# Computing Matrix Symmetrizers, Part 2: new Methods using Eigendata and Linear Means; a Comparison*

Froilán Dopico† and Frank Uhlig‡

**Abstract**

Over any field $\mathbb{F}$ every square matrix $A$ can be factored into the product of two symmetric matrices as $A = S_1 \cdot S_2$ with $S_i = S_i^T \in \mathbb{F}^{n,n}$ and either factor can be chosen nonsingular, as was discovered by Frobenius in 1910.
Frobenius' symmetric matrix factorization has been lying almost dormant for a century. The first successful method for computing matrix symmetrizers, i.e., symmetric matrices $S$ such that $SA$ is symmetric, was inspired by an iterative linear systems algorithm of Huang and Nong (2010) in 2013 [29, 30]. The resulting iterative algorithm has solved this computational problem over $\mathbb{R}$ and $\mathbb{C}$, but at high computational cost.
This paper develops and tests another linear equations solver, as well as eigen- and principal vector or Schur Normal Form based algorithms for solving the matrix symmetrizer problem numerically. Four new eigendata based algorithms use, respectively, SVD based principal vector chain constructions, Gram-Schmidt orthogonalization techniques, the Arnoldi method, or the Schur Normal Form of $A$ in their formulations. They are helped by Datta's 1973 method that symmetrizes unreduced Hessenberg matrices directly.
The eigendata based methods work well and quickly for generic matrices $A$ and create well conditioned matrix symmetrizers through eigenvector dyad accumulation. But all of the eigen based methods have differing deficiencies with matrices $A$ that have ill-conditioned or complicated eigen structures with nontrivial Jordan normal forms. Our symmetrizer studies for matrices with ill-conditioned eigensystems lead to two open problems of matrix optimization.

**Keywords:** symmetric matrix factorization, symmetrizer, symmetrizer computation, eigenvalue method, linear equation, principal subspace computation, matrix optimization, numerical algorithm, MATLAB code

**AMS :** 65F10, 65J10, 15A23, 15B57

## 1 Introduction

105 years ago, F. G. Frobenius [11, p. 421] apparently was the first to discover that every square matrix $A_{n,n}$ over a field $\mathbb{F}$ can be written as the product of two symmetric matrices $A = S_1 S_2$, that one of the factors $S_i \in \mathbb{F}^{n,n}$ can always be chosen nonsingular, that there always exist at least $n$ linear independent symmetric matrices $S = S^T$ that symmetrize $A$ so that $SA = (SA)^T = A^T S^T = A^T S$ is symmetric, and that if $Y_1$ and $Y_2$ are two symmetric symmetrizers of $A_{n,n}$ then so are all linear combinations of $Y_1$ and $Y_2$. This symmetric matrix factorization of $A = S^{-1} \cdot (S \cdot A)$ for a nonsingular symmetrizer $S = S^T$ of $A$ then apparently lay dormant for a long time.

Matrix factorizations such as LR, QR, the polar factorization, the sign factorization or the SVD have become immensely important in modern numerical computations, in part thanks to Alston Householder's promotion of a "decompositional approach to matrix computations" that was included in Jack Dongarra and Francis Sullivan's list [9], see also [5], as one of the "Top 10 Algorithms of the 20th Century". Thus there has been a long desire to compute symmetric factorizations $A = S_1 S_2$ of square matrices as well. Frobenius' matrix symmetrization theorem is often called Taussky's Theorem in the engineering literature where it occurs in systems theory such as in modal analysis of non-conservative systems [4] and in asymmetric linear dynamical systems [1], as well as in control theory [38]. [38] also contains an exact matrix symmetrizer algorithm that can be used for low dimensional matrices $A$. Extensions of the symmetric matrix factorization to Hilbert spaces and self-adjoint operator factorizations for boundary value PDEs are treated in [17]. The symmetrizer equation appears naturally in Hamiltonian dynamics, see [42, p. 272 bottom] e.g, and it is essential for finding compatible Poisson brackets in the theory and applications of bi-Hamiltonian systems and Hamiltonian flows. In the theoretic realm, many relations between

*F. Dopico was partially supported by the *Ministerio de Economía y Competitividad* of Spain through the research grant MTM2012-32542.

†Departamento de Matemáticas, Universidad Carlos III de Madrid, Avda. Universidad 30, 28911 Leganés, Spain (dopico@math.uc3m.es)

‡Department of Mathematics and Statistics, Auburn University, Auburn, AL 36849-5310 (uhligfd@auburn.edu)

symmetrizers $S$ and symmetrized matrices $A$ are known, see the introduction of [29] and its bibliography, as well as Taussky [26]. But all previous attempts to compute matrix symmetrizers numerically in the 1960s and early 1970s eventually proved unstable and were abandoned until the iterative method of [29, 30] succeeded. For further details of some of the computational history of this problem, again see [29].
The simplest idea to establish Frobenius' result over $\mathbb{C}$ uses the Jordan normal form $J_{n,n} = \mathrm{diag}(J_1, ..., J_k) = X^{-1}AX$ of a given square matrix $A$. Each Jordan block $J_i$ is symmetrized by the counterdiagonal unit matrix

$$E_i = \begin{pmatrix} 0 & \dots & \dots & 0 & 1 \\ \vdots & & \iddots & 1 & 0 \\ \vdots & \iddots & \iddots & \iddots & \vdots \\ 0 & 1 & \iddots & & \vdots \\ 1 & 0 & \dots & \dots & 0 \end{pmatrix}$$

of conformal size. By accumulating the $E_i$ in a block diagonal matrix $E_{n,n} = \mathrm{diag}(E_1, ..., E_k)$ we see that $E = E^T$ with $EJ = J^T E^T = J^T E$. Consequently $(X^{-T}EX^{-1})^T = X^{-T}EX^{-1}$ and with $A = XJX^{-1}$ we obtain

$$\begin{aligned} (X^{-T}EX^{-1})A &= X^{-T}EX^{-1}XJX^{-1} = X^{-T}EJX^{-1} = \\ &= X^{-T}J^TE^TX^{-1} = X^{-T}X^TA^TX^{-T}E^TX^{-1} = A^T(X^{-T}E^TX^{-1}) \,, \end{aligned} \qquad (I)$$

i.e., $X^{-T}EX^{-1}$ is a nonsingular symmetrizer of $A$. We note for computational purposes that the condition of this symmetrizer of $A$ hinges on, but is generally not equal to, the square of the condition number for $X$ in $(I)$.

How can the classical Jordan normal form from the 19th century (or any other matrix normal form) be used today to compute matrix symmetrizers for square matrices?
The early unsuccessful attempts from before and around 1970 by Howland and his students [15], [8], [6], [7] and by Trapp [28] started out by finding symmetrizers for matrix normal forms and lateron replaced the given matrix $A$ by a similar Hessenberg matrix $H$ and tried to compute $A$'s symmetrizers in terms of symmetrizers for $H$, followed by the obvious congruences à la $(I)$ to get back to $A$. The idea to work on Hessenberg symmetrizers instead of on $A$ itself was seemingly inspired by the role of Hessenberg matrices in Francis' and Kublanovskaya's QR type eigenvalue algorithms of 1960/1961. QR can find the whole ensemble of eigenvalues of any square matrix in a backward stable manner for the first time and then the eigenvectors can be computed via additional methods in such a way that each particular computed eigenpair is also stably computed[1]. The QR eigenvalue algorithm has become one backbone of much of our modern computations and technologies, see [2, 5, 9]. The success of QR for generic, i.e., diagonalizable matrices then brought the more general defective matrix normal form problem to the fore: how can one extract principal vectors for defective matrices and their principal vector subspaces in a stable fashion? Given the eigenvalues of $A$, a number of stable algorithms have been developed by Kublanovskaya [20], Ruhe [21], Wilkinson [41], Golub and Wilkinson [13], Kågström and Ruhe [18] and others. A common feature of these algorithms has to be emphasized: all of them assume that repeated or clustered eigenvalues have been reliably identified beforehand. Such an identification, however, is only possible in particular situations. But, in general, it has never been achieved numerically and may not even be possible.

This is the background of our paper and current work. The first three new eigendata based algorithms, numbered (3), (4), and (5) start with what Frobenius [11, p. 421 bottom] had discovered when studying matrix commutators, i.e., essentially formula $(I)$ above. First we try to find and resolve clusters of repeated eigenvalues as they are computed in Francis' algorithm on a circle rather than at its center, see [13, sect 3] e.g. Then we compute

[1] Once the whole set of eigenvalues of a matrix $A$ has been computed by QR, each eigenvector $\hat{v}_i$ associated with each eigenvalue $\hat{\lambda}_i$ can be computed either with inverse iteration on a Hessenberg matrix or by solving a (quasi) triangular system associated with the Schur form of $A$, followed in both cases by proper matrix multiplications [2, p. 42]. The Schur form procedure is the one used by default by LAPACK when all the eigenvectors are desired. It can be proved that the whole ensemble of computed eigenvalues are the exact eigenvalues of a nearby matrix $A + E$, with $\|E\| = O(\mathbf{u})\|A\|$ and $\mathbf{u}$ the unit roundoff of the computer. However, for eigenvectors one can only prove (for both methods mentioned above) that each computed eigenpair $(\hat{\lambda}_i, \hat{v}_i)$ is the exact eigenpair of $A + E_i$, with $\|E_i\| = O(\mathbf{u})\|A\|$, but with differing $E_i$ for each eigenpair $(\hat{\lambda}_i, \hat{v}_i)$, see [2, p. 107].

the principal vectors of $A$; via the SVD in method (3) for example as detailed by Wilkinson [41] and Golub and Wilkinson [13] and by other methods in (4) and (5).
Our most straight forward eigen based algorithm (3), coded in `rightsymmAfulljordan8l.m` [32] uses formula $(I)$ directly: the eigenvectors are computed from QR and the principal vector chains for defective eigenvalues are provided by the SVD following Golub and Wilkinson [13] via the standard recurrence formula for Jordan chains from maximal principal vector grade on down.
Algorithm (4), coded in `rightsymmAGSplusLinEqul.m` [33] computes ONBs (orthonormal bases) via QR factorizations for each principal subspace of $A$ and represents the linear transform $A$ on these subspaces by dense lower dimensional matrices. The latter are then symmetrized by the linear equations method (2) `Symmlinequatl.m` [31], to be described below, inside algorithm (4).
Our most sophisticated eigendata algorithm (5), coded in `rightsymmAjordanArnoldi1l.m` [34], starts from principal vectors of maximal grade for each repeated eigenvalues of $A$ and uses Arnoldi's method [3] to find an orthonormal basis for the associated principal vector subspace. It then constructs a partial symmetrizer from the Arnoldi Hessenberg matrix representation of $A$ as restricted to this principal vector subspace. To do so, we use its computed ONB and Datta's Hessenberg symmetrizer method [6], and then apply formula $(I)$ appropriately.

We compare the new eigendata based symmetrizer algorithms with method (1), which is the iterative method `symmorthlongv.m` [30] from [29] that is based on Jianguo Huang and Liwei Nong's two stage algorithm [16] for solving general linear equations of the form $T(x) = f$.

We also consider a new method (2), coded in `Symmlinequatl.m` [31] to solve the linear symmetrizer equation $SA = A^T S$ for the unknown entries $s_{i,j}$ of $S$ directly and use partial sparsity methods for large dimensions $n$.

Finally we describe and test a new hybrid method (6), coded in `rightsymmSchurplusLinEqu2l.m` [35], for the matrix symmetrizer problem that uses the Schur Normal Form of $A$ to find invariant subspace bases for $A$ and the linear equations method (2) to symmetrize $A$'s restrictions to those subspaces, again combined with a transformation of type $(I)$.

Section 2 explains the computational matrix symmetrizer problem further, followed by Section 3 that describes our new algorithms in more detail. This is followed by numerical tests and comparisons for various test matrices of sizes n = 1 through n = 1,000 in Section 4. MATLAB codes of all six linear, eigen- and principal vector, and Schur based symmetrizer algorithms are available at [36]. In the last section, Section 5, we discuss a simple example to illustrate the difficulties with computing nonsingular well-conditioned symmetrizers. We pose two related non-convex matrix optimization problems whose solution would lead to improved methods for finding matrix symmetrizers in all cases quickly via eigen based methods.

# 2 The Matrix Symmetrizer Problem

Historically, a symmetric matrix $S$ has been called a symmetrizer of $A \in \mathbb{F}^{n,n}$ if $SA$ is symmetric. Note that here the symmetrizer $S$ operates from the left on $A$. Eigenvalues are generally defined by right hand side matrix-vector multiplication formulas as in $Ax = \lambda x$. This leads to two interrelated types of symmetrizers.

**Definition :**
A symmetric matrix $S$ for which $SA$ is symmetric is called a *left-side symmetrizer* of $A \in \mathbb{F}^{n,n}$.
A symmetric matrix $S$ for which $AS$ is symmetric is called a *right-side symmetrizer* of $A \in \mathbb{F}^{n,n}$.

Some of the facts that now follow were a part of E. Desaultes master's thesis [8] in 1968 which we repeat for completeness here.

**Proposition 1 :**
If $S$ is a left-side symmetrizer of $A$, then $S$ is a right-side symmetrizer of $A^T$ and vice versa: if $S$ is a right-side symmetrizer of A, then $S$ is a left-side symmetrizer of $A^T$.

The proof follows directly from $SA = A^T S$.

The transposition between left-sided symmetrizers and right-sided ones allows us to relate the matrix symmetrizer problem to the eigenvalue problem as follows.

**Theorem 1 :**

If $A \in \mathbb{F}^{n,n}$ is diagonalizable for an eigenvector basis collected in the columns of $V \in \mathbb{F}^{n,n}$, then $VV^T$ is a nonsingular right symmetrizer of $A$.

**Proof :** If we assume that $AV = VD$ for the diagonal eigenvalue matrix $D$ then $AV \cdot V^T = VDV^T$ is symmetric, as is the nonsingular matrix $VV^T$. □

For our purposes we call any matrix of the form

$$S = VFV^T,$$

a *naive right symmetrizer* of $A_{n,n}$, when $V$ is an $n \times n$ eigenvector matrix of $A_{n,n}$ and $F$ is any diagonal $n \times n$ matrix. The fact that $VFV^T$ is a right symmetrizer of $A$ follows again from $AV = VD$ with $D$ diagonal, since this implies that $A(VFV^T) = VDFV^T$ is symmetric. For a diagonalizable matrix with non-repeated eigenvalues, it is easy to see that all its right symmetrizers are of the form $VFV^T$ for arbitrary diagonal matrices $F$. However, to find a symmetric factorization of $A$ from $A \cdot S = S_1$ with $S = S^T$ and $S_1 = S_1^T$ as $A = S_1 S^{-1}$ we need to compute right symmetrizers $S$ of $A$ with reasonably small condition numbers. It is not clear how to choose a diagonal matrix $F$ so that $VFV^T$ is well conditioned especially when the nonsingular eigenvector matrix $V$ of $A$ is ill conditioned; see the Open Problem # 1 at the end of Section 5. In this context, we emphasize that $\text{cond}_2(VFV^T) \leq \left(\text{cond}_2(VF^{1/2})\right)^2$, where $\text{cond}_2$ stands for the spectral condition number of a matrix, and, in fact, both numbers can be very different. The equality $\text{cond}_2(VFV^T) = \left(\text{cond}_2(VF^{1/2})\right)^2$ can only be guaranteed if $V$ is real and $F$ is real, positive and diagonal. If in this case one chooses $F$ so that the 2-norms of the columns of $VF^{1/2}$ become equal, then $\text{cond}_2(VFV^T)$ is minimal for such $F$ up to a factor of $n$. This is a consequence of a classical result by van der Sluis [37]. However, to consider only positive diagonal matrices $F$ may dramatically limit our chance of finding well conditioned symmetrizers $VFV^T$. See our discussions in Example 1 of this section and also the example and the symmetrizer in formula (10) of Section 5.

As Frobenius [11] has noted, matrix symmetrizers always exist over the same base field as that of the given matrix $A$. Real symmetrizers of a real matrix $A \in \mathbb{R}^{n,n}$ with complex eigenvalues can be found from eigendata as follows. If $A$ is diagonalizable, then the two eigenvectors corresponding to a pair of complex conjugate eigenvalues of $A$ can be chosen as complex conjugates of each other. With this choice, the naive symmetrizer $VV^T$ built from such an eigenvector basis $V$ of $A$ as in Theorem 1 is real. For general real matrices $A$, the next proposition alleys this conundrum further.

**Proposition 2 :**

If $S = R + iU = S^T \in \mathbb{C}^{n,n}$ with $R, U \in \mathbb{R}^{n,n}$ is a right symmetrizer of $A \in \mathbb{R}^{n,n}$, then both $R$ and $U$ are real right symmetrizers of $A$.

**Proof :** If $S = S^T \in \mathbb{C}^{n,n}$ then clearly $R = R^T$ and $U = U^T \in \mathbb{R}^{n,n}$. And $AS = AR + iAU = SA^T = RA^T + iUA^T$ implies $AR = RA^T$ and $AU = UA^T$. □

Theorem 2 below provides a rounding error analysis for computing naive symmetrizers $VFV^T$ with $F$ diagonal from an eigenvector basis matrix $V$ for $A$. Our results are stated for the spectral or 2-norm, but the Frobenius norm can also be used for an equivalent result, see [14, Chapter 6]. Since the right symmetrizer problem is equivalent to finding symmetric solutions of the particular Sylvester equation $AS - SA^T = 0$, we perform the error analysis in terms of the relative residual as usually done in error analyses of algorithms for Sylvester equations [14, section 16.1]. At this point we emphasize that *residual error bounds for Sylvester equations cannot be translated directly into backward errors for the matrix coefficients*, i.e., back to $A$ in our case. This is well known and it is well explained in [14, section 16.2]. Therefore tiny relative residual errors are the best to be expected when computing symmetrizers. Theorem 2 provides such error bounds for naive symmetrizers computed from eigenvector bases that were computed by LAPACK or MATLAB. Since these software libraries provide *eigenvectors with 2-norms equal to* 1 we use unit norm eigenvectors in the proof of Theorem 2. Part (b) of Theorem 2 shows that the relative residuals of any naive symmetrizer corresponding to real eigenvectors and positive diagonal matrices $F$ always are

of order of the unit roundoff of the computer. From the point of view of rounding error analysis this is the best that can be expected for a residual error. Since complex matrices (and many real matrices) have complex eigenvector matrices $V$, Theorem 2(b) does not cover all possible situations. In addition, one might want to look beyond real positive diagonal matrices $F$ that optimize the condition of $VFV^T$ even when $V$ is real. Therefore we will consider general diagonal matrices $F \in \mathbb{C}^{n,n}$ and general $V \in \mathbb{C}^{n,n}$ in part (a) of Theorem 2. However, in this more general case, a penalty factor creeps into the residual error estimates. This factor is quite natural, it should be expected and it is the best in the following sense: it measures the cancellation rounding errors incurred when performing the matrix multiplications of $VFV^T$. Recall that $\|V\|_F = \sqrt{n}$. Thus, this extra factor measures the sensitivity of $VFV^T$ to tiny perturbations in $V$ and in $F$. Although Theorem 2 shows the power of reliable eigenvalue eigenvector algorithms for computing symmetrizers from the rounding errors point of view, these methods do not and cannot guarantee any other desirable property for naively computed symmetrizers such as invertibility, full rank or moderate condition number. Thus further efforts are needed to obtain matrix symmetrizers with additional necessary or desirable properties through eigen-based algorithms. This will be further exemplified in Example 1 that follows Theorem 2, as well as in our numerical tests in Section 4 and it is the main topic of Section 5.

**Theorem 2 :**

Let $A$ be an $n \times n$ real or complex matrix. Let $\hat{V}$ be a matrix whose columns accumulate the eigenvectors of $A$ computed by LAPACK or MATLAB from the eigenvalues provided by Francis' QR algorithm in a computer with unit roundoff $\mathbf{u}$. Let $F = \mathrm{diag}(f_1, \ldots, f_n)$ be any $n \times n$ diagonal matrix. Then the following statements hold up to first order in $\mathbf{u}$.

(a) If $\hat{Y}$ is the computed version of $\hat{V}F\hat{V}^T$, then there exists a polynomial $q(n)$ of low degree such that

$$\frac{\|A\hat{Y} - \hat{Y}A^T\|_2}{\|A\|_2\|\hat{Y}\|_2} \;\leq\; q(n)\,\mathbf{u}\,\frac{\sum_{i=1}^n |f_i|}{\|\hat{Y}\|_2}\,.$$

(b) In addition, if $\hat{V}$ is a real matrix and $F$ is a real and positive diagonal matrix, then

$$\frac{\|A\hat{Y} - \hat{Y}A^T\|_2}{\|A\|_2\|\hat{Y}\|_2} \leq\; n\,q(n)\,\mathbf{u}.$$

**Proof :** The proof involves standard error analysis. Therefore, we only sketch the main steps. Second order terms in $\mathbf{u}$ are discarded in our analysis. Let $\hat{v}_1, \ldots, \hat{v}_n$ be the columns of $\hat{V}$ and $\hat{\lambda}_1, \ldots, \hat{\lambda}_n$ be the corresponding eigenvalues as computed. Our starting point, see [2, p. 107], is that each eigenpair $(\hat{v}_i, \lambda_i)$ is stably computed with the residual

$$\|A\,\hat{v}_i - \hat{\lambda}_i\,\hat{v}_i\|_2 \leq p(n)\,\mathbf{u}\,\|A\|_2 \qquad \text{for each } \; i = 1, \ldots, n\,. \tag{1}$$

Here we use the 2-norm, both for vectors and matrices and $p(n)$ is a low degree polynomial in $n$. Let us denote $h_i := A\,\hat{v}_i - \hat{\lambda}_i\,\hat{v}_i$. The exact matrix $Y = \hat{V}F\hat{V}^T$ can be expressed dyadically as $Y = \sum_{i=1}^n f_i\,\hat{v}_i\,\hat{v}_i^T$. Therefore

$$AY = \sum_{i=1}^n f_i\,(A\hat{v}_i)\,\hat{v}_i^T = \sum_{i=1}^n f_i\,(\hat{\lambda}_i\,\hat{v}_i + h_i)\,\hat{v}_i^T = \sum_{i=1}^n f_i\,\hat{\lambda}_i\,\hat{v}_i\,\hat{v}_i^T + \sum_{i=1}^n f_i\,h_i\,\hat{v}_i^T \tag{2}$$

and

$$YA^T = \sum_{i=1}^n f_i\,\hat{\lambda}_i\,\hat{v}_i\,\hat{v}_i^T + \sum_{i=1}^n f_i\,\hat{v}_i\,h_i^T\,. \tag{3}$$

Combining (2) and (3), we obtain

$$AY - YA^T = \sum_{i=1}^n f_i\,(\,h_i\,\hat{v}_i^T - \hat{v}_i\,h_i^T\,). \tag{4}$$

Next, we can bound the spectral or 2-norm of the matrix difference in (4) to the first order in $\mathbf{u}$ by taking into account (1) and that both LAPACK and MATLAB compute unit eigenvectors, i.e., $\|\hat{v}_i\|_2 = 1 + O(\mathbf{u})$. Thus

$$\|AY - YA^T\|_2 \leq 2 \sum_{i=1}^{n} |f_i| \, \|h_i\|_2 \, \|\hat{v}_i\|_2 \leq 2\, p(n)\, \mathbf{u}\, \|A\|_2 \sum_{i=1}^{n} |f_i| \,. \tag{5}$$

We note that the standard roundoff error bounds of matrix multiplication, see e.g. [14, p. 71], imply

$$\hat{Y} = Y + \tilde{E}, \tag{6}$$

with

$$\|\tilde{E}\|_2 \leq \|\tilde{E}\|_F \leq \tilde{p}(n)\, \mathbf{u} \|\hat{V}F\|_F \|\hat{V}^T\|_F \leq n\, \tilde{p}(n)\, \mathbf{u} \sqrt{|f_1|^2 + \cdots + |f_n|^2} \leq n\, \tilde{p}(n)\, \mathbf{u} \sum_{i=1}^{n} |f_i| \,. \tag{7}$$

Here $\tilde{p}(n)$ is a first degree polynomial in $n$ and we have used that $\|\hat{V}\|_F = \sqrt{n} + O(\mathbf{u})$. Therefore

$$A\hat{Y} - \hat{Y}A^T = AY - YA^T + A\tilde{E} - \tilde{E}A^T,$$

and

$$\begin{aligned} \|A\hat{Y} - \hat{Y}A^T\|_2 &\leq \|AY - YA^T\|_2 + \|A\tilde{E} - \tilde{E}A^T\|_2 \\ &\leq \|AY - YA^T\|_2 + 2\|A\|_2\|\tilde{E}\|_2 \\ &\leq 2\,(p(n) + n\,\tilde{p}(n))\, \mathbf{u}\, \|A\|_2 \sum_{i=1}^{n} |f_i| \end{aligned}$$

from (5), (6), and (7). This proves (a) with $q(n) := 2\,(p(n) + n\,\tilde{p}(n))$.
To prove part (b), note that for real matrices $\hat{V}$ and positive diagonal matrices $F$ equation (6) yields

$$\|\hat{Y}\|_2 = \|Y\|_2 + O(\mathbf{u}) = \|\hat{V}F^{1/2}\|_2^2 + O(\mathbf{u}) \geq \|\hat{V}F^{1/2}\|_F^2 \,/\, n + O(\mathbf{u}) = (\sum_{i=1}^{n} f_i)/n + O(\mathbf{u}) \,,$$

where the last equality follows from the fact that the $i$-th column of $\hat{V}F^{1/2}$ has norm equal to $f_i^{1/2} + O(\mathbf{u})$. The result follows from combining this lower bound with the bound in part (a). □

Even if a computed eigenvector matrix for $A$ is real, Example 1 below will illustrate why considering diagonal matrices $F$ with positive as well as negative and possibly different size entries may be advantageous for computing naive symmetrizers $\hat{Y} = \hat{V}F\hat{V}^T$ with desirable properties.

**Example 1:**

Let us consider that an eigenvector matrix $\hat{V}$ with *normalized* eigenvectors has been computed from

$$V = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ -1 & 1 & 1+\delta \end{bmatrix},$$

where $\delta = 10^{-3}$. In the next table, we display the condition numbers, the norms, and the penalty factors $\sum_{i=1}^{3} |f_i| / \|\hat{Y}\|_2$ that appear in Theorem 2(a) for the symmetrizers $\hat{Y} = \hat{V}F\hat{V}^T$ computed with $\hat{V}$ for different choices of $F$. All computations are performed using MATLAB.

| $F$ | $\text{cond}_2(\hat{Y})$ | $\|\hat{Y}\|_2$ | $\sum_{i=1}^{3} \|f_i\|/\|\hat{Y}\|_2$ |
|---|---|---|---|
| $\text{diag}(1,1,1)$ | $7.2048e+07$ | 2.0000 | 1.5000 |
| $\text{diag}(1,1,-1)$ | $4.2441e+03$ | 1.0000 | 3.0000 |
| $\text{diag}(1,1000,-1000)$ | 7.3407 | 1.1509 | $1.7386e+03$ |

The symmetrizer $\hat{Y}_1$ in the first row of our table uses the most simple choice $F = I_3$. Therefore its condition number is $(\text{cond}_2(\hat{V}))^2$ and this is very large since the last two columns of $\hat{V}$ are almost parallel. However, its residual error bound is nearly perfect, i.e., of order unit roundoff, by Theorem 2(b). The symmetrizer $\hat{Y}_2$ in the second row corresponding to $F = \text{diag}(1,1,-1)$ with negative entries has a much smaller condition number than $\hat{Y}_1$ and likewise a near perfect residual error bound, again in fact and according to Theorem 2(a). Therefore, $\hat{Y}_2$ should be preferred to $\hat{Y}_1$. Finally, the symmetrizer $\hat{Y}_3$ constructed from $F = \text{diag}(1,1000,-1000)$ in the bottom row of our table is extremely well conditioned, but it looses three digits in the residual error bound. Thus it is unclear whether $\hat{Y}_3$ should be preferred over $\hat{Y}_2$.

This example shows that a near optimal selection of the entries of $F$ plays the key role for computing well-conditioned symmetrizers with tiny residuals. However, this is an open problem, to design methods to find adequate diagonal matrices $F$. It is not even clear, in general, whether the condition number can be improved without worsening the residual error bound significantly, as it happens in the symmetrizer $\hat{Y}_2$ in Example 1. More on these questions are in Section 5, where we solve a simple example algebraically and the solution indicates that to find "good" matrices $F$ may be very difficult.

After the diagonalizable matrix case discussions in Theorem 2 and Example 1, we now proceed to the non-diagonalizable defective matrix case for $A$ with repeated eigenvalues and principal vectors of grades larger than one. Such an $A$ has the Jordan normal form $J$ with $AX = XJ$ for $J = \text{diag}(J_1, ..., J_k)$ that is composed of Jordan blocks $J_i$ on its diagonal for a nonsingular matrix $X$ with principal vector chains and eigenvectors of $A$ in its columns. As $J$ allows a ready right (and left) side symmetrizer $E = \text{diag}(E_i)$ as explained earlier, $X \cdot E \cdot X^T$ will symmetrize $A$ from the right since $AX = XJ$ implies that $A(XEX^T) = XJEX^T$ is symmetric. And thus $XEX^T$ is a nonsingular right-side symmetrizer of $A$, at least in theory.
However, a naive and straightforward implementation of this approach often leads to disaster. For truly defective matrices $A$ the symmetrizers $XEX^T$ are generally computed with high rank deficiencies and even if they happen to have full rank, their condition numbers will often exceed those of the symmetrizers computed with Uhlig's iterative method [29] by many orders of magnitude, with multiplication factors in the $10^6$ and much higher ranges for defective $A$.
What goes wrong? Loosely speaking, the condition number of a symmetrizer of the form $XEX^T$ above is mostly affected by the condition of $X$, squared[2] according to [24], as the condition of the interior factor $E = \text{diag}(E_i)$ equals 1. A simple visual inspection of principal vector chains computed from a top grade $k$ principal vector $p_k$ on down for $A$ and the respective eigenvalue $\lambda$ via the classical recursion $p_{j-1} = (A - \lambda I)p_j$ reveals that such principal vector chains become quickly near stationary, see also [13, sect 12]. This can make $X$ unacceptably ill conditioned which in turn affects the conditioning of such a naively computed right-side symmetrizer $XEX^T$ of $A$. Rank deficiencies of symmetrizers computed via eigendata methods happen regularly in case $A$ is defective.

# 3 The Six Symmetrizer Algorithms

The methods (1) and (2) compute matrix symmetrizers by linear means, such as through repeated matrix multiplications or via solving a linear system of related equations.

Methods (3) through (5) are instead based on eigenanalyses and the Jordan normal form of the given $A_{n,n}$ whose symmetrizers we seek to compute. In each of these eigendata based algorithms we start off by performing the same first few steps as follows:

[2] We emphasize that this statement should be understood as a "generic statement", since $\text{cond}_2(XEX^T) \leq (\text{cond}_2(X))^2$ and both numbers can be very different, as can be easily checked via random tests in MATLAB. Note also that even $\text{cond}_2(XX^T) \ll (\text{cond}_2(X))^2$ may happen for complex (nonreal) matrices $X$.

*1st step :* For the given $A_{n,n}$, compute the eigenvalues $\lambda$ via Francis QR and then the eigenvectors (accumulated in the columns of $V$) by either solving a (quasi) triangular systems or by Hessenberg inverse iteration, followed in both cases by appropriate matrix multiplications.

*2nd step :* Form $Y = V * V^T$ with $V_{n,n}$ from Step 1 and check its properties. If rank$(Y) < n$ we check for clusters of eigenvalues. Here we crudely judge eigenvalues to be repeated when they are relatively less than between .01 and .1 units apart. This threshold is adjustable inside the codes. We use the average of each 'repeated' set of eigenvalues as one eigenvalue of $A$ and assign it the cumulative multiplicity. Then we collect all $j < n$ eigenvectors for the non-repeated eigenvalues of $A$ in a new $V_{n,j}$ and replace $Y$ by $V_{n,j} * V_{n,j}^T$.

*3rd step :* If there are repeated eigenvalues and rank$(Y) < n$, then for each repeated eigenvalue $\lambda$ :

> We find the Jordan structure for $\lambda$ and $A$ by computing a complete set of its principal vectors and collect them in the columns of the matrix $P_V$, ordered by decreasing grade. This is based on the SVD method of Wilkinson [41, method 1] and implemented according to the extended version by Wilkinson and Golub in section 9 of [13, p. 594]. It uses the SVD of $B = A - \lambda I$ and iterates upon the principal vector grades.

Our three eigen based symmetrizer algorithms (3), (4), and (5) differ only by how they use the Jordan structure information of $A$ obtained in step 3. For details in implementation and output, see below. Observe that according to [13] the principal vectors provided by the 3rd step above do not form Jordan chains and this is the reason why further development is necessary. The joint complexity of the first two steps above is $O(n^3)$ flops. The complexity of the 3rd step is also "almost always" $O(n^3)$ flops, but it may grow up to $O(n^4)$ flops in the rare cases when $O(n)$ clusters of eigenvalues have been identified for $A$ in the 2nd step or when Jordan blocks of size $O(n)$ occur for $A$. The cost of the remaining steps of each of the eigen based algorithms (3) and (5) is $O(n^3)$ flops, so each of them has "almost always" global complexity $O(n^3)$ and very rarely $O(n^4)$. For the eigen based algorithm (4), the remaining steps may cost up to $O(n^4)$ flops, since it uses the linear symmetrizer method (2). However, again, this only happens in the rare case when $A$ has Jordan blocks of size $O(n)$ and, therefore, we can also state that the global cost of algorithm (4) is "almost always" $O(n^3)$ flops and very rarely $O(n^4)$.

Method (6) starts off with performing the 1st and 2nd steps above. But instead of the 3rd step, it then uses the Schur Normal Form of $A$, already computed by Francis QR, and computes invariant subspace bases for each identified cluster of eigenvalues of $A$. In method (6) - differing from methods (3), (4), and (5) - multiple Jordan chain spaces for the same eigenvalue of $A$ are combined and treated as one invariant subspace. Then $A$'s representation on each distinct invariant subspace is computed and symmetrized by the direct linear symmetrizer algorithm (2). In this way we find symmetrizers of $A$ in a hybrid eigen-linear based fashion according to a formula similar to $(I)$. The joint cost of steps 1, 2, and the computation of the bases of invariant subspaces via the Schur Normal Form takes $O(n^3)$ flops, but the use of algorithm (2) may increase the total cost of method (6) very rarely to $O(n^4)$ flops.

## 3.1 Method (1), a Linear two Stage Iterative Scheme

Method (1) is based on [16]. It uses an iterative linear two stage method as detailed by [29]. Its MATLAB code is stored at [30]. The method uses two matrix multiplies and six matrix norm computations per iteration step. Unfortunately it is slow to converge. In theory it must converge in at most $n^2$ steps for any $n$ by $n$ matrix $A$ due to the proven orthogonality of both of its code-internal iterates. In practice, however, orthogonality is soon jeopardized and reorthogonalization is often required for swift convergence. In case of fast convergence in $O(n^2)$ iteration steps without reorthogonalization, method (1) has a complexity of $O(n^5)$; with reorthogonalization its complexity rises to $O(n^7)$. However this method always seems to find a well conditioned full rank symmetrizer for any given $A$. And thus - except for its slowness - it is the standard against which all our other symmetrizer algorithms will be measured. Uhlig [29, 30] gives a full description and contains many tests of its qualities.

## 3.2 Method (2), the direct Linear Equations Approach to find S

Method (2) was developed to allow us to follow one referee's suggestion to try Gram-Schmidt type orthogonalizations to find orthogonal principal vector subspace bases of $A_{n,n}$ instead of using Arnoldi's method as in algorithm (5) below. Here we specifically acknowledge the help of Luke Oeding of Auburn University who insisted that a direct method can be coded and programmed to find the unknown entries $s_{i,j}$ of $S = S^T$ with $SA$ symmetric via

a system of $n(n-1)/2$ linear equations in the unknown entries of $S$.

For $S_{n,n} = (s_{i,j}) = S^T$ with $n(n+1)/2$ unknown entries $s_{i,j} = s_{j,i}$, we now consider the matrix symmetrizer problem as a linear equations problem given by $SA = A^T S$. Clearly

$$\begin{aligned}(SA)_{i,j} &= \sum_{k=1}^{n} s_{i,k} a_{k,j} \\ &= \sum_{k=1}^{i} s_{i,k}\, a_{k,j} + \sum_{l=i+1}^{n} s_{l,i}\, a_{l,j}\ ,\end{aligned}$$

and

$$\begin{aligned}(SA)_{j,i} &= \sum_{r=1}^{n} s_{j,r} a_{r,i} \\ &= \sum_{r=1}^{j} s_{j,r} a_{r,i} + \sum_{r=j+1}^{n} s_{r,j} a_{r,i}\end{aligned}$$

where both sum expressions on the respective second line use the symmetry of $S$ to express the $(i,j)$ and $(j,i)$ entries of $SA$ only via the lower triangular entries $s_{i,j}$ with $i \geq j$ of $S$.

Sorting the lower triangular entries of $S$ column wise into one vector

$$s = (s_{1,1}, s_{2,1}, ..., s_{n,1}, s_{2,2}, ..., s_{n,2}, ..., s_{i,j}, ..., s_{n,n})^T \in \mathbb{F}^{n(n+1)/2}$$

of unknowns $s_{i,j}$ for $i \geq j$ we obtain $n(n-1)/2$ linear equations by equating $(SA)_{i,j}$ and $(SA)_{j,i}$ $(= (A^T S)_{i,j})$ for the $n(n+1)/2$ lower triangular entries of $S = S^T$. The diagonal entry equations $(SA)_{i,i} = (A^T S)_{i,i}$ are tautologies and thus redundant. This gives us a seemingly sparse linear system whose zero - nonzero pattern we exemplify below for $n = 10$.

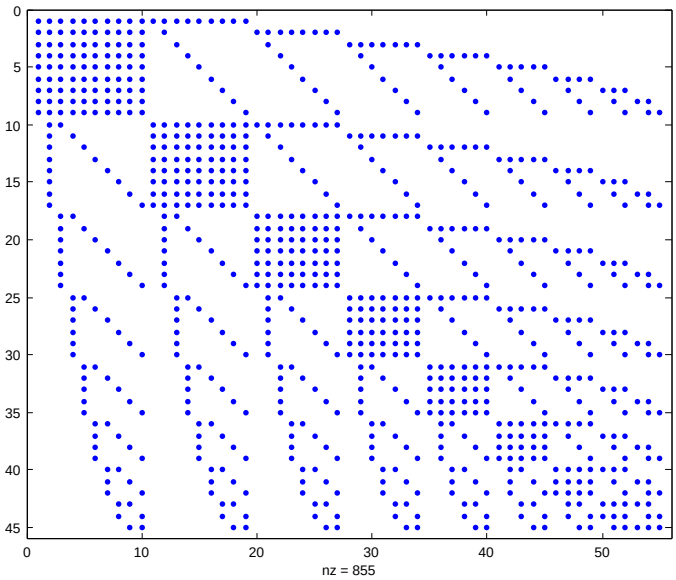


Figure 1 : Symmetrizer linear system's sparsity structure for a random entry matrix $A_{10,10}$ : 55 unknowns and 45 linear equations with 855 nonzero entries

Obviously for a dense $n$ by $n$ matrix $A$, each row of the associated linear system contains at most $2n-1$ nonzero entries that link its $n(n+1)/2$ unknowns and it does so in $n(n-1)/2$ equations. Thus for each $n$ by $n$ matrix $A$ there is an at least $n$ dimensional subspace of symmetrizers, reproving one of Frobenius' further results that there are at least $n$ linearly independent matrix symmetrizers for every square matrix $A$.

In the MATLAB code `Symmlinequatl.m` [31] we use the built-in MATLAB nullspace function `null.m` to solve the homogeneous system $SA - A^TS = O$ for the lower triangular entries of $S$ for sizes of $A$ below the break-even point $n = 54$. For larger $n$ we use its sparse version `nulls.m` as coded by Sivan Toledo [27] in 2005. We observe that a full basis for the symmetrizer space is obtained in either case. Any particular symmetrizer can be constructed by picking one of the matrices in this basis or by computing a random linear combination of all computed symmetrizers in the basis.

Note that to find the entries of $S$ in this way requires us to solve a rectangular system of linear equations of size $n(n-1)/2$ by $n(n+1)/2$. Thus for $n \leq 54$ where we use `null.m` this requires an $O(n^6)$ effort, while using sparse methods for larger $n$ reduces the complexity to $O(n^4)$ under ideal conditions. This is clearly born out in the runtime comparisons of the next section and besides, it limits method (2) to dimensions $n \ll 200$ due to the necessary system matrix generation and their storage.

## 3.3 Method (3), an Eigendata and Jordan Normal Form Algorithm

Method (3) - as all eigendata based methods - has $O(n^3)$ complexity, except in very rare circumstances when its complexity may be $O(n^4)$ as explained earlier. The method is based on the analysis preceding formula $(I)$ in the introduction. If $AX = XJ$ for a Jordan normal form matrix $J = diag(J_i)$ of $A$ and if $X$ is comprised of principal vector chains in its columns, then $S = Xdiag(E_i)X^T$ is a right side symmetrizer of $A$ for counterdiagonal unit matrices $E_i$ that are conformal in size with the Jordan blocks $J_i$. Since in theory $X$ is nonsingular, so is $S$. For constructing the principal vector chains for each repeated eigenvalue $\lambda$ of $A$, method (3) computes the principal vectors of highest grade in the 3rd step and uses the recursion $p_{j-1} = (A - \lambda I)p_j$ to find full principal vector chains as Jordan theory prescribes.
However, in this method nothing can be said about the conditioning of $S$ or its numerical invertibility. When the eigenvalues of $A$ are ill conditioned and the eigenvector or principal vector basis for $A$ leads to an ill conditioned $X$, then $S$ will very often be doubly ill conditioned since $X$ appears twice in the formulation of $S = Xdiag(E_i)X^T$, see Taussky [24] e.g. for the conditioning of matrix products and the earlier footnote at the end of Section 2.
This is one serious drawback for method (3). To remedy, we next try to represent the principal vector subspaces for $A$ differently so that the square of cond$(X)$ does not mar the conditioning of a symmetrizer $S$ of $A$ in methods (4) and (5) below.

## 3.4 Method (4), using Orthonormal Bases for Principal Subspaces and method (2)

To improve on method (3) at the same generic $O(n^3)$ complexity rate, here we first orthogonalize each complete grade $k$ principal vector chain for $A_{n,n} \in \mathbb{R}^{n,n}$ or $\mathbb{C}^{n,n}$ instead and then symmetrize $A$'s matrix representation with respect to the principal vector subspace ONBs as follows.
Assume that a principal vector chain is stored in the columns of $P_{n,k}$. We can compute a partial symmetrizer $S$ of $A$ as follows from $P$. Note that $A_{n,n}P_{n,k} = P_{n,k}J_{k,k}$ with $P$ and a Jordan block $J \in \mathbb{C}^{k,k}$ for an eigenvalue $\lambda$ of $A$. Assume further that $P_{n,k} = Q_{n,k} \cdot R_{k,k}$ is the reduced QR factorization of $P$, whose computation requires $O(nk^2)$ flops with $k \leq n$. With

$$P = \begin{pmatrix} \vdots & \cdots & \vdots \\ p_1 & \cdots & p_k \\ \vdots & \cdots & \vdots \end{pmatrix}_{n,k} \quad \text{and} \quad Q = \begin{pmatrix} \vdots & \cdots & \vdots \\ q_1 & \cdots & q_k \\ \vdots & \cdots & \vdots \end{pmatrix}_{n,k}$$

we have span$\{p_i\}$ = span$\{q_i\}$ and $Q^*Q = I_k$. Let $M_{k,k} = Q^*AQ$ represent the linear transformation induced by $A$ with respect to the ONB $\{q_1, ..., q_k\}$ of the principal vector subspace under consideration. If $L_{k,k} = L^T$ is a right side symmetrizer of $M$, found by applying Proposition 1 to $M^T$ in method (2), for example, then

$$A \cdot Q \cdot L \cdot Q^T = Q \cdot M \cdot L \cdot Q^T \ .$$

Thus $S = QLQ^T$ is a right side symmetrizer for $A$ of rank at most $k$ whose condition number equals that of $L$. To obtain a symmetrizer of possibly full rank for $A$ we accumulate these partial symmetrizers in $Y = Y^T$.

Since cond$(Q) = 1$ for each partial symmetrizer, in this method the possibly ill conditioned principal vector chain matrices $P$ do not appear twice in $Y$ as they did in method (3) and therefore their condition numbers do not affect the conditioning of $Y$ as adversely. The total cost of the additional work of method (4) over the three first steps described at the beginning of Section 3 is $O(n^3)$ flops, assuming that the size of each $M$ is small and, so, the cost of each application of method (2) is bounded by a moderate constant.

Note that method (4) tries to cure the "most inner" ill conditioning in the symmetrizer $Y$, i.e., that coming from the potential ill conditioning of each particular principal vector chain. Method (5) below tries to remedy the same source of ill conditioning differently. However, none of these two methods can remedy an ill conditioning in $Y$ that is caused by two (or more) principal subspaces of $A$ that are almost parallel.

### 3.5 Method (5), using Arnoldi and Datta for $A$'s Principal Subspace Representations

As we have just seen, each Jordan block $J_i$ of size $k$ by $k$ for $A$ represents the underlying linear transformation $A : \mathbb{F}^n \to \mathbb{F}^n$ when restricted to the respective principal vector space. The Jordan block $J_i$ represents $A$ with respect to the principal vector chain basis $p_k, ..., p_1$. Differing from method (4), one can find an orthonormal basis of the principal vector subspace

$$\text{span}\{p_k, ..., p_1\} = \text{span}\{p_k, p_{k-1} = (A - \lambda I)p_k, ..., p_1 = (A - \lambda I)^{k-1}p_k\}$$

for $A$ very simply by viewing it as a Krylov subspace and using Arnoldi's iterative method [3] while operating with $B = A - \lambda I$ on each respective highest grade $k$ principal vector $p_k$ for $k - 1$ iteration steps. This approach gives us an ONB $W_i$ for each principal vector subspace of dimension $k$ together with a $k$ by $k$ Hessenberg matrix representation $H_i$ of $A$ as restricted to the respective principal subspace. Here the Arnoldi process ends after $k - 1$ steps, as we have 'lucky convergence' by design, and the resulting Hessenberg matrix $H_i$ is unreduced having exactly one Jordan block as its Jordan normal form, i.e., $H_i$ is non-decomposing. Thus $AW_i = W_iH_i$ for the $n$ by $k$ matrix $W_i$ whose columns are mutually orthonormal $n$-vectors. Appending all such $W_i$ blocks in $W = [W_1, W_2, ...]_{n,n}$ block-column wise and forming $H_{n,n} = \text{diag}(H_i)$ with each $H_i$ a non-decomposing Hessenberg matrix, we then have

$$AW = WH \; . \tag{8}$$

$W$ contains mutually orthonormal column vectors in each strip of vectors that form an ONB of a particular principal vector subspace of $A$.
A simple algorithm to symmetrize each non-decomposing $H_i$ of size $m$ by $m$ was given by Datta [7]:
Let

$$H = \begin{pmatrix} h_{1,1} & & & & & \\ h_{2,1} & \ddots & & & * & \\ 0 & \ddots & \ddots & & & \\ \vdots & \ddots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \ddots & \\ 0 & \dots & \dots & 0 & h_{m.m-1} & h_{m,m} \end{pmatrix} \text{ and } Z_H = \begin{pmatrix} | & & | & & | \\ z_1 & \dots & z_k & \dots & z_m \\ | & & | & & | \end{pmatrix}$$

both be of size $m$ by $m$ where $H$ is an unreduced upper Hessenberg matrix ($h_{i,i-1} \neq 0$ for $i = 2, ..., m$) and $Z_H = Z_H^T$ is symmetric. Then the symmetrizer equation $HZ_H = Z_HH^T$ has only symmetric solutions [25] and it can be solved from any given last column (or row) $z_m$ of $Z_H$ by comparing the left and right hand side columns in $HZ_H = Z_HH^T$, starting with $m$ and going back. For $m$ this amounts to solving $Hz_m = h_{m,m-1}z_{m-1} + h_{m,m}z_m$ for $z_{m-1}$. And for general $m - 1 \geq k \geq 1$ the $k$th column of $Z_H$ then becomes

$$z_k = \left( Hz_{k+1} - \sum_{j=k+1}^{m} h_{k+1,j}z_j \right) / h_{k+1,k} \; .$$

Note that according to [25] every solution $Y$ of $AY = YA^T$ is symmetric if and only if the square matrix $A$ is nonderogatory, which an unreduced Hessenberg matrix $H_{m,m}$ is by default. This is so because all its eigenvalues have geometric multiplicity 1 as $\text{rank}(H - \lambda I_m) \geq m - 1$ for all $\lambda \in \mathbb{C}$.

The right symmetrizers $Z_{H_i}$ computed via Datta's method for each non-decomposing Hessenberg matrix $H_i$ in the block diagonal matrix $H$ of (8) is accumulated in the block diagonal matrix $S_H = \text{diag}(Z_{H_i})$. From (8), we see that $A(WS_HW^T) = WHS_HW^T$ is symmetric and thus $W \cdot S_H \cdot W^T$ is a nonsingular right-side symmetrizer of $A$.

By combining Arnoldi's and Datta's methods here, the potential ill-conditioning of symmetrizers that are computed via the naive eigen- and principal vector chain based method (3) shifts from the generally ill-conditioned principal vector chains to the individual symmetrizers of the associated Hessenberg matrices $H_i$. In our tests the symmetrizer condition numbers from the Arnoldi plus Datta based algorithm were often rather modest with corresponding relative residuals of order unit roundoff despite the fact that Arnoldi's method is not backward stable and neither is Datta's. Yet this method often gives full-rank symmetrizers with condition numbers similar to the ones of the iterative method [29] when the conditioning of $A$'s eigenvalues is good, and even for some defective matrices whose eigenvalue condition numbers may well exceed $10^{20}$ as long as $A$ has relatively small Jordan blocks of sizes below 10. For matrices $A$ with Jordan block sizes above 12, Datta's method shows signs of instability. But even for $A$ with simpler eigenstructures our new hybrid Arnoldi-Datta based method sometimes fails compared to the iterative method of [29]. Reasons for this are unclear.

Here are further details of our eigenanalysis based symmetrizer algorithm (5).

*The 1st, 2nd and the beginning of step 3* are as detailed at the start of this section.

*Step 3 continued :*

For each highest grade $k$ principal vector in turn, we find an ONB for its respective principal subspace via the Householder based Arnoldi method as described by Y. Saad [22, Algorithm 6.3, p.157], i.e., we compute $A$'s Hessenberg matrix representation $H_{k,k}$ with respect to an ONB $V$. This gives us $AV = VH$ for the matrix $V_{n,k}$ with orthonormal columns. Then we compute a symmetrizer $(Z_H)_{k,k}$ for $H$ by Datta's algorithm [7] from a random last column. We add the symmetric matrix $V \cdot Z_H \cdot V^T$ to the current symmetrizer $Y$ and append the orthonormal principal space basis, found in $V$, to the columns of a matrix called $V_V$.

Then we decrease the principal vector grade levels until there is a larger number of principal vectors in one grade $m$ than in the grade above. In this case we find all principal vectors in $P_V$ of grade $m$ that do not belong to the column space of $V_V$ through a QR decomposition of the augmented matrix $[V_V, P_m]$ where $P_m$ consists of all grade $m$ column vectors originally collected in $P_V$. Taking the vectors in $P_V$ that correspond to sizable non-zero diagonal entries $r_{i,i}$ in $R$ from the QR decomposition gives us the top grade principal vectors of a complete set of precisely $m$ dimensional principal subspaces for $A$ and $\lambda$.

Then we proceed as above with Householder Arnoldi to find an ONB for all precisely $m$ dimensional principal subspaces and the corresponding Hessenberg matrix representations of $A$. We find their respective symmetrizers, again with Datta's method [7], and update the symmetrizer $Y$ as before. We repeat this process for decreasing values of $m$ until the partial symmetrizers for all principal subspaces have been computed and added to $Y$, the desired symmetrizer of $A$.

## 3.6 Method (6), using the Schur Normal Form, invariant subspace bases and method (2)

This algorithm was inspired on section 7.6.2 of Golub and van Loan [12, p.396 - 397] which deals with invariant subspace computations via reordering the eigenvalues in the Schur Normal Form that was computed by Francis' QR algorithm. This procedure was originally introduced by Ruhe [21] and improved by Stewart [23].
Our algorithm uses the MATLAB built in Schur m-functions `schur.m`, which computes the Schur Normal Form via Francis' QR algorithm, and `ordschur.m`, which arranges the eigenvalues in the Schur form in any order prescribed by the user. Our method reads the eigenvalues off the diagonal of the computed triangular Schur form for $A$ and checks for clusters or repeats as all our eigendata based methods (3) through (5) do. Then, for each identified cluster with a repeated eigenvalue $\lambda$, we use `ordschur.m` to rewrite the Schur Normal Form for $A$

as $A = UT_\lambda U^*$ so that the eigenvalue of the cluster appears at the top of the diagonal of $T_\lambda$ with its proper multiplicity $m_\lambda$. An orthonormal subspace basis for the invariant subspace of $A$ that is associated with this cluster then appear in the first $m_\lambda$ columns of $U$, which we store in a matrix $U_t$. From $T_\lambda$ we then select the top left $m_\lambda$ by $m_\lambda$ block $T_t$ and symmetrize this generally much smaller matrix via the direct linear equations method (2) by computing $L = L^T \in \mathbb{C}^{m_\lambda, m_\lambda}$ so that $T_t \cdot L$ is symmetric. Then $AU_t = U_t T_t$ gives us

$$A \cdot (U_t \cdot L \cdot U_t^T) = U_t \cdot T_t \cdot L \cdot U_t^T \ .$$

Thus $U_t \cdot L \cdot U_t^T$ is a partial right side symmetrizer of $A$ of rank $m_\lambda$ if $L$ is nonsingular. The global right symmetrizer of $A$ is obtained by adding all the partial symmetrizers for all clustered eigenvalues to the symmetrizer $Y$ derived from the non-repeated (or non-clustered) eigenvalues that have been computed in the 2nd step, see the beginning of Section 3.
Method (6) differs from the eigen-principal vectors based methods (3) through (5) in using the Schur Normal Form of $A$ to compute bases of invariant subspaces. This allows us to avoid the use of principal subspaces and avoids a detailed Jordan analysis of $A$. The operations count for algorithm (6) is again $O(n^3)$ in the generic case and $O(n^4)$ in exceptional cases, due to the possible expense of the linear equation method (2) to find $L$ in case of large clustered eigenvalue sets for $A$.

# 4 Numerical Results and Comparisons

Here we compare the results of the iterative symmetrizer algorithm (1), the direct linear equations method (2), our three new eigen-principal vectors based algorithms (3), (4), and (5), and the Schur based method (6). Note that several of the algorithms rely on randomization. The iterative code [30] for method (1) starts with a random start-up matrix, method (5) computes each Hessenberg matrix symmetrizer from a random last column, method (2) computes symmetrizers via a random linear combination of the basis it computes for the symmetrizer matrix space and this randomization is inherited by methods (4) and (6). Thus the algorithms (1), (2), and (4) through (6) compute different symmetrizers for the same matrix $A$ in separate calls. To compare all methods we average their output over a number of runs.

First we repeat the tests for the matrices used in [29].
The first test there involved the modification $M_K = K + 3K^T$ for the Kahan matrix $K$ of dimension 35 and our test averages over 5 or 10 runs. Here and in what follows, all named matrices are taken from the gallery set of matrices in MATLAB.

| $M_K$ 35 by 35 | max eig cond no | symm. error average | cond($Y$) average | rank($Y$) average | runtime average | # of runs |
|---|---|---|---|---|---|---|
| method (1) (without orthog) | | 5.7952e-10 | 2.3358e+06 | 35 | 7.7128e-01 | 5 |
| method (2) | | 2.8921e-15 | 3.7900e+04 | 35 | 2.5151e-01 | 10 |
| method (3) | 29 | 3.6121e-15 | 2.8107e+04 | 35 | 4.5725e-03 | 10 |
| method (4) | 29 | 3.6121e-15 | 2.8107e+04 | 35 | 4.5093e-03 | 10 |
| method (5) | 29 | 3.6121e-15 | 2.8107e+04 | 35 | 3.5759e-03 | 10 |
| method (6) | | 2.8789e-15 | 2.8102e+04 | 35 | 5.4104e-03 | 10 |

Clearly $M_K$ has well conditioned eigenvalues and all algorithms perform near equally well. Note also that the eigenvalue-principal vector and Schur based methods (3) through (6) are up to 130 times faster here than the linear methods (1) and (2).

In our tables, the 'symmetrizer error' entries always refer to $\|A*Y - Y*A^T\|_2/\|A*Y\|_2$ where $Y$ is a computed right side symmetrizer of $A$. This is more stringent than the $\|A*Y - Y*A^T\|_2/(\|A\|_2 * \|Y\|_2)$ relative error measure that is more commonly used to check the quality of computed solutions of matrix equations. The cond(Y) and rank(Y) or rank(V) entries in our tables use the respective built-in MATLAB functions (see MATLAB online documentation for more details) and the specified data is the average value for the respective numbers of runs.

Now we repeat the same test for the Kahan matrix $K_{35,35}$ itself, again with averaging over five or ten runs.

| $K$ 35 by 35 | max eig cond no | symm. error average | cond($Y$) average | rank($Y$) average | runtime average | # of runs |
|---|---|---|---|---|---|---|
| method (1) (with orthog) | | 1.0193e-09 | 1.3277e+09 | 35 | 5.0713e+00 | 5 |
| method (2) | | 5.3056e-15 | 1.3516e+10 | 35 | 2.6672e-01 | 10 |
| method (3) | 2.6e+8 | 1.4002e-12 | 9.2693e+15 | 32 | 6.9770e-03 | 10 |
| method (4) | 2.6e+8 | 1.4002e-12 | 9.2693e+15 | 32 | 4.3004e-03 | 10 |
| method (5) | 2.6e+8 | 1.4002e-12 | 9.2693e+15 | 32 | 3.6281e-03 | 10 |
| method (6) | | 1.4002e-12 | 9.2693e+15 | 32 | 3.9125e-03 | 10 |

Here all eigen-principal vectors and Schur based methods (3) through (6) fail to find full rank symmetrizers of $K$. The culprit might be the somewhat elevated eigenvalue condition number of some of the eigenvalues of the Kahan matrix. We have used [30] with re-orthogonalization here since the non orthogonalized iterative method (1) converges very slowly and needs over 40,000 iterations to succeed in this example.

Our next example involves the modified Frank matrix $F$ of size 35 by 35 with its parameter $k$ set to 20.

| $F$ ($k = 20$) 35 by 35 | max eig cond no | symm. error average | cond($Y$) average | rank($Y$) average | runtime average | # of runs |
|---|---|---|---|---|---|---|
| method (1) (with orthog) | | 4.0164e-10 | 2.9527e+09 | 35 | 6.8527e+00 | 5 |
| method (2) | | 1.1927e-15 | 2.1622e+15 | 34 | 2.5491e-01 | 10 |
| method (3) | 4.6e+8 | 2.2213e-10 | 2.7215e+17 | 26 | 7.7943e-03 | 10 |
| method (4) | 4.6e+8 | 2.2213e-10 | 2.7215e+17 | 26 | 7.4105e-03 | 10 |
| method (5) | 4.6e+8 | 2.2213e-10 | 2.7215e+17 | 26 | 6.4398e-03 | 10 |
| method (6) | | 4.2376e-15 | 9.9170e+17 | 20 | 5.9974e-03 | 10 |

In this example all eigen-principal vectors and Schur based symmetrizer methods (3) through (6) again suffer fatally in all categories except for speed. And even the direct linear equations method (2) suffers from rank deficiency.

The last gallery matrix example in [29] involves the Hanowa matrix $H$ of size 36 by 36. It has perfectly conditioned eigenvalues. And all six methods perform very similarly and equally well here. Note, however, that the eigendata and Schur based methods are again much faster, between 30 and 80 times, than the non eigenbased methods (1) and (2).

| $H$ 36 by 36 | max eig cond no | symm. error average | cond($Y$) average | rank($Y$) average | runtime average | # of runs |
|---|---|---|---|---|---|---|
| method (1) (without orthog) | | 3.2518e-10 | 4.6952e+00 | 36 | 8.7459e-02 | 5 |
| method (2) | | 2.0894e-15 | 1.0101e+01 | 36 | 2.2584e-01 | 10 |
| method (3) | 1 | 3.9414e-16 | 1.0000e+00 | 36 | 3.3107e-03 | 10 |
| method (4) | 1 | 3.9414e-16 | 1.0000e+00 | 36 | 2.6154e-03 | 10 |
| method (5) | 1 | 3.9414e-16 | 1.0000e+00 | 36 | 2.1424e-03 | 10 |
| method (6) | | 3.9414e-16 | 1.0000e+00 | 36 | 5.6850e-03 | 10 |

Note that for the first four examples the eigen-principal vector based algorithms (3) through (5) compute exactly the same symmetrizer since they never reach their differing repeated eigenvalues, defective matrix stages for $M_K$, $K$, $F$, or $H$. This will change below.

Next we test our six algorithms on dense matrices with known Jordan structure. For this purpose a block diagonal matrix $A$ is constructed with blocks made up of upper triangular random entry matrices, except for two assigned eigenvalues that alternate down their diagonals. Such an $A$ is also tested when transformed via a random complex unitary similarity to become a dense complex matrix $C_{comd}$ with the same Jordan structure.

Our first dense example matrix $B$ of this construction has size 27 by 27 and Jordan blocks of sizes 4, 4, 3, 3, 1, 1 for the eigenvalue $\pi$, as well as Jordan blocks of sizes 3, 3, 3, 2 for the eigenvalue $e$.

| $B$ 27 by 27 | max eig cond no | symm. error average | cond($Y$) average | rank($Y$) average | runtime average | # of runs |
|---|---|---|---|---|---|---|
| method (1) (with orthog) | | 1.7727e-09 | 6.8907e+08 | 27 | 2.8787e+00 | 5 |
| method (2) | | 5.5284e-15 | 1.0110e+10 | 27 | 5.0232e-02 | 10 |
| method (3) | 1.6e+45 | 4.1865e-16 | 9.3515e+18 | 24 | 5.6763e-03 | 10 |
| method (4) | 1.6e+45 | 1.8857e-11 | 1.1225e+21 | 23.7 | 1.1943e-02 | 10 |
| method (5) | 1.6e+45 | 7.9316e-09 | 4.2267e+10 | 25 | 1.0573e-02 | 10 |
| method (6) | | 9.6874e-15 | 1.4322e+12 | 27 | 1.4155e-02 | 10 |

The next example of this kind involves an upper triangular matrix $(C_{ut})_{23,23}$ with eigenvalues 1 and $-10$ and Jordan blocks of sizes 5, 4 and 3 for $\lambda = 1$, as well as Jordan blocks of sizes 5, 4 and 2 for $\lambda = -10$.

| $C_{ut}$ 23 by 23 | max eig cond no | symm. error average | cond($Y$) average | rank($Y$) average | runtime average | # of runs |
|---|---|---|---|---|---|---|
| method (1) (with orthog) | | 9.1618e-08 | 2.3985e+10 | 23 | 1.5451e+00 | 5 |
| method (2) | | 3.0685e-14 | 1.0515e+14 | 22.9 | 3.2153e-02 | 10 |
| method (3) | 3e+60 | 1.2017e-10 | 2.4372e+19 | 22 | 4.9113e-03 | 10 |
| method (4) | 3e+60 | 2.8395e-07 | 3.1620e+18 | 22.9 | 1.4870e-02 | 10 |
| method (5) | 3e+60 | 8.6875e-13 | 3.3969e+15 | 21 | 1.0437e-02 | 10 |
| method (6) | | 8.6396e-12 | 8.1691e+11 | 23 | 9.8288e-03 | 10 |

This matrix $C_{ut}$ has extremely ill-conditioned eigenvalues and only methods (1) and (6) succeeds in computing always full rank symmetrizers.

Repeating with a complex unitarily similar dense version $C_{comd} \in \mathbb{C}^{23,23}$ of $C_{ut}$, method (3) fails to find full rank symmetrizers while all other methods perform just as well as or better than they did on $C_{ut}$.

| $C_{comd}$ 23 by 23 | max eig cond no | symm. error average | cond($Y$) average | rank($Y$) average | runtime average | # of runs |
|---|---|---|---|---|---|---|
| method (1) (with orthog) | | 1.0436e-06 | 2.1885e+06 | 23 | 2.2187e+00 | 5 |
| method (2) | | 3.3918e-14 | 1.2436e+11 | 23 | 5.9782e-02 | 10 |
| method (3) | 1.7e+11 | 8.3683e-12 | 6.1584e+17 | 15 | 5.6568e-03 | 10 |
| method (4) | 1.7e+11 | 4.2505e-01 | 4.2133e+04 | 23 | 1.7365e-02 | 10 |
| method (5) | 1.7e+11 | 8.5060e-02 | 8.0431e+05 | 23 | 1.2191e-02 | 10 |
| method (6) | | 3.3190e-14 | 7.2333e+10 | 23 | 1.3997e-02 | 10 |

We conclude from these tests that the eigen-principal vectors and Schur based methods (3) through (6) are much faster than the linear means methods (1) and (2) if they succeed to find a nonsingular symmetrizer. Yet the iterative method (1) (with or without re-orthogonalization) always seems to succeed and converge to a full rank and low condition number symmetrizer. In particular, the eigen-principal vectors and Schur based methods (3) through (6) often fail, especially when the eigenvalues of $A$ are ill conditioned.

We conclude our tests with real random entry matrices $R_{50,50}$, $R_{100,100}$, $R_{200,200}$, $R_{300,300}$, $R_{500,500}$ and $R_{1000,1000}$ and our methods as long as they do not take more than a few seconds of CPU time.
Specifically we create 5 such matrices for each dimension $n = 50, 100, 200, 300, 500, 1000$ using MATLAB's `randn.m` function for the table below. There we record the averages of their relative symmetrizing errors, of the condition numbers of their computed symmetrizers, of their ranks, as well as of the CPU times for each matrix set in each dimension. Note that for non defective matrices $R$, randomization effects occur only in our linear methods (1) and (2), but not at all in the eigen-principal vectors or Schur based methods (3) through (6).

| random entry matrix $R$ 50 by 50 | range of eig cond nos for the set of $R$ | symm. error average | cond($Y$) average | rank($Y$) average | runtime average | # of test matrices |
|---|---|---|---|---|---|---|
| method (1) (without orthog) | | 2.1247e-10 | 1.4421e+03 | 50 | 7.4510e-01 | 5 |
| method (2) | | 2.9846e-15 | 1.6910e+03 | 50 | 1.9221e+00 | 5 |
| method (3) | 12 to 32 | 6.3114e-15 | 3.5308e+03 | 50 | 1.2582e-02 | 5 |
| method (4) | 12 to 32 | 6.3114e-15 | 3.5308e+03 | 50 | 8.4248e-03 | 5 |
| method (5) | 12 to 32 | 6.3114e-15 | 3.5308e+03 | 50 | 1.4551e-02 | 5 |
| method (6) | | 6.2200e-15 | 2.9624e+03 | 50 | 1.3444e-02 | 5 |
| $R$ 100 by 100 | | | | | | |
| method (1) (without orthog) | | 2.0876e-10 | 8.4251e+03 | 100 | 1.1775e+01 | 5 |
| method (2) | | 8.8165e-14 | 7.4259e+04 | 100 | 2.2242e+01 | 5 |
| method (3) | 9 to 40 | 9.3032e-15 | 7.6748e+03 | 100 | 3.0089e-02 | 5 |
| method (4) | 9 to 40 | 9.3032e-15 | 7.6748e+03 | 100 | 3.2975e-02 | 5 |
| method (5) | 9 to 40 | 9.3032e-15 | 7.6748e+03 | 100 | 2.8666e-02 | 5 |
| method (6) | | 8.1411e-15 | 6.9116e+03 | 100 | 5.8637e-02 | 5 |
| $R$ 200 by 200 | | | | | | |
| method (1) (without orthog) | | 1.7520e-10 | 3.6727e+04 | 200 | 2.6469e+02 | 5 |
| method (3) | 22 to 68 | 1.1745e-14 | 1.7961e+04 | 200 | 1.8043e-01 | 5 |
| method (4) | 22 to 68 | 1.1745e-14 | 1.7961e+04 | 200 | 1.8913e-01 | 5 |
| method (5) | 22 to 68 | 1.1745e-14 | 1.7961e+04 | 200 | 1.6873e-01 | 5 |
| method (6) | | 1.0682e-14 | 2.2990e+04 | 200 | 4.9192e-01 | 5 |
| $R$ 300 by 300 | | | | | | |
| method (3) | 23 to 157 | 1.3146e-14 | 9.4047e+04 | 300 | 4.1324e-01 | 5 |
| method (4) | 23 to 157 | 1.3146e-14 | 9.4047e+04 | 300 | 4.6076e-01 | 5 |
| method (5) | 23 to 157 | 1.3146e-14 | 9.4047e+04 | 300 | 3.7754e-01 | 5 |
| method (6) | | 1.2226e-14 | 9.2929e+04 | 300 | 1.6173e+00 | 5 |
| $R$ 500 by 500 | | | | | | |
| method (3) | 71 to 91 | 1.5967e-14 | 6.6757e+04 | 500 | 1.5002e+00 | 5 |
| method (4) | 71 to 91 | 1.5967e-14 | 6.6757e+04 | 500 | 1.7679e+00 | 5 |
| method (5) | 71 to 91 | 1.5967e-14 | 6.6757e+04 | 500 | 1.5570e+00 | 5 |
| method (6) | | 1.4498e-14 | 7.8587e+04 | 500 | 8.7191e+00 | 5 |
| $R$ 1000 by 1000 | | | | | | |
| method (3) | 79 to 497 | 1.6896e-14 | 1.0361e+06 | 1000 | 1.0834e+01 | 5 |
| method (4) | 79 to 497 | 1.6896e-14 | 1.0361e+06 | 1000 | 1.3191e+01 | 5 |
| method (5) | 70 to 497 | 1.6896e-14 | 1.0361e+06 | 1000 | 1.0861e+01 | 5 |
| method (6) | | 1.6678e-14 | 9.6336e+05 | 1000 | 7.1673e+01 | 5 |

For generic non-defective matrices with moderate eigenvalue condition numbers, all our eigen-principal vectors methods (3), (4), (5) and the Schur method (6) perform near equally well and, besides, the eigen based methods (3) through (5) compute the same symmetrizer, since clusters of eigenvalues are rarely encountered for random entry matrices and thus only the 1st and 2nd steps at the beginning of Section 3 are executed in most generic cases. Their average run times are very similar and they are each many times faster for large dimensional non defective matrices $A_{n,n}$ than our linear based algorithms (1) and (2) which should essentially not be used whenever $n \gg 120$.

Unfortunately, when eigenvalues repeat or the eigenvector system is badly conditioned for $A$, the new eigen-principal vectors and Schur based symmetrizer algorithms (3) through (6) often fail to find nonsingular and well conditioned symmetrizers as illustrated above.

Finally we note that the zero matrix $O_n$ is a symmetrizer for every $n$ by $n$ matrix $A$, but it carries no information of $A$ and is utterly useless for factoring $A = S_1 S_2$ into the product of two symmetric matrices $S_i$. Likewise rank deficient symmetrizers do not help to factor a given matrix $A$ as the product of two symmetric ones. For low dimensional defective matrices or low dimensional matrices with ill conditioned eigensystems the more cumbersome iterative method (1) and method (2) have a decisive edge over our new eigen-principal vectors and Schur based algorithms.

We have also tested two potential pre-conditioners for the matrix symmetrizer problem and our three eigen-principal vectors based methods, but unfortunately to little avail. One preconditioner is used successfully in the iterative method (1) which starts off with replacing $A_{n,n}$ by $A - \text{trace}(A)/n \cdot I_n$. This, however, had no beneficial effect in our eigen-principal vectors based methods. Another potential preconditioner was suggested by one referee, namely to try David Watkins' idea of not balancing the matrix before applying the QR algorithm for eigenvalues, see [39]. While this helps in some different cases to reduce the eigencondition numbers by around one order of magnitude, it gives equal or worse rank symmetrizers for our eigen-principal vectors based methods (3), (4), and (5) and our test matrices. Therefore we have currently abandoned preconditioning for the eigen based approach to finding symmetrizers.

# 5 Acknowledgements, Assessment and Open Optimization Problems

Matrix symmetrizers and the iterative algorithm (1) of Uhlig [29] were the subject of the second author's talk at the LAA editors' conference upon 40 years of *Linear Algebra and its Applications* (LAA) and Hans Schneider's retirement as editor-in-chief of LAA for 36 years, held in Wisconsin in October 2012. Immediately after the talk a debate ensued between the two authors of this paper and Paul Van Dooren on the value of stable methods for computing Jordan normal forms and applying them to the symmetrizer problem in light of formula $(I)$.
This made us reconsider the problem in a different light and led to the first naive version of an eigen based symmetrizer method in method (3) using classical principal vector chains. However, the results were often rather poor, symmetrizer conditioning and rank wise. Golub and Wilkinson [13, sect. 12] may have anticipated this problem when they said on p. 604 *'... since the vectors in the [principal vector] chains may be arbitrarily near to linear dependence'* and then asked for further discussions in the community. Golub and Wilkinson then continue in [13, sect 13] to discuss the Frank matrix eigenvalue behavior under QR in great detail and in [13, sect. 14] on how to calculate ONBs of invariant subspaces. Unfortunately, we can not apply these ideas efficiently to our problem. However, [13] has inspired our Method (6) which is based on the very expensive linear equations method (2). Eventually we struck upon Arnoldi's method, that is now mainly used for large sparse matrix problems, and we applied it to our small dense ones. Arnoldi is used by us to find an ONB of a principal vector space for one eigenvalue starting from a maximal grade principal vector. It gives us a representation of the matrix action on that space in Hessenberg form. This approach (5) turns out to work modestly well, being many orders of magnitude better than the first naive principal vector chains only based symmetrizer algorithm (3). But while fast, it is not perfect. One of the referees suggested that we ought to consider and work with ONBs for each principal vector subspace of $A$ instead. This has led us to develop the eigen based method (4) and to discover the linear equations method (2) as needed to implement (4), as well as to the Schur Normal Form approach (6) which essentially combines all separate principal vector subspaces for the same eigenvalue of $A$ into one invariant subspace and performs similarly else wise.

The work ahead consists of trying to overcome several compounded problems in this realm: *for one* we need to reconfigure and rethink how to approach defective matrix eigenvalue computations, such as how to overcome the circular smearing of repeated eigenvalues computed by QR (or by any other backward stable algorithm) and how to distill the actual eigenvalues if circles of them are discovered in QR. This circular smearing comes from the combination of two effects: that QR computes the exact eigenvalues of a nearby perturbed matrix and that perturbations of defective (or of almost defective) matrices lead to perturbed eigenvalues arranged approximately on the vertices on regular polygons, according to classical well-known results in eigenvalue perturbation theory (see [19, Ch. 2] or [40, p. 77 ff]). In fact, if the input data of defective matrix is not stored exactly in the computer, perturbation theory implies that circular smearing of eigenvalues already occurs for the exact eigenvalues of the stored nearby matrix on which QR works. This is one intrinsic bottleneck for computing well conditioned sym-

metrizers via eigenanalysis based methods, and this is a long standing and well known open problem in Numerical Linear Algebra.
*For another*, we need to learn how to find well conditioned symmetrizers via speedy eigen- and principal vectors or Schur based algorithms, not only when $A$ has Jordan blocks of size larger than $1 \times 1$, but also when $A$ is diagonalizable, i.e. non-defective, and when its eigenvector matrix is highly ill conditioned. Example 1 in Section 2 shows that to reach this goal may be possible but nontrivial.
If these two bottlenecks were removed, then *thirdly* a further set of open problems involves finding optimization strategies on how to add newly computed partial symmetrizers to an earlier compounded symmetrizing matrix $Y$ of $A$ in such a way as to maximize the rank of the resulting $Y$. As the space of symmetrizers of any given matrix is linear, one might add or subtract any multiple of a partial symmetrizer of $A$ to $Y$, but which multiple?

We now illustrate the second complication that can arise when computing symmetrizers from eigen-data. Here we study a diagonalizable matrix with ill conditioned eigenvalues such as the original, unmodified Frank matrix $F_{35}$ of size 35 by 35.
The original Frank matrix $F_n$ of [10] is an integer upper Hessenberg matrix of the form

$$F_n = \begin{pmatrix} n & n-1 & \dots & \dots & 3 & 2 & 1 \\ n-1 & n-1 & & & 3 & 2 & 1 \\ 0 & n-2 & \ddots & & 3 & 2 & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & & 0 & 3 & 3 & 2 & 1 \\ \vdots & & & 0 & 2 & 2 & 1 \\ 0 & \dots & \dots & \dots & 0 & 1 & 1 \end{pmatrix}_{n,n} .$$

According to [10] and [13], the eigenvalues of Frank matrices are all real and positive, they are distinct and come in reciprocal pairs, and the largest $[n/2]$ eigenvalues are rather well conditioned while the smaller ones less than 1 become increasingly ill conditioned. Its worst conditioned eigenvalue has a condition number of 6.8e+09 for $n = 35$. Although Francis' QR algorithm is perfectly backward stable, the large condition numbers of the smallest eigenvalues of $F_{35}$ imply that QR may compute these eigenvalues with large forward errors. This is confirmed by numerical tests, since QR computes the small eigenvalues of $F_{35}$ as lying on a circle with a diameter of around 6 units and, besides, it misses $F$'s central eigenvalue 1 altogether, see Figure 2.

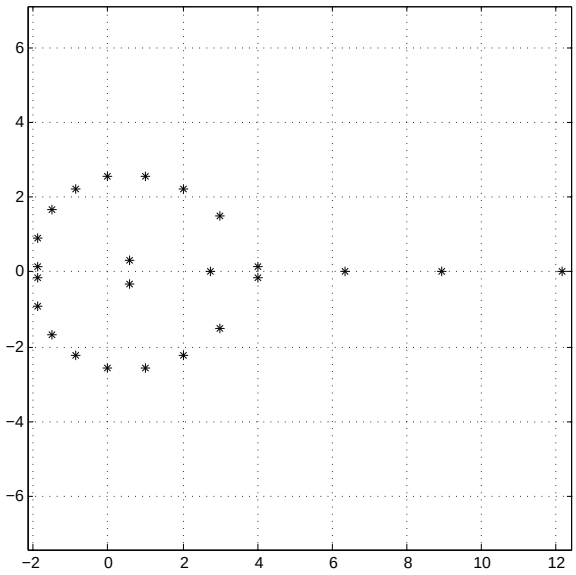


Figure 2: Small eigenvalues of $F_{35}$ are smeared around a circle in $\mathbb{C}$ by Francis QR method for $F_{35}x = \lambda x$.

To compute more accurate eigenvalues and eigenvectors for $F_{35}$, we rely on both Frank's and Golub's and Wilkinson's work, see [10, 13]: they suggested an equivalent generalized eigenvalue problem formulation $Ax = \lambda Bx$ that can find the eigenvalues of Frank matrices $F_n$ with smaller forward errors, because the eigenvalue condition numbers for this generalized eigenvalue problem are much smaller than those of $F_n$. Here $A = B \cdot F_n$ for

$$B = \begin{pmatrix} 1 & -1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & -1 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix}_{n,n} .$$

Since this generalized eigenvalue problem $Ax = \lambda Bx$ is exactly constructed (i.e., without rounding errors) and MATLAB's QZ algorithm `eig(A,B)` is backward stable, it finds a set of more accurate real eigenvalues that exceed 1 for $F_{35}$ than QR does, because as mentioned above, the eigenvalue condition numbers of $Ax = \lambda Bx$ are smaller than those of $F_{35}$. But it also misplaces the small real eigenvalues onto a circle in $\mathbb{C}$, yet with a smaller diameter of around 1 unit than QR as a consequence of the better conditioning of the eigenvalues of $Ax = \lambda Bx$. Moreover QZ for $A$ and $B$ computes the central eigenvalue $\lambda = 1$ reasonably well as 1.00097 with $0.1\%$ inaccuracy, see Figure 3.

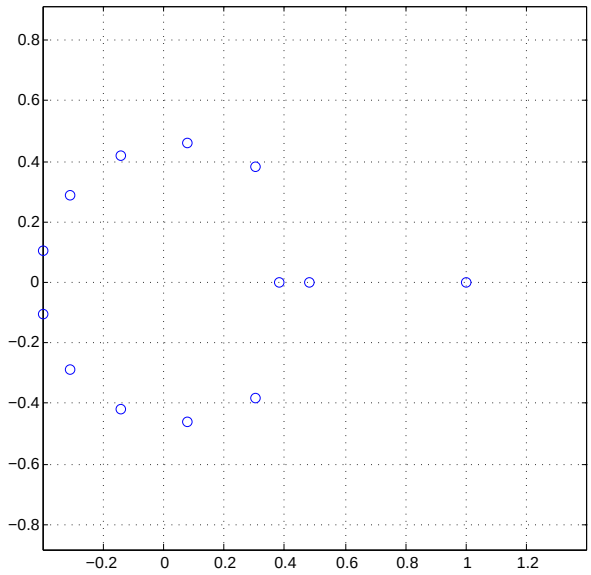


Figure 3: Small eigenvalues of $F_{35}$ are located on a small circle in $\mathbb{C}$ by the QZ method for $Ax = \lambda Bx$.

The table below shows the results for the naive symmetrizer $Y = V \cdot V^T$ that is obtained from the eigenvectors $V$ computed by MATLAB for the eigenvalues provided by the Francis QR and the QZ algorithms, as well as by an extended QZ algorithm where the reciprocals of the computed eigenvalues above 1 are used for the under 1 eigenvalues of $F_{35}$, the central eigenvalue of $F$ is set by hand to 1 and all respective eigenvectors are computed by inverse iteration. Our results below are ordered by increasing symmetrizer rank, with the computed results from our methods (1) through (6) interspersed where they fit.

| original Frank matrix from [10], 35 by 35 | max eigenvalue condition number | relative symm. error | cond($V$) | cond($Y$) | rank($Y$) |
|---|---|---|---|---|---|
| QZ with reciprocals and inverse iteration | | 4e-16 | 6.7e+16 | 1.5e+18 | 23 |
| QZ only | | 4.1e-15 | 6.1e+17 | 3.6e+18 | 25 |
| Schur method (6) | | 1.4e-10 | | 3.6e+17 | 28 |
| Francis QR, without clustering | 6.9e+9 | 2.7e-11 | 3.5e+10 | 6.5e+16 | 31 |
| eigendata method (3) with cluster radius = 0.1 | 6.9e+9 | 1.0e-10 | | 3.17e+16 | 32 |
| eigendata method (4) with cluster radius = 0.1 | 6.9e+9 | 1.8e-10 | | 4.29e+16 | 32 |
| eigendata method (5) with cluster radius = 0.1 | 6.9e+9 | 3.6e-08 | | 5.68e+16 | 33 |
| linear equations method (2) | | 2.7e-15 | | 5.4e+14 | 34 |
| iterative method (1) (with orthog) | | 9.1e-11 | | 4.3e+10 | 35 |

It is perplexing that the extended QZ algorithm that is combined with inverse iteration for the first entry line in the above table uses the most precise eigenvalue and eigenvector data available for this matrix, but it turns out to be most unreliable with regards to rank and finding an invertible symmetrizer. The eigen- and principal vector based symmetrizer algorithms and Francis QR, on the other hand, are almost good enough but have the most sloppily computed eigenvalues, see Figure 2 again. And even the direct linear equations solver (2) misses to compute a full rank symmetrizer. And only the iterative scheme (1) computes an invertible symmetrizer for the original Frank matrix. Therefore we now state a related open problem, namely: *fourthly* why and how is the iterative method (1) of [30] seemingly impervious to eigenvalue conditioning issues of $A$ and how does it generally compute symmetrizers with low condition numbers, often lower by orders of magnitude than our current best eigen- and principal vector or Schur based algorithms.

Given that our computed naive eigenvector based symmetrizers all have the form $Y = V \cdot V^T$ for an eigenvector matrix $V$ of $F_{35}$, but suffer eigenvalue ill-conditioning or rank deficiencies, one way to solve this problem might be to replace $Y$ judiciously by $Y_D = V \cdot D \cdot V^T$ and try to minimize the condition number of $Y_D$ over all nonsingular diagonal $D$, see the discussion in Example 1 and the first Open Problem below.

We wrap up this study of symmetrizer and eigenstructure computations with two open problems in matrix optimization, the first of which will be exemplified through a simple 2 by 2 matrix example that can be solved algebraically and complements and extends the discussion in Example 1. We use the Frobenius norm here instead of the spectral norm to keep the developments as simple as possible. Note that for 2 by 2 matrices $\|A\|_2 \leq \|A\|_F \leq \sqrt{2}\,\|A\|_2$ and, therefore, the differences between these norms are slight.

Consider the matrix

$$A = \begin{bmatrix} 0 & 1 \\ 0 & \delta \end{bmatrix} \tag{9}$$

where $\delta > 0$ is a small real parameter. The column and row eigenvectors $x_{..}$ and $y_{..}$ for $A$'s eigenvalues $\lambda_0 = 0$ and $\lambda_\delta = \delta$ on the right side and left side, respectively, are

$$\begin{aligned} &\text{for } \lambda_0 = 0, \quad x_0 = [1,0]^T, \quad y_0 = [-\delta, 1], \\ &\text{for } \lambda_\delta = \delta, \quad x_\delta = [1,\delta]^T, \quad y_\delta = [0,1]\,. \end{aligned}$$

The near singularity of $A$ is unessential here since one can shift $A$ to $A + bI_2$ and this does not change the eigenvectors, nor the set of right (or left) symmetrizers. Of course, each specific pair $S_1,\ S_2$ of symmetrizers for $A = S_1 S_2$ is changed by shifting.

The Wilkinson condition numbers for the eigenvalues of $A$ are

$$\text{cond}(0) = \text{cond}(\delta) = \frac{\sqrt{1+\delta^2}}{\delta}.$$

These are huge if $\delta \ll 1$. A right side eigenvector matrix of $A$ is

$$V = \begin{bmatrix} 1 & 1 \\ 0 & \delta \end{bmatrix}.$$

Its condition number is huge if $\delta \ll 1$. As $A$ is diagonalizable and non derogatory, all right side symmetrizers of $A$ have the form

$$Y = V \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix} V^T = \begin{bmatrix} (d_1 + d_2) & d_2\delta \\ d_2\delta & d_2\delta^2 \end{bmatrix} \tag{10}$$

for arbitrary parameters $d_i$. The norm of the general symmetrizer $Y$ is

$$\|Y\|_F = \sqrt{|d_1 + d_2|^2 + 2|d_2|^2\delta^2 + |d_2|^2\delta^4}\,. \tag{11}$$

Its inverse is

$$Y^{-1} = \frac{1}{d_1 d_2 \delta^2} \begin{bmatrix} d_2\delta^2 & -d_2\delta \\ -d_2\delta & (d_1 + d_2) \end{bmatrix} \tag{12}$$

and therefore its condition number in the Frobenius norm is

$$\mathrm{cond}_F(Y) = \|Y\|_F \|Y^{-1}\|_F = \frac{|d_1 + d_2|^2}{|d_1|\,|d_2|}\frac{1}{\delta^2} + \frac{|d_2|}{|d_1|}\delta^2 + 2\frac{|d_2|}{|d_1|}\,. \tag{13}$$

This formula implies that for $\delta \ll 1$ and our $A$, well conditioned right side symmetrizers of the form $V \cdot D \cdot V^T$ must use a matrix $D = \mathrm{diag}(d_1, d_2)$ with a small value for $|d_1 + d_2|$ or with $d_1 \approx -d_2$, i.e., matrices $D$ with entries of opposite signs and near equal magnitude. This implies that well conditioned right symmetrizers are very particular and would be found very rarely by a random assignment of $d_1$ and $d_2$. In addition, according to Theorem 2(a), in order to guarantee a tiny residual error bound for floating point computations[3], the ratio

$$\frac{|d_1| + |d_2|}{\|Y\|_F} = \frac{|d_1| + |d_2|}{\sqrt{|d_1 + d_2|^2 + 2|d_2|^2\delta^2 + |d_2|^2\delta^4}} = \frac{|d_1| + |d_2|}{|d_1 + d_2|}\frac{1}{\sqrt{1 + \dfrac{2|d_2|^2\delta^2 + |d_2|^2\delta^4}{|d_1 + d_2|^2}}}, \tag{14}$$

should be moderate. But, if $\delta \ll 1$, then (14) is moderate if and only if $(|d_1| + |d_2|)/|d_1 + d_2|$ is moderate. Equations (13) and (14) show that, in this example, it is not possible to minimize the condition number and the residual error bound simultaneously. A compromise must be reached between these two goals to achieve an optimal symmetrizer. The following relation involving the coefficient of the first term in (13) gives a clue on how to get this compromise

$$\frac{|d_1| + |d_2|}{|d_1 + d_2|} \geq \frac{2\sqrt{|d_1||d_2|}}{|d_1 + d_2|} = \frac{2}{\sqrt{\dfrac{|d_1 + d_2|^2}{|d_1|\,|d_2|}}}.$$

To make the discussion more concrete, we now compare the results for three different choices of $d_1$ and $d_2$ in (10): first we consider the symmetrizer $Y_1$ for the standard naive choice $d_1 = d_2 = 1$, then the symmetrizer $Y_2$ for the compromise choice $d_1 = 1 + 1/\sqrt{\delta}$, $d_2 = -1/\sqrt{\delta}$, and finally $Y_3$ for the well conditioned choice $d_1 = -1/\delta$ and $d_2 = 1/\delta$. These symmetrizes are

$$Y_1 = \begin{bmatrix} 2 & \delta \\ \delta & \delta^2 \end{bmatrix}, \quad Y_2 = \begin{bmatrix} 1 & -\sqrt{\delta} \\ -\sqrt{\delta} & -\delta^{3/2} \end{bmatrix}, \quad \text{and} \quad Y_3 = \begin{bmatrix} 0 & 1 \\ 1 & \delta \end{bmatrix},$$

respectively, and we summarize the results for them assuming $\delta \ll 1$ and according to (11)-(13)-(14) in the next table

[3] Recall that Theorem 2 was proved with eigenvectors of norm 1 as they are computed by MATLAB and LAPACK. The second column norm of $V$ for the symmetrizer $Y$ in (10) is $\sqrt{1 + \delta^2} \neq 1$. But for $\delta^2 \ll 1$ this is very close to one. For simplicity we have avoided normalizing the second column of $V$ as it does not lead to any difference in our discussion.

| $[d_1, d_2]$ | $\text{cond}_F(Y)$ | $\|Y\|_F$ | $(\|d_1\| + \|d_2\|)/\|Y\|_F$ |
|---|---|---|---|
| $[1\ ,\ 1]$ | $\approx 4/\delta^2$ | $\approx 2$ | $\approx 1$ |
| $[1 + 1/\sqrt{\delta}\ ,\ -1/\sqrt{\delta}]$ | $\approx 1/\delta$ | $\approx 1$ | $\approx 2/\sqrt{\delta}$ |
| $[-1/\delta\ ,\ 1/\delta]$ | $\approx 2$ | $\approx \sqrt{2}$ | $\approx \sqrt{2}/\delta$ |

If we use $\delta = 10^{-8}$ and the unit roundoff $\mathbf{u} \approx 10^{-16}$, then in floating point arithmetic according to Theorem 2 and the table above, the standard naive symmetrizer $Y_1$ yields a perfect residual error bound of order $\mathbf{u}$ but it is numerically singular since $\text{cond}_F(Y_1) \approx 4 * 10^{16}$; the near perfectly conditioned symmetrizer $Y_3$ has a small condition number of 2 with a residual error bound $\approx 10^{-8}$, as given by the last column of the table times $\mathbf{u}$; and the compromise symmetrizer $Y_2$ achieves $\text{cond}_F(Y_2) \approx 10^8$ with a medium sized residual error bound $\approx 10^{-12}$. The $\delta^{-2}$ dependence of $\text{cond}_F(Y)$ in this example makes the naive symmetrizer $Y_1$ useless, while both $Y_2$ and $Y_3$ retain moderately small residuals with almost reasonable conditioning.

We make two remarks on the example for symmetrizers (10) as discussed above. First, the reader should not draw the conclusion that to compute symmetrizers with a much smaller condition number than the "standard naive symmetrizer $VV^T$" for any diagonalizable matrix $A$ implies residual error bounds significantively larger than $\mathbf{u}$. This is generally false as illustrated by Example 1 in Section 2 for the choice $\text{diag}(1, 1, -1)$. Secondly, generalizing from (10) to the $n$ by $n$ case, it might be that in some instances only very particular choices of the $d_i$ in $D = \text{diag}(d_i)$ can give well conditioned naive symmetrizers of the form $V \cdot D \cdot V^T \in \mathbb{F}^{n,n}$. This indicates that simply adding freshly computed partial symmetrizers to our current best symmetrizer $Y$ – as we currently do for lack of a better method – might be one reason that our eigendata methods (3) to (6) often compute rank deficient symmetrizers. How can this be remedied?

The iterative method (1) of [29] computes a nonsingular symmetrizer for our example matrix $A_{2,2}$ in (9) with $\delta = \text{eps}$, the machine constant, as

$$Y_{iter} = \begin{pmatrix} 0.7437.. & 1.2563.. \\ 1.2563.. & 0 \end{pmatrix}.$$

Its 2-norm condition number is 1.771.. , which is excellent. The direct linear solver (2) obtains the nonsingular symmetrizer

$$Y_{lin} = \begin{pmatrix} 0.8485.. & 0.7336.. \\ 0.7336.. & 1.629\text{e-}16 \end{pmatrix}$$

with 2-condition number 14.12. However, our general eigen- and principal vector based algorithms (3) through (5) all compute the same numerically singular symmetrizer for $A$ and $\delta = \text{eps}$ as

$$Y_{eig} = \begin{pmatrix} 4.0000 & 4.4409\text{e-}16 \\ 4.4409\text{e-}16 & 9.8608\text{e-}32 \end{pmatrix}$$

with condition number $8.11 \cdot 10^{31}$. And the Schur based method (6) obtains the nonsingular symmetrizer

$$Y_{Schur} = \begin{pmatrix} 1.2213.. & 1.6617.. \\ 1.6617.. & 0 \end{pmatrix}$$

with condition number 5.205. The relative symmetrizing errors for each of these methods are all at or below the machine constant eps.

This example and our analysis lead to the following open and intriguing matrix optimization problem:

**First Open Matrix Optimization Problem {# 1}:**

Given a nonsingular $n$ by $n$ matrix $V$, find a set of nonzero parameters $d_i$ so that for $D = \text{diag}(d_i)$ the matrix condition number of

$$S = V * D * V^T$$

is minimal.
Besides, find lower and upper bounds for

$$M_D(V) = \text{Min}_{D \text{ diagonal}} \{\text{cond}(V * D * V^T)\}$$

from $V$ in any suitable matrix norm.

Motivated by Theorem 2, these problems should be contemplated in floating point computational settings with the following constrained matrix optimization problem: find a set of nonzero parameters $d_i$ so that for $D = \text{diag}(d_i)$ the matrix condition number of

$$S = V * D * V^T$$

is minimal under the constraint $\|V\|^2 \left(\sum_{i=1}^{n} |d_i|/\|S\|\right) \leq \alpha$, where $\alpha$ is a fixed constant that determines the admissable residual error bound.

To rephrase these problems, we are looking for, with or without constraints, the best conditioned matrix inside the subspace of matrices spanned by the column vector dyads $\{v_1 \cdot v_1^T, ..., v_n \cdot v_n^T\}$ of a given matrix $V$ with columns $v_i$.

The linear equations based symmetrizer algorithm (2) differs from our other methods by the fact that it computes a full basis of the linear subspace of matrix symmetrizers for $A$ through the use of the `null.m` and `nulls.m` m-files. This leads us to pose a second and related open matrix optimization problem:

**Second Open Matrix Optimization Problem {# 2}:**

Given a basis for a subspace of $n$ by $n$ matrices, how can one determine a nonsingular matrix in the subspace, and how can one find one with a small condition number, if such exists?

***Afterthoughts***

The eigen- and principal vector and Schur based methods are titillating with their speed, accuracy and well conditioned symmetrizer computations for the generic and the well conditioned matrix eigenvalue case, but they are near useless should the matrix $A$ have repeated or ill conditioned eigenvalues, let alone an involved Jordan structure. For relatively small $n < 120$, the iterative method (1) serves us well in all cases with well conditioned symmetrizers, but it has a huge computational cost.
Can matrix optimization techniques help at all for eigen- and principal vector and Schur based symmetrizer algorithms and how? What is going on with matrix symmetrizers, geometrically, numerically and computationally?

---

**Acknowledgements**. We are grateful for the support of and suggestions from referees and members of the mathematics communities in many places, in England, China, Germany, Hong Kong, Korea, Malaysia, Spain, the USA and elsewhere, as well as for the advice and critique of all who have helped us shape and expand this study.

# References

[1] S. ADHIKARI, *On symmertrizable systems of second kind*, J. of Applied Mechanics, 67 (2000), p. 797-802.

[2] E. ANDERSON, Z. BAI, C. BISCHOF, S. BLACKFORD, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, AND D. SORENSEN, *LAPACK Users' Guide*, 3rd ed., SIAM (1999).

[3] WALTER EDWIN ARNOLDI, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quarterly of Appl. Math, 9 (1951). p. 17 - 29.

[4] ATUL BHASKAR, *Taussky's theorem, symmetrizability and modal analysis revisited*, Proc. R. Soc. London A, 457 (2001), p. 2455-2480.

[5] BARRY A. CIPRA, *The best of the 20th century: Editors name top 10 algorithms*, SIAM News, 33, no. 4 (2000), `http://www.siam.org/news/news.php?id=637` .

[6] BISWA NATH DATTA, *Quadratic forms, matrix equations and the matrix eigenvalue problem*, Ph.D. thesis, University of Ottawa, 1972.

[7] BISWA NATH DATTA, *An algorithm for computing a symmetrizer of a Hessenberg matrix*, manuscript, (1973), 2 p.

[8] E. J. DESAUTELS, *Symmetrizing matrices*, MS thesis, University of Ottawa, 1968, 42 p;
`http://www.ruor.uottawa.ca/bitstream/10393/10879/1/EC52206.PDF`

[9] JACK DONGARRA AND FRANCIS SULLIVAN, *The top 10 algorithms*, Comput. Sc. Eng., 2 (2000), p. 22 - 23.

[10] WERNER L. FRANK, *Computing eigenvalues of complex matrices by determinant evaluation and by methods of Danilewski and Wielandt*, J. SIAM, 6 (1958), p. 378 - 392.

[11] FERDINAND GEORG FROBENIUS, *Über die mit einer Matrix vertauschbaren Matrizen*, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin (1910), p. 3 - 15; also in *Gesammelte Abhandlungen*, Band 3, Springer 1968, p. 415 - 427.

[12] GENE H. GOLUB AND CHARLES F. VAN LOAN, *Matrix Computations*, 4th ed., Johns Hopkins University Press (2013), 756 p.

[13] GENE H. GOLUB AND JAMES H. WILKINSON, *Ill-conditioned eigensystems and the computation of the Jordan canonical form*, SIAM Review, 18 (1976), p. 578 - 619.

[14] NICHOLAS J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM (2002).

[15] JAMES LUCIEN HOWLAND AND F. J. FARREL, *Matrix symmetrizing methods for the algebraic eigenvalue problem*, A.C.M Nat. Conference, Colorado, 1963.

[16] JIANGUO HUANG AND LIWEI NONG, *An iterative algorithm for solving finite-dimensional linear operator equations* $T(x) = f$ *with applications*, Linear Algebra Appl, 432 (2010), p.1176 - 1188.

[17] D. J. INMAN AND C. L. OLSEN, *Dynamics of symmetrizable nonconservative systems*, ASME J. of Applied Mechanics, 55 (1988), p. 206-212.

[18] BO KÅGSTRÖM AND AXEL RUHE, *An algorithm for numerical computation of the Jordan normal form of a complex matrix*. ACM Trans. Math. Software, 6 (1980), p. 398 - 419.

[19] TOSIO KATO, *Perturbation Theory for Linear Operators*, corrected 2nd ed., Springer-Verlag, Berlin (1980).

[20] VERA KUBLANOVSKAYA, *A method for solving the complete problem of eigenvalues of a degenerate matrix* (Russian), Z. Vycisl. Mat. i Mat. Fiz., 6 (1966), p. 611 - 620; translated in USSR Comp. Math. and Math. Phys., 6 (1968), p. 1 - 14.

[21] AXEL RUHE, *An algorithm for numerical determination of the structure of a general matrix*, Nordisk Tidskr. Informationsbehandling (BIT), 10 (1970), p. 196 - 216.

[22] YOUSEF SAAD, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM (2003).

[23] G. W. STEWART, *Algorithm 406: HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix*, ACM Trans. Math. Softw., 2 (1976), p. 275-280.

[24] OLGA TAUSSKY, *Notes on numerical analysis. II. Note on the condition of matrices*, Math. Tables and Other Aids to Computation 4, (1950), p. 111 - 112.

[25] OLGA TAUSSKY AND HANS ZASSENHAUS, *On the similarity transformation between a matrix and its transpose*, Pacific J Math, 9 (1959), p. 893 - 896.

[26] OLGA TAUSSKY, *The role of symmetric matrices in the study of general matrices*, Linear Algebra Appl., 5 (1972), p. 147 - 154.

[27] SIVAN TOLEDO, *MATLAB code of* `nulls.m`, available at `www.tau.ac.il/~stoledo/Tools/nulls.m` , (2005).

[28] LOTHAR TRAPP, *Ein Algorithmus zur Berechnung der Symmetrizer einer beliebigen reellen Matrix und einige Anwendungen*, Diplomarbeit, Uni Würzburg, Germany, 1975, 149 p.

[29] FRANK UHLIG, *Computing matrix symmetrizers, finally possible via the Huang and Nong algorithm*, Linear and Multilinear Algebra, 61 (2013), p. 954 - 969, http://dx.doi.org/10.1080/03081087.2012.716427

[30] FRANK UHLIG, *Method (1) MATLAB m-file* `symmorthlongv.m` , available at `http://www.auburn.edu/~uhligfd/m_files/MatrixSymm/symmorthlongv.m` , (2012).

[31] FRANK UHLIG, *Method (2) MATLAB m-file* `Symmlinequatl.m` , available at `http://www.auburn.edu/~uhligfd/m_files/MatrixSymm/Symmlinequatl.m` , (2014).

[32] FRANK UHLIG, *Method (3) MATLAB m-file* `rightsymmAfulljordan8l.m` , available at `http://www.auburn.edu/~uhligfd/m_files/MatrixSymm/rightsymmAfulljordan8l.m`, (2014).

[33] FRANK UHLIG, *Method (4) MATLAB m-file* `rightsymmAGSplusLinEqul.m`, available at `http://www.auburn.edu/~uhligfd/m_files/MatrixSymm/rightsymmAGSplusLinEqul.m` , (2014).

[34] FRANK UHLIG, *Method (5) MATLAB m-file* `rightsymmAjordanArnoldi1l.m` , available at `http://www.auburn.edu/~uhligfd/m_files/MatrixSymm/rightsymmAjordanArnoldi1l.m` , (2013).

[35] FRANK UHLIG, *Method (6) MATLAB m-file* `rightsymmSchurplusLinEqu2l.m` , available at `http://www.auburn.edu/~uhligfd/m_files/MatrixSymm/rightsymmSchurplusLinEqu2l.m` , (2014).

[36] FRANK UHLIG, *Matrix Symmetrizer MATLAB m-files folder with all the above m-files and testing programs*, available at `http://www.auburn.edu/~uhligfd/m_files/MatrixSymm` , (2014).

[37] ABRAHAM VAN DER SLUIS, *Condition numbers and equilibration of matrices*, Numer. Math., 14 (1969), p. 14-23.

[38] V. CH. VENKAIAH AND S. K. SEN, *Computing a matrix symmetrizer exactly using modified multiple modulus residue arithmetic*, J. of Computational and Applied Mathematics, 21 (1988), p. 27-40.

[39] DAVID S. WATKINS, *A case where balancing is harmful*, Electron. Trans. Numer. Anal., 23 (2006), p. 1 - 4.

[40] JAMES H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford (1965).

[41] JAMES H. WILKINSON, *Invariant subspaces*, manuscript, Gatlinburg VI (December 1974), Hopfen am See, Germany, 8 p.

[42] PUMEI ZHANG, *Algebraic properties of compatible Poisson brackets*, Regular and Chaotic Dynamics, 19 (2014), p. 267-288 .

spysymm10.pdf

Frank35eig.pdf

FrankeigQZ.pdf