Closer to the solution: restarted GMRES with adaptive preconditioning

H.T. Stoppels

July 17, 2014

Abstract

The generalized minimum residual (GMRES) method proposed by Saad and Schultz in 1986 is one of the most popular iterative algorithms for solving systems of linear equations $Ax = b$ for non-hermitian matrices A. The method computes the optimal approximation for which the 2-norm of the residual is minimal over a corresponding Krylov subspace. One drawback of the conventional GMRES method is its linearly increasing costs per iterations, which may become prohibitively large in practical applications.

Ordinary restarted GMRES overcomes this difficulty at the cost of loss of the optimality property. This thesis provides variants of restarted GMRES that try to recover the optimality property by recycling approximate spectral information gathered during the iterations, to build adaptive preconditioners. This may result in better clustering of the eigenvalues around point one of the spectrum, and consequently less iteration steps than the ordinary restarted GMRES method. The theoretical background of the the new family of solvers is presented, and its numerical properties are illustrated by Matlab experiments on realistic matrix problems arising from different fields of application.

Contents

1 Introduction

In this thesis we are interested in solving linear systems of the form

$$Ax = b \tag{1.1}$$

where $A \in \mathbb{C}^{n \times n}$ is the coefficient matrix which is a large general matrix, $x \in \mathbb{C}^n$ is the unknown solution vector and $b \in \mathbb{C}^n$ is the right-hand side vector. These linear systems typically arise from the discretization of Partial Differential Equations (PDEs) and Integral Equations (IEs). In the case of PDEs where finite difference, finite elements or finite volume methods are applied, the coefficient matrix is typically sparse, while in the other case of IEs using the boundary element method, the coefficient matrix is typically dense.

Direct methods like Gaussian elimination with pivoting are seen as the most robust solvers for general linear systems as they obtain the exact solution for general matrices, but can only be used for moderately sized systems as their memory usage is high. Especially in the case of sparse matrices, Gaussian elimination produces fill-in which may destroy the sparsity, resulting in storage costs of $\mathcal{O}(n^2)$.

Iterative methods on the other hand use low memory — they typically only store A, b and few additional vectors — to approximate the solution of a linear system, but are not as robust as direct methods in the sense that they do not always converge for general matrices. Some iterative methods exploit properties of the coefficient matrix, such as symmetricity and positive definiteness in the case of the Conjugate Gradient method, and in practice packages of iterative solvers first identify the coefficient matrix to see if it matches the criteria for a special-purpose solver.

However, when A does not fit into a special category of matrices and no special-purpose solvers exists, the linear system is more challenging and one must use black-box solvers. Development of efficient solvers of this type is crucial, as large general matrices arise in a variety of applications.

GMRES, developed by Saad and Schultz in the late 1980s is one of the most popular solvers for general linear systems. In this thesis we will inspect its properties and propose a family of variants of GMRES, which exploit by-products of the algorithm to improve convergence.

The outline of the thesis is as follows: first iterative methods for linear systems are considered in general, where classical iterative methods are used to introduce the Krylov subspace. In section 3 an overview is given of different Krylov subspace methods. Subsequently, section 4 gives insight into the GRMES algorithm, its convergence properties and the need for restarted GMRES. With the convergence bounds in mind, section 5 introduces the concept of preconditioning, which is a technique applied prior to and independent from the solving phase to improve convergence. Section 5 introduces most importantly modern spectral preconditioning techniques, which apply specifically to Krylov subspace methods to improve an initial preconditioner using spectral information, resulting in a better rate of convergence. Finally, these spectral preconditioners are used to adaptively update an initial preconditioner from by-products during the GMRES

algorithm, giving rise to a family of novel GMRES variants. These new methods are tested and compared to GMRES-DR, a state of the art GMRES variant, by means of numerical experiments on real-world problems in section 6.

2 Iterative Methods

In this section classical iterative methods like Jacobi and Gauss–Seidel are recalled and briefly inspected, with the goal to introduce a modern class of Krylov subspace methods in the next section.

Classical iterative methods can be retrieved by splitting the coefficient matrix A into parts $A = N + P$ where P is non-singular. For example, the Jacobi method uses $P = diag(A)$ and Gauss–Seidel defines P as the strictly upper triangular part of A [29]. If $e_k \equiv x - x_k$ denotes the error and $r_k \equiv b - Ax_k$ the residual of an approximate solution x_k for $k = 0, 1, \ldots$ with x_0 the initial guess, the error equation

$$Ae_k = A(x - x_k) = b - Ax_k = r_k \tag{2.1}$$

can be solved for e_k to find the true solution $x = x_k + A^{-1}r_k$. Classical iterative methods approximate A^{-1} by P^{-1}, which is easier to invert, so that the next approximate solution is

$$x_{k+1} = x_k + P^{-1}r_k. \tag{2.2}$$

Since P^{-1} is not necessarily a good approximation of the inverse of A and fixes the direction for the next approximation, one might want to have control over the length of the vector $P^{-1}r_k$ at each step, by adding a parameter $\alpha_k \in \mathbb{C}$, which leads to *non-stationary iterative methods* of the form

$$x_{k+1} = x_k + \alpha_k P^{-1}r_k \tag{2.3}$$

referred to as the non-stationary Richardson iteration. The optimal choice for the parameters α_k are a point of discussion, but as we will see in section 3.2, there is no need to retrieve them explicitly in modern Krylov subspace methods.

For ease, we take $P = I$, but we will inspect non-identity choices for P in section 5.1 as they lead to preconditioned systems. It can be verified from the definition of the residual and equation 2.3 that the residuals for the non-stationary Richardson method are related by

$$r_{k+1} = (I - \alpha_k A)r_k$$

for all $k \geq 0$, so that generally the residual r_k is a product of a polynomial $p_{k-1} \in \mathbb{P}_{k-1}$ in A of degree $k - 1$ with $p(0) = 1$ and the initial residual:

$$r_k = \prod_{i=0}^{k-1}(I - \alpha_i A)r_0 = p_{k-1}(A)r_0. \tag{2.4}$$

And by substitution in equation (2.3) it follows that any approximation x_k generated by the non-stationary Richardson method can be written as a combination of the initial guess x_0 and a polynomial in A:

$$x_k = x_0 + p_{k-1}(A)r_0. \tag{2.5}$$

Thus the non-stationary Richardson method can reach any element in the affine space $x_0 + \mathcal{K}_k$, where $\mathcal{K}_k = \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$ is a Krylov subspace, while stationary methods will only generate a fixed sequence of approximations in this affine space.

Note that from the perspective of approximation theory A^{-1} is approximated by a polynomial as $A^{-1} \approx p_{k-1}(A)$. In the next section we will see that the inverse of A can exactly be expressed as a polynomial of highest degree n, so that the solution $x = x_0 + A^{-1}r_0$ lies within the affine space $x_0 + \mathcal{K}_n$.

3 Krylov subspace methods

In the previous subsection it was already seen that classical iterative methods iterate within an affine space $x_0 + \text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$, where the latter space is a Krylov subspace. In this section we will show that the exact solution to a linear system is an element of such a subspace. This is a powerful result for iterative methods which use a Krylov subspace as their search space for the solution; the exact solution can be found in a fixed number of iterations, if the Krylov subspace is enlarged at each iteration. Note also that a new basis vector for the Krylov subspace is simply found by a single matrix-vector product with the previous basis vector.

Various iterative methods have been proposed which use this subspace as a search space for the solution. These Krylov subspace methods will be classified in this section, and the Arnoldi algorithm will be discussed for finding a numerically stable basis for the Krylov subspace.

Definition 3.1. A *Krylov subspace* $\mathcal{K}_k(A, b)$ is a subspace of $\mathbb{C}^n$ spanned by k vectors, induced by a matrix $A \in \mathbb{C}^{n\times n}$ and a vector $b \in \mathbb{C}^n$ as

$$\mathcal{K}_k(A, b) \equiv \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}. \tag{3.1}$$

As a consequence, every element $u \in \mathcal{K}_k(A, b)$ of a Krylov subspace can be expressed as a polynomial $p_{k-1} \in \mathbb{P}_{k-1}$ as $u = p_{k-1}(A)b$ with the property that $p(0) = 1$.

Krylov subspaces are of particular interest because the solution x to $Ax = b$ lies within a Krylov subspace if A is non-singular. To show this, we construct a polynomial $q(A)$ of lowest degree for which $q(A) = 0$. By the Cayley-Hamilton theorem we know that the degree of this polynomial does not exceed n as A satisfies its own characteristic equation. But it can be shown that the polynomial can even have a lower minimal degree.

Definition 3.2. The *minimal polynomial with respect to* $A \in \mathbb{C}^{n\times n}$ is the unique, monic polynomial $q(A)$ over A with lowest degree such that

$$q(A) = 0. \tag{3.2}$$

Suppose A has distinct eigenvalues $\{\lambda_1, \ldots, \lambda_d\}$ with $d \leq n$ where each eigenvalue λ_j has index m_j. Denote by $m \equiv \sum_{j=1}^{d} m_j$ the sum of the indices, then

$$q(A) = \prod_{j=1}^{d} (A - \lambda_j I)^{m_j} \tag{3.3}$$

is the minimal polynomial. If we expand the polynomial to

$$q(A) = \alpha_0 I + \alpha_1 A + \cdots + \alpha_m A^m \tag{3.4}$$

for all $\alpha_j \in \mathbb{C}$, it is clear from equation 3.3 that $\alpha_0 \neq 0$ since the eigenvalues are non-zero. By using $I = A^{-1}A$ and moving terms, the inverse of A can be written as a polynomial of highest degree $m - 1$.

$$A^{-1} = -\frac{1}{\alpha_0} \sum_{j=0}^{m-1} \alpha_{j+1} A^j \tag{3.5}$$

Therefore, the solution $x = A^{-1}b$ is an element of the Krylov subspace $\mathcal{K}_m(A, b)$.

Note. Also for Krylov subspace methods, usually a guess $x_0 \in \mathbb{C}^n$ is given for the solution, which means that the error equation $Ae = r_0 = b - Ax_0$ is to be solved.

From a numerical point of view, it is necessary to find a better basis for the Krylov subspace, as the basis vectors in definition 3.1 are exactly the first iterates of the power method. Since the power method converges to the eigenvector corresponding to the eigenvalue of largest magnitude if it exists, the basis vectors tend to become numerically dependent due to finite precision. To overcome this problem, Arnoldi's algorithm is often used to create a orthonormal basis for Krylov subspaces.

3.1 Arnoldi algorithm

The Arnoldi algorithm [2] explicitly constructs an orthonormal basis $\{v_1, \ldots, v_{i-1}\}$ for the Krylov subspace $K(A, r_0)$ and a Hessenberg matrix $H_m = (h_{ij})$. The basis vectors form the matrix $V_m = [v_1 \ \ldots \ v_m]$. The algorithm starts with a normalized vector v_1, and orthogonalizes each new basis vector Av_i to all previous vectors $\{v_1, \ldots, v_i\}$ by use of the modified Gram-Schmidt [6] process. The algorithm terminates after step j if $h_{j+1,j} = 0$. If the Arnoldi algorithm does not terminate at the mth step, it can also be formulated in terms of the matrices V_{m+1} and H_m.

Proposition 3.1. *If the orthonormal matrix $V_{m+1} \in \mathbb{C}^{n \times m}$ and the Hessenberg matrix $\hat{H}_m \in \mathbb{C}^{(m+1) \times m}$ are generated by m steps of the Arnoldi algorithm and if $H_m \in \mathbb{C}^{m \times m}$ denotes H_m without its last row, then both the following three identities hold:*

$$V_m^H A V_m = H_m, \tag{3.6}$$

$$AV_m = V_{m+1} \hat{H}_m. \tag{3.7}$$

$$= V_m H_m + h_{m+1,m} v_{m+1} e_m^T. \tag{3.8}$$

Algorithm 1 The Arnoldi algorithm

1: $v_1 := r_0 / \|r_0\|$
2: **for** $j = 1, \ldots, m$ **do**
3: **for** $i = 1, \ldots, j$ **do**
4: $h_{ij} = (Av_j, v_i)$
5: **end for**
6: $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
7: $h_{j+1,j} = \|\hat{v}_{j+1}\|_2$
8: **if** $h_{j+1,j} = 0$ **then**
9: **return**
10: **else**
11: $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$
12: **end if**
13: **end for**

Proof. Let e_i denote the ith standard basis vector. Since $h_{ij} = 0$ for $i + 1 > j$, it holds that

$$\begin{aligned} AV_m &= [Av_1, \ldots, Av_{m-1}, Av_m] \\ &= \left[\hat{v}_2 + v_1 h_{11}, \quad \ldots, \quad \hat{v}_m + \sum_{i=1}^{m-1} h_{ij} v_i, \quad \hat{v}_{m+1} + \sum_{i=1}^{m} h_{ij} v_i \right] \\ &= \left[\sum_{i=1}^{m} h_{i1} v_i, \quad \ldots, \quad \sum_{i=1}^{m} h_{i,m-1} v_i, \quad \hat{v}_{m+1} + \sum_{i=1}^{m} h_{im} v_i \right] \\ &= V_m H_m + \hat{v}_{m+1} e_m^T = V_m H_m + h_{m+1,m} v_{m+1} e_m^T. \end{aligned} \tag{3.9}$$

The last equality of equation (3.9) follows from line 11 of the Arnoldi algorithm 1. Since V_m is orthonormal and v_{m+1} is orthogonal to all columns of V_m, premultiplication of equation (3.9) with V_m^H proves the first identity

$$V_m^H A V_m = H_m.$$

The second identity follows by extending the summations of equation (3.9) one index further to $m + 1$ and recalling that $h_{m+1,j} = 0$ for $j \neq m$ and $\hat{v}_{m+1} = h_{m+1,m} v_{m+1}$, implying

$$AV_m = V_{m+1} \hat{H}_m.$$

The third identity occurs already in equation (3.9). □

3.2 Classification of Krylov subspace methods

As mentioned in section 2, classical stationary methods generate a pre-determined sequence of approximate solutions $\{x_0, x_1, \ldots\}$ within an affine space $x_0 + \mathcal{K}_k(A, r_0)$. Non-stationary methods on the other hand can generate any sequence of approximate solutions in the same space, but it may not be immediately clear how to pick the optimal

parameter α_{k+1} at each kth iteration. Also, the best choice of parameters $\alpha_1, \ldots, \alpha_k$ for the kth iteration might not be optimal for the $(k+1)$th iteration.

Furthermore, the definition of optimality should be clarified; it can either mean that the error or residual is minimal with respect to all possible solutions in the Krylov subspace, or that the residual is orthogonal to a specific subspace of $\mathbb{C}^n$.

Krylov subspace methods incorporate these different optimality properties, leading to different classes of methods. Generally, they can be defined as methods which extract the approximate solution x_k from a search space $\mathcal{V}$ of dimension k under certain conditions with respect to a test space $\mathcal{W}$. Within each class, methods can be distinguished by the search and test spaces they use and by the types of matrices they apply to.

The general classes of subspace methods are presented below, where we assume for simplicity to no initial guess is given so that the initial residual $r_0 = b$. Furthermore $x_k \in \mathcal{V}$ denotes the approximate solution for a k-dimensional search space $\mathcal{V}$.

Ritz–Galerkin The Ritz–Galerkin approach defines the search and test space equally as $\mathcal{V} = \mathcal{W} = \mathcal{K}_k(A, r_0)$ and extracts x_k from $\mathcal{V}$ such that the residual r_k is perpendicular to $\mathcal{V}$:

$$b - Ax_k \perp \mathcal{V}. \tag{3.10}$$

If the columns of $V_k \in \mathbb{C}^{n \times k}$ form a basis for $\mathcal{V}$, then the unknown can be written as $x_k = V_k y$ for a $y \in \mathbb{C}^k$, and the requirement of equation (3.10) is equivalent to

$$V_k^H A V_k y = V_k^H b, \tag{3.11}$$

which is to be solved for a $y \in \mathbb{C}^k$. This method is of particular use for Hermitian matrices A, since that property keeps preserved for the smaller coefficient matrix $V_k^H A V_k$.

Examples of this approach include FOM and the Conjugate Gradient (CG) method. The latter is historically the first Krylov subspace method, proposed in the 1950s and popularized in the 1970s [24]. It applies to SPD matrices, so that when V_k is formed by the Arnoldi algorithm, $V_k^H A V_k$ is tri-diagonal since it is hermitian and Hessenberg. In that case the Arnoldi algorithm simplifies to a memory-cheap three-term recurrence. By positive definiteness, equation 3.11 can be solved using Cholesky decomposition.

Petrov–Galerkin The Petrov–Galerkin approach is similar to the Ritz-Galerkin approach, with the main difference that it defines another test space such that $\mathcal{V} = \mathcal{K}_k(A, r_0) \neq \mathcal{W}$. The residual r_k is again required to be perpendicular to the test space $\mathcal{W}$:

$$b - Ax_k \perp \mathcal{W}. \tag{3.12}$$

If the columns of $V_k, W_k \in \mathbb{C}^{n \times k}$ form a basis for respectively $\mathcal{V}$ and $\mathcal{W}$, then the unknown can be written as $x_k = V_k y$ for a $y \in \mathbb{C}^k$, and the requirement of equation (3.12) becomes

$$W_k^H (b - A V_k y) = 0 \Leftrightarrow W_k^H A V_k y = W_k^H b. \tag{3.13}$$

Robust methods within this class like BiCGSTAB and QMR apply to general matrices and choose $\mathcal{W}$ in such a way that $W_k^H A V_k$ is tri-diagonal. This results in relatively little memory usage but longer iterations [17, 33].

Minimal residual Minimal residual methods require that the normed residual is minimal for $x_k \in \mathcal{V}$:

$$x_k = \underset{x^* \in \mathcal{V}}{\arg\min} \|b - Ax^*\| . \tag{3.14}$$

Examples include GMRES [30] and MINRES [28]. In the next section, GMRES will be treated in depth.

Minimal error Minimal error methods require that $x_k \in \mathcal{V}$ is chosen such that the normed error $e_k = \|x - x_k\|$ is minimal. Although this might be a surprising requirement at first sight, since the true solution is unknown, it is possible for special subspaces $\mathcal{V}$. It is implemented in for example SYMMLQ [28].

For all these types of subspace methods, the matrices of the type $W_k^H A V_k$ are often by-products of the method itself. The Arnoldi algorithm as seen in Proposition 3.1 gives for instance a relation $V_k^H A V_k = H_k$ where H_k is Hessenberg.

4 GMRES

The main idea behind the GMRES method is to minimize the residual in Euclidean norm over all vectors of the k dimensional affine space $x_0 + \mathcal{K}_k(A, r_0)$, where $x_0 \in \mathbb{C}^n$ is an initial guess and the latter space is the Krylov subspace induced by A and r_0. This space is iteratively enlarged until a satisfactory approximate solution is found. Thus formally, at every kth iteration of GMRES the approximate solution x_k is

$$x_k \equiv \underset{z \in x_0 + \mathcal{K}_k(A, r_0)}{\arg\min} \|Az - b\|_2 . \tag{4.1}$$

It is trivial that the Krylov subspace generated at each iteration contains the previous Krylov subspaces, i.e. $\mathcal{K}_k(A, r_0) \subset \mathcal{K}_{k+1}(A, r_0)$ for all k. Therefore it holds that the sequence of norms of minimal residuals found by GMRES are non-increasing, referred to as the optimality property of GMRES.

To find such a minimizing vector $x_k \in x_0 + \mathcal{K}_k(A, r_0)$, write $x_k = x_0 + y$ for a $y \in \mathcal{K}_k(A, r_0)$. Let the matrix V_k be obtained by the Arnoldi algorithm to span the Krylov subspace, then there exists a vector $z \in \mathbb{C}^k$ such that $y = V_k z$. By using the properties of the Arnoldi decomposition, we obtain:

$$\begin{aligned} r_k = b - Ax_k &= b - A(x_0 + y) \\ &= r_0 - AV_k z \\ &= \|r_0\|_2 v_1 - V_{k+1} \hat{H}_k z \\ &= V_{k+1}(\|r_0\|_2 e_1 - \hat{H}_k z). \end{aligned}$$

The third equality follows from the second identity of Proposition 3.1. For clarity, write $\gamma = \|r_0\|_2$. Since V_{k+1} is orthonormal, we can express the normed residual as

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \left\|\gamma e_1 - \hat{H}_k z\right\|_2. \tag{4.2}$$

Minimization of equation (4.2) with respect to z is equivalent to finding the least-squares solution of

$$\hat{H}_k z = \gamma e_1. \tag{4.3}$$

Least-squares problems can be solved by general QR-decomposition techniques, but the sparse Hessenberg structure of $\hat{H}_k$ suggests that a special-case technique can be applied which zeroes out only the sub-diagonal entries. For instance, the QR decomposition can be obtained by applying multiple Givens rotations which zero out sub-diagonal entries one by one.

Definition 4.1. A *Givens matrix* $G_i^m(c,s) : \mathbb{S}^2 \to \mathbb{C}^{m\times m}$ is a unitary matrix of the form

$$G_i^m(c,s) = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c & s & & & \leftarrow i \\ & & & -\bar{s} & c & & & \leftarrow i+1 \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} \tag{4.4}$$

where $\mathbb{S}^2 = \{(c,s) \in \mathbb{C}^2 : |c|^2 + |s|^2 = 1\}$ and $1 \le i \le m-1$. A matrix-vector product with a Givens matrix is called a *Givens rotation*. The arguments (c,s) and the superscript are dropped if they are clear in context.

Definition 4.2. Let the *sign* of a complex number $z \in \mathbb{C}$ be defined as

$$\operatorname{sign} z = \begin{cases} \frac{z}{|z|} & \text{if } z \neq 0 \\ 1 & \text{if } z = 0 \end{cases}. \tag{4.5}$$

Proposition 4.1. *Let $x \in \mathbb{C}^m$ and $1 \le i \le m-1$. If $x_{i+1} \neq 0$, then $G_i^m(c,s)$ with*

$$\begin{aligned} c &= \frac{x_i}{\sqrt{|x_i|^2 + |x_{i+1}|^2}}, \\ s &= \operatorname{sign}(x_i)\frac{\bar{x}_{i+1}}{\sqrt{|x_i|^2 + |x_{i+1}|^2}} \end{aligned} \tag{4.6}$$

is a Givens matrix. If $y = G_i x$, then

1. $y_{i+1} = 0$,

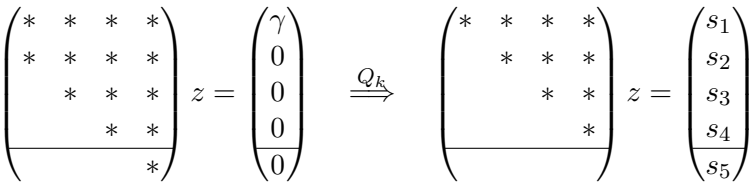


*Figure 1: Effect of Givens rotation on a 5×5 matrix. *-symbols represent non-zero elements*

2. $y_i = \text{sign}(x_i)\sqrt{|x_i|^2 + |x_{i+1}|^2}$,
3. $y_j = x_j$ *for all* $j < i$ *and* $j > i + 1$.

Proof. The matrix is a Givens matrix since $|c|^2 + |s|^2 = 1$ for all $x_i, x_{i+1} \in \mathbb{C}$. The three remaining statements are easily verified by performing the product $G_i x$ and substituting c and s:

$$y = G_i x = \begin{pmatrix} x_1 & \dots & x_{i-1} & cx_i + sx_{i+1} & -\bar{s}x_i + cx_{i+1} & x_{i+2} & \dots & x_m \end{pmatrix}^T. \tag{4.7}$$

□

Now we wish to construct a unitary matrix Q_k which zeros out the subdiagonal entries of $\hat{H}_k$ when premultiplying as $Q_k \hat{H}_k$. Denote by $\hat{R}_k^{(i)}$ the matrix $\hat{H}_k$ that is premultiplied by i Givens matrices $G_i^{k+1}(c_i, s_i)$ as below, where the arguments and superscript are dropped for clarity:

$$\hat{R}_k^{(i)} \equiv G_i G_{i-1} \cdots G_1 \hat{H}_k. \tag{4.8}$$

Let the coefficients (c_ℓ, s_ℓ) of all Givens matrices $G(c_\ell, s_\ell)$ be defined as in Proposition 4.1 with the ℓth column of $\hat{R}_k^{(\ell-1)}$ as the vector x.

By the Hessenberg structure of $\hat{H}_k$ and Proposition 4.1, it follows that $\hat{R}_k^{(k)}$ is upper triangular. Now define $Q_k \equiv G_k G_{k-1} \cdots G_1$. This matrix is unitary as it is the product of unitary matrices. It simplifies equation (4.8) to:

$$Q_k \hat{H}_k = \hat{R}_k^{(k)}. \tag{4.9}$$

By orthogonality of Q_k, we can simplify the norm of equation (4.2) to

$$\left\| \gamma e_1 - \hat{H}_k z \right\|_2 = \left\| Q_k (\gamma e_1 - \hat{H}_k z) \right\|_2 = \left\| \hat{s} - \hat{R}_k^{(k)} z \right\|_2, \tag{4.10}$$

where $\hat{s} \equiv \gamma Q_k e_1$. The difference between the structures of the left and right hand side of equation (4.10) is depicted in Figure 1.

Since $\hat{R}_k^{(k)}$ is upper triangular with a bottom row consisting of only zeroes, the least-squares problem can be solved by backward substitution. Thus we only have to consider the equation without its last row. If we let R_k denote $\hat{R}_k^{(k)}$ without its last row and

similarly let s be $\hat{s}$ without its last entry, the minimizer z of equation (4.10) can be expressed as $z = R_k^{-1}s$ and is simply obtained by backward substitutions. Furthermore, the residual is the last entry of the right hand side $\hat{s}$.

Lastly, the approximate solution x_k to the original problem $Ax = b$ is retrieved by computing

$$x_k = x_0 + V_k z. \tag{4.11}$$

Theorem 4.1. *The normed GMRES residual at the kth iteration is retrieved as* $\|r_k\|_2 = |\hat{s}_{k+1}|$*, where* $\hat{s}_{k+1}$ *is the last entry of* $\hat{s}$*.*

Proof. From previous arguments it follows that

$$\begin{aligned} b - Ax_k &= V_{k+1}(\gamma e_1 - \hat{H}_k z) \\ &= V_{k+1}Q_k^T Q_k(\gamma e_1 - \hat{H}_k z) \\ &= V_{k+1}Q_k^T(\hat{s} - \hat{R}_k^{(k)} z). \end{aligned} \tag{4.12}$$

But z minimizes $\left\|\hat{s} - \hat{R}_k^{(k)} z\right\|_2 = \hat{s}_{k+1}$ and since both V_{k+1} and Q_k are unitary, it follows that

$$\|b - Ax_k\|_2 = |\hat{s}_{k+1}|, \tag{4.13}$$

which finishes the proof. □

Note. The practical use of Theorem 4.1 is that it allows to look up the GMRES residual norm at each iteration without solving the the least-squares problem. Thus, only if the GMRES residual is under a user specified tolerance, the backward substitution has to be performed.

To summarize this section we provide in Algorithm 2 the pseudocode which coincides with the GMRES algorithm when $M = I$ and $m = \infty$.

4.1 Convergence of GMRES

Although no sharp bounds for the speed of convergence of GMRES for general non-hermitian matrices A are known yet, it is widely experienced in practical applications that clustered eigenvalues of the coefficient matrix A away from the origin are crucial for fast convergence, while a spread eigenvalue distribution or the presence of single eigenvalues close to the origin can stagnate GMRES.

The argument for this behaviour runs as follows in the assumption that A is diagonalizable, which is a reasonable in finite-precision arithmetic. The residual at the kth step of GMRES for an approximate solution $x_k \in x_0 + \mathcal{K}_k(A, r_0)$ can be expressed as

$$r_k = b - Ax_k = b - A(x_0 + y) = r_0 - Ay$$

for a $y \in \mathcal{K}_k(A, r_0)$. Thus it follows that

$$r_k \in r_0 + A\mathcal{K}_k(A, r_0) = r_0 + \text{span}\{Ar_0, A^2 r_0, \ldots, A^k r_0\}.$$

Algorithm 2 The GMRES(m) algorithm with left preconditioning. The expression $h_{:,l}$ denotes the lth column of H_i

function GMRES(A, b, x_0, m, M)
  $x_0^{(1)} = x_0$ ▷ Initial residual
  for $j = 1, 2, \ldots$ **do**
    $r := M(b - Ax^{(j)})$
    $v_1 := r / \|r\|_2$
    $s := \|r\|_2 e_1$
    for $i = 1, 2, \ldots, m$ **do**
      $w := MAv_i$ ▷ Start Arnoldi process
      for $k = 1, \ldots, i$ **do**
        $h_{k,i} = (w, v_k)$
        $w \leftarrow w - h_{k,i} v_k$
      end for
      $h_{i+1,i} = \|w\|_2$
      $v_{i+1} = w / h_{i+1,i}$
      for $k = 1, \ldots, i-1$ **do** ▷ Apply previous Givens rotations
        $h_{:,i} \leftarrow G_k h_{:,i}$
      end for
      Construct G_i from $h_{i,i}$ and $h_{i+1,i}$ ▷ Construct new Givens rotation
      $h_{:,i} \leftarrow G_i h_{:,i}$ ▷ Apply new Givens rotation
      $s \leftarrow G_i s$
      if $s_{i+1} <$ tol **then**
        Solve $H_i y = s$ for y with backward substitutions
        return $x = x_0 + V_i y$
      end if
    end for
    Solve $H_m y = s$ for y with backward substitutions
    $x_0 \leftarrow x_0 + V_m y$
  end for
end function

which allows us to write the residual as a product of a polynomial over A of highest degree k and the initial residual: $r_k = p_k(A)r_0$, with the property that $p_k(0) = 1$. Suppose A is diagonalizable such that the eigendecomposition is given by $A = X\Lambda X^{-1}$ where $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix containing the eigenvalues of A and the columns of X are the corresponding eigenvectors. Then it is not hard to derive the following bound

$$\begin{aligned} &\|r_k\|_2 = \min_{p_k \in \mathbb{P}_k^1} \|p_k(A)r_0\|_2 \leq \kappa_2(V) \min_{p_k \in \mathbb{P}_k^1} \|p_k(\Lambda)\|_2 \|r_0\|_2 ; \\ &\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa_2(V) \min_{\substack{p_k \in \mathbb{P}_k \\ p_k(0)=1}} \max_{i=1,\ldots,n} |p_k(\lambda_i)|, \end{aligned} \tag{4.14}$$

where $\kappa_2(V) = \|V\|_2 \left\|V^{-1}\right\|_2$ is the condition number of V. Without loss of generality, the eigenvectors can be scaled to minimize the condition number of V.

The bound of equation (4.14) depends both on the distribution of the eigenvalues and the conditioning of the eigenmatrix X. In regard to the first quantity, if the number of iterations must be low, the order of the polynomial will be low as well. To minimize the quantity this polynomial must take small values at all eigenvalues λ_i, which can only be possible when the eigenvalues are clustered. Since $p_k(0) = 1$ is a requirement, this is only possible if the eigenvalues are clustered away from the origin.

The second quantity depends on the normality of A. For normal matrices $\kappa_2(V)$ drops out and the bound of equation (4.14) depends solely on the distribution of the eigenvalues; moreover, the bound is known to be sharp. For highly non-normal matrices however, the eigenvector matrix V can be ill-conditioned so that one can only conclude that the convergence may be slow, or that the bound is undescriptive. In this event, one can resort to other bounds arising from research on the field of values and pseudospectra as summarized in [16].

However, it should be noted that there are cases where the eigenvalue distribution alone does not determine the convergence of GMRES. A dramatic example stems from [19] where it is shown that any non-increasing sequence $\{r_0, r_1, \ldots, r_{n-1}\}$ is a possible residual sequence for the first $n-1$ GMRES iterations for a matrix A with any desired eigenvalues. Thus, for any eigenspectrum one can find a matrix for which GMRES does not make any progress until the last step.

These observations have not reduced the popularity of GMRES, mostly because these matrices are rarely encountered in applications and secondly because GMRES is usually applied to preconditioned matrices with more advantageous convergence properties, as discussed in detail in section 5.

Another motivation for the popularity of GMRES stems from an article by Van der Horst and Vuik in [34], where the super-linear convergence behaviour of GMRES is studied. This means that the rate of convergence only improves after a number of iterations, when the smallest eigenvalues of the Hessenberg matrix H_m are sufficient approximations of the smallest eigenvalues of A. In other words, the Hessenberg matrix should capture the smallest eigenvalues of A.

The eigenvalues of the Hessenberg matrix are referred to as Ritz values, and it is easy to see that they converge to eigenvalues of A.

Definition 4.3. Given a matrix $A \in \mathbb{C}^n$ and a linear subspace $\mathcal{K} \subset \mathbb{C}^n$, a pair $(\theta, y) \in \mathbb{C} \times \mathbb{C}^n$ is called a *Ritz pair with respect to* $\mathcal{K}$ if

$$Ay - \theta y \perp \mathcal{K}. \tag{4.15}$$

In detail, θ is called a *Ritz value* and y is called a *Ritz vector*.

For Krylov subspace methods we can choose the Krylov subspace $\mathcal{K}_m(A, r_0)$ itself.

Proposition 4.2. *All eigenpairs (θ, x) of the Hessenberg matrix H_m from the Arnoldi algorithm are Ritz pairs (θ, y) of A with $y = V_m x$.*

Proof. The first statement follows from the first identity of Proposition 3.1:

$$\begin{aligned} H_m x &= \theta x \\ V_m^H A V_m x &= \theta V_m^H V_m x \\ V_m^H (Ay - \theta y) &= 0 \end{aligned} \tag{4.16}$$

Since V_m in a basis for $\mathcal{K}_m(A, r_0)$, it follows that

$$Ay - \theta y \perp \mathcal{K}_m(A, r_0), \tag{4.17}$$

which means that (θ, y) is a Ritz-pair of A. □

4.2 Improvements and constraints

A difficulty of GMRES concerns the linearly increasing amount of storage costs and operational costs with the iterations. This is due to the Arnoldi algorithm, which requires the storage of a basis vector for the Krylov subpace at each iteration, and orthogonalization of this vector with respect to all previous basis vectors. Although the number of iterations necessary for GMRES to find the exact solution is at most n in real arithmetic, this value can be too large due to memory restraints. Thus, the optimality property of GMRES comes at a cost when the number of iterations increases.

4.2.1 GMRES(m)

The easiest way to circumvent the linearly increasing memory and computational costs of GMRES is to repeatedly restart the GMRES process after a fixed number of m iterations with the solution at step m as an initial guess for the next restart, resulting in the GMRES(m) or restarted GMRES method. This method is most popular for large linear systems.

Thus, for each $k = 0, 1, \ldots$ denote by $x_0^{(k)}$ the initial guess, apply m steps of GMRES to obtain the approximate solution $x_m^{(k)}$ and finally set $x_0^{(k+1)} = x_m^{(k)}$. The k-loop

and ordinary GMRES-loop are respectively referred to as outer and inner loop. The pseudocode is provided in Algorithm 2, where the idea of preconditioning is already incorporated.

The draw-back of this solution is the loss of the optimality property. While GMRES terminates in at most n steps, this property is not necessarily achieved for GMRES(m).

4.2.2 Deflated restart

In an attempt to recover the optimality property in GMRES(m), information from the previous GMRES run can be recycled to improve the Krylov subspace for the next run.

As mentioned earlier, if the smallest eigenvalues of A are sufficiently approximated by Ritz values, the convergence of GMRES becomes super-linear. This happens after a couple iterations, so that for restarted GMRES the benefit of super-linear convergence marginally extends to a single inner iteration. If however, (approximate) eigenvectors of A are available, one can augment the Krylov subspace with those approximate eigenvectors in the next run to force small eigenvalues immediately in the Hessenberg matrix.

This idea was first incorporated in the Deflated Restart method GMRES-DR(m, ℓ), which was published in 2002 [27]. It exploits the Hessenberg matrix H_m at the end of an outer iteration to approximate a set of ℓ eigenvectors $\{u_1, \ldots, u_\ell\}$ corresponding to eigenvalues of A close to the origin and augments the Krylov subspace as

$$\mathcal{K} = \text{span}\{u_1, \ldots, u_\ell, r_0, Ar_0, \ldots, A^k\}.$$

for the next outer iteration.

Example 1. Let A be a diagonal matrix such that $a_{ii} = 1 - 0.8^i$ for $i = 1, \ldots, 500$. Let B be diagonal as well with $b_{11} = 0.001$, $b_{22} = 0.005$ and $b_{jj} = a_{jj}$ for $j = 3, \ldots, 500$. Then A and B have most of their eigenvalues clustered around 1, but B has two outliers near the origin. Fix the right-hand side b by setting the solution 1, and set the initial guess 0 and set the stopping criterion $\|b - Ax\|_2 / \|b\|_2 \leq 10^{-10}$. Then GMRES(5) converges for A and B in respectively 21 and 118 total inner iterations.

Example 2. Using the same matrix B and right-hand side as in example 1, GMRES-DR(5, 2) finishes in just 23 total inner iterations. This amounts to almost the same convergence speed as GMRES(5) for the A-matrix, where the two smallest eigenvalues are 'removed'.

To summarize the above results, examples 1 and 2 show the effect of the presence of small eigenvalues near the origin on the convergence of GMRES(m), as well as the success of GMRES-DR in such a situation. It is verified that GMRES performs better if the coefficient matrix has a more clustered eigenvalue distribution away from the origin. Since general matrices can not be assumed to have such an eigenvalue distribution, we will see in the next section how preconditioning can achieve this.

5 Preconditioning

Most improvement in the rate of convergence of Krylov subspace methods can be made , by reformulating the problem into an equivalent system with more advantageous convergence properties. This process is called preconditioning, and can be applied as follows:

Definition 5.1. Let M, M_1 and M_2 be $n \times n$ complex, non-singular matrices, we can distinguish three types of *preconditioners* for the system $Ax = b$.

Left preconditioning: $MAx = Mb$.

Right preconditioning: $AMy = b$ where the solution can be retrieved by computing $x = My$.

Split preconditioning: $M_1AM_2y = M_1b$ where the solution can be retrieved by computing $x = M_2y$.

In all cases mentioned in Definition 5.1, the matrices MA, AM and M_1AM_2 should somehow approximate the identity matrix. The motivation for this stems from the discussion on the bound of equation (4.14) for the convergence of GMRES. If for instance $MA \approx I$ in a sense, one can expect the eigenvalues of MA to be clustered around the point 1 in the complex plane, so that the effect of the preconditioner is advantageous for the rate of convergence.

However, another desirable feature of preconditioners is that its construction involves low additional storage costs and computational costs. These properties are conflicting: the optimal preconditioner is $M = A^{-1}$, but constructing this preconditioner is as expensive as solving the linear system itself. And vice versa requires $M = I$ minimal costs, yet it has no effect on the system. Furthermore, the optimal choice for a preconditioner and its parameters is problem-dependent, and the search for robust and cheap preconditioners is still an area which has not completely been explored.

First we will inspect several types of preconditioners. Section 5.1 is introductory, while section 5.2 shows well-established techniques that are already used in combination with Krylov solvers. Subsequently in section 5.3 we will see techniques to improve a given preconditioner, which makes it a small step to introduce the main contribution of this thesis.

Note. Algorithm 2 implements left preconditioning for GMRES(m). The stopping criterion is then determined by the preconditioned residual. Right preconditioning makes the stopping criterion independent from the preconditioner.

5.1 Stationary methods as preconditioners

At the end of section 2, the Richardson scheme was analysed for the special case where $P = I$. However, for stationary methods like Gauss–Seidel and Jacobi, the non-identity choice for P acts as a left preconditioner on $Ax = b$, as can be seen from:

$$x_{k+1} = x_k + P^{-1}r_k = x_k + (P^{-1}b - P^{-1}Ax_k). \tag{5.1}$$

It follows that the scheme iterates within the affine space $x_0 + \mathcal{K}_k(P^{-1}A, P^{-1}b)$.

5.2 Well-established preconditioning techniques

Preconditioners described in section 5.1 are usually not suited to reduce the number of Krylov iterations significantly, so better preconditioners have been proposed and analysed. These can roughly be categorized chronologically into incomplete factorization methods and sparse approximate inverses. The former category employs approximate decomposition techniques such as incomplete LU decomposition to create preconditioners of the form $M^{-1} = \tilde{L}\tilde{U}$ where $A \approx \tilde{L}\tilde{U}$, while the latter category tries to directly approximate the inverse of A by minimizing the distance of AM to the identity matrix under certain conditions.

Although preconditioners are expressed as separate matrices throughout this thesis, in practice it is usually redundant to assemble the preconditioner as a matrix if the preconditioning operations can be performed implicitly. This is especially the case for incomplete factorization methods, where only the L and U factors have to be stored. Worth noting is that preconditioners in the sparse approximate inverse category are assembled explicitly. Thus, one could categorize the above mentioned preconditioning techniques as implicit and explicit respectively as well.

5.2.1 Incomplete factorization methods

In the case of sparse matrices A, a factorization method like LU decomposition causes fill-in during the elimination process, so that the total number of non-zero elements of the factors is likely to be large with respect to the number of non-zeros of A.

Since additional storage costs can not be afforded for large matrices, several ideas have been proposed to reduce the costs. First of all, reordering the unknowns to create a more banded matrix is for instance an option. Subsequently, one can enforce sparsity by performing an incomplete factorization, which incorporates dropping strategies for entries in the factors to reduce the amount of fill-in.

The methods can be distinguished by the dropping strategies they implement. For instance, one idea is to define a subset $\mathcal{S} \subset \{(i,j) : i,j = 1,\ldots,n\}$ of positions of matrix entries and limit updates in the Gaussian elimination process to entries in $\mathcal{S}$. That is, for each row k and $i,j > k$:

$$a_{ij} \leftarrow \begin{cases} a_{ij} - a_{ik}a_{kk}^{-1}a_{kj} & \text{when } (i,j) \in \mathcal{S} \\ a_{ij} & \text{when } (i,j) \notin \mathcal{S} \end{cases}. \tag{5.2}$$

If $\mathcal{S}$ is chosen to be the set of positions of non-zero entries of A, the no-fill ILU factorization is obtained, abbreviated by ILU(0).

The no-fill factorization method has the advantage that the storage costs are known a priori and that the factorization can be computed cheaply, and is most successful when applied to M-matrices in combination with the CG-method [25]. The downside is that it does not provide a reliable factorization for general matrices.

A more subtle approach is to allow fill-in on positions of zero-entries of A as well under certain conditions. One of these conditions can be the level of fill. Each nonzero element in A gets an initial level of fill-in of 0, but when at position (i,j) a fill-in

element u_{ij} is created by a formula $u_{ij} = -a_{ik}a_{kk}^{-1}a_{kj}$, the level of fill of (i,j) becomes $level(i,j) = level(i,k) + level(k,j) + 1$ [29]. If the level of fill is above a given integer ℓ, the element u_{ij} is dropped. This method is referred to as $ILU(\ell)$, and works especially well if the matrix is (nearly) diagonally dominant. In that case it is typical that the magnitude of a fill-in element is small if the level of fill-in is high.

Another condition is to drop fill-in elements u_{ij} at a position (i,j) if they have a relatively small magnitude with respect to the norm of the corresponding ith row of A. This method is called ILUT(τ), where τ defines the threshold.

Both the conditions of ILU(ℓ) and ILUT(τ) make it difficult to predict the total storage costs for the factors. To overcome this problem, one can fix the maximal amount of fill-in in each row of the LU factors by an integer p. If more than p fill-ins occur in a single row, only the p largest, off-diagonal entries in magnitude are retained and the remaining are dropped. This algorithm can be combined with both ILU(ℓ) and ILUT(τ).

A more recent class of incomplete factorization methods takes into account the conditioning of the factors L and U. Suppose that $\tilde{L}\tilde{U}$ is a good approximation to A in the sense that the error $E \equiv A - \tilde{L}\tilde{U}$ has a small norm, and that the amount of storage costs is low. This does not necessarily imply that the quality of the preconditioner is good, since the error can be amplified if the factors L and U are ill-conditioned. For instance when using split preconditioning, the preconditioned matrix is given by $\tilde{L}^{-1}A\tilde{U}^{-1} = \tilde{L}^{-1}E\tilde{U}^{-1}$.

This observation has lead to the introduction of inverse-based ILU methods, which postpone rows and columns which add significantly to the size of $\|\tilde{L}\|$ and $\|\tilde{U}\|$ to the end of the matrix during the elimination process. Suppose the leading $k \times k$ block B of A has been decomposed into an incomplete LDU decomposition with $\tilde{L}, \tilde{D}, \tilde{U} \in \mathbb{C}^{k\times k}$ and $n-k$ rows and columns have been postponed, one obtains a decomposition of the form

$$A = \begin{pmatrix} B & C \\ D & E \end{pmatrix} \approx \begin{pmatrix} \tilde{L} & 0 \\ \tilde{X} & I \end{pmatrix} \begin{pmatrix} \tilde{D} & 0 \\ 0 & \tilde{S} \end{pmatrix} \begin{pmatrix} \tilde{U} & \tilde{Y} \\ 0 & I \end{pmatrix}, \tag{5.3}$$

where $\tilde{S}$ is an approximation to the Schur complement $S = E - DB^{-1}C$. One can define the approximate Schur complement in different ways — the simplest being $\tilde{S} = E - \tilde{X}\tilde{D}\tilde{Y}$ — leading to different bounds on the inverse error. If $\tilde{S}$ is of small in size, which happens if only few rows and columns are postponed, one can use direct methods to solve linear systems involving the Schur complement. Otherwise, a more careful decomposition can be continued on $\tilde{S}$, using a lower dropping threshold [7]. A pictorial sketch of the latter algorithm is shown in Figure 2.

5.2.2 Sparse approximate inverses

Since incomplete factorization methods are difficult to implement in parallel and require many problem-specific parameters, different preconditioning methods have been proposed in the past decades. These methods include sparse approximate inverse techniques like SPAI [21] and AINV [4], which directly approximate the inverse of A. At first sight this might seem doubtful, since the inverse of a sparse matrix is usually dense [14].

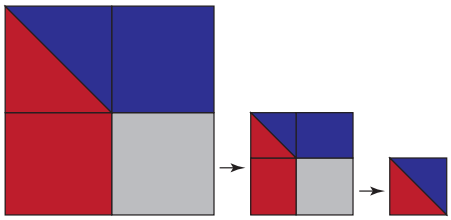


Figure 2: Multi-level ILU

These methods are very promising nonetheless, since often a decay pattern arises in the inverse matrix so that sparsity can be ensured by neglecting small entries. And most importantly, these methods are easily parallelizable, in contrast to factorization methods from section 5.2.1.

For example, for banded matrices that satisfy $\|A\| \leq 1$ and $\left\|A^{-1}\right\| \leq r \in \mathbb{R}$, which arise from finite element and finite difference methods, it can be shown that there exists constants $0 < \rho < 1$ and $C \in \mathbb{R}$ depending only on r and the bandwidth of A so that

$$|(A^{-1})_{ij}| \leq C\rho^{|i-j|}$$

for all $i, j = 1, \ldots, n$ [12]; an example is shown in Figure 3. Also, Greengard studied the decay of the Green's function, which in a sense describes the inverse of differential operators and thus in discretized fashion corresponds to the inverse of coefficent matrices arising from discretized differential equations [20]. Other results can be found in [13, 15, 23].

The main idea behind SPAI is to find a preconditioner M from a set of matrices $\mathcal{S}$ with a fixed sparsity pattern, which minimizes the Frobenius norm

$$\min_{M \in \mathcal{S}} \|I - AM\|_F \quad \Longleftrightarrow \quad \min_{M \in \mathcal{S}} \|e_i - Am_i\|_2 \tag{5.4}$$

for all $i = 1, \ldots, n$ where m_i denotes the ith column of M. Thus, each column of M can be found in a separate process, allowing parallelization.

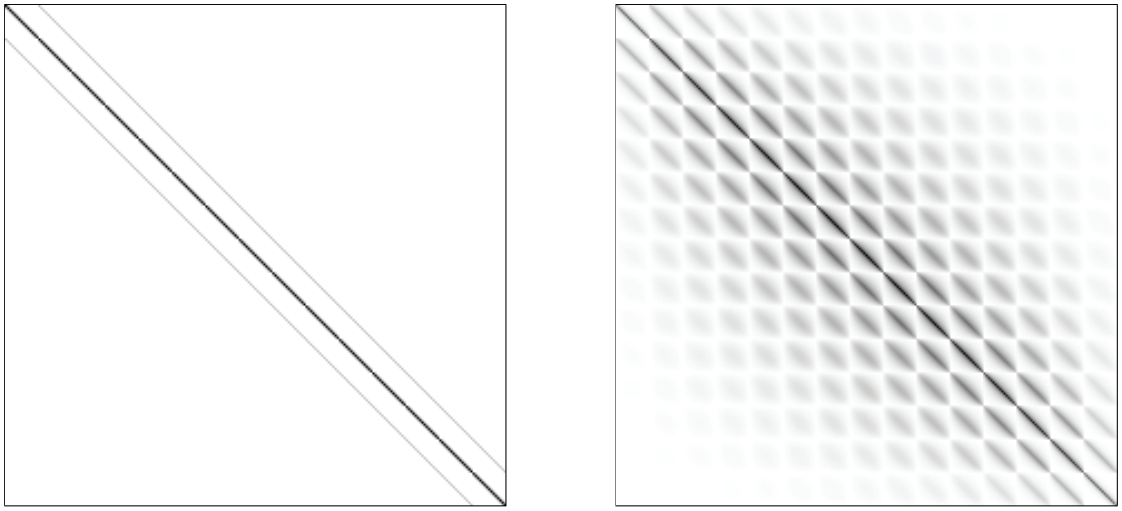
These methods have successfully been applied on dense matrices arising from the field of electromagnetism using Krylov subspace methods [1]. This success is thanks to the ability of SPAI to cluster the eigenvalues of the preconditioned coefficient matrix. For instance, with use of Gershgorin circles it can be proved [10] that if

$$\|e_j - Am_j\|_1 < \tau \tag{5.5}$$

for all j, then each eigenvalue λ of AM lies within the disc of radius τ around 1:

$$|\lambda - 1| < \tau.$$

One difficulty of SPAI is that the optimal sparsity structure $\mathcal{S}$ such that (5.5) is satisfied, is generally not known in advance. Only in special cases a specific structure may be chosen [22], but for general matrices the current solution is to adaptively update an initial structure, which is more expensive in implementation [11, 21].



(a) Coefficient matrix A (b) Inverse of A

Figure 3: Heat map of the magnitude of entries from a 250×250 coefficient matrix and its inverse, resulting from a 5-point discretization of the Poisson equation on a 50×50 mesh. White indicates zero entries, black indicates large values.

AINV bypasses this problem by forming two sets of vectors $\{z_i\}$ and $\{w_i\}$ for $i = 1, \ldots, n$ which are A-biconjugate: $w_i^T A z_j = 0$ for all $i \neq j$. If $W = [w_1, \ldots, w_n]$ and $Z = [z_1, \ldots, z_n]$, then $W^T A Z = D$ such that D is diagonal, so that $A^{-1} = Z D^{-1} W^T$. Sparsity of the inverse is enforced by dropping small entries from the vectors [5]. AINV however, is harder to parallelize [3].

5.3 Two-level spectral preconditioning

Although preconditioners mentioned in section 5.2 are often capable of clustering the eigenvalues away from the origin, it still encounters that a few outliers remain close to the origin. The presence of a single or a few eigenvalues close to the origin can negatively affect the convergence speed of subspace methods as was verified in example 1. Under these circumstances, augmented subspace methods like GMRES-DR might significantly improve the rate of convergence with respect to ordinary restarted GMRES.

In this section however, we will explore a different approach referred to as spectral preconditioning. The main idea is to improve a given preconditioner M, by moving small eigenvalues away from the origin. In this sense, spectral preconditioners can be seen as two-level preconditioners: an improved preconditioner is constructed from a given preconditioner.

Thus, in contrast to augmented subspace methods, spectral preconditioners deflate small eigenvalues from the coefficient matrix independently from the solving phase, while augmented subspace methods retain the small eigenvalues, but capture the blocking eigenspaces immediately in the Krylov subspace.

Within the family of two-level spectral preconditioners, the effect of the preconditioner on the eigenspectrum may differ: on the one hand there are deflation preconditioners which move a subset of eigenvalues to a given position $\sigma \in \mathbb{C}$, on the other hand there are coarse grid preconditioners which move a subset of eigenvalues only close to σ. In most cases, σ is chosen as 1 in the complex plane, assuming that most eigenvalues are already clustered near 1.

5.3.1 Several spectral preconditioners

In this section we will discuss four two-level, spectral preconditioners M_{coa}, M_{res}, M_{add} and M_{mul}. All of them are defined implicitly by the algorithms of Figure 4, where each algorithm computes the product $z = M_\star x$ for a vector $x \in \mathbb{C}^n$.

The first spectral preconditioner M_{coa} is a coarse grid preconditioner, which was published in [8]. It shifts a subset of the eigenvalues one unit along the real axis in the positive direction. We will follow the lines of [18] for the proof of its effect on the eigenspectrum of $M_{coa}A$.

The more novel additive and multiplicative deflation preconditioners M_{add} and M_{mul} were proposed in [9] in 2007. Not only do they perform an exact shift of the smallest eigenvalues of MA to 1 in the complex plane, but they cluster the remaining eigenvalues closer to 1 as well.

Lastly, the deflation preconditioner M_{res} has not been defined explicitly in the articles mentioned, as it was incorporated within the multiplicative and additive spectral preconditioners. We will define it explicitly for ease of reference.

The common ground for all these spectral preconditioners is as follows. Let M be the initial preconditioner and $\{\lambda_1, \ldots, \lambda_n\}$ be the set of eigenvalues of MA ordered by their magnitudes, where multiple eigenvalues are repeated:

$$|\lambda_1| \leq |\lambda_2| \leq \cdots \leq |\lambda_n|.$$

Also, let $\{x_1, \ldots x_n\}$ be the set of associated eigenvectors. Furthermore, let $\mathcal{U}$ denote a k-dimensional right-invariant eigenspace of MA associated with the k smallest eigenvalues of MA, for which $U \in \mathcal{C}^{n\times k}$ forms a basis. That is,

$$MAU = UJ \tag{5.6}$$

for a matrix $J \in \mathbb{C}^{k\times k}$ where J has the eigenvalues $\{\lambda_1, \ldots, \lambda_k\}$.

If A is diagonalizable, an obvious choice for J and U is to define $J = diag(\lambda_1, \ldots, \lambda_k)$ and U as the set of associated eigenvectors $\{x_1, \ldots, x_k\}$. Otherwise a Jordan decomposition can be used. However, multiple bases for $\mathcal{U}$ are possible, where some choices may have computational benefits, as we will see later.

A last prerequisite is that we assume a matrix $W \in \mathbb{C}^{n\times k}$ which is chosen such that the W^HAU is non-singular. This allows us to define two last entities

$$\begin{aligned} A_c &\equiv W^HAU, \\ M_c &\equiv UA_c^{-1}W^H, \end{aligned} \tag{5.7}$$

where the $k \times k$ matrix A_c is referred to as the coarse grid operator, in line with the terminology in multigrid methods, and M_c the coarse preconditioner.

We will first inspect the preconditioners M_{coa}, M_{res} and M_{exa}, which are defined in Algorithms 3, 4 and 5 respectively. The differences between M_{coa} and M_{res} are very subtle in implementation, yet the effects on the eigenspectra are completely different: M_{coa} performs a shift along the real axis, while M_{res} exactly shifts the small eigenvalues to 1 in the complex plane. The price to pay for the exact shift of M_{res} is an additional matrix-vector product, in comparison with M_{coa}. The spectral preconditioner M_{exa} has the same effect as M_{res} on the spectrum, but M_{exa} is cheaper as it does not use matrix-vector products with A. To perform the exact shift, it depends on the eigenvalues of MA.

Proposition 5.1. *The matrix representation of M_{coa}, M_{res} and M_{exa} as defined implicitly in algorithms 3 and 4 are respectively:*

$$\begin{aligned} M_{coa} &= M + M_c, \\ M_{res} &= M + M_c - M_c A M, \\ M_{exa} &= M + M_c - U J A_c^{-1} W^H. \end{aligned} \tag{5.8}$$

Proof. The proof follows by simple substitutions □

Proposition 5.2. *The eigenvalues $\{\nu_1, \ldots, \nu_n\}$ of $M_{coa}A$ are*

$$\nu_i = \begin{cases} \lambda_i & \text{if } i > k \\ 1 + \lambda_i & \text{if } i \leq k \end{cases}.$$

Proof. The proof is to show that $M_{coa}A$ is similar to a matrix whose eigenvalues are ν_i for all i. Let $U^\perp \in \mathbb{C}^{n-k}$ be an orthogonal complement of U and denote by $[U \,|\, U^\perp]$ the full rank $n \times n$ matrix formed by the columns of U and $U^\perp$, then by equation (5.6) the following similarity relation holds:

$$MA[U \,|\, U^\perp] = [UJ \,|\, MAU^\perp] = [U \, U^\perp] \begin{pmatrix} J & E \\ 0 & F \end{pmatrix}, \tag{5.9}$$

for certain matrices E and F, where F has the eigenvalues $\{\lambda_{k+1}, \ldots, \lambda_n\}$. Also

$$\begin{aligned} M_c A[U \,|\, U^\perp] &= [U(W^H AU)^{-1} W^H AU \,|\, U A_c^{-1} W^H AU^\perp] \\ &= [U \,|\, U^\perp] \begin{pmatrix} I & A_c^{-1} W^H AU^\perp \\ 0 & 0 \end{pmatrix}. \end{aligned} \tag{5.10}$$

Consequently by combining equations (5.9) and (5.10)

$$(M + M_c)A[U \,|\, U^\perp] = [U \,|\, U^\perp] \begin{pmatrix} I + J & E + A_c^{-1} W^H AU^\perp \\ 0 & F \end{pmatrix}, \tag{5.11}$$

which is the similarity relation which concludes the proof. □

Figure 4: Two-level spectral preconditioners M_{coa}, M_{res}, M_{exa}, M_{add} and M_{mul}

Algorithm 3: $z = M_{coa}x$

function $M_{coa}(x,\, M,\, A,\, U,\, W)$
  $c_1 = Mx$
  $c_2 = U(W^HAU)^{-1}W^Hx$
  return $z = c_1 + c_2$
end function

Algorithm 4: $z = M_{res}x$

function $M_{res}(x,\, M,\, A,\, U,\, W)$
  $c_1 = Mx$
  $r = x - Ac_1$
  $c_2 = U(W^HAU)^{-1}W^Hr$
  return $z = c_1 + c_2$
end function

Algorithm 5: $z = M_{exa}x$

function $M_{res}(x,\, M,\, A,\, U,\, W,\, J)$
  $c_1 = Mx$
  $c_2 = U(I - J)(W^HAU)^{-1}W^Hx$
  return $z = c_1 + c_2$
end function

Algorithm 6: Multiplicative: $z = M_{mul}x$

function $M_{mul}(x,\, M,\, A,\, U,\, W,\, \mu_1,\, \mu_2)$
  $z = 0$
  for $i = 1, \ldots, \mu_1$ **do**
    $z \leftarrow z + \omega M(x - Az)$
  end for
  $r = x - Az$
  $z \leftarrow z + U(W^HAU)^{-1}W^Hr$
  for $i = 1, \ldots, \mu_2$ **do**
    $z \leftarrow z + \omega M(x - Az)$
  end for
  return z
end function

Algorithm 7: Additive: $z = M_{add}x$

function $M_{add}(x,\, M,\, A,\, U,\, W,\, \mu_1,\, \mu_2)$
  $c_1 = 0$
  $c_2 = U(W^HAU)^{-1}W^Hx$
  for $i = 1, \ldots, \mu_1 + \mu_2$ **do**
    $c_1 \leftarrow c_1 + \omega M(x - Ac_1)$
  end for
  $c_1 \leftarrow (I - UW^H)c_1$
  return $z = c_1 + c_2$
end function

Proposition 5.3. *The eigenvalues* $\{\nu_1, \ldots, \nu_n\}$ *of* $M_{res}A$ *are*

$$\nu_i = \begin{cases} \lambda_i & \text{if } i > k \\ 1 & \text{if } i \leq k \end{cases}.$$

Proof. Let $U^\perp$, E and F be as in Proposition 5.2. Since we already have the similarity relation of equation (5.11), we only need to find such a relation for M_cAMA:

$$\begin{aligned} M_cAMA[U \,|\, U^\perp] &= M_cA[U \,|\, U^\perp] \begin{pmatrix} J & E \\ 0 & F \end{pmatrix} \\ &= [U \,|\, U^\perp] \begin{pmatrix} I & A_c^{-1}W^HAU^\perp \\ 0 & 0 \end{pmatrix} \begin{pmatrix} J & E \\ 0 & F \end{pmatrix} \\ &= [U \,|\, U^\perp] \begin{pmatrix} J & E + A_c^{-1}W^HAU^\perp F \\ 0 & 0 \end{pmatrix} \end{aligned} \tag{5.12}$$

Combining equations (5.11) and (5.12) yields:

$$(M + M_c - M_cAM)[U \,|\, U^\perp] = [U \,|\, U^\perp] \begin{pmatrix} I & A_c^{-1}W^HAU^\perp(I - F) \\ 0 & F \end{pmatrix}, \tag{5.13}$$

which finishes the proof. □

Proposition 5.4. *The eigenvalues* $\{\nu_1, \ldots, \nu_n\}$ *of* $M_{exa}A$ *are*

$$\nu_i = \begin{cases} \lambda_i & \text{if } i > k \\ 1 & \text{if } i \leq k \end{cases}.$$

Proof. Let $U^\perp$, E and F be as in Proposition 5.2. By combining (5.11) and

$$UJA_c^{-1}W^HA[U\,U^\perp] = [U\,U^\perp] \begin{pmatrix} J & JA_c^{-1}W^HAU^\perp \\ 0 & 0 \end{pmatrix}, \tag{5.14}$$

we find the similarity relation

$$(M + M_c - UJA_c^{-1}W^H)A[U \,|\, U^\perp] = [U \,|\, U^\perp] \begin{pmatrix} I & E + (I - J)A_c^{-1}W^HAU^\perp \\ 0 & F \end{pmatrix},$$

which finishes the proof. □

Other recent spectral preconditioners involve the philosophy of multigrid methods. Multigrid methods were first introduced in the 1970s to solve PDEs discretized on a large mesh and were later generalized and popularized by Brandt, MacCormick and Rüge in the 1980s for general linear systems [32]. They exploit the fact that classical iterative methods quickly damp high frequencies of the error vector. By coarsening the grid, either physically for PDEs or in an algebraic sense in general, the low frequencies can be damped in fewer iterations.

More precisely, a classical iterative method is applied to a linear system for only a few steps to obtain an approximate solution x^*. Subsequently the residual $r = b - Ax^*$ is extrapolated to a coarser grid as $r_c = Rr$ where R is the restriction operator, on which the error equation $A_c e = r_c$ is solved, with A_c the coarse operator. The latter system is relatively small and can be solved by a direct method. Lastly the approximate error is interpolated back to the fine grid, where the solution x^* is updated and an iterative method is applied again for only a few steps to damp the high frequencies of the error. This process is called a V-cycle.

Algorithms 6 and 7 use a given preconditioner M as an approximate inverse of A for a weighted stationary iterative method which acts as a smoother. In this case $A_c = W^H A V_k$ is seen as the coarse space operator, V_k as the interpolation operator and W^H the restriction operator, so that the expression $z \leftarrow z + V(A_c)^{-1} W^H r$ restricts the residual, solves the coarse error equation, interpolates the error back to the fine space and updates the solution.

The additive spectral preconditioner uses a filter operator $(I - UW^H)$ which is constructed to remove the components in the directions of U. A natural way to obtain this is to define W such that $(I - UW^H)U = 0$, which is satisfied when $W^H U = I$.

Lemma 5.1. *Let $x_{k+1} = x_k + \omega M r_k$ define the weighted Richardson iteration for $Ax = b$ with $r_k = b - Ax_k$, $\omega \in \mathbb{C}$ and $x_0 \in \mathbb{C}^n$. Then*

$$x_k = (I - \omega MA)^k x_0 + (I - (I - \omega MA)^k) A^{-1} b$$

Proof. Expanding the expression yields $x_{k+1} = (I - \omega MA)x_k + \omega Mb$. Define $B \equiv I - \omega MA$ and $f \equiv \omega Mb$. The consistency relation $x = Bx + f$ is easily verified for the exact solution x, so that $f = (I - B)A^{-1}b$. By induction on k it follows that $x_k = B^k x_0 + (\sum_{i=0}^{k-1} B^i) f$. By substituting the expression for f, all terms of the sum cancel out, except the first and last one, resulting in $x_k = B^k x_0 + (I - B^k)A^{-1}b$ for all $k = 0, 1, \ldots$. This concludes the proof when $B = I - \omega MA$ is substituted. □

Proposition 5.5. *Let W be such that $(I - UW^H)U = 0$. Then the explicit form of M_{add} as defined in Algorithm 7 is given by*

$$M_{add} = (I - UW^H)(I - (I - \omega MA)^{\mu_1 + \mu_2})A^{-1} + M_c \tag{5.15}$$

Proof. The proof follows by use of Lemma 5.1 and simple substitutions. □

Proposition 5.6. *The eigenvalues of $\{\nu_1, \ldots, \nu_n\}$ of $M_{add}A$ are given as*

$$\nu_i = \begin{cases} 1 & \text{if } i \leq k \\ 1 - (1 - \omega\lambda_i)^{\mu_1 + \mu_2} & \text{if } i > k \end{cases} \tag{5.16}$$

Proof. Let $U^\perp$, E and F be as in Proposition 5.2. We will find expressions for $M_{add}AU$ and $M_{add}AU^\perp$ to show that $M_{add}A$ is similar to a matrix with the desired eigenvalues.

By induction on j it follows that $(I-\omega MA)^j U = U(I-\omega J)^j$ for $j = 1, 2, \ldots$, so that

$$\begin{aligned} M_{add}AU &= (I - UW^H)(I - (I-\omega MA)^{\mu_1+\mu_2})U + U \\ &= (I-UW^H)U - (I-UW^H)U(I-\omega J)^{\mu_1+\mu_2} + U \\ &= U. \end{aligned} \tag{5.17}$$

Also, by noting that $MAU^\perp = UE + U^\perp F$ it is easy to show by induction that $(I - \omega MA)^j U^\perp = U^\perp(I-\omega F)^j + UP_j$ for a matrix $P_j \in \mathbb{C}^{k\times(n-k)}$ for all $j = 1, 2, \ldots$. This allows us to write:

$$\begin{aligned} &M_{add}AU^\perp \\ &= (I-UW^H)(I-(I-\omega MA)^{\mu_1+\mu_2})U^\perp + UA_c^{-1}W^HAU^\perp \\ &= (I-UW^H)U^\perp - (I-UW^H)(U^\perp(I-\omega F)^{\mu_1+\mu_2} + UP_{\mu_1+\mu_2}) + UA_c^{-1}W^HAU^\perp \\ &= U^\perp(I-(I-\omega F)^{\mu_1+\mu_2}) + U((I-W^HU)P_{\mu_1+\mu_2} - W^HU^\perp(I+(I-\omega F)^{\mu_1+\mu_2}) \\ &\quad + A_c^{-1}W^HAU^\perp) \\ &= U^\perp(I-(I-\omega F)^{\mu_1+\mu_2}) + UQ \end{aligned} \tag{5.18}$$

with $Q \in \mathbb{C}^{k\times(n-k)}$ denoting the unimportant expression between brackets, so that when combining equations (5.17) and (5.18)

$$M_{add}A[U \,|\, U^\perp] = [U \,|\, U^\perp] \begin{pmatrix} I & Q \\ 0 & I-(I-\omega F)^{\mu_1+\mu_2} \end{pmatrix} \tag{5.19}$$

defines the similarity relation between $M_{add}A$ and a matrix clearly having the desired eigenvalues. □

Proposition 5.7. *The explicit form of M_{mul} as defined in Algorithm 6 is given by:*

$$M_{mul} = A^{-1} - (I-\omega MA)^{\mu_2}(I-M_cA)(I-\omega MA)^{\mu_1}A^{-1}. \tag{5.20}$$

Proof. The proof follows by applying Lemma 5.1. □

Proposition 5.8. *The eigenvalues of $\{\nu_1, \ldots, \nu_n\}$ of $M_{mul}A$ are given as*

$$\nu_i = \begin{cases} 1 & \text{if } i \le k \\ 1-(1-\omega\lambda_i)^{\mu_1+\mu_2} & \text{if } i > k \end{cases} \tag{5.21}$$

Proof. The proof is very similar to the proof of Proposition 5.6. We refer to [9] for details. □

6 Adaptive GMRES

In the previous section multiple two-level, spectral preconditioners were introduced which are used in cases where a standard preconditioner M fails to cluster all of the eigenvalues away from the origin. All of these spectral preconditioners assume that right eigenvectors associated to the smallest eigenvalues of MA are available. However, since our goal is to solve a single, general linear system, there is no computational gain if the eigenvectors of MA were to be computed prior to the solving phase.

But the GMRES method is particularly suitable to cheaply approximate spectral information of MA, as the Ritz pairs can be computed from the Hessenberg matrix H_m. This approximate spectral information can then be used to implement the spectral preconditioners from section 5.3.

The above process roughly describes the action of a novel family of GMRES variants we will introduce in this section. There are however a few important notes on this procedure. First of all, since the restart value m is usually determined by memory and computational limits, there is no guarantee that the approximate eigenpairs have converged sufficiently well to the eigenpairs of A. Secondly, there is no guarantee that all of the smallest eigenvalues of A are even present in the eigenspectrum of H_m.

To sidestep these difficulties, we only allow approximate eigenvectors that match certain convergence criteria, as stated in section 6.1. Secondly, we compute the eigenpairs of the Hessenberg matrix after each outer iteration of GMRES, to assure that small eigenvalues appearing only after several outer iterations will be deflated as well. This means that we no longer use two-level spectral preconditioning, but multilevel or adaptive spectral preconditioning. We will therefore refer to the method presented in this section as Adaptive GMRES, or A-GMRES(m, j) shortly, where m is the restart value and j the number of Ritz vectors computed per outer iteration.

Let $M^{(i-1)}$ be the left preconditioner at the ith outer iteration of GMRES(m), where $M^{(0)} = M$ is the initial preconditioner. Let $X^{(i)}$ be a matrix consisting of filtered eigenvectors of H_m computed at the end of the ith outer iteration, so that $U^{(i)} \equiv V_m X^{(i)}$ forms an approximate basis for a right-invariant of $M^{(i-1)}A$. Then $M^{(i)}$ can be defined as a two-level spectral preconditioner such as M_{coa}, M_{res}, M_{exa}, M_{add} or M_{mul} where the first level preconditioner is taken as $M^{(i-1)}$. An example of the following procedure is shown in Algorithm 8.

Note. The procedure of Algorithm 8 defines each preconditioner $M^{(i)}$ recursively as a function of the previous preconditioner $M^{(i-1)}$. From a theoretical point of view this allows any combination of spectral preconditioners, but from a pragmatic point of view, the choices are limited. For example, M_{mul} and M_{add} require $\mu_1 + \mu_2$ applications of the first-level preconditioner, so that implementing them after a few outer iterations, increases the computational costs by multiple recursive calls. Therefore, it makes sense to define them only in the first iteration as $M^{(1)}$, which also has the benefit that the eigenspectrum is clustered immediately. From then on, the following preconditioners $M^{(j)}$ will be either M_{coa}, M_{res} or M_{exa} for $j = 2, 3, \ldots$. Taking this into account also allows a non-recursive, procedural implementation, by the nature of M_{coa}, M_{res} and

M_{exa}.

Algorithm 8 Adaptive GMRES

function A-GMRES(m, j; A, b, x_0)
  for $i = 1, 2, \ldots$ **do**
    $r = M^{(i-1)}(b - Ax_0)$
    Perform a single inner iteration of GMRES(m)
    Compute j the smallest eigenpairs (θ_k, x_k) of H_m.
    for $k = 1, \ldots, j$ **do**
      Append x_k as a column to $X^{(i)}$ if $|\lambda_k| < \epsilon_1$ and $\eta(A, V_m x_k) < \epsilon_2$.
    end for
    Compute $U^{(i)} \equiv V_m X^{(i)}$
    Define $W^{(i)}$ such that $(W^{(i)})^H A U^{(i)}$ is non-singular
    Let M_* be a two-level spectral preconditioner, either M_{coa}, M_{res}, M_{mul} or M_{add}
    Define $M^{(i)} \equiv M_*(M^{(i-1)}, U^{(i)}, W^{(i)})$
  end for
end function

6.1 Computation and quality of approximate eigenpairs

As we assume eigenvectors of MA are not given and computing them directly is too expensive, we use Ritz vectors as approximations for the construction of spectral preconditioners. We will both inspect Ritz pairs and harmonic Ritz pairs with measures for their quality.

Note. The eigenpair computation from the Hessenberg matrix H_m should be an inexpensive process. One can either retrieve a fixed number of ℓ eigenpairs corresponding to the smallest eigenvalues of H_m using a sparse matrix eigensolver, but another possibility is to compute the full set of eigenpairs using the QR algorithm. Especially for small or moderate restart values m the latter choice may pay out, since GMRES implicitly decomposes $\hat{H}_m$ into a QR decomposition via Givens rotations. That means that the whole set-up phase for the QR algorithm has already been performed, so that the iterative phase can start immediately.

In this subsection we refer to the coefficient matrix as A for simplicity, but it can be interchanged freely with any preconditioned coefficient matrix MA. This is because the Arnoldi algorithm constructs basis V_m and Hessenberg matrix H_m for a preconditioned system as seen in Algorithm 2.

Proposition 6.1. *All Ritz pairs (θ, y) of A with $y = V_m x$ and (x, θ) an eigenpair of H_m have a residual of*

$$\|Ay - \theta y\|_2 = |h_{m+1,m}||x_m|, \tag{6.1}$$

where x_m is the mth entry of x.

Proof. The proof follows from the third identity of Proposition 3.1:

$$\begin{aligned} Ay - \theta y &= AV_m x - \theta V_m x \\ &= (V_m H_m + h_{m+1,m} v_{m+1} e_m^T) x - \theta V_m x \\ &= V_m (H_m x - \theta x) + h_{m+1,m} v_{m+1} e_m^T x \\ &= h_{m+1,m} v_{m+1} e_m^T x \\ &= h_{m+1,m} x_m v_{m+1}. \end{aligned} \tag{6.2}$$

Since v_{m+1} has unit Euclidean norm by construction, it follows that

$$\|Ay - \theta y\|_2 = |h_{m+1,m}||x_m|, \tag{6.3}$$

which concludes the proof. □

Proposition 4.2 tells that given a Ritz pair (θ, x), the pair $(\theta, V_m x)$ is likely to be a good approximation of an eigenpair of A if $h_{m+1,m}$ and the mth component of the Ritz vector x are small. On top of that, the practical power of Proposition 4.2 is that it provides a way to compute the eigenproblem residual without involving expensive matrix products.

But the residual does not provide sufficient quantitative insight to the reliability of the approximation if the eigenpair is ill-conditioned; that is, if $(\tilde{\lambda}, \tilde{x})$ denotes an approximation to an exact eigenpair (λ, x), it is possible that the normed eigenresidual $\|A\tilde{x} - \tilde{\lambda}\tilde{x}\|$ is small, while the distances $\|x - \tilde{x}\|$ or $|\lambda - \tilde{\lambda}|$ are large. Therefore it is better to use a different measure for the quality, for instance the backward error. Multiple definitions are possible, but in this thesis we stick to the definition where the backward error is defined as the minimal size of a relative perturbation δA of A, so that the Ritz pair is the exact eigenpair for the perturbed coefficient matrix $A + \delta A$.

Definition 6.1. Given a matrix $A \in \mathbb{C}^{n \times n}$ and a vector $x \in \mathbb{C}^n$, the *Rayleigh quotient* is defined as

$$R(A, x) = \frac{x^H A x}{x^H x} \tag{6.4}$$

Clearly for an exact eigenpair (x, λ) of a matrix A, it holds that $R(A, x) = \lambda$. But also for a Ritz pair we have such an identity.

Proposition 6.2. *Let (θ, y) be a Ritz pair with respect to $\mathcal{K}_m(A, r_0)$, where $y = V_m x$. Then $R(A, y) = \theta$.*

Proof. Since (x, θ) is an exact eigenpair of H_m, it follows that

$$R(A, y) = \frac{y^H A y}{y^H y} = \frac{x^H V_m^H A V_m x}{x^H V_m^H V_m x} = \frac{x^H H_m x}{x^H x} = \theta. \tag{6.5}$$

□

Definition 6.2. The *backward error of an approximate eigenvector $\tilde{x}$ of A* is defined as

$$\eta(A,\tilde{x}) = \min_{\delta A}\left\{\tau > 0 : \frac{\|\delta A\|}{\|A\|} \leq \tau : (A+\delta A)\tilde{x} = R(A,\tilde{x})\tilde{x}\right\}$$

Proposition 6.3. *Definition 6.2 can be expressed explicitly as*

$$\eta(A,\tilde{x}) = \frac{\|A\tilde{x} - R(A,\tilde{x})\tilde{x}\|}{\|A\|\,\|\tilde{x}\|}.$$

Proof. In the proof the quantity $\|\delta A\| / \|A\|$ is bounded from below; subsequently it is shown that the bound is attained for a special choice of δA. From the equality $(A+\delta A)\tilde{x} = R(A,\tilde{x})\tilde{x}$ it follows by taking norms that

$$\|\delta A\|\,\|\tilde{x}\| \geq \|R(A,\tilde{x})\tilde{x} - A\tilde{x}\|.$$

Thus

$$\frac{\|\delta A\|}{\|A\|} \geq \frac{\|R(A,\tilde{x})\tilde{x} - A\tilde{x}\|}{\|A\|\,\|\tilde{x}\|}. \tag{6.6}$$

Now take $\delta A = \left\|(R(A,\tilde{x})\tilde{x} - A\tilde{x})\tilde{x}^T\right\| / \|\tilde{x}\|^2$. Then per definition it follows that

$$\|\delta A\| = \frac{1}{\|\tilde{x}\|^2}\max_{z\neq 0}\frac{\left\|(R(A,\tilde{x})\tilde{x} - A\tilde{x})\tilde{x}^T z\right\|}{\|z\|}.$$

Let α denote the angle between $\tilde{x}$ and z. Since $\tilde{x}^T z$ is a scalar value, it follows that

$$\begin{aligned}\frac{\|\delta A\|}{\|A\|} &= \frac{\|R(A,\tilde{x})\tilde{x} - A\tilde{x}\|}{\|A\|\,\|\tilde{x}\|^2}\max_{z\neq 0}\frac{|\tilde{x}^T z|}{\|z\|}\\ &= \frac{\|R(A,\tilde{x})\tilde{x} - A\tilde{x}\|}{\|A\|\,\|\tilde{x}\|^2}\max_{\alpha\in[0,\pi]}\|\tilde{x}\|\,|\cos\alpha|\\ &= \frac{\|R(A,\tilde{x})\tilde{x} - A\tilde{x}\|}{\|A\|\,\|\tilde{x}\|}\end{aligned}$$

Thus the bound of equation 6.6 is attained, so that

$$\eta(A,\tilde{x}) = \frac{\|A\tilde{x} - R(A,\tilde{x})\tilde{x}\|}{\|A\|\,\|\tilde{x}\|}$$

as desired. □

By using 2-norms in Proposition 6.3, we can use Proposition 6.1 to express the backward error for a Ritz vector $y = V_m x$ of A, with x an eigenvector of H_m, as $\eta(A, V_m x) = |h_{m+1,m}||x_m|/(\|A\|_2\,\|V_m x\|_2)$. The spectral norm $\|A\|_2$ is however expensive to retrieve, since it requires the computation of the largest singular value of the $n \times n$ matrix A. To remedy this, one can bound the norm from below with the spectral norm of H_m.

Proposition 6.4. *Let H_m and V_m be generated from m steps of the Arnoldi algorithm, and let $(\theta, V_m x)$ be a Ritz pair of A,where (θ, x) is an eigenpair of H_m and x is assumed to be normalized in the 2-norm. Then*

$$\eta(A, V_m x) \leq \frac{|h_{m+1,m}||x_m|}{\|H_m\|_2}$$

.

Proof. Note that $\|H_m\|_2 = \left\|V_m^H A V_m\right\|_2 \leq \left\|V_m^H\right\|_2 \|A\|_2 \|V_m\|_2$. But $\|V_m\|_2 = 1$ since it is orthogonal and if v_1 denotes the first column of V:

$$\left\|V_m^H\right\|_2 = \sup_{x \neq 0} \frac{\left\|V_m^H x\right\|_2}{\|x\|_2} \geq \frac{\left\|V_m^H v_1\right\|_2}{\|v_1\|_2} = 1.$$

Thus $\|H_m\|_2 \leq \|A\|_2$. Furthermore

$$\eta(A, V_m x) = \frac{|h_{m+1,m}||x_m|}{\|A\|_2 \|V_m x\|_2}.$$

Since V_m is orthogonal by construction and $\|x\|_2 = 1$, we have that

$$\eta(A, V_m x) \leq \frac{|h_{m+1,m}||x_m|}{\|H_m\|_2}$$

□

The bound of 6.4 provides a cheap bound which only requires the computation of the spectral norm of the Hessenberg matrix.

However, in [26] it is suggested that Ritz values converge more regular for exterior than interior eigenvalues. A solid argument is only given for SPD matrices, but the behaviour has been observed for general matrices as well. One remedy is to find the exterior eigenvalues of A^{-1}, but that would result in circularities since it would either require inversion of A or result in another linear system involving A. Yet we proceed by defining harmonic Ritz values as the Ritz values of the inverse of A, and by applying a skew projection to eliminate the need of solving linear systems, in the lines of [31].

Definition 6.3. A pair (θ, g) is a *harmonic Ritz pair of A with respect to $\mathcal{V}$*, if (θ^{-1}, g) is a Ritz pair of A^{-1} with respect to $\mathcal{V}$.

Proposition 6.5. *A pair (θ, g) is a harmonic Ritz pair of A with respect to $\mathcal{W} = A\mathcal{V}$ if and only if*

$$Au - \theta u \perp A\mathcal{V}$$

for a $u \in \mathcal{V}$.

Proof. By definition it holds that $g = AVz$ for a $z \in \mathbb{C}^k$, so that

$$V^H A^H (A^{-1} AVz - \theta^{-1} AVz) = 0 \Leftrightarrow V^H A^H (AVz - \theta Vz) = 0. \tag{6.7}$$

Define $u \equiv Vz$, so that equation 6.7 reads $Au - \theta u \perp A\mathcal{V}$. □

Proposition 6.5 can be simplified in our case where V_m is generated by the Arnoldi algorithm. We have $V_m^H A^H A V_m z = \theta V_m^H A^H V_m z$ for a $z \in \mathbb{C}^k$. By the identities of Proposition 3.1, we find for the right-hand side $V_m^H A V_m = H_m$ and for the left-hand side

$$\begin{aligned} V_m^H A^H A V_m &= (AV_m)^H A V_m \\ &= (V_m H_k + h_{m+1,m} v_{m+1} e_m^T)^H (V_m H_m + h_{m+1,m} v_{m+1} e_m^T) \\ &= H_m^H H_m + |h_{m+1,m}|^2 e_m e_m^T, \end{aligned} \tag{6.8}$$

so that if we define $f \equiv (H_m)^{-H} e_m$, the expression becomes

$$\left(H_m + |h_{m+1,m}|^2 f e_m^T\right) z = \theta z. \tag{6.9}$$

Unlike the case of ordinary Ritz pairs, the backward error for harmonic Ritz pairs does not have a very elegant and cheap expression. We restate the upper bound from [18], where $V_m x$ is again assumed to be normalized in the 2-norm:

$$\eta(A, V_m x) \leq \frac{|h_{m+1,m}||x_m|}{\|H_m\|_2} \sqrt{|h_{m+1,m}^2 \|x^H f x - f\|_2^2 + 1}. \tag{6.10}$$

Note. The rank-one update $H_m + |h_{m+1,m}|^2 H_m^{-H} e_m e_m^T$ only updates the last column of H_m and therefore does not destroy the Hessenberg structure of the matrix. This implies that the complexity of the computation of harmonic Ritz pairs is retained with respect to ordinary Ritz pairs, if the QR algorithm is applied.

6.2 Computational costs

We will briefly summarize the computational costs of adaptive GMRES. Since adaptive GMRES extends restarted GMRES, we will only focus on the additional costs. These consist of computation of spectral information from the Hessenberg matrix at each outer iteration, a computation for the accuracy for each eigenvector, construction of each level of the preconditioner and lastly the cost of applying each level of the preconditioner.

We assume that the computational effort for the spectral information is not a main issue, as it was pointed out in multiple experiments that spectral preconditioners are useful if the backward error in the computation of eigenpairs of H_m itself is around 10^{-3} (see for instance [8]). Therefore, the number of iterations to compute the spectral information should not be very large.

For the check on the quality of the Ritz pairs, the spectral norm $\|H_m\|_2$ is to be computed once at each outer iteration. This can be implemented using a couple iterations of the power method. Also, this computation does not have to be performed in high accuracy.

Lastly, the construction and application of each level of a spectral preconditioner is the most expensive additional work. For its construction, computation of $A_c^{(i)} = (W^{(i)})^H A U^{(i)}$ is necessary at each outer iteration, which consists of two matrix-matrix products.

A reduction of complexity of this computation can be obtained for the M_{coa} and M_{exa} preconditioners, by choosing $W^{(i)} = (M^{(i-1)})^H U^{(i)}$ under the assumption $A_c = (W^{(i)})^H A U^{(i)}$ is invertible. Let $U^{(i)} = V_m X^{(i)}$ where $X \in \mathbb{C}^{m \times j}$ consists of j eigenvectors of the matrix H_m and let $\Theta^{(i)} \in \mathbb{C}^{j \times j}$ be a diagonal matrix consisting of the Ritz values such that $H_m X^{(i)} = X^{(i)} \Theta^{(i)}$. Then it follows that

$$A_c = (W^{(i)})^H A U^{(i)} = U^{(i)} M^{(i-1)} A U^{(i)} = (X^{(i)})^H H_m X^{(i)} = (X^{(i)})^H X^{(i)} \Theta^{(i)}. \quad (6.11)$$

When using harmonic Ritz values, let the columns of $X^{(i)}$ be the eigenvectors of $(H_m + |h_{m+1,m}|^2 f e_m^T)$ and the diagonal matrix $\Theta^{(i)}$ the corresponding harmonic Ritz values, so that $(H_m + |h_{m+1,m}|^2 f e_m^T) X^{(i)} = X^{(i)} \Theta^{(i)}$. Then

$$A_c = (X^{(i)})^H X^{(i)} \Theta^{(i)} - |h_{m+1,m}|^2 (X^{(i)})^H f e_m^T X^{(i)}. \quad (6.12)$$

The implementation for this choice of $W^{(i)}$ is quite simple: we use proposition 5.1 to write the preconditioner $M^{(i)}$ as

$$\begin{aligned} M^{(i)} &= M^{(i-1)} + M_c \\ &= M^{(i-1)} + U^{(i)} A_c^{-1} U^{(i)} M^{(i-1)} \\ &= (I + U^{(i)} A_c^{-1} U^{(i)}) M^{(i-1)}. \end{aligned}$$

Thus the ith level can be applied separately from the other levels. However, it is not guaranteed that for this choice of $W^{(i)}$, the matrix A_c is always non-singular.

In application, the least expensive and therefore most promising preconditioners are M_{coa} and M_{exa}, which do not require matrix-vector products involving A. Furthermore, M_{res} requires a single matrix-vector product, M_{add} requires $\mu_1 + \mu_2$ matrix-vector products and lastly M_{mul} requires $\mu_1 + \mu_2 + 1$ matrix-vectors products.

Each outer iteration i requires m applications of the preconditioner $M^{(i)}$ during the Arnoldi process. However, the number of matrix-vector products to apply the preconditioner $M^{(i)}$ grows approximately linearly with the number of outer iterations, if M_{res}, M_{add} or M_{mul} are chosen.

6.3 Implementation details

If the coefficient matrix A is real, there may exist conjugate eigenvectors pairs $v_1 = \bar{v}_2$, causing additional computational costs in complex arithmetic. This can easily be circumvented by replacing these vectors by a real basis for the space they span: $v_1 \leftarrow \mathrm{Re}(v_1)$ and $v_2 \leftarrow \mathrm{Im}(v_1)$.

In practice the preconditioner $M^{(i)}$ is not assembled, but all necessary data — $A_c^{(q)}$, $U^{(i)}$, $W^{(i)}$ and in the case of M_{exa} the Ritz values — is pushed to a stack. Whenever a vector is to be preconditioned, it is passed to a routine that iterates over the levels $1, \ldots, i$ and applies each level of the preconditioner using the data from the stack. This is a non-recursive procedure.

7 Results

In this section we will focus on the performance of adaptive GMRES with different preconditioning techniques, compared to GMRES-DR. We will first see a detailed example, secondly we will focus on the number of iterations required for convergence for several test problems, and lastly we will try to reduce the amount of work in terms of matrix-vector products as much as possible.

Four different coefficient matrices A arising from real-world applications are used, borrowed from the Matrix Market[1] collection. The matrix properties are listed in Table 1.

Table 1: Test matrix properties

Name	Type	Size	$nnz(A)$	$\kappa_2(A)$	Discipline
`GRE1107`	Real antisymmetric	1107×1107	5664	$9.7 \cdot 10^6$	Simulation in computer systems
`HOR131`	Real antisymmetric	434×434	4710	$1.3 \cdot 10^5$	Flow in networks
`ORSIRR1`	Real antisymmetric	1030×1030	6858	?	Oil reservoir simulation
`MEMPLUS`	Real antisymmetric	17758×17758	126150	?	Electronic circuit design

For each test problem, we set the unknown x to a constant column vector 1, which fixes the right-hand side b. Furthermore, the stopping criterion is such that the scaled residual of the unpreconditioned system $\|b - Ax^*\|_2 / \|b\|_2 < \epsilon$ where $\epsilon = 10^{-10}$ is the tolerance and x^* the approximate solution. This stopping criterion is independent from the preconditioner and therefore allows a reliable comparison between each method.

For Adaptive GMRES, five different sequences of spectral preconditioners are considered as listed in Table 2. If not otherwise mentioned, we define $W^{(i)} = U^{(i)}$ at all levels $i = 1, 2, \ldots$. The number of Ritz pairs computed from the Hessenberg matrix per outer iterations varies at each test between 0 and 6. The Ritz vectors are filtered based on the location of the Ritz values in the complex plane and their backward error. In all tests the Ritz values must lie within a disc of radius 0.1 around the origin, and the threshold for the backward error is problem-dependent. If the number of Ritz pairs is 0, the method coincides with ordinary restarted GMRES. The Ritz values are computed with a backward error of only 10^{-3} with respect to the Hessenberg matrix itself, using the `eigs` command of Matlab, which is a wrapper for ARPACK.

To start with a detailed example of the algorithms, we inspect a single solve of $Ax = b$ with A the `HOR131` matrix, using Adaptive GMRES with a restart value 30. First an incomplete factorization ILUT(0.2) is applied to A to obtain the initial preconditioner $M^{(0)}$. The eigenvalue distribution of $M^{(0)}A$ is shown in Figure 5a. Clearly, several eigenvalues are still close to the origin, blocking the rate of convergence of GMRES(30).

[1] All these matrices are freely available at `http://math.nist.gov/MatrixMarket/`.

Table 2: Names for a family of Adaptive GMRES methods, depending on the sequence of spectral preconditioner used at each level

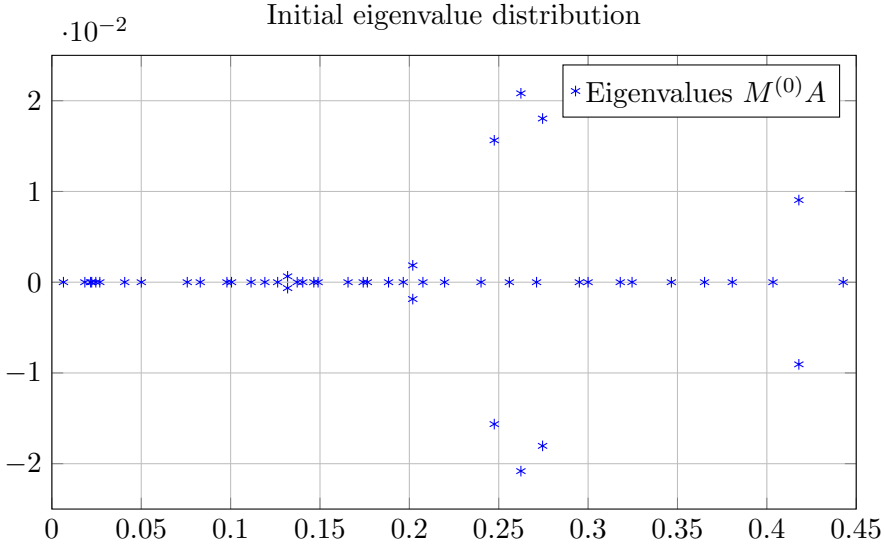
Name	$M^{(1)}$	$M^{(2)}$	$M^{(3)}$	$M^{(4)}$	$\cdots$
A-GMRES M_{coa}	M_{coa}	M_{coa}	M_{coa}	M_{coa}	$\cdots$
A-GMRES M_{exa}	M_{exa}	M_{exa}	M_{exa}	M_{exa}	$\cdots$
A-GMRES $M_{mul+res}$	$M_{mul}(\mu_1 = 1, \mu_2 = 1, \omega = 2/3)$	M_2	M_2	M_2	$\cdots$
A-GMRES $M_{add+coa}$	$M_{add}(\mu_1 = 1, \mu_2 = 1, \omega = 2/3)$	M_1	M_1	M_1	$\cdots$
A-GMRES $M_{mul+coa}$	$M_{mul}(\mu_1 = 1, \mu_2 = 1, \omega = 2/3)$	M_1	M_1	M_1	$\cdots$

However, using A-GMRES$(30, 2)$ with M_{coa} for each level of the spectral preconditioner and 10^{-3} as an upperbound for the backward error of the Ritz vectors, we are able to shift small eigenvalues from the third outer iteration on, as shown in Table 3, resulting in a convergence within 5 outer iterations. In the first outer iteration, the Ritz pairs vectors have not converged sufficiently to satisfy the backward error bound. In the second iteration both Ritz pairs have converged sufficiently, so that they are deflated in the third outer iteration. The effect is immediately visible in Figure 5b, where inner iterations are plotted against the norm of the residual. It is also clearly visible that Adaptive GMRES with M_{coa} converges similarly to GMRES-DR in terms of number of iterations. The whole eigenspectrum $M^{(5)}A$ is plotted in Figures 6a and a close up around the origin is given in Figure 7a. In the latter figure it is clearly visible that small eigenvalues have successfully been deflated, and in the first figure it is visible that the rest of the spectrum has been transformed as well, because approximate eigenvectors are used.

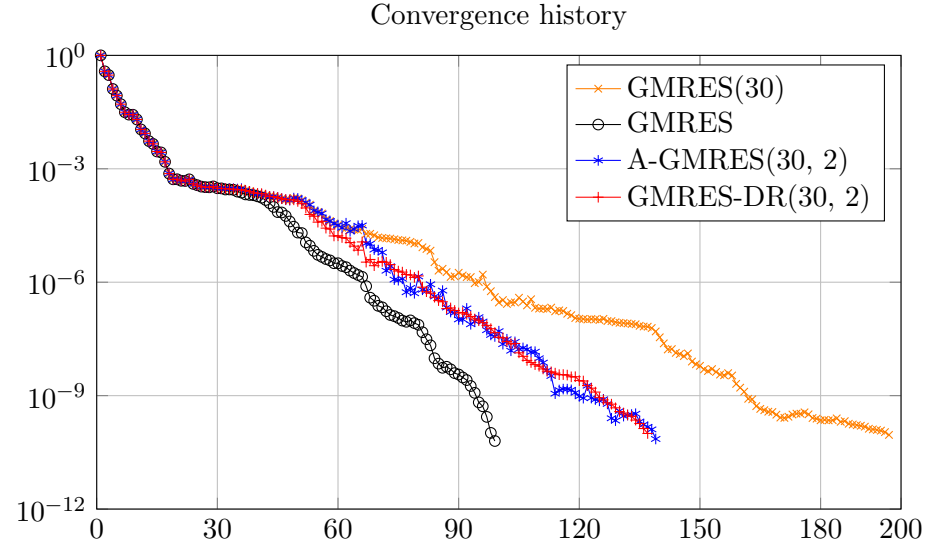
Figures 6b and 7b show similarly the eigenspectra when Adaptive GMRES with $M_{mul+res}$ is applied to A with $\mu_1 = 2$ and $\mu_1 = 1$. In that case, convergence is obtained in just 2 outer iterations.

Table 3: `HOR131`*: GMRES(30) with ILUT*$(2 \cdot 10^{-1})$ *and* M_{coa} *as preconditioner and* $\eta = 0.001$*.* † *and* ✓ *denote respectively if an eigenvalue is used or not to improve the preconditioner*

Iteration	1		2		3		4		5	
Eigenvalue	$6.16 \cdot 10^{-2}$	$6.88 \cdot 10^{-2}$	$7.76 \cdot 10^{-3}$	$1.84 \cdot 10^{-2}$	$2.23 \cdot 10^{-2}$	$3.70 \cdot 10^{-2}$	$2.46 \cdot 10^{-2}$	$4.46 \cdot 10^{-2}$	$2.39 \cdot 10^{-2}$	$4.84 \cdot 10^{-2}$
$\|MAx - \lambda x\|_2$	$4.49 \cdot 10^{-2}$	$5.04 \cdot 10^{-2}$	$1.91 \cdot 10^{-3}$	$4.35 \cdot 10^{-3}$	$1.53 \cdot 10^{-3}$	$1.04 \cdot 10^{-3}$	$1.35 \cdot 10^{-3}$	$6.00 \cdot 10^{-3}$	$5.19 \cdot 10^{-4}$	$3.14 \cdot 10^{-3}$
Backw. err.	$9.91 \cdot 10^{-3}$	$9.91 \cdot 10^{-3}$	$4.22 \cdot 10^{-4}$	$9.61 \cdot 10^{-4}$	$3.22 \cdot 10^{-4}$	$2.19 \cdot 10^{-4}$	$2.59 \cdot 10^{-4}$	$1.15 \cdot 10^{-3}$	$9.82 \cdot 10^{-5}$	$5.94 \cdot 10^{-4}$
Used	†	†	✓	✓	✓	✓	✓	†	✓	✓



(a) Eigenvalue distribution of `HOR131` preconditioned with ILUT(0.2).



(b) Convergence history corresponding to table 3. A-GMRES and GMRES-DR show similar convergence behaviour; both methods convergence significantly faster with respect to restarted GMRES. The first outer iteration coincides as expected, and the second A-GMRES run coincides with GMRES(m) since no eigenvectors are used in the first outer cycle.

Figure 5: Initial eigenvalue distribution and convergence history of `HOR131`

Figure 6: Eigenvalue distribution of `HOR131` *with* $ILUT(0.2)$ *as initial preconditioner and an approximate maximum backward error of* $\eta \approx 10^{-3}$*; restart after* 30 *iterations, shifting 2 eigenvalues at a time*

(a) M_{coa} (b) $M_{add+coa}$, $\mu_1 = 1$, $\mu_2 = 2$

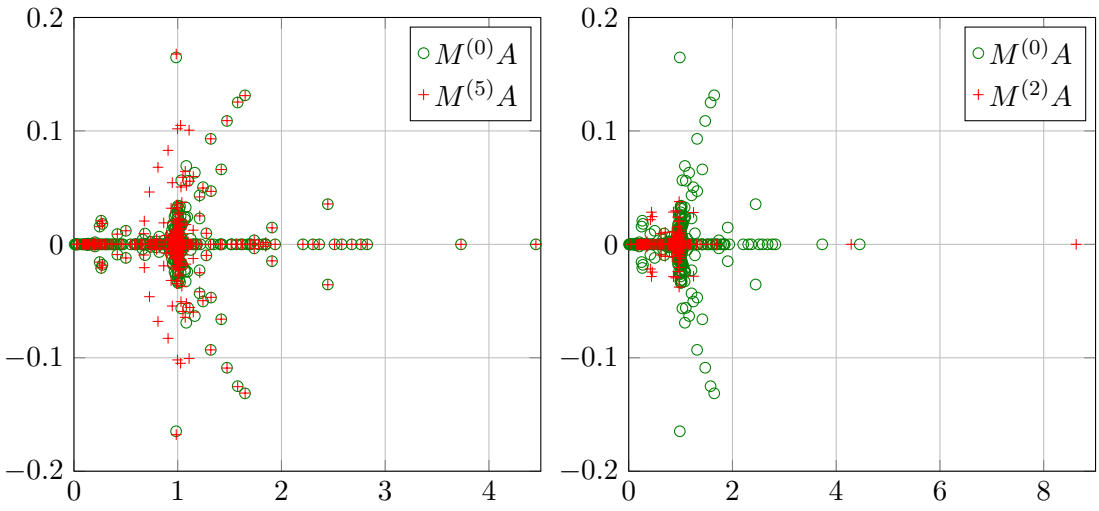
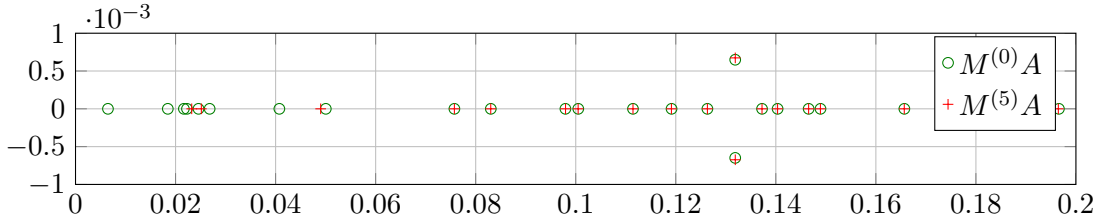
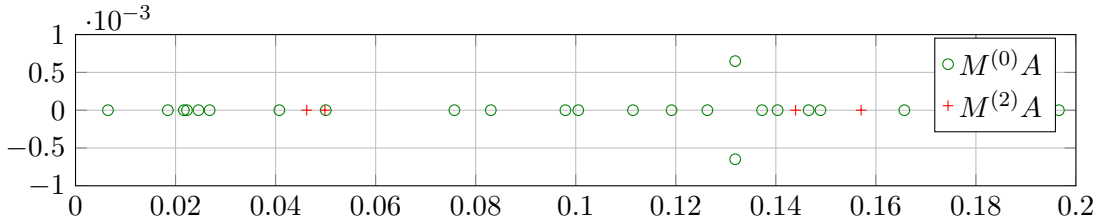


Figure 7: Close-ups of the origin of Figure 6

(a) M_{coa}



(b) $M_{add+coa}$, $\mu_1 = 1$, $\mu_2 = 2$



To give more extensive results, we provide detailed information on the performance of Adaptive GMRES in tables 4, 5, 6 and 7, where both the total number of inner iterations and total matrix-vector products with A are considered for different restart values and

number of Ritz vectors. Since matrix-vector products with A are the most expensive operations, they provide a good measure for the actual performance.

The `GRE1107` matrix is of particular interest, as shown in Table 4. It is considered as a tough matrix, because the ILUT preconditioner does perform too well in clustering the eigenvalues away from the origin. It is clear that GMRES(m) does not converge when only the initial preconditioner is used. Also, GMRES-DR does not converge for small restart values and shows unpredictable behaviour for larger restart values. Even when the restart number is increased to 40, all members of the family of Adaptive GMRES methods performs better with respect to number of iterations. When it comes to the number of matrix-vector products, M_{coa} performs significantly best of all. The multiplicative and additive preconditioner allow convergence for smaller restart values, but their gain in iteration steps comes at a significantly higher computational cost with respect to GMRES-DR and A-GMRES M_{coa}. Lastly, the unpredictable behaviour of GMRES-DR for this example might illustrate the need for quality checks of the Ritz vectors, since A-GMRES converges clearly more regular.

In the case of the `MEMPLUS` matrix, simple incomplete preconditioners such as ILUT result either in zero-pivots, ill-conditioned factors or high memory usage. Therefore the matrix is initially reordered and preconditioned with a multilevel inverse-based factorization of ILUPACK, using a drop tolerance of 1 for the standard elimination process, a threshold of 0.1 for the approximate Schur complement and a bound of 10^6 on the inverse triangular factors. Again, the M_{coa} preconditioner has very similar convergence behaviour to GMRES-DR concerning iterations and costs, as seen in Table 7.

Lastly, it is observed in all examples that $M_{mul+coa}$ shows very similar convergence behaviour as $M_{mul+res}$. The exact shift does not pay out, making $M_{mul+coa}$ a better option. In fact, since $M_{add+coa}$ shows very similar convergence behaviour to $M_{mul+coa}$, it is the best option from the multiplicative and additive variants, as it omits an additional matrix-vector product per inner iteration.

7.1 Reducing the number of matrix-vector products

In order to reduce the total number of matrix-vector products, we incorporate the idea of equation (6.12) combined with a cheap computation of the residual at each restart. The first idea saves a maximum number of j matrix-vector products per outer iteration, where j is the total number of Ritz vectors used. For the previous results of tables 4, 5, 6 and 7, the ith residual at the ith outer iteration was computed as $r_0^{(i)} = M^{(i)} \left(b - Ax_0^{(i)}\right)$. This computation is reliable, but requires a matrix-vector product and application of i levels of preconditioners. A cheaper option is to compute the residual preconditioned by $M^{(i-1)}$ using the identity

$$M^{(i-1)}(b - Ax^{(i-1)}) = r_0^{(i-1)} - V_{m+1}\hat{H}_m z, \tag{7.1}$$

where z is the solution to the least squares problem. The quantity of equation (7.1) has to be preconditioned only with the ith level of $M^{(i)}$. This reduces the number of matrix-vector products by 1 per outer iteration and saves the application of $(i-1)$ levels

of the preconditioner at each ith iteration. In return, it might result in propagation of rounding errors; therefore more iterations may be necessary for convergence.

By using either the spectral preconditioner M_{coa} or M_{exa} for each level of Adaptive GMRES, the total number of matrix-vector products equals the number of inner iterations plus 1, where the last product is due to the computation of the initial residual. This coincides with the number of matrix-vector products of GMRES-DR, so that we only have to compare the total amount of total inner iterations.

Tables 8, 9 and 10 show the number of iterations of the above mentioned implementations in comparison to GMRES-DR for the `GRE1107`, `ORSIRR1` and `MEMPLUS` matrices respectively. The first observation is that in the case of the `GRE1107` matrix, these implementations of Adaptive GMRES do not convergence for a restart value of 30, unlike the results of Table 4. However, for $m = 35$ with more than 1 Ritz vector and $m = 40$ for any number of Ritz vectors, the results are almost identical to Table 4. For the `ORSIRR1` and `MEMPLUS` matrices, the convergence is almost identical to the previous results as well. This makes Adaptive GMRES with M_{coa} and M_{exa} very promising methods.

Table 4: Number of iterations/matrix-vector products for `GRE1107` *with* $M^{(0)}$ *defined as* $ILUT(0.01)$ *and* $\eta = 0.01$. *'DR' refers to GMRES-DR and 'A' to Adaptive GMRES.* † *means no convergence within 500 iterations was obtained.*

	Dimension eigenspace						
Method	0	1	2	3	4	5	6
DR(25)	†	†	†	†	†	†	†
A(25) $M_{add+coa}$	†	†	†	†	†	†	†
A(25) $M_{mul+coa}$	†	277/697	269/661	269/661	269/661	270/665	269/661
A(25) $M_{mul+res}$	†	269/875	268/868	269/875	269/875	268/868	268/868
A(25) M_{coa}	†	†	†	†	†	†	†
DR(30)	†	†	†	†	†	†	155/156
A(30) $M_{add+coa}$	†	120/315	90/222	90/222	90/222	90/222	90/222
A(30) $M_{mul+coa}$	†	118/400	115/388	115/388	115/388	115/388	116/392
A(30) $M_{mul+res}$	†	117/483	107/405	114/440	110/420	114/440	114/440
A(30) M_{coa}	†	330/351	234/250	234/250	234/250	234/250	234/250
DR(35)	†	†	†	†	123/124	†	†
A(35) $M_{add+coa}$	†	70/146	70/146	70/146	70/146	70/146	70/146
A(35) $M_{mul+coa}$	†	70/182	70/182	70/182	70/182	70/182	70/182
A(35) $M_{mul+res}$	†	70/182	70/182	70/182	70/182	70/182	70/182
A(35) M_{coa}	†	166/178	105/113	105/113	107/117	105/113	107/117
DR(40)	†	225/226	112/113	112/113	104/105	†	92/93
A(40) $M_{add+coa}$	†	77/156	74/149	74/149	73/147	73/147	73/147
A(40) $M_{mul+coa}$	†	77/194	73/180	73/180	71/173	71/173	71/173
A(40) $M_{mul+res}$	†	77/194	73/180	73/180	71/173	71/173	71/173
A(40) M_{coa}	†	155/162	110/118	106/115	80/86	80/86	80/86

Table 5: Number of iterations/matrix-vector products for `ORSIRR1` *with* $M^{(0)}$ *defined as* $ILUT(0.05)$ *and* $\eta = 0.001$. *'DR' refers to GMRES-DR and 'A' to Adaptive GMRES.* † *means no convergence within 500 iterations was obtained.*

	Dimension eigenspace					
Method	0	1	2	3	4	5
DR(10)	104/105	97/98	85/86	86/87	87/88	84/85
A(10) $M_{add+coa}$	104/115	59/154	59/154	59/154	59/154	59/154
A(10) $M_{mul+coa}$	104/115	59/197	59/197	59/197	59/197	59/197
A(10) $M_{mul+res}$	104/115	58/222	58/222	58/222	58/222	58/222
A(10) M_{coa}	104/115	82/95	82/95	82/95	82/95	82/95
DR(15)	98/99	92/93	84/85	80/81	77/78	77/78
A(15) $M_{add+coa}$	98/105	56/151	54/146	54/146	54/146	54/146
A(15) $M_{mul+coa}$	98/105	55/191	54/188	54/188	54/188	54/188
A(15) $M_{mul+res}$	98/105	55/229	53/218	53/218	53/218	53/218
A(15) M_{coa}	98/105	77/88	77/88	77/88	77/88	77/88
DR(20)	93/94	88/89	82/83	77/78	77/78	75/76
A(20) $M_{add+coa}$	93/98	57/140	55/136	55/136	55/136	55/136
A(20) $M_{mul+coa}$	93/98	57/179	54/169	54/169	54/169	54/169
A(20) $M_{mul+res}$	93/98	56/192	54/184	54/184	54/184	54/184
A(20) M_{coa}	93/98	76/83	75/85	75/85	75/85	75/85

Table 6: Number of iterations/matrix-vector products for `HOR131` *with* $M^{(0)}$ *defined as* $ILUT(0.2)$ *and* $\eta = 0.001$. *'DR' refers to GMRES-DR and 'A' to Adaptive GMRES.* † *means no convergence within 500 iterations was obtained.*

	Dimension eigenspace					
Method	0	1	2	3	4	5
DR(20)	235/236	†	152/153	146/147	140/141	†
A(20) $M_{add+coa}$	235/247	113/234	113/234	113/234	113/234	113/234
A(20) $M_{mul+coa}$	235/247	112/286	112/286	112/286	112/286	112/286
A(20) $M_{mul+res}$	235/247	110/321	110/321	110/321	110/321	110/321
A(20) M_{coa}	235/247	148/158	148/158	148/158	148/158	148/158

Table 7: Number of iterations/matrix-vector products for `MEMPLUS` *with* $M^{(0)}$ *an ILUPACK preconditioner and* $\eta = 0.01$

	Dimension eigenspace						
Method	0	1	2	3	4	5	6
DR(20)	196/197	190/191	180/181	173/174	156/157	122/123	120/121
A(20) $M_{add+coa}$	196/206	122/308	121/307	109/269	101/246	101/246	101/246
A(20) $M_{mul+coa}$	196/206	120/382	101/310	101/311	94/277	94/277	94/277
A(20) $M_{mul+res}$	196/206	120/508	101/379	101/380	94/328	94/328	94/328
A(20) M_{coa}	196/206	151/165	121/137	121/138	121/138	121/138	121/138
DR(40)	170/171	161/162	157/158	147/148	123/124	117/118	108/109
A(40) $M_{add+coa}$	170/176	101/256	83/200	81/196	78/189	78/189	78/189
A(40) $M_{mul+coa}$	170/176	101/330	83/255	78/237	77/235	77/235	77/235
A(40) $M_{mul+res}$	170/176	101/385	83/279	78/256	76/248	76/248	76/248
A(40) M_{coa}	170/176	147/156	121/134	105/117	103/116	103/116	103/116

Table 8: Number of iterations for `GRE1107` *with* $M^{(0)}$ *defined as* $ILUT(0.01)$ *and* $\eta = 0.01$. *'DR' refers to GMRES-DR and 'A' to Adaptive GMRES, using* $(W^{(i)})^H = (M^{(i)})^H U^{(i)}$ *and equation* (6.12). † *means no convergence within 500 iterations was obtained.*

	Dimension eigenspace					
Method	1	2	3	4	5	6
DR(30)	†	†	†	†	†	155
A(30) M_{coa}	†	†	†	†	†	†
A(30) M_{exa}	†	†	†	†	†	†
DR(35)	†	†	†	123	†	†
A(35) M_{coa}	†	132	105	105	105	105
A(35) M_{exa}	†	132	104	104	104	104
DR(40)	225	112	112	104	†	92
A(40) M_{coa}	146	112	99	79	79	79
A(40) M_{exa}	144	112	100	79	79	79

Table 9: Number of iterations for `ORSIRR1` *with* $M^{(0)}$ *defined as* $ILUT(0.05)$ *and* $\eta = 0.01$. *'DR' refers to GMRES-DR and 'A' to Adaptive GMRES, using* $(W^{(i)})^H = (M^{(i)})^H U^{(i)}$ *and equation* (6.12).

	Dimension eigenspace				
Method	1	2	3	4	5
DR(10)	97	85	86	87	84
A(10) M_{coa}	85	85	85	85	85
A(10) M_{exa}	85	85	85	85	85
DR(15)	92	84	80	77	77
A(15) M_{coa}	98	81	80	80	80
A(15) M_{exa}	98	80	79	79	79
DR(20)	88	82	77	77	75
A(20) M_{coa}	77	75	75	75	75
A(20) M_{exa}	77	75	75	75	75

Table 10: Number of iterations for `MEMPLUS` *with* $M^{(0)}$ *an ILUPACK preconditioner and* $\eta = 0.01$

	Dimension eigenspace					
Method	1	2	3	4	5	6
DR(20)	190	180	173	156	122	120
A(20) M_{coa}	192	129	125	125	125	125
DR(40)	161	157	147	123	117	108
A(40) M_{coa}	134	129	115	105	105	105

8 Conclusion and discussion

The novel family of GMRES variants 'Adaptive GMRES' which is based on approximate, multilevel spectral preconditioners, turns out to improve the convergence of restarted GMRES significantly. Throughout the numerical results, the convergence rates in terms of number of iterations are very similar to those of the state-of-the art variant GMRES-DR. There are even cases where A-GMRES outperforms GMRES-DR when it comes both to number of iterations and number of matrix-vector products.

Especially the M_{coa} and M_{exa} spectral preconditioners show great potential, as they can be constructed completely from by-products of GMRES, without any reference to A, while still performing very well on all test problems. The more novel spectral preconditioners M_{mul} and M_{add} allow to reduce the number of iterations as they cluster the complete eigenspectrum, and can most effectively be implemented on tough problems where restarted GMRES does not converge. However, in cases where ordinary restarted GMRES already converges, the amount of work in terms of matrix-vector products for M_{mul} and M_{add} may not pay out.

There is still a lot to investigate about the method proposed. For instance, the effect of filtering of the Ritz vectors based on the backward error has not been analysed thoroughly; it is assumed to be a good measure for the quality for construction of spectral preconditioners. Also, since GMRES-DR, which has no eigenvector filtering implemented, shows irregular convergence behaviour on for instance the `GRE1107` matrix while Adaptive GMRES is quite stable, it might be of interest to see how GMRES-DR performs with the same filtering routine. However, in this thesis GMRES-DR is treated as a black-box. Furthermore, M_{exa} has not been tested on very large matrices.

Lastly, the family of methods has successfully been tested on a dense and complex matrix arising from the field of electromagnetism, but the results have unfortunately not been included in the numerical results.

9 Acknowledgements

I would really like to thank dr. B. Carpentieri for his numerous suggestions and for providing a very good introduction into this particular field of mathematics.

A Matlab code

A.1 Main

```
1 function [x, error, innerIterations, flag, resvec, PRODUCTS] =
    newGMRES( A, b, x, preconditioner, iterations, tol,
    dimCoarse, harmonic )
2 global PRODUCTS;
3
4 % initialization
```

```
5  outputInfo = 1;
6  innerIterations = 0;
7  flag = 0;
8  resvec = zeros(iterations.restart * iterations.max + 1, 1);
9  colors = 'rgbyk';
10 maxOuter = ceil(iterations.max / iterations.restart);
11 preconditioner.spectral.levels = 0;
12 PRODUCTS = 0;
13
14 AIsReal = isreal(A);
15
16 bnrm2 = norm( b );
17 if( bnrm2 == 0.0 )
18   bnrm2 = 1.0;
19 end
20
21 resvec(1) = norm(b-A*x, 2) / bnrm2;
22
23 % Lucky guess
24 if ( resvec(1) < tol.gmres )
25   return
26 end
27
28 % initialize workspace
29 n = size(A, 1);
30 m = iterations.restart;
31 V = zeros(n,m+1);
32 H = zeros(m+1,m);
33 R = zeros(m+1,m);
34 Veps          = cell(maxOuter, 1);
35 Av            = cell(maxOuter, 1);
36 cs(1:m)       = zeros(m,1);
37 sn(1:m)       = zeros(m,1);
38 e1            = zeros(m+1,1);
39 e1(1)         = 1.0;
40
41 % Begin iteration
42 for iter = 1:maxOuter
43
44   if outputInfo == 1
45     fprintf('\n\nIteration #%d\n', iter);
46   end
47
```

```
48    PRODUCTS = PRODUCTS + 1;
49    r = precondition( b-A*x, A, preconditioner, Av, Veps, 1 );
50    rnorm = norm( r );
51    V(:,1) = r / rnorm;
52    s = rnorm*e1;
53
54    % Arnoldi
55    for i = 1:m
56      PRODUCTS = PRODUCTS + 1;
57      w = precondition( A*V(:, i), A, preconditioner, Av, Veps, 1
          );
58
59      for k = 1:i
60        t = V(:, k)'*w;
61        H(k,i)= t;
62        w = w - t*V(:,k);
63      end
64
65      H(i+1,i) = norm( w );
66      V(:,i+1) = w / H(i+1,i);
67
68      R(:, i) = H(:, i);
69
70      % Givens rotation
71      innerIterations = 1 + innerIterations;
72      for k = 1:i-1
73        temp = cs(k)*R(k,i) + sn(k)*R(k+1,i);
74        R(k+1,i) = -conj(sn(k))*R(k,i) + cs(k)*R(k+1,i);
75        R(k,i) = temp;
76      end
77
78      % ith rotational matrix
79      [cs(i),sn(i)] = rotmat( R(i,i), R(i+1,i) );
80
81      % approximate residual norm
82      temp  = cs(i)*s(i);
83      s(i+1) = -conj(sn(i))*s(i);
84      s(i)  = temp;
85
86      R(i,i) = cs(i)*R(i,i) + sn(i)*R(i+1,i);
87      R(i+1,i) = 0;
88
89      y = R(1:i,1:i) \ s(1:i);
```

```
90      xResult = x + V(:, 1:i)*y;
91      error  = norm(b - A*xResult, 2) / bnrm2;
92      resvec(innerIterations+1) = error;
93
94      if outputInfo
95        fprintf('Resvec: %e\n', error);
96      end
97
98      % Update approximation
99      if ( error <= tol.gmres )
100       x = xResult;
101       return;
102     end
103   end
104
105
106   % Compute eigenvalues
107   if iter >= 1
108
109     if harmonic
110       % Compute Harmonic Ritz values
111       em = zeros(m,1);
112       em(m) = 1;
113       f = H(1:m, 1:m)' \ em;
114       Hmatrix = H(1:m, 1:m) + H(m+1,m)^2*f*em';
115     else
116       Hmatrix = H(1:m, 1:m);
117     end
118
119     % Harmonic Ritz values
120     [EigVecH, Ds] = get_eigenpairs(Hmatrix, dimCoarse, tol.
          forwardEig, 0);
121
122     % Remove bad eigenvectors
123     l = 1;
124
125     vectors = V(:,1:m)*EigVecH;
126     while l <= size(EigVecH, 2)
127
128       % Check radius
129       if abs(Ds(l)) > tol.radiusEig
130         % Check if the eigenvalue lies outside the circle
131         if outputInfo
```

```
132             fprintf('Outside radius: %s\n', num2str(Ds(l)));
133           end
134           EigVecH(:, l) = [];
135           Ds(l) = [];
136           l = l - 1;
137         else
138           % Calculate estimated backward error
139           if harmonic
140             backErr = abs(H(m+1,m)*EigVecH(m, l))*sqrt( 1 + H(m
                  +1, m)^2 * norm( -f + EigVecH(m, l)' * f * EigVecH
                  (m, l), 2 )^2 )/ norm(H(1:m, 1:m), 2);
141           else
142             backErr = abs(H(m+1,m)*EigVecH(m, l))/norm(H(1:m, 1:m
                  ), 2);
143           end
144
145           % Remove inaccurate eigvectors
146           if backErr > tol.backwardEig
147             EigVecH(:, l) = [];
148             Ds(l) = [];
149             l = l - 1;
150             if outputInfo
151               fprintf('Backward error too large: %e\n', backErr);
152             end
153           else
154             % Check residual if output is needed
155             if outputInfo == 1
156               residual = precondition(A*vectors(:, l), A,
                      preconditioner, Av, Veps, 0 ) - Ds(l)*vectors(:,
                       l);
157               fprintf( 'Eigenvalue: %s, MAx-lx = %e', num2str(Ds(
                      l)), norm( residual ) );
158               fprintf(' (used!)\n');
159             end
160           end
161         end
162
163         l = l + 1;
164       end
165
166       % If any eigenvectors are left
167       if size(EigVecH, 2) > 0
```

```
168         preconditioner.spectral.levels = 1 + preconditioner.
                spectral.levels;
169
170         % Split vectors
171         if(AIsReal && ~isreal(EigVecH))
172           EigVecH = filterEigenpairs(EigVecH);
173         end
174
175         % Compute approximate eigenvector of MA
176         Veps{preconditioner.spectral.levels} = V(:,1:m)*EigVecH;
177
178         % Compute coarse grid operator
179         PRODUCTS = PRODUCTS + size(EigVecH, 2);
180         Av{preconditioner.spectral.levels} = Veps{preconditioner.
                spectral.levels}' * A * Veps{preconditioner.spectral.
                levels};
181
182         if outputInfo == 1
183           fprintf('Used %d vectors\n', size(EigVecH, 2));
184           fprintf('Condition Av = %1.3e\n', cond(Av{
                  preconditioner.spectral.levels}));
185         end
186       end
187     end
188
189     x = xResult;
190   end
191
192   % Not converged?
193   if ( error > tol.gmres )
194     flag = 1;
195   end
```

A.2 Preconditioning implementation

```
1   function [z] = precondition( vec, A, data, Av, Veps, count )
2     global PRODUCTS;
3
4     if data.spectral.levels == 0 || data.spectral.type == 0
5       % STANDARD PRECONDITIONER
6       z = feval(data.func, data.initial, vec);
7       return;
8
9     elseif data.spectral.type == 1
```

```
10      % add+coa SPECTRAL PRECONDITIONER
11
12      % High frequency correction
13      z = zeros(size(vec,1), 1 );
14      for i=1:(data.spectral.mu1+data.spectral.mu2)
15
16        if count
17          PRODUCTS = PRODUCTS + 1;
18        end
19
20        z = z + data.spectral.relax * feval(data.func, data.
              initial, vec - A*z);
21      end
22
23      % High frequency correction
24      z = z - Veps{1}*(Veps{1}'*z);
25
26      % Low frequency
27      for i = 1:data.spectral.levels
28        z = z + Veps{i} *( Av{i} \ ( Veps{i}'*vec ) );
29      end
30
31      return;
32
33    elseif data.spectral.type == 2
34      % mul+res SPECTRAL PRECONDITIONER
35
36      z = zeros(size(vec,1), 1);
37      % Pre-smoothing
38      for i=1:data.spectral.mu1
39        if count
40          PRODUCTS = PRODUCTS + 1;
41        end
42
43        z = z + data.spectral.relax * feval(data.func, data.
              initial, vec - A*z);
44      end
45
46      % Coarse grid correction
47      if count
48        PRODUCTS = PRODUCTS + 1;
49      end
50
```

```
51      z = z + Veps{1} * (Av{1} \ (Veps{1}'*(vec - A*z)));
52
53      % Post-smoothing
54      for i=1:data.spectral.mu2
55        if count
56          PRODUCTS = PRODUCTS + 1;
57        end
58
59        z = z + data.spectral.relax * feval(data.func, data.
              initial, vec - A*z);
60      end
61
62      for i = 2:data.spectral.levels
63        if count
64          PRODUCTS = PRODUCTS + 1;
65        end
66
67        z = z + Veps{i} * (Av{i} \ (Veps{i}'*(vec - A*z)));
68      end
69
70      return;
71
72    elseif data.spectral.type == 3
73      % coa PRECONDITIONER
74      z = feval(data.func, data.initial, vec);
75
76      for i=1:data.spectral.levels
77        z = z +  Veps{i} *( Av{i} \ ( Veps{i}'*vec ) );
78      end
79
80      return;
81
82    elseif data.spectral.type == 4
83      % mul+coa SPECTRAL PRECONDITIONER
84
85      z = zeros(size(vec,1), 1);
86      % Pre-smoothing
87      for i=1:data.spectral.mu1
88        if count
89          PRODUCTS = PRODUCTS + 1;
90        end
91
```

```
92        z = z + data.spectral.relax * feval(data.func, data.
              initial, vec - A*z);
93      end
94
95      % Coarse grid correction
96      if count
97        PRODUCTS = PRODUCTS + 1;
98      end
99
100     z = z + Veps{1} * (Av{1} \ (Veps{1}'*(vec - A*z)));
101
102     % Post-smoothing
103     for i=1:data.spectral.mu2
104       if count
105         PRODUCTS = PRODUCTS + 1;
106       end
107
108       z = z + data.spectral.relax * feval(data.func, data.
              initial, vec - A*z);
109     end
110
111     for i = 2:data.spectral.levels
112       z = z + Veps{i} * (Av{i} \ (Veps{i}'*vec));
113     end
114
115     return;
116   end
117 end
```

A.3 Eigenpair implementation

```
1  function [ vectors_out ] = filter_eigenpairs( vectors )
2    vectors_out = [];
3    left = size(vectors, 2);
4
5    while left > 0
6      current = vectors(:, 1);
7      if isreal(current)
8        vectors_out = [vectors_out, current];
9        vectors(:, 1) = [];
10     else
11       real_part = real(current);
12       imag_part = imag(current);
13       vectors_out = [vectors_out, real_part, imag_part];
```

```
14
15          if left == 1
16            vectors(:, 1) = [];
17          elseif left > 1,
18            vectors(:, 1:2) = [];
19          end
20        end
21        left = size(vectors,2);
22      end
23    end
```

```
1   function [ vectors, values ] = get_eigenpairs( matrix, number,
        varargin )
2
3     options.tol = varargin{1};
4     options.disp = varargin{2};
5     [vectors, values] = eigs(matrix, number, 'sm', options );
6     values = diag(values);
7   end
```

A.4 Givens rotation

```
1   function [ c, s ] = rotmat( a, b )
2   % Compute the Givens rotation matrix parameters for a and b.
3
4   if ( b == 0.0 ),
5     c = 1.0;
6     s = 0.0;
7   elseif ( abs(b) > abs(a) ),
8     temp = a / b;
9     s = 1.0 / sqrt( 1.0 + temp^2 );
10    c = temp * s;
11  else
12    temp = b / a;
13    c = 1.0 / sqrt( 1.0 + temp^2 );
14    s = temp * c;
15  end
```

References

[1] G. Alléon, M. Benzi, and L. Giraud. Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics. *Numerical Algorithms*, 16(1):1–15, 1997.

[2] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. 9:17–29, 1951.

[3] M. Benzi, J. Marin, and M. Tůma. A two-level parallel preconditioner based on sparse approximate inverses. In D.R. Kincaid and eds A.C. Elster, editors, *Iterative Methods in Scientific Computation IV, IMACS Series in Computational and Applied Mathematics*, pages 167–178. IMACS, New Brunswick, NJ, 1999.

[4] M. Benzi and C.D. Meyer. A direct projection method for sparse linear systems. *SIAM Journal on Scientific Computing*, 16(5):1159–1176, 1995.

[5] M. Benzi, C.D. Meyer, and M. Tuma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, 1996.

[6] Å. Björck and C.C. Paige. Loss and recapture of orthogonality in the modified gram-schmidt algorithm. *SIAM Journal on Matrix Analysis and Applications*, 13(1):176–190, 1992.

[7] Matthias Bollhöfer and Yousef Saad. Multilevel preconditioners constructed from inverse-based ILUs. *SIAM Journal on Scientific Computing*, 27(5):1627–1650, 2006.

[8] B. Carpentieri, I.S. Duff, and L. Giraud. A class of spectral two-level preconditioners. *SIAM Journal on Scientific Computing*, 25(2):749–765, 2003.

[9] B. Carpentieri, L. Giraud, and S. Gratton. Additive and multiplicative two-level spectral preconditioning for general linear systems. *SIAM Journal on Scientific Computing*, 29(4):1593–1612, 2007.

[10] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM Journal on Scientific Computing*, 19(3):995–1023, 1998.

[11] J.D.F. Cosgrove, J.C. Diaz, and A. Griewank. Approximate inverse preconditionings for sparse linear systems. *International Journal of Computer Mathematics*, 44(1-4):91–110, 1992.

[12] S. Demko. Inverses of band matrices and local convergence of spline projections. *SIAM Journal on Numerical Analysis*, 14(4):616–619, 1977.

[13] S. Demko, W.F. Moss, and P.W. Smith. Decay rates for inverses of band matrices. *Mathematics of computation*, 43(168):491–499, 1984.

[14] I.S. Duff, A.M. Erisman, C.W. Gear, and J.K. Reid. Sparsity structure and gaussian elimination. *ACM SIGNUM Newsletter*, 23(2):2–8, 1988.

[15] V. Eijkhout and B. Polman. Decay rates of inverses of banded m-matrices that are near to Toeplitz matrices. *Linear Algebra and its Applications*, 109:247–277, 1988.

[16] M. Embree. How descriptive are GMRES convergence bounds? 1999.

[17] R.W. Freund and N.M. Nachtigal. QMR: a quasi-minimal residual method for non-hermitian linear systems. *Numerische Mathematik*, 60(1):315–339, 1991.

[18] L. Giraud, S. Gratton, and E. Martin. Incremental spectral preconditioners for sequences of linear systems. *Applied Numerical Mathematics*, 57(11):1164–1180, 2007.

[19] A. Greenbaum, V. Pták, and Z. Strakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM Journal on Matrix Analysis and Applications*, 17(3):465–469, 1996.

[20] L. Greengard. *The rapid evaluation of potential fields in particle systems*. MIT press, 1988.

[21] M.J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[22] M.J. Grote and H.D. Simon. Parallel preconditioning and approximate inverses on the connection machine. In *PPSC*, pages 519–523, 1993.

[23] Y.A. Kuznetsov. New algorithms for approximate realization of implicit difference schemes. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 3(2):99–114, 1988.

[24] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards*, 49(1):33–53, 1952.

[25] J.A. Meijerink and H.A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of computation*, 31(137):148–162, 1977.

[26] R.B. Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra and its Applications*, 154:289–309, 1991.

[27] R.B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM Journal on Matrix Analysis and Applications*, 16(4):1154–1171, 1995.

[28] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[29] Y. Saad. *Iterative methods for sparse linear systems*. Siam, 2003.

[30] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[31] G.L.G. Sleijpen and H.A. Van der Vorst. A jacobi–davidson iteration method for linear eigenvalue problems. *SIAM Review*, 42(2):267–293, 2000.

[32] K. Stüben. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128(1):281–309, 2001.

[33] H.A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644, 1992.

[34] H.A. Van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *Journal of computational and applied mathematics*, 48(3):327–341, 1993.