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 3

2 Iterative Methods 4

3 Krylov subspace methods 5
   3.1 Arnoldi algorithm 6
   3.2 Classification of Krylov subspace methods 7

4 GMRES 9
   4.1 Convergence of GMRES 12
   4.2 Improvements and constraints
      4.2.1 GMRES(m) 15
      4.2.2 Deflated restart 16

5 Preconditioning 17
   5.1 Stationary methods as preconditioners 17
   5.2 Well-established preconditioning techniques
      5.2.1 Incomplete factorization methods 18
      5.2.2 Sparse approximate inverses 19
      5.3 Two-level spectral preconditioning
         5.3.1 Several spectral preconditioners 22

6 Adaptive GMRES 28
   6.1 Computation and quality of approximate eigenpairs 29
   6.2 Computational costs 33
   6.3 Implementation details 34

7 Results 35
   7.1 Reducing the number of matrix-vector products 39

8 Conclusion and discussion 44

9 Acknowledgements 44

A Matlab code 44
   A.1 Main 44
   A.2 Preconditioning implementation 49
   A.3 Eigenpair implementation 52
   A.4 Givens rotation 53
1 Introduction

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

\[ \mathbf{Ax} = \mathbf{b} \]  

(1.1)

where \( \mathbf{A} \in \mathbb{C}^{n \times n} \) is the coefficient matrix which is a large general matrix, \( \mathbf{x} \in \mathbb{C}^n \) is the unknown solution vector and \( \mathbf{b} \in \mathbb{C}^n \) 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 \( \mathbf{A} \), \( \mathbf{b} \) 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 \( \mathbf{A} \) 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 $P = \text{diag}(A)$ 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

$$ Ae_k = A(x - x_k) = b - Ax_k = r_k $$

(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

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

(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

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

(2.3)

referred to as the non-stationary Richardson iteration. The optimal choice for the parameters $\alpha_k$ 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:

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

(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 $A$:

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

Thus the non-stationary Richardson method can reach any element in the affine space $x_0 + K_k$, where $K_k = \text{span}\{r_0, A r_0, \ldots, A^{k-1} r_0\}$ 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 $x_0 + K_n$.

3 Krylov subspace methods

In the previous subsection it was already seen that classical iterative methods iterate within an affine space $x_0 + \text{span}\{r_0, A r_0, \ldots, A^{k-1} r_0\}$, 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 $K_k(A, b)$ 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

$$K_k(A, b) \equiv \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}. \quad (3.1)$$

As a consequence, every element $u \in K_k(A, b)$ of a Krylov subspace can be expressed as a polynomial $p_{k-1} \in \mathbb{F}_{k-1}$ 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

$$q(A) = 0. \quad (3.2)$$
Suppose $A$ has distinct eigenvalues $\{\lambda_1, \ldots, \lambda_d\}$ with $d \leq n$ where each eigenvalue $\lambda_j$ has index $m_j$. Denote by $m = \sum_{j=1}^{d} m_j$ the sum of the indices, then

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

(3.3)

is the minimal polynomial. If we expand the polynomial to

$$q(A) = \alpha_0 I + \alpha_1 A + \cdots + \alpha_m A^m$$

(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 $m - 1$.

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

(3.5)

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

*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 an 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 $m$th 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 $H_m \in \mathbb{C}^{(m+1) \times m}$ 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:

$$V_m^H A V_m = H_m,$$  

(3.6)

$$AV_m = V_{m+1} H_m.$$  

(3.7)

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

(3.8)
Algorithm 1 The Arnoldi algorithm

1: \( v_1 := r_0 / \|r_0\| \)
2: \( \text{for } j = 1, \ldots, m \text{ do} \)
3: \( \text{for } i = 1, \ldots, j \text{ do} \)
4: \( h_{ij} = \langle Av_j, v_i \rangle \)
5: \( \text{end for} \)
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: \( \text{if } h_{j+1,j} = 0 \text{ then} \)
9: \( \text{return} \)
10: \( \text{else} \)
11: \( v_{j+1} = \hat{v}_{j+1} / h_{j+1,j} \)
12: \( \text{end if} \)
13: \( \text{end for} \)

Proof. Let \( \epsilon_i \) denote the \( i \)th standard basis vector. Since \( h_{ij} = 0 \) for \( i + 1 > j \), it holds that

\[
AV_m = [Av_1, \ldots, Av_{m-1}, Av_m]
\]

\[
= \begin{bmatrix}
\hat{v}_2 + v_1 h_{11}, & \ldots, & \hat{v}_m + \sum_{i=1}^{m-1} h_{ij} v_i, & \hat{v}_{m+1} + \sum_{i=1}^{m} h_{ij} v_i \\
\sum_{i=1}^{m} h_{i1} v_i, & \ldots, & \sum_{i=1}^{m} h_{i,m-1} v_i, & \hat{v}_{m+1} + \sum_{i=1}^{m} h_{im} v_i
\end{bmatrix}
\]

(3.9)

The last equality of equation (3.9) follows from line 11 of the Arnoldi algorithm. 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

\[
V_m^H AV_m = H_m.
\]

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

\[
AV_m = V_{m+1} H_m.
\]

The third identity occurs already in equation (3.9). \( \square \)

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 \( x_0 + K_k(A, r_0) \). 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 $\alpha_{k+1}$ at each $k$th iteration. Also, the best choice of parameters $\alpha_1, \ldots, \alpha_k$ for the $k$th 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 $\mathcal{V} = \mathcal{W} = K_k(A,r_0)$ and extracts $x_k$ from $\mathcal{V}$ such that the residual $r_k$ is perpendicular to $\mathcal{V}$:

$$b - Ax_k \perp \mathcal{V}. \quad (3.10)$$

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

$$V_k^H A V_k y = V_k^H b, \quad (3.11)$$

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 $\mathcal{V} = K_k(A,r_0) \neq \mathcal{W}$. The residual $r_k$ is again required to be perpendicular to the test space $\mathcal{W}$:

$$b - Ax_k \perp \mathcal{W}. \quad (3.12)$$

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

$$W_k^H (b - A V_k y) = 0 \Leftrightarrow W_k^H A V_k y = W_k^H b. \quad (3.13)$$
Robust methods within this class like BiCGSTAB and QMR apply to general matrices and choose $W$ 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 $x_k \in \mathcal{V}$:

$$ x_k = \arg \min_{x^* \in \mathcal{V}} \| b - A x^* \|. \quad (3.14) $$

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 $x_0 + K_k(A, r_0)$, 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 $k$th iteration of GMRES the approximate solution $x_k$ is

$$ x_k \equiv \arg \min_{z \in x_0 + K_k(A, r_0)} \| Az - b \|_2. \quad (4.1) $$

It is trivial that the Krylov subspace generated at each iteration contains the previous Krylov subspaces, i.e. $K_k(A, r_0) \subset K_{k+1}(A, r_0)$ 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 $x_k \in x_0 + K_k(A, r_0)$, write $x_k = x_0 + y$ for a $y \in K_k(A, r_0)$. 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:

$$ r_k = b - A x_k = b - A(x_0 + y) $$
$$ = r_0 - AV_k z $$
$$ = \|r_0\|_2 v_1 - V_{k+1} \hat{H}_k z $$
$$ = V_{k+1}(\|r_0\|_2 e_1 - \hat{H}_k z). $$
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
\[
\|r_k\|_2 = \|b - Ax_k\|_2 = \|\gamma e_1 - \hat{H}_k z\|_2.
\]
(4.2)

Minimization of equation (4.2) with respect to \( z \) is equivalent to finding the least-squares solution of
\[
\hat{H}_k z = \gamma e_1.
\]
(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
\[
G_i^m(c, s) = \begin{pmatrix}
1 & & & & & \\
& \ddots & & & & \\
& & 1 & & & \\
& & c & s & & \\
& & -\bar{s} & c & & \\
& & & & \ddots & \\
& & & & & 1
\end{pmatrix}
\]
(4.4)

where \( \mathbb{S}^2 = \{(c, s) \in \mathbb{C}^2 : |c|^2 + |s|^2 = 1\} \) and \( 1 \leq i \leq m - 1 \). 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
\[
\text{sign } z = \begin{cases}
\frac{z}{|z|} & \text{if } z \neq 0 \\
1 & \text{if } z = 0
\end{cases}
\]
(4.5)

Proposition 4.1. Let \( x \in \mathbb{C}^m \) and \( 1 \leq i \leq m - 1 \). If \( x_{i+1} \neq 0 \), then \( G_i^m(c, s) \) with
\[
c = \frac{x_i}{\sqrt{|x_i|^2 + |x_{i+1}|^2}}, \\
s = \text{sign}(x_i) \frac{x_{i+1}}{\sqrt{|x_i|^2 + |x_{i+1}|^2}}
\]
(4.6)
is a Givens matrix. If \( y = G_i x \), then
1. \( y_{i+1} = 0 \),
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 $s$:

$$y = G_i x = (x_1 \ldots x_{i-1} cx_i + sx_{i+1} - \bar{s}x_i + cx_{i+1} x_{i+2} \ldots x_m)^T. \quad (4.7)$$

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:

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

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

By the Hessenberg structure of $\hat{H}_k$ and Proposition 4.1, it follows that $\hat{R}_k^{(i)}$ 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:

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

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

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

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

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

$$x_k = x_0 + V_kz.$$  \hspace{1cm} (4.11)

**Theorem 4.1.** The normed GMRES residual at the $k$th 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

$$b - Ax_k = V_{k+1}(\gamma e_1 - \hat{H}_kz)$$
$$= V_{k+1}Q_k^TQ_k(\gamma e_1 - \hat{H}_kz)$$
$$= V_{k+1}Q_k^T(\hat{s} - \hat{R}_k^{(k)}z).$$  \hspace{1cm} (4.12)

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

$$\|b - Ax_k\|_2 = |\hat{s}_{k+1}|,$$  \hspace{1cm} (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 $k$th step of GMRES for an approximate solution $x_k \in x_0 + K_k(A, r_0)$ can be expressed as

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

for a $y \in K_k(A, r_0)$. Thus it follows that

$$r_k \in r_0 + AK_k(A, r_0) = r_0 + \text{span}\{Ar_0, A^2r_0, \ldots, A^kr_0\}.$$
Algorithm 2 The GMRES($m$) algorithm with left preconditioning. The expression $h_{\cdot,i}$ denotes the $i$th column of $H$.

1: function GMRES($A, b, x_0, m, M$) 
2: \hspace{1cm} $x_0^{(1)} = x_0$ \hspace{1cm} $\triangleright$ Initial residual 
3: \hspace{1cm} for $j = 1, 2, \ldots$ do 
4: \hspace{1.5cm} $r := M(b - Ax^{(j)})$ 
5: \hspace{1.5cm} $v_1 := r / \|r\|_2$ 
6: \hspace{1.5cm} $s := \|r\|_2 e_1$ 
7: \hspace{1.5cm} for $i = 1, 2, \ldots, m$ do 
8: \hspace{3cm} $w := MAv_i$ \hspace{1cm} $\triangleright$ Start Arnoldi process 
9: \hspace{3cm} for $k = 1, \ldots, i$ do 
10: \hspace{4cm} $h_{k,i} = (w, v_k)$ 
11: \hspace{4cm} $w \leftarrow w - h_{k,i}v_k$ 
12: \hspace{3cm} end for 
13: \hspace{3cm} $h_{i+1,i} = \|w\|_2$ 
14: \hspace{3cm} $v_{i+1} = w / h_{i+1,i}$ 
15: \hspace{3cm} for $k = 1, \ldots, i - 1$ do \hspace{1cm} $\triangleright$ Apply previous Givens rotations 
16: \hspace{4cm} $h_{k,i} \leftarrow G_k h_{k,i}$ 
17: \hspace{4cm} end for 
18: \hspace{3cm} Construct $G_i$ from $h_{i,i}$ and $h_{i+1,i}$ \hspace{1cm} $\triangleright$ Construct new Givens rotation 
19: \hspace{3cm} $h_{i,i} \leftarrow G_i h_{i,i}$ \hspace{1cm} $\triangleright$ Apply new Givens rotation 
20: \hspace{3cm} $s \leftarrow G_i s$ 
21: \hspace{3cm} if $s_{i+1} < \text{tol}$ then 
22: \hspace{4cm} Solve $H_i y = s$ for $y$ with backward substitutions 
23: \hspace{4cm} return $x = x_0 + V_i y$ 
24: \hspace{3cm} end if 
25: \hspace{3cm} end for 
26: \hspace{1cm} Solve $H_m y = s$ for $y$ with backward substitutions 
27: \hspace{1cm} $x_0 \leftarrow x_0 + V_m y$ 
28: \hspace{1cm} end for 
29: 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 $A \equiv X \Lambda X^{-1}$ where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ 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

$$\|r_k\|_2 = \min_{p_k \in \mathbb{P}_k} \|p_k(A)r_0\|_2 \leq \kappa_2(V) \min_{p_k \in \mathbb{P}_k} \|p_k(A)\|_2 \|r_0\|_2;$$

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa_2(V) \min_{p_k \in \mathbb{P}_k} \max_{p_k(0)=1, i=1,\ldots,n} |p_k(\lambda_i)|,$$

where $\kappa_2(V) = \|V\|_2 \|V^{-1}\|_2$ 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 $\lambda_i$, 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 uninformative. 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 $H_n$ are sufficient approximations of the smallest eigenvalues of $A$. In other words, the Hessenberg matrix should capture the smallest eigenvalues of $A$. 

14
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 $K \subseteq \mathbb{C}^n$, a pair $(\theta, y) \in \mathbb{C} \times \mathbb{C}^n$ is called a Ritz pair with respect to $K$ if

$$Ay - \theta y \perp K. \quad (4.15)$$

In detail, $\theta$ is called a Ritz value and $y$ is called a Ritz vector.

For Krylov subspace methods we can choose the Krylov subspace $K_m(A, r_0)$ itself.

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

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

$$H_m x = \theta x$$
$$V_m^H A V_m x = \theta V_m^H V_m x$$
$$V_m^H (Ay - \theta y) = 0 \quad (4.16)$$

Since $V_m$ in a basis for $K_m(A, r_0)$, it follows that

$$Ay - \theta y \perp K_m(A, r_0), \quad (4.17)$$

which means that $(\theta, y)$ is a Ritz-pair of $A$. \qed

### 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 subspace 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 drawback 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(\( m, \ell \)),
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 \( \ell \) eigenvectors \( \{ u_1, \ldots, u_\ell \} \) corresponding to
eigenvalues of \( A \) close to the origin and augments the Krylov subspace as

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

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 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 $x = MMy$.

**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:

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

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

17
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 $S \subset \{(i,j) : i,j = 1,\ldots,n\}$ of positions of matrix entries and limit updates in the Gaussian elimination process to entries in $S$. That is, for each row $k$ and $i,j > k$:

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

If $S$ 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 $\text{level}(i, j) = \text{level}(i, k) + \text{level}(k, j) + 1$ \cite{29}. If the level of fill is above a given integer $\ell$, the element $u_{ij}$ is dropped. This method is referred to as $\text{ILU}(\ell)$, 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 $i$th row of $A$. This method is called $\text{ILUT}(\tau)$, where $\tau$ defines the threshold.

Both the conditions of $\text{ILU}(\ell)$ and $\text{ILUT}(\tau)$ 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 $\text{ILU}(\ell)$ and $\text{ILUT}(\tau)$.

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 $\tilde{L}^{-1}Au^{-1} = \tilde{L}^{-1}E\tilde{U}^{-1}$.

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

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

(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 $\tilde{S} = E - XD\tilde{Y}$ — 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 \cite{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 \cite{21} and AINV \cite{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 \cite{14}.
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 \( \|A^{-1}\| \leq r \in \mathbb{R} \), 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 \( i, j = 1, \ldots, n \); 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 coefficient 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 \( S \)
with a fixed sparsity pattern, which minimizes the Frobenius norm
\[
\min_{M \in S} \|I - AM\|_F \iff \min_{M \in S} \|e_i - Am_i\|_2 \quad (5.4)
\]
for all \( i = 1, \ldots, n \) where \( m_i \) denotes the \( i \)th 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
\[
\|e_j - Am_j\|_1 < \tau \quad (5.5)
\]
for all \( j \), then each eigenvalue \( \lambda \) of \( AM \) lies within the disc of radius \( \tau \) around 1:
\[
|\lambda - 1| < \tau.
\]

One difficulty of SPAI is that the optimal sparsity structure \( S \) 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].
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 \(^4\). 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 \( \sigma \). In most cases, \( \sigma \) 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 \( z = Mx \) 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 \( U \) denote a \( k \)-dimensional right-invariant eigenspace of \( MA \) associated with the \( k \) smallest eigenvalues of \( MA \), for which \( U \in \mathbb{C}^{n \times k} \) forms a basis. That is,

\[ MAU = UJ \]

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 \( J = \text{diag}(\lambda_1, \ldots, \lambda_k) \) and \( U \) as the set of associated eigenvectors \( \{x_1, \ldots, x_k\} \). Otherwise a Jordan decomposition can be used. However, multiple bases for \( U \) 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

\[ A_c \equiv W^HAU, \]
\[ M_c \equiv UA_c^{-1}W^H, \]

(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:

\[
M_{coa} = M + M_c,
\]
\[
M_{res} = M + M_c - M_c AM,
\]
\[
M_{exa} = M + M_c - U J A_c^{-1} W^H. \tag{5.8}
\]

**Proof.** The proof follows by simple substitutions \hfill \square

**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 $\nu_i$ for all $i$. Let $U^\perp \in \mathbb{C}^{n-k}$ be an orthogonal complement of $U$ and denote by $[U \mid U^\perp]$ 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:

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

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

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

Consequently by combining equations (5.9) and (5.10)

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

which is the similarity relation which concludes the proof. \hfill \square
Algorithm 3: $z = M_{\text{coa}} x$

1: function $M_{\text{coa}}(x, M, A, U, W)$
2: \hspace{1em} $c_1 = Mx$
3: \hspace{1em} $c_2 = U(W^H AU)^{-1}W^H x$
4: \hspace{1em} return $z = c_1 + c_2$
5: end function

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

1: function $M_{\text{res}}(x, M, A, U, W)$
2: \hspace{1em} $c_1 = Mx$
3: \hspace{1em} $r = x - Ac_1$
4: \hspace{1em} $c_2 = U(W^H AU)^{-1}W^H r$
5: \hspace{1em} return $z = c_1 + c_2$
6: end function

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

1: function $M_{\text{exa}}(x, M, A, U, W, J)$
2: \hspace{1em} $c_1 = Mx$
3: \hspace{1em} $c_2 = U(I - J)(W^H AU)^{-1}W^H x$
4: \hspace{1em} return $z = c_1 + c_2$
5: end function

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

1: function $M_{\text{mul}}(x, M, A, U, W, \mu_1, \mu_2)$
2: \hspace{1em} $z = 0$
3: \hspace{1em} for $i = 1, \ldots, \mu_1$ do
4: \hspace{2em} $z \leftarrow z + \omega M(x - Az)$
5: \hspace{1em} end for
6: \hspace{1em} $r = x - Az$
7: \hspace{1em} $z \leftarrow z + U(W^H AU)^{-1}W^H r$
8: \hspace{1em} for $i = 1, \ldots, \mu_2$ do
9: \hspace{2em} $z \leftarrow z + \omega M(x - Az)$
10: \hspace{1em} end for
11: \hspace{1em} return $z$
12: end function

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

1: function $M_{\text{add}}(x, M, A, U, W, \mu_1, \mu_2)$
2: \hspace{1em} $c_1 = 0$
3: \hspace{1em} $c_2 = U(W^H AU)^{-1}W^H x$
4: \hspace{1em} for $i = 1, \ldots, \mu_1 + \mu_2$ do
5: \hspace{2em} $c_1 \leftarrow c_1 + \omega M(x - Ac_1)$
6: \hspace{1em} end for
7: \hspace{1em} $c_1 \leftarrow (I - UW^H)c_1$
8: \hspace{1em} return $z = c_1 + c_2$
9: 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 $M_{cAMA}$:

$$
M_{cAMA}[U|U^\perp] = M_{cA}[U|U^\perp] \begin{pmatrix} J & E \\ 0 & F \end{pmatrix}
$$

Combining equations (5.11) and (5.12) yields:

$$(M + M_{c} - M_{cAM})[U|U^\perp] = [U|U^\perp] \begin{pmatrix} I & A_{c}^{-1}WHAU^\perp(I - F) \\ 0 & F \end{pmatrix},$$

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

$$
UJA_{c}^{-1}WHA[UU^\perp] = [UU^\perp] \begin{pmatrix} J & 0 \\ 0 & F \end{pmatrix},$$

we find the similarity relation

$$(M + M_{c} - UJA_{c}^{-1}WHAU^\perp)[U|U^\perp] = [U|U^\perp] \begin{pmatrix} I & E + (I - J)A_{c}^{-1}WHAU^\perp \\ 0 & F \end{pmatrix},$$

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 Rühe 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 \(x_{k+1} = x_k + \omega Mr_k\) 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. \(\square\)

**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

\[
M_{add} = (I - U^H)(I - (I - \omega MA)\mu_1 + \mu_2)A^{-1} + M_c
\] (5.15)

**Proof.** The proof follows by use of Lemma 5.1 and simple substitutions. \(\square\)

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

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

26
By induction on \( j \) it follows that \((I - \omega MA)^j U = U(I - \omega J)^j\) for \( j = 1, 2, \ldots \), so that
\[
M_{\text{add}} A U = (I - U W^H) (I - (I - \omega MA)^{\mu_1 + \mu_2}) U + U \\
= (I - U W^H) U - (I - U W^H) U (I - \omega J)^{\mu_1 + \mu_2} + U \tag{5.17}
\]

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:
\[
M_{\text{add}} A U^\perp = (I - U W^H) (I - (I - \omega MA)^{\mu_1 + \mu_2}) U^\perp + U A^{-1} W^H A U^\perp \\
= (I - U W^H) U^\perp - (I - U W^H) (U^\perp (I - \omega F)^{\mu_1 + \mu_2} + UP_{\mu_1} + \mu_2) + U A^{-1} W^H A U^\perp \\
= U^\perp (I - (I - \omega F)^{\mu_1 + \mu_2}) + U ((I - W^H U) P_{\mu_1} + \mu_2 - W^H U^\perp (I + (I - \omega F)^{\mu_1 + \mu_2}) \\
+ A^{-1} W^H A U^\perp) \\
= U^\perp (I - (I - \omega F)^{\mu_1 + \mu_2}) + U Q \tag{5.18}
\]

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)
\[
M_{\text{add}} A [U | U^\perp] = [U | U^\perp] \begin{pmatrix} I & Q \\ 0 & I - (I - \omega F)^{\mu_1 + \mu_2} \end{pmatrix} \tag{5.19}
\]
defines the similarity relation between \( M_{\text{add}} A \) and a matrix clearly having the desired eigenvalues.

**Proposition 5.7.** The explicit form of \( M_{\text{mul}} \) as defined in Algorithm 6 is given by:
\[
M_{\text{mul}} = A^{-1} - (I - \omega MA)^{\mu_2} (I - M_c A) (I - \omega MA)^{\mu_1} A^{-1} \tag{5.20}
\]

**Proof.** The proof follows by applying Lemma 5.1.

**Proposition 5.8.** The eigenvalues of \( \{\nu_1, \ldots, \nu_n\} \) of \( M_{\text{mul}} A \) are given as
\[
\nu_i = \begin{cases} 
1 & \text{if } i \leq k \\
1 - (1 - \omega \lambda_i)^{\mu_1 + \mu_2} & \text{if } i > k
\end{cases} \tag{5.21}
\]

**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 $i$th 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 $i$th 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...
Algorithm 8 Adaptive GMRES

1: function A-GMRES(m, j; A, b, x₀ )
2: for i = 1, 2, . . . do
3: \( r = M^{(i-1)}(b - Ax_0) \)
4: Perform a single inner iteration of GMRES(m)
5: Compute \( j \) the smallest eigenpairs \((\theta_k, x_k)\) of \( H_m \).
6: for \( k = 1, \ldots, j \) do
7: Append \( x_k \) as a column to \( X^{(i)} \) if \(|\lambda_k| < \epsilon_1\) and \( \eta(A, V_m x_k) < \epsilon_2 \).
8: end for
9: Compute \( U^{(i)} \equiv V_m X^{(i)} \)
10: Define \( W^{(i)} \) such that \( (W^{(i)})^H A U^{(i)} \) is non-singular
11: Let \( M_* \) be a two-level spectral preconditioner, either \( M_{coa}, M_{res}, M_{mul} \) or \( M_{add} \)
12: Define \( M^{(i)} \equiv M_* (M^{(i-1)}, U^{(i)}, W^{(i)}) \)
13: end for
14: 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 \( \ell \) 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 \( H_m \) 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 \((\theta, y)\) of \( A \) with \( y = V_m x \) and \((x, \theta)\) an eigenpair of \( H_m \) have a residual of

\[
\|Ay - \theta y\|_2 = \|h_{m+1,m}\| |x_m|,
\]

where \( x_m \) is the \( m \)th entry of \( x \).
Proof. The proof follows from the third identity of Proposition 3.1:

\[
Ay - \theta y = AV_m x - \theta V_m x \\
= (V_m H_m + h_{m+1,m} v_{m+1} e_{m}^T) x - \theta V_m x \\
= V_m (H_m x - \theta x) + h_{m+1,m} v_{m+1} e_{m}^T x \\
= h_{m+1,m} v_{m+1} e_{m}^T x = h_{m+1,m} v_{m+1} e_{m}^T x.
\]

(6.2)

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

\[
\|Ay - \theta y\|_2 = |h_{m+1,m}| |x_m|,
\]

which concludes the proof. \( \square \)

Proposition 4.2 tells that given a Ritz pair \((\theta, x)\), 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 \(m\)th 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 \((\lambda, x)\), 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 \(\delta A\) 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

\[
R(A, x) = \frac{x^H A x}{x^H x}.
\]

(6.4)

Clearly for an exact eigenpair \((x, \lambda)\) 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 \((\theta, y)\) be a Ritz pair with respect to \(K_m(A, r_0)\), where \(y = V_m x\). Then \(R(A, y) = \theta\).

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

\[
R(A, y) = \frac{y^H A y}{y^H y} = \frac{x^H V_m^H A V_m x}{x^H V_m^H V_m x} = \frac{x^H H_m x}{x^H x} = \theta.
\]

(6.5) \( \square \)
Definition 6.2. The \textit{backward error of an approximate eigenvector} \( \tilde{x} \) of \( A \) is defined as
\[
\eta(A, \tilde{x}) = \min_{\delta A} \left\{ \tau > 0 : \left\| \delta A \right\| \left\| A \right\| \leq \tau : (A + \delta A) \tilde{x} = R(A, \tilde{x}) \tilde{x} \right\}
\]

Proposition 6.3. Definition 6.2 can be expressed explicitly as
\[
\eta(A, \tilde{x}) = \frac{\left\| A \tilde{x} - R(A, \tilde{x}) \tilde{x} \right\|}{\left\| A \right\| \left\| \tilde{x} \right\|}.
\]

Proof. In the proof the quantity \( \frac{\left\| \delta A \right\|}{\left\| A \right\|} \) is bounded from below; subsequently it is shown that the bound is attained for a special choice of \( \delta A \). From the equality \( (A + \delta A) \tilde{x} = R(A, \tilde{x}) \tilde{x} \) it follows by taking norms that
\[
\left\| \delta A \right\| \left\| \tilde{x} \right\| \geq \left\| R(A, \tilde{x}) \tilde{x} - A \tilde{x} \right\|. \tag{6.6}
\]
Now take \( \delta A = \frac{\left\| (R(A, \tilde{x}) \tilde{x} - A \tilde{x}) \tilde{x}^T \right\|}{\left\| \tilde{x} \right\|^2} \). Then per definition it follows that
\[
\frac{\left\| \delta A \right\|}{\left\| A \right\|} \geq \frac{\left\| R(A, \tilde{x}) \tilde{x} - A \tilde{x} \right\|}{\left\| A \right\| \left\| \tilde{x} \right\|}. \tag{6.6}
\]
Let \( \alpha \) denote the angle between \( \tilde{x} \) and \( z \). Since \( \tilde{x}^T z \) is a scalar value, it follows that
\[
\frac{\left\| \delta A \right\|}{\left\| A \right\|} = \frac{\left\| R(A, \tilde{x}) \tilde{x} - A \tilde{x} \right\|}{\left\| A \right\| \left\| \tilde{x} \right\|^2} \max_{z \neq 0} \frac{\left\| \tilde{x}^T z \right\|}{\left\| z \right\|}.
\]
Thus the bound of equation 6.6 is attained, so that
\[
\eta(A, \tilde{x}) = \frac{\left\| A \tilde{x} - R(A, \tilde{x}) \tilde{x} \right\|}{\left\| A \right\| \left\| \tilde{x} \right\|}
\]
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
\[
\eta(A, V_m x) = \frac{\left\| h_{m+1, m} \right\| \left\| x_m \right\|}{\left\| A \right\| \left\| V_m x \right\|}.
\]
The spectral norm \( \left\| A \right\| \) 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 $(\theta, x)$ 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 $\|H_m\|_2 = \|V_m^H A V_m\|_2 \leq \|V_m^H\|_2 \|A\|_2 \|V_m\|_2$. But $\|V_m\|_2 = 1$ since it is orthogonal and if $v_1$ denotes the first column of $V$:

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

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

Proposition 6.5. A pair $(\theta, g)$ is a harmonic Ritz pair of $A$ with respect to $W = AV$ if and only if

$$Au - \theta u \perp AV$$

for a $u \in V$.

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

$$V^H A^H (A^{-1} AV z - \theta^{-1} AV z) = 0 \iff V^H A^H (AV z - \theta V z) = 0. \quad (6.7)$$

Define $u \equiv V z$, so that equation 6.7 reads $Au - \theta u \perp AV$. 

32
Proposition 6.5 can be simplified in our case where \( V_m \) is generated by the Arnoldi algorithm. We have \( V_m^H A^H V_m z = \theta V_m^H A^H V_m z \) for \( z \in \mathbb{C}^k \). 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
\[
V_m^H A^H V_m = (A V_m)^H A V_m = (V_m H_k + h_{m+1,m} v_{m+1} e_m^T)^H (V_m H_m + h_{m+1,m} v_{m+1} e_m^T)
\]
so that if we define \( f = (H_m)^{-H} e_m \), the expression becomes
\[
(H_m + |h_{m+1,m}|^2 f e_m^T) z = \theta z.
\]

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

\( \text{Note.} \) 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 $M_{\text{coa}}$ and $M_{\text{exa}}$ 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

$$M(i) = M(i-1) + M_c$$

$$= M(i-1) + U(i) A_c^{-1} U(i) M(i-1)$$

$$= (I + U(i) A_c^{-1} U(i)) M(i-1).$$

Thus the $i$th 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 $M_{\text{coa}}$ and $M_{\text{exa}}$, which do not require matrix-vector products involving $A$. Furthermore, $M_{\text{res}}$ requires a single matrix-vector product, $M_{\text{add}}$ requires $\mu_1 + \mu_2$ matrix-vector products and lastly $M_{\text{mul}}$ requires $\mu_1 + \mu_2 + 1$ matrix-vector 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 $M_{\text{res}}$, $M_{\text{add}}$ or $M_{\text{mul}}$ 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: $v_1 \leftarrow \text{Re}(v_1)$ and $v_2 \leftarrow \text{Im}(v_1)$.

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 $M_{\text{exa}}$ 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\footnote{All these matrices are freely available at http://math.nist.gov/MatrixMarket/} collection. The matrix properties are listed in Table\footnote{All these matrices are freely available at http://math.nist.gov/MatrixMarket/}.

Table 1: Test matrix properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Size</th>
<th>$\text{nnz}(A)$</th>
<th>$\kappa_2(A)$</th>
<th>Discipline</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRE1107</td>
<td>Real antisymmetric</td>
<td>$1107 \times 1107$</td>
<td>5664</td>
<td>$9.7 \cdot 10^6$</td>
<td>Simulation in computer systems</td>
</tr>
<tr>
<td>HOR131</td>
<td>Real antisymmetric</td>
<td>$434 \times 434$</td>
<td>4710</td>
<td>$1.3 \cdot 10^5$</td>
<td>Flow in networks</td>
</tr>
<tr>
<td>ORSIR1</td>
<td>Real antisymmetric</td>
<td>$1030 \times 1030$</td>
<td>6858</td>
<td>$?$</td>
<td>Oil reservoir simulation</td>
</tr>
<tr>
<td>MEMPLUS</td>
<td>Real antisymmetric</td>
<td>$17758 \times 17758$</td>
<td>126150</td>
<td>$?$</td>
<td>Electronic circuit design</td>
</tr>
</tbody>
</table>

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\footnote{All these matrices are freely available at http://math.nist.gov/MatrixMarket/}. 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 \texttt{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\footnote{All these matrices are freely available at http://math.nist.gov/MatrixMarket/}.

Clearly, several eigenvalues are still close to the origin, blocking the rate of convergence of GMRES(30).
Table 2: Names for a family of Adaptive GMRES methods, depending on the sequence of spectral preconditioner used at each level

<table>
<thead>
<tr>
<th>Name</th>
<th>$M^{(1)}$</th>
<th>$M^{(2)}$</th>
<th>$M^{(3)}$</th>
<th>$M^{(4)}$</th>
<th>\ldots</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-GMRES $M_{\text{coa}}$</td>
<td>$M_{\text{coa}}$</td>
<td>$M_{\text{coa}}$</td>
<td>$M_{\text{coa}}$</td>
<td>$M_{\text{coa}}$</td>
<td>\ldots</td>
</tr>
<tr>
<td>A-GMRES $M_{\text{exa}}$</td>
<td>$M_{\text{exa}}$</td>
<td>$M_{\text{exa}}$</td>
<td>$M_{\text{exa}}$</td>
<td>$M_{\text{exa}}$</td>
<td>\ldots</td>
</tr>
<tr>
<td>A-GMRES $M_{\text{mul+res}}$</td>
<td>$M_{\text{mul}}(\mu_1 = 1, \mu_2 = 1, \omega = 2/3)$</td>
<td>$M_2$</td>
<td>$M_2$</td>
<td>$M_2$</td>
<td>\ldots</td>
</tr>
<tr>
<td>A-GMRES $M_{\text{add+coa}}$</td>
<td>$M_{\text{add}}(\mu_1 = 1, \mu_2 = 1, \omega = 2/3)$</td>
<td>$M_1$</td>
<td>$M_1$</td>
<td>$M_1$</td>
<td>\ldots</td>
</tr>
<tr>
<td>A-GMRES $M_{\text{mul+coa}}$</td>
<td>$M_{\text{mul}}(\mu_1 = 1, \mu_2 = 1, \omega = 2/3)$</td>
<td>$M_1$</td>
<td>$M_1$</td>
<td>$M_1$</td>
<td>\ldots</td>
</tr>
</tbody>
</table>

However, using A-GMRES(30, 2) with $M_{\text{coa}}$ 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 $M_{\text{coa}}$ 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 $M_{\text{mul+res}}$ 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_{\text{coa}} as preconditioner and \eta = 0.001. † and ✓ denote respectively if an eigenvalue is used or not to improve the preconditioner

<table>
<thead>
<tr>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eigenvalue</td>
<td>6.16 \cdot 10^{-2}</td>
<td>6.88 \cdot 10^{-2}</td>
<td>7.76 \cdot 10^{-3}</td>
<td>1.84 \cdot 10^{-2}</td>
<td>2.23 \cdot 10^{-2}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MAx - \lambda x</td>
<td></td>
<td>_2</td>
<td>4.49 \cdot 10^{-2}</td>
</tr>
<tr>
<td>Backw. err.</td>
<td>9.91 \cdot 10^{-3}</td>
<td>9.91 \cdot 10^{-3}</td>
<td>4.22 \cdot 10^{-4}</td>
<td>9.61 \cdot 10^{-4}</td>
<td>3.22 \cdot 10^{-4}</td>
</tr>
<tr>
<td>Used</td>
<td>†</td>
<td>†</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

(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, \( M_{\text{coa}} \) 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 \( M_{\text{coa}} \). 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 \( M_{\text{coa}} \) 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 \( M_{\text{mul+coa}} \) shows very similar convergence behaviour as \( M_{\text{mul+res}} \). The exact shift does not pay out, making \( M_{\text{mul+coa}} \) a better option. In fact, since \( M_{\text{add+coa}} \) shows very similar convergence behaviour to \( M_{\text{mul+coa}} \), 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 \( i \)th residual at the \( i \)th outer iteration was computed as \( r_{0}^{(i)} = M^{(i)}(b - A x_{0}^{(i)}) \).

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

\[
M^{(i-1)}(b - A x^{(i-1)}) = r_{0}^{(i-1)} - V_{m+1} H_{m} z, \tag{7.1}
\]

where \( z \) is the solution to the least squares problem. The quantity of equation (7.1) has to be preconditioned only with the \( i \)th 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 \( i \)th 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 \( M_{\text{coa}} \) or \( M_{\text{exa}} \) 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 \( M_{\text{coa}} \) and \( M_{\text{exa}} \) 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dimension eigenspace</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>DR(25)</td>
<td>†</td>
</tr>
<tr>
<td>A(25) ( M_{\text{add+coa}} )</td>
<td>†</td>
</tr>
<tr>
<td>A(25) ( M_{\text{mul+coa}} )</td>
<td>277/697</td>
</tr>
<tr>
<td>A(25) ( M_{\text{mul+res}} )</td>
<td>269/875</td>
</tr>
<tr>
<td>A(25) ( M_{\text{coa}} )</td>
<td>†</td>
</tr>
<tr>
<td>DR(30)</td>
<td>†</td>
</tr>
<tr>
<td>A(30) ( M_{\text{add+coa}} )</td>
<td>120/315</td>
</tr>
<tr>
<td>A(30) ( M_{\text{mul+coa}} )</td>
<td>118/400</td>
</tr>
<tr>
<td>A(30) ( M_{\text{mul+res}} )</td>
<td>117/483</td>
</tr>
<tr>
<td>A(30) ( M_{\text{coa}} )</td>
<td>330/351</td>
</tr>
<tr>
<td>DR(35)</td>
<td>†</td>
</tr>
<tr>
<td>A(35) ( M_{\text{add+coa}} )</td>
<td>70/146</td>
</tr>
<tr>
<td>A(35) ( M_{\text{mul+coa}} )</td>
<td>70/182</td>
</tr>
<tr>
<td>A(35) ( M_{\text{mul+res}} )</td>
<td>70/182</td>
</tr>
<tr>
<td>A(35) ( M_{\text{coa}} )</td>
<td>166/178</td>
</tr>
<tr>
<td>DR(40)</td>
<td>†</td>
</tr>
<tr>
<td>A(40) ( M_{\text{add+coa}} )</td>
<td>225/226</td>
</tr>
<tr>
<td>A(40) ( M_{\text{mul+coa}} )</td>
<td>77/156</td>
</tr>
<tr>
<td>A(40) ( M_{\text{mul+res}} )</td>
<td>77/194</td>
</tr>
<tr>
<td>A(40) ( M_{\text{coa}} )</td>
<td>155/162</td>
</tr>
</tbody>
</table>
Table 5: Number of iterations/matrix-vector products for ORSIR1 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.

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

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR(20)</td>
<td>235/236</td>
<td>†</td>
<td>152/153</td>
<td>146/147</td>
<td>140/141</td>
<td>†</td>
</tr>
<tr>
<td>A(20) $M_{\text{add+coa}}$</td>
<td>235/247</td>
<td>113/234</td>
<td>113/234</td>
<td>113/234</td>
<td>113/234</td>
<td>113/234</td>
</tr>
<tr>
<td>A(20) $M_{\text{mul+coa}}$</td>
<td>235/247</td>
<td>112/286</td>
<td>112/286</td>
<td>112/286</td>
<td>112/286</td>
<td>112/286</td>
</tr>
<tr>
<td>A(20) $M_{\text{mul+res}}$</td>
<td>235/247</td>
<td>110/321</td>
<td>110/321</td>
<td>110/321</td>
<td>110/321</td>
<td>110/321</td>
</tr>
</tbody>
</table>
Table 7: Number of iterations/matrix-vector products for MEMPLUS with $M^{(0)}$ an ILU-PACK preconditioner and $\eta = 0.01$

<table>
<thead>
<tr>
<th>Method</th>
<th>Dimension eigenspace</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR(20)</td>
<td></td>
<td>196/197</td>
<td>190/191</td>
<td>180/181</td>
<td>173/174</td>
<td>156/157</td>
<td>122/123</td>
<td>120/121</td>
</tr>
<tr>
<td>A(20) $M_{\text{add+coa}}$</td>
<td></td>
<td>196/206</td>
<td>122/308</td>
<td>121/307</td>
<td>109/269</td>
<td>101/246</td>
<td>101/246</td>
<td>101/246</td>
</tr>
<tr>
<td>A(20) $M_{\text{mul+coa}}$</td>
<td></td>
<td>196/206</td>
<td>120/382</td>
<td>101/310</td>
<td>101/311</td>
<td>94/277</td>
<td>94/277</td>
<td>94/277</td>
</tr>
<tr>
<td>A(20) $M_{\text{mul+res}}$</td>
<td></td>
<td>196/206</td>
<td>120/508</td>
<td>101/379</td>
<td>101/380</td>
<td>94/328</td>
<td>94/328</td>
<td>94/328</td>
</tr>
<tr>
<td>A(20) $M_{\text{coa}}$</td>
<td></td>
<td>196/206</td>
<td>151/165</td>
<td>121/137</td>
<td>121/138</td>
<td>121/138</td>
<td>121/138</td>
<td>121/138</td>
</tr>
<tr>
<td>DR(30)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>†</td>
</tr>
<tr>
<td>A(30) $M_{\text{coa}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>†</td>
</tr>
<tr>
<td>A(30) $M_{\text{exa}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>†</td>
</tr>
<tr>
<td>DR(35)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A(35) $M_{\text{coa}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A(35) $M_{\text{exa}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DR(40)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A(40) $M_{\text{coa}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A(40) $M_{\text{exa}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dimension eigenspace</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR(30)</td>
<td></td>
<td>†</td>
<td>†</td>
<td>†</td>
<td>†</td>
<td>†</td>
<td>155</td>
</tr>
<tr>
<td>A(30) $M_{\text{coa}}$</td>
<td></td>
<td>†</td>
<td>†</td>
<td>†</td>
<td>†</td>
<td>†</td>
<td></td>
</tr>
<tr>
<td>A(30) $M_{\text{exa}}$</td>
<td></td>
<td>†</td>
<td>†</td>
<td>†</td>
<td>†</td>
<td>†</td>
<td></td>
</tr>
<tr>
<td>DR(35)</td>
<td></td>
<td>†</td>
<td>†</td>
<td>†</td>
<td>123</td>
<td>†</td>
<td></td>
</tr>
<tr>
<td>A(35) $M_{\text{coa}}$</td>
<td></td>
<td>†</td>
<td>132</td>
<td>105</td>
<td>105</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>A(35) $M_{\text{exa}}$</td>
<td></td>
<td>†</td>
<td>132</td>
<td>104</td>
<td>104</td>
<td>104</td>
<td></td>
</tr>
<tr>
<td>DR(40)</td>
<td></td>
<td>225</td>
<td>112</td>
<td>112</td>
<td>104</td>
<td>†</td>
<td>92</td>
</tr>
<tr>
<td>A(40) $M_{\text{coa}}$</td>
<td></td>
<td>146</td>
<td>112</td>
<td>99</td>
<td>79</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>A(40) $M_{\text{exa}}$</td>
<td></td>
<td>144</td>
<td>112</td>
<td>100</td>
<td>79</td>
<td>79</td>
<td>79</td>
</tr>
</tbody>
</table>
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).

<table>
<thead>
<tr>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR(10)</td>
<td>97</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>84</td>
</tr>
<tr>
<td>A(10) $M_{coa}$</td>
<td>85</td>
<td>85</td>
<td>85</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>A(10) $M_{exa}$</td>
<td>85</td>
<td>85</td>
<td>85</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>DR(15)</td>
<td>92</td>
<td>84</td>
<td>80</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>A(15) $M_{coa}$</td>
<td>98</td>
<td>81</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>A(15) $M_{exa}$</td>
<td>98</td>
<td>80</td>
<td>79</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>DR(20)</td>
<td>88</td>
<td>82</td>
<td>77</td>
<td>77</td>
<td>75</td>
</tr>
<tr>
<td>A(20) $M_{coa}$</td>
<td>77</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
<tr>
<td>A(20) $M_{exa}$</td>
<td>77</td>
<td>75</td>
<td>75</td>
<td>75</td>
<td>75</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR(20)</td>
<td>190</td>
<td>180</td>
<td>173</td>
<td>156</td>
<td>122</td>
<td>120</td>
</tr>
<tr>
<td>A(20) $M_{coa}$</td>
<td>192</td>
<td>129</td>
<td>125</td>
<td>125</td>
<td>125</td>
<td>125</td>
</tr>
<tr>
<td>DR(40)</td>
<td>161</td>
<td>157</td>
<td>147</td>
<td>123</td>
<td>117</td>
<td>108</td>
</tr>
<tr>
<td>A(40) $M_{coa}$</td>
<td>134</td>
<td>129</td>
<td>115</td>
<td>105</td>
<td>105</td>
<td>105</td>
</tr>
</tbody>
</table>
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

```
function [x, error, innerIterations, flag, resvec, PRODUCTS] = 
    newGMRES( A, b, x, preconditioner, iterations, tol, 
    dimCoarse, harmonic )
global PRODUCTS;
% initialization
```
outputInfo = 1;
innerIterations = 0;
flag = 0;
resvec = zeros(iterations.restart * iterations.max + 1, 1);
colors = 'rgbyk';
maxOuter = ceil(iterations.max / iterations.restart);
preconditioner.spectral.levels = 0;
PRODUCTS = 0;
AIsReal = isreal(A);
bnrm2 = norm(b);
if (bnrm2 == 0.0)
bnrm2 = 1.0;
end
resvec(1) = norm(b-A*x, 2) / bnrm2;

% Lucky guess
if (resvec(1) < tol.gmres)
return
end

% initialize workspace
n = size(A, 1);
m = iterations.restart;
V = zeros(n,m+1);
H = zeros(m+1,m);
R = zeros(m+1,m);
Veps = cell(maxOuter, 1);
Av = cell(maxOuter, 1);
cs(1:m) = zeros(m,1);
sn(1:m) = zeros(m,1);
e1 = zeros(m+1,1);
e1(1) = 1.0;

% Begin iteration
for iter = 1:maxOuter
if outputInfo == 1
fprintf('

Iteration #
', iter);
end
PRODUCTS = PRODUCTS + 1;

r = precondition( b−A*x, A, preconditioner, Av, Veps, 1 );

rnorm = norm( r );

V(:,1) = r / rnorm;

s = rnorm*e1;

% Arnoldi

for i = 1:m
    PRODUCTS = PRODUCTS + 1;
    w = precondition( A*V(:,i), A, preconditioner, Av, Veps, 1 );

    for k = 1:i
        t = V(:,k)'*w;
        H(k,i) = t;
        w = w - t*V(:,k);
    end

    H(i+1,i) = norm( w );
    V(:,i+1) = w / H(i+1,i);
    R(:,i) = H(:,i);

end

% Givens rotation

innerIterations = 1 + innerIterations;

for k = 1:i-1

    temp = cs(k)*R(k,i) + sn(k)*R(k+1,i);
    R(k+1,i) = -conj(sn(k))*R(k,i) + cs(k)*R(k+1,i);
    R(k,i) = temp;

end

% ith rotational matrix

[cs(i),sn(i)] = rotmat( R(i,i), R(i+1,i) );

% approximate residual norm

temp = cs(i)*s(i);

s(i+1) = -conj(sn(i))*s(i);

s(i) = temp;

R(i,i) = cs(i)*R(i,i) + sn(i)*R(i+1,i);
R(i+1,i) = 0;

y = R(1:i,1:i) \ s(1:i);
xResult = x + V(:, 1:i)*y;
error = norm(b - A*xResult, 2) / bmrn2;
resvec(innerIterations+1) = error;
if outputInfo
  fprintf('Resvec: %e\n', error);
end
% Update approximation
if (error <= tol.gmres )
  x = xResult;
  return;
end
end

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

% Split vectors
if (AIsReal && ~isreal(EigVecH))
    EigVecH = filterEigenpairs(EigVecH);
end

% Compute approximate eigenvector of MA
Veps{preconditioner.spectral.levels} = V(:,1:m)*EigVecH;

% Compute coarse grid operator
PRODUCTS = PRODUCTS + size(EigVecH, 2);
Av{preconditioner.spectral.levels} = Veps{preconditioner.spectral.levels}'*A*Veps{preconditioner.spectral.levels};

if outputInfo == 1
    fprintf('Used %d vectors\n', size(EigVecH, 2));
    fprintf('Condition Av = %1.3e\n', cond(Av{preconditioner.spectral.levels}));
end

end

x = xResult;
end

A.2 Preconditioning implementation

function [z] = precondition( vec, A, data, Av, Veps, count )
global PRODUCTS;

if data.spectral.levels == 0 || data.spectral.type == 0
    % STANDARD PRECONDITIONER
    z = feval(data.func, data.initial, vec);
    return;
elseif data.spectral.type == 1
% add+coa SPECTRAL PRECONDITIONER

% High frequency correction
z = zeros(size(vec,1), 1);
for i=1:(data.spectral.mul+data.spectral.mul2)
    if count
        PRODUCTS = PRODUCTS + 1;
    end
    z = z + data.spectral.relax * feval(data.func, data.
        initial, vec - A*z);
end

% High frequency correction
z = z - Veps{1}*(Veps{1}'*z);

% Low frequency
for i = 1:data.spectral.levels
    z = z + Veps{i} *( Av{i} \ ( Veps{i}'*vec ));
end
return;

elseif data.spectral.type == 2
% mul+res SPECTRAL PRECONDITIONER
z = zeros(size(vec,1), 1);
% Pre-smoothing
for i=1:data.spectral.mul
    if count
        PRODUCTS = PRODUCTS + 1;
    end
    z = z + data.spectral.relax * feval(data.func, data.
        initial, vec - A*z);
end

% Coarse grid correction
if count
    PRODUCTS = PRODUCTS + 1;
end
\( z = z + V_{\text{eps}} \{1\} \times (A_{\text{eps}} \{1\} \backslash (V_{\text{eps}} \{1\}^T \times (\text{vec} - A^*z))) ; \)

\% Post-smoothing

for \( i = 1: \text{data.spectral.mu2} \)
    if count
        PRODUCTS = PRODUCTS + 1;
    end
    \( z = z + \text{data.spectral.relax} \times \text{feval}(\text{data.func}, \text{data}.\text{initial}, \text{vec} - A*z) ; \)
end

for \( i = 2: \text{data.spectral.levels} \)
    if count
        PRODUCTS = PRODUCTS + 1;
    end
    \( z = z + V_{\text{eps}} \{i\} \times (A_{\text{eps}} \{i\} \backslash (V_{\text{eps}} \{i\}^T \times \text{vec} ) ) ; \)
end

return ;

elseif \( \text{data.spectral.type} == 3 \)
    \% coa PRECONDITIONER
    \( z = \text{feval}(\text{data.func}, \text{data}.\text{initial}, \text{vec}) ; \)
    for \( i = 1: \text{data.spectral.levels} \)
        \( z = z + V_{\text{eps}} \{i\} \times (A_{\text{eps}} \{i\} \backslash (V_{\text{eps}} \{i\}^T \times \text{vec} ) ) ; \)
    end
    return ;

elseif \( \text{data.spectral.type} == 4 \)
    \% mul+coa SPECTRAL PRECONDITIONER
    \( z = \text{zeros}(\text{size}(\text{vec}, 1), 1) ; \)
    \% Pre-smoothing
    for \( i = 1: \text{data.spectral.mul} \)
        if count
            PRODUCTS = PRODUCTS + 1;
        end
    end
z = z + data.spectral.relax * feval(data.func, data.
   initial, vec - A*z);

% Coarse grid correction
if count
    PRODUCTS = PRODUCTS + 1;
end

z = z + Veps{1} * (Av{1} \ (Veps{1}'*(vec - A*z)));

% Post-smoothing
for i = 1:data.spectral.mu2
    if count
        PRODUCTS = PRODUCTS + 1;
    end
    z = z + data.spectral.relax * feval(data.func, data.
       initial, vec - A*z);
end

for i = 2:data.spectral.levels
    z = z + Veps{i} * (Av{i} \ (Veps{i}'*vec));
end
return;
end

A.3 Eigenpair implementation

function [ vectors_out ] = filter_eigenpairs( vectors )
    vectors_out = [];
    left = size(vectors, 2);
    while left > 0
        current = vectors(:, 1);
        if isreal(current)
            vectors_out = [vectors_out, current];
            vectors(:, 1) = [];
        else
            real_part = real(current);
            imag_part = imag(current);
            vectors_out = [vectors_out, real_part, imag_part];
        end
    end
if left == 1
    vectors(:, 1) = []; 
else if left > 1,
    vectors(:, 1:2) = []; 
end
end
left = size(vectors, 2);
end
end

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

A.4 Givens rotation

function [ c , s ] = rotmat( a , b )
% Compute the Givens rotation matrix parameters for a and b.
if ( b == 0.0 ),
c = 1.0;
s = 0.0;
elseif ( abs(b) > abs(a) ),
temp = a / b;
s = 1.0 / sqrt( 1.0 + temp^2 );
c = temp * s;
else
    temp = b / a;
c = 1.0 / sqrt( 1.0 + temp^2 );
s = temp * c;
end

References


