the standard Rayleigh–Ritz procedure for the pair (A, B), performed over the subspace spanned by the columns of V. Similarly, Q contains an orthonormal basis of the left invariant subspace and can be used to retrieve the left eigenvectors.

2.2. The "Q-free" partial generalized Schur form. We start by addressing the question of whether it is possible to eliminate Q from decomposition (4) and obtain an eigenvalue-revealing Schur-type form that contains only the right Schur vectors V and triangular factors. Such a form would provide a numerically stable alternative to the eigenvalue decomposition (3). It would also give an opportunity to handle the standard and generalized eigenproblems in a uniform manner. Most importantly, it would allow us to define a numerical scheme that emphasizes computation only of the generalized right Schur vectors, which are often the ones needed in a real application. The left Schur basis Q can be obtained as a by-product of computation.

For example, if R_B is nonsingular, then Q can be eliminated in a straightforward manner by multiplying the bottom equation in (4) by R_B^{-1} and substituting the resulting expression for Q into the top equation. This yields the desirable decomposition $AV = BVM_A$, where $M_A = R_B^{-1}R_A$ is triangular with eigenvalues on the diagonal. Similarly, assuming that R_A is nonsingular, one can obtain $AVM_B = BV$, where $M_B = R_A^{-1}R_B$ is also triangular, with inverted eigenvalues on the diagonal.

However, the above treatment involves inversions of R_A or R_B, and consequently could cause numerical instabilities whenever the triangular matrices to be inverted have very small diagonal entries. Moreover, in the presence of both zero and infinite eigenvalues, R_A and R_B are singular and cannot be inverted at all. Therefore, it would be ideal to construct a Q-free Schur form for any regular (A, B) that does not rely on the inverse of the triangular matrices R_A and R_B.

In the rest of the section, we show that such a "Q-free" partial Schur form is possible. We start with the following lemma.

LEMMA 1. *Let R_A and R_B be upper triangular matrices such that $|R_A(j,j)| + |R_B(j,j)| \neq 0$ for all $1 \leq j \leq k$. Then for any upper triangular G_1 and G_2, and*

any $\tau_j \in \mathbb{C}$*, such that*

$$G_1(j,j) = \begin{cases} \tau_j, & |R_A(j,j)| < |R_B(j,j)|, \\ \dfrac{1-\tau_j R_B(j,j)}{R_A(j,j)} & \text{otherwise} \end{cases} \tag{5}$$

and

$$G_2(j,j) = \begin{cases} \dfrac{1-\tau_j R_A(j,j)}{R_B(j,j)}, & |R_A(j,j)| < |R_B(j,j)|, \\ \tau_j & \text{otherwise} \end{cases} \tag{6}$$

the matrix $G = R_A G_1 + R_B G_2$ *is upper triangular with* $G(j,j) = 1$ *for all* $1 \le j \le k$.

Proof. The matrix G is a combination of upper triangular matrices and, hence, is upper triangular. For each diagonal entry of G, we have $G(j,j) = R_A(j,j)G_1(j,j) + R_B(j,j)G_2(j,j)$, where $R_A(j,j)$ and $R_B(j,j)$ do not equal zero simultaneously. It is then easy to check that for any $\tau_j \in \mathbb{C}$, (5) and (6) satisfy $R_A(j,j)G_1(j,j) + R_B(j,j)G_2(j,j) = 1$ for all $1 \le j \le k$. ∎

Clearly, the choice of G_1 and G_2 in (5) and (6) is not unique, because it depends on the value of the parameter τ_j and allows freedom in the choice of entries above diagonal in G_1 and G_2. As we will discuss below, in practice, the values τ_j are properly fixed and G_1, G_2 are chosen to be diagonal.

The following theorem gives the existence of a "Q-free" generalized Schur form.

THEOREM 2. *For any regular pair* (A,B) *there exists a matrix* $V \in \mathbb{C}^{n\times k}$ *with orthonormal columns and upper triangular matrices* $M_A, M_B \in \mathbb{C}^{k\times k}$ *such that*

$$AVM_B = BVM_A, \tag{7}$$

and $\lambda_j = M_A(j,j)/M_B(j,j)$ *are (possibly infinite) eigenvalues of* (A,B).

Proof. We start from the generalized Schur decomposition (4). Let G_1 and G_2 be upper triangular with diagonal elements defined by (5) and (6). Right-multiplying both sides of the top and bottom equalities in (4) by G_1 and G_2, respectively, and then summing up the results gives an equivalent system

$$\begin{cases} AV &= QR_A, \\ AVG_1 + BVG_2 &= QG, \end{cases} \tag{8}$$

where

$$G = R_A G_1 + R_B G_2.$$

Since (A,B) is a regular pair, $R_A(j,j)$ and $R_B(j,j)$ do not equal zero simultaneously for the same j. Thus, by Lemma 1, G is upper triangular with $G(j,j) = 1$ for all j. In particular, the latter implies that the inverse of G exists. Hence, we can eliminate Q from (8).

It follows from the bottom equality in (8) that $Q = (AVG_1 + BVG_2)G^{-1}$. Substituting it into the top identity gives $AV = (AVG_1 + BVG_2)G^{-1}R_A$, which implies (7) with

$$M_A = G_2 G^{-1} R_A, \quad M_B = I - G_1 G^{-1} R_A. \tag{9}$$

Furthermore, since the diagonal entries of G are all ones, $G^{-1}(j,j) = 1$ for all j. Thus, from (9), the diagonal elements of M_A and M_B are related to those of R_A and R_B by

$M_A(j,j) = G_2(j,j)R_A(j,j)$ and $M_B(j,j) = 1 - G_1(j,j)R_A(j,j)$. In particular, using definitions (5) and (6) of $G_1(j,j)$ and $G_2(j,j)$, if $|R_A(j,j)| < |R_B(j,j)|$, we obtain

$$M_A(j,j) = \frac{1-\tau_j R_A(j,j)}{R_B(j,j)} R_A(j,j), \quad M_B(j,j) = 1 - \tau_j R_A(j,j), \tag{10}$$

which implies that $M_A(j,j)/M_B(j,j) = R_A(j,j)/R_B(j,j) \equiv \lambda_j$, provided that τ_j is chosen such that $1 - R_A(j,j)\tau_j \neq 0$.

Similarly, if $|R_A(j,j)| \geq |R_B(j,j)|$,

$$M_A(j,j) = \tau_j R_A(j,j), \quad M_B(j,j) = \tau_j R_B(j,j). \tag{11}$$

Hence, by choosing $\tau_j \neq 0$, we get $M_A(j,j)/M_B(j,j) = R_A(j,j)/R_B(j,j) \equiv \lambda_j$. ∎

Given the traditional generalized Schur decomposition (4), Theorem 2 suggests a simple approach for obtaining the "Q-free" form (7). Namely, one starts with setting up the upper triangular matrices G_1, G_2, and G defined according to Lemma 1 and then applies (9) to evaluate M_A and M_B. The right Schur basis V is the same in (4) and (7). Note that the definition (9) of the factors M_A and M_B does not rely on inverting R_A or R_B and only requires an inverse of a triangular matrix G which has all ones on the diagonal. Hence, it avoids inverting (nearly) singular triangular matrices. Additionally, the proposed formulas (5) and (6), as well as (10), always have the largest modulus number between $R_A(j,j)$ and $R_B(j,j)$ in the denominator, which mitigates potential numerical issues introduced by the small diagonal elements of R_A and R_B.

Decomposition (7) is not unique, as it depends on the choice of G_1 and G_2. Therefore, in order to fix particular M_A and M_B, in practice, we choose G_1 and G_2 to be diagonal and set $\tau_j = 0$ if $|R_A(j,j)| < |R_B(j,j)|$ and $\tau_j = 1$ otherwise. This yields the diagonal matrices

$$G_1(j,j) = \begin{cases} 0, & |R_A(j,j)| < |R_B(j,j)|, \\ \dfrac{1 - R_B(j,j)}{R_A(j,j)} & \text{otherwise} \end{cases} \tag{12}$$

and

$$G_2(j,j) = \begin{cases} 1/R_B(j,j), & |R_A(j,j)| < |R_B(j,j)|, \\ 1 & \text{otherwise} \end{cases} \tag{13}$$

for which, by (10), $M_A(j,j) = R_A(j,j)/R_B(j,j)$, $M_B(j,j) = 1$ if $|R_A(j,j)| < |R_B(j,j)|$, and, by (11), $M_A(j,j) = R_A(j,j)$, $M_B(j,j) = R_B(j,j)$ otherwise. Note that it is also possible to choose τ_j that give M_A and M_B such that $|M_A(j,j)|^2 + |M_B(j,j)|^2 = 1$, but our implementation follows the simple formulas (12) and (13).

3. The GPLHR algorithm for computing a partial Schur form. For *Hermitian* problems, a class of powerful eigensolvers is based on preconditioned block iterations; see, e.g., [19, 32, 46]. Such methods fit into a unified framework consisting of two key ingredients: generation of the trial (or search) subspace and extraction of the approximate eigenvectors. Namely, given a number of eigenvector approximations and a *preconditioner*, at each step i, a low-dimensional trial subspace $\mathcal{Z}^{(i)}$ is constructed. If properly chosen, $\mathcal{Z}^{(i)}$ contains improved approximations to the wanted eigenvectors, which are extracted from the subspace using a suitable projection procedure and are then used to start a new iteration. In this section, we extend this framework to the non-Hermitian case.

The results of section 2.2 are central to our derivations. In particular, instead of the standard partial generalized Schur decomposition (4), we focus on the "Q-free" form (7) and seek approximation of the right Schur vectors V and the associated triangular factors M_A and M_B. As we shall see, the approximation to the left Schur vectors Q, as well as to triangular matrices R_A and R_B, can easily be obtained as a by-product of the proposed scheme.

3.1. Construction of the trial subspace. Let $V^{(i)}$, $M_A^{(i)}$, and $M_B^{(i)}$ be approximations to V, M_A, and M_B in (2.2) at an iteration i such that $V^{(i)}$ has k orthonormal columns, and $M_A^{(i)}$ and $M_B^{(i)}$ are k-by-k upper triangular. In order to define a trial subspace that can be used for searching a new approximation $V^{(i+1)}$, we adopt the following subspace orthogonal correction viewpoint.

We are interested in constructing a subspace $\mathcal{K}^{(i)}$ that provides a good representation of the correction $C^{(i)}$ such that

$$A(V^{(i)}+C^{(i)})(M_B^{(i)}+\Delta_B^{(i)}) = B(V^{(i)}+C^{(i)})(M_A^{(i)}+\Delta_A^{(i)}), \tag{14}$$

where $V^{(i)*}C^{(i)} = 0$, and $\Delta_A^{(i)}$, $\Delta_B^{(i)}$ are corrections of the triangular factors $M_A^{(i)}$ and $M_B^{(i)}$, respectively. Once the "correction subspace" $\mathcal{K}^{(i)}$ is determined, the trial subspace can be defined as the subspace sum $\mathcal{Z}^{(i)} = \mathrm{col}\{V^{(i)}\} + \mathcal{K}^{(i)}$, where $\mathrm{col}\{V^{(i)}\}$ denotes the column space of $V^{(i)}$.

After rearranging the terms in (14), we get

$$AC^{(i)}M_B^{(i)} - BC^{(i)}M_A^{(i)} = -(AV^{(i)}M_B^{(i)} - BV^{(i)}M_A^{(i)}) + BV\Delta_A^{(i)} - AV\Delta_B^{(i)},$$

where $V = V^{(i)} + C^{(i)}$ is the matrix of the exact right Schur vectors. Then, using (4), we arrive at the identity

$$AC^{(i)}M_B^{(i)} - BC^{(i)}M_A^{(i)} = -(AV^{(i)}M_B^{(i)} - BV^{(i)}M_A^{(i)}) + Q(R_B\Delta_A^{(i)} - R_A\Delta_B^{(i)}), \tag{15}$$

where Q is the matrix of the exact left Schur vectors. Let $P_{Q^\perp} = I - Q^{(i)}Q^{(i)*}$ be an orthogonal projector onto $\mathrm{col}\{Q^{(i)}\}^\perp$, where $Q^{(i)}$ is a matrix whose orthonormal columns represent a basis of $\mathrm{col}\{\delta_B AV^{(i)} + \delta_A BV^{(i)}\}$ for some scalars $\delta_A, \delta_B \in \mathbb{C}$ such that $|\delta_A| + |\delta_B| \neq 0$. Then applying $P_{Q^\perp}$ to both sides of (15) and neglecting the high-order term $(I - Q^{(i)}Q^{(i)*})Q(R_B\Delta_A^{(i)} - R_A\Delta_B^{(i)})$ gives the equation

$$(P_{Q^\perp}AP_{V^\perp})CM_B^{(i)} - (P_{Q^\perp}BP_{V^\perp})CM_A^{(i)} = -P_{Q^\perp}(AV^{(i)}M_B^{(i)} - BV^{(i)}M_A^{(i)}), \tag{16}$$

whose solution C, constrained to satisfy $V^{(i)*}C = 0$, provides an approximation to the desired correction $C^{(i)}$ of (14). Here, $P_{V^\perp} = I - V^{(i)}V^{(i)*}$ is an orthogonal projector onto $\mathrm{col}\{V^{(i)}\}^\perp$ and, hence, $C = P_{V^\perp}C$.

Equation (16) represents a *generalized Sylvester equation* [39] and can be viewed as a block extension of the standard single vector JD correction equation [8, 40], where the right-hand side is given by the projected residual of problem (7). Note that a connection between the subspace correction and solution of a Sylvester equation was also observed in [31], though not in the context of the generalized Schur form computation. Thus, a possible option is to (approximately) solve (16) for a block correction C and then set $\mathcal{K}^{(i)} = \mathrm{col}\{C\}$.

However, with this approach, it is not straightforward to solve the matrix equation (16) efficiently. Under certain assumptions, one can reduce (16) to the *standard* Sylvester equation, but this is generally expensive for large problems.

Instead, we do not seek the solution of the generalized Sylvester equation and use a different strategy to generate the correction subspace $\mathcal{K}^{(i)}$. To this end, we consider (16) as an equation of the general form

$$L(C) = F, \quad V^{(i)*}C = 0, \tag{17}$$

where $L(C) \equiv (P_{Q^\perp}AP_{V^\perp})CM_B^{(i)} - (P_{Q^\perp}BP_{V^\perp})CM_A^{(i)}$ and $F \equiv -P_{Q^\perp}(AV^{(i)}M_B^{(i)} - BV^{(i)}M_A^{(i)})$. Since $L(\cdot)$ is a linear operator, standard subspace projection techniques for solving linear systems (see, e.g., [11, 35]) can be formally applied to (17). Thus, we can define a preconditioned Krylov-like subspace for (17). However, instead of using this subspace to solve (17), we place it directly in an eigensolver's trial subspace.

More precisely, let $T_L : \mathrm{col}\{Q^{(i)}\}^\perp \to \mathrm{col}\{V^{(i)}\}^\perp$ be a *preconditioner* for the operator $L(\cdot)$. Then, following the analogy with the Krylov subspace methods for (block) linear systems [11, 12, 35], one can expect that the solution C of (17) can be well represented in the *preconditioned block Krylov subspace*

$$\text{block span}\left\{T_L(F), T_L\left(L(T_L(F)), \ldots, (T_L L)^m(T_L(F))\right)\right\},$$

generated by the operator $T_L(L(\cdot))$ and the starting block $T_L(F) \in \mathbb{C}^{n\times k}$, which contains all blocks of the form $\sum_{\ell=0}^{m}(T_L L)^\ell(T_L(F))G_\ell$, with $G_\ell \in \mathbb{C}^{k\times k}$ [12]. In particular, this implies that each column of C should be searched in

$$\mathcal{K}^{(i)} \equiv \mathcal{K}_{m+1}^{(i)} = \mathrm{col}\{W^{(i)}, S_1^{(i)}, S_2^{(i)}, \ldots, S_m^{(i)}\}, \tag{18}$$

where the blocks $K = [W^{(i)}, S_1^{(i)}, S_2^{(i)}, \ldots, S_m^{(i)}]$ represent what we call the Krylov–Arnoldi sequence, generated by a preconditioned Arnoldi-like procedure for problem (17) or, equivalently, for the generalized Sylvester equation (16). This procedure is stated in Algorithm 1, where the input parameter m determines the subspace size.

Algorithm 1 Preconditioned block Arnoldi-type procedure for (17)

Input: preconditioning operator T_L, the parameter m.
Output: the Krylov–Arnoldi basis $K = [W, S_1, \ldots, S_m]$

1: $l \leftarrow 0$; $W \leftarrow \mathrm{orth}^a(T_L(F))$;
2: **for** $l = 1 \to m$ **do**
3: $\quad S_l \leftarrow T_L(L(S_{l-1}))$; $S_l \leftarrow S_l - W(W^*S_l)$;
4: $\quad$ **for** $j = 1 \to l-1$ **do**
5: $\quad\quad S_l \leftarrow S_l - S_j(S_j^*S_l)$;
6: $\quad$ **end for**
7: $\quad S_l \leftarrow \mathrm{orth}(S_l)$;
8: **end for**
9: Return $K = [W, S_1, \ldots, S_m]$.

[a] Throughout, $\mathrm{orth}(V)$ denotes a procedure for orthonormalizing columns of V.

Thus, given the subspace (18), capturing the orthogonal correction C, the eigensolver's trial subspace $\mathcal{Z}^{(i)}$ can be defined as $\mathrm{col}\{V^{(i)}\} + \mathcal{K}_{m+1}^{(i)}$, i.e.,

$$\mathcal{Z}^{(i)} = \mathrm{col}\{V^{(i)}, W^{(i)}, S_1^{(i)}, S_2^{(i)}, \ldots, S_m^{(i)}\}. \tag{19}$$

Clearly, the Krylov–Arnoldi sequence vectors K, generated by Algorithm 1, represent a system of orthonormal columns that are orthogonal to $V^{(i)}$ due to the choice of the

preconditioner $T_L : \mathrm{col}\{Q^{(i)}\}^\perp \to \mathrm{col}\{V^{(i)}\}^\perp$. Hence, the trial subspace $\mathcal{Z}^{(i)}$ in (19) is spanned by orthonormal vectors, which gives a stable basis.

A desirable feature of the preconditioner T_L for problem (17) (or (16)) is that it should approximate the inverse of the operator $L(\cdot)$. One way to construct such an approximation is to replace the upper triangular matrices $M_A^{(i)}$ and $M_B^{(i)}$ in the expression for $L(\cdot)$ by diagonals $\sigma_A I$ and $\sigma_B I$, respectively, where $\sigma_A/\sigma_B = \sigma$. This results in an approximation of $L(\cdot)$ that is given by the matrix $(I - Q^{(i)}Q^{(i)*})(\sigma_B A - \sigma_A B)(I - V^{(i)}V^{(i)*})$. Thus, one can choose the preconditioner as

$$T_L = (I - V^{(i)}V^{(i)*})T(I - Q^{(i)}Q^{(i)*}), \tag{20}$$

where $T \approx (\sigma_B A - \sigma_A B)^\dagger$. Throughout, we consider T as a preconditioner for eigenvalue problem (1), which should be distinguished from the preconditioner T_L for the generalized Sylvester equation. Note that, in practice, assuming that σ is a finite shift that is different from an eigenvalue, we can set $\sigma_A = \sigma$ and $\sigma_B = 1$ so that $T \approx (A - \sigma B)^{-1}$ is an approximate shift-and-invert operator.

Finally, given the definition (20) of the preconditioner T_L, we can derive explicit expressions for the blocks in the Krylov–Arnoldi sequence generated from Algorithm 1. It is easy to check that

$$W^{(i)} = \mathrm{orth}((I - V^{(i)}V^{(i)*})T(I - Q^{(i)}Q^{(i)*})(AV^{(i)}M_B^{(i)} - BV^{(i)}M_A^{(i)})) \tag{21}$$

and, for $l = 1, \ldots, m$,

$$\begin{array}{rcl} \hat{S}_l^{(i)} & = & (I - V^{(i)}V^{(i)*})T(I - Q^{(i)}Q^{(i)*})(AS_{l-1}^{(i)}M_B^{(i)} - BS_{l-1}^{(i)}M_A^{(i)}), \\ S_l^{(i)} & = & \mathrm{orth}(P_{S^\perp}^{(l-1)}(I - W^{(i)}W^{(i)*})\hat{S}_l^{(i)}), \end{array} \tag{22}$$

where $S_0^{(i)} \equiv W^{(i)}$ and $P_{S^\perp}^{(l-1)}$ is the orthogonal projector onto $\mathrm{col}\{S_1^{(i)}, \ldots, S_{l-1}^{(i)}\}^\perp$,

$$P_{S^\perp}^{(l-1)} = (I - S_{l-1}^{(i)}S_{l-1}^{(i-1)*})(I - S_{l-2}^{(i)}S_{l-2}^{(i-1)*}) \cdots (I - S_1^{(i)}S_1^{(i-1)*}). \tag{23}$$

We remark that the above construction of the trial subspace (19) is similar in spirit to that used to devise the BIFP Krylov subspace methods in [10, 32] for computing extreme eigenpairs of Hermitian problems. The conceptual difference, however, is that we put the block correction equation (16) in the center stage, whereas the BIFP methods are based on Krylov(-like) subspaces delivered by a simultaneous solution of problems $(A - \sigma_j B)w = r$ for k different shifts σ_j. In particular, this allows us to discover the projectors $I - V^{(i)}V^{(i)*}$ and $I - Q^{(i)}Q^{(i)*}$ in (20). These projectors are then blended into formulas (21)–(23) that define the trial subspace. Their presence has a strong effect on the eigensolver's robustness, as demonstrated in section 5.

3.2. The trial subspace for standard eigenvalue problem. If $B = I$, instead of the generalized form (4), we seek the standard partial Schur decomposition

$$AV = VR, \tag{24}$$

where $V \in \mathbb{C}^{n\times k}$ contains orthonormal Schur vectors and $R \in \mathbb{C}^{k\times k}$ is upper triangular with the wanted eigenvalues of A on its diagonal. It is clear that (24) is a special case of the "Q-free" form (7) with $B = I$, $M_A = R$, and $M_B = I$. Therefore, the derivation of the trial subspace, described in the previous section, is directly applicable here.

In particular, let $V^{(i)}$ be an approximate Schur basis, and let $M_A^{(i)}$ and $M_B^{(i)}$ be the associated upper triangular matrices that approximate R and I, respectively. Then, following the arguments of section 3.1, from (14) with $B = I$, we get

$$AC^{(i)}M_B^{(i)} - C^{(i)}M_A^{(i)} = -(AV^{(i)}M_B^{(i)} - V^{(i)}M_A^{(i)}) + V(\Delta_A^{(i)} - R\Delta_B^{(i)}). \tag{25}$$

This equality is the analogue to (15) for a standard eigenvalue problem. Here, in order to approximate the correction $C^{(i)}$, it is natural to apply the projector $P_{V^\perp} = I - V^{(i)}V^{(i)*}$ to both sides of (25) and neglect the term $(I - V^{(i)}V^{(i)*})V(\Delta_A^{(i)} - R\Delta_B^{(i)})$. As a result, we obtain the equation

$$(P_{V^\perp}AP_{V^\perp})CM_B^{(i)} - CM_A^{(i)} = -P_{V^\perp}(AV^{(i)}M_B^{(i)} - V^{(i)}M_A^{(i)}), \tag{26}$$

where the solution C, which is orthogonal to $V^{(i)}$, approximates the desired $C^{(i)}$. Note that in contrast to the correction equation (16) for the generalized eigenvalue problem, which is based on two projectors $P_{V^\perp}$ and $P_{Q^\perp}$, (26) involves only one projector $P_{V^\perp}$. This is expected, as both $V^{(i)}$ and $Q^{(i)}$approximate the same vectors, V.

Considering (26) as an equation of the general form (17), where $L(C) = (P_{V^\perp}AP_{V^\perp})CM_B^{(i)} - CM_A^{(i)}$ and $F = -P_{V^\perp}(AV^{(i)}M_B^{(i)} - V^{(i)}M_A^{(i)})$, we apply m steps of Algorithm 1 with the preconditioner

$$T_L = (I - V^{(i)}V^{(i)*})T(I - V^{(i)}V^{(i)*}). \tag{27}$$

This yields the trial subspace (19), where

$$W^{(i)} = \mathrm{orth}((I - V^{(i)}V^{(i)*})T(I - V^{(i)}V^{(i)*})(AV^{(i)}M_B^{(i)} - V^{(i)}M_A^{(i)})) \tag{28}$$

and, for $l = 1, \ldots, m$,

$$\begin{array}{rcl} \hat{S}_l^{(i)} & = & (I - V^{(i)}V^{(i)*})T(I - V^{(i)}V^{(i)*})(AS_{l-1}^{(i)}M_B^{(i)} - S_{l-1}^{(i)}M_A^{(i)}), \\ S_l^{(i)} & = & \mathrm{orth}(P_{S^\perp}^{(l-1)}(I - W^{(i)}W^{(i)*})\hat{S}_l^{(i)}). \end{array} \tag{29}$$

As before, we assume that $S_0^{(i)} \equiv W^{(i)}$ in (29) and that $P_{S^\perp}^{(l-1)}$ is the orthogonal projector defined in (23); the preconditioner T in (27) is chosen as an approximation of the shift-and-invert operator $(A - \sigma I)^{-1}$.

3.3. The harmonic Schur–Rayleigh–Ritz procedure. Let $\mathcal{Z}^{(i)}$ be a trial subspace of dimension $s = (m+2)k$ at iteration i defined by (19) with (21)–(23). We try to find a new orthonormal approximate Schur basis $V^{(i+1)}$ and the corresponding k-by-k upper triangular matrices $M_A^{(i+1)}$ and $M_B^{(i+1)}$ such that each column of $V^{(i+1)}$ belongs to $\mathcal{Z}^{(i)}$ and $M_A^{(i+1)}, M_B^{(i+1)}$ approximate the triangular factors in (7).

To fulfill this task, we use a harmonic Rayleigh–Ritz projection [26, 29], adapted to the case of the "Q-free" Schur form (7). Specifically, we let $\mathcal{U}^{(i)} = (A - \sigma B)\mathcal{Z}^{(i)}$ be a *test subspace*, where we assume that the target shift σ is not an eigenvalue. Then the approximations $V^{(i+1)}$, $M_A^{(i+1)}$, and $M_B^{(i+1)}$ can be determined by requiring each column of the residual of (7) to be orthogonal to this test subspace, i.e.,

$$AV^{(i+1)}M_B^{(i+1)} - BV^{(i+1)}M_A^{(i+1)} \perp \mathcal{U}^{(i)}. \tag{30}$$

This yields the projected problem

$$(U^*AZ)YM_B^{(i+1)} = (U^*BZ)YM_A^{(i+1)}, \tag{31}$$

where $Y \in \mathbb{C}^{s\times k}$, $M_A^{(i+1)}, M_B^{(i+1)} \in \mathbb{C}^{k\times k}$ are unknown, and $Z, U \in \mathbb{C}^{n\times s}$ contain orthonormal columns spanning $\mathcal{Z}^{(i)}$ and $\mathcal{U}^{(i)}$, respectively. Once we have computed Y, $M_A^{(i+1)}$, and $M_B^{(i+1)}$, approximations to the Schur vectors, determined by (30), are given by $V^{(i+1)} = ZY$.

Let us now consider the solution of the projected problem (31). We first show that the projected matrix pair (U^*AZ, U^*BZ) is regular.

PROPOSITION 3. *Let Z, $U = \mathrm{orth}((A-\sigma B)Z) \in \mathbb{C}^{n\times s}$ contain orthonormal basis vectors of the trial and test subspaces, and assume that σ is different from any eigenvalue of (A, B). Then the projected pair (U^*AZ, U^*BZ) is regular.*

Proof. Assume, on the contrary, that (U^*AZ, U^*BZ) is singular. Then, by definition of a singular matrix pair, regardless of the choice of σ,

$$\det(U^*AZ - \sigma U^*BZ) = 0. \tag{32}$$

Since U is an orthonormal basis of $\mathrm{col}\{(A-\sigma B)Z\}$, we can write $(A-\sigma B)Z = UF$, where F is an s-by-s square matrix. This matrix is nonsingular, i.e., $\det(F) \neq 0$, because $A-\sigma B$ is nonsingular. Hence, $U = (A-\sigma B)ZF^{-1}$, and

$$\begin{aligned} U^*AZ - \sigma U^*BZ &= F^{-*}Z^*(A-\sigma B)^*AZ - \sigma F^{-*}Z^*(A-\sigma B)^*BZ \\ &= F^{-*}Z^*(A-\sigma B)^*(A-\sigma B)Z. \end{aligned}$$

Thus, from the above equality and (32), we have

$$0 = \det(U^*AZ - \sigma U^*BZ) = \det(F^{-*})\det(Z^*(A-\sigma B)^*(A-\sigma B)Z).$$

Since $\det(F) \neq 0$, this identity implies that $\det(Z^*(A-\sigma B)^*(A-\sigma B)Z) = 0$. But the matrix $Z^*(A-\sigma B)^*(A-\sigma B)Z$ is Hermitian positive semidefinite and its singularity implies that $\det(A-\sigma B) = 0$, which contradicts the assumption that $\sigma \notin \Lambda(A, B)$. ∎

Thus, problem (31) is of the form (7) and the matrix pair (U^*AZ, U^*BZ) is regular whenever σ is not an eigenvalue of (A, B). Therefore, (31) can be solved by the approach described in section 2.2.

Specifically, one first employs the standard sorted QZ algorithm [24] to compute the full generalized Schur form of (U^*AZ, U^*BZ). This will produce two unitary matrices $\bar{Y}_L, \bar{Y}_R \in \mathbb{C}^{s\times s}$ of the left and right generalized Schur vectors, along with the upper triangular $\bar{R}_A, \bar{R}_B \in \mathbb{C}^{s\times s}$, ordered in such a way that the ratios $\bar{R}_1(j,j)/\bar{R}_2(j,j)$ of the first k diagonal elements are closest to σ, and the corresponding Schur vectors are located at the k leading columns of $\bar{Y}_L$ and $\bar{Y}_R$. Then we let $Y_L = \bar{Y}_L(:,1{:}k), Y_R = \bar{Y}_R(:,1{:}k) \in \mathbb{C}^{s\times k}$ and obtain the desired Schur basis in (31) as $Y = Y_R$. The triangular factors are given as $M_A^{(i+1)} = G_2G^{-1}\tilde{R}_A$ and $M_B^{(i+1)} = I - G_1G^{-1}\tilde{R}_A$, where $G = \tilde{R}_AG_1 + \tilde{R}_BG_2$, $\tilde{R}_A \equiv \bar{R}_A(1{:}k,1{:}k)$ and $\tilde{R}_B \equiv \bar{R}_B(1{:}k,1{:}k)$, and G_1, G_2 are diagonal matrices defined by (12) and (13) with R_A, R_B replaced with $\tilde{R}_A$, $\tilde{R}_B$, respectively.

As a result, the described extraction approach, which we call a harmonic Schur–Rayleigh–Ritz (SRR) procedure, constructs a new basis $V^{(i+1)} = ZY$ of the approximate Schur vectors (the harmonic Schur–Ritz vectors) and the upper triangular matrices $M_A^{(i+1)}$, $M_B^{(i+1)}$ such that the ratios of their diagonal entries are approximations to the desired eigenvalues. Note that in the case of a standard eigenvalue problem the proposed use of formulas (12) and (13) for evaluating the matrices $M_A^{(i+1)}$ and $M_B^{(i+1)}$ indeed guarantees that they converge to R and I, as $\tilde{R}_A$ and $\tilde{R}_B$ get closer to R and I, respectively.

Although both the construction of the trial subspace and the presented harmonic SRR procedure aim at finding the factors V, M_A, and M_B of the "Q-free" form (7), it is easy to see that, as a by-product, the scheme can also produce approximations to the left Schur vectors Q and the upper triangular factors R_A, R_B of the conventional generalized Schur decomposition (4) if $B \neq I$. Specifically, the former is given by $Q^{(i+1)} = UY_L$, whereas the latter correspond to $\tilde{R}_A$ and $\tilde{R}_B$.

Note that the computed $Q^{(i+1)}$ can be conveniently exploited at iteration $i+1$ to construct the new test subspace $\mathcal{U}^{(i+1)} = (A-\sigma B)\mathcal{Z}^{(i+1)}$ and, if $B \neq I$, define the projector $P_{Q^\perp}$, because $Q^{(i+1)}$ represents an orthonormal basis of $\mathrm{col}\{(A-\sigma B)V^{(i+1)}\}$.

PROPOSITION 4. *Let* Z, $U = \mathrm{orth}((\mathrm{A}-\sigma\mathrm{B})\mathrm{Z}) \in \mathbb{C}^{\mathrm{n}\times\mathrm{s}}$ *contain orthonormal basis vectors of the trial and test subspaces. Assume that* $V = ZY_R$, $Q = UY_L \in \mathbb{C}^{n\times k}$ *are the approximate right and left generalized Schur vectors resulting from the harmonic SRR procedure, along with the upper triangular factors* $\tilde{R}_A \equiv \bar{R}_A(\mathtt{1:k,1:k})$ *and* $\tilde{R}_B \equiv \bar{R}_B(\mathtt{1:k,1:k})$. *Then*

$$(A-\sigma B)V = Q(\tilde{R}_A - \sigma\tilde{R}_B);$$

i.e., $Q = UY_L$ *represents an orthonormal basis of* $\mathrm{col}\{(A-\sigma B)V\}$.

Proof. Since UU^* is an orthogonal projector onto $\mathrm{col}\{(A-\sigma B)Z\}$ and $V = ZY_R$, we have

$$(A-\sigma B)V = UU^*(A-\sigma B)ZY_R = U(U^*AZ)Y_R - \sigma U(U^*BZ)Y_R.$$

But $(U^*AZ)Y_R = Y_L\tilde{R}_A$ and $(U^*BZ)Y_R = Y_L\tilde{R}_B$ (resulting from the QZ factorization of (U^*AZ, U^*BZ)), which, combined with the equality $Q = UY_L$, gives the desired result. ∎

Finally, we remark that another standard way to extract Schur vectors is to choose the test subspace $\mathcal{U}^{(i)}$ to be the same as the trial subspace $\mathcal{Z}^{(i)}$. This yields the standard Rayleigh–Ritz-type procedure. However, the drawback of this approach is that it may not be successful for extracting the *interior* eigenvalues and their Schur vectors [8, 29], unless the preconditioning is sufficiently strong, in which case both projection procedures often deliver similar results. Additionally, the standard Rayleigh–Ritz does not immediately produce approximations of the left Schur vectors $Q^{(i+1)}$ that may be of interest and are needed to define the projector $P_{Q^\perp}$. Therefore, in our algorithmic developments, we rely on the systematic use of the *harmonic* SRR procedure.

3.4. Augmenting the trial subspace. At every iteration i, the construction of the trial subspace $\mathcal{Z}^{(i)}$ starts with the current approximate Schur basis $V^{(i)}$, which is used to determine the remaining blocks in (19). A natural question is whether it is possible to further enhance (19) if additional information is available at the present stage of computation, without incurring any significant extra calculations. Namely, we are interested in defining a block $P^{(i)}$ of additional search directions such that the $(m+3)k$-dimensional subspace

$$\mathcal{Z}^{(i)} = \mathrm{col}\{V^{(i)}, W^{(i)}, S_1^{(i)}, \ldots, S_m^{(i)}, P^{(i)}\} \tag{33}$$

contains better approximations to the desired Schur basis than (19).

When solving Hermitian eigenvalue problems, a common technique to accelerate the eigensolver's convergence is to utilize an approximate solution computed at the previous step. The same idea can be readily applied to the non-Hermitian case. For example, given approximate right generalized Schur vectors $V^{(i-1)}$ from the previous

iteration, we can use them to expand the trial subspace (19) by setting $P^{(i)}$ to $V^{(i-1)}$ or, in an LOBPCG fashion, to a combination of $V^{(i-1)}$ and $V^{(i)}$ such that

$$P^{(i)} = V^{(i)} - V^{(i-1)}C_V^{(i-1)}, \tag{34}$$

where $C_V^{(i-1)}$ is an appropriate matrix [19, 20]. Clearly, in exact arithmetic, both options lead to the same subspace (33).

Unlike the case of Hermitian eigenvalue problems, where utilizing approximation from the previous step often leads to a significant acceleration of the eigensolver's convergence, our numerical experience suggests a different picture for non-Hermitian problems. While in many cases the use of the additional search directions based on $V^{(i-1)}$ indeed lead to a noticeable improvement in convergence, we also observe situations in which no or only minor benefit can be seen.

A better approach for augmenting (19) is to define $P^{(i)}$ as a block of k extra approximate Schur vectors. Specifically, if $\bar{Y}_R$ contains the full (ordered) right Schur basis of the projected pair (U^*AZ, U^*BZ), where U and Z are orthonormal bases of the test and trial subspaces $\mathcal{U}^{(i-1)}$ and $\mathcal{Z}^{(i-1)}$, respectively, then we let

$$P^{(i)} \equiv V_{k+1:2k}^{(i)} = Z\bar{Y}_R(:, \texttt{k+1:2k}), \tag{35}$$

where $Y_R(:, \texttt{k+1:2k})$ denotes a submatrix of Y_R lying in the columns $k+1$ through $2k$. As we demonstrate numerically in section 5, this choice generally leads to a better augmented subspace (33), significantly accelerating and stabilizing the convergence. Therefore, we use it by default in our implementation.

The use of formulation (35) can be viewed as a *thick restart* [25, 40, 42, 47], which retains more (harmonic) Ritz approximations than needed after collapsing the trial subspace. In particular, it suggests that GPLHR should transfer $2k$ approximate Schur vectors from one iteration to another. These Schur vectors correspond to the $2k$ approximate eigenvalues closest to σ. The k leading approximations then constitute the block $V^{(i)}$, whereas the k remaining ones are placed in $P^{(i)}$.

3.5. The GPLHR algorithm. We are now ready to combine the developments of the previous sections and introduce the GPLHR algorithm.

Starting with an initial guess $V^{(0)}$, at each iteration i, GPLHR constructs a trial subspace (33) using the preconditioned Krylov–Arnoldi sequence (21)–(23) if $B \neq I$, or (28)–(29) otherwise, and performs the harmonic SRR procedure to extract an updated set of Schur vector approximations with the corresponding upper triangular factors. The detailed description is summarized in Algorithm 2.

An attractive feature of Algorithm 2 is that it is based only on block operations, which enables efficient level-3 BLAS routines for basic dense linear algebra, as well as utilization of sparse matrix times block (matblock) kernels. The method also provides a unified treatment of standard and generalized eigenproblems. In the latter case, it also allows handling infinite eigenvalues if the chosen shift σ is sufficiently large.

Some caution should be exercised when choosing the shift σ. It is generally desirable that σ is not very close to λ to avoid numerical instability. An optimal choice of σ is outside the scope of this paper.

In the case of a generalized eigenproblem, a possible way to determine convergence is based on assessing column norms of the blocks W_A and W_B in step 7. These blocks represent natural residuals of the generalized Schur form (4). Hence, the convergence of j initial columns $v_1, \ldots, v_j$ and $q_1, \ldots, q_j$ in V and Q can be declared if $\|W_A(:, \texttt{j})\|^2 + \|W_B(:, \texttt{j})\|^2$ is sufficiently small. If $B = I$, a similar approach, based on examining the norms of the column of the residual $AVM_B - VM_A$, can be used.

Algorithm 2 The GPLHR algorithm

Input: A regular pair (A, B) of n-by-n matrices, shift $\sigma \in \mathbb{C}$ different from any eigenvalue of (A, B), preconditioner T, starting guess of the Schur vectors $V^{(0)} \in \mathbb{C}^{n\times k}$, and the subspace expansion parameter m.

Output: If $B \neq I$, then approximate generalized Schur vectors $V, Q \in \mathbb{C}^{n\times k}$ and the associated upper triangular matrices $R_A, R_B \in \mathbb{C}^{k\times k}$ in (4) such that $\lambda_j = R_A(j,j)/R_B(j,j)$ are the k eigenvalues of (A, B) closest to σ. If $B = I$, then the Schur vectors $V \in \mathbb{C}^{n\times k}$ and the associated triangular matrix $R \in \mathbb{C}^{k\times k}$ in (24) such that $\lambda_j = R(j,j)$ are the k eigenvalues of A closest to σ.

1: $V \leftarrow \texttt{orth}(V^{(0)})$; $Q \leftarrow \texttt{orth}((A-\sigma B)V)$; $P \leftarrow [\,]$;
2: $[R_A, R_B, Y_L, Y_R] \leftarrow \text{ordqz}(Q^*AV, Q^*BV, \sigma)$[a]; $V \leftarrow VY_R$; $Q \leftarrow QY_L$;
3: Set k-by-k diagonal matrices G_1 and G_2 by (12) and (13); $G \leftarrow R_AG_1 + R_BG_2$.
4: $M_A \leftarrow G_2G^{-1}R_A$; $M_B \leftarrow I - G_1G^{-1}R_A$;
5: **while** convergence not reached **do**
6: **if** $B \neq I$ **then**
7: $W_A \leftarrow AV - QR_A$; $W_B \leftarrow BV - QR_B$;
8: $W \leftarrow (I - VV^*)T(I - QQ^*)(W_AM_B - W_BM_A)$;
9: **else**
10: $W \leftarrow (I - VV^*)T(I - VV^*)(AVM_B - VM_A)$;
11: **end if**
12: $W \leftarrow \text{orth}(W)$; $S_0 \leftarrow W$; $S \leftarrow [\,]$;
13: **for** $l = 1 \to m$ **do**
14: **if** $B \neq I$ **then**
15: $S_l \leftarrow (I - VV^*)T(I - QQ^*)(AS_{l-1}M_B - BS_{l-1}M_A)$;
16: **else**
17: $S_l \leftarrow (I - VV^*)T(I - VV^*)(AS_{l-1}M_B - S_{l-1}M_A)$;
18: **end if**
19: $S_l \leftarrow S_l - W(W^*S_l)$; $S_l \leftarrow S_l - S(S^*S_l)$; $S_l \leftarrow \text{orth}(S_l)$; $S \leftarrow [S\ S_l]$;
20: **end for**
21: $P \leftarrow P - V(V^*P)$; $P \leftarrow P - W(W^*P)$; $P \leftarrow P - S(S^*P)$; $P \leftarrow \text{orth}(P)$;
22: Set the trial subspace $Z \leftarrow [V,\ W,\ S_1, \ldots,\ S_m,\ P]$;
23: $\hat{Q} \leftarrow \texttt{orth}((A-\sigma B)[W,\ S_1,\ \ldots,\ S_m, P])$; $\hat{Q} \leftarrow \hat{Q} - Q(Q^*\hat{Q})$;
24: Set the test subspace $U \leftarrow [Q, \hat{Q}]$; $[\bar{R}_A, \bar{R}_B, \bar{Y}_L, \bar{Y}_R] \leftarrow \text{ordqz}(U^*AZ, U^*BZ, \sigma)$;
25: $Y_R \leftarrow \bar{Y}_R(\texttt{:,1:k})$, $Y_L \leftarrow \bar{Y}_L(\texttt{:,1:k})$, $R_A \leftarrow \bar{R}_A(\texttt{1:k,1:k})$, $R_B \leftarrow \bar{R}_B(\texttt{1:k,1:k})$;
26: $P \leftarrow WY_W + S_1Y_{S_1} + \cdots + S_mY_{S_m} + PY_P$, where $Y_R \equiv [Y_V^T,\ Y_W^T,\ Y_{S_1}^T,\ \ldots,\ Y_{S_m}^T, Y_P^T]^T$ is a conforming partitioning of Y_R;
27: $V \leftarrow VY_V + P$; $Q \leftarrow UY_L$;
28: $P \leftarrow Z\bar{Y}_R(\texttt{:,k+1:2k})$;[b]
29: Set k-by-k diagonal matrices G_1 and G_2 by (12) and (13); $G \leftarrow R_AG_1 + R_BG_2$;
30: $M_A \leftarrow G_2G^{-1}R_A$; $M_B \leftarrow I - G_1G^{-1}R_A$;
31: **end while**
32: If $B = I$, then $R \leftarrow M_B^{-1}M_A$.

[a] $[R_A, R_B, Y_L, Y_R] \leftarrow \text{ordqz}(\Phi, \Psi, \sigma)$ computes the full generalized Schur decomposition for an input pair (Φ, Ψ), ordered to ensure that $|R_A(i,i)/R_B(i,i) - \sigma| \leq |R_A(j,j)/R_B(j,j) - \sigma|$ for $i < j$.

[b] This step can be disabled if the LOBPCG-style search direction of step 26 is preferred.

One can observe that the block W, formed in step 8 of the algorithm, equals $(I - VV^*)T(I - QQ^*)(AVM_B - BVM_A)$, because, by definition of W_A and W_B,

$$\begin{aligned}(I - QQ^*)(W_AM_B - W_BM_A) &= (I - QQ^*)((AV - QR_A)M_B - (BV - QR_B)M_A)\\ &= (I - QQ^*)(AVM_B - BVM_A).\end{aligned}$$

Thus, W is indeed the projected preconditioned residual of problem (7).

The GPLHR algorithm performs $(m+1)$ multiplications of A and B with a block of k vectors per iteration. In addition to storing Z and U, it requires AZ and BZ to be stored. The size of these blocks, which is $s = (m+2)k$ for (19) and $s = (m+3)k$ for (33), depends on m. There is a tradeoff between faster convergence which can be achieved by having a larger m and memory usage. In most of our numerical experiments, we use $m = 1$ to keep memory usage low.

A common feature of block iterations is that some approximations can converge faster than others. If some columns $v_1, \dots, v_j$ and $q_1, \dots, q_j$ in V and Q have converged, they can be "locked" using the "soft locking" strategy (see, e.g., [20]), the same way as done for Hermitian eigenproblems. With this approach, in subsequent iterations, one retains the converged vectors in V and Q but removes the corresponding columns from the blocks $W, S_1, \dots, S_m$ and P. Note that locking of the Schur vectors in Algorithm 2 should always be performed in order, i.e., v_{j+1} can be locked only after the preceding Schur vectors $v_1, \dots, v_j$ have converged and been locked.

The use of "locking" can have a strong impact on the convergence behavior of the method, especially when a large number of vectors have converged. As locking proceeds, an increasing number of columns of $W, S_1, \dots, S_m$ and P is removed, leading to a shrinkage of the trial subspace. As a result, the trial subspaces can become excessively small, hindering the convergence rate and robustness. In contrast to the Hermitian eigenproblems, the effect of the subspace reduction can be significantly more pronounced in the non-Hermitian case, which generally requires searching in subspaces of a larger dimension.

In order to overcome this problem, we propose automatically adjusting the parameter m. In our implementation, we increase m to the largest integer such that $m(k-q) \le m_0 k$, where m_0 is the initial value of m and q is the number of converged eigenpairs. Specifically, we recalculate m as $m \leftarrow \min\{\lfloor m_0 k/(k-q) \rfloor, 20\}$.

Note that step 28 of Algorithm 2 is optional. It provides means to switch between the LOBPCG-style search directions (34) and those defined by (35). We use the latter by default.

Finally, we emphasize that Algorithm 2 is capable of computing larger numbers of Schur vectors incrementally, using *deflation*, similar to Hermitian eigenproblems. That is, if V and Q are the computed bases of k right and left generalized Schur vectors, the additional k generalized Schur vectors can be computed by applying GPLHR to the deflated pair $((I-QQ^*)A(I-VV^*), (I-QQ^*)B(I-VV^*))$ if $B \neq I$, or to the matrix $(I-VV^*)A(I-VV^*)$ otherwise. This process can be repeated until the desired number of vectors is obtained.

3.6. Alternative formulas for residual computation. As has been pointed out in section 2.2, the "Q-free" generalized Schur form (7) is not unique. Therefore, the computation of the residual in (21) can, in principle, be based on a different formula rather than the expression $AV^{(i)}M_B^{(i)} - BV^{(i)}M_A^{(i)}$ with $M_A^{(i)}, M_B^{(i)}$ given by (12), (13), and (9). For example, if either (or both) of the upper triangular factors $\tilde{R}_A, \tilde{R}_B$, resulting from the harmonic projection, are nonsingular at all iterations, which is the most common case in the majority of practical computations, then the residual can be formulated as $AV^{(i)}(\tilde{R}_A^{-1}\tilde{R}_B) - BV^{(i)}$ or $AV^{(i)} - BV^{(i)}(\tilde{R}_B^{-1}\tilde{R}_A)$.

All these formulas are asymptotically equivalent, in the sense that the correspondingly defined residuals vanish as $V^{(i)}$ converges to the desired right Schur vectors. One would naturally question whether their choice could affect the performance of GPLHR before it reaches the asymptotic mode. Our extensive numerical experience indicates

that the effect of the choice of the residual formulas is very mild. Other factors, e.g., the quality of the preconditioner, the subspace dimension parameter m, appropriate use of projectors $P_{V^\perp}$ and $P_{Q^\perp}$ for developing the trial subspace, and the choice of the additional search directions $P^{(i)}$, all have much stronger impact on the behavior of GPLHR. Therefore, we always construct the residual as in (21), which is well defined for all regular pencil (A, B). The same considerations apply to the S-vectors (22).

3.7. The GPLHR algorithm for partial eigenvalue decomposition. If eigenvalues of (A, B) are nondeficient and the targeted eigenvectors X are known to be reasonably well conditioned, then one would prefer to tackle the partial eigenvalue decomposition (3) directly, without resorting to the Schur form computations. Such situations, in particular, can arise when the non-Hermitian problem is obtained as a result of some perturbation of a Hermitian eigenproblem, e.g., as in the context of the EOM-CC [13, 14] or complex scaling methods [3, 15, 33] in quantum chemistry.

It is straightforward to apply the described framework to devise an eigenvector-based algorithm for computing the partial decomposition (3). In particular, using exactly the same orthogonal correction argument, we can obtain a version of GPLHR that operates on the approximate eigenvectors rather than Schur vectors. This scheme, which we refer to as GPLHR-EIG, is stated in Algorithm 3 of Appendix A.

4. Preconditioning. The choice of the preconditioner is crucial for the overall performance of the GPLHR methods. While the discussion of particular preconditioning options is outside the scope of this paper, below we briefly address several general points related to a proper setting of the preconditioning procedure.

An attractive feature of the GPLHR algorithms is that the definition of an appropriate preconditioner T is very broad. We only require T to be a form of an approximate inverse of $(\sigma_B A - \sigma_A B)$. Hence, T can be constructed using any available preconditioning technique that makes the solution of the shifted linear systems $(\sigma_B A - \sigma_A B)w = r$ easier to obtain. For a survey of various preconditioning strategies we refer the reader to [4, 35]. Note that, in practice, we expect $T \approx (A - \sigma B)^{-1}$ if eigenvalues of interest are finite and $T \approx B^\dagger$ if infinite eigenvalues are wanted. For example, in the latter case, one can let $T \approx (B + \tau I)^{-1}$, where τ is a small regularization parameter.

If T is not sufficient to ensure the eigensolver's convergence with a small search subspace, it can be used as a preconditioner for an approximate iterative solve of $(\sigma_B A - \sigma_A B)w = r$, e.g., using the GMRES algorithm [37] or IDR(s) [41]. To enhance the performance by fully utilizing BLAS-3 operations, block linear solvers should be used whenever available. Then the preconditioned linear solver itself can be used in place of the operator T to precondition the GPLHR algorithm. Thus, by setting the number of iterations of the linear solver or by adjusting its convergence threshold, one obtains a flexible framework for "refining" the preconditioning quality, which is especially helpful for computing interior eigenvalues.

Finally, if T admits a fast computation, it can be periodically updated in the course of iterations, in such a way that the shift in the preconditioning operator is adjusted according to the current eigenvalue approximations. For example, this strategy was adopted in the GPLHR implementation [50], where T was diagonal.

5. Numerical experiments. The goal of this section is two-fold. First, we would like to demonstrate numerically the significance of several algorithmic components of GPLHR, such as the use of additional search directions $P^{(i)}$ or the application of a projector prior to preconditioning with T. Second, we are interested in compar-

ing GPLHR with a number of existing state-of-the-art eigensolvers for non-Hermitian eigenvalue problems, which includes the block GD (BGD), the JDQR/JDQZ algorithm, and the implicitly restarted Arnoldi method available in ARPACK [23].

TABLE 1
Test problems.

Problem	Type	n	σ	Preconditioner	Spectr. region
A15428	stand.	15428	-40.871	ILU(10^{-3})	interior
AF23560	stand.	23560	$-40+300i$	ILU(10^{-3})	largest mod.
CRY10000	stand.	10000	8.0	ILU(10^{-3})	largest Re
DW8192	stand.	8192	1.0	ILU(10^{-3})	rightmost
LSTAB_NS	gen.	12619	0	GMRES(25)+LSC	rightmost
MHD4800	gen.	4800	$-0.1+0.5i$	$(A-\sigma B)^{-1}$	interior
PDE2961	stand.	2961	10.0	ILU(10^{-3})	largest Re
QH882	stand.	882	$-150+180i$	GMRES(5)+ILU(10^{-5})	interior
RDB3200L	stand.	3200	$2i$	ILU(10^{-3})	rightmost
UTM1700	gen.	1700	0	GMRES(15)+ILU(10^{-4})	leftmost

Table 1 summarizes the test problems considered throughout this section. These test problems originate from a number of real applications [3, 15, 33, 6, 49, 50]. Some of these matrices (AF23560, CRY10000, and DW8192) can be obtained from MATRIX MARKET.[1] All of our tests are performed in MATLAB.

Throughout, unless otherwise explicitly stated, the value of the trial subspace size parameter m is set to 1. As a preconditioner T we use triangular solves with incomplete L and U factors computed by the `ilu` routine of MATLAB with thresholding (denoted by ILU(t), where t is the threshold value). If a more efficient preconditioner is needed, then T corresponds to several steps of the ILU-preconditioned GMRES, further referred to as GMRES(s)+ILU(t), where s is the number of GMRES steps. The only exception is LSTAB_NS, where the action of T is achieved by GMRES(25) preconditioned by an ideal version of the least squares commutator (LSC) preconditioner [7]. The default preconditioners for all test problems are also listed in Table 1.

TABLE 2
Numbers of iterations performed by GPLHR to compute $k=5$ eigenpairs with different choices of the search directions $P^{(i)}$.

Problem	$P^{(i)}_{\text{thick}}$	$P^{(i)}_{\text{LOBPCG}}$	No $P^{(i)}$	Problem	$P^{(i)}_{\text{thick}}$	$P^{(i)}_{\text{LOBPCG}}$	No $P^{(i)}$
A15428	12	14	16	MHD4800	137	DNC	DNC
AF23560	10	DNC	153	PDE2961	13	DNC	DNC
CRY10000	27	DNC	DNC	QH882	15	DNC	30
DW8192	118	55	329	RDB3200L	10	12	12
LSTAB_NS	38	DNC	DNC	UTM1700	16	23	21

5.1. Effects of the additional search directions. In Table 2, we report numbers of iterations performed by the GPLHR methods to compute five eigenpairs with different choices of the search directions $P^{(i)}$. In particular, we compare the default "thick-restarted" option (35) with the LOBPCG-style choice (34), and the variant without $P^{(i)}$, which corresponds to the nonaugmented subspace (19).

[1] http://math.nist.gov/MatrixMarket/

As has already been mentioned, in practice, the quality of approximations produced by GPLHR should be assessed through norms of the Schur residuals. However, in order to facilitate comparisons with other methods, at each step, we transform the current approximate Schur basis $V^{(i)}$ into eigenvector approximations and declare convergence of a given pair (λ, x) using its relative eigenresidual norm. That is, we consider (λ, x) converged, and soft-lock it from further computations, if $\|Ax - \lambda Bx\|/\|Ax\| < 10^{-8}$. Note that, in this case, locking is performed in a contiguous fashion, i.e., eigenvector x_{j+1} can be locked only after the preceding vectors $x_1, \ldots, x_j$ have converged and been locked.

Table 2 clearly demonstrates that the choice $P^{(i)} = V^{(i)}_{k+1:2k}$, denoted by $P^{(i)}_{\text{thick}}$, generally leads to the fastest and most robust convergence behavior of GPLHR. Such a construction of the search directions requires a negligibly low cost, without extra matrix-vector products or preconditioning operations. However, their presence in the trial subspace indeed significantly accelerates and stabilizes the convergence, as can be observed by comparing the corresponding iteration counts with the case where the block $P^{(i)}$ is not appended to the subspace (35). Here, "DNC" denotes cases where the method failed to reduce the relative eigenresidual norm below the tolerance level of 10^{-8} for all eigenpairs within 500 iterations.

Furthermore, it is evident from Table 2 that the choice $P^{(i)}_{\text{thick}}$ generally outperforms the LOBPCG-style formula (34), denoted by $P^{(i)}_{\text{LOBPCG}}$. Specifically, the former always leads to convergence and yields lower iteration counts for all of the test problems, except for DW8192. In our experience, however, this situation is not common, and the use of (35) typically results in a much more robust convergence behavior.

5.2. Expansion of the trial subspace. The size of the GPLHR trial subspace is controlled by the parameter m. Generally, increasing m leads to a smaller iteration count. However, since the expansion of the GPLHR trial subspace requires extra matrix-vector multiplications and preconditioning operations, using larger values of m also increases the cost of each GPLHR step. Therefore, the parameter m should be chosen large enough to ensure stable convergence, but at the same time should be sufficiently small to maintain a relatively low cost of iterations.

In most cases, using larger values of m is counterproductive, as can be seen from Figure 1. The figure shows how the numbers of iterations and preconditioned matrix-vector products change as m grows. If a good preconditioner is at hand, such as ILU(0.001) for AF23560 in the figure, then it is common that the iteration count stabilizes at $m = 1$ or $m = 2$, and therefore further increase of the parameter is not needed, as it leads to redundant computations. However, increasing m can be more effective when less efficient preconditioners are used, e.g., ILU(0.5) in our test. In this case, a relatively poor preconditioner is compensated by a larger trial subspace. In particular, in the AF23560 example with ILU(0.5), the optimal value of m in terms of computational work is 4.

5.3. Effects of projectors. Our derivation of the GPLHR method in section 3 suggested that the preconditioner T should be preceded by application of the projector $(I - Q^{(i)}Q^{(i)*})$ (or $(I - V^{(i)}V^{(i)*})$ if $B = I$) and then followed by the projection $(I - V^{(i)}V^{(i)*})$ (see (20) and (27)). This motivated the occurrence of the corresponding projectors on both sides of T at steps 8 and 15 (if $B \neq I$) and at steps 10 and 17 (if $B = I$) of Algorithm 2.

While the presence of the left projector $I - V^{(i)}V^{(i)*}$ can be viewed as part of the orthogonalization procedure on the trial subspace, and is thus natural, it is of interest

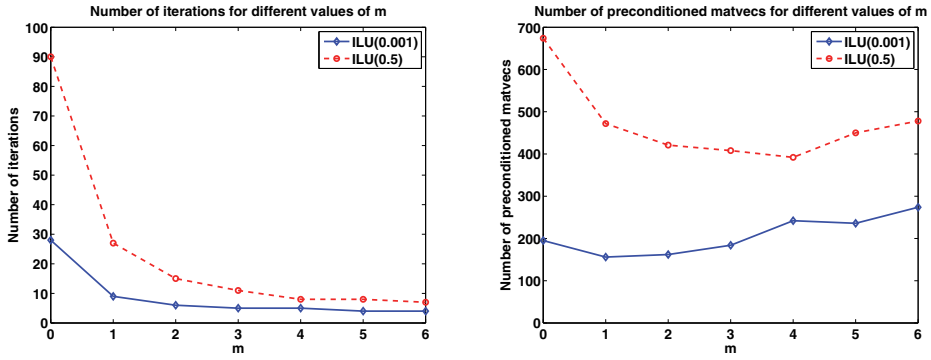

FIG. 1. *Numbers of iterations (left) and preconditioned matrix-vector products (right) performed by GPLHR, preconditioned with* $ILU(0.001)$ *and* $ILU(0.5)$*, to compute* $k = 10$ *eigenpairs of AF*23560 *with different values of the subspace parameter* m*.*

to see whether the action of the right projection, performed before the application of T, has any effect on the GPLHR convergence. For this reason, we compare the GPLHR scheme in Algorithm 2 to its variant where the right projector is omitted. The corresponding iteration counts are reported in Table 3.

TABLE 3

Numbers of iterations performed by GPLHR with and without the projection at the preconditioning step to compute $k = 10$ *eigenpairs.*

Problem	With proj.	W/o proj.	Problem	With proj.	W/o proj.
A15428	14	DNC	MHD4800	12	12
AF23560	9	9	PDE2961	10	11
CRY10000	13	17	QH882	47	43
DW8192	44	45	RDB3200L	12	21
LSTAB_NS	49	84	UTM1700	25	26

It can be observed from Table 3 that, in most of the tests, applying the projector before applying applying T gives a smaller number of iterations. In several cases (A15428, LS_STAB, and RDB3200L), the difference in the iteration count is substantial. The only example where the absence of the projector yields slightly fewer iterations is QH882. However, this only happens for a small value of the parameter m, and the effects of the projection become apparent and favorable as m is increased and the convergence is stabilized.

5.4. Comparison with BGD. We now compare GPLHR to the classic BGD method used to solve non-Hermitian problems [27, 38]. This eigensolver is popular, in particular, in quantum chemistry; see, e.g., [50]. For a fair comparison, in BGD, we apply the harmonic Rayleigh–Ritz procedure [29] to extract approximate eigenvectors and use the same projected preconditioner (20) or (27), depending on whether the eigenproblem is generalized or standard, respectively. In fact, once equipped with the projected preconditioning, BGD can be viewed as a generalization of the JD methods [8, 40] to the block case.

To ensure that both schemes have the same memory constraint, we consider a restarted variant of BGD, where the search subspace is collapsed after reaching the dimension of $(m + 3)k$, with the k current approximate eigenvectors (the harmonic Ritz vectors) used to start the next cycle. We denote this scheme as BGD$((m+3)k)$.

In particular, if $m = 1$, we get BGD($4k$).

Our first experiment, reported in Figure 2, attempts to assess the effect of different preconditioners on the convergence of both methods.

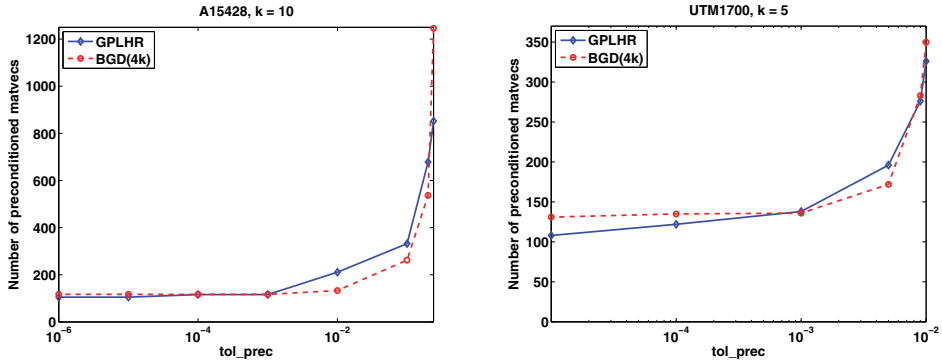

FIG. 2. *Comparison of GPLHR* ($m = 1$) *and BGD*($4k$) *with different preconditioning quality for computing several eigenpairs of A*15428 *(left) and UTM*1700 *(right) closest to* σ. *The preconditioner T corresponds to an application of the ILU-preconditioned GMRES with stopping tolerance tol_prec.*

In order to emulate a variety of preconditioners, we set T to an approximate solve of $(A-\sigma B)w = r$ using the (full) preconditioned GMRES. The quality of T can then be adjusted by varying the GMRES convergence tolerance, denoted by *tol_prec*. Clearly, lower values *tol_prec* correspond to a more effective preconditioner, whereas increasing *tol_prec* leads to its deterioration.

Figure 2 shows how the numbers of (preconditioned) matrix-vector products in GPLHR and BGD($4k$) change with respect to the quality of preconditioning for problems A15428 (left) and UTM1700 (right). The reported preconditioned matrix-vector product (matvec) counts exclude those from the "inner" GMRES iterations. As a GMRES preconditioner we use ILU(10^{-2}) for A15428 and ILU(10^{-4}) for UTM1700.

Figure 2 demonstrates that, under the same memory constraint, GPLHR tends to exhibit better performance if the preconditioner is either reasonably strong (i.e., *tol_prec*$\leq 10^{-3}$ in both plots) or is relatively weak (i.e., *tol_prec* around 10^{-1} on the left plot and around 10^{-2} on the right). At the same time, one can see that BGD($4k$) can outperform GPLHR for an "intermediate" preconditioning quality, i.e., corresponding to *tol_prec* between 10^{-3} and 10^{-1} in the left plot and between 10^{-3} and 0.009 in the right plot of the figure.

In Table 4, we compare the numbers of iterations (#it) and (preconditioned) matvecs (#pmv) performed by GPLHR and BGD($4k$) for all of our test problems with the default preconditioners listed in Table 1. Here, the matvec counts also include those accrued at "inner" GMRES iterations within the preconditioning step.

It can be seen from Table 4 that GPLHR is generally more robust than BGD($4k$). The former was always able to attain the requested level of the solution accuracy, whereas the latter sometimes failed to converge. Furthermore, one can see that in the examples where BGD($4k$) is successful, GPLHR typically performs fewer preconditioned matvecs and, hence, is more efficient, with the only exceptions being the QH882 matrix and, to a lesser extent, RDB3200L.

We should point that BGD can construct a larger subspace to accelerate convergence if there is a sufficient amount of memory. Figure 3 shows that BGD may require around two to three times more storage than GPLHR to achieve a comparable con-

TABLE 4

The numbers of iterations and preconditioned matrix-vector products performed by GPLHR ($m = 1$) and BGD($4k$).

		GPLHR		BGD($4k$)				GPLHR		BGD($4k$)	
Problem	k	#it	#pmv	#it	#pmv	Problem	k	#it	#pmv	#it	#pmv
A15428	3	17	98	DNC	DNC	MHD4800	3	30	158	DNC	DNC
	5	11	100	DNC	DNC		5	170	1062	DNC	DNC
	10	14	239	41	220		10	12	206	79	432
AF23560	3	10	52	DNC	DNC	PDE2961	3	12	70	53	123
	5	10	84	48	118		5	11	98	46	114
	10	9	156	39	232		10	11	178	50	207
CRY10000	3	29	146	142	237	QH882	3	18	588	23	348
	5	24	192	DNC	DNC		5	14	708	22	450
	10	14	245	DNC	DNC		10	52	4332	92	2190
DW8192	3	70	377	DNC	DNC	RDB3200L	3	8	46	16	41
	5	51	474	363	1447		5	9	80	21	76
	10	48	799	450	1808		10	11	180	33	163
LSTAB_NS	3	51	5616	306	8843	UTM1700	3	17	1312	68	1760
	5	37	6552	DNC	DNC		5	16	2080	44	2208
	10	41	15184	DNC	DNC		10	25	6048	125	6704

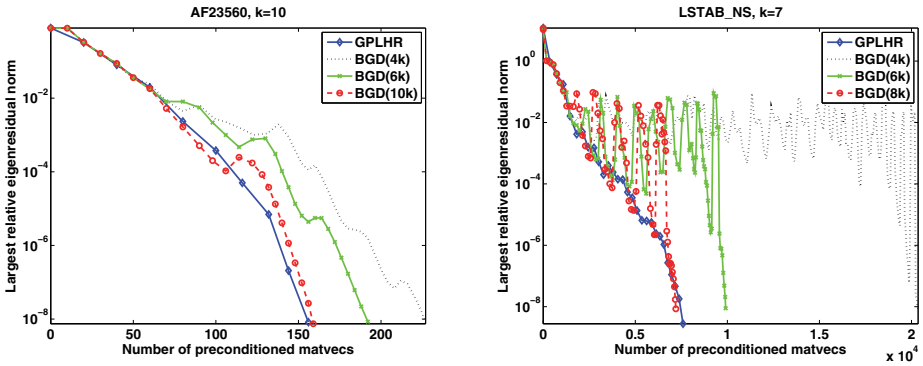

FIG. 3. *Comparison of GPLHR ($m = 1$) and BGD(t) with different values of t.*

vergence rate. Thus, the GPLHR scheme is preferred to BGD in cases where memory is tight. For all the test problems in this example we use the default preconditioners listed in Table 1.

5.5. Comparison with JDQR/JDQZ. As was mentioned in the introduction, the GPLHR method is substantially different from JDQR/JDQZ. The former is based on block iterations, whereas the latter are single vector schemes which compute one eigenpair after another in a sequential fashion. The goal of our experiments below is to point out the advantages of using the block GPLHR iterations for practical large-scale eigenvalue computations. In our comparisons, we use the `jdqr.m` and `jdqz.m` MATLAB implementations of the JDQR/JDQZ algorithms.[2]

As a GPLHR preconditioner T we employ a fixed number of iterations of preconditioned GMRES (it_G) for the system $(A-\sigma B)w = r$, where r is a vector to which the preconditioner T is applied. Throughout, we use the standard single vector GMRES to solve systems corresponding to different right-hand sides separately. However, in practice, it may be more efficient to use block linear solvers for a simultaneous solution of all systems. In the JDQR/JDQZ context, the preconditioner is applied through solving the correction equation. Therefore, in order to match the complexity and quality of the GPLHR preconditioning to the correction equation solve, the solution of the latter is approximated by applying it_G steps of GMRES with the same preconditioner.

[2] http://www.staff.science.uu.nl/~sleij101/

The number it_G is set to the smallest value that yields the convergence of GPLHR.

In our tests, we use the harmonic Rayleigh–Ritz option for eigenvector extraction in JDQR/JDQZ. To ensure the same memory constraint for both methods, we set the maximum size of the JDQR/JDQZ search subspace to $4k$ (the parameter m in GPLHR is set to 1 in all of the runs). After restarting, the JDQR/JDQZ retains $k+5$ harmonic Ritz vectors, which is a default value in `jdqr.m` and `jdqz.m`.

In order to have a similar locking of the converged Schur vectors, we modify the GPLHR stopping criterion to be the same as in `jdqr.m` and `jdqz.m`. Namely, we consider the right Schur vector v_j and an approximate eigenvalue λ_j to be converged only if the preceding vectors $\bar{V} = [v_1, v_2, \ldots, v_{j-1}]$ have converged and the norm of the eigenresidual of the deflated problem $((I - \bar{Q}\bar{Q}^*)A(I - \bar{V}\bar{V}^*), (I - \bar{Q}\bar{Q}^*)B(I - \bar{V}\bar{V}^*))$, evaluated at (λ_j, v_j), is below the given tolerance level, set to 10^{-8} in our tests. Here, the matrix $\bar{Q} = [q_1, q_2, \ldots, q_{j-1}]$ contains the converged left Schur vectors, and $\bar{Q} = \bar{V}$, if $B = I$.

TABLE 5
Comparison of GPLHR and JDQR algorithms for standard eigenproblems.

			GPLHR			JDQR		
Problem	k	it_G	#it	#mv	#prec	#it	#mv	#prec
A15428	1	15	13	417	416	20	321	341
	3	10	16	949	946	45	502	541
	5	10	14	1347	1342	57	642	685
AF23560	1	0	13	27	26	DNC	DNC	DNC
	3	0	14	75	72	50	57	101
	5	0	14	117	112	69	84	139
	10	0	12	214	204	112	147	225
CRY10000	1	3	46	369	368	62	249	311
	3	3	58	1243	1240	182	735	911
	5	4	45	1915	1910	218	1105	1309
	10	3	42	3102	3092	DNC	DNC	DNC
DW8192	1	0	175	351	350	659	660	1318
	3	0	81	423	420	274	281	549
	5	0	59	535	530	270	258	541
	10	0	49	831	821	336	371	673
PDE2961	1	7	48	769	768	DNC	DNC	DNC
	3	0	20	117	114	48	55	97
	5	0	17	149	144	57	72	115
	10	0	15	242	232	86	121	173
QH882	1	5	24	289	288	26	157	183
	3	5	24	783	780	DNC	DNC	DNC
	5	5	17	893	888	41	261	288
	10	5	50	4294	4284	DNC	DNC	DNC
RDB3200L	1	0	31	63	62	54	55	109
	3	0	20	121	118	71	78	143
	5	0	22	213	208	102	117	205
	10	0	20	394	384	183	218	367

In Table 5, we report results for standard eigenproblems, where GPLHR is compared with the JDQR algorithm. The table contains numbers of eigensolver iterations (#it), matrix-vector products (#mv), and applications of the preconditioner (#prec) used within GMRES solves. In all runs, we apply it_G GMRES steps with ILU(10^{-2}) as the GPLHR preconditioner T and as the JD correction equation solve, except for the QH882 problem, where a stronger ILU(10^{-5}) preconditioner is used within GMRES. Note that the zero number it_G means that T is applied through a single

shot of the ILU preconditioner, without the "inner" GMRES steps.

One can see from Table 5 that GPLHR generally performs significantly fewer iterations than JDQR. At the same time, each GPLHR iteration is more expensive, requiring $(m+1)k$ matrix-vector products and preconditioning operations, while JDQR only performs a single matrix-vector multiply plus the correction equation solve per step. As a result, the total number of matrix-vector products and preconditionings is, in most cases, larger for GPLHR, especially if larger numbers k of eigenpairs are wanted. However, generally, the increase is mild, typically around a factor of 2. On the other hand, GPLHR allows for a possibility to group matrix-vector products into a single matrix-block multiply in actual parallel implementation, which can lead to a two to three times speedup (see, e.g., [1]) compared to separate multiplications involving single vectors, as in JDQR. Therefore, we expect that GPLHR schemes will ultimately outperform the JDQR/JDQZ family for large-scale eigenproblems on modern parallel computers. Optimal parallel implementations of GPLHR are, however, outside the scope of this paper and will be the subject of further study.

Note that if $it_G = 0$, JDQR often produces a larger #prec count, i.e., it requires more ILU applications compared to GPLHR. This is because of the way the correction equation is solved in JDQR/JDQZ [8, section 2.6], which requires an extra application of the preconditioner to form the correction equation. Thus, one can expect that GPLHR gives a faster time to solution in cases where preconditioning is expensive.

TABLE 6
Comparison of GPLHR and JDQZ algorithms for generalized eigenproblems.

			GPLHR			JDQZ		
Problem	k	it_G	#it	#mv	#prec	#it	#mv	#prec
MHD4800	1	0	20	41	40	DNC	DNC	DNC
	3	0	15	79	76	DNC	DNC	DNC
	5	0	12	101	96	DNC	DNC	DNC
	10	0	9	165	155	DNC	DNC	DNC
LSTAB_NS	1	25	5	261	260	DNC	DNC	DNC
	3	25	49	5307	5304	DNC	DNC	DNC
	5	25	28	4893	4888	DNC	DNC	DNC
	10	25	36	12984	12974	DNC	DNC	DNC
UTM1700	1	15	5	161	160	MCV	MCV	MCV
	3	15	13	963	960	16	263	280
	5	15	13	1669	1664	31	511	545
	10	15	21	5034	5024	166	2691	2865

Table 6 contains the comparison results for generalized eigenproblems, where GPLHR is compared with the JDQZ algorithm. In MHD4800, we set the GMRES preconditioner to $(A - \sigma B)^{-1}$, in LSTAB_NS to the ideal least squares commutator preconditioner, and in UTM1700 to ILU(10^{-4}).

Table 6 shows that, for a given preconditioner and trial subspace size, JDQZ failed to converge for MHD4800 and LSTAB_NS. Slightly increasing it_G allowed us to restore the JDQZ convergence for MHD4800. However, even with a larger number of it_G, the method misconverged, i.e., missed several wanted eigenpairs and instead returned those that are not among the k closest to σ. For LSTAB_NS, we were not able to restore the convergence by either increasing the number it_G of GMRES iterations or by allowing a larger trial subspace.

We would like to emphasize the remarkable robustness of GPLHR. Given an appropriate preconditioner T, in all of our tests, the method consistently obtained the correct results, without convergence stagnation or misconvergence, as was observed

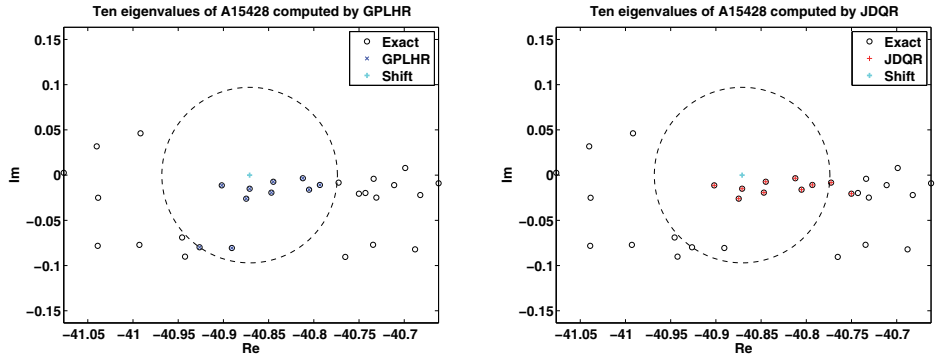

FIG. 4. *Ten eigenvalues of A*15428 *closest to* $\sigma = -40.8710$ *computed by GPLHR (left) and JDQR/JDQZ (right) algorithms.*

in several JDQR/JDQZ runs (in Tables 5 and 6, "DNC" means that the method did not converge and "MCV" denotes misconvergence). The observed robustness is not surprising, as the block methods have traditionally been recommended for properly resolving eigenvalue clusters [30].

We have also noticed that JDQZ/JDQR are more likely to misconverge, i.e., discover incorrect eigenvalues, when a larger number of eigenpairs (say, $k \geq 10$) is needed. Such a situation is demonstrated in Figure 4, where $k = 10$ eigenpairs of A15428 closest to σ are computed by different eigensolvers. It can be seen that GPLHR found all the targeted eigenvalues (within the dashed circle), whereas JDQR missed a few of them. This can possibly be caused by the repeated deflations performed by JDQR/JDQZ, which can accumulate error and lead to an inaccurate solution. As a GPLHR preconditioner T for A15428 we use an approximate solve of $(A-\sigma I)w = r$ by GMRES preconditioned with ILU(10^{-2}) and convergence tolerance (*tol_prec*) of 10^{-3}. The GMRES solve with the same ILU(10^{-2}) preconditioner is applied to the correction equation in JDQR.

5.6. Comparison with ARPACK. In our last set of tests we compare GPLHR with the implicitly restarted Arnoldi method available in ARPACK through the MATLAB function `eigs`. Here, we restrict our attention to the top three examples in Table 1 which are among the largest in our collection. The results for these problems are representative for large-scale eigenproblems.

In order to compute k eigenvalues closest to the target shift σ, ARPACK has to be used in a shift-invert mode that requires solving linear systems of the form $(A - \sigma B)x = b$ accurately.

Table 7(left) shows the difference in time required by GPLHR and ARPACK to compute a number of eigenvalues closest to σ. Here, the shift-and-invert in ARPACK is performed by means of a sparse LU factorization, whereas GPLHR only uses the incomplete ILU factors as a preconditioner. One can see that, even for problems of the size of several tens of thousands, GPLHR can achieve almost 10x speed up. Note that the results reported in Table 7(left) include time required to compute the full and incomplete triangular factors in ARPACK and GPLHR, respectively. All of our results were obtained on an iMac computer running Mac OS X 10.8.5, MATLAB R2013a, with a 3.4 GHz Intel Core i7 processor and 16GB 1600MHz DDR3 memory.

Another way to perform shift-and-invert in ARPACK is by using an iterative lin-

TABLE 7

Left: Time required by GPLHR with T given by ILU(10^{-3}) and ARPACK with the shift-and-invert operator computed through the LU decomposition of $A-\sigma I$. Right: Time required by GPLHR with T given by it_G steps of preconditioned GMRES and ARPACK with shift-and-invert operator defined through a preconditioned GMRES solve of $(A-\sigma I)w = r$ to full accuracy. GMRES is preconditioned with ILU(10^{-2}).

Problem	k	GPLHR	ARPACK	Problem	k	it_G	GPLHR	ARPACK
A15428	10	335	3006	A15428	10	7	153	196
AF23560	5	2	8	AF23560	5	5	6	131
CRY10000	5	1.4	1.9	CRY10000	5	5	6	95

ear solver to compute the solution of a system $(A-\sigma B)w = r$ to high accuracy. The GPLHR algorithm does not require such an accurate solution to such a type of linear systems. The performance of ARPACK and GPLHR in such a setting is compared in Table 7(right), which reports time required by both solvers to obtain the solution. Here, ARPACK's shift-and-invert is performed using GMRES, preconditioned by ILU(10^{-2}). The GPLHR preconditioner T is given by only a few (it_G) GMRES steps with the same ILU(10^{-2}) preconditioner.

6. Conclusions. We have presented the generalized preconditioned locally harmonic residual (GPLHR) method for computing eigenpairs of non-Hermitian regular matrix pairs (A, B) that correspond to the eigenvalues closest to a given shift σ. GPLHR is a block eigensolver, which takes advantage of a preconditioner when it is available and does not require an exact or highly accurate shift-and-invert procedure. The method allows multiple matrix-vector products to be computed simultaneously and can take advantage of BLAS3 through blocking. As a result, it is suitable for high performance computers with many processing units.

Our numerical experiments demonstrated the robustness and reliability of the proposed algorithm. We compared GPLHR to several state-of-the-art eigensolvers (including BGD, JDQR/JDQZ, and implicitly restarted Arnoldi) for a variety of eigenvalue problems coming from different applications. Our results show that GPLHR is competitive to the established approaches in general. It is often more efficient, especially if there is a limited amount of memory.

Appendix A. The GPLHR-EIG algorithm. In Algorithm 3, we present an eigenvector-based version of the GPLHR algorithm, called GPLHR-EIG.

The work flow of Algorithm 3 is nearly identical to that of the Schur vector–based GPLHR in Algorithm 2. The main algorithmic difference is that GPLHR-EIG discards the harmonic Ritz values computed in step 23 and, instead, defines eigenvalue approximations (α_j, β_j) using bilinear forms $\alpha_j = x_j^* A x_j$ and $\beta_j = x_j^* B x_j$, where x_j are the columns of X at a given iteration ($j = 1, \ldots, k$). Clearly, with this choice, the ratios α_j/β_j are exactly the Rayleigh quotients for (A, B) evaluated at the corresponding harmonic Ritz vectors. This approach is motivated by the fact that the Rayleigh quotients represent optimal eigenvalue approximations and is common in the harmonic Rayleigh–Ritz-based eigenvalue computations; see, e.g., [26, 29, 46].

The complexity and memory requirements of Algorithm 3 are comparable to those of Algorithm 2. Note that it is not necessary to keep both the approximate eigenvectors X and the orthonormal basis V, since X can be rewritten with V. Therefore, no additional memory for storing V is needed in practice.

Although in the numerical experiments of section 5 we report only the results for

Algorithm 3 The GPLHR-EIG algorithm for partial eigendecomposition (3)

Input: A pair (A, B) of n-by-n matrices, shift $\sigma \in \mathbb{C}$ different from any eigenvalue of (A, B), preconditioner T, starting guess of eigenvectors $X^{(0)} \in \mathbb{C}^{n\times k}$, and the subspace expansion parameter m.

Output: If $B \neq I$, then approximate eigenvectors $X \in \mathbb{C}^{n\times k}$ and the associated diagonal matrices $\Lambda_A, \Lambda_B \in \mathbb{C}^{k\times k}$ in (3) such that $\lambda_j = \Lambda_A(j,j)/\Lambda_B(j,j)$ are the k eigenvalues of (A, B) closest to σ.
If $B = I$, then approximate eigenvectors $X \in \mathbb{C}^{n\times k}$ and the associated diagonal matrix $\Lambda \in \mathbb{C}^{k\times k}$ of k eigenvalues of A closest to σ.

1: $X \leftarrow X^{(0)}$; $Q \leftarrow (A - \sigma B)X$; $P \leftarrow [\,]$;
2: Normalize columns of X ($x_j \leftarrow x_j/\|x_j\|$); $\Lambda_A \leftarrow \operatorname{diag}\{X^*AX\}$; $\Lambda_B \leftarrow \operatorname{diag}\{X^*BX\}$;
3: **while** convergence not reached **do**
4: $V \leftarrow \operatorname{orth}(X)$; $Q \leftarrow \operatorname{orth}(Q)$;
5: **if** $B \neq I$ **then**
6: $W \leftarrow (I - VV^*)T(I - QQ^*)(AX\Lambda_B - BX\Lambda_A)$;
7: **else**
8: $W \leftarrow (I - VV^*)T(I - VV^*)(AX\Lambda_B - X\Lambda_A)$;
9: **end if**
10: $W \leftarrow \operatorname{orth}(W)$; $S_0 \leftarrow W$; $S \leftarrow [\,]$;
11: **for** $l = 1 \to m$ **do**
12: **if** $B \neq I$ **then**
13: $S_l \leftarrow (I - VV^*)T(I - QQ^*)(AS_{l-1}\Lambda_B - BS_{l-1}\Lambda_A)$;
14: **else**
15: $S_l \leftarrow (I - VV^*)T(I - VV^*)(AS_{l-1}\Lambda_B - S_{l-1}\Lambda_A)$;
16: **end if**
17: $S_l \leftarrow S_l - W(W^*S_l)$; $S_l \leftarrow S_l - S(S^*S_l)$; $S_l \leftarrow \operatorname{orth}(S_l)$; $S \leftarrow [S\ S_l]$;
18: **end for**
19: $P \leftarrow P - V(V^*P)$; $P \leftarrow P - W(W^*P)$; $P \leftarrow P - S(S^*P)$; $P \leftarrow \operatorname{orth}(P)$;
20: Set the trial subspace $Z \leftarrow [V,\ W,\ S_1, \ldots,\ S_m,\ P]$;
21: $\hat{Q} \leftarrow \texttt{orth}((A - \sigma B)[W,\ S_1,\ \ldots,\ S_m, P])$; $\hat{Q} \leftarrow \hat{Q} - Q(Q^*\hat{Q})$;
22: Set the test subspace $U \leftarrow [Q, \hat{Q}]$;
23: Solve the projected eigenproblem (U^*AZ, U^*BZ); set $\bar{Y}$ to be the matrix of all eigenvectors ordered according to their eigenvalues' closeness to σ, ascendingly;
24: $Y \leftarrow \bar{Y}(\texttt{:,1:k})$;
25: $P \leftarrow WY_W + S_1Y_{S_1} + \cdots + S_mY_{S_m} + PY_P$, where $Y \equiv [Y_V^T, Y_W^T, Y_{S_1}^T, \ldots, Y_{S_m}^T, Y_P^T]^T$ is a conforming partition of Y;
26: $X \leftarrow VY_V + P$; $Q \leftarrow (A - \sigma B)X$;
27: $P \leftarrow Z\bar{Y}(\texttt{:,k+1:2k})$;[a]
28: Normalize columns of X ; $\Lambda_A \leftarrow \operatorname{diag}\{X^*AX\}$; $\Lambda_B \leftarrow \operatorname{diag}\{X^*BX\}$;
29: **end while**
30: If $B = I$, then $\Lambda \leftarrow \Lambda_A$.

[a] This step can be disabled if the LOBPCG-style search direction of step 25 is preferred.

the Schur vector–based variant of GPLHR (Algorithm 2), the performance of GPLHR-EIG was found to be similar for the problems under consideration. Therefore, we do not report results for Algorithm 3. Instead we refer the reader to [50], where a version of GPLHR-EIG has been benchmarked for a specific application.

Finally, let us remark that GPLHR-EIG is not quite suitable for computing larger subsets of eigenpairs using the deflation technique, unless information about left eigenvectors is available. In contrast to the Schur vector–based GPLHR, the computed set

of right eigenvectors cannot be directly used for deflation because of their nonorthogonality. Thus, Algorithm 3 should be invoked in situations where the number of desired eigenpairs k is sufficiently small to ensure their efficient computation in a single run of the algorithm with the block size (at least) k.

REFERENCES

[1] H. M. Aktulga, A. Buluç, S. Williams, and C. Yang, *Optimizing sparse matrix-multiple vectors multiplication for nuclear configuration interaction calculations*, in Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2014), 2014.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[3] E. Balslev and J. M. Combes, *Spectral properties of many-body Schrödinger operators with dilatation-analytic interactions*, Comm. Math. Phys., 22 (1971), pp. 280–294.

[4] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[5] K. A. Cliffe, A. Spence, and S. J. Tavener, *The numerical analysis of bifurcation problems with application to fluid mechanics*, Acta Numer., 9 (2000), pp. 39–131.

[6] H. C. Elman, A. Ramage, and D. J. Silvester, *IFISS: A Matlab toolbox for modelling incompressible flow*, ACM Trans. Math. Software, 33 (2007), 14.

[7] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers*, 2nd ed., Oxford University Press, New York, 2014.

[8] D. R. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.

[9] M. A. Freitag and A. Spence, *Shift-invert Arnoldi's method with preconditioned iterative solves*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 942–969.

[10] G. H. Golub and Q. Ye, *An inverse free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems*, SIAM J. Sci. Comput., 24 (2002), pp. 312–334.

[11] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[12] M. H. Gutknecht, *Block Krylov space methods for linear systems with multiple right-hand sides: An introduction*, in Modern Mathematical Models, Methods and Algorithms for Real World Systems, I. S. Duff, A. H. Siddiqi, and O. Christensen, eds., Anamaya, New Delhi, 2007, pp. 420–447.

[13] M. Head-Gordon and T. J. Lee, *Single reference coupled cluster and perturbation theories of electronic excitation energies*, in Recent Advances in Coupled Cluster Methods, R. J. Bartlett, ed., World Scientific, Singapore, 1997, pp. 221–253.

[14] T. Helgaker, P. Jorgensen, and J. Olsen, *Molecular Electronic-Structure Theory*, Wiley, Chichester, 2000.

[15] Y. K. Ho, *The method of complex coordinate rotation and its applications to atomic collision processes*, Phys. Rep., 99 (1983), pp. 1–68.

[16] M. E. Hochstenbach and G. L. G. Sleijpen, *Harmonic and refined Rayleigh–Ritz for the polynomial eigenvalue problem*, Numer. Linear Algebra Appl., 15 (2008), pp. 35–54.

[17] I. Kamwa, D. Lefebvre, and L. Loud, *Small signal analysis of hydro-turbine governors in large interconnected power plants*, IEEE Power Engineering Society Winter Meeting, Vol. 2, 2002, pp. 1178–1183.

[18] W. Kerner, *Computing the complex eigenvalue spectrum for resistive magnetohydrodynamics*, in Large Scale Eigenvalue Problems, J. Cullum and R. A. Willoughby, eds., Elsevier, Amsterdam, 1986, pp. 241–265.

[19] A. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comput., 23 (2001), pp. 517–541.

[20] A. V. Knyazev, M. E. Argentati, I. Lashuk, and E. E. Ovtchinnikov, *Block locally optimal preconditioned eigenvalue xolvers (BLOPEX) in hypre and PETSc*, SIAM J. Sci. Comput., 29 (2007), pp. 2224–2239.

[21] J. S. Langer, *Instabilities and pattern formation in crystal growth*, Rev. Modern Phys., 52 (1980), pp. 1–28.

[22] J. S. Langer and H. Muller-Krumbhaar, *Theory of dendritic growth-i. Elements of a stability analysis*, Acta Metallurgica, 26 (1978), pp. 1681–1687.

[23] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Software Environ.

Tools 6, SIAM, Philadelphia, 1998.
[24] C. B. Moler and G. W. Stewart, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Numer. Anal., 10 (1973), pp. 241–256.
[25] R. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1230.
[26] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl., 154/156 (1991), pp. 289–309.
[27] R. B. Morgan, *Generalizations of Davidson's method for computing eigenvalues of large nonsymmetric matrices*, J. Comput. Phys., 101 (1992), pp. 287–291.
[28] R. B. Morgan and D. S. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 817–825.
[29] R. B. Morgan and M. Zeng, *Harmonic projection methods for large non-symmetric eigenvalue problems*, Numer. Linear Algebra Appl., 5 (1998), pp. 33–55.
[30] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Classics Appl. Math. 20, SIAM, Philadelphia, 1998.
[31] B. Philippe and Y. Saad, *On correction equations and domain decomposition for computing invariant subspaces*, Comput. Methods Appl. Mech. Engrg., 196 (2007), pp. 1471–1483.
[32] G. Quillen and Q. Ye, *A block inverse-free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems*, J. Comput. Appl. Math., 233 (2010), pp. 1298–1313.
[33] W. P. Reinhardt, *Complex coordinates in the theory of atomic and molecular structure and dynamics*, Annu. Rev. Phys. Chem., 33 (1982), pp. 223–255.
[34] M. Robbé, M. Sadkane, and A. Spence, *Inexact inverse subspace iteration with preconditioning applied to non-Hermitian eigenvalue problems*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 92–113.
[35] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelpha, 2003.
[36] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, revised ed., Classics Appl. Math. 66, SIAM, Philadelphia, 2011.
[37] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[38] M. Sadkane, *Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems*, Numer. Math., 64 (1993), pp. 195–211.
[39] V. Simoncini, *Computational methods for linear matrix equations*, SIAM Rev., to appear.
[40] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.
[41] P. Sonneveld and M. B. van Gijzen, *IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations*, SIAM J. Sci. Comput., 31 (2008), pp. 1035–1062.
[42] A. Stathopoulos, Y. Saad, and K. Wu, *Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods*, SIAM J. Sci. Comput., 19 (1998), pp. 227–245.
[43] G. W. Stewart, *On the sensitivity of the eigenvalue problem $Ax = \lambda Bx$*, SIAM J. Numer. Anal., 9 (1972), pp. 669–686.
[44] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*, Academic Press, Boston, MA, 1990.
[45] L. N. Trefethen and M. Embree, *Spectra and Pseudospectra*, Princeton University Press, Princeton, NJ, 2005.
[46] E. Vecharynski and A. Knyazev, *Preconditioned locally harmonic residual method for computing interior eigenpairs of certain classes of Hermitian matrices*, SIAM J. Sci. Comput., 37 (2015), pp. S3–S29.
[47] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.
[48] F. Xue and H. C. Elman, *Fast inexact subspace iteration for generalized eigenvalue problems with spectral transformation*, Linear Algebra Appl., 435 (2011), pp. 601–622.
[49] F. Xue and H. C. Elman, *Fast inexact implicitly restarted Arnoldi method for generalized eigenvalue problems with spectral transformation*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 433–459.
[50] D. Zuev, E. Vecharynski, C. Yang, N. Orms, and A. I. Krylov, *New algorithms for iterative matrix-free eigensolvers in quantum chemistry*, J. Comput. Chem., 36 (2015), pp. 273–284.