DELFT UNIVERSITY OF TECHNOLOGY

REPORT 07-09

# A Minimal Residual Method for Shifted Skew-Symmetric Systems

R. Idema, C. Vuik

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2007

# A Minimal Residual Method for Shifted Skew-Symmetric Systems

R. Idema and C. Vuik*

**Abstract**

We describe the MRS$^3$ solver, a Minimal Residual method based on the Lanczos algorithm that solves problems from the important class of linear systems with a shifted skew-symmetric coefficient matrix using short vector recurrences. The MRS$^3$ solver is theoretically compared with other Krylov solvers and illustrated by some numerical experiments.

**Keywords:** Lanczos, Shifted Skew-Symmetric, Minimal Residual, Krylov

**AMS Subject Classification:** 65F10

## 1 Introduction

In this paper we explore Krylov subspace methods that can solve systems of linear equations of the form

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ is a shifted skew-symmetric matrix, i.e.

$$A = \alpha I + N \ , \quad \alpha \in \mathbb{R} \ , \quad N^T = -N. \tag{2}$$

Throughout this paper we will use $I$ for the identity matrix of appropriate size, $M$ for symmetric matrices, and $N$ for skew-symmetric matrices as above. Further we will use the abbreviation SSS for shifted skew-symmetric.

Shifted skew-symmetric systems arise in many scientific and engineering applications. Two important examples are Computational Fluid Dynamics and Linear Programming. In Computational Fluid Dynamics, SSS systems arise when dealing with Navier-Stokes equations with a large [8] or a small [9] Reynolds number (see also [1]). To illustrate the usefulness of having a fast solver for SSS systems, below we will treat an application in CFD and one in LP.

*Delft University of Technology, J.M. Burgerscentrum, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft Institute of Applied Mathematics, Mekelweg 4, 2628 CD Delft, The Netherlands, **e-mail: c.vuik@tudelft.nl**

For an application in CFD we consider $B$ to be a large non-singular matrix, which is a discrete version of an advection-diffusion problem. The Hermitian splitting can be used to decompose $B$ in its symmetric part $M$ and its skew-symmetric part $N$:

$$B = M + N, \text{ where } M = \frac{B + B^T}{2} \text{ and } N = \frac{B - B^T}{2}.$$

If the diffusion is important, i.e., if the Reynolds number is small, and if the symmetric part $M$ of $B$ is nonsingular, $M^{-1}$ can be used as a preconditioner to solve a system $B\mathbf{x} = \mathbf{b}$. Note that to compute $\mathbf{v} = M^{-1}\mathbf{w}$ efficiently, multigrid can be used. The preconditioning can be done as follows:

$$M^{-\frac{1}{2}} B M^{-\frac{1}{2}} \mathbf{y} = M^{-\frac{1}{2}} \mathbf{b}, \text{ where } \mathbf{x} = M^{-\frac{1}{2}} \mathbf{y}. \tag{3}$$

This equation can be rewritten as:

$$(I + M^{-\frac{1}{2}} N M^{-\frac{1}{2}})\mathbf{y} = M^{-\frac{1}{2}} \mathbf{b},$$

which is an SSS system (compare [9]).

On the other hand if advection is dominant, i.e., if the Reynolds number is large,

$$\left(I - (\alpha I + N)^{-1}(M - \alpha I)\right)(\alpha I + N)^{-1}$$

can be used as a preconditioner (see [8] eq. (1.7) and (3.1)). Applying this preconditioner to a vector $\mathbf{w}$ implies that SSS systems of the form $(\alpha I + N)\mathbf{v} = \mathbf{w}$ have to be solved.

For an application in Linear Programming consider interior point methods, a popular way of solving LP problems. When solving a Linear Program with such a method, using a self-dual embedding of the problem takes slightly more computational time per iteration but has several important advantages such as having a centered starting point and detecting infeasibility by convergence, as described in [16]. Therefore, most modern solvers use such an embedding.

Interior point methods are iterative schemes that search for an optimal solution from within the strictly feasible set. In each iteration a step $\Delta\mathbf{x}_i$ to add to the current solution is generated. To calculate this step, a large sparse system of the following form needs to be solved:

$$(D_i + N)\Delta\mathbf{x}_i = \mathbf{b}_i \tag{4}$$

where $N$ is a skew-symmetric matrix and $D_i$ is a diagonal matrix with strictly positive entries on the diagonal.

Using the same preconditioning as in equation (3) we can rewrite system (4) into:

$$(I + D_i^{-\frac{1}{2}} N D_i^{-\frac{1}{2}})\mathbf{y}_i = D_i^{-\frac{1}{2}} \mathbf{b}_i, \text{ where } \mathbf{y}_i = D_i^{\frac{1}{2}} \Delta\mathbf{x}_i. \tag{5}$$

This again is an SSS system. Note that the preconditioning in this case is in fact diagonal scaling.

In the next section we will overview some existing algorithms that can be used to solve SSS systems. We discuss some of their advantages, disadvantages and limitations when solving these systems. In Section 3 we will present a general Lanczos based SSS solver, followed in Section 4 by a comparison of the treated solvers. In Section 5 we treat the results of our numerical experiments, and finally in Section 6 we present our conclusions with respect to solving SSS systems.

# 2 Overview of existing methods

In this section we will give an overview of some existing Krylov subspace methods that can be used to solve shifted skew-symmetric systems. Krylov subspace methods are iterative solvers for systems of linear equations. In iteration $j$ a Krylov subspace method approximates the solution with $\mathbf{x}_j$, where

$$\mathbf{x}_j \in \mathbf{x}_0 + \mathcal{K}_j(A, \mathbf{r}_0).$$

Here $\mathbf{x}_0$ is the initial solution, $A$ is the coefficient matrix of the system, $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ is the initial residual and $\mathcal{K}_j(A, \mathbf{r}_0)$ is the Krylov subspace:

$$\mathcal{K}_j(A, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{j-1}\mathbf{r}_0\}.$$

Since in every iteration the Krylov subspace is expanded, a new approximation within the larger subspace can be generated that is never worse than the previous one.

Two important properties we aspire in Krylov subspace methods are optimality and short recurrences. An algorithm has the optimality property if the generated approximation for the solution is, measured in some norm, the best within the current Krylov subspace. The short recurrences property is satisfied if the algorithm can generate the next approximation using only data from the last few iterations.

For general coefficient matrices the above properties cannot be satisfied simultaneously. However methods satisfying both properties do exist for matrices of the form

$$A = e^{i\theta}(\sigma I + T)\,, \text{ where } \theta \in \mathbb{R},\ \sigma \in \mathbb{C},\ T^H = T.$$

These results are due to Faber and Manteuffel [5], [6]. Taking $\theta = \pi/2$, $\sigma = -i\alpha$ with $\alpha \in \mathbb{R}$ and $T = -iN$ with $N^T = -N$, we get a matrix as given in equation (2). This implies that a Krylov subspace method for SSS systems exists that has both optimality and short recurrences.

## 2.1 General methods

An SSS problem can be solved with any solver for general matrix equations. We will name a few widely used methods that nicely illustrate the findings of Faber and Manteuffel mentioned above.

GMRES [18] generates optimal approximations to the solution, but needs vectors from all the previous iterations to do so. The same holds for the GCR method [22].

Bi-CGSTAB [20] uses short recurrences but does not have the optimality property. Usually it converges fast but it is not robust.

Finally, CGNR [15] solves the normal equations $A^TA\mathbf{x} = A^T\mathbf{b}$ with the CG method. This solver achieves both optimality and short recurrences, but uses a different Krylov subspace. Since the condition number is squared when working with $A^TA$, convergence can be very slow. This method is used for solving SSS systems by Golub and Vanderstraeten in their treatment of the preconditioning of matrices with a large skew-symmetric part [7].

## 2.2 Generalized Conjugate Gradient method

The Generalized Conjugate Gradient method, proposed by Concus and Golub [2] and Widlund [24], is an iterative Lanczos method for solving systems $A\mathbf{x} = \mathbf{b}$ where $A$ has a positive definite symmetric part $M$. The SSS matrix (2) satisfies this requirement if $\alpha > 0$, and for $\alpha < 0$ we can easily meet it by solving $-A\mathbf{x} = -\mathbf{b}$. Thus we can use this method to solve any SSS system with $\alpha \neq 0$.

---

**Algorithm 2.1 (Generalized Conjugate Gradient)**

*Let* $\boldsymbol{x}_{-1} = \boldsymbol{x}_0 = 0\,,\ j = 0$

*While* not converged *do*

*Solve* $M\mathbf{v}_j = \boldsymbol{b} - A\mathbf{x}_j$
$\rho_j = (M\mathbf{v}_j, \mathbf{v}_j)$
*If* $j = 0$
    $\omega_j = 1$
*Else*
    $\omega_j = (1 + (\rho_j/\rho_{j-1})/\omega_{j-1})^{-1}$
*Endif*
$\mathbf{x}_{j+1} = \mathbf{x}_{j-1} + \omega_j(\mathbf{v}_j + \mathbf{x}_j - \mathbf{x}_{j-1})$
$j = j + 1$

*End while*

---

The Generalized Conjugate Gradient method does not have the optimality property. However, it has been proved that the iterates are optimal in some affine subspace other than the Krylov subspace [4].

In practice this method is rarely used, as it has been superseded by the CGW method by the same authors. Therefore we will not go into any further details on this method in this paper.

## 2.3 CGW method

Related to the Generalized Conjugate Gradient method presented above is the algorithm that Saad describes under the name CGW in Section 9.6 of his book [17]. This method also solves systems of linear equations with a coefficient matrix with positive definite symmetric part. But it does so using a two-term recursion, as opposed to the three-term recursion used by the Generalized Conjugate Gradient method.

Below we present the CGW algorithm. Therein $M$ is again the symmetric part of $A$. Note that this algorithm is identical to the preconditioned CG method, except for the minus sign used in the calculation of $\beta_j$ (see [17], Section 9.2). Further note that when $M = I$ we have $\mathbf{z}_j = \mathbf{r}_j$, which considerably simplifies the algorithm.

**Algorithm 2.2 (CGW)**

*Let* $\boldsymbol{x}_0$ *be given,* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $j = 0$
*Solve* $M\boldsymbol{z}_0 = \boldsymbol{r}_0$, $\boldsymbol{p}_0 = \boldsymbol{z}_0$

*While* not converged *do*

$\alpha_j = (\boldsymbol{r}_j, \boldsymbol{z}_j)/(A\boldsymbol{p}_j, \boldsymbol{z}_j)$
$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j \boldsymbol{p}_j$
$\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A\boldsymbol{p}_j$
*Solve* $M\boldsymbol{z}_{j+1} = \boldsymbol{r}_{j+1}$
$\beta_j = -(\boldsymbol{z}_{j+1}, \boldsymbol{r}_{j+1})/(\boldsymbol{z}_j, \boldsymbol{r}_j)$
$\boldsymbol{p}_{j+1} = \boldsymbol{z}_{j+1} + \beta_j \boldsymbol{p}_j$
$j = j + 1$

*End while*

The CGW method can be used to solve an SSS system (1,2) with $\alpha \neq 0$. For such a system we can force $M = I$ to simplify the algorithm. This is easily done by solving one of the following systems that are equivalent to the SSS system (1,2) when $\alpha \neq 0$:

$$(I + N/\alpha)\mathbf{x} = \mathbf{b}/\alpha \quad \text{or} \quad (I + N/\alpha)(\alpha\mathbf{x}) = \mathbf{b}.$$

The CGW algorithm uses short recurrences, but as it is a Galerkin method (see [11] p. 13) it does not have the optimality property.

## 2.4 Huang, Wathen, Li method

Huang, Wathen and Li [12] described a method to solve the SSS system (1,2) with $\alpha = 0$. We will denote this method by HWL, after the names of the authors. The HWL algorithm actually generates the same approximations to the solution as the CGNR method, as we will prove here. As the CGNR method solves $A^T A\mathbf{x} = A^T\mathbf{b}$, both methods satisfy the optimality property in $\mathcal{K}_j(A^T A, A^T\mathbf{r}_0)$ but generally not in $\mathcal{K}_j(A, \mathbf{r}_0)$, and are more susceptible to rounding errors when dealing with ill-conditioned systems.

Below the CGNR algorithm 2.3 and the HWL algorithm 2.4 are given.

**Algorithm 2.3 (Conjugate Gradient on the Normal Equations)**

*Let* $\boldsymbol{x}_0$ *be given,* $j = 0$
$\boldsymbol{r}_0 = A^T\boldsymbol{b} - A^TA\boldsymbol{x}_0$, $\boldsymbol{p}_0 = \boldsymbol{r}_0$

*While* not converged *do*

$\alpha_j = (\boldsymbol{r}_j, \boldsymbol{r}_j)/(A^TA\boldsymbol{p}_j, \boldsymbol{p}_j) = (\boldsymbol{r}_j, \boldsymbol{r}_j)/(A\boldsymbol{p}_j, A\boldsymbol{p}_j)$
$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \alpha_j\boldsymbol{p}_j$
$\boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \alpha_j A^TA\boldsymbol{p}_j$
$\beta_j = (\boldsymbol{r}_{j+1}, \boldsymbol{r}_{j+1})/(\boldsymbol{r}_j, \boldsymbol{r}_j)$
$\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j\boldsymbol{p}_j$
$j = j + 1$

*End while*

**Algorithm 2.4 (Huang, Wathen, Li)**

*Let* $\tilde{\boldsymbol{x}}_0$ *be given,* $j = 0$
$\tilde{\boldsymbol{r}}_0 = \boldsymbol{b} - A\tilde{\boldsymbol{x}}_0$, $\tilde{\boldsymbol{p}}_0 = A\tilde{\boldsymbol{r}}_0$

*While* not converged *do*

$\tilde{\alpha}_j = (\tilde{\boldsymbol{r}}_j, A\tilde{\boldsymbol{p}}_j)/(A\tilde{\boldsymbol{p}}_j, A\tilde{\boldsymbol{p}}_j)$
$\tilde{\boldsymbol{x}}_{j+1} = \tilde{\boldsymbol{x}}_j + \tilde{\alpha}_j\tilde{\boldsymbol{p}}_j$
$\tilde{\boldsymbol{r}}_{j+1} = \boldsymbol{b} - A\tilde{\boldsymbol{x}}_{j+1}$
$\tilde{\beta}_j = (A^2\tilde{\boldsymbol{p}}_j, A\tilde{\boldsymbol{r}}_{j+1})/(A\tilde{\boldsymbol{p}}_j, A\tilde{\boldsymbol{p}}_j)$
$\tilde{\boldsymbol{p}}_{j+1} = A\tilde{\boldsymbol{r}}_{j+1} + \tilde{\beta}_j\tilde{\boldsymbol{p}}_j$
$j = j + 1$

*End while*

Starting with identical initial solutions we have for $j = 0$:

$$\mathbf{x}_j = \tilde{\mathbf{x}}_j \tag{6}$$
$$\mathbf{r}_j = A^T\tilde{\mathbf{r}}_j = -A\tilde{\mathbf{r}}_j \tag{7}$$
$$\mathbf{p}_j = -\tilde{\mathbf{p}}_j \tag{8}$$

We will prove by induction that these equalities hold for all $j$. For this we will need the following equations from [12]:

$$(\tilde{\mathbf{r}}_j, A\tilde{\mathbf{p}}_i) = 0\,, \quad i \leq j \tag{9}$$
$$(A\tilde{\mathbf{r}}_i, A\tilde{\mathbf{r}}_j) = 0\,, \quad i \neq j \tag{10}$$

We also use the fact that for SSS system (1,2) with $\alpha = 0$ we have $A^T = -A$.

Assume equation (6)–(8) hold for certain $j$. Then:

$$\begin{aligned} \tilde{\alpha}_j &= (\tilde{\mathbf{r}}_j, A\tilde{\mathbf{p}}_j)/(A\tilde{\mathbf{p}}_j, A\tilde{\mathbf{p}}_j) \text{ and} \\ \alpha_j &= (\mathbf{r}_j, \mathbf{r}_j)/(A\mathbf{p}_j, A\mathbf{p}_j) = (A\tilde{\mathbf{r}}_j, A\tilde{\mathbf{r}}_j)/(A\tilde{\mathbf{p}}_j, A\tilde{\mathbf{p}}_j)\,. \end{aligned}$$

If we define $\tilde{\beta}_{-1} = 0$ and $\tilde{\mathbf{p}}_{-1} = 0$ we can write using (9):

$$(\tilde{\mathbf{r}}_j, A\tilde{\mathbf{p}}_j) = (\tilde{\mathbf{r}}_j, A(A\tilde{\mathbf{r}}_j + \tilde{\beta}_{j-1}\tilde{\mathbf{p}}_{j-1})) = (\tilde{\mathbf{r}}_j, AA\tilde{\mathbf{r}}_j) = -(A\tilde{\mathbf{r}}_j, A\tilde{\mathbf{r}}_j)$$

Thus we have $\alpha_j = -\tilde{\alpha}_j$, and with $\mathbf{p}_j = -\tilde{\mathbf{p}}_j$ this gives us $\mathbf{x}_{j+1} = \tilde{\mathbf{x}}_{j+1}$. Writing

$$\tilde{\mathbf{r}}_{j+1} = \mathbf{b} - A\tilde{\mathbf{x}}_{j+1} = \mathbf{b} - A(\tilde{\mathbf{x}}_j + \tilde{\alpha}_j\tilde{\mathbf{p}}_j) = \tilde{\mathbf{r}}_j - \tilde{\alpha}_j A\tilde{\mathbf{p}}_j$$

it also easily follows that:

$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j A^T A\mathbf{p}_j = -(A\tilde{\mathbf{r}}_j - A\tilde{\alpha_j}A\tilde{\mathbf{p}}_j) = -A\tilde{\mathbf{r}}_{j+1}\,.$$

It only remains to prove that given equation (6)–(8) we have $\mathbf{p}_{j+1} = -\tilde{\mathbf{p}}_{j+1}$. Is it easy to see that for this to hold we need to have $\beta_j = \tilde{\beta}_j$.

Using equation (10) we write:

$$\begin{aligned} (\mathbf{r}_{j+1}, \mathbf{r}_{j+1}) &= (\mathbf{r}_j - \alpha_j A^T A\mathbf{p}_j, -A\tilde{\mathbf{r}}_{j+1}) \\ &= (-A\tilde{\mathbf{r}}_j - \tilde{\alpha}_j A^T A\tilde{\mathbf{p}}_j, -A\tilde{\mathbf{r}}_{j+1}) \\ &= (A\tilde{\mathbf{r}}_j - \tilde{\alpha}_j A^2\tilde{\mathbf{p}}_j, A\tilde{\mathbf{r}}_{j+1}) = -\tilde{\alpha}_j(A^2\tilde{\mathbf{p}}_j, A\tilde{\mathbf{r}}_{j+1}). \end{aligned}$$

And using equation (9) we find:

$$\begin{aligned} (\mathbf{r}_j, \mathbf{r}_j) &= (A\tilde{\mathbf{r}}_j, A\tilde{\mathbf{r}}_j) = (\tilde{\mathbf{p}}_j - \tilde{\beta}_{j-1}\tilde{\mathbf{p}}_{j-1}, A\tilde{\mathbf{r}}_j) \\ &= (-A\tilde{\mathbf{p}}_j + \tilde{\beta}_{j-1}A\tilde{\mathbf{p}}_{j-1}, \tilde{\mathbf{r}}_j) = (-A\tilde{\mathbf{p}}_j, \tilde{\mathbf{r}}_j) \\ &= (-A\tilde{\mathbf{p}}_j, \tilde{\mathbf{r}}_{j+1} + \tilde{\alpha}_j A\tilde{\mathbf{p}}_j) = -\tilde{\alpha}_j(A\tilde{\mathbf{p}}_j, A\tilde{\mathbf{p}}_j) \end{aligned}$$

Thus

$$\beta_j = \frac{(\mathbf{r}_{j+1}, \mathbf{r}_{j+1})}{(\mathbf{r}_j, \mathbf{r}_j)} = \frac{-\tilde{\alpha}_j(A^2\tilde{\mathbf{p}}_j, A\tilde{\mathbf{r}}_{j+1})}{-\tilde{\alpha}_j(A\tilde{\mathbf{p}}_j, A\tilde{\mathbf{p}}_j)} = \tilde{\beta}_j\,,$$

which concludes the proof that the HWL algorithm generates the same approximations as the CGNR method. The practical validity of this property is easily confirmed by numerical experiments.

# 3 MRS$^3$ solver

In the previous section we described various existing methods to solve SSS systems (1,2). These methods all have their own drawback. The general methods do not achieve both short recurrences and optimality, and the specialized methods only work for either $\alpha = 0$ or $\alpha \neq 0$.

In this section we will present a solver for SSS systems that satisfies both the short recurrences and the optimality property, and can be used for all values of $\alpha \in \mathbb{R}$. This Minimal Residual method for Shifted Skew-Symmetric systems, or MRS$^3$, is a Krylov subspace method that is based on the Lanczos algorithm [13].

## 3.1 Non-symmetric Lanczos algorithm

We start the derivation of our algorithm with the Lanczos algorithm for non-symmetric matrices $A$.

---

**Algorithm 3.1 (Non-symmetric Lanczos algorithm)**

*Let* $\boldsymbol{q}_0 = \hat{\boldsymbol{q}}_0 = 0\,,\ j = 0$
*Choose* $\boldsymbol{p}_1$, $\hat{\boldsymbol{p}}_1$ *with* $\hat{\boldsymbol{p}}_1^T \boldsymbol{p}_1 \neq 0$

*While* $\hat{\boldsymbol{p}}_{j+1}^T \boldsymbol{p}_{j+1} \neq 0$ *do*

$j = j + 1$
$\alpha_j = \hat{\mathbf{q}}_j^T A \mathbf{q}_j$
*Choose* $\beta_j$ *and* $\gamma_j$ *such that* $\beta_j \gamma_j = \hat{\boldsymbol{p}}_j^T \boldsymbol{p}_j$
$\mathbf{q}_j = \boldsymbol{p}_j / \gamma_j$
$\hat{\mathbf{q}}_j = \hat{\boldsymbol{p}}_j / \beta_j$
$\boldsymbol{p}_{j+1} = A\mathbf{q}_j - \alpha_j \mathbf{q}_j - \beta_j \mathbf{q}_{j-1}$
$\hat{\boldsymbol{p}}_{j+1} = A^T \hat{\mathbf{q}}_j - \alpha_j \hat{\mathbf{q}}_j - \gamma_j \hat{\mathbf{q}}_{j-1}$

*End while*

---

The algorithm is such that the following orthogonality relations hold:

$$\hat{\mathbf{q}}_i^T \mathbf{q}_j = \delta_{ij} = \begin{cases} 1\,, & i = j \\ 0\,, & i \neq j \end{cases} \tag{11}$$

Thus if we define $Q_j = \begin{bmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_j \end{bmatrix}$ and $\hat{Q}_j = \begin{bmatrix} \hat{\mathbf{q}}_1 & \cdots & \hat{\mathbf{q}}_j \end{bmatrix}$ we have

$$\hat{Q}_j^T Q_j = I\,.$$

From the coefficients calculated in the Lanczos algorithm we build the following tridiagonal matrix, that is called the Ritz matrix:

$$T_j = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \gamma_2 & \alpha_2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_j \\ 0 & \cdots & 0 & \gamma_j & \alpha_j \end{bmatrix} \tag{12}$$

Now we can write the recurrences of the Lanczos method in matrix form:

$$\begin{aligned} AQ_j &= Q_j T_j + \mathbf{p}_{j+1}\mathbf{e}_j^T \\ A^T\hat{Q}_j &= \hat{Q}_j T_j^T + \hat{\mathbf{p}}_{j+1}\mathbf{e}_j^T \end{aligned} \tag{13}$$

An important property of the non-symmetric Lanczos algorithm is that the vectors $\mathbf{q}_1, \ldots, \mathbf{q}_j$ form a basis for the Krylov subspace $\mathcal{K}_j(A, \mathbf{q}_1)$ and the vectors $\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_j$ form one for the Krylov subspace $\mathcal{K}_j(A^T, \hat{\mathbf{q}}_1)$. Because of this the algorithm can be used to solve systems of linear equation.

A drawback of the non-symmetric Lanczos algorithm is that it can suffer from breakdown. The algorithm stops if $\hat{\mathbf{p}}_{j+1}^T \mathbf{p}_{j+1} = 0$. If $\mathbf{p}_{j+1} = 0$ the vectors $\mathbf{q}_1, \ldots, \mathbf{q}_j$ span an $A$-invariant subspace, and if $\hat{\mathbf{p}}_{j+1} = 0$ the vectors $\hat{\mathbf{q}}_1, \ldots, \hat{\mathbf{q}}_j$ span an $A^T$-invariant subspace. In either case we say the termination of the algorithm is regular. However if $\mathbf{p}_{j+1} \neq 0$ and $\hat{\mathbf{p}}_{j+1} \neq 0$ at termination, we have a serious breakdown. In this case the algorithm can not be used to generate a full basis for the Krylov subspace.

### 3.2 Shifted skew-symmetric Lanczos algorithm

The original Lanczos algorithm [13] was for symmetric matrices and has some important advantages over the non-symmetric variant presented above. For shifted skew-symmetric matrices a similar advantageous form exists, which we will derive here.

We start from Algorithm 3.1 with $\hat{\mathbf{q}}_0 = \mathbf{q}_0 = 0$, $\mathbf{p}_1$ arbitrary and $\hat{\mathbf{p}}_1 = -\mathbf{p}_1$. Further we choose $\beta_1 = -\gamma_1 = ||\mathbf{p}_1||_2$. We then have $\hat{\mathbf{q}}_1 = \mathbf{q}_1$, and using equation (11) we find:

$$\alpha_1 = \hat{\mathbf{q}}_1^T A \mathbf{q}_1 = \alpha \hat{\mathbf{q}}_1^T \mathbf{q}_1 + \hat{\mathbf{q}}_1^T N \mathbf{q}_1 = \alpha \hat{\mathbf{q}}_1^T \mathbf{q}_1 + \mathbf{q}_1^T N \mathbf{q}_1 = \alpha \,.\, ^{1}$$

Assuming that $\hat{\mathbf{q}}_j = \mathbf{q}_j$, $\hat{\mathbf{q}}_{j-1} = \mathbf{q}_{j-1}$, $\alpha_j = \alpha$ and $\beta_j = -\gamma_j$ we have

$$\mathbf{p}_{j+1} = (\alpha I + N)\mathbf{q}_j - \alpha \mathbf{q}_j - \beta_j \mathbf{q}_{j-1} = N\mathbf{q}_j - \beta_j \mathbf{q}_{j-1} \,,$$

and thus the generated vectors satisfy the following relation:

$$\hat{\mathbf{p}}_{j+1} = (\alpha I + N)^T \hat{\mathbf{q}}_j - \alpha \hat{\mathbf{q}}_j - \gamma_j \hat{\mathbf{q}}_{j-1} = -N\mathbf{q}_j + \beta_j \mathbf{q}_{j-1} = -\mathbf{p}_{j+1}$$

[1] For skew-symmetric $N$ we have $\mathbf{x}^T N \mathbf{x} = (\mathbf{x}^T N \mathbf{x})^T = \mathbf{x}^T N^T \mathbf{x} = -\mathbf{x}^T N \mathbf{x}$ and since $a = -a \Leftrightarrow a = 0$ we find $\mathbf{x}^T N \mathbf{x} = 0$ for all vectors $\mathbf{x}$.

Thus we can take $\beta_{j+1} = -\gamma_{j+1} = ||\mathbf{p}_{j+1}||_2$ which gives us that $\hat{\mathbf{q}}_{j+1} = \mathbf{q}_{j+1}$ and $\alpha_{j+1} = \alpha$. By the induction principle we conclude that, with the above used starting vectors, the choice $\beta_j = -\gamma_j = ||\mathbf{p}_j||_2$ is valid for all $j$ and results in $\hat{\mathbf{q}}_j = \mathbf{q}_j$ for all $j$.

With these results we can justify the following adaptation of the Lanczos algorithm for SSS matrices $A = \alpha I + N$.

---

**Algorithm 3.2 (Shifted skew-symmetric Lanczos algorithm)**

*Let* $\boldsymbol{q}_0 = 0\,,\ j = 0$
*Choose* $\boldsymbol{p}_1$ *with* $\boldsymbol{p}_1 \neq \mathbf{0}$ *and let* $\beta_1 = ||\boldsymbol{p}_1||_2$

*While* $\beta_{j+1} > 0$ *do*

$j = j + 1$
$\boldsymbol{q}_j = -\boldsymbol{p}_j/\beta_j$
$\boldsymbol{p}_{j+1} = N\boldsymbol{q}_j - \beta_j \boldsymbol{q}_{j-1}$
$\beta_{j+1} = ||\boldsymbol{p}_{j+1}||_2$

*End while*

---

Besides the obvious fact that the computation work is greatly reduced with respect to the non-symmetric Lanczos algorithm, the SSS Lanczos algorithm also has the nice property that serious breakdown will (in exact arithmetic) not occur, as $\beta_j = ||\mathbf{p}_{j+1}||_2 = 0 \Leftrightarrow \mathbf{p}_{j+1} = 0$.

Since $\hat{\mathbf{q}}_j = \mathbf{q}_j$ it follows directly from equation (11) that $Q_j$ is orthogonal. Also, the Lanczos recurrences (13) are reduced to the one equation

$$AQ_j = Q_j T_j + \mathbf{p}_{j+1}\mathbf{e}_j^T\,, \tag{14}$$

where $T_j$ is the tridiagonal shifted skew-symmetric Ritz matrix given by:

$$T_j = \begin{bmatrix} \alpha & \beta_2 & 0 & \cdots & 0 \\ -\beta_2 & \alpha & \beta_3 & \ddots & \vdots \\ 0 & -\beta_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_j \\ 0 & \cdots & 0 & -\beta_j & \alpha \end{bmatrix}$$

Using the fact that $\mathbf{p}_{j+1} = -\beta_{j+1}\mathbf{q}_{j+1}$ we can also write the recurrence relation (14) in the following form:

$$AQ_j = Q_{j+1}T_j^+, \tag{15}$$

where the $(j+1)\times j$ extended Ritz matrix $T_j^+$ is defined as:

$$T_j^+ = \begin{bmatrix} & T_j & \\ 0 \cdots 0 & & -\beta_{j+1} \end{bmatrix} = \begin{bmatrix} \alpha & \beta_2 & 0 & \cdots & 0 \\ -\beta_2 & \alpha & \beta_3 & \ddots & \vdots \\ 0 & -\beta_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_j \\ 0 & \cdots & 0 & -\beta_j & \alpha \\ 0 & \cdots & \cdots & 0 & -\beta_{j+1} \end{bmatrix}.$$

### 3.3 Solving shifted skew-symmetric systems

Krylov subspace methods can be categorized by the way the approximation $\mathbf{x}_j = \mathbf{x}_0 + \mathbf{s}_j$ of the solution $\mathbf{x}$ is calculated. Minimal residual methods choose $\mathbf{s}_j$ such that the residual $||\mathbf{r}_j||_2$ is minimized. Orthogonal residual (or Galerkin) methods calculate $\mathbf{s}_j$ such that $Q_j^T \mathbf{r}_j = 0$. We will follow the minimal residual path, because it satisfies the optimality property described in Section 2.

We start the solver with an initial guess $\mathbf{x}_0$. Then in each iteration $j$ we will calculate $\mathbf{s}_j \in \mathcal{K}_j(A, \mathbf{r}_0)$ such that $\mathbf{x}_j = \mathbf{x}_0 + \mathbf{s}_j$ is a good approximation of $\mathbf{x}$. As a measure for the error in $\mathbf{x}_j$ we use $||\mathbf{r}_j||_2$. The vectors generated by the above derived shifted skew-symmetric Lanczos algorithm 3.2 will be used to rewrite the error $||\mathbf{r}_j||_2$ to such a form that we can calculate $\mathbf{s}_j$ from it.

We start Algorithm 3.2 with $\mathbf{p}_1 = \mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$. Then, since the columns of $Q_j$ form a basis for $\mathcal{K}_j(A, \mathbf{q}_1)$ and

$$\mathbf{q}_1 = -\mathbf{p}_1 / ||\mathbf{p}_1||_2 = -\mathbf{r}_0 / ||\mathbf{r}_0||_2 \tag{16}$$

the columns of $Q_j$ also form a basis for the Krylov subspace $\mathcal{K}_j(A, \mathbf{r}_0)$. Therefore for all $\mathbf{s}_j \in \mathcal{K}^j(A, \mathbf{r}_0)$ a $\boldsymbol{\xi}_j \in \mathbb{R}^j$ exists such that

$$\mathbf{s}_j = Q_j \boldsymbol{\xi}_j \,. \tag{17}$$

Thus we can write:

$$||\mathbf{r}_j||_2 = ||\mathbf{b} - A\mathbf{x}_j||_2 = ||\mathbf{b} - A\mathbf{x}_0 - A\mathbf{s}_j||_2 = ||\mathbf{r}_0 - AQ_j\boldsymbol{\xi}_j||_2 \,.$$

Now we use equations (15) and (16) to get:

$$||\mathbf{r}_j||_2 = ||\mathbf{r}_0 - Q_{j+1}T_j^+\boldsymbol{\xi}_j||_2 = ||Q_{j+1}(-||\mathbf{r}_0||_2\mathbf{e}_1 - T_j^+\boldsymbol{\xi}_j)||_2 \,.$$

The matrix $Q_{j+1}$ is orthogonal and the 2-norm is invariant with respect to orthogonal transformations, thus we have

$$||\mathbf{r}_j||_2 = ||\left(||\mathbf{r}_0||_2\mathbf{e}_1 + T_j^+\boldsymbol{\xi}_j\right)||_2 \,. \tag{18}$$

A minimal residual is therefore obtained by choosing $\boldsymbol{\xi}_j = \hat{\boldsymbol{\xi}}_j$ where

$$\hat{\boldsymbol{\xi}}_j = \arg\min_{\boldsymbol{\xi}_j \in \mathbb{R}^j} ||\left(||\mathbf{r}_0||_2\mathbf{e}_1 + T_j^+\boldsymbol{\xi}_j\right)||_2 \,, \tag{19}$$

i.e., $\hat{\boldsymbol{\xi}}_j$ is the least-squares solution of the linear system

$$T_j^+ \boldsymbol{\xi}_j = -||\mathbf{r}_0||_2 \mathbf{e}_1 \,. \tag{20}$$

This least-squares solution can be found with the help of Givens rotations. We will show that it is not necessary to apply rotations to the entire matrix $T_j$ in each iteration. We can store the rotated matrix instead of $T_j$, and update it every iteration.

A Givens rotation of a vector is the multiplication with a square orthogonal matrix of the form

$$G_{\mathbf{y}}(k,l) = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & c & \ddots & \ddots & s & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & -s & \ddots & \ddots & c & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \ddots & \ddots & 0 & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \text{row } k \\ \\ \\ \text{row } l \\ \\ \\ \end{matrix}$$

$$\text{where} \quad c = \frac{\mathbf{y}_k}{\sqrt{\mathbf{y}_k^2 + \mathbf{y}_l^2}} \quad \text{and} \quad s = \frac{\mathbf{y}_l}{\sqrt{\mathbf{y}_k^2 + \mathbf{y}_l^2}} \,.$$

The composition of this matrix is such that if $\tilde{\mathbf{y}} = G_{\mathbf{y}}(k,l)\,\mathbf{y}$, then $\tilde{\mathbf{y}}_i = \mathbf{y}_i$ for all $i \notin \{k,l\}$ and $\tilde{\mathbf{y}}_l = 0$.

We define the following shorthand notation for the Givens rotations we are going to use:

$$G_j^i = G_{\boldsymbol{\tau}_j^i}(i, i+1) \;\;, \quad i = 1, \ldots, j$$

where the transformation vector $\boldsymbol{\tau}_j^i$ is given by

$$\begin{aligned} \boldsymbol{\tau}_j^1 &= \mathbf{t}_j^1 \\ \boldsymbol{\tau}_j^i &= \left( G_j^{i-1} \cdots G_j^1 \right) \mathbf{t}_j^i \;\;, \quad i = 2, \ldots, j \,. \end{aligned}$$

Here $\mathbf{t}_j^i$ denotes column $i$ of the extended Ritz matrix $T_j^+$.

Using these rotations we define the transformed matrix

$$\tilde{U}_j = \left( G_j^j \cdot G_j^{j-1} \cdots G_j^2 \cdot G_j^1 \right) T_j^+ \,. \tag{21}$$

Since all entries of $\mathbf{t}_j^i$ are 0, except those at index $i+1$, $i$ and $i-1$, we have

$$\tilde{\mathbf{u}}_j^i = \left( G_j^j \cdot G_j^{j-1} \cdots G_j^2 \cdot G_j^1 \right) \mathbf{t}_j^i = \left( G_j^i \cdot G_j^{i-1} \cdot G_j^{i-2} \right) \mathbf{t}_j^i \,, \tag{22}$$

where $\tilde{\mathbf{u}}_j^i$ is column $i$ of the matrix $\tilde{U}_j$. Due to the special structure of $\mathbf{t}_j^i$ and the rotations, all entries of $\tilde{\mathbf{u}}_j^i$ are 0, except those at index $i$, $i-1$ and $i-2$, therefore

$$\tilde{\mathbf{u}}_{j+k}^i = \begin{bmatrix} \tilde{\mathbf{u}}_j^i \\ \mathbf{0}_k \end{bmatrix} \;, \; k \geq 0 \,, \tag{23}$$

where $\mathbf{0}_k$ is the $k$-dimensional 0-vector. Note that the vectors have dimension $\dim \tilde{\mathbf{u}}_j^i = \dim \mathbf{t}_j^i = j+1$, and that the last entry of each vector $\tilde{\mathbf{u}}_j^i$ is 0, i.e., $\tilde{\mathbf{u}}_j^i(j+1) = 0$.

From an algorithmic point of view, equation (23) means that in iteration $j > 1$ we can construct $\tilde{\mathbf{u}}_j^i$ for $i < j$ directly from $\tilde{\mathbf{u}}_{j-1}^i$, without having to apply Givens rotations. The only vector that has to be calculated using these rotations is $\tilde{\mathbf{u}}_j^j$. And equation (22) shows that we only need three rotations for this.

From the above results it follows that the matrix $\tilde{U}_j$ is of the form

$$\tilde{U}_j = \begin{bmatrix} U_j \\ 0 \cdots 0 \end{bmatrix}.$$

The matrix $U_j$ can be iteratively constructed using

$$U_j = \begin{bmatrix} U_{j-1} & \mathbf{u}_j^j \\ 0 \cdots 0 & \end{bmatrix}, \tag{24}$$

where the vector $\mathbf{u}_j^j$ is implicitly defined by

$$\tilde{\mathbf{u}}_j^j = \begin{bmatrix} \mathbf{u}_j^j \\ 0 \end{bmatrix}.$$

The matrix $U_j$ is a $j \times j$ matrix with the following sparsity structure:

$$U_j = \begin{bmatrix} * & 0 & * & & & \\ & * & 0 & * & & \\ & & * & 0 & * & \\ & & & * & 0 & * \\ & & & & * & 0 \\ & & & & & * \end{bmatrix}.$$

For proof of this structure, and the precise form of the non-zero elements, see Appendix A.

Now define the transformed vector

$$\tilde{\mathbf{v}}_j = -\left(G_j^j \cdot G_j^{j-1} \cdots G_j^2 \cdot G_j^1\right) ||\mathbf{r}_0||_2 \mathbf{e}_1 . \tag{25}$$

Note that if we define $\tilde{\mathbf{v}}_0 = ||\mathbf{r}_0||_2$ we can write

$$\tilde{\mathbf{v}}_j = G_j^j \begin{bmatrix} \tilde{\mathbf{v}}_{j-1} \\ 0 \end{bmatrix}, \; j > 0 .$$

Writing $\tilde{\mathbf{v}}_j = [\mathbf{v}_j \;\; \varepsilon_j]^T$ we can construct the vector $\tilde{\mathbf{v}}_j$ using

$$\tilde{\mathbf{v}}_j = G_j^j \begin{bmatrix} \mathbf{v}_{j-1} \\ \varepsilon_{j-1} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{j-1} \\ \mu_j \\ \varepsilon_j \end{bmatrix},$$

where the values of $\mu_j$ and $\varepsilon_j$ are determined by the Givens rotation $G_j^j$. Thus the vector $\mathbf{v}_j$ can be iteratively constructed using

$$\mathbf{v}_j = \begin{bmatrix} \mathbf{v}_{j-1} \\ \mu_j \end{bmatrix} . \tag{26}$$

Since a Givens rotation is an orthogonal transformation, using equation (18) and definitions (21) and (25) we find

$$||\mathbf{r}_j||_2 = || \, (T_j^+ \boldsymbol{\xi}_j + ||\mathbf{r}_0||_2 \mathbf{e}_1) \, ||_2 = || \, \tilde{U}_j \boldsymbol{\xi}_j - \tilde{\mathbf{v}}_j \, ||_2 \, .$$

Thus the solution $\hat{\boldsymbol{\xi}}_j$ of equation (19) is equal to the least-squares solution of the system

$$\tilde{U}_j \boldsymbol{\xi}_j = \tilde{\mathbf{v}}_j \Rightarrow \begin{bmatrix} U_j \\ 0 \cdots 0 \end{bmatrix} \boldsymbol{\xi}_j = \begin{bmatrix} \mathbf{v}_j \\ \varepsilon_j \end{bmatrix} .$$

From this result it is trivial that $\hat{\boldsymbol{\xi}}_j$ is equal to the solution of the system

$$U_j \boldsymbol{\xi}_j = \mathbf{v}_j \, , \tag{27}$$

and that the residual error is given by

$$||\mathbf{r}_j||_2 = \varepsilon_j \, . \tag{28}$$

To determine the minimal residual approximation $\mathbf{x}_j$ we need to calculate $\mathbf{s}_j$ from equation (17). If we calculate $\hat{\boldsymbol{\xi}}_j$ as the solution of system (27) and then multiply with $Q_j$ directly, we would need to store the entire matrix $Q_j$ in memory. Thus the algorithm would not yield the short recurrences property described in Section 2.

To overcome this problem we can use the technique that is also applied in the MINRES algorithm (see [14]). We define the matrix $W_j = Q_j U_j^{-1}$, then:

$$W_j U_j = Q_j \tag{29}$$

and

$$\mathbf{s}_j = Q_j U_j^{-1} \mathbf{v}_j = W_j \mathbf{v}_j \, . \tag{30}$$

Further we introduce the following notation:

$$W_j = \begin{bmatrix} \mathbf{w}_j^1 \cdots \mathbf{w}_j^j \end{bmatrix} \quad \text{and} \quad W_j^i = \begin{bmatrix} \mathbf{w}_j^1 \cdots \mathbf{w}_j^i \end{bmatrix} .$$

For $j = 1$ equation (29) has a unique solution that is easily determined:

$$\mathbf{w}_1^1 u_{1,1} = \mathbf{q}_1 \Rightarrow \mathbf{w}_1^1 = \frac{1}{u_{1,1}} \mathbf{q}_1 \, .$$

Now suppose that $j = i$ with $i > 1$, and that we have a unique solution of equation (29) for $j = i - 1$, then the equation is

$$\begin{bmatrix} W_i^{i-1} & \mathbf{w}_i^i \end{bmatrix} \begin{bmatrix} U_{i-1} & \mathbf{u}_i^i \\ 0 \cdots 0 & \end{bmatrix} = \begin{bmatrix} Q_{i-1} & \mathbf{q}_i \end{bmatrix}$$

which can be split in the equations

$$W_i^{i-1} U_{i-1} = Q_{i-1} \tag{31}$$
$$W_i \mathbf{u}_i^i = \mathbf{q}_i \,. \tag{32}$$

Due to our assumption equation (31) has the unique solution

$$W_i^{i-1} = W_{i-1} \,,$$

and due to the special structure of $\mathbf{u}_i^i$ equation (32) is uniquely solved by

$$\begin{aligned} \mathbf{w}_2^2 &= \frac{1}{u_{2,2}} \mathbf{q}_2 \\ \mathbf{w}_i^i &= \frac{1}{u_{i,i}} \left( \mathbf{q}_i - u_{i-2,i} \mathbf{w}_i^{i-2} \right) \,, \quad i > 2 \,. \end{aligned}$$

Thus it is proved by induction that we can unambiguously define $\mathbf{w}_i = \mathbf{w}_j^i$, and that equation (29) is uniquely solved by the matrix $W_j$ with columns

$$\mathbf{w}_i = \begin{cases} \dfrac{1}{u_{i,i}} \mathbf{q}_i \,, & i \in \{1, 2\} \\ \dfrac{1}{u_{i,i}} \left( \mathbf{q}_i - u_{i-2,i} \mathbf{w}_{i-2} \right) \,, & i \in \{2, j\} \end{cases} \tag{33}$$

This solution uses only short recurrences, as required.

The final step is to find the approximating solution $\mathbf{x}_j$. Using equations (26) and (30) we can write

$$\mathbf{s}_j = W_j \mathbf{v}_j = W_{j-1} \mathbf{v}_{j-1} + \mu_j \mathbf{w}_j = \mathbf{s}_{j-1} + \mu_j \mathbf{w}_j \,.$$

Thus the approximation of the solution in iteration $j$ is given by

$$\mathbf{x}_j = \mathbf{x}_0 + \mathbf{s}_j = \mathbf{x}_{j-1} + \mu_j \mathbf{w}_j \,. \tag{34}$$

Combining all the above results, we now present the MRS$^3$ solver Algorithm 3.3. To make the algorithm easier to read we have used the following simplified notations: $\mathbf{u}_j$ for $\tilde{\mathbf{u}}_j$, $\mathbf{v}_j$ for $\tilde{\mathbf{v}}_j$ and $G_j$ for $G_j^j$.

Note that in this version of the algorithm we use Givens rotations to calculate the vectors $\mathbf{u}_j$. Another option would be to use the knowledge of these vectors derived in Appendix A.

**Algorithm 3.3 (MRS$^3$)**

*Choose* $\boldsymbol{x}_0$ *and set the residual error tolerance* $\tau$
*Let* $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$, $N = A - \alpha I$
*Let* $j = 0$, $\boldsymbol{q}_0 = 0$, $\boldsymbol{p}_1 = \boldsymbol{r}_0$, $\beta_1 = ||\boldsymbol{p}_1||_2$, $\varepsilon = \beta_1$

*While* $\beta_{j+1} > 0$ *and* $\varepsilon > \tau$ *do*

$j = j + 1$
$\boldsymbol{q}_j = -\boldsymbol{p}_j/\beta_j$
$\boldsymbol{p}_{j+1} = N\boldsymbol{q}_j - \beta_j \boldsymbol{q}_{j-1}$
$\beta_{j+1} = ||\boldsymbol{p}_{j+1}||_2$

*If* $j = 1$
$\boldsymbol{u}_j = [\,\alpha \quad -\beta_{j+1}\,]^T$
$\boldsymbol{v}_j = [\,-||\boldsymbol{r}_0|| \quad 0\,]^T$
$G_j = GivensRotation(\boldsymbol{u}_j, j, j+1)$
$\boldsymbol{u}_j = G_j \boldsymbol{u}_j$
$\boldsymbol{v}_j = G_j \boldsymbol{v}_j$
$\boldsymbol{w}_j = \boldsymbol{q}_j/\boldsymbol{u}_j(j)$
*Elseif* $j = 2$
$\boldsymbol{u}_j = [\,\beta_j \quad \alpha \quad -\beta_{j+1}\,]^T$
$\boldsymbol{v}_j = [\,\boldsymbol{v}_{j-1} \quad 0\,]^T$
$G_j = GivensRotation(\boldsymbol{u}_j, j, j+1)$
$\boldsymbol{u}_j = G_j G_{j-1} \boldsymbol{u}_j$
$\boldsymbol{v}_j = G_j \boldsymbol{v}_j$
$\boldsymbol{w} = \boldsymbol{q}_j/\boldsymbol{u}_j(j)$
*Else*
$\boldsymbol{u}_j = [\,\boldsymbol{0}_{j-2} \quad \beta_j \quad \alpha \quad -\beta_{j+1}\,]^T$
$\boldsymbol{v}_j = [\,\boldsymbol{v}_{j-1} \quad 0\,]^T$
$G_j = GivensRotation(\boldsymbol{u}_j, j, j+1)$
$\boldsymbol{u}_j = G_j G_{j-1} G_{j-2} \boldsymbol{u}_j$
$\boldsymbol{v}_j = G_j \boldsymbol{v}_j$
$\boldsymbol{w}_j = (\boldsymbol{q}_j - \boldsymbol{u}_j(j-2)\boldsymbol{w}_{j-2})/\boldsymbol{u}_j(j)$
*Endif*
$\boldsymbol{x}_j = \boldsymbol{x}_{j-1} + \boldsymbol{v}_j(j)\boldsymbol{w}_j$
$\varepsilon = \boldsymbol{v}_j(j+1)$

*End while*

# 4 Comparison of MRS$^3$ with existing methods

Like GMRES, the MRS$^3$ method computes the minimal residual using the Krylov subspace $\mathcal{K}_j(A, \mathbf{r}_0)$. Assuming that $A$ is non-singular, this implies that in exact arithmetic both methods generate the same approximation to the solution in each iteration, i.e.,

$$x_j^{\mathrm{MRS}^3} = x_j^{GMRES}\,.$$

As MRS$^3$ uses short recurrences, it is more efficient in finding this solution than GMRES.

The relation between the residual of a minimal residual method MR and that of a Galerkin method G, is known from (2.29) of [10]. If the minimal residual method does not stagnate, i.e., if

$$c = \frac{\|r_j^{\mathrm{MR}}\|_2}{\|r_{j-1}^{\mathrm{MR}}\|_2} < 1\,,$$

then the norm of the residuals satisfy the identity

$$\|r_j^{\mathrm{MR}}\|_2 = \sqrt{1-c^2}\|r_j^{\mathrm{G}}\|_2\,. \tag{35}$$

It follows directly that, if the MRS$^3$ method does not stagnate, the calculated residuals are always smaller than those of Galerkin methods like CGW, so

$$\|r_j^{\mathrm{MRS}^3}\|_2 < \|r_j^{CGW}\|_2\,.$$

Furthermore, relation (35) can be used to understand the so-called peak-plateau connection [3, 21, 23]. The peak-plateau connection is the phenomenon that a peak in the residual norm history of a Galerkin method is accompanied by a plateau, i.e., the norm nearly stagnating, in the residual norm history of a minimal residual method.

In Table 1 we have set out some important properties of the computational load, as well as a couple general properties of MRS$^3$ and other algorithms treated in the previous chapters.

| | matvec products | inner products | vector updates | vector memory | $\alpha$ | optimality |
|---|---|---|---|---|---|---|
| MRS$^3$ | 1 | 1 | 3 | 6 | all | yes |
| CGW | 1 | 4 | 3 | 4 | $\neq 0$ | no |
| GMRES | 1 | $\frac{k+1}{2}$ | $\frac{k+1}{2}$ | k+3 | all | yes |
| Bi-CGSTAB | 2 | 3 | 3 | 7 | all | no |
| CGNR | 2 | 2 | 2 | 4 | all | no |

Table 1: Important properties of MRS$^3$ and other solvers

Note that these numbers can differ with the exact implementation of the algorithm. For the CGW method we based the numbers on an implementation that solves $(I + N/\alpha)\mathbf{x} = (\mathbf{b}/\alpha)$, as this is a lot more efficient. Further note that with optimality we mean that the method satisfies the optimality property within the Krylov subspace $\mathcal{K}_j(A, \mathbf{r}_0)$.

# 5 Numerical results

In this section we will compare the MRS$^3$ algorithm numerically with the CGW, GMRES, Bi-CGSTAB and CGNR methods. Also we will numerically verify the theoretical results from Section 4 for MRS$^3$ versus the CGW algorithm.

As we mentioned in the introduction of this paper, SSS systems frequently occur in the solution of advection-diffusion problems. Therefore we will use matrices of a form found in this field of expertise to numerically explore the MRS$^3$ method. The matrices we will consider are of the form

$$A = \alpha I + N\,,$$

where $A \in \mathbb{R}^{n\times n}$, $n = n_1 \cdot n_2$, $h_1 = \frac{1}{n_1}$, $h_2 = \frac{1}{n_2}$, and with the matrix $N$ a skew-symmetric block tridiagonal matrix of which the nonzero blocks are given by

$$S_{ii} = \frac{1}{2h_1} tridiag(-1, 0, 1), \text{ for } i = 1, \ldots, n_2\,,$$

and

$$S_{i,i+1} = \frac{1}{2h_2} diag(\gamma), \text{ for } i = 1, \ldots, n_2 - 1\,,$$

for varying values of $\gamma$.

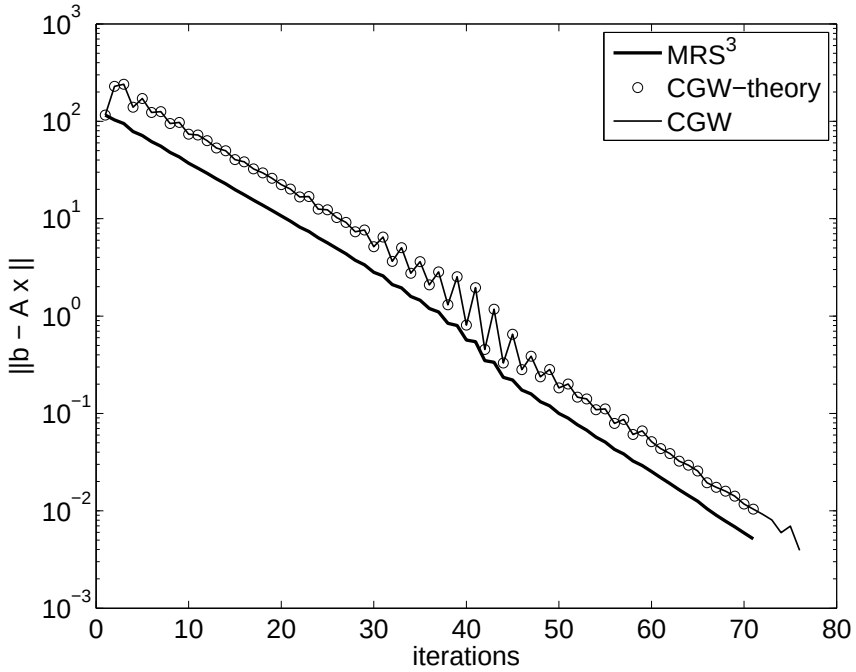


Figure 1: Convergence of MRS$^3$ and CGW with $\alpha = 5$, and $\gamma = 1$

### 5.1 Numerical comparison of MRS$^3$ and CGW

As noted in Section 2.3, the CGW method can only be used if $\alpha \neq 0$. We choose $n_1 = n_2 = 20$, $\alpha = 5$, and $\gamma = 1$ and compare the norm of the residuals of the MRS$^3$ and CGW methods. Furthermore we check the identity of equation (35) by plotting the expected residual norm for CGW based on the computed residual norm for MRS$^3$, and comparing it to the computed residual norm for the CGW algorithm. The methods are stopped when $\frac{\|r_j\|_2}{\|r_0\|_2} < 5 \cdot 10^{-5}$.

The results are given in Figure 1. Note that the theoretical prediction for the norm of the CGW residual is correct. Further note that this figure nicely illustrated the peak-plateau connection mentioned in Section 4.

### 5.2 Numerical comparison of MRS$^3$, GMRES and Bi-CGSTAB

In this section we compare the MRS$^3$ algorithm with the general Krylov methods GMRES and Bi-CGSTAB. In order to make the memory and work requirements comparable we use GMRES(3), which means that GMRES is restarted every 3 iterations. In our experiments we have also checked that full GMRES indeed leads to the same results as MRS$^3$.

The results of our experiments are given in Figure 2, 3, and 4. It appears that all methods converge well if $\alpha$ is large. However for small $\alpha$, or $\alpha = 0$, only the MRS$^3$ method leads to acceptable results. With respect to Bi-CGSTAB, we should note that for these problems $\mathrm{Bi-CGSTAB}(l)$ [19] may be a better alternative.

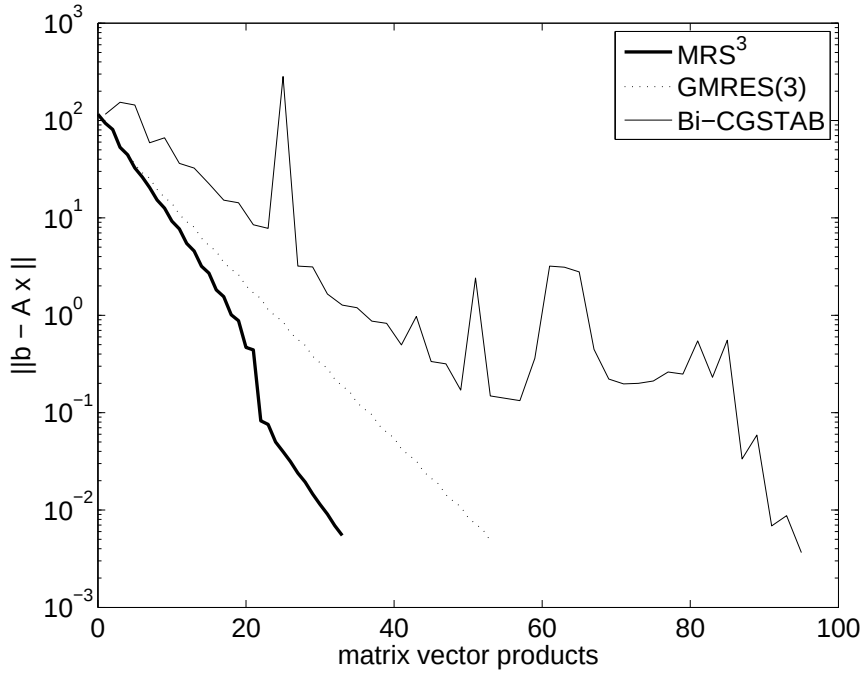


Figure 2: Convergence of MRS$^3$, GMRES and Bi-CGSTAB with $\alpha = 500$, and $\gamma = 100$

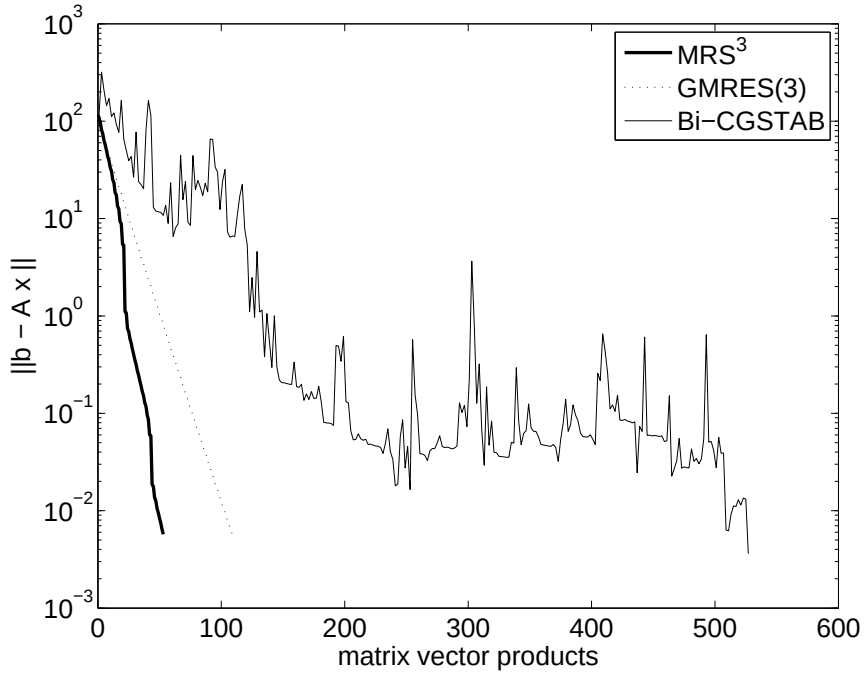


Figure 3: Convergence of MRS$^3$, GMRES and Bi-CGSTAB with $\alpha = 250$, and $\gamma = 100$

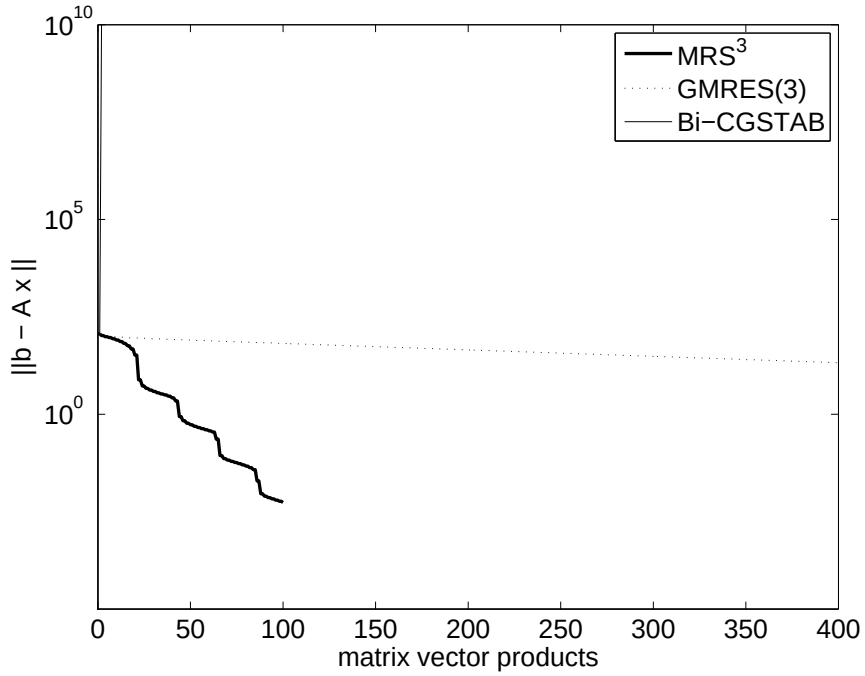


Figure 4: Convergence of MRS$^3$, GMRES and Bi-CGSTAB with $\alpha = 0$, and $\gamma = 100$

### 5.3 Numerical comparison of MRS$^3$ and CGNR

Finally, we compare the MRS$^3$ and CGNR methods. The CGNR method is used in [8] to solve SSS systems. It appears, that if the system is well conditioned the required number of matrix vector products of both methods are comparable, however if the system is ill conditioned the CGNR method breaks down. This is illustrated by the results given in Table 2, where $\gamma = 1$.

| | condition | matrix vector products | |
|---|---|---|---|
| $\alpha$ | number | MRS$^3$ | CGNR |
| 1 | $3.9 \cdot 10$ | 226 | 245 |
| $10^{-4}$ | $3.9 \cdot 10^5$ | 312 | 337 |
| $10^{-8}$ | $3.9 \cdot 10^9$ | 328 | 521 |
| $10^{-12}$ | $3.9 \cdot 10^{13}$ | 655 | n.c. |

Table 2: Numerical results for MRS$^3$ and CGNR

## 6 Conclusions

This paper is started by showing the importance of a fast solver for shifted skew-symmetric matrix systems. Theory by Faber and Manteuffel [5], [6] demonstrates that an algorithm that is optimal and uses short recurrences should exist, however such an algorithm was not yet available. In Section 3 we have presented such an algorithm, the MRS$^3$ (Minimal Residual Method for Shifted Skew-Symmetric Systems) solver.

By theory and numerical experiments in Section 4 and 5, we have shown that the MRS$^3$ method generally outperforms its common alternatives. As a minimal residual method it converges faster than Galerkin methods, like the CGW algorithm, that do not satisfy the optimality property, while at the same time also allowing $\alpha = 0$ where CGW does not work. Full GMRES converges as fast as MRS$^3$ but is not a valid option due to its complexity, whereas restarted GMRES variants have a good complexity but cannot maintain the fast convergence. For the specific problem of SSS (Shifted Skew-Symmetric) systems Bi-CGSTAB seems to have convergence problems, especially for small $\alpha$, while the complexity is also worse than that of MRS$^3$. And though the performance of the CGNR method is comparable to that of MRS$^3$ for many problems, it breaks down for small $\alpha$.

We conclude that the proposed MRS$^3$ solver performs very well for the important class of shifted skew-symmetric matrix systems $A = \alpha I + N$. The complexity of the algorithm is very good, in generally it converges very fast, and it can be used for all values of $\alpha$. Especially for small $\alpha$, or $\alpha = 0$, MRS$^3$ seems to perform much better than the well known alternatives.

# A Structure of the rotated Ritz matrix

In this appendix we will look in-depth at the rotated Ritz matrix $U_j$ defined in Section 3.3. To this end we define

$$\begin{aligned} Z_1 &= \alpha^2\,, \\ Z_2 &= Z_1 + \beta_2^2\,, \\ Z_3 &= Z_2 + \beta_3^2\,, \\ Z_i &= \tfrac{Z_1 Z_3 \cdots Z_{i-1}}{Z_2 Z_4 \cdots Z_{i-2}} + \beta_i^2\,, \quad i > 3\,,\ i \text{ even}\,, \\ Z_i &= \tfrac{Z_2 Z_4 \cdots Z_{i-1}}{Z_3 Z_5 \cdots Z_{i-2}} + \beta_i^2\,, \quad i > 3\,,\ i \text{ odd}\,. \end{aligned}$$

Let, as in Section 3.3,

$$G_j^i = G_{\boldsymbol{\tau}_j^i}\,(i, i+1)\ \ , \quad j \geq i$$

denote Givens rotation $i$ at iteration $j$, and let

$$\begin{aligned} c_i &= \tfrac{\boldsymbol{\tau}_j^i(i)}{\sqrt{\boldsymbol{\tau}_j^i(i) + \boldsymbol{\tau}_j^i(i+1)}} \\ s_i &= \tfrac{\boldsymbol{\tau}_j^i(i+1)}{\sqrt{\boldsymbol{\tau}_j^i(i) + \boldsymbol{\tau}_j^i(i+1)}} \end{aligned}$$

denote the coefficients of $G_j^i$. Note that for $j \geq i$ the values of $c_i$ and $s_i$ are indeed independent of $j$ due to the special structure of $\boldsymbol{\tau}_j^i$.

For $j = 1$ we have $\boldsymbol{\tau}_1^1 = \mathrm{t}_1^1$ and find

$$c_1 = \frac{\sqrt{Z_1}}{\sqrt{Z_2}}\ , \quad s_1 = \frac{-\beta_2}{\sqrt{Z_2}}\,,$$

$$\mathbf{u}_1^1 = G_1^1 \boldsymbol{\tau}_1^1 = G_1^1 \mathrm{t}_1^1 = \begin{bmatrix} \alpha c_1 - \beta_2 s_1 \\ -\alpha s_1 - \beta_2 c_1 \end{bmatrix} = \begin{bmatrix} \sqrt{Z_2} \\ 0 \end{bmatrix}.$$

For $j = 2$ we have

$$\boldsymbol{\tau}_2^2 = G_2^1 \mathrm{t}_2^2 = \begin{bmatrix} \beta_2 c_1 + \alpha s_1 \\ -\beta_2 s_1 + \alpha c_1 \\ -\beta_3 \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{Z_2} \\ -\beta_3 \end{bmatrix}.$$

Thus we find

$$c_2 = \frac{\sqrt{Z_2}}{\sqrt{Z_3}}\ , \quad s_2 = \frac{-\beta_3}{\sqrt{Z_3}}\,,$$

$$\mathbf{u}_2^2 = G_2^2 \boldsymbol{\tau}_2^2 = \begin{bmatrix} 0 \\ \sqrt{Z_2} c_2 - \beta_3 s_2 \\ -\sqrt{Z_2} s_2 - \beta_3 c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{Z_3} \\ 0 \end{bmatrix}.$$

Now assume that

$$\begin{aligned} c_i &= \sqrt{\tfrac{Z_2 Z_4 \cdots Z_i}{Z_3 Z_5 \cdots Z_{i+1}}}\,, \quad i > 2\,,\ i \text{ even}\,, \\ c_i &= \sqrt{\tfrac{Z_1 Z_3 \cdots Z_i}{Z_2 Z_4 \cdots Z_{i+1}}}\,, \quad i > 2\,,\ i \text{ odd}\,, \\ s_i &= \tfrac{-\beta_{i+1}}{\sqrt{Z_{i+1}}}\,. \end{aligned} \tag{36}$$

Obviously this is true for $i = 1, 2$. We will show by induction that it holds for all $i > 0$.

For $j > 2$ we have

$$\boldsymbol{\tau}_j^j = G_j^{j-1} G_j^{j-2} \mathrm{t}_j^j = G_j^{j-1} \begin{bmatrix} \mathbf{0}_{j-3} \\ \beta_j s_{j-2} \\ \beta_j c_{j-2} \\ \alpha \\ -\beta_{j+1} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{j-3} \\ \beta_j s_{j-2} \\ \beta_j c_{j-2} c_{j-1} + \alpha s_{j-1} \\ -\beta_j c_{j-2} s_{j-1} + \alpha c_{j-1} \\ -\beta_{j+1} \end{bmatrix},$$

where $\mathbf{0}_i$ denotes the null vector with dimension $i$.

Suppose that $j$ is even. Using (36) we then find

$$\begin{aligned} &\beta_j c_{j-2} c_{j-1} + \alpha s_{j-1} && = \\ &\beta_j \sqrt{\tfrac{Z_2 Z_4 \cdots Z_{j-2}}{Z_3 Z_5 \cdots Z_{j-1}}} \sqrt{\tfrac{Z_1 Z_3 \cdots Z_{j-1}}{Z_2 Z_4 \cdots Z_j}} + \alpha \tfrac{-\beta_j}{\sqrt{Z_j}} && = \\ &\beta_j \sqrt{\tfrac{Z_1}{Z_j}} + \alpha \tfrac{-\beta_j}{\sqrt{Z_j}} && = \quad 0\,. \end{aligned}$$

Further we can write

$$\begin{aligned} &-\beta_j c_{j-2} s_{j-1} + \alpha c_{j-1} && = \\ &-\beta_j \sqrt{\tfrac{Z_2 Z_4 \cdots Z_{j-2}}{Z_3 Z_5 \cdots Z_{j-1}}} \tfrac{-\beta_j}{\sqrt{Z_j}} + \alpha \sqrt{\tfrac{Z_1 Z_3 \cdots Z_{j-1}}{Z_2 Z_4 \cdots Z_j}} && = \\ &\alpha \sqrt{\tfrac{Z_1 Z_3 \cdots Z_{j-1}}{Z_2 Z_4 \cdots Z_j}} + \tfrac{\beta_j^2}{\sqrt{Z_j}} \sqrt{\tfrac{Z_2 Z_4 \cdots Z_{j-2}}{Z_3 Z_5 \cdots Z_{j-1}}} && = \\ &\tfrac{Z_1}{\sqrt{Z_j}} \sqrt{\tfrac{Z_3 \cdots Z_{j-1}}{Z_2 Z_4 \cdots Z_j - 2}} + \tfrac{\beta_j^2}{\sqrt{Z_j}} \sqrt{\tfrac{Z_2 Z_4 \cdots Z_{j-2}}{Z_3 Z_5 \cdots Z_{j-1}}} && = \\ &\sqrt{\tfrac{Z_2 Z_4 \cdots Z_{j-2}}{Z_3 Z_5 \cdots Z_{j-1}}} \left( \tfrac{\frac{Z_1 Z_3 \cdots Z_{j-1}}{Z_2 Z_4 \cdots Z_j - 2}}{\sqrt{Z_j}} + \tfrac{\beta_j^2}{\sqrt{Z_j}} \right) && = \\ &\sqrt{\tfrac{Z_2 Z_4 \cdots Z_{j-2}}{Z_3 Z_5 \cdots Z_{j-1}}} \sqrt{Z_j} && = \quad \sqrt{\tfrac{Z_2 Z_4 \cdots Z_j}{Z_3 Z_5 \cdots Z_{j-1}}}\,. \end{aligned}$$

Therefore, assuming (36) for $i = 1 \ldots j-1$, for even $j > 2$ we have

$$\boldsymbol{\tau}_j^j = \begin{bmatrix} \mathbf{0}_{j-3} \\ \frac{-\beta_{j-1}\beta_j}{Z_{j-1}} \\ 0 \\ \sqrt{\frac{Z_2 Z_4 \cdots Z_j}{Z_3 Z_5 \cdots Z_{j-1}}} \\ -\beta_{j+1} \end{bmatrix}.$$

From this it easily follows that indeed for $j$, again assumption (36) holds, and that

$$\mathbf{u}_j^j = G_j^j \boldsymbol{\tau}_j^j = \begin{bmatrix} \mathbf{0}_{j-3} \\ \frac{-\beta_{j-1}\beta_j}{Z_{j-1}} \\ 0 \\ \sqrt{\frac{Z_2 Z_4 \cdots Z_j}{Z_3 Z_5 \cdots Z_{j-1}}} c_j - \beta_{j+1} s_j \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{j-3} \\ \frac{-\beta_{j-1}\beta_j}{Z_{j-1}} \\ 0 \\ \sqrt{Z_{j+1}} \\ 0 \end{bmatrix}.$$

In the same way we can prove that for odd $j > 2$ we have

$$\boldsymbol{\tau}_j^j = \begin{bmatrix} \mathbf{0}_{j-3} \\ \frac{-\beta_{j-1}\beta_j}{Z_{j-1}} \\ 0 \\ \sqrt{\frac{Z_1 Z_3 \cdots Z_j}{Z_2 Z_4 \cdots Z_{j-1}}} \\ -\beta_{j+1} \end{bmatrix},$$

that thus assumption (36) holds for all $j > 0$, and that for odd $j > 2$ again

$$\mathbf{u}_j^j = \begin{bmatrix} \mathbf{0}_{j-3} \\ \frac{-\beta_{j-1}\beta_j}{Z_{j-1}} \\ 0 \\ \sqrt{Z_{j+1}} \\ 0 \end{bmatrix}.$$

## References


[1] Z. Bai, G.H. Golub, and M.K. Ng. Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. *SIAM J. Matrix Anal. Appl.*, 24:603–626 (electronic), 2003.

[2] P. Concus and G.H. Golub. A generalized conjugate gradient method for nonsymmetric systems of linear equations. In R. Glowinski and J.L. Lions, editors, *Lecture Notes in Economics and Mathematical Systems*, volume 134, pages 50–56. Springer-Verlag, Berlin, 1967.

[3] J.K. Cullum. Peaks, plateaus, numerical instabilities in a Galerkin minimal residual pair of methods for solving $Ax = b$. *Appl. Numer. Math.*, 19:255–278, 1995.

[4] S.C. Eisenstat. A note on the generalized conjugate gradient method. *SIAM J. Numer. Anal.*, 20:358–361, 1983.

[5] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21:352–362, 1984.

[6] V. Faber and T. Manteuffel. Orthogonal error methods. *SIAM J. Numer. Anal.*, 24:170–187, 1987.

[7] G.H. Golub and D. Vanderstraeten. On the preconditioning of matrices with skew-symmetric splittings. *Numerical Algorithms*, 25:223–239, 2000.

[8] G.H. Golub and D. Vanderstraeten. On the preconditioning of matrices with skew-symmetric splittings. *Numerical Algorithms*, 25:223–239, 2003.

[9] G.H. Golub and A.W. Wathen. An iteration for indefinite systems and its application to the Navier-Stokes equations. *SIAM J. Sci. Comput.*, 19:530–539, 1998.

[10] M.H. Gutknecht and M. Rozložník. By how much can residual minimization accelerate the convergence of orthogonal residual methods? *Numer. Algorithms*, 27:189–213, 2001.

[11] M.H. Gutknecht and M. Rozložník. A framework for generalized conjugate gradient methods—with special emphasis on contributions by Rüdiger Weiss. *Appl. Numer. Math.*, 41:7–22, 2002.

[12] Y. Huang, A.J. Wathen, and L. Li. An iterative method for skew-symmetric systems. *Information*, 2:147–153, 1999.

[13] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand.*, 45:255–282, 1950.

[14] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[15] C.C. Paige and M.A. Saunders. LSQR : An algorithm for sparse linear equations and sparse least squares. *A.C.M. Trans. Math. Softw.*, 8:43–71, 1982.

[16] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization: An Interior Point Approach*. John Wiley & Sons, New York, 1997.

[17] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition, 2003.

[18] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[19] G.L.G. Sleijpen, H.A. van der Vorst, and D.R. Fokkema. BiCGstab($l$) and other hybrid Bi-CG methods. *Numer. Algorithms*, 7:75–109, 1994.

[20] H.A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.

[21] H.A. van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *J. Comp. Appl. Math.*, 48:327–341, 1993.

[22] H.A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *Num. Lin. Alg. Appl.*, 1:369–386, 1994.

[23] H.F. Walker. Residual smoothing and peak/plateau behavior in Krylov subspace methods. *Appl. Numer. Math.*, 19:279–286, 1995.

[24] O. Widlund. A Lanczos method for a class of nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 15:801–812, 1978.