

Simultaneous iterative reduction of large sparse matrix pairs with shift-and-invert Lanczos and Arnoldi methods

Roger B. Sidje

0000-0001-5353-3642

The University of Alabama,

Department of Mathematics,

Tuscaloosa, AL 35487, USA

Email: roger.b.sidje@ua.edu

Abstract—We describe iterative methods for simultaneously reducing large sparse matrix pairs using congruence transformations. For a symmetric pair, the reduction is tridiagonal-tridiagonal, while it is Hessenberg-Hessenberg for a nonsymmetric pair. We show how these reduction schemes relate to the well-established shift-and-invert Lanczos and Arnoldi methods when initialized with suitably chosen starting vectors, thus revealing capabilities that had so far remained obscured and untapped. In the setting of solving the generalized eigenvalue problem, we show furthermore that the tridiagonal-tridiagonal form simplifies to become tridiagonal-diagonal at no extra cost, and the Hessenberg-Hessenberg form to Hessenberg-triangular at no extra cost too. Illustrative numerical results are given.

I. INTRODUCTION

CONSIDER the problem of simultaneously reducing a matrix pair (A, B) with a congruence transformation. We assume the matrix pair (or matrix pencil) regular, i.e., $\det(A - \lambda B)$ is not identically zero. For full symmetric matrices, we can cite the direct tridiagonal-tridiagonal reduction methods of Garvey et al. [1] and Sidje [2], or the tridiagonal-diagonal reduction of Tisseur [3]. Such direct methods are however impractical for large sparse matrices since they add fill-in through successive updates of the matrices. Looking at earlier works of Nour-Omid et al. [4], [5], we see how they built on Ericsson and Ruhe [6] who first introduced the shift-and-invert Lanczos algorithm for solving the symmetric generalized eigenvalue problem $Ax = \lambda Bx$. However, those past works did not set out to purposely reduce the two large sparse matrices separately as we do here. More recently, in the nonsymmetric context, we can cite the Hessenberg-triangular reduction of Sorensen [7] who used a generalized Arnoldi procedure relying on two unitary Krylov bases, and there is also Li and Ye [8] who were able to produce upper and lower Hessenberg forms.

Our present study applies to large sparse pairs that are either symmetric or nonsymmetric. We will make the realization that the methods described in [1], [2] are direct methods that replicate the well-established shift-and-invert (SaI) Lanczos and Arnoldi algorithms under suitably chosen starting vectors. This newfound equivalence reveals the capability of the SaI

Lanczos to simultaneously tridiagonalize a symmetric pair in an iterative manner, while the SaI Arnoldi achieves a simultaneous Hessenberg reduction of a general pair. To our knowledge, these findings are new insights that have remarkably remained unheralded and untapped, making the Lanczos and Arnoldi algorithms, with their SaI extensions, the gifts that keep giving. The reader will soon see how we arrived at this realization by a circuitous route, which may explain why it remained obscured for so long after these algorithms were introduced, even as many other reduction methods, such as those cited earlier, were proposed in the meantime.

Before delving into the details, it is worth noting upfront that the SaI operation will amount to evaluating $(A - \sigma B)^{-1}v$ with a given v , which in our sparse context can itself be handled using classical iterative solvers based on Lanczos or Arnoldi recurrences, or other techniques meant for sparse linear systems. Also, the SaI operation here is typically acting on different vectors with the same single shift σ that is fixed at the very beginning, i.e., a constant sparse matrix $A - \sigma B$ is the coefficient matrix of a series of linear systems with different right-hand sides. This is a special situation that has benefited from extensive research. For example, a preconditioner can be reused; iterative methods can start with convenient initial guesses that are improved based on earlier solutions; and so forth, see, e.g., [9], [10], [11]. Moreover, a direct sparse solver may even become the method of choice when appropriate, since multiple linear systems have to be solved with the same sparse matrix, allowing a factorization to be computed and reused.

A natural application where a simultaneous reduction may be useful is the large sparse generalized eigenvalue problem $Ax = \lambda Bx$. Many methods have already been proposed for this problem in its own right, usually by implicit conversion to a standard eigenvalue problem. And to cope with size, a variety of projection schemes have sought to gradually expand a search subspace. In the symmetric case, there is Nour-Omid et al. [4] that we cited before, but also Parlett and Chen [12] where the Lanczos algorithm is reformulated with a semi (or pseudo) inner product. Unlike our present approach

that explicitly seeks from the outset to obtain a tridiagonal-tridiagonal reduction of the pair (A, B) , both [4], [12] get a tridiagonal-diagonal reduction with respect to a transformed problem. To avoid SaI operations, Golub and Ye [13] forgo the simultaneous reduction of the matrix pair and instead aim for a tridiagonal-general reduction. We can also cite the much different Jacobi-Davidson method that uses a preconditioner [14], with its equivalence to the inexact Rayleigh quotient iteration discussed in [15], [16]. The generalized eigenvalue problem often originates from the quadratic eigenvalue problem, deepening their intersection in the literature. Extensive references may be found in the survey of Tisseur and Meerbergen [17].

It should also be noted that many theoretical results exist on decompositions [18]. Just to give an example in our context, take a symmetric pair (A, B) where one matrix is positive definite, say B . There is a diagonal-diagonal decomposition as follows: form $P_1^T B P_1 = I$, where I is the identity matrix of compatible size and P_1 is invertible; then form $P_2^T (P_1^T A P_1) P_2 = \Lambda$, where Λ is the diagonal matrix of eigenvalues and P_2 is orthonormal. Hence, taking $P = P_1 P_2$ gives $(P^T A P, P^T B P) = (\Lambda, I)$. While this is neat, it ignores the difficulty of extracting the eigendecompositions in the first place. By contrast, our study is about computational building blocks to support practical algorithms or applications. In fact, our iterative simultaneous reduction is enabling, in that, it allows exploring different areas where a partial reduction may be useful. Examples include linear systems of the form $(A - \omega B)x = b$ over many values of ω . Another application could be a constrained optimization problem that consists in minimizing $x^T A x$ subject to (varying) constraints on $x^T B x$. Consider also a linear system of ordinary differential equations in the form $Ax + B\dot{x} = f$ as seen in moment matching or model reduction such as, e.g., [19], [20], [21], where using a matrix Q to simultaneously transform $(Q^T A Q, Q^T B Q) = (H, G)$ could make the reduced (H, G) easier to deal with than (A, B) . Importantly too, the end problem becomes smaller upon using a rectangular matrix Q_m (e.g., a truncation of Q as typically done in projection methods). The efficiency is thus aided by the possibility to reuse and exploit a reduced (and truncated) structure of (H, G) .

The remainder of the presentation is organized as follows. Section II is dedicated to the symmetric case. Section III is a generalization to the nonsymmetric case. Section IV establishes some theoretical results, notably the equivalence between methods. Section V shows how the reduction can impact applications such as the generalized eigenvalue problem. Section VI reports numerical experiments to illustrate certain practical aspects. Section VII finally gives concluding remarks.

II. REDUCTION TO TRIDIAGONAL-TRIDIAGONAL FORM

With I being the identity matrix of appropriate size depending on its context, denote e_k its k th column. The following statement sums up the outcomes of [1], [2] in a helpful way to frame the rest of our narrative.

Theorem 2.1: Given two n -by- n real symmetric matrices A, B , and a nonzero real parameter σ such that $C := A - \sigma B$

is nonsingular, assuming no breakdown in [1], [2], we can obtain a congruence transformation

$$(A, B) \mapsto (Q^T A Q, Q^T B Q)$$

that simultaneously reduces the pair to

$$Q^T A Q = T, \quad Q^T B Q = S, \quad Q, T, S \in \mathbb{R}^{n \times n}, \quad (1)$$

where Q is nonsingular and the resulting T and S are tridiagonal, and furthermore

$$Q^T C Q = T - \sigma S = \Delta, \quad \Delta \in \mathbb{R}^{n \times n}, \quad (2)$$

where Δ is diagonal.

The works in [1], [2] focused on how to directly construct Q using elementary transformations. We hinted earlier at a bridge linking such direct algorithms to an iterative SaI Lanczos method. This connection is veiled until revealed, and it is kept further shrouded by the fact that the SaI Lanczos had so far not been touted to our knowledge as able to simultaneously tridiagonalize a pair of symmetric matrices out of the box. Our present contribution will make the connection clear, obtaining as end result an iterative algorithm that performs the task. But both the direct and iterative algorithms can break down, explaining the caveat in Theorem 2.1. Our exposition will underline the details.

A. Direct method

We quickly summarize how the direct reduction is done in [2] to set the scene and allow the reader to clearly pinpoint where a breakdown may occur. Assuming the pair (A, B) regular, i.e., $A - \sigma B$ nonsingular for some scalar σ , consider the partitioning

$$A = \begin{pmatrix} a_{11} & u^T \\ u & \tilde{A} \end{pmatrix}, \quad B = \begin{pmatrix} b_{11} & v^T \\ v & \tilde{B} \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 \\ \tilde{x} & \tilde{Q} \end{pmatrix}, \quad (3)$$

where $\tilde{A}$ and $\tilde{B}$ are symmetric too. For economy of notation, Q here is the transformation for just one step. A justification of the special form imposed to Q in (3) is detailed in [2]. Focusing on finding the unknowns $\tilde{x}$ and $\tilde{Q}$ in (3), write

$$Q^T A Q = \begin{pmatrix} 1 & 0 \\ \tilde{x} & \tilde{Q} \end{pmatrix}^T \begin{pmatrix} a_{11} & u^T \\ u & \tilde{A} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \tilde{x} & \tilde{Q} \end{pmatrix} \quad (4a)$$

$$= \begin{pmatrix} a_{11} + 2\tilde{x}^T u + \tilde{x}^T \tilde{A} \tilde{x} & (u + \tilde{A} \tilde{x})^T \tilde{Q} \\ \tilde{Q}^T (u + \tilde{A} \tilde{x}) & \tilde{A} \end{pmatrix}, \quad (4b)$$

and

$$Q^T B Q = \begin{pmatrix} 1 & 0 \\ \tilde{x} & \tilde{Q} \end{pmatrix}^T \begin{pmatrix} b_{11} & v^T \\ v & \tilde{B} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \tilde{x} & \tilde{Q} \end{pmatrix} \quad (5a)$$

$$= \begin{pmatrix} b_{11} + 2\tilde{x}^T v + \tilde{x}^T \tilde{B} \tilde{x} & (v + \tilde{B} \tilde{x})^T \tilde{Q} \\ \tilde{Q}^T (v + \tilde{B} \tilde{x}) & \tilde{B} \end{pmatrix}. \quad (5b)$$

Hence, if in (4) and (5), we construct $\tilde{x}$ and $\tilde{Q}$ such that

$$\tilde{Q}^T (u + \tilde{A} \tilde{x}) = \beta e_1, \quad \tilde{Q}^T (v + \tilde{B} \tilde{x}) = \nu e_1, \quad (6)$$

we will achieve the first simultaneous reduction step with some $\alpha, \beta, \mu, \nu \in \mathbb{R}$,

$$Q^T A Q = \left(\begin{array}{c|ccc} \alpha & \beta & 0 & \cdots & 0 \\ \beta & & & & \\ 0 & & \tilde{Q}^T \tilde{A} \tilde{Q} & & \\ \vdots & & & & \\ 0 & & & & \end{array} \right), \quad (7a)$$

$$Q^T B Q = \left(\begin{array}{c|ccc} \mu & \nu & 0 & \cdots & 0 \\ \nu & & & & \\ 0 & & \tilde{Q}^T \tilde{B} \tilde{Q} & & \\ \vdots & & & & \\ 0 & & & & \end{array} \right). \quad (7b)$$

The implication of (6) is that $\tilde{Q}^T(u + \tilde{A}\tilde{x})$ and $\tilde{Q}^T(v + \tilde{B}\tilde{x})$ must be linearly dependent, meaning

$$\tilde{Q}^T(u + \tilde{A}\tilde{x}) = \sigma \tilde{Q}^T(v + \tilde{B}\tilde{x}), \quad 0 \neq \sigma \in \mathbb{R}, \quad (8)$$

and having $\tilde{Q}$ nonsingular, (8) would give

$$u + \tilde{A}\tilde{x} = \sigma(v + \tilde{B}\tilde{x}).$$

Note the subtlety that (8) allows σ to be any nonzero scalar. With σ chosen so that $\tilde{A} - \sigma\tilde{B}$ is nonsingular, we are led to $(\tilde{A} - \sigma\tilde{B})\tilde{x} = -(u - \sigma v)$, which gives

$$\tilde{x} = -(\tilde{A} - \sigma\tilde{B})^{-1}(u - \sigma v). \quad (9)$$

Equation (9) fully characterizes how to uniquely obtain $\tilde{x}$, following which we have some freedom in the choice of $\tilde{Q}$ in (6), such as picking the Householder reflector targeted at the vector $v + \tilde{B}\tilde{x}$, or other choices discussed in [2] and not repeated here. The continuing reduction steps amount to transferring the same process to the pair $(\tilde{Q}^T \tilde{A} \tilde{Q}, \tilde{Q}^T \tilde{B} \tilde{Q})$ and so on. Another note is that σ is changeable at each step, although doing so makes the cost of the method prohibitive, and muddles the SaI Lanczos connection that we wish to show later. Assume no breakdown across $n - 1$ steps and revert to denoting Q as the final product of the transformations. A variable σ would make $Q^T(A - B)Q$ simply tridiagonal, whereas a fixed σ additionally leads to $Q^T(A - \sigma B)Q = T - \sigma S = \text{diag.}$, as stated in (2).

We now turn our focus to identifying where and how a breakdown may occur, something not fully detailed before. Observe that if $\tilde{A} - \sigma\tilde{B}$ is singular in (9), we cannot reference its inverse to get $\tilde{x}$ anymore. To discuss this further, consider the explicit form of the inverse of $A - \sigma B$ based on the partitioning in (3),

$$(A - \sigma B)^{-1} = \begin{pmatrix} a_{11} - \sigma b_{11} & u^T - \sigma v^T \\ u - \sigma v & \tilde{A} - \sigma\tilde{B} \end{pmatrix}^{-1} = Z \quad (10a)$$

$$= \begin{pmatrix} \zeta & z^T \\ z & \tilde{Z} \end{pmatrix} = \begin{pmatrix} \frac{\det(\tilde{A} - \sigma\tilde{B})}{\det(A - \sigma B)} & z^T \\ z & \tilde{Z} \end{pmatrix}. \quad (10b)$$

The value of ζ in the last equality in (10) is readily seen by applying Cramer's rule for the $(1,1)$ -element of the inverse. The fact that the matrix pair is assumed regular ensures that

the inverse in (10) exists with a suitable choice of σ . This also explains why we cannot conjointly have $\zeta = 0$ and $z = 0$ there, as that would contradict the existence of the inverse $(A - \sigma B)^{-1} = Z$ in the first place. Using the equality $(A - \sigma B)(\zeta, z^T)^T = e_1$ and the partitioning in (10), it is clear that

$$\begin{cases} \zeta = e_1^T (A - \sigma B)^{-1} e_1 = \frac{\det(\tilde{A} - \sigma\tilde{B})}{\det(A - \sigma B)} \\ \zeta(u - \sigma v) + (\tilde{A} - \sigma\tilde{B})z = 0 \end{cases} \quad (11a)$$

Keeping in mind (9), we see that if $\zeta \neq 0$, then (11b) implies that $z = -\zeta(\tilde{A} - \sigma\tilde{B})^{-1}(u - \sigma v) = \zeta\tilde{x}$ and therefore

$$\tilde{x} = \frac{1}{e_1^T (A - \sigma B)^{-1} e_1} [(A - \sigma B)^{-1} e_1]_{2:n}. \quad (12)$$

We note in passing that the equality in (12) is the one used in practice to compute $\tilde{x}$ instead of (9). Indeed, (12) has the advantage of allowing us to keep a single LDL^T factorization of $A - \sigma B$ throughout the entire reduction. It was first suggested in [1] through a different derivation. Observe now this concern: even though $\det(A - \sigma B) \neq 0$, we may well have $\det(\tilde{A} - \sigma\tilde{B}) = 0$, causing $\zeta = 0$ in (11a) and thus invalidating the division in (12) under two possible scenarios:

$$\zeta = e_1^T (A - \sigma B)^{-1} e_1 = 0$$

with

$$\begin{cases} u - \sigma v = 0, \\ \text{or} \\ u - \sigma v \neq 0. \end{cases} \quad (13a)$$

$$(13b)$$

While case (13a) is benign and does not halt the reduction since we can take $\tilde{x} = 0$ in (9) and keep going, case (13b) is a serious breakdown, and dealing with it may involve to locally change σ , or globally start over with a modified pair, say $(M^T A M, M^T B M)$ using some matrix M , and hope that this serious breakdown will not reoccur. The lack of guarantee is why Theorem 2.1 has the caveat of assuming that no breakdown has occurred.

B. Iterative method

We now turn our attention to reformulating the reduction iteratively. As it will be revealed later, this lays the groundwork for the future claim that the direct method we just summarized above is an incarnation of a Lanczos method. Let

$$\begin{aligned} Q &= [q_1, \dots, q_n], \quad P = Q^{-T} = [p_1, \dots, p_n], \\ T &= \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \ddots & & \\ & \ddots & \ddots & \beta_n & \\ & & \beta_n & \alpha_n & \end{pmatrix}, \quad S = \begin{pmatrix} \mu_1 & \nu_2 & & & \\ \nu_2 & \mu_2 & \ddots & & \\ & \ddots & \ddots & \nu_n & \\ & & \nu_n & \mu_n & \end{pmatrix} \\ \Delta &= \begin{pmatrix} \delta_1 & & & & \\ & \ddots & & & \\ & & & & \delta_n \end{pmatrix}. \end{aligned} \quad (14)$$

Denote, respectively, $Q_k = [q_1, \dots, q_k]$, $P_k = [p_1, \dots, p_k]$, the matrices consisting of the first k columns of Q and P ; while T_k , S_k , and Δ_k are the k -by- k leading principal submatrices of T , S and Δ , respectively. Assuming (1) and (2) as working hypotheses, we write

$$AQ = PT, \quad Q = (A - \sigma B)^{-1} P \Delta, \quad (15)$$

and multiplying on the right by e_k to extract the k th column, we get the coupled relations

$$\begin{cases} q_k = \delta_k (A - \sigma B)^{-1} p_k \\ Aq_k = \beta_k p_{k-1} + \alpha_k p_k + \beta_{k+1} p_{k+1} \end{cases}, \quad k = 1, \dots, n,$$

where we are using, for convenience of notation, the convention that $\beta_1 p_0 \equiv 0 \equiv \beta_{n+1} p_{n+1}$ at the boundaries. Operations involved throughout our description are assumed to be defined, otherwise there is a breakdown. The biorthogonality condition $P^T Q = I$ implies that

$$\alpha_k = q_k^T A q_k, \quad \delta_k = \frac{1}{p_k^T (A - \sigma B)^{-1} p_k},$$

and we are left with some freedom on how to choose the scaling factor β_{k+1} . All this is reminiscent of the biorthogonal Lanczos process. Drawing from what is done in that process, we can choose β_{k+1} in various ways. Define

$$\begin{cases} \hat{p}_{k+1} \equiv \beta_{k+1} p_{k+1} = Aq_k - \alpha_k p_k - \beta_k p_{k-1} \\ \hat{q}_{k+1} \equiv (A - \sigma B)^{-1} \hat{p}_{k+1} = \beta_{k+1} (A - \sigma B)^{-1} p_{k+1} \end{cases} \quad (16a)$$

meaning

$$\begin{cases} p_{k+1} = \hat{p}_{k+1} / \beta_{k+1} \\ q_{k+1} = \hat{q}_{k+1} / \beta_{k+1} \end{cases}, \quad (16b)$$

and observe that, irrespective of the choice of β_{k+1} , we always have

$$\begin{aligned} \delta_{k+1} &= \frac{1}{p_{k+1}^T (A - \sigma B)^{-1} p_{k+1}} \\ &= \frac{1}{(\hat{p}_{k+1} / \beta_{k+1})^T (\hat{q}_{k+1} / \beta_{k+1})} \\ &= \frac{\beta_{k+1}^2}{\hat{p}_{k+1}^T \hat{q}_{k+1}}. \end{aligned}$$

Below are some possible choices at this junction:

- Choice of taking $|\delta_{k+1}| = 1$ so that both p_{k+1} and q_{k+1} in (16b) are scaled from $\hat{p}_{k+1}$ and $\hat{q}_{k+1}$ with the same factor β_{k+1} , i.e.,

$$|\beta_{k+1}| = |\hat{p}_{k+1}^T \hat{q}_{k+1}|^{1/2}, \quad (17a)$$

$$\delta_{k+1} = \beta_{k+1}^2 / \hat{p}_{k+1}^T \hat{q}_{k+1} = \text{sign}(\hat{p}_{k+1}^T \hat{q}_{k+1}). \quad (17b)$$

- Choice of requiring p_{k+1} to be of unit 2-norm, i.e., $1 \equiv \|p_{k+1}\|_2 = \|\hat{p}_{k+1} / \beta_{k+1}\|_2$, thus

$$|\beta_{k+1}| = \|\hat{p}_{k+1}\|_2, \quad \delta_{k+1} = \frac{\|\hat{p}_{k+1}\|_2^2}{\hat{p}_{k+1}^T \hat{q}_{k+1}}. \quad (17c)$$

- Choice of requiring q_{k+1} to be of unit 2-norm, i.e., $1 \equiv \|q_{k+1}\|_2 = \|\delta_{k+1} (A - \sigma B)^{-1} p_{k+1}\|_2 = \frac{\beta_{k+1}^2 \|\hat{q}_{k+1} / \beta_{k+1}\|_2}{|\hat{p}_{k+1}^T \hat{q}_{k+1}|} = \frac{|\beta_{k+1}| \|\hat{q}_{k+1}\|_2}{|\hat{p}_{k+1}^T \hat{q}_{k+1}|}$, leading to

$$|\beta_{k+1}| = \frac{|\hat{p}_{k+1}^T \hat{q}_{k+1}|}{\|\hat{q}_{k+1}\|_2}, \quad \delta_{k+1} = \frac{\hat{p}_{k+1}^T \hat{q}_{k+1}}{\|\hat{q}_{k+1}\|_2^2}. \quad (17d)$$

- Choice of requiring both p_{k+1} and q_{k+1} of unit 2-norm, which Parlett et al. [22, Sect. 8] found to have some merit but the tradeoff is that $P^T Q$ is diagonal without necessarily being I , and so the algorithm and discussion would have to be recast to not piggyback on the simplification that $P^T Q = I$.

Remark 2.1: The exact sign of β_{k+1} has no theoretical importance in the choices above. For the sake of remaining concise we adopt (17a), which further causes D to take the form of a signature matrix, $D = J \equiv \text{diag}(\pm 1)$. Algorithm 1 summarizes the overall steps in pseudo-code with this choice and the sign of $\beta_{k+1} > 0$.

ALGORITHM 1: Tridiagonal-tridiagonal reduction by 3-term loop

Version 1(a)

1. Set $p_0 := 0$;
2. Choose $\hat{p}_1 \neq 0$;
3. $\hat{q}_1 := (A - \sigma B)^{-1} \hat{p}_1$;
4. $\beta_1 := |\hat{p}_1^T \hat{q}_1|^{1/2}$;
5. $\delta_1 := \text{sign}(\hat{p}_1^T \hat{q}_1)$;
6. $p_1 := \hat{p}_1 / \beta_1$;
7. $q_1 := \hat{q}_1 (\delta_1 / \beta_1)$;
8. $\alpha_1 := q_1^T A q_1$;
9. **for** $k := 1 : m$ **do** % 1 : $n - 1$ for all
10. $\hat{p}_{k+1} := Aq_k - \alpha_k p_k - \beta_k p_{k-1}$;
11. $\hat{q}_{k+1} := (A - \sigma B)^{-1} \hat{p}_{k+1}$;
12. $\beta_{k+1} := |\hat{p}_{k+1}^T \hat{q}_{k+1}|^{1/2}$;
13. $\delta_{k+1} := \text{sign}(\hat{p}_{k+1}^T \hat{q}_{k+1})$;
14. $p_{k+1} := \hat{p}_{k+1} / \beta_{k+1}$;
15. $q_{k+1} := \hat{q}_{k+1} (\delta_{k+1} / \beta_{k+1})$;
16. $\alpha_{k+1} := q_{k+1}^T A q_{k+1}$;
17. **endfor**
18. Set T, Δ as (14); $S := (T - \Delta) / \sigma$;

Version 1(b)

1. Set $p_0 := 0$;
 2. Choose $\hat{p}_1 \neq 0$; *Cost:*
 3. **for** $k := 1 : m + 1$ **do** % 1 : n for all
 4. $\hat{q}_k := (A - \sigma B)^{-1} \hat{p}_k$; *Sal*
 5. $\beta_k := |\hat{p}_k^T \hat{q}_k|^{1/2}$; *2n*
 6. $\delta_k := \text{sign}(\hat{p}_k^T \hat{q}_k)$;
 7. $p_k := \hat{p}_k / \beta_k$; *n*
 8. $q_k := \hat{q}_k (\delta_k / \beta_k)$; *n*
 9. $\alpha_k := q_k^T A q_k$; *MV + 2n*
 10. **if** $k = n$ **break**;
 11. $\hat{p}_{k+1} := Aq_k - \alpha_k p_k - \beta_k p_{k-1}$; *4n*
 12. **endfor**
 13. Set T, Δ as (14); $S := (T - \Delta) / \sigma$;
-

Algorithm 1(a) is a reader-friendly mapping of the formulas that we have detailed, making it easier to relate their indexing to the iterations. But observe there that the code-section that initializes p_1 and q_1 at the beginning (lines 3–8) is similar to the code-section that computes p_{k+1} and q_{k+1} at the end (lines 11–16). After folding these two code-sections, we get a simpler and more compact version that would otherwise

be nontrivial to deduce from scratch without Algorithm 1(a) or re-indexing the derivation to be k -based. Algorithm 1(b) shows this compact version. Readers wishing to quickly test the method can see a MATLAB prototype called `sptritri` in the appendix (where online readers may select and copy-paste). Note that β_1 is computed but not kept in the layout of T in (14), and for improved stability, the three-term recurrence is in practice implemented by intermingling updates, as coefficients become available. Another remark is that calculations that would be wasted with known zero vectors, such as p_0 , are avoided in practice.

Marginal notes in Algorithm 1(b) detail the cost of its operations. It follows that m iterations cost about $m \cdot (\text{SaI} + \text{MV} + 10n)$ flops, neglecting terms that depend on m only, since it is expected that $m \ll n$ in practice. Here, SaI and MV denote, respectively, the cost of the action of a shift-and-invert operation and a matrix-vector product.

If we perform m iterations of Algorithm 1(a) with no breakdown, or equivalently $(m+1)$ iterations of Algorithm 1(b) for compatibility of writing, we would get a partial tridiagonalization satisfying the following Lanczos-like relations,

$$\begin{cases} Q_m^T P_m = I \\ AQ_m = P_m T_m + \beta_{m+1} p_{m+1} e_m^T \\ BQ_m = P_m S_m + \nu_{m+1} p_{m+1} e_m^T \\ Q_m^T AQ_m = T_m \\ Q_m^T BQ_m = S_m \\ T_m - \sigma S_m = \Delta_m \end{cases} \quad (18)$$

A breakdown occurs if there is a division by zero at some iteration. This would happen if $\beta_{k+1} = 0$ in line 14 of Algorithm 1(a), or $\beta_k = 0$ in line 7 of Algorithm 1(b). To not repeatedly reference the two versions in the following fine points, we only map indices to Algorithm 1(a). There is a *benign* (a.k.a. *happy*) breakdown situation if we conjointly have $\beta_{k+1} = 0$ and $\hat{p}_{k+1} = 0 = \hat{q}_{k+1}$, meaning $p_{k+1} = 0 = q_{k+1}$. So that, if we set $m \leftarrow k$ and thus $p_{m+1} \leftarrow 0$ in Eq. (18), we get $AQ_m = P_m T_m$ and $BQ_m = P_m S_m$. And using $Q_m = C^{-1} P_m \Delta_m$, we get $AC^{-1} P_m = P_m T_m \Delta_m^{-1}$, which means that P_m spans an invariant subspace of AC^{-1} . Also, $AQ_m = P_m T_m$ implies $C^{-1} AQ_m = C^{-1} P_m \Delta_m \Delta_m^{-1} T_m = Q_m \Delta_m^{-1} T_m$, which likewise means that Q_m spans an invariant subspace of $C^{-1} A$. This outcome is also present in the direct method of section II-A if $\beta = 0 = \nu$ in Eqs. (6-7). A benign breakdown is not a fatal outcome because the process can continue using a new set of appropriately chosen p_{k+1} and q_{k+1} , although in practice it signifies a happy ending in applications that leverage invariant subspaces. We do however get a *serious* breakdown if $\beta_{k+1} = 0$ but $\hat{p}_{k+1} \neq 0$ is an isotropic vector of $(A - \sigma B)^{-1}$ that causes $\hat{p}_{k+1}^T (A - \sigma B)^{-1} \hat{p}_{k+1} = 0 = \hat{p}_{k+1}^T \hat{q}_{k+1}$. Such a breakdown is impossible if $A - \sigma B$ is definite, meaning $v^T (A - \sigma B) v \neq 0$ for any vector $v \neq 0$. But, just as in the direct method, there is the risk that a breakdown can occur with certain input data.

III. REDUCTION TO HESSENBERG-HESSENBERG FORM

This section adapts algorithmic concepts to general matrices. The main result is first given below, showing how

a regular pair may be reduced to a Hessenberg-Hessenberg form that mathematically simplifies to a tridiagonal-tridiagonal form when the pair is symmetric as seen earlier. The next section IV will discuss theoretical aspects for pairs that are either symmetric and nonsymmetric.

Theorem 3.1: Given two general n -by- n real matrices A, B , and a nonzero real parameter σ such that $C := A - \sigma B$ is nonsingular, assuming no breakdown in the algorithm, we can obtain a congruence transformation

$$(A, B) \mapsto (Q^T A Q, Q^T B Q)$$

that simultaneously reduces the pair to

$$Q^T A Q = H, \quad Q^T B Q = G, \quad Q, H, G \in \mathbb{R}^{n \times n}, \quad (19)$$

where Q is nonsingular and the resulting H and G are upper Hessenberg, and furthermore

$$Q^T C Q = H - \sigma G = R, \quad R \in \mathbb{R}^{n \times n}, \quad (20)$$

where R is upper triangular.

A. Direct method

A direct computational procedure aimed at small nonsymmetric matrices can be derived in terms of elementary transformations as done for the symmetric case in section II-A. The main points of difference arise from the fact that not both upper and lower elements are eliminated when the pair (A, B) is not symmetric anymore. We will not dwell further on specifics pertaining to the direct method given that our focus is on large sparse matrices. We instead turn our attention to describing an iterative Arnoldi-like method that generalizes the Lanczos-like method given earlier.

B. Iterative method

We proceed by construction. Let $Q = [q_1, \dots, q_n]$, $Q_k = [q_1, \dots, q_k]$, $P = Q^{-T} = [p_1, \dots, p_n]$, $P_k = [p_1, \dots, p_k]$. From (19) and (20), we should have

$$AQ = PH, \quad Q = (A - \sigma B)^{-1} P R, \quad (21)$$

and

$$R = [P^T (A - \sigma B)^{-1} P]^{-1}. \quad (22)$$

Denote H_k , G_k , and R_k the k -by- k leading principal submatrices of H , G and R , respectively. With R (and R^{-1}) upper triangular, letting $P_{k+1} = [P_k, p_{k+1}]$, (22) implies that the last column of R_{k+1} is

$$R_{k+1} e_{k+1} = [P_{k+1}^T (A - \sigma B)^{-1} P_{k+1}]^{-1} e_{k+1} \quad (23a)$$

$$= \begin{bmatrix} R_k^{-1} & P_k^T (A - \sigma B)^{-1} p_{k+1} \\ 0 & p_{k+1}^T (A - \sigma B)^{-1} p_{k+1} \end{bmatrix}^{-1} e_{k+1} \quad (23b)$$

$$= r_{k+1, k+1} \begin{bmatrix} -R_k P_k^T (A - \sigma B)^{-1} p_{k+1} \\ 1 \end{bmatrix} \quad (23c)$$

where

$$r_{k+1, k+1} = \frac{1}{p_{k+1}^T (A - \sigma B)^{-1} p_{k+1}}.$$

Although we shall see later that it is not necessary to know R in advance, (23) means that R can be incrementally built columnwise as p_{k+1} and $(A - \sigma B)^{-1}p_{k+1}$ become available. Note that in exact arithmetic R is a diagonal matrix when the pair is symmetric, but this property is usually lost in practice due to floating-point arithmetic.

Now, to obtain the recurrence for generating the p_k , multiply (21) on the right by e_k to extract the k th column, and having $H = [h_{ik}]$ upper Hessenberg, we get

$$Aq_k = \sum_{i=1}^{k+1} h_{ik} p_i, \\ q_k = (A - \sigma B)^{-1} P_k R_k e_k = W_k R_k e_k,$$

with $W_k \equiv (A - \sigma B)^{-1} P_k$. This definition of W_k and the fact that R is upper triangular also imply that

$$Q_k = W_k R_k. \quad (24)$$

We provide additional observations to make the subsequent algorithmic details easier to read. By writing

$$Aq_k - \sum_{i=1}^k h_{ik} p_i = h_{k+1,k} p_{k+1},$$

it is seen that the next iterate p_{k+1} follows from knowing q_k and P_k . Moreover, $\forall i = 1, \dots, k$, we have $h_{ik} = q_i^T A q_k$ owing to the biorthogonality condition $P^T Q = I$. Understandably, for improved numerical stability, these coefficients are computed in practice with a scheme like MGS (Modified Gram-Schmidt) than CGS (Classical Gram-Schmidt), just as is done when implementing the Arnoldi process. Also note that there is some freedom in picking $h_{k+1,k}$ as was the case in picking β_{k+1} in section II, and we can emulate the choices described there. Indeed, if we set

$$Aq_k - \sum_{i=1}^k h_{ik} p_i = h_{k+1,k} p_{k+1} \equiv \hat{p}_{k+1}, \\ \hat{q}_{k+1} \equiv (A - \sigma B)^{-1} \hat{p}_{k+1},$$

then from (23) the $(k+1, k+1)$ -diagonal element of R is

$$r_{k+1,k+1} = \frac{1}{p_{k+1}^T (A - \sigma B)^{-1} p_{k+1}} \quad (25a)$$

$$= \frac{h_{k+1,k}^2}{\hat{p}_{k+1}^T \hat{q}_{k+1}} \quad (25b)$$

so that if we take in particular

$$h_{k+1,k} = |\hat{p}_{k+1}^T \hat{q}_{k+1}|^{1/2}, \quad (26)$$

we get the simplification

$$r_{k+1,k+1} = \text{sign}(\hat{p}_{k+1}^T \hat{q}_{k+1}) = \pm 1.$$

However, using (25) extends seamlessly to the complex case (when complex scalar products come into effect). The rest of our presentation retains (26). We could have taken $h_{k+1,k} = \|\hat{p}_{k+1}\|_2$ or $h_{k+1,k} = \|\hat{q}_{k+1}\|_2$ to emulate the other choices.

To recap, the next iteration in the nonsymmetric case involves the relations

$$\begin{cases} \hat{p}_{k+1} \equiv Aq_k - \sum_{i=1}^k h_{ik} p_i = h_{k+1,k} p_{k+1}, & h_{ik} = q_i^T A q_k \\ \hat{q}_{k+1} \equiv (A - \sigma B)^{-1} \hat{p}_{k+1} \\ h_{k+1,k} = |\hat{p}_{k+1}^T \hat{q}_{k+1}|^{1/2} \text{ for choice (26), or chose differently} \\ W_{k+1} \equiv (A - \sigma B)^{-1} P_{k+1} = [W_k, w_{k+1}], \\ w_{k+1} \equiv (A - \sigma B)^{-1} p_{k+1} = \hat{q}_{k+1} / h_{k+1,k} \\ r_{k+1,k+1} = \frac{h_{k+1,k}^2}{\hat{p}_{k+1}^T \hat{q}_{k+1}}, \\ R_{k+1} e_{k+1} = r_{k+1,k+1} \begin{bmatrix} -R_k P_k^T w_{k+1} \\ 1 \end{bmatrix} \\ q_{k+1} = W_{k+1} R_{k+1} e_{k+1} = r_{k+1,k+1} (I - W_k R_k P_k^T) w_{k+1} \\ = \frac{r_{k+1,k+1}}{h_{k+1,k}} (I - Q_k P_k^T) \hat{q}_{k+1} \end{cases}.$$

The last equality for q_{k+1} uses (24) and brings out the non-orthogonal projection involved. Also apparent now is that neither R nor the intermediate variable W need to be stored. Only the diagonal entries of R are explicitly used. Letting $S = R^{-1}$, it proves easier to build S (rather than R) columnwise by just storing

$$S_{k+1} e_{k+1} = \begin{bmatrix} P_k^T \hat{q}_{k+1} / h_{k+1,k} \\ 1/r_{k+1,k+1} \end{bmatrix},$$

and then recovering R by inverting S at the very end, with both R and $S = R^{-1}$ available to applications where useful. Worthy of further note is that the biorthogonality of P and Q means that the evaluation of $(I - Q_k P_k^T) \hat{q}_{k+1} / h_{k+1,k}$ needed for q_{k+1} can be implemented with a MGS-like scheme where the computed coefficients from the dot products handily give the entries of S as by-products.

These details are put together in the pseudo-code shown in Algorithm 2, where it should be stressed that the subscripts on the temporary variables $\hat{p}_{k+1}$ and $\hat{q}_{k+1}$ are kept for readability, but these variables are not stored as arrays. Yet another remark is that the diagonal elements of R are obviously available since they are the reciprocals of the diagonal elements of S , but their appearance in the pseudo-code is also for the sake of readability. In real arithmetic, the algorithm starts with two vectors p_1 and q_1 that satisfy $p_1^T q_1 = 1$ and $r_{11} = 1/p_1^T (A - \sigma B)^{-1} p_1 = \pm 1$. The appendix has a MATLAB prototype called `sphesshess` for the convenience of the reader wishing to try out the method right away. It is also closer to how the method could be implemented in other imperative programming languages.

Algorithm 2 can break down, as in the previous section, if a division by zero occurs. There are two cases: a *benign* (or *happy*) breakdown with $\hat{p}_{k+1} = 0$, ending with an invariant subspace. Or a *serious* breakdown if $\hat{p}_{k+1} \neq 0$ turns out to be an isotropic vector of $(A - \sigma B)^{-1}$, causing $\hat{p}_{k+1}^T \hat{q}_{k+1} = 0 = \hat{p}_{k+1}^T (A - \sigma B)^{-1} \hat{p}_{k+1}$. Here too, the algorithm is guaranteed to not have such a breakdown if $A - \sigma B$ is definite. In the

absence of a breakdown, setting the outermost for-loop in statement 7 to a range $m \leq n$ will interrupt the algorithm before reaching n steps and produce a partial Hessenberg-Hessenberg reduction with Arnoldi-like relations:

$$\begin{cases} P_m^T Q_m = I \\ A Q_m = P_m H_m + h_{m+1,m} p_{m+1} e_m^T \\ B Q_m = P_m G_m + g_{m+1,m} p_{m+1} e_m^T \\ Q_m^T A Q_m = H_m \\ Q_m^T B Q_m = G_m \\ H_m - \sigma G_m = R_m \end{cases} \quad (27)$$

The cost of Algorithm 2 is about $(m+1) \cdot \text{SaI} + m \cdot \text{MV} + 4n(m+1)^2$ operations by adding the fixed cost of initialization to the cost of m iterations, again neglecting terms that depend on m only, when assuming $m \ll n$.

ALGORITHM 2: Hessenberg-Hessenberg reduction

```

1. Choose  $\hat{p}_1 \neq 0$ ;
2.  $\hat{q}_1 := (A - \sigma B)^{-1} \hat{p}_1$ ;
3.  $r_{11} := |\hat{p}_1^T \hat{q}_1| / |\hat{p}_1^T \hat{q}_1|$ ; % sign( $\hat{p}_1^T \hat{q}_1$ ) per choice (26)
4.  $s_{11} := r_{11}^{-1}$ ;
5.  $p_1 := \hat{p}_1 / |\hat{p}_1^T \hat{q}_1|^{1/2}$ ;
6.  $q_1 := r_{11} \hat{q}_1 / |\hat{p}_1^T \hat{q}_1|^{1/2}$ ;
7. for  $k := 1 : m$  do %  $1 : n$  for all
8.    $w := A q_k$ ;
9.   for  $i := 1 : k$  do
10.     $h_{ik} := q_i^T w$ ;
11.     $w := w - h_{ik} p_i$ ;
12.   endfor
13.   if  $k = n$  break;
14.    $\hat{p}_{k+1} := w$ ;
15.    $\hat{q}_{k+1} := (A - \sigma B)^{-1} \hat{p}_{k+1}$ ;
16.    $h_{k+1,k} := |\hat{p}_{k+1}^T \hat{q}_{k+1}|^{1/2}$ ;
17.    $r_{k+1,k+1} := h_{k+1,k}^2 / |\hat{p}_{k+1}^T \hat{q}_{k+1}|$ ; % sign( $\hat{p}_{k+1}^T \hat{q}_{k+1}$ ) if real
18.    $s_{k+1,k+1} := r_{k+1,k+1}^{-1}$ ;
19.    $p_{k+1} := \hat{p}_{k+1} / h_{k+1,k}$ ;
20.   % evaluate  $q_{k+1} := r_{k+1,k+1} (I - Q_k P_k^T) \hat{q}_{k+1} / h_{k+1,k}$ 
21.    $w := \hat{q}_{k+1}$ ;
22.   for  $i := 1 : k$  do
23.     $t := p_i^T w$ ;
24.     $w := w - t \cdot q_i$ ;
25.     $s_{i,k+1} := t / h_{k+1,k}$ ;
26.   endfor
27.    $q_{k+1} := (r_{k+1,k+1} / h_{k+1,k}) \cdot w$ ;
28. endfor
29.  $R := S^{-1}$ ;  $G := (H - R) / \sigma$ ;
```

IV. CHARACTERIZATION OF THE EXISTENCE OF THE SIMULTANEOUS REDUCTION

As we have stressed before, the algorithms outlined so far can break down, and so they did not constructively prove that the entire simultaneous reduction can be performed in one stretch. This section looks at the conditions under which we can assume the complete reduction of a pair $A, B \in \mathbb{R}^{n \times n}$, given a nonzero real parameter σ such that $C := A - \sigma B$ is nonsingular. The pair is symmetric or nonsymmetric depending on the context.

A. Equivalence of the simultaneous reduction with a decomposition of a moment matrix

We begin by placing our algorithms within the broader family of Krylov subspace projection methods and by specifying the spanning sets involved. As done before, we will be referencing Algorithm 1(a) for uniformity of indexing.

Lemma 4.1: *If there are $m \leq n$ successful steps of either Algorithm 1(a) for a symmetric pair, or Algorithm 2 for a nonsymmetric pair, then the resulting P_m is a basis of the Krylov subspace*

$$\mathcal{K}_m(p_1, A(A - \sigma B)^{-1}) = \mathcal{K}_m(p_1, B(A - \sigma B)^{-1}), \quad (28a)$$

while q_1 is by construction a scalar multiple of $(A - \sigma B)^{-1} p_1$, and Q_m is a basis of

$$\mathcal{K}_m(q_1, (A - \sigma B)^{-1} A) = \mathcal{K}_m(q_1, (A - \sigma B)^{-1} B). \quad (28b)$$

Proof. Proceed by induction on m , starting first by showing that P_m is a basis of $\mathcal{K}_m(p_1, A(A - \sigma B)^{-1}) \equiv \text{span}\{p_1, AC^{-1}p_1, \dots, (AC^{-1})^{m-1}p_1\}$. With the claim obvious at $m = 1$, assume it true up to m . Then, it follows from either (18) or (27) that $p_{m+1} \in Aq_m + \text{span}\{p_1, \dots, p_m\}$, and with the induction assumption that $q_m \in \mathcal{K}_m(q_1, C^{-1}A) = \mathcal{K}_m(C^{-1}p_1, C^{-1}A)$, there are two polynomials φ_{m-1} and ψ_{m-1} of degree $m - 1$ such that

$$\begin{aligned} p_{m+1} &= A\varphi_{m-1}(C^{-1}A)C^{-1}p_1 + \psi_{m-1}(AC^{-1})p_1 \\ &= \varphi_{m-1}(AC^{-1})AC^{-1}p_1 + \psi_{m-1}(AC^{-1})p_1. \end{aligned}$$

Therefore $p_{m+1} \in \mathcal{K}_{m+1}(p_1, AC^{-1})$. Finally, to see that we can indeed equate the subspaces as (28a) states, write

$$\begin{aligned} \mathcal{K}_m(p_1, AC^{-1}) &= \mathcal{K}_m(p_1, [(A - \sigma B) + \sigma B](A - \sigma B)^{-1}) \\ &= \mathcal{K}_m(p_1, B(A - \sigma B)^{-1}). \end{aligned}$$

A similar reasoning leads to the proof for Q_m and the equality in (28b). $\square$

Theorem 4.1: *Let $m \leq n$, $C := A - \sigma B$, and denote the Krylov matrix arising from (28a) in Lemma 4.1 as*

$$K_m \equiv [p_1 \quad AC^{-1}p_1 \quad \dots \quad (AC^{-1})^{m-1}p_1].$$

There are m successful steps of Algorithm 1(a) for a symmetric pair (or Algorithm 2 for a nonsymmetric pair), iff the moment matrix $K_m^T C^{-1} K_m$ has nonsingular leading principal submatrices. If so, the factorization $K_m^T C^{-1} K_m = L_m D_m L_m^T$ (or $L_m D_m U_m$ in the nonsymmetric case) permits a simultaneous reduction $(Q_m^T A Q_m, Q_m^T B Q_m)$ that is tridiagonal-tridiagonal in the symmetric case with $P_m \equiv K_m L_m^{-T}$, $Q_m \equiv C^{-1} P_m D_m^{-1}$ (or Hessenberg-Hessenberg in the nonsymmetric case with $P_m \equiv K_m L_m^{-T}$, $Q_m \equiv C^{-1} K_m U_m^{-1} D_m^{-1}$).

Proof. Clearly, if no breakdown occurs in our algorithms up to m steps, the Krylov matrix K_m has full rank and the matrix L_m that solves the equation

$$K_m = P_m L_m^T \quad (29)$$

is a nonsingular lower triangular matrix (not necessarily with unit diagonal). Eq. (29) is a consequence of the nested

format of the $K_1, \dots, K_m$, and the correspondingly nested $P_1, \dots, P_m$ that are their respective bases by Lemma 4.1. Focusing for now on the symmetric case and using the equality $Q_m = C^{-1}P_m\Delta_m$ that is satisfied by construction of Algorithm 1(a), we infer from (29) that

$$K_m^T C^{-1} K_m = L_m P_m^T C^{-1} P_m L_m^T \quad (30a)$$

$$= L_m P_m^T Q_m \Delta_m^{-1} L_m^T \quad (30b)$$

$$= L_m \Delta_m^{-1} L_m^T. \quad (30c)$$

The last equality in (30) signifies that m successful steps of Algorithm 1(a) lead to the existence of the LDL^T factorization of the moment matrix $K_m^T C^{-1} K_m$, a property well known to require having nonzero leading principal minors, i.e., nonsingular leading principal submatrices.

To establish the converse, that the existence of the factorization implies the existence of the simultaneous reduction, assume now that we have the factorization

$$K_m^T C^{-1} K_m = L_m D_m L_m^T, \quad (31)$$

where L_m is lower triangular and D_m diagonal (using a different symbol than Δ_m to avoid a conflict of notation). A point to stress is that, in the standard LDL^T factorization (31), the factors are not contingent on a particular algorithm used to obtain them. If (31) does exist, take

$$\Delta_m \equiv D_m^{-1}, \quad P_m \equiv K_m L_m^{-T}, \quad (32a)$$

$$Q_m \equiv C^{-1} K_m L_m^{-T} \Delta_m = C^{-1} P_m \Delta_m. \quad (32b)$$

To see that (32) really fulfills the desired tridiagonal-tridiagonal reduction, observe that (31) and (32) automatically ensure the biorthogonality of P_m and Q_m because

$$\begin{aligned} P_m^T Q_m &= (K_m L_m^{-T})^T C^{-1} K_m L_m^{-T} \Delta_m \\ &= L_m^{-1} K_m^T C^{-1} K_m L_m^{-T} D_m^{-1} \\ &= L_m^{-1} L_m D_m L_m^T L_m^{-T} D_m^{-1} \\ &= I. \end{aligned}$$

We likewise verify the tridiagonalization through direct calculation, starting with

$$\begin{aligned} Q_m^T A Q_m &= Q_m^T A C^{-1} K_m L_m^{-T} \Delta_m \\ &= Q_m^T \begin{bmatrix} AC^{-1} p_1 & \dots & (AC^{-1})^m p_1 \end{bmatrix} L_m^{-T} \Delta_m \\ &= Q_m^T \begin{bmatrix} K_m \begin{bmatrix} 0 \\ I_{m-1} \end{bmatrix} & (AC^{-1})^m p_1 \end{bmatrix} L_m^{-T} \Delta_m \\ &= Q_m^T \begin{bmatrix} P_m L_m^T \begin{bmatrix} 0 \\ I_{m-1} \end{bmatrix} & (AC^{-1})^m p_1 \end{bmatrix} L_m^{-T} \Delta_m \\ &= \begin{bmatrix} L_m^T \begin{bmatrix} 0 \\ I_{m-1} \end{bmatrix} & Q_m^T (AC^{-1})^m p_1 \end{bmatrix} L_m^{-T} \Delta_m. \end{aligned}$$

The first matrix in the last equality is upper Hessenberg since $L_m^T \begin{bmatrix} 0 \\ I_{m-1} \end{bmatrix}$ is the upper triangular matrix L_m^T with its first column deleted. The subsequent post-multiplication by the upper triangular matrix $L_m^{-T} \Delta_m$ does not destroy the upper Hessenberg form. Hence $Q_m^T A Q_m$ is upper Hessenberg, and symmetry implies that it is tridiagonal. This is also the form

of $Q_m^T B Q_m$ since by definition (32), $Q_m = C^{-1} P_m \Delta_m$, i.e., $Q_m^T C Q_m = \Delta_m$, and so $\sigma Q_m^T B Q_m = Q_m^T A Q_m - \Delta_m$ since $C := A - \sigma B$. (Repeating the steps above in the nonsymmetric case extends the reasoning to the simultaneous Hessenberg-Hessenberg reduction using $K_m^T C^{-1} K_m = L_m D_m U_m$, $P_m \equiv K_m L_m^{-T}$, $Q_m \equiv C^{-1} K_m U_m^{-1} D_m^{-1} = C^{-1} P_m R_m$, $R_m \equiv L_m^T U_m^{-1} D_m^{-1}$.) $\square$

B. Equivalence of the simultaneous tridiagonalization with a SaI biorthogonal Lanczos

The reader has seen the circuitous route that led to Algorithm 1(b), which strikingly looks like a three-term Lanczos algorithm *after the fact*. Moreover, the equivalence shown in the preceding section IV-A connects the existence of the simultaneous reduction to the existence of the LDL^T or LDU factorization of the moment matrix $K_m^T C^{-1} K_m$, independently of our proposed algorithms. (And in fact, the LDL^T and LDU routes are yet more algorithms, albeit unpractical.) It is the sort of connection that is well engrained in the traditional Lanczos setting, as seen in [12, Sect. 5] or Saad [23] and elsewhere. From our presentation so far, one may wonder if an underlying Lanczos algorithm has not been hiding in plain sight all along. But exactly which variant it could be is still not immediate until revealed. The title of this section gives the game away and we now spotlight the findings that are new to our knowledge.

ALGORITHM 3: Biorthogonal Lanczos

1. Choose v_1 and w_1 with $v_1^T w_1 = 1$;
2. Set $\gamma_1 := 0$; $\tilde{\gamma}_1 := 0$; $v_0 := 0$; $w_0 := 0$;
3. **for** $k := 1 : m$
4. $\theta_k := w_k^T \hat{A} v_k$;
5. $\hat{v}_{k+1} := \hat{A} v_k - \theta_k v_k - \gamma_k v_{k-1}$;
6. $\hat{w}_{k+1} := \hat{A}^T w_k - \theta_k w_k - \tilde{\gamma}_k w_{k-1}$;
7. $\gamma_{k+1} := |\hat{w}_{k+1}^T \hat{v}_{k+1}|^{1/2}$;
8. $\tilde{\gamma}_{k+1} := \text{sign}(\hat{w}_{k+1}^T \hat{v}_{k+1}) \gamma_{k+1}$;
9. $v_{k+1} := \hat{v}_{k+1} / \gamma_{k+1}$;
10. $w_{k+1} := \hat{w}_{k+1} / \tilde{\gamma}_{k+1}$;
11. **endfor**

For completeness and ease of reference, we have reproduced in Algorithm 3 the biorthogonal Lanczos process that takes as input a general matrix $\hat{A}$ and two starting vectors v_1 and w_1 , with $v_1^T w_1 = 1$. Note that the chosen scaling strategy in lines 7-8 lead to coefficients that satisfy $\gamma_{k+1} = \pm \tilde{\gamma}_{k+1}$, and consequently, v_{k+1} and w_{k+1} in lines 9-10 are scaled from $\hat{v}_{k+1}$ and $\hat{w}_{k+1}$ with the same factor in magnitude, much like the impact of (17a). Assuming no breakdown up to m steps, Algorithm 3 will return $V_m \equiv [v_1, \dots, v_m]$ and $W_m \equiv [w_1, \dots, w_m]$ that are biorthogonal, and if we arrange the computed coefficients into the tridiagonal matrix

$$\Upsilon_m = \begin{pmatrix} \theta_1 & \gamma_2 & & \\ \tilde{\gamma}_2 & \theta_2 & \ddots & \\ & \ddots & \ddots & \gamma_m \\ & & \tilde{\gamma}_m & \theta_m \end{pmatrix},$$

we have these well-known fundamental relations

$$\begin{cases} \hat{A}V_m &= V_m \Upsilon_m + \tilde{\gamma}_{m+1}v_{m+1}e_m^T \\ \hat{A}^TW_m &= W_m \Upsilon_m^T + \gamma_{m+1}w_{m+1}e_m^T \\ W_m^TV_m &= I_m \end{cases} \quad (33)$$

With the above background and notation, we can now state the equivalence below, in which we use Algorithm 1(a) for uniformity of indexing.

Theorem 4.2: *Consider a regular symmetric pair (A, B) and a SaI parameter σ such that $C := A - \sigma B$ is nonsingular. Suppose that the simultaneous reduction Algorithm 1(a) is applied with a starting vector p_1 , and that the biorthogonal Lanczos Algorithm 3 is applied using the general matrix $\hat{A} := C^{-1}A$ with matching starting vectors $w_1 := p_1$ and $v_1 := q_1$ ($\equiv C^{-1}p_1/p_1^T C^{-1}p_1$) as in Algorithm 1(a). If the algorithms do not break down up to iteration m , then $\forall k = 2 : m + 1$,*

$$q_k = \pm v_k, \quad p_k = \pm w_k, \quad \beta_k = \pm \gamma_k = \pm \tilde{\gamma}_k; \quad (34a)$$

and $\forall k = 1 : m$,

$$\alpha_k = \pm \theta_k. \quad (34b)$$

Proof. On one hand, if in Algorithm 3 that implements the biorthogonal Lanczos we input $\hat{A} := C^{-1}A$, then from (33), we get on return

$$\begin{cases} (C^{-1}A)V_m = V_m \Upsilon_m + \tilde{\gamma}_{m+1}v_{m+1}e_m^T \\ (C^{-1}A)^TW_m = W_m \Upsilon_m^T + \gamma_{m+1}w_{m+1}e_m^T \end{cases} \quad (35)$$

On the other hand, by construction of Algorithm 1(a) using the scaling choice in Remark 2.1, the truncated forms of (15) and (18) amount to

$$Q_m = C^{-1}P_m\Delta_m, \quad \Delta_m = J_m = \text{diag}(\pm 1), \quad (36)$$

$$AQ_m = P_mT_m + \beta_{m+1}p_{m+1}e_m^T. \quad (37)$$

Keeping in mind (36) and the fact that $\Delta_m^2 = I$, multiply (37) on the left by C^{-1} and on the right by Δ_m ,

$$\begin{aligned} C^{-1}AQ_m\Delta_m &= C^{-1}P_m\Delta_m\Delta_mT_m\Delta_m + \beta_{m+1}C^{-1}p_{m+1}e_m^T\Delta_m \\ &= Q_m\Delta_mT_m\Delta_m + \delta_m\beta_{m+1}\delta_{m+1}q_{m+1}e_m^T. \end{aligned} \quad (38)$$

If we go back to (37) and substitute Q_m using (36), we also have

$$AC^{-1}P_m\Delta_m = P_m\Delta_m\Delta_mT_m + \beta_{m+1}p_{m+1}e_m^T. \quad (39)$$

Now reprint (38) and (39) together, we get the following coupled equalities (with added parentheses for clarity), and recall that A , C and T_m are symmetric,

$$\begin{cases} (C^{-1}A)(Q_m\Delta_m) = (Q_m\Delta_m)(T_m\Delta_m) + (\delta_m\beta_{m+1})(\delta_{m+1}q_{m+1})e_m^T \\ (C^{-1}A)^T(P_m\Delta_m) = (P_m\Delta_m)(T_m\Delta_m)^T + \beta_{m+1}p_{m+1}e_m^T \end{cases} \quad (40)$$

It becomes apparent at this point that (35) and (40) describe essentially identical relations because the Lanczos vectors

are unique to within a diagonal scaling once the starting vectors are specified. With the scaling strategies adopted in the algorithms and the choice of identical starting vectors, the uniqueness holds up to signs as asserted in (34). $\square$

The equivalence stated in Theorem 4.2 establishes the capability of the biorthogonal Lanczos algorithm to perform the simultaneous tridiagonalization of Theorem 2.1 when using appropriate starting vectors and SaI operations. Even with the benefit of hindsight at this stage, the claim is not self-evident in retrospect, and this may explain why it was camouflaged for this long. Intriguingly too, Algorithm 1 acts like a transpose-free incarnation of Algorithm 3, since its steps use only matrix-vector products and SaI operations.

Now that we have the equivalence, extrapolating that it makes Theorem 2.1 free of conditions would still be premature because the biorthogonal Lanczos algorithm can itself break down. It is however well known that *serious* breakdowns may be avoided through differently chosen starting vectors (v_1, w_1) , see, e.g., [24]. But the equivalence shown in Theorem 4.2 requires that these two vectors satisfy the dependent relation $v_1 = C^{-1}w_1/w_1^TC^{-1}w_1$, narrowing the choices and leaving unclear the existence under this attached requirement. Hence, a rigorous justification is still needed (not pursued here) to fully characterize the *existence of the complete simultaneous reduction*.

V. REVISITING THE REDUCED PROBLEM IN APPLICATIONS

A. Complex variants

Our presentation has so far deliberately sought to preserve real arithmetic for applications with real data. We now discuss adaptations to account for complex data in the algorithmic details and the equivalences shown in the previous sections. A complex shift σ gives an immediate example that triggers complex arithmetic, with furthermore $A - \sigma B$ not Hermitian even if A and B are symmetric.

To clearly see what happens in the broader complex setting, we revisit the details of the discussion in section IV-A. Assume we have the more general factorization

$$K_m^*C^{-1}K_m = L_mD_mU_m,$$

where the factors are now complex, e.g., due to $\sigma \in \mathbb{C}$, and define

$$P_m \equiv K_mL_m^{-*}, \quad Q_m \equiv C^{-1}K_mU_m^{-1}D_m^{-1} = C^{-1}P_mR_m,$$

$$R_m \equiv L_m^*U_m^{-1}D_m^{-1} = P_m^*(A - \sigma B)P_m,$$

Thus $P_m^*Q_m = I$ and R_m is upper triangular. Moreover, from the steps of the discussion done earlier, $H_m \equiv Q_m^*AQ_m$ and $G_m \equiv Q_m^*BQ_m$ can be shown to be upper Hessenberg. What this implies is that Theorem 3.1 still holds in the general complex setting.

It then follows that when the pair (A, B) is symmetric or Hermitian, we infer that the upper Hessenberg factors satisfy $H_m^* = H_m$ and $G_m^* = G_m$, thus implying that $T_m \equiv H_m$ and $S_m \equiv G_m$ are tridiagonal Hermitian. Hence, when σ is

complex and the pair is symmetric or Hermitian, taking the conjugate transpose of both sides of

$$Q_m^* A Q_m - \sigma Q_m^* B Q_m = R_m$$

we get

$$Q_m^* A Q_m - \bar{\sigma} Q_m^* B Q_m = R_m^*,$$

so that subtracting the two equations, we have

$$-(\sigma - \bar{\sigma}) S_m = R_m - R_m^*.$$

Consequently, in order for $R_m - R_m^*$ to conform with the tridiagonal Hermitian S_m , the matrix R_m , which is already upper triangular, must be upper bidiagonal; and it further collapses to diagonal real whenever $\sigma - \bar{\sigma} = 0$, i.e., σ real. All this suggests slight adjustments in Theorem 2.1 to accommodate real or complex shifts and pairs that are symmetric or Hermitian: the tridiagonal-tridiagonal form still holds but a diagonal or bidiagonal matrix may arise in (2). Also note the subtlety that the tridiagonal factor T_m always remains entirely real since the diagonal of T_m and S_m are already real by being Hermitian matrices, with additionally the off-diagonal of T_m real by the construction choice (26); and $T_m - \sigma S_m = R_m$ also leads to S_m real whenever R_m is diagonal real (i.e., when σ is real as noted above). In summary, there can be customized three-term variants for symmetric or Hermitian pairs using real or complex shifts although we omit algorithmic details for each variant here. The main takeaway is that complex shifts (or complex matrices in general) can be handled at the cost of complex arithmetic.

B. Generalized eigenvalue problem (GEP)

We now take a further look at the projected problem resulting from the large sparse generalized eigenvalue problem $Ax = \lambda Bx$ since it is one of the most prominent applications suitable for a simultaneous reduction. After m successful steps of Algorithm 1 for a symmetric pair, suppose that we have obtained

$$(Q_m^T A Q_m, Q_m^T B Q_m) = (T_m, S_m),$$

where $Q_m \in \mathbb{R}^{n \times m}$, $T_m, S_m \in \mathbb{R}^{m \times m}$. Letting $x = Q_m y$, the original $Ax = \lambda Bx$ can thus be converted to a smaller m -by- m problem

$$Q_m^T A Q_m y = \lambda Q_m^T B Q_m y \Rightarrow T_m y = \lambda S_m y, \quad (41)$$

which can be considerably easier to deal with when $m \ll n$, as is typical with projection methods. Recall that we can choose $T_m - \sigma S_m = \Delta_m = \text{diag}(\pm 1) \equiv J_m$, we shall use J_m here to emphasize the fact that it is a signature matrix and so $J_m^{-1} = J_m$. Subtracting $\sigma S_m y$ from both sides of the last equality in (41), we get

$$(T_m - \sigma S_m) y = (\lambda - \sigma) S_m y \quad (42a)$$

$$\Rightarrow J_m y = (\lambda - \sigma) S_m y \quad (42b)$$

$$\Rightarrow y = (\lambda - \sigma) J_m S_m y, \quad (42c)$$

which simplifies to a *standard* tridiagonal eigenvalue problem

$$\hat{S}_m y = \hat{\lambda} y, \quad \hat{S}_m \equiv J_m S_m, \quad \hat{\lambda} \equiv \frac{1}{\lambda - \sigma}, \quad (43)$$

where furthermore $\hat{S}_m$ has the special feature of being *pseudosymmetric*. It is remarkable that the simple process in Algorithm 1(b) can give this elegant outcome at no extra cost, pointing to how the equivalent SaI Lanczos formulation of section IV-B has had this capacity all along. This also adds interest to [1], [2] for solving the dense problem. The resulting tridiagonal eigenvalue problem (43) itself is well known, e.g., it is discussed by Parlett and Chen [12, Sect. 8]. In [25], Brebner and Grad specifically tuned the HR method to exploit the pseudosymmetric structure, and there is the more recent root finding approach of Bini et al. [26] that improves numerical stability by not expanding $\det(\hat{S}_m - \hat{\lambda} I)$ into a polynomial representation.

In the nonsymmetric case where the simultaneous reduction using Algorithm 2 leads to $H_m y = \lambda G_m y$ with H_m and G_m upper Hessenberg, and $H_m - \sigma G_m = R_m$ upper triangular, we end up with a reduced problem in Hessenberg-triangular form,

$$(H_m - \sigma G_m) y = (\lambda - \sigma) G_m y \quad (44a)$$

$$\Rightarrow R_m y = (\lambda - \sigma) G_m y \quad (44b)$$

$$\Rightarrow G_m y = \hat{\lambda} R_m y, \quad \hat{\lambda} \equiv \frac{1}{\lambda - \sigma}. \quad (44c)$$

This therefore achieves the preliminary step of the QZ method at no extra cost, and so even the nonsymmetric variant (27) opens a neat gateway to computing the Ritz values with classical methods for dense problems as found in textbooks such as [27].

Remarks

- As Lemma 4.1 made clear, the algorithms build a basis of $\mathcal{K}_m(q_1, (A - \sigma B)^{-1} B)$, which is the same search subspace of the standard shift-and-invert Lanczos and Arnoldi methods. Since Krylov subspace methods tend to approximate dominant eigenvalues, (43) and (44) suggest that the computed Ritz values would approximate the shifted reciprocals of eigenvalues close to σ , as standard methods do, reminding us again that the algorithms are incarnations of these methods.

- The algorithms are implemented in terms of matrix-vector products with A , but it is easy to see from our derivation that they can be instead reformulated in terms of matrix-vector products with B (or the roles of A and B can be switched), allowing to use whichever matrix-vector product is cheaper/convenient.

- We saw earlier how a complex shift brings in complex arithmetic. We would like to add that, with a σ different from the one used in the reduction, we still have $(T_m - \sigma S_m) y = (\lambda - \sigma) S_m y$ in (42) or $(H_m - \sigma G_m) y = (\lambda - \sigma) G_m y$ in (44), suggesting that the decomposition can be kept across different shifts. Such developments are however beyond the scope of the discussion intended here. And given the extensive body of work that exists on the GEP, it remains to be seen

if the ability to have multiple shifts and/or retain all the complex information brings any numerical advantage over past shift-and-invert approaches that are formulated in terms of inner products needing to avoid complex arithmetic by using $\Re[(A - \sigma B)^{-1}]$ or $\Im m[(A - \sigma B)^{-1}]$ as shown in [12, Sect. 3].

• If the reduction is applied to $(A, B + \gamma I)$, we get (in the symmetric case for example)

$$\begin{aligned} Q_m^T A Q_m &= T_m, & Q_m^T (B + \gamma I) Q_m &= S_m, \\ Q_m^T (A - \sigma(B + \gamma I)) Q_m &= T_m - \sigma S_m = \Delta_m. \end{aligned}$$

We can thus project the generalized eigenvalue problem $Ax = \lambda Bx$ as $Q_m^T A Q_m y = \lambda Q_m^T B Q_m y$ so that

$$T_m y = \lambda (S_m - \gamma Q_m^T Q_m) y, \quad x = Q_m y, \quad (45)$$

showing that we get a reduced tridiagonal-general problem with the parameter γ as a stabilizer of the reduction. The tridiagonal-tridiagonal (or Hessenberg-Hessenberg in the non-symmetric case) can thus be traded with a stabilized form that is tridiagonal-general (or Hessenberg-general). This is another intriguing upshot that entails no change in our proposed algorithms, while enabling a suitable choice of γ that may seek to make the shift-and-invert operation with $(A - \sigma(B + \gamma I))^{-1}$ more robust, e.g., from indefinite to definite and breakdown-free. The added flexibility comes with the question as to whether subsequent solvers could possibly exploit the special structure of the general term in (45) without expanding $Q_m^T Q_m$ explicitly.

C. Others

Aside from the above and the examples cited in the introduction, we can also mention an application to special pairs such as $(A, B) = (A, A^T)$ with $\sigma = -1$, yielding T-palindromic pencils [28], although it is not our intention to be exhaustive.

VI. NUMERICAL EXPERIMENTS

The matrices used in the experiments come from the popular Harwell-Boeing collection [29] and the set of NEPs (Non-Hermitian Eigenvalue Problems) of Bai et al. [30]. Table I lists the problems and some of their characteristics: n , $nnz(A)$, $nnz(B)$, which are, respectively, the order and number of nonzero entries in A and B . The symbol SYM indicates symmetric problems.

We ran the MATLAB scripts in the appendix with three different values of σ and a random starting vector obtained with the built-in function `rand(n,1)` and we reset the generator using `rng('default')` between calls. First, we take $\sigma = 1$, which is an obvious choice that many users may be tempted to use. Secondly, we take $\sigma = \pm \|A\|_1 / \|B\|_1$, which was suggested by Garvey et al. [1, Sect. 5]. The effective sign is chosen to maximize $\|A - \sigma B\|_1$, thus minimizing the risk of cancellation when computing $A - \sigma B$. Here is the snippet of MATLAB code used in the experiments:

```
sigma = norm(A,1)/norm(B,1);
if norm(A+sigma*B,1) > norm(A-sigma*B,1)
    sigma = -sigma;
end
```

Finally, we take $\sigma = \sigma_{\text{opt}} \approx \text{argmin}_{\gamma} \text{cond}_1(A - \gamma B)$, where $\text{cond}_1(X) = \|X\|_1 \|X^{-1}\|_1$ is the 1-norm condition number of the matrix X . We estimated it with MATLAB's built-in function `condest` to avoid the explicit computation of the full blown inverse of a large matrix. Solving such a costly optimization problem is not meant to be done in practice and was merely done here for reference. We used MATLAB's built-in function `fminsearch`, which attempts to find a local minimizer of the objective function, and we supplied the initial guess $\sigma = \pm \|A\|_1 / \|B\|_1$ just discussed. Our experiments show that the easy choice $\sigma = 1$ may not generally be the ideal default. By contrast, the choice $\sigma = \pm \|A\|_1 / \|B\|_1$ proved quite effective. This is all the more appealing because it is relatively simple to compute. The values of the resulting condition numbers are reported on Table II.

TABLE I
PROBLEMS AND THEIR CHARACTERISTICS

	(A, B)	n	$nnz(A)$	$nnz(B)$
SYM	lund_a, lund_b	147	2449	2441
SYM	crystk01, crystm01	4875	315891	105339
SYM	bcsstk38, bcsstm38	8032	355460	10485
	bfwa398, bfwb398	398	3678	2910
	dw256A, dw256B	512	2480	2500
	dwa512, dwb512	512	2480	2500
	bfwa782, bfwb782	782	7514	5982
	odepa400, odepb400	400	1201	399
	mhda416, mhdb416	416	8562	2312
	mhd3200a, mhd3200b	3200	68026	18316
	mhd4800a, mhd4800b	4800	102252	27520

TABLE II
VALUES OF THE RESULTING CONDITION NUMBERS

Pair (A, B)	$\text{cond}_1(A - \sigma B)$			
	$\sigma = 1$	$\sigma = \pm \frac{\ A\ _1}{\ B\ _1}$	$\sigma = \sigma_{\text{opt}}$	
SYM lund_a, lund_b	5.5e+06	6.7e+04	4.9e+04	
SYM crystk01, crystm01	6.6e+05	1.8e+02	1.7e+02	
SYM bcsstk38, bcsstm38	6.3e+16	1.2e+17	9.1e+15	
	bfwa398, bfwb398	7.6e+03	2.0e+01	
	dw256A, dw256B	2.9e+01	4.4e+00	
	dwa512, dwb512	2.9e+01	4.4e+00	
	bfwa782, bfwb782	4.6e+03	2.2e+01	
	odepa400, odepb400	8.3e+05	4.8e+02	
	mhda416, mhdb416	Inf	6.7e+09	
	mhd3200a, mhd3200b	Inf	2.7e+13	
	mhd4800a, mhd4800b	Inf	1.4e+14	

Results are presented on Table III. We assess the methods using the scaled residuals and a measure of the loss of biorthogonality:

$$\begin{aligned} \text{Res}_A &= \frac{\|T_m - Q_m^T A Q_m\|_2}{\|A\|_2 \|Q_m\|_2^2}, \quad \text{Res}_B = \frac{\|S_m - Q_m^T B Q_m\|_2}{\|B\|_2 \|Q_m\|_2^2}, \\ \text{BiOrth} &= \frac{\|I - P_m^T Q_m\|_2}{\|P_m\|_2 \|Q_m\|_2}, \end{aligned}$$

where T_m and S_m should be substituted with H_m and G_m for nonsymmetric pairs. The quantity BiOrth gives a sense of the loss of biorthogonality after m steps, whereas Res_A and Res_B check the accuracy of the reduction processes. The

TABLE III
THE QUANTITIES BiOrth, Res_A, Res_B FOR DIFFERENT σ AT $m = 15$

Pair (A, B)	$\sigma = 1$			$\sigma = \pm \ A\ _1 / \ B\ _1$			$\sigma = \sigma_{\text{opt}}$		
	BiOrth	Res _A	Res _B	BiOrth	Res _A	Res _B	BiOrth	Res _A	Res _B
SYM lund_a, lund_b	E-04	E-07	E-05	E-15	E-18	E-17	E-16	E-18	E-18
SYM crystk01, crystm01	E-03	E-03	E-01	E-15	E-16	E-16	E-15	E-16	E-16
SYM bcsstk38, bcsstm38	E-05	E-17	E-13	E-16	E-27	E-28	E-06	E-13	E-13
bfwa398, bfwb398	E-02	E-04	E+02	E-14	E-17	E-15	E-15	E-17	E-15
dw256A, dw256B	E-15	E-16	E-15	E-15	E-16	E-16	E-15	E-16	E-15
dwa512, dwb512	E-15	E-16	E-15	E-15	E-16	E-16	E-15	E-16	E-15
bfwa782, bfwb782	E-02	E+78	—	E-14	E-17	E-15	E-14	E-17	E-15
odepa400, odepb400	—	—	—	E-14	E-17	E-15	E-13	E-15	E-13
mhda416, mhdb416	NA	NA	NA	E-17	E-22	E-21	E-17	E-23	E-22
mhd3200a, mhd3200b	NA	NA	NA	E-19	E-27	E-25	E-19	E-28	E-25
mhd4800a, mhd4800b	NA	NA	NA	E-18	E-28	E-25	E-19	E-28	E-25

machine epsilon is about 10^{-16} when using double precision floating-point arithmetic. Table III gives a snapshot for all problems after $m = 15$ iterations. We clearly see that the tempting choice $\sigma = 1$ is riskier. On the reported problems, it had the most high condition numbers and failures in the algorithms, c.f. Tables II and III. It is worth clarifying that when $\text{cond}_1(A - \sigma B) = \text{Inf}$ with $\sigma = 1$ on Table II, this already implies that the reduction is not applicable, explaining their corresponding NA on Table III. The markers “—” indicate other failures resulting from breakdowns, which may be detected by checking for $q_k^T p_k \approx 0$ in a production code, but the prototype MATLAB scripts made no provision to detect these. For the other choices of σ , the residuals indicate that the methods can be fairly effective.

On the whole, we see that the proposed methods succeed when given a good choice of σ , and with the measure of the loss of biorthogonality more pronounced than the loss of accuracy assessed from the residuals. This reminds us how the risk of the loss of orthogonality in the standard Lanczos has motivated partial or selective re-orthogonalization strategies there. Our simultaneous setting, which embeds furthermore shift-and-invert operations, can only add to the challenge. The gist of our presentation was to introduce the proposed methods. The scope here can only be a proof-of-concept compared to the extensive body of work that has already gone in the Lanczos and Arnoldi settings. Whether an occurrence of the loss of biorthogonality in our simultaneous variants has any bearing with the convergence of generalized eigenvalues is an open question, among restart or deflation strategies and the other directions of further investigation that could spring from here.

VII. CONCLUSION

We have described methods to simultaneously reduce a regular matrix pair (A, B) to tridiagonal-tridiagonal form when the pair is symmetric, or to Hessenberg-Hessenberg form when the pair is nonsymmetric. The methods require shift-and-invert (SaI) operations but make no other assumption (such as requiring B to be invertible and/or positive definite). We showed how the symmetric variant was a SaI Lanczos in disguise with its benefits (three-term recurrence, matrix free), while the nonsymmetric variant was its Arnoldi-like

counterpart. The discovery of such connections is novel. We also discussed how the methods extend from real to complex settings, and illustrated their usefulness by showing how they project a large sparse generalized eigenvalue problem onto a reduced one that can be passed directly to classical solvers expecting the standard tridiagonal form in the symmetric case, or the Hessenberg-triangular form in the nonsymmetric case. But further work is needed to assess how the overall process compares numerically with traditional approaches in practice, as well as other ramifications in applications that may benefit from a partial simultaneous reduction in general.

APPENDIX

```

function [T, S, D, Q, P] = spritri(A, B, s, m)
% sparse tridiagonal-tridiagonal reduction with a
% 3-term recurrence of the symmetric pair (A,B)
% by a congruence transformation (possibly partial)
% Input:
% - symmetric pair (A,B)
% - nonzero real shift parameter s such that
%   A-s*B is invertible
% - optional number of steps m. If omitted,
%   m is set to the order of the matrices n,
%   and a complete reduction should be assumed.
% Output:
% - T, S: tridiagonal, D: diagonal. All size m-by-m
% - Q, P: rectangular of size n-by-m, such that:
% - Q'*AQ = T; Q'*B*Q = S; P'*Q = I; T - s*S = D

n = length(A);
if nargin < 4, m = n; end
C = A - s*B;
p = rand(n,1); % or another vector
for k = 1:m
    q = C\p;
    dot = p'*q;
    betak = sqrt(abs(dot));
    D(k,k) = sign(dot); % betak^2/dot
    P(:,k) = p*(1/betak);
    Q(:,k) = q*(D(k,k)/betak);
    p = A*Q(:,k);
    if k > 1
        p = p - betak*P(:,k-1);
        T(k,k-1) = betak; T(k-1,k) = T(k,k-1);
        S(k,k-1) = betak/s; S(k-1,k) = S(k,k-1);
    end
    T(k,k) = Q(:,k)'*p;
    S(k,k) = (T(k,k) - D(k,k))/s;
    if k == n, break, end
    p = p - T(k,k)*P(:,k);
end

```

```

function [H, G, R, Q, P] = sphesshess(A, B, s, m)
% Hessenberg-Hessenberg reduction of the sparse
% general pair (A,B) by congruence transformation
% (possibly partial)
% Input:
% - general pair (A,B)
% - nonzero shift parameter s such that
%   A-s*B is invertible
% - optional number of steps m. If omitted,
%   m is set to the order of the matrices n,
%   and a complete reduction should be assumed.
% Output:
% - H, G: upper Hessenberg with an extra row
%   (like the Hbar matrix in Arnoldi),
%   both are rectangular of size (m+1)-by-m
% - Q, P: rectangular of size n-by-(m+1)
%   such that: P'*Q = I; and:
% - Q(:,1:m)'*A*Q(:,1:m) = H(1:m,1:m);
% - Q(:,1:m)'*B*Q(:,1:m) = G(1:m,1:m);
% - H(1:m,1:m) - s*G(1:m,1:m) = R(1:m,1:m);
% - R upper triangular

n = length(A);
if nargin < 4, m = n; end;
C = A-s*B;
p = rand(n,1); % or another vector
q = C\p;
dot = p'*q;
sdot = sqrt(abs(dot));
r11 = sdot^2/dot; % sign(dot) if dot is real
S(1,1) = 1/r11;
P(:,1) = p*(1/sdot);
Q(:,1) = q*(r11/sdot);
for k = 1:m
    p = A*Q(:,k);
    for i = 1:k
        H(i,k) = Q(:,i)'*p;
        p = p - H(i,k)*P(:,i);
    end
    if k == n, break; end
    q = C\p;
    dot = p'*q;
    H(k+1,k) = sqrt(abs(dot));
    rk1k1 = H(k+1,k)^2/dot; % sign(dot) if dot is real
    S(k+1,k+1) = 1/rk1k1;
    P(:,k+1) = p*(1/H(k+1,k));
    for i = 1:k
        dot = P(:,i)'*q;
        q = q - dot*P(:,i);
        S(i,k+1) = dot/H(k+1,k);
    end
    Q(:,k+1) = (rk1k1/H(k+1,k))*q;
end
R = inv(S);
% recover G = (H-R)/s;
mk = size(H,2);
G = H; G(1:mk,1:mk) = G(1:mk,1:mk)-R(1:mk,1:mk);
G = G/s;

```

REFERENCES

- [1] S. D. Garvey, F. Tisseur, M. I. Friswell, J. E. T. Penny, and U. Prells, "Simultaneous tridiagonalization of two symmetric matrices," *Int. J. Numer. Meth. Engng.*, vol. 57, no. 12, pp. 1643–1660, 2003. doi: <https://doi.org/10.1002/nme.733>
- [2] R. B. Sidje, "On the simultaneous tridiagonalization of two symmetric matrices," *Numer. Math.*, vol. 118, no. 3, pp. 549–566, 2011. doi: <https://doi.org/10.1007/s00211-010-0357-9>
- [3] F. Tisseur, "Tridiagonal-diagonal reduction of symmetric indefinite pairs," *SIAM J. Matrix Anal. Appl.*, vol. 26, no. 1, pp. 215–232, 2004. doi: <https://doi.org/10.1137/S0895479802414783>
- [4] B. Nour-Omid, B. N. Parlett, T. Ericsson, and P. S. Jensen, "How to implement the spectral transformation," *Math. Comp.*, vol. 48, pp. 663–673, 1987. doi: <https://doi.org/10.1090/s0025-5718-1987-0878698-5>
- [5] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*. Society for Industrial and Applied Mathematics, 2011, doi: <https://doi.org/10.1137/1.9781611970739>.
- [6] T. Ericsson and A. Ruhe, "The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems," *Math. Comp.*, vol. 35, no. 152, pp. 1251–1268, 1980. doi: <https://doi.org/10.2307/2006390>
- [7] D. C. Sorensen, "Truncated QZ methods for large scale generalized eigenvalue problems," *ETNA*, vol. 7, pp. 141–162, 1998. [Online]. Available: <https://eudml.org/doc/119806>
- [8] R.-C. Li and Q. Ye, "Simultaneous similarity reductions for a pair of matrices to condensed forms," *Commun. Math. Stat.*, vol. 2, no. 2, pp. 139–153, 2014. doi: <https://doi.org/10.1007/s40304-014-0033-y>
- [9] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, "Recycling Krylov subspaces for sequences of linear systems," *SIAM J. Sci. Comput.*, vol. 28, no. 5, p. 1651–1674, Sep. 2006. doi: <https://doi.org/10.1137/040607277>
- [10] B. N. Parlett, "A new look at the Lanczos algorithm for solving symmetric systems of linear equations," *Linear Algebra Appl.*, vol. 29, pp. 323–346, 1980. doi: [https://doi.org/10.1016/0024-3795\(80\)90248-7](https://doi.org/10.1016/0024-3795(80)90248-7)
- [11] Y. Saad, "On the Lanczos method for solving symmetric linear systems with several right-hand sides," *Math. Comp.*, vol. 48, pp. 651–662, 1987. doi: <https://doi.org/10.2307/2007834>
- [12] B. N. Parlett and H. C. Chen, "Use of indefinite pencils for computing damped natural modes," *Linear Algebra Appl.*, vol. 140, pp. 53–88, 1990. doi: [https://doi.org/10.1016/0024-3795\(90\)90222-X](https://doi.org/10.1016/0024-3795(90)90222-X)
- [13] G. H. Golub and Q. Ye, "An inverse free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems," *SIAM J. Sci. Comput.*, vol. 24, no. 1, pp. 312–334, 2002. doi: <https://doi.org/10.1137/S1064827500382579>
- [14] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst, "Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems," *BIT Num. Math.*, vol. 36, no. 3, pp. 595–633, 1996. doi: <https://doi.org/10.1007/BF01731936>
- [15] M. A. Freitag, A. Spence, and E. Vainikko, "Rayleigh quotient iteration and simplified Jacobi-Davidson with preconditioned iterative solves for generalised eigenvalue problems," Department of Mathematics, University of Bath, Bath, UK, Tech. Rep., 2008.
- [16] D. B. Szyld and F. Xue, "Efficient preconditioned inner solves for inexact Rayleigh quotient iteration and their connections to the single-vector Jacobi-Davidson method," *SIAM J. Mat. Anal. Appl.*, vol. 32, pp. 993–1018, 2011. doi: <https://doi.org/10.1137/100807922>
- [17] F. Tisseur and K. Meerbergen, "The quadratic eigenvalue problem," *SIAM Review*, vol. 43, no. 2, pp. 235–286, 2001. doi: <https://doi.org/10.1137/S0036144500381988>
- [18] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 2012, doi: <https://doi.org/10.1017/CBO9780511810817>.
- [19] Z. Bai, "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems," *Appl. Num. Math.*, vol. 43, no. 1, pp. 9–44, 2002. doi: [https://doi.org/10.1016/S0168-9274\(02\)00116-2](https://doi.org/10.1016/S0168-9274(02)00116-2)
- [20] Y. Chahlaoui, K. A. Gallivan, A. Vandendorpe, and P. Van Dooren, "Model reduction of second-order systems," in *Dimension Reduction of Large-Scale Systems*, P. Benner, D. C. Sorensen, and V. Mehrmann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. doi: <https://doi.org/10.1007/3-540-27909-1>. ISBN 978-3-540-27909-9 pp. 149–172.
- [21] K. Gallivan, G. Grimme, and P. Van Dooren, "A rational Lanczos algorithm for model reduction," *Numer. Algorithms*, vol. 12, no. 1, pp. 33–63, 1996. doi: <https://doi.org/10.1007/BF02141740>
- [22] B. N. Parlett, D. R. Taylor, and Z. A. Liu, "A look ahead Lanczos algorithm for unsymmetric matrices," *Math. Comp.*, vol. 44, pp. 105–124, 1985. doi: <https://doi.org/10.2307/2007796>
- [23] Y. Saad, "The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems," *SIAM J. Numer. Anal.*, vol. 19, no. 3, pp. 485–506, 1982. doi: <https://doi.org/10.1137/0719031>
- [24] W. Joubert, "Lanczos methods for the solution of nonsymmetric systems of linear equations," *SIAM J. Mat. Anal. Appl.*, vol. 13, no. 3, pp. 926–943, 1992. doi: <https://doi.org/10.1137/0613056>
- [25] M. A. Brebner and J. Grad, "Eigenvalues of $Ax = \lambda Bx$ for real symmetric matrices A and B computed by reduction to a pseudosymmetric form and the HR process," *Linear Algebra Appl.*, vol. 43, pp. 99–118, 1982. doi: [https://doi.org/10.1016/0024-3795\(82\)90246-4](https://doi.org/10.1016/0024-3795(82)90246-4)

- [26] D. Bini, L. Gemignani, and F. Tisseur, "The Ehrlich–Aberth method for the nonsymmetric tridiagonal eigenvalue problem," *SIAM J. Mat. Anal. Appl.*, vol. 27, no. 1, pp. 153–175, 2005. doi: <https://doi.org/10.1137/S0895479803429788>
- [27] G. H. Golub and C. F. Van Loan, *Matrix Computations - 4th Edition*. Philadelphia, PA: Johns Hopkins University Press, 2013, doi: 10.1137/1.9781421407944. [Online]. Available: <https://epubs.siam.org/doi/abs/10.1137/1.9781421407944>
- [28] F. De Terán, "Canonical forms for congruence of matrices and T-palindromic matrix pencils: a tribute to H. W. Turnbull and A. C. Aitken," *SeMA Journal*, vol. 73, no. 1, pp. 7–16, 2016. doi: <https://doi.org/10.1007/s40324-015-0052-y>
- [29] I. S. Duff, R. G. Grimes, and J. G. Lewis, "Sparse matrix test problems," *ACM Trans. Math. Software*, vol. 15, no. 1, pp. 1–14, 1989. doi: <https://doi.org/10.1145/62038.62043>
- [30] Z. Bai, D. Day, J. W. Demmel, and J. J. Dongarra, "A test matrix collection for non-Hermitian eigenvalue problems (release 1.0)," Department of Computer Science, University of Tennessee, Knoxville, TN, USA, LAPACK Working Note 123, Tech. Rep. UT-CS-97-355, Mar. 1997. [Online]. Available: <http://www.netlib.org/lapack/lawnspdf/lawn123.pdf>