Krylov subspace methods for large linear systems of equations*

G.L.G. Sleijpen† and H.A. van der Vorst†

June 2, 1993

Abstract

When solving PDE's by means of numerical methods one often has to deal with large systems of linear equations, specifically if the PDE is time-independent or if the time-integrator is implicit. For real life problems, these large systems can often only be solved by means of some iterative method. Even if the systems are preconditioned, the basic iterative method often converges slowly or even diverges. We discuss and classify algebraic techniques to accelerate the basic iterative method. Our discussion includes methods like CG, GCR, ORTHODIR, GMRES, CGNR, Bi-CG and their modifications like GMRESR, CG-S, Bi-CGSTAB. We place them in a frame, discuss their convergence behavior and their advantages and drawbacks.

1 Introduction

Our aim is to compute acceptable approximations for the solution x of the equation

$$Ax = b, \tag{1}$$

where A and b are given, A is a non-singular $n \times n$-matrix, A is sparse, n is large and b an n-vector. We will assume A and b to be real, but our methods are easily generalized to the complex case.

Such equations arise frequently when solving numerically discretized PDE's, for instance, for stationary problems or with implicit time-integration. Solving non-linear equations by means of, for instance, Newton's method may also lead to such systems.

If n is large, direct methods are often too expensive and one has to switch to iterative methods. Usually the basic iterative method (Richardson) does not converge or converges very slowly. By preconditioning (Jacobi, Gauss–Seidel, SSOR, ILU, etc.), one may speed up the convergence, but it may still be too slow or even fail to converge. In this paper, we discuss algebraic acceleration techniques for these basic preconditioned iterative methods: we concentrate on techniques based on the matrix and discard the origin of the equation (for a discussion of Multigrid techniques, we refer to the next paper.

These acceleration techniques have resulted in methods like Conjugate Gradients (CG), GMRES, Bi-CG, and others.

For symmetric positive definite problems CG is the optimal acceleration technique. In practice, however, one has often to solve unsymmetric problems; diffusion-advection problems almost always lead to unsymmetric systems. For these problems, there are many generalizations of CG that accelerate (or aim to accelerate) the basic iterative method. They iteratively produce a sequence (x_k) (either explicitly or implicitly) of approximations x_k of the solution x.

*Part of this paper has been published in [34].

†Mathematical Institute, University of Utrecht, P.O.Box 80.010, NL-3508 TA Utrecht, The Netherlands.

In this paper we will focus on optimal iterative algorithms for nonsymmetric systems. These algorithms fall in two classes: GMRES and variants (optimal methods with long recurrences), and Bi-CG and variants (quasi-optimal methods with short recurrences).

By mixing methods from of these classes one hopes to preserve some good properties and to get rid of, or to weaken, the bad ones. Bi-CGSTAB is such a method. It was developed to stabilize Bi-CG by mixing it with a GMRES type of method.

There is no robust method that solves all problems:

(a) for mathematical reasons,
(b) for reasons of stability and
(c) in the event of only a limited amount of memory.

Sub (a): Consider some methods that are mathematically different (and not for subtle implementation reasons); for instance, GMRES, Bi-CGSTAB and CG applied to the normal equations. Then there are different linear systems $Ax = b$ such that each of the methods is the winner for one of these systems and the worst method for one of the others (see, [25]). Although the examples have to be selected to achieve this, they are certainly not exceptional. One may attempt to classify problems for which certain methods are superior (over others). Unfortunately, it will be hard in practice to determine the class in which a problem fits and, what will be more likely, the problem will often not fit nicely in one class but may have aspects of several classes. These aspects may turn up in different phases of a computation. It may even be the case that by slightly increasing the dimension of the problem, through grid-refinement, advantages of a selected method are lost.
Sub (b): In practice, the computational process will be contaminated by evaluation errors due to finite precision arithmetic. Methods that, for theoretical reasons, should perform the best for certain types of problems may fail to converge at all due these errors. Whether the evaluation errors affect the speed of convergence or not may depend dramatically on the type of problem and its size.
Sub (c): In view of the restricted computational resources, methods may have to be adapted if the size of the matrices involved is too large. Methods that are superior for a certain type of problem may not be adaptable or may lose their superiority when adapted.

In spite of these observations, we have to take the situation as it is. There is no practical alternative and many problems can be solved much faster or even can be solved at all only because of these techniques.

The CG method was introduced in 1954 by Hestenes and Stiefel [19]. But it did not get much attention until in 1971. Then Reid [29] showed how this method can be used to our advantage for certain systems. Since then there has been a growing interest in the CG method mainly for its use as an acceleration technique [2]. Meijerink and van der Vorst [23] showed how efficient the CG method can be if combined with the ILU preconditioner. Pure generalizations of CG to solve non-symmetric equations were introduced in the late seventies and early eighties. GMRES and variants use orthogonal basis. Algorithms of this type were proposed by Vinsome [41], Young and Jea [47], Saad [30], Elman [9], Axelsson [3] and others. The GMRES algorithm of Saad and Schultz [32] from 1986 seems to be the most popular one of this type. In 1952, Lanczos [21] used bi-orthogonality relations to reduce iteratively a matrix to tri-diagonal form and he suggested how to solve non-symmetric linear systems from the reduced tri-diagonal form. Later, his ideas were adapted for a variety of methods, such as Bi-CG [11], QMR [13] and others. In 1989, Sonneveld [38] introduced a "squared Bi-CG algorithm", the CG-S algorithm: he used the observation that the computational effort in the Bi-CG algorithm to get bi-orthogonality can be used for an additional reduction of the residuals. The "squared-QMR" algorithm [12] is the most robust and efficient algorithm of pure Bi-CG type. The steps in the GMRES-like algorithms are increasingly expensive while the algorithms of Bi-CG type often converge slowly and are sensitive to evaluation errors. Recently, a number of mixed methods were introduced that often suffer less from these drawbacks (see [44], [46], [7], [31], [5], [4], [20], [33] etc.). A lot of research is still focussed on this subject.

In this paper, we present some of the most powerful methods, discuss the convergence behavior in relation to properties of A, give algorithmical realizations of the methods, show how they are related and discuss briefly their advantages and drawbacks. In our discussions we will sacrifice mathematical rigor for short presentations. In particular, we will not discuss all kind of degenerate cases, and refer the reader to textbooks for a more concise description (e.g., [40, 15]).

1.1 Notations and conventions

We choose some inner product $(\cdot, \cdot)$ on the space of n-vectors, $\|\cdot\|$ is the associated norm.

In the analysis of the convergence behavior the spectral radius of certain matrices plays an important role: if B is a square matrix then the **spectral radius** $\rho(B)$ of B is $\max|\lambda|$, where the maximum is taken over all eigenvalues λ of B. $\sigma(B)$ is the spectrum of B, the collection of all eigenvalues of B.

The **condition number** $\mathcal{C}(V)$ of a matrix V is the quotient of the largest and smallest singular value of V. If V is a non-singular $n \times n$ matrix then $\mathcal{C}(V) = \|V\|\,\|V^{-1}\|$. If, in addition, V is real and symmetric then $\mathcal{C}(V) = \max|\lambda| / \min|\lambda|$, where the maximum and minimum is taken over all $\lambda \in \sigma(V)$.

A square matrix B is **diagonalizable** if $D \equiv V^{-1}BV$ is a diagonal matrix for some non-singular matrix V. The columns of V are eigenvectors of B and the diagonal elements of D are eigenvalues. The **condition number $\mathcal{C}_E(B)$ of the eigenvector problem** of B is defined by $\mathcal{C}_E(B) \equiv \min \mathcal{C}(V)$ where the minimum is taken over all non-singular matrices V for which $V^{-1}BV$ is diagonal. B is diagonalizable and $\mathcal{C}_E(B) = 1$ if, for instance, B is real and symmetric.

We assume that together with A also some $n \times n$ non-singular matrix K has been given that serves as a preconditioner (cf. 2.1); AK^{-1} is the preconditioned matrix. This preconditioned matrix plays an important role. It determines the convergence behavior of the methods to be discussed. For ease of presentation we denote this matrix by $\mathbb{A}$: $\mathbb{A} \equiv AK^{-1}$.

In the discussion of the convergence behavior we assume that $\mathbb{A}$ is diagonalizable, thus avoiding technical details. By employing the notion of "pseudo spectrum" [25, 39] one also can obtain results on the convergence in case $\mathbb{A}$ is non-diagonalizable. This approach gives results that, also in the diagonalizable case, may be better, especially if $\mathcal{C}_E(\mathbb{A})$ is large. We refer to the literature for details.

The **Krylov subspace** $\mathcal{K}_k(\mathbb{A}; v_0)$ of order k generated by $\mathbb{A}$ and v_0 is the space spanned by $v_0, \mathbb{A}v_0, \ldots, \mathbb{A}^{k-1}v_0$. Our iterative methods take their updates from such Krylov subspaces. In order to avoid technical subtilities, we will assume throughout this paper that the vectors $v_0, \ldots, \mathbb{A}^{k-1}v_0$ are linearly independent. This assumption limits implicitly the size of k: for instance, by "for all k" we actually mean "for all k for which the dimension of the Krylov subspace $\mathcal{K}_k(\mathbb{A}; v_0)$ of interest is k". In practice, we are interested in k's that are small compared to n.

We call a sequence $v_0, v_1, \ldots$ of n-vectors a **basis of the Krylov space** generated by $\mathbb{A}$ if, for each k, the vectors $v_0, \ldots, v_{k-1}$ span the Krylov subspace $\mathcal{K}_k(\mathbb{A}; v_0)$.

The elements of Krylov subspaces can be described in terms of polynomials: if $\mathcal{P}_k$ denotes the space of all polynomials of degree $\leq k$ then

$$\mathcal{K}_k(\mathbb{A}; v_0) = \{q(\mathbb{A})v_0 \,\big|\, q \in \mathcal{P}_{k-1}\} \tag{2}$$

The residuals of our methods can be described by polynomials that are 1 in 0. $\mathcal{P}_k^1$ is the set of all polynomials q of degree $\leq k$ with $q(0) = 1$.

```
choose x_0,
k = -1,     r_0 = b - A x_0
repeat until ||r_{k+1}|| is small enough:
    { k := k + 1
    { solve u_k from K u_k = r_k,  c_k = A u_k,
    { x_{k+1} = x_k + u_k,    r_{k+1} = r_k - c_k
```

ALGORITHM 1: The basic iterative algorithm.

2 The basic iterative method

2.1 The method

Classical iterative methods (Jacobi, Gauss–Seidel, SOR, etc.) are based on a splitting of A:

$$A = K - R \tag{3}$$

in which K is a non-singular $n \times n$-matrix. Starting with some suitable approximation x_0 for the solution x the sequence of approximations x_k is produced by iterating

$$x_{k+1} = x_k + K^{-1}(b - Ax_k) \quad \text{for all} \quad k. \tag{4}$$

From the expression (5) below for the **errors** $e_k \equiv x - x_k$ and (6) for the **residuals** $r_k \equiv b - Ax_k$ we see that the convergence of (x_k) to x is determined by how close $K^{-1}A$ or AK^{-1} is to I: the convergence of (e_k) is determined by $K^{-1}A$ (cf. (5)), while the convergence of (r_k) depends on AK^{-1} (cf. (6)).

The matrix K^{-1} is almost never constructed explicitly. It is often more economical to solve $u \equiv K^{-1}r$ from $Ku = r$. This means that K has to be chosen so that this system can be solved very efficiently, for instance $K = I$ (Richardson) or $K = \text{diag}(A)$ (Jacobi) are popular choices. Other popular choices for K are the lower triangular part of A (Gauss-Seidel), and the tridiagonal part of A (line-Jacobi). For the slightly more complicated Incomplete LU decompositions see, for instance, [23, 24]. These standard processes converge slowly, but as we will see they can be accelerated by combining information of successive iteration steps.

Note that the method in (4) is equivalent to the Richardson iteration applied to the preconditioned system $Bx = c$, with $B = K^{-1}A$ and $c = K^{-1}b$. Hence, there would be no loss of generality in taking the Richardson iteration as our starting point. The subsequent theory to accelerate the basic iteration method in (4) actually is a theory for the preconditioned Richardson iteration process.

Since $x = x + K^{-1}(b - Ax)$ we have that

$$e_{k+1} = (I - K^{-1}A)e_k, \tag{5}$$

and, since $r_k = Ae_k$,

$$r_0 = b - Ax_0, \quad r_{k+1} = r_k - Au_k \quad \text{with} \quad u_k = K^{-1}r_k. \tag{6}$$

The sequence of errors converges to 0 whenever the sequence of residuals does. We would like to stop the iteration if $\|e_k\|$ is small enough. Normally, e_k will not be available and stopping criteria have to be based upon the knowledge of r_k.

Note that

$$x_{k+1} = x_k + u_k. \tag{7}$$

The equations (6) and (7) constitute a practical algorithmical representation of the basic iterative method in (4) given in ALGORITHM 1.

In (6) we "correct" the k-th residual r_k to obtain a new and hopefully smaller residual r_{k+1} by some vector of the form Au_k, and we know the original u_k. We use this original u_k, or **searching direction**, in (7) to update the approximation x_k. All the subsequent methods are of this type:

Construct some c_k and u_k for which $c_k = Au_k$ and
for which $r_{k+1} \equiv r_k - c_k$ is small in some sense.
Then r_{k+1} is the residual of $x_{k+1} \equiv x_k + u_k$.

Our theoretical and practical efforts will be directed towards producing in an efficient way a sequence (r_k) which converges quickly to 0. As a "side product" we will then be able to compute efficiently the sequence (x_k) of approximations.

2.2 Convergence

From (6), we have that

$$r_k = (I - \mathbb{A})^k = (I - AK^{-1})^k r_0 \quad \text{for all} \quad k. \tag{8}$$

The preconditioned matrix $\mathbb{A} \equiv AK^{-1}$ plays an important role also in the other methods to be discussed in this paper.

In each iteration step the residual will essentially be reduced by a factor equal to the spectral radius $\rho(I - \mathbb{A})$ of the matrix $I - \mathbb{A}$. To be more precise

$$\frac{\|r_k\|}{\|r_0\|} \leq \mathcal{C}_E(\mathbb{A})\ \rho(I - \mathbb{A})^k \quad \text{for all} \quad k. \tag{9}$$

If one has to carry out a large number of iteration steps, then the reduction of the residual in each step will eventually be (almost) equal to this spectral radius (except in the non-generic case where r_0 does not have a component in certain eigenvector directions). So, we may state that $\rho(I - \mathbb{A})$ is the (asymptotic) **reduction factor** of the **residual** of the iterative process. This factor $\rho(I - \mathbb{A})$ will be manifest in an early stage of the iteration process already if $\mathcal{C}_E(\mathbb{A})$ is not too large. For convergence one should have that $\rho(I - \mathbb{A}) < 1$. For fast convergence, $\rho(I - \mathbb{A})$ should be small.

Improving the convergence

By including a parameter, i.e. by working with $\frac{1}{\alpha}K$ instead of K the residual (asymptotic) reduction factor becomes $\rho(I - \alpha\mathbb{A})$, which may be smaller than $\rho(I - \mathbb{A})$ for some $\alpha \neq 1$. Therefore, for some appropriate **relaxation parameter** α one may improve the convergence.

Example 1 Suppose $\sigma(\mathbb{A}) \subset [\lambda_1, \lambda_n] \subset (0, \infty)$. Put $\mathcal{C} := \frac{\lambda_n}{\lambda_1}$.
With $\alpha := \frac{2}{\lambda_1 + \lambda_n}$ we have that

$$\rho(I - \alpha\mathbb{A}) = 1 - \alpha\lambda_1 = \frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} = \frac{1 - \frac{1}{\mathcal{C}}}{1 + \frac{1}{\mathcal{C}}}.$$

Hence, in this case

$$\frac{\|r_k\|}{\|r_0\|} \leq \mathcal{C}_E(\mathbb{A}) \left(\frac{1 - \frac{1}{\mathcal{C}}}{1 + \frac{1}{\mathcal{C}}}\right)^k \approx \mathcal{C}_E(\mathbb{A}) \exp\left(-\frac{2k}{\mathcal{C}}\right). \tag{10}$$

We now can estimate how many iteration steps are required to reduce the residual by a factor $\varepsilon > 0$:

$$\frac{\|r_k\|}{\|r_0\|} \leq \varepsilon \quad \text{if} \quad k \gtrsim \frac{1}{2}\mathcal{C} \log \frac{C_E(\mathbb{A})}{\varepsilon}. \tag{11}$$

If A is the d-dimensional Laplace operator discretized by symmetric finite differences and K stems from ILU preconditioning then $\mathbb{A}$ is diagonalizable, $\mathcal{C} = \mathcal{O}(h^{-2}) \quad (h \to 0)$, where h is the mesh size and $\mathcal{C}_E(\mathbb{A}) = \sqrt{\mathcal{C}(K)} = \mathcal{O}(1) \quad (h \to 0)$. In this case[1], the required number of iterations will be proportional to h^{-2}.

In order to make this relaxation procedure practical one should have fairly accurate estimates for the extremal eigenvalues of $\mathbb{A}$ (cf. [40]). Unfortunately, in general this information is not readily available. However, since one is free to select a different relaxation parameter α_k in each iteration step, one can easily obtain relaxation parameters that help to reduce the residual. For instance, by choosing α_k such that $\|r_{k+1}\| = \|r_k - \alpha_k \mathbb{A} r_k\|$ is minimal, or equivalently, $r_{k+1} \perp \mathbb{A} r_k$, we obtain the so-called **LMR** (**local minimum residual**) method (related to the steepest descent method). Although this is certainly an improvement over the standard iteration approach, it can still not compete with the methods to come. However, as we will see, this LMR approach gives the clue for the global minimum residual methods.

3 Krylov subspace methods

For the k-th residual in the minimal residual method, we have that

$$r_k = (I - \alpha_k \mathbb{A})(I - \alpha_{k-1}\mathbb{A}) \cdot \ldots \cdot (I - \alpha_1 \mathbb{A}) r_0. \tag{12}$$

Obviously r_k is a special combination of the vectors r_0, $\mathbb{A} r_0$, $\ldots$, $\mathbb{A}^k r_0$. These vectors form a basis for the Krylov subspace $\mathcal{K}_{k+1}(\mathbb{A}; r_0)$, and the question arises whether we may be able to select a better vector from this subspace.

With $q_k(t) = \prod_{j=1}^{k}(1 - \alpha_j t)$, q_k is a polynomial of degree $\leq k$ and $q_k(0) = 1$. Since $q_k(0) = 1$, there is a polynomial $\tilde{q}_k$ of degree $\leq k-1$ for which $q_k(t) = 1 - t\tilde{q}_k(t)$. Hence, we can rewrite (12) as

$$r_k = q_k(\mathbb{A}) r_0 = r_0 - \mathbb{A}\tilde{q}_k(\mathbb{A}) r_0. \tag{13}$$

We call any **method** that produces a sequence (x_k) of n-vectors **polynomial** if, for each k, the k-th residual $r_k = b - Ax_k$ is as in (13), for some $q_k \in \mathcal{P}_k^1$. Then $x_k = x_0 + K^{-1}\tilde{q}_k(\mathbb{A}) r_0$. Hence the update $K(x_k - x_0)$ of the preconditioned initial approximate Kx_0 belongs to the Krylov subspace $\mathcal{K}_k(\mathbb{A}; r_0)$. For this reason, polynomial methods are also called **Krylov subspace methods**.
Obviously, the basic iterative process is a polynomial method as well (with $q_k(t) = (1-t)^k$).

The residual r_k of the Krylov subspace method as in (13) can be bounded by

$$\frac{\|r_k\|}{\|r_0\|} \leq \|q_k(\mathbb{A})\| \leq \mathcal{C}_E(\mathbb{A}) \max\left\{ |q_k(\lambda)| \,\Big|\, \lambda \in \sigma(\mathbb{A}) \right\}. \tag{14}$$

LMR optimizes the reduction of r_{k-1} only by taking a factor times $\mathbb{A} r_{k-1}$. By taking into account the images $\mathbb{A} r_0, \ldots, \mathbb{A} r_{k-1}$ of all previous residuals, we may be able to obtain a smaller residual r_k. With $r_j = q_j(\mathbb{A}) r_0$, the r_j form a basis of this Krylov subspace $\mathcal{K}_k(\mathbb{A}; r_0)$ if any of the polynomials q_j is of exact degree $j \quad (j < k)$: then

$$\mathcal{K}_k(\mathbb{A}; r_0) = \operatorname{span}(r_0, r_1, \ldots, r_{k-1}). \tag{15}$$

r_j is the j-th residual of a Krylov subspace method whenever $r_j - r_0 \in \mathbb{A}\mathcal{K}_j(\mathbb{A}; r_0)$. Now, let us suppose that all r_j for $j < k$ are residuals of a Krylov subspace method. Then, $r_k - r_{k-1} \in$

[1] Note that the dimensionality of the underlying domain on which the Laplace operator acts does not play a role.

$\mathbb{A}\mathcal{K}_k(\mathbb{A}; r_0)$ if and only if $r_k - r_0 \in \mathbb{A}\mathcal{K}_k(\mathbb{A}; r_0)$. Hence, r_k equals r_{k-1} except for some linear combination of $\mathbb{A}r_j$ ($j < k$) if and only if r_k is the k-th residual of some Krylov subspace method. Apparently, instead of looking for combinations of images of previous residuals that reduce r_{k-1} well we may also look for polynomials $q_k \in \mathcal{P}_k^1$ that reduce r_0 well.

Recall that the speed of convergence of the basic iterative process may be improved by relaxation parameters, either by selecting à priori an appropriate parameter or by selecting parameters that depend on the computed residuals. Similarly, in the class of Krylov subspace methods one can distinguish two approaches: (i) the **stationary method** where one selects a sequence $(\tilde{q}_k)$ of polynomials in advance and (ii) the **optimal** or **non-stationary method** where one generates (optimal) polynomials in the course of the computation. We wish to point out that, for efficiency reasons, the polynomials are hardly ever manifest in the algorithmic form of any of the Krylov subspace methods: the representation of the residuals by means of polynomials mainly serves theoretical purposes.

3.1 Stationary Krylov subspace methods

Before turning to the optimal methods in the next section, we briefly discuss the stationary ones.

In view of (14), for stationary methods, one would like to have the q_k in $\mathcal{P}_k^1$ that minimizes the expression $\max_{\lambda\in\sigma(\mathbb{A})} |q_k(\lambda)|$. Unfortunately, as observed before, the spectrum is unknown. Instead, one determines the q_k that approximately solves the minimization problem:

$$\text{find} \quad q \in \mathcal{P}_k^1 \quad \text{for which} \quad \max_{\lambda\in\mathbf{G}} |q(\lambda)| = \varepsilon_k(\mathbf{G}), \tag{16}$$

where $\mathbf{G}$ is some "nice" set in $\mathbb{C}$ that contains $\sigma(\mathbb{A})$ tightly and that does not contain 0 (recall that $q_k(0) = 1$), and where $\varepsilon_k(\mathbf{G})$ is defined by

$$\varepsilon_k(\mathbf{G}) \equiv \min_{q\in\mathcal{P}_k^1} \max \left\{ |q(\lambda)| \,\Big|\, \lambda \in \mathbf{G} \right\}. \tag{17}$$

For the method defined by the approximately minimizing q_k, (14), the fact that

$$\sigma(\mathbb{A}) \subset \mathbf{G} \quad \Rightarrow \quad \max\{|q(\lambda)| \,\big|\, \lambda \in \sigma(\mathbb{A})\} \leq \max\{|q(\lambda)| \,\big|\, \lambda \in \mathbf{G}\} \tag{18}$$

and the following proposition gives information on its ability to reduce residuals in case $\mathbf{G}$ is an ellipse $\mathbf{E}$ (to be more precise, in case $\mathbf{G}$ is a set of which the boundary is an ellipse) containing $\sigma(\mathbb{A})$.

Proposition 1 *Consider some $\beta_1, \beta_2 \in \mathbb{C}$.*
For $\rho \geq 1$, $\mathbf{E}_\rho$ is the ellipse with focii β_1, β_2 given by:

$$\mathbf{E}_\rho \equiv \left\{ \zeta \in C \,\Big|\, \frac{|\zeta - \beta_1| + |\zeta - \beta_2|}{|\beta_1 - \beta_2|} \leq \frac{1}{2}(\rho + \rho^{-1}) \right\}.$$

If $R \geq 1$ *is such that* $\frac{|\beta_1|+|\beta_2|}{|\beta_1-\beta_2|} = \frac{1}{2}(R + R^{-1})$, *(0 is on the boundary of $\mathbf{E}_R$) then*

$$\varepsilon_k(\mathbf{E}_\rho) \leq \frac{\rho^k + \rho^{-k}}{R^k + R^{-k}} \qquad \textit{for any } \rho \geq 1.$$

The proposition can also be used if $\mathbf{G}$ is the union of a set $\mathbf{F}$ and an ellipse $\mathbf{E}$ containing $\sigma(\mathbb{A})\setminus \mathbf{F}$, where $\mathbf{F}$ consists of (hopefully) accurate approximations $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_l$ of a few isolated eigenvalues $\lambda_1, \ldots, \lambda_l$ of $\mathbb{A}$. In such a case, consider q_k of the form $q_k(t) = \widehat{q}_{k-l}(t)\Pi_{j=1}^l(1 - t/\tilde{\lambda}_j)$, where $\widehat{q}_{k-l} \in \mathcal{P}_{k-l}^1$ solves (approximately) the minimization problem for $\mathbf{E}$.

The proof of the proposition uses shifted and scaled Chebychev polynomials (where the interval $[-1, +1]$ is shifted to $[\beta_1, \beta_2]$ and the polynomials are scaled such that their value in 0 is 1). In general, however, these adapted Chebychev polynomials do not solve the minimization problems on the ellipse exactly, but they are quite useful as we also will see below.

Example 1 (continued 1)
The interval $[\lambda_1, \lambda_n]$ is the degenerated ellipse $\mathbf{E}_\rho$ with focii $\beta_1 = \lambda_1$, $\beta_2 = \lambda_n$ and $\rho = 1$. Now

$$R = \frac{1+\sqrt{\frac{1}{\mathcal{C}}}}{1-\sqrt{\frac{1}{\mathcal{C}}}}.$$

Therefore, if we use shifted and scaled Chebychev polynomials (Chebychev iteration), we can bound the residual r_k by

$$\frac{\|r_k\|}{\|r_0\|} \lesssim \mathcal{C}_E(\mathbb{A}) \left(\frac{1-\sqrt{\frac{1}{\mathcal{C}}}}{1+\sqrt{\frac{1}{\mathcal{C}}}}\right)^k \approx \mathcal{C}_E(\mathbb{A}) \exp\left(-\frac{2k}{\sqrt{\mathcal{C}}}\right). \tag{19}$$

Compared to the basic iterative method with optimal relaxation parameter, the number of steps, we need[2] to reduce the initial residual, by a factor ε, is then less by a multiplicative factor $\sqrt{\mathcal{C}}$:

$$\frac{\|r_k\|}{\|r_0\|} \leq \varepsilon \quad \text{if} \quad k \gtrsim \frac{1}{2}\sqrt{\mathcal{C}} \log \frac{\mathcal{C}_E(\mathbb{A})}{\varepsilon}. \tag{20}$$

In case of the discretized Laplace operator, the number of steps required is proportional to h^{-1} (compare this to the $\mathcal{O}(h^{-2})$ for the unaccelerated standard method) .

In an arbitrary stationary polynomial method, x_k and r_k are computed as follows.
Find an appropriate polynomial $q_k \in \mathcal{P}_k^1$. Factorize $q_k(t) = \prod_{j=1}^k (1-\alpha_j t)$.
Choose an x_0. Take, $\tilde{x}_0 = x_0$, $\tilde{r}_0 = r_0 = b - Ax_0$, and compute iteratively $\tilde{x}_j$ and $\tilde{r}_j$ by

$$\tilde{r}_j = \tilde{r}_{j-1} - \alpha_j \mathbb{A} \tilde{r}_{j-1}, \quad \tilde{x}_j = \tilde{x}_{j-1} + \alpha_j K^{-1} \tilde{r}_{j-1} \quad (j = 1, \ldots, k). \tag{21}$$

Then $r_k = \tilde{r}_k$ and $x_k = \tilde{x}_k$.
Since the polynomials q_k will be designed to produce small residuals r_k, one may not expect that the intermediate residuals $\tilde{r}_j$ are small. So, in general, the stationary methods are rather inefficient if one has to monitor the residuals r_k in order to obtain the approximation with the desired accuracy. Therefore, the stationary polynomial methods are usually applied as follows (either (i) or (ii)).

- **(i)** Choose $k \in \mathbb{N}$ such that the residual r_k is known to be small enough. Perform one cycle (21).
- **(ii)** Choose some $k \in \mathbb{N}$. Perform one cycle (21). If the residual r_k computed by (21) turns out not to be small enough then perform another cycle (21) now starting with x_k instead of x_0. Repeat this approach until the resulting residual is small enough.

In this last application, one actually uses the basic iterative method in which every single step by itself involves an iteration of k-steps: thus, the stationary method is a preconditioning method known as **polynomial preconditioning**.

As observed above, the shifted and scaled Chebychev polynomials do not solve the minimization problem on an ellipse $\mathbf{E}$ exactly. However, as far as it concerns minimization, they do quite well. Moreover, they allow an efficient iterative use of the stationary polynomial method: there exists an algorithm, the so-called **Chebychev iteration**, in which the computation of r_k only involves r_{k-1} and r_{k-2} plus one matrix-vector multiplication, two vector updates, but no inner-products. None of the polynomial methods (except for obvious ones like the basic iterative method and LMR) can compute r_k using r_{k-1} only. Chebychev iteration produces rather efficiently the sequence of residuals r_k. Since it does not require the evaluation of inner products it is of special interest for implementation on certain parallel computers. The residuals converge

[2] Both the estimates in (10) and (19) are asymptotically sharp in some generic sense.

relatively fast towards 0 (relatively as compared to the basic iteration with optimal relaxation parameter or the LMR method) specially in cases where the spectrum of $\mathbb{A}$ can be enclosed by a flat ellipse that does not contain 0.

The success of stationary polynomial methods depends on how well one can bound the spectrum of $\mathbb{A}$ in advance. Since, in general the spectrum is unknown, this is not an easy job. However, specifically if one wishes to solve a number of closely related equations, the effort in finding (near) optimal polynomials may be worthwhile. A method that works well for one of the equations may perform well also for the others that are "close by". Moreover, one may extract information concerning the spectrum in the course of the computation, information that can be used to improve the polynomials. One may encounter closely related systems of equations, for instance, on consecutive time levels when solving a time dependent partial differential equation by the method of lines, using an implicit time integrator, or in consecutive iteration steps when solving non-linear equations by Newton's method.

4 Optimal Krylov subspace methods

For (non-stationary) polynomial methods, one would like to have residuals r_k as in (13) that are as small as possible. In the numerical literature, this is interpreted in two different ways. One way is to determine a polynomial q_k in $\mathcal{P}_k^1$ so that $\|r_k\|$ is minimal, where $r_k = q_k(\mathbb{A})r_0$, i.e.

$$\|r_k\| = \min\{\|q(\mathbb{A})r_0\| \,\big|\, q \in \mathcal{P}_k^1\} = \min\{\|r_0 - \mathbb{A}p(\mathbb{A})r_0\| \,\big|\, p \in \mathcal{P}_{k-1}\} \tag{22}$$

$$= \min\{\|r_0 - \mathbb{A}u\| \,\big|\, u \in \mathcal{K}_k(\mathbb{A}; r_0)\} \tag{23}$$

$$= \min\{\|r_0 - c\| \,\big|\, c \in \mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)\} \quad (\Leftrightarrow \;\; r_k \perp \mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)\;), \tag{24}$$

the other way is to determine r_k in such a way that

$$r_k \perp \mathcal{K}_k(\mathbb{A}; r_0), \tag{25}$$

i.e., the new residual should be free of components in the space that has already been explored. Note that the Krylov subspaces in both approaches are generated with different vectors (cf. (24) and (25)).

Though the first approach, the **MR** (global **Minimum Residual**) approach, seems more obvious, it leads to slightly more complicated algorithms, whereas the second approach, the **OR** (**Orthogonal Residual**) approach, has led to more popular algorithms. For instance, if the preconditioned matrix $\mathbb{A}$ is symmetric, then the orthogonality approach leads to the famous Conjugate Gradients method. Successful application of this method is restricted to symmetric *positive definite* matrices, while for other symmetric matrices it is better to rely on the MINRES [27] algorithm, which is based on the MR approach.

Likewise, in the unsymmetric case we obtain GMRES [32] by following the minimization approach, and FOM [30], if we follow the orthogonality approach. Depending on the particular construction of the minimal residual, or the orthogonal residual, we obtain differently named methods, but if they do not break down, then in exact arithmetic they produce the same results as either GMRES or FOM. For instance, methods as GCR, ORTHOMIN, GENCR, and ORTHODIR, are mathematically equivalent with GMRES, and besides disadvantages they may also have certain advantages over GMRES. For instance, in GMRES the approximated solution is only available at the very end of the iteration process, while in the less-robust GCR algorithm one has the current approximation available in each iteration step.

Although GMRES is more popular and more efficient than GCR, we mainly concentrate on GCR: GCR allows adaptations — algorithms that do not represent the MR approach — that are often more efficient with respect to both use of memory and computational costs. GMRES is less adjustable to these situations. Moreover, there is a variant of GCR that is competitive to

GMRES (cf. 4.2.2). We refer the reader to literature for more details on the other algorithms of MR- and OR-type. We will discuss GCR, ORTHODIR, GMRES, and FOM.

4.1 Convergence

In this subsection, we consider, for the discussion of convergence, some sequence of residuals r_k^M that satisfy the minimization property. We assume exact arithmetic. For finite precision arithmetic the situation is much more complicated and less well understood (see, e.g. [17, 26, 43]).

There is a close relationship between the convergence behavior of the MR approach and the OR approach given by the following proposition (cf. [45]).

Proposition 2 *Consider the sequence* (r_k^F) *of residuals obtained by the OR approach with* $r_0^F = r_0^M = r_0$. *We put* $\sigma_k \equiv \|r_k^M\|/\|r_{k-1}^M\|$. *Then* $\sigma_k \leq 1$ *and*

$$\|r_k^F\| = \frac{1}{\sqrt{1-\sigma_k^2}}\|r_k^M\| \quad \textit{if} \quad \sigma_k < 1.$$

Apparently, from a practical point of view there is not much difference in the speed of convergence between both processes: if MR converges well, that is there is significant progress in one iteration step, then OR does about as well in the same iteration step. However, when MR stagnates temporarily ($\sigma_k \approx 1$), then OR may break down. Therefore, we will restrict our discussion of the convergence to the MR approach.

The following theorem estimates the reduction of the residuals.

Theorem 3 *Then*

$$\frac{\|r_k^M\|}{\|r_0\|} \leq \min_{q \in \mathcal{P}_k^1} \frac{\|q(\mathbb{A})r_0\|}{\|r_0\|} \leq \min_{q \in \mathcal{P}_k^1} \|q(\mathbb{A})\| \leq \mathcal{C}_E(\mathbb{A})\varepsilon_k(\sigma(\mathbb{A})), \tag{26}$$

where ε_k *is as defined in* (17).

Apparently, among all Krylov subspace methods, the MR approach gives the best reduction for the residual and it does not require any à priori knowledge of $\sigma(\mathbb{A})$. If, for instance, $\sigma(\mathbb{A})$ can be bounded by an ellipse $\mathbf{E}$ then proposition 1 gives à priori information on the norm of r_k^M, because: $\varepsilon_k(\sigma(\mathbb{A})) \leq \varepsilon_k(\mathbf{E})$.

The MR approach has a number of advantages over any stationary polynomial method:

(i) it does not require any information on the spectrum of $\mathbb{A}$;

(ii) it attains the fastest speed of convergence possible by any Krylov subspace method;

(iii) the convergence accelerates in the course of the computation.

In view of (26), (i) and (ii) need no additional comment. However, we wish to point out that, although the convergence properties do not depend on the knowledge of the spectrum, they do essentially depend on the spectrum. Given a matrix $\mathbb{A}$, we can not change its spectrum and thus influence the speed of convergence. However, only A is given and the preconditioning K has been added precisely for the purpose of improving the convergence. Therefore, it is important, for the construction of a good preconditioner, to know how the speed of convergence and spectral properties are related (see, for instance, [42]). Also with this observation in mind, we will now discuss (iii).

The acceleration can be understood as follows.
Note that r_k^M is perpendicular to $\mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)$. As we learn from the power method (for the computation of eigenvalues), the Krylov space $\mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)$ contains a vector that approximates the dominant eigenvector (i.e. the eigenvector of which the eigenvalue has largest absolute

value). Since, for any $\kappa \in \mathbb{C}$, this Krylov space is equal to $\mathcal{K}_k(\mathbb{A} - \kappa I; \mathbb{A} r_0)$ the Krylov space tends to contain good approximations of *all* extremal eigenvectors for increasing k, that is of all eigenvectors of which the eigenvalues are extremal (the eigenvalues that lie on the boundary of the convex hull of the spectrum; the shift by κ does not affect the eigenvectors, but any extremal eigenvalue can be shifted by some κ to be dominant). Moreover, from the k-th step on, the process actually takes place in a space perpendicular to $\mathcal{K}_k(\mathbb{A}; \mathbb{A} r_0)$ (as we will see in e.g. section 6), hence in a space approximately perpendicular to the span of all extremal eigenvectors. Therefore, from the k-th step on, the convergence behaves as if these extremal eigenvectors are not longer present. If the spectrum determines the convergence (as (26) suggests and experiments confirm) then, from the k-th step on, the convergence can be considered to be determined by a smaller spectrum (the original one minus the extremal eigenvalues) and a higher speed may be expected from then on. For a more extended discussion we refer to [37].

The following theorem (cf. [45]) describes how the residuals decrease when certain eigenvalues have been approximated by so-called **Ritz values** (of $\mathbb{A}$ of order k), that is by some eigenvalues of $P_k \mathbb{A}_{|\mathcal{K}_k(\mathbb{A}; r_0)}$, where P_k is the orthogonal projection on the Krylov space $\mathcal{K}_k(\mathbb{A}; r_0)$. The theorem does not tell when and which eigenvalues or eigenvectors have been approximated in the process.

Theorem 4 *For $k \in \mathbf{N}$, $\lambda \in \sigma(\mathbb{A})$ and a Ritz value θ of $\mathbb{A}$ of order k we have that*

$$\frac{\|r^M_{k+l}\|}{\|r^M_k\|} \leq \frac{F_k}{\sqrt{1-\sigma_k^2}} \mathcal{C}_E(\mathbb{A})\, \varepsilon_l(\sigma(\mathbb{A}) \backslash \{\lambda\}) \quad \textit{for all} \quad l \in \mathbb{N}, \tag{27}$$

where,

$$F_k \equiv \frac{|\theta|}{|\lambda|} \max\left\{ \frac{|\mu - \lambda|}{|\mu - \theta|} \,\Big|\, \mu \in \sigma(\mathbb{A}), \mu \neq \lambda \right\} \qquad \textit{and} \qquad \sigma_k \equiv \frac{\|r^M_k\|}{\|r^M_{k-1}\|}.$$

The theorem is of interest if, in the k–th step, some Ritz value θ approximates some eigenvalue λ: in this case $F_k \approx 1$. Although the theorem gives upper bounds only, we may interpret its claim as follows. If, in addition, there is some residual reduction in the k-th step (that is $\sigma_k < 1 - \delta$) then, from this step on, the eigenvalue λ does not seem to effect the convergence of the residuals anymore (cf. (26)).

Here, for clarity of presentation, we have formulated the theorem for one eigenvalue λ; however, the theorem can also be formulated for a set of eigenvalues (see [45]).

The effect of the approximation of the eigenvalues by Ritz values on the convergence depends on the distribution of the eigenvalues of $\mathbb{A}$.

Example 1 (continued 2) Suppose $\sigma(\mathbb{A}) = \{\lambda_1, \ldots, \lambda_n\}$ with $\lambda_1 < \lambda_2 < \ldots < \lambda_n$. Initially the residual reduction depends on $\mathcal{C} = \frac{\lambda_n}{\lambda_1}$, while after a while it depends on $\frac{\lambda_n}{\lambda_2}$.

Approximation of the smallest eigenvalue λ_1 by some Ritz value yields an acceleration of the convergence. An approximation of the largest eigenvalue λ_n by some Ritz value has the same effect, that is, it also causes an acceleration. However, it depends on the distribution of the eigenvalues whether this acceleration is of interest. For instance, with $\lambda_2 = \mu_1 \lambda_1$ and $\lambda_{n-1} = \mu_n \lambda_n$, "stripping" λ_1 reduces $\mathcal{C}$ by a multiplicative factor $\frac{1}{\mu_1}$ while "stripping" λ_n reduces $\mathcal{C}$ by a factor μ_n. In case of the discretized Laplace operator, and ILU preconditioning , $\mu_1 \approx 2$ and $0 \leq 1 - \mu_n = \mathcal{O}(h^2)$. If we precondition the discretized Laplace operator by MILU (cf. [42]) then $\mu_1 - 1 = \mathcal{O}(h^2)$ and $\mu_n \approx \frac{1}{2}$.

By the power method argument, we expect fast convergence of Ritz values especially to the extremal eigenvalues. For the general situation, there are no theorems to support this expectation. For the symmetric case, the Kaniel-Paige-Saad estimates tell when extremal eigenvalues are approximated by Ritz values (the following estimates are somewhat sharper than the classical ones; see [35]). However, the conclusions seem also to hold for more general situations, especially if $\mathcal{C}_E(\mathbb{A})$ is modest.

Theorem 5 *Assume $\mathbb{A}$ is symmetric. Let $\lambda_1,\ldots,\lambda_n$ be the eigenvalues of $\mathbb{A}$ numbered in increasing order with eigenvectors $v_1, \ldots, v_n$. Let τ be the tangent of the angle between the initial residual r_0 and the eigenvector v_1. Let $\beta := \frac{\lambda_2-\lambda_1}{\lambda_n-\lambda_1}$ be the* **gap ratio** *between λ_1 and λ_2. Consider a $k \in \mathbb{N}$. Let θ_1 be the smallest Ritz value of order k. Then*[3]

$$0 \leq \frac{\theta_1 - \lambda_1}{\lambda_n - \lambda_1} \leq \begin{cases} \tau^2\beta^2 e^2 \exp(-[4k+2]\sqrt{\beta}) & \text{if} \quad 2k-1 \geq \frac{1}{\sqrt{\beta}} \\ \frac{1}{(4k-2)^2}\min(\log(4\tau^2+2), 4\tau). \end{cases} \tag{28}$$

The theorem can be generalized for more than one eigenvalue.

From this theorem, we learn that if the extremal eigenvalues, that are small in absolute value, are separated relatively well (in the sense of gap-ratios), the acceleration in the convergence will occur early (the same holds with respect to the large extremal eigenvalues).

In summary we conclude: if the extremal eigenvalues close to zero are well separated then

(i) we expect large accelerations in the convergence and

(ii) the accelerations will occur in an early stage of the process.

Except for a factor like $\mathcal{C}_E(\mathbb{A})$, the size of the errors, residuals and the speed of convergence are independent of the basis that has been chosen to represent the problem. If $\mathbb{A}$ is diagonalizable, these quantities only depend on:

(i) the distribution of the eigenvalues of $\mathbb{A}$;

(ii) the sizes $|\mu_1|^2, \ldots, |\mu_n|^2$, where $r_0 = \sum_{j=1}^n \mu_j v_j$ and $v_1, \ldots, v_n$ are the eigenvectors of $\mathbb{A}$;

(iii) $\mathcal{C}_E(\mathbb{A})$.

The distribution of the eigenvalues determines the convergence behavior: speed and acceleration strongly depend on it. The sizes $|\mu_j|$ of the eigenvector components of the initial residual do not seem to be of much influence: moderate variations on these sizes seem to yield only minor changes in convergence behavior without essentially changing its character (see [36] and [35]). These observations are only correct in case $\mathcal{C}_E(\mathbb{A})$ is not too large. If $\mathcal{C}_E(\mathbb{A})$ is large anything may happen (example 6 can be viewed as an extreme illustration for this observation; see also [18]).

4.2 Algorithms

The standard approach for the construction of the r_k with the orthogonality or minimality property is to construct an orthogonal basis for the Krylov subspace $\mathcal{K}_k(\mathbb{A};\tilde{c}_0)$ where either $\tilde{c}_0 = r_0$ as in GMRES or $\tilde{c} = \mathbb{A}r_0$ as in GCR. This can be done in various ways, one more stable than another, but the most often followed approach is to construct this basis by the Modified Gram-Schmidt method. We discuss this method first.

4.2.1 Modified Gram-Schmidt

Let $\tilde{v}_0, \tilde{v}_1, \ldots$ be a sequence of independent vectors. For each k, let $\mathbf{V}_k$ be the space spanned by $\tilde{v}_0, \ldots, \tilde{v}_{k-1}$. For efficient and stable projections on the $\mathbf{V}_k$ it is useful to construct an orthogonal sequence of vectors $v_0, v_1, \ldots$ such that $v_0, \ldots, v_{k-1}$ form a basis of $\mathbf{V}_k$ for any k.

We will apply this construction in the situation where $\tilde{v}_0, \tilde{v}_1, \tilde{v}_2, \ldots$ is a basis of the Krylov space of $\mathbb{A}$: then $\mathbf{V}_k = \mathcal{K}_k(\mathbb{A};\tilde{v}_0)$ for any k.

[3] If θ_1 is the approximation of λ_1 in the k-th step of the power method with optimal shift then the scaled error can be bounded by $\tau^2\exp(-[4k+2]\beta)$. Here, the Krylov subspace approach also improves the convergence speed by a square root: recall that in (19) the $\sqrt{\mathcal{C}}$ is involved while in (10) $\mathcal{C}$ plays a role.

Orthogonalization

Take $v_0 = \tilde{v}_0$. Suppose $v_0, \ldots, v_{k-1}$ have the desired properties.
To explain how to construct v_k, consider the map P_k given by

$$P_k \equiv \sum_{j=0}^{k-1} \frac{v_j v_j^T}{v_j^T v_j}. \tag{29}$$

Then P_k is the orthogonal projection on $\mathbf{V}_k$ and $(I - P_k)y \perp \mathbf{V}_k$ for any vector y. Now, let

$$v_k \equiv \tilde{v}_k - P_k \tilde{v}_k = \tilde{v}_k - \sum_{j=0}^{k-1} \tilde{\beta}_{jk} v_j \quad \text{with} \quad \tilde{\beta}_{jk} \equiv \frac{(\tilde{v}_k, v_j)}{(v_j, v_j)}. \tag{30}$$

Then v_k is perpendicular to $v_0, \ldots, v_{k-1}$ and, clearly, $v_0, \ldots, v_k$ span $\mathbf{V}_{k+1}$: v_k is our next orthogonal vector. Here, we orthogonalize by the well-known Gram-Schmidt process.

Note that

$$I - P_k = I - \sum_{j=0}^{k-1} \frac{v_j v_j^T}{v_j^T v_j} = (I - \frac{v_{k-1} v_{k-1}^T}{v_{k-1}^T v_{k-1}}) \cdot \ldots \cdot (I - \frac{v_0 v_0^T}{v_0^T v_0}). \tag{31}$$

Hence, the orthogonalization can be performed recursively. The stability of this way of computing an orthogonal system of vectors is satisfactory: evaluation errors arising in the course of the computation are also projected and subtracted and thus suppressed. So, v_k can be computed by the **mod.G-S** (**modified Gram-Schmidt**) process:

$$v_k^{(0)} = \tilde{v}_k, \quad v_k^{(j+1)} = v_k^{(j)} - \tilde{\beta}_{jk} v_j \ \ (j = 0, \ldots, k-1), \quad v_k = v_k^{(k)}, \tag{32}$$

where, in exact arithmetic, the $\tilde{\beta}_{jk} = (v_k^{(j)}, v_j)/(v_j, v_j)$ coincide with the ones in (30).

Orthonormalization

Obviously, one can easily produce a sequence of ortho*normal* vectors as well.
If, in addition, $v_0, \ldots, v_{k-1}$ are normalized ($\|v_j\| = 1$) then $\hat{v}_k = (I - P_k)\tilde{v}_k$ and $v_k = \frac{1}{\beta_{kk}} \hat{v}_k$ with $\beta_{kk} = \|\hat{v}_k\|$ yields the next orthonormal vector v_k. The orthogonalization can be performed in the modified way (32). Now

$$\beta_{kk} v_k = \tilde{v}_k - \sum \beta_{jk} v_j \quad \text{with} \quad \beta_{jk} = (\tilde{v}_k, v_j). \tag{33}$$

Note that these β_{jk} are precisely the $\tilde{\beta}_{jk} \|v_j\|$ of the non-normalized orthogonalization process.

Hessenberg matrices

We finally make some observations that will be useful:
(a) for computing Ritz-values (to approximate eigenvalues of $\mathbb{A}$) and
(b) to explain the GMRES algorithm.

Let $v_0, v_1, \ldots$ be the orthonormalized sequence of vectors. Let B be the **Hessenberg** matrix

$$B = \begin{bmatrix} \beta_{01} & \beta_{02} & \beta_{03} & \cdots \\ \beta_{11} & \beta_{12} & \beta_{13} & \cdots \\ 0 & \beta_{22} & \beta_{23} & \cdots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{34}$$

Now, suppose $\tilde{v}_j = \mathbb{A} v_{j-1}$ for all j. Then B is the matrix of $\mathbb{A}$ with respect to the basis $v_0, v_1, \ldots$.

Sub (a) Clearly, in this situation, $\mathbf{V}_k \subset \mathcal{K}_k(\mathbb{A}; v_0)$. One can even show that these spaces coincide (assuming that $\mathcal{K}_k(\mathbb{A}; v_0)$ is of dimension k). The upper left $k \times k$-block B_k of B is the matrix of the projection $P_k \mathbb{A}_{|\mathbf{V}_k}$ of $\mathbb{A}$ on $\mathbf{V_k}$ with respect to the orthonormal basis $v_0, \ldots, v_{k-1}$. The eigenvalues of B_k are the k-th order Ritz values of $\mathbb{A}$ (with respect to the Krylov space $\mathcal{K}_k(\mathbb{A}; v_0)$). In practice, we are interested in the case where $k \ll n$. Therefore, and since B_k is Hessenberg, we can easily compute the eigenvalues of B_k, for instance by means of the QR-algorithm (see [15]). This approach to compute approximations of eigenvalues of $\mathbb{A}$ is known as **Arnoldi's** algorithm.

Sub (b) If V_k is the $n \times k$-matrix with $v_0, \ldots, v_{k-1}$ in its consecutive columns and $\underline{B}_k$ is the upper left $(k+1) \times k$-block of B then $\mathbb{A} V_k = V_{k+1} \underline{B}_k$.

4.2.2 Generalized conjugate residuals and related algorithms

Suppose $\tilde{c}_0, \tilde{c}_1, \ldots$ is a basis of the Krylov space of $\mathbb{A}$ generated by $\tilde{c}_0 = \mathbb{A} r_0$. We may apply the orthogonalization construction of the previous subsection (taking $\tilde{v}_j = \tilde{c}_j$). Let $c_0, c_1, \ldots$ be the resulting sequence of orthogonal vectors. Then $P_k r_0$ is the best approximation of r_0 in $\mathcal{K}_k(\mathbb{A}; \mathbb{A} r_0) = \mathcal{K}_k(\mathbb{A}; \tilde{c}_0)$ and $r_k = (I - P_k) r_0$ is the k-th residual in the MR approach (cf. (24)). The correction $P_k r_0$ can easily be computed by inner products and vector updates.

In order to update the approximation, the corrections of the residuals should be of the form Au where we have to know u, the update of the approximation. Suppose we have chosen the updates $\tilde{u}_j$, $\tilde{u}_0 = K^{-1} r_0$, and the $\tilde{c}_j$, with $\tilde{c}_j \equiv A \tilde{u}_j$, form a basis of the Krylov space generated by $\mathbb{A}$. Then the u_j for which $c_j = A u_j$ can easily be computed: because, if

$$c_k = \tilde{c}_k - \sum_{j=0}^{k-1} \tau_{jk} c_j, \quad u_k \equiv \tilde{u}_k - \sum_{j=0}^{k-1} \tau_{jk} u_j \quad \text{with} \quad \tau_{jk} \equiv (\tilde{c}_k, c_j)/(c_j, c_j). \tag{35}$$

and if $A u_j = c_j$ for $j < k$ then obviously $A u_k = c_k$. The computation of c_k and of u_k involves the same inner products. The c_k and the τ_{jk} can be computed by mod.G-S (see (32)). Since (in exact arithmetic) the "modified" τ_{ij} and the original ones coincide, the u_k can also be computed by an analog of (32).

For ease of reference, we recall the definition of P_k and we introduce the map Q_k.

$$P_k \equiv \sum_{j=0}^{k-1} \frac{c_j c_j^T}{c_j^T c_j} \quad \text{and} \quad Q_k \equiv \sum_{j=0}^{k-1} \frac{u_j c_j^T}{c_j^T c_j}. \tag{36}$$

Note that

$$P_k = A Q_k, \quad c_k = \tilde{c}_k - P_k \tilde{c}_k \quad \text{and} \quad u_k = \tilde{u}_k - Q_k \tilde{c}_k. \tag{37}$$

Now, $x_k = x_0 + Q_k r_0$. By (31), we actually have that

$$x_{k+1} = x_k + \alpha_k u_k \quad \text{and} \quad r_{k+1} = r_k - \alpha_k c_k \quad \text{with} \quad \alpha_k = (r_k, c_k)/(c_k, c_k). \tag{38}$$

Remains to explain how to choose the $\tilde{u}_k$ such that the $\tilde{c}_k \equiv A \tilde{u}_k$ form a basis of the Krylov space of $\mathbb{A}$ generated by $\mathbb{A} r_0$. For any k, we have to choose a $\tilde{u}_k$ in $K^{-1} \mathcal{K}_{k+1}(\mathbb{A}; r_0)$ that does not belong to $K^{-1} \mathcal{K}_k(\mathbb{A}; r_0)$. In literature, one can find two choices for $\tilde{u}_k$: GCR and ORTHODIR.

Generalized conjugate residuals, ORTHOMIN

Recall that $r_k = r_0 + \zeta_1 \mathbb{A} r_0 + \ldots + \zeta_k \mathbb{A}^k r_0$ for some real ζ_i. We may choose $\tilde{u}_k = K^{-1} r_k$: if $\zeta_k \neq 0$ then $\tilde{u}_k$ has the desired property. This choice and the equations (37) (or (35)) and (38) lead to **GCR** (Generalized Conjugate Residuals) [8, 9] or **ORTHOMIN** [41] given in ALGORITHM 2.

In order to facilitate the presentation of (i) a cheap variant of GCR, and (ii) a way to compute the spectrum of $\mathbb{A}$ (see 4.4), we represent the relations in GCR in terms of matrices.

```
choose x_0,
k = -1,   r_0 = b - A x_0,
repeat until ||r_{k+1}|| is small enough:
    { k := k + 1,
    { solve ũ from K ũ = r_k,  c̃ := A ũ,
    { compute by mod.G-S u_k = ũ - Q_k c̃,  c_k = c̃ - P_k c̃,
    { γ_k = (c_k, c_k),  α_k = (r_k, c_k)/γ_k,
    { x_{k+1} = x_k + α_k u_k,  r_{k+1} = r_k - α_k c_k
```

ALGORITHM 2: The GCR algorithm.

A matrix representation of GCR We define the $n \times k$ matrix R_k as the matrix with columns $r_0, r_1, \ldots, r_{k-1}$ and the $n \times k$ matrix C_k as the matrix with the orthonormal vectors $\frac{1}{\sqrt{\gamma_0}}c_0, \ldots, \frac{1}{\sqrt{\gamma_{k-1}}}c_{k-1}$ as its columns. Now, the GCR relation $\tilde{c} = A\tilde{u} = \mathbb{A}r_k = c_k + P_k\tilde{c}$ leads to $\mathbb{A}R_k = C_k T_k$, where T_k is the $k\times k$ upper-triangular matrix with the scaled projection-coefficients $\tau_{ij}\sqrt{\gamma_i}$ (cf. (35) and ALGORITHM 2) in the strict upper-triangle and $\sqrt{\gamma_i}$ on its diagonal. Since the columns of C_k form an orthonormal system, we have that $C_k^T C_k$ is the $k \times k$ identity matrix.

Let D_k the $k \times k$ diagonal matrix with diagonal coefficients $\alpha_0\sqrt{\gamma_0}, \ldots, \alpha_{k-1}\sqrt{\gamma_{k-1}}$. Since $r_{k+1} = r_k - \alpha_k c_k$ we have that $R_{k+1}S_k = C_k D_k$, where S_k is the $(k+1) \times k$ matrix given by $S_k e_j = e_j - e_{j+1} \quad (j = 1, \ldots, k)$.

Summarizing,

$$\mathbb{A}R_k = C_k T_k \quad \text{with} \quad C_k^T C_k = I \quad \text{and} \quad R_{k+1}S_k = C_k D_k. \tag{39}$$

Break down GCR may **break down** before we have an accurate approximation. This happens if $\alpha_k = 0$ for some $k \le n$. Then $r_{k+1} \in \mathcal{K}_k(\mathbb{A}; r_0)$ and $c_{k+1} = 0$. Consequently, the c_j's are not independent (now, also $\gamma_{k+1} = 0$ and division by γ_{k+1} is impossible).

Example 4 Consider the equation with $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $K = I$ and $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.
The choice $x_0 = 0$ yields $r_0 = b$ and $\alpha_0 = (r_0, Ar_0)/(c_0, c_0) = 0$.

The ORTHODIR algorithm in the next subsection avoids this breakdown. One can prove that this breakdown does not occur if, for instance, $A^T + A$ is positive definite, or, equivalently, if $\mathrm{Re}(Ay, y) > 0$ for all y.

Costs We do not need to store the α_k. Moreover, x_{k-1} and r_{k-1} are only needed in the $(k-1)$-th step and can then be overwritten by x_k and r_k, respectively (we have labeled the x_k, r_k and α_k with an index k, for ease of discussion only). The computational costs in the $(k-1)$-th step and the memory requirement between the $(k-1)$-th and k-th step can be summarized by:

$$\begin{array}{ll} \text{computational effort:} & 1 \text{ Mv}, \quad k \text{ dot}, \quad 2k \text{ axpy} \\ \text{memory requirement:} & k \times \text{“}c_j\text{”}, \quad k \times \text{“}u_j\text{”}, \quad x_k, \quad r_k \end{array} \tag{40}$$

where 1 **Mv** is the time needed to solve an equation $K\tilde{u} = r$ and to evaluate the matrix-vector multiplication $A\tilde{u}$, 1 **dot** the time needed for one inner-product and 1 **axpy** for one vector update (of the form $\alpha x + y$). As an alternative, one can compute c_k by evaluating Au_k. This requires an additional matrix-vector multiplication, while evaluating $\tilde{c} - P_k\tilde{c}$ requires k axpy. For expensive Mv the alternative may be unattractive.

A cheap variant of GCR We will show that there is no need to compute the u_j nor to update x_j in each step. The computation of the approximate solution x_k can be postponed until the norm of the residual is small enough. As will turn out, as far as it concerns large n-vectors, this saves on memory ($\approx 50\%$) and on computational costs ($\approx 33\%$): in the k-th step we only have to store x_0, r_0, r_k and $k\times$"c_j" and instead of $2k$ axpy, we only need k axpy.

We employ the matrix representation of GCR in (39).
If GCR does not break down then $\mathcal{K}_k(\mathbb{A}; r_0) = \{R_k\vec{\gamma}\,\big|\,\vec{\gamma}\in\mathbb{R}^k\}$ and T_k is non-singular.
Therefore, if $\vec{\gamma} = (\gamma_0, \ldots, \gamma_{k-1})^T$ solves the minimization problem

$$\|r_k\| = \min\left\{\|r_0 - \mathbb{A}u\|\,\Big|\,u\in\mathcal{K}_k(\mathbb{A}; r_0)\right\} = \min_{\vec{\gamma}\in\mathbb{R}^k}\|r_0 - \mathbb{A}R_k\vec{\gamma}\| \tag{41}$$

then $x_k = x_0 + K^{-1}R_k\vec{\gamma}$. Since $\mathbb{A}R_k = C_kT_k$ and $C_k^TC_k = I$ (cf. (39)), the solution of this least-square problem (41) is given by $\vec{\gamma} = T_k^{-1}C_k^Tr_0$. In practice, we are interested in the case where $k \ll n$. Hence, since T_k is upper-triangular, $\vec{\gamma}$ can be computed cheaply from r_0 and $c_0, \ldots, c_{k-1}$. Since we wish to save on memory, we don't want to store all the r_j's. Fortunately, the r_j, needed in the computation of x_k from $R_k\vec{\gamma}$ can easily be reconstructed from r_0, the c_j's and the α_j's, because: $r_j = r_{j-1} - \alpha_{j-1}c_{j-1} = \ldots = r_0 - \alpha_0c_0 - \ldots - \alpha_{j-1}c_{j-1}$. Therefore, if W_k is the $k\times k$ upper-triangular matrix of which all upper-triangular entries are 1 we have that

$$\begin{aligned}&x_k = x_0 + y_k \text{ where } y_k \text{ is the solution of}\\ &Ky_k = (\gamma_0 + \ldots + \gamma_{k-1})r_0 - C_{k-1}D_{k-1}W_{k-1}(\gamma_1, \ldots, \gamma_{k-1})^T.\end{aligned} \tag{42}$$

This variant of GCR is comparable to the well-known GMRES algorithm [32] to be discussed in 4.2.3: both are algorithmical representations of the MR approach requiring the same amount of memory and the same number of large vector operations in each step. The algorithms are based on the same principle: they both produce only *one* sequence of orthogonal vectors (the sequence of the GCR variant spans $\mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)$, while the one of GMRES spans $\mathcal{K}_k(\mathbb{A}; r_0)$) and they do not assemble x_k as long as the norm of the residual is not small enough. In contrast to GCR and its variant, GMRES does not break down. However, one can easily formulate a cheap variant of ORTHODIR [6] that is as robust and as cheap as GMRES. As a side product, GCR computes all approximations x_k. If one wishes to track the progress of the computations by monitoring the approximations one can not employ the cheap variant of GCR (nor GMRES).

Modifications The obvious disadvantage of GCR as well as the other methods like GMRES, FOM, ORTHODIR, etc., is that the amount of storage grows (all c_j's have to be stored), and the work per iteration step grows also (all the orthogonalization work increases per step). Therefore, the price for the optimality of GCR (it delivers the minimum residual over all Krylov subspace methods) may be too high in practical situations. An obvious way out is to restart after each m iteration steps. The method is then referred to as GCR(m). Restarting is the most popular strategy in practice, but, unfortunately, the convergence behavior may depend strongly on the choice of m. An unlucky choice may result in slow convergence or even in no convergence at all: by a new start the information that accelerates the speed of convergence (see 4.1) is lost and has to be collected again. In contrast to GMRES (or to the cheap variant of GCR), GCR allows to follow other strategies, for instance to maintain only the last m vectors u_j (and c_j; truncation) or some appropriate linear combination of u_j's (assemblage) for the search space.

(a) Restart. Select a $m \in \mathbb{N}$. Compute x_m and r_m with GCR. If $\|r_m\|$ is not small enough apply the GCR algorithm again, now choosing x_m as a new start. Repeat this iteration until the residual is small enough.

(b) Truncation. Fix some $m \in \mathbb{N}$. Replace in the algorithm, P_k and Q_k by $\tilde{P}_k$ and $\tilde{Q}_k$ respectively, where one obtains the tilded maps by restricting to the summation of the last m j's in (36): let j run from $k - m$ to $k - 1$.

(c) Dynamic assemblage. Instead of working with the complete Krylov space $\mathcal{K}_k(\mathbb{A};\mathbb{A}r_0)$, for $j < k$, we wish to select j independent vectors $\mathbf{c}_0,\ldots,\mathbf{c}_{j-1}$ in this Krylov space. These vectors should span a subspace on which the problem $Ax = b$ can be represented (almost) as well as on the complete Krylov space. Although the new $\mathbf{c}_i$'s will be constructed in a GCR way, they need not to coincide with the c_i's of the GCR algorithm.

Suppose that in the k_{j-1}-th step, $\mathbf{c}_0,\ldots,\mathbf{c}_{j-1}$ have been selected in $\mathcal{K}_k(\mathbb{A};\mathbb{A}r_0)$ where $k = k_{j-1}$. Now, construct, c_{j+i}, α_i and r_{k+i} for $i = 0, 1, \ldots$ as in the GCR algorithm with P_{j+i} as in (36) using $\mathbf{c}_0,\ldots,\mathbf{c}_{j-1}$, $c_j,\ldots,c_{j+i}$ instead of $c_0,\ldots,c_{j+i}$. If the reduction from r_k to r_{k+l} is sufficiently small then we construct a new $\mathbf{c}_j$ as a combination of $c_j,\ldots,c_{j+l}$. Take $k_j = k + l$ and $\mathbf{c}_j = \sum_{i=0}^{l} \alpha_i c_{j+i}$: $\mathbf{c}_j$ is the projection of r_k (and of r_0) onto the space spanned by the $c_j,\ldots,c_{j+l}$. Next, construct new $c_{j+1},\ldots,c_{j+l'}$ (overwriting the old ones), find a new $\mathbf{c}_{j+1}$, etc.. Repeat this approach until the norm of the residual is small enough. Parallel to c_{j+i}, r_{k+i} and $\mathbf{c}_j$, compute u_{j+i}, x_{k+i} and $\mathbf{u}_j$.

We give the algorithm to compute the residuals, we leave the details in the computation of x_k to the reader.

Choose some $\delta \in (0, 1)$. The k-th step to compute the residuals runs as follows (with P_{j+l} as indicated above)

$$\begin{array}{l} k := k+1,\ \texttt{if}\ \|r_k\| \geq \delta\rho\ \texttt{then}\ i := i+1 \\ \qquad\qquad\qquad\quad \texttt{else}\ \mathbf{c}_j := \sum_{p=0}^{i} \alpha_{j+p} c_{j+p},\ i := 0,\ j := j+1,\ \rho := \|r_k\| \\ c_{j+i} := (I - P_{j+i})\mathbb{A}r_k, \quad \alpha_{j+i} := (r_k, c_{j+i})/(c_{j+i}, c_{j+i}), \qquad r_{k+1} := r_k - \alpha_{j+i}c_{j+i} \end{array}$$

(d) Static assemblage. This modification runs along the same lines as the dynamic assemblage one: here, we form repeatedly a new $\mathbf{c}_j$ from the $c_j,\ldots,c_{j+l-1}$ after a fixed number of l steps, while in (c), we formed a new $\mathbf{c}_j$ if the assembled $\mathbf{c}_j$ reduced the residual sufficiently. Again, we give the algorithm — the l-static assemblage GCR algorithm — only for the residuals.

Choose some $l \in \mathbb{N}$. The k-th step to compute the residuals runs as follows (with P_{j+l} as indicated in (c))

$$\begin{array}{l} k := k+1,\ \texttt{if}\ i < l-1\ \texttt{then}\ i := i+1 \\ \qquad\qquad\qquad\quad \texttt{else}\ \mathbf{c}_j := \sum_{p=0}^{l-1} \alpha_{j+p} c_{j+p},\ i := 0,\ j := j+1 \\ c_{j+i} := (I - P_{j+i})\mathbb{A}r_k, \quad \alpha := (r_k, c_{j+i})/(c_{j+i}, c_{j+i}), \quad r_{k+1} := r_k - \alpha_{j+i}c_{j+i} \end{array}$$

Discussion Note that neither the c_j's nor the $\mathbf{c}_j$'s computed in one of the ways described above coincide with the "classic" GCR c_j's. In (b), (c) and (d) they form an orthogonal system with computable originals. In (a), the c_j's ever produced are even not orthogonal — which explains why this adaptation is not as attractive (efficient) as the other ones. But in neither of these adaptations they form a *basis* of $\mathbb{A}\mathcal{K}_k(\mathbb{A}; r_0)$.

Any of the modifications (b) – (d) can be restarted after a number of iteration steps: for $m, l \in \mathbb{N}$, GCR(m, l) denotes the following static assemblage variant. As in (d), after any l "classic GCR" steps the last l c's are assembled to form a new $\mathbf{c}$, after ml steps of this l-static assemblage GCR, the algorithm starts over taking the obtained approximation as a new starting vector. In average, the steps of the GCR(m, l) algorithm are as expensive as the steps of the GCR$(m + l)$ algorithm. However, as experiments show, in practice, the GCR(m, l) often converges faster than GCR$(m+l)$. This faster convergence may be understood as follows. Until the ml-th step of GCR(m, l), the algorithm uses information from all the previous steps: it uses the $\mathbf{c}$'s and c's constructed so far. These $\mathbf{c}$'s and c's span a space that almost contain r_0 (hopefully): the $\mathbf{c}$'s were constructed to do so. While, in any $j(m + l) + 1$-th step, the GCR$(m + l)$ algorithm starts with the construction of a new approximating space. However, there is no mechanism to avoid that the new space is close to the old one: by neglecting the old space, the algorithm may start about the same construction.

We introduced two modification of GCR based on assemblage. Obviously, one can think of many others. Research, to find good assemblage strategies, is still in progress and, as observed before, what is best depends on the problem and on the computer. Some of the experience is reported in [6].

Finally, note that GCR(1) is precisely the LMR algorithm. So, any GCR(m) can be viewed as an improvement of the LMR algorithm.

ORTHODIR

If the c_k are as computed by GCR then

$$(I-P_k)\mathbb{A}r_{k-1} = (I - \frac{c_{k-1}c_{k-1}^T}{c_{k-1}^T c_{k-1}})(I-P_{k-1})\mathbb{A}r_{k-1} = (I - \frac{c_{k-1}^T c_{k-1}}{c_{k-1}^T c_{k-1}})c_{k-1} = 0.$$

Hence,

$$c_k = (I-P_k)\mathbb{A}(r_{k-1} - \alpha_{k-1}c_{k-1}) = -\alpha_{k-1}(I-P_k)\mathbb{A}c_{k-1}.$$

Therefore, we may as well choose $K^{-1}c_{k-1}$ for $\tilde{u}_k$. Then $c_k = (I-P_k)\mathbb{A}c_{k-1}$: except for the multiplicative scalar factor $-\alpha_{k-1}$, this c_k coincides with the GCR one. The resulting algorithm **ORTHODIR** [47] is the GCR algorithm in which "`solve` $\tilde{u}$ `from` $K\tilde{u} := r_k$" is replaced by "`solve` $\tilde{u}$ `from` $K\tilde{u} := c_{k-1}$".

GCR breaks down if $\alpha_{k-1} = 0$: then, in GCR, c_k=0 and the $c_0,\ldots,c_k$ do not span $\mathcal{K}_{k+1}(\mathbb{A};\mathbb{A}r_0)$. ORTHODIR does not use α_{k-1} in the computation of its c_k and, thus, avoids this breakdown. One can even show that this algorithm does *not* break down at all (see observation (a) on Hessenberg matrices in subsection 4.2.1 and note that $\tilde{c}_k = \mathbb{A}c_{k-1}$). If $\alpha_{k-1} = 0$ then in both algorithms $r_k = r_{k-1}$: they both stagnate in the k-th step.

Obviously, the conclusions for GCR concerning the computational efforts, the memory requirements and the possibilities to derive truncated and assembled forms, as well as to restart, also hold for ORTHODIR. Although ORTHODIR is (more) robust, we have discussed GCR in more detail, since it is easier to understand why modifications on these algorithms (that save memory and speed up the computation) work so well, by viewing them as modifications of GCR (see section 6).

4.2.3 Generalized minimal residuals

In contrast to the GCR algorithm, the **GMRES** (**Generalized Minimal RESiduals**) algorithm [32], to be described below, is based on the construction of an orthonormal basis $v_0, v_1, \ldots$ for the Krylov space of $\mathbb{A}$ generated by r_0 and not on one generated by $\mathbb{A}r_0$. The minimization problem (M) is reformulated with respect to this orthonormal basis and subsequently solved.

Let $v_0, v_1, \ldots$ be a sequence of orthonormal vectors produced by mod.G-S as in subsection 4.2.1 with $\tilde{v}_0 = r_0$ and $\tilde{v}_j = \mathbb{A}v_{j-1}$ for all j. Using the notations as in our observations on Hessenberg matrices in 4.2.1, we have that $\mathbb{A}V_k = V_{k+1}\underline{B}_k$. Since the v_j form an orthonormal system the matrix V_{k+1} preserves distances: $\|V_{k+1}\vec{\alpha}\| = \|\vec{\alpha}\|$ $(\vec{\alpha} \in \mathbb{R}^{k+1})$. We write $\rho_j = \|r_j\|$ and denote the first canonical basis vector in $\mathbb{R}^{k+1}$ by e_1. Now, (23) can be reformulated as follows.

$$\begin{aligned} \|r_k\| &= \min\left\{\|r_0 - \mathbb{A}u\| \,\middle|\, u \in \mathcal{K}_k(\mathbb{A};r_0)\right\} = \min_{\vec{\gamma}\in\mathbb{R}^k}\|r_0 - \mathbb{A}V_k\vec{\gamma}\| \\ &= \min_{\vec{\gamma}}\|\rho_0 V_{k+1}e_1 - V_{k+1}\underline{B}_k\vec{\gamma}\| = \min_{\vec{\gamma}}\|\rho_0 e_1 - \underline{B}_k\vec{\gamma}\| \end{aligned} \tag{43}$$

In practice, we are interested in $k \ll n$. This "small" minimization problem (43) can easily be solved by a QR-decomposition. Since $\underline{B}_k$ is a Hessenberg matrix, the decomposition in the k–th step can be used to perform the decomposition in the $k+1$-th step efficiently by recursively applying Givens rotations. Note that the solution of (43) also yields ρ_k without computing r_k explicitly. The Givens rotations even allow a computation of the ρ_k without computing the $\vec{\gamma}$ explicitly. Therefore, in this way, we do not need to assemble the residuals at all and we only have to compute the approximated solution x_m if ρ_m is small enough: there is no need to compute x_k (nor $\vec{\gamma}$) in an earlier stage. If $\vec{\gamma}^G = (\gamma_0^G, \ldots, \gamma_{m-1}^G)^T$ is the solution of (43) in $\mathbb{R}^m$ (with $k = m$ now) for which ρ_m is sufficiently small then

$$x_k = x_0 + y_k \quad \text{where } y_k \text{ is the solution of} \quad K y_k = \sum_{j=0}^{m-1} \gamma_j^G v_j. \tag{44}$$

GMRES The GMRES algorithm computes recursively $\tilde{v}_k = \mathbb{A} v_{k-1}$; by mod.G-S the next orthonormal vector v_k; the last column of the Hessenberg matrices $\underline{B}_k$; and ρ_k, with Givens rotations. It stops the iteration for $k = m$ if ρ_m is small enough. Then the solution $\vec{\gamma}^G$ of the minimization problem $\min_{\vec{\gamma}} \|\rho_0 e_1 - \underline{B}_k \vec{\gamma}\|$ is computed with Givens rotations and the actual approximation x_m is assembled (for details, see [32]).

Costs The algorithm is efficient since it computes only the final approximation x_m and it does not compute any (intermediate) r_k. But what is more important, it only computes and stores one sequence of vectors, the v_j's, while in GCR-type of methods the two sequences of u_j's and c_j's are required. The computational costs in the $(k-1)$-th step and the memory requirements from the $(k-1)$-th to the k-th step, in terms of the large n-vector operations, can be summarized by:

$$\begin{array}{ll} \text{computational effort:} & 1 \text{ Mv}, \quad k \text{ dot}, \quad k \text{ axpy} \\ \text{memory requirement:} & x_0, \quad k \times \text{“}v_j\text{”} \end{array} \tag{45}$$

Further, in any step one k-vector operation and a few scalar operations are performed (the application of all previous Givens rotations to the new k-vector of β_{jk} and the construction and application of the next Givens rotation). Each Givens rotation can be characterized by 1 or 2 reals. The reals for the k Givens rotations, a $k \times k$ upper triangular matrix and a k-vector ($\rho_0 e_1$ transformed by the Givens rotations) have to be stored.

Discussion If large vector operations dominate the costs of the algorithm, the GMRES algorithm is cheaper than GCR by a factor 0.66 (assuming that 1 dot is about as expensive as one axpy). Since, as a side product, GCR also computes all approximations x_k, GCR is useful if one wishes to track the progress of the computations by monitoring the approximations. GMRES does not suffer from breakdown. As in GCR, the costs for the GMRES algorithm grow per iteration step. In case GMRES becomes too expensive one has to modify the algorithm. GMRES(m) [32], restarted GMRES, is widely used in practice. By restarting, we lose the information on which acceleration of the convergence is based. Unfortunately, the GMRES algorithm does not allow a truncated or an assembled form.

4.2.4 Full orthogonalization Method

The **FOM** (**Full orthogonalization Method**) algorithm [30] represents the OR approach (cf. section 4). Like GMRES it computes orthonormal vectors v_j that span $\mathcal{K}_k(\mathbb{A}; r_0)$ recursively by mod.G-S and it computes the Hessenberg matrix $\underline{B}_k$. If B_k is the $k \times k$ upper block matrix of $\underline{B}_k$, then $B_k \vec{\gamma} = \rho_0 e_1$ is the matrix vector representation of (25) with respect to the basis $v_0, \ldots, v_{k-1}$ of $\mathcal{K}_k(\mathbb{A}; r_0)$. FOM solves $\vec{\gamma}^F = (\gamma_0^F, \ldots, \gamma_{k-1}^F)^T$ from $B_k \vec{\gamma} = \rho_0 e_1$ and obtains x_k by

$$x_k = x_0 + y_k \quad \text{where } y_k \text{ is the solution of} \quad K y_k = \sum_{j=0}^{m-1} \gamma_j^F v_j. \tag{46}$$

FOM breaks down in the k-th step if B_k is singular. Since GMRES solves least square problems (involving the matrix $\underline{B}_k$) it avoids the danger of breakdown.

```
choose x_0,
k = -1,   r_0 = b - A x_0,   u_{-1} = 0,
repeat until ||r_{k+1}|| is small enough:
  { k := k + 1,
    select a non-singular matrix H_k,
    solve ũ := H_k r_k,   compute c̃ := A ũ,
    α := (r_k, c̃)/(c̃, c̃),
    x_{k+1} = x_k + α ũ,   r_{k+1} = r_k - α c̃
```

ALGORITHM 3: The LMR algorithm with variable preconditioners.

4.3 Updating the preconditioner

LMR (cf. 2.2) may be adapted by using a different preconditioner in each step: see ALGORITHM 3.

Information obtained in the course of the computation may be used to improve the preconditioner. We will show that this happens *implicitly* in the MR approach.

Let $c_0, \ldots, c_{k-1}$ be some orthogonal system for which there are u_j with $c_j = Au_j$. Let P_k be the orthonormal projection on span $(c_0, \ldots, c_{k-1})$ and let Q_k be such that $P_k = AQ_k$ (as in (36)). Now, let

$$H_k = K^{-1} + Q_k(I - \mathbb{A}). \tag{47}$$

Then $I - AH_k = I - \mathbb{A} - P_k(I - \mathbb{A}) = (I - P_k)(I - \mathbb{A})$. Note that $I - P_k$ is the orthogonal projection on the space perpendicular to span $(c_0, \ldots, c_{k-1})$. Therefore, if $\mathbb{A}$ approximates I in some sense then AH_k is a better approximation of I and H_k is a better preconditioner than K^{-1}.

Note that $H_{j+1} = H_j + \frac{1}{c_j^T c_j} u_j\left((I - \mathbb{A})c_j\right)^T$: H_{j+1} is a correction of H_j by a rank-one matrix.

If the c_j and u_j are as in the GCR algorithm, then $r_k = (I - P_k)r_0$ and

$$\tilde{u} = H_k r_k = K^{-1} r_k - Q_k \mathbb{A} r_k + Q_k(I - P_k) r_0 = K^{-1} r_r - Q_k \mathbb{A} r_k = u_k : \tag{48}$$

GCR may be viewed as LMR in which the preconditioner is rank-one updated in each iteration step.

Apparently, GCR itself produces a preconditioner. Obviously, this preconditioner can be used if one has to solve some closely related equations as in the case of time-dependent PDE's or when using Newton's method. On the new time-levels or in the new Newton steps, one can improve these preconditioner by employing assembly-strategies, as explained in subsection 4.2.2.

4.4 Techniques to compute eigenvalues

GMRES and FOM produce orthonormal vectors that span $\mathcal{K}_k(\mathbb{A}, r_0)$ by mod.G-S. The coefficients used in the orthogonal projections constitute a Hessenberg matrix $\underline{B}_k$. Since vectors of the form $\tilde{v}_k = \mathbb{A} v_{k-1}$ were orthogonalized and normalized, the $k \times k$ upper block B_k of $\underline{B}_k$ is the matrix of $\mathbb{A}$ projected on that Krylov space with respect to an orthonormal basis (see the observation (a) on Hessenberg matrices in subsection 4.2.1). Since B_k is Hessenberg and $k \ll n$ in the case of interest, we can cheaply compute the eigenvalues of B_k. These eigenvalues are k-th order Ritz values of $\mathbb{A}$ (cf. 4.1). Hence, as a side product, these methods (as well as GCR, ORTHODIR and other methods) allow a cheap computation of Ritz values and, this provides information about the spectrum of $\mathbb{A}$.

This information is useful if one wishes to assess the effect of the preconditioner. The insight gained in this way may press for another preconditioner or may help to select the most appropriate algorithm. It also may help to construct a better preconditioner (see, for instance, [5]. In [20], a preconditioner is constructed by means of rank-one updates. The rank-one matrices in that paper are constructed from estimates of eigenvalues). Moreover, in a context of discretized PDE's, it also may give information on the stability of the discretization method.

In order to understand that Ritz values can also be computed cheaply as a side product of GCR, we consider the matrix representation of GCR in (39). We find that $\mathbb{A}C_k = \mathbb{A}R_{k+1}S_kD_k^{-1} = C_{k+1}\underline{B}_k$, where $\underline{B}_k = T_{k+1}S_kD_k^{-1}$ is $(k+1)\times k$ Hessenberg. Its $k\times k$ upper block B_k is Hessenberg as well and is the matrix of $\mathbb{A}$ projected on the Krylov subspace $\mathcal{K}_k(\mathbb{A};\mathbb{A}r_0)$ with respect to the orthonormal basis of normalized c_j. Apparently, also for GCR, the Hessenberg matrix B_k can easily be computed (from the projection coefficients and the scalars α_j and γ_j).

5 Conjugate Gradients for Normal Equations

Although in the context of advection-diffusion equations, A hardly ever is symmetric, we can easily find a preconditioner for which the preconditioned matrix $\mathbb{A}$ is symmetric: $K^{-1} = A^T$. The symmetry can be used to improve greatly the efficiency of the separate steps of CGR. The resulting algorithm CRNR and the related CGNR and LSQR [28] are robust: they do not break down. Although, the choice $K^{-1} = A^T$ leads to cheap iteration steps it may slow down the speed of convergence dramatically. For this reason, in spite of the cheap steps, these algorithms are not attractive. We include them in our discussion since they are useful to improve the robustness of other algorithms by mixing them with the others.

5.1 Conjugate residual and conjugate gradient

Consider GCR and assume for the moment that $\mathbb{A}$ is symmetric. Since

$$\begin{aligned} r_k \perp \operatorname{span}(c_0,\ldots,c_{k-1}) &= \mathbb{A}\,\mathcal{K}_k(\mathbb{A};r_0) \supset \mathbb{A}\,(\mathbb{A}\mathcal{K}_{k-1}(\mathbb{A};r_0)) \\ &= \mathbb{A}\operatorname{span}(c_0,\ldots,c_{k-2}), \end{aligned}$$

we see that $(r_k, \mathbb{A}c_j) = 0$ if $j < k-1$. Hence, by symmetry, $(\mathbb{A}r_k, c_j) = 0$ if $j < k-1$. Therefore, $u_k = K^{-1}r_k - \beta_{k-1}u_{k-1}$ and $c_k = \mathbb{A}r_k - \beta_{k-1}c_{k-1}$, where $\beta_k = (\mathbb{A}r_k, c_{k-1})/(c_{k-1}, c_{k-1})$ (cf. (32)). Apparently, in the symmetric case, in the GCR algorithm,

$$\texttt{compute by mod.G-S}\ \ u_k = \tilde{u} - Q_k\tilde{c},\ \ c_k = \tilde{c} - P_k\tilde{c},$$

may be replaced by

$$u_k = \tilde{u} - \beta_k u_{k-1}, \quad c_k = \tilde{c} - \beta_k c_{k-1} \quad \text{with} \quad \beta_k \equiv (\tilde{c}, c_{k-1})/\gamma_{k-1}. \tag{49}$$

Moreover, using the symmetry and orthogonality relations, one can show that

$$\beta_k = -\frac{\rho_k}{\rho_{k-1}} \quad \text{and} \quad \alpha_k = \frac{\rho_k}{\gamma_k} \quad \text{where} \quad \rho_k = (\tilde{c}, r_k), \tag{50}$$

thus saving another dot. This results in the **CR** (**Conjugate Residuals**) algorithm.

CR is cheap in memory and computational costs: each step requires 1 Mv, 2 dot and 4 axpy. However, this efficient form of GCR for symmetric problems is sensitive to errors due to finite precision arithmetic. In the GCR algorithm these errors also are projected orthogonally to the previously constructed set of vectors c_j while CR projects orthogonally only on one vector. Consequently, the orthogonality of the sequence $c_0, \ldots, c_k$ may be (and will be) lost if k increases. As a consequence, in actual computation, CR may show no accelerations (cf. 4.1). In spite of

```
choose x_0,
k = -1,   r_0 = b - A x_0,   u_{-1} = 0,   ρ_{-1} = 1,
repeat until ||r_{k+1}|| is small enough:
    { k := k + 1,
      ũ := A^T r_k,
      ρ_k = ||r_k||^2,  β_k = -ρ_k/ρ_{k-1},
      u_k = ũ - β_k u_{k-1},  c_k = A u_k,
      γ_k = ||u_k||^2,  α_k = ρ_k/γ_k,
      x_{k+1} = x_k + α_k u_k,  r_{k+1} = r_k - α_k c_k
```

ALGORITHM 4: The CGNR algorithm.

this instability and possible lack of acceleration the speed of convergence is relatively high. One can show (using Paige's theory in [26]) that, in spite of evaluation errors, for the residuals r_k computed by the CG algorithm we have

$$\frac{\|r_k\|}{\|r_0\|} \lesssim \left(\frac{1-\sqrt{\frac{\lambda_1}{\lambda_n}}}{1+\sqrt{\frac{\lambda_1}{\lambda_n}}}\right)^k \approx \exp\left(-2k\sqrt{\frac{\lambda_1}{\lambda_n}}\right) \quad \text{if} \quad \sigma(\mathbb{A}) \subset [\lambda_1, \lambda_n]. \tag{51}$$

If $\mathbb{A}$ is symmetric and positive definite then CR does not break down and $[x, y] := (\mathbb{A}^{-1}x, y)$ defines an inner product. If we substitute the definition of the new inner product in the CR algorithm formulated with respect to the new inner product we find one form of the well-known **CG** (**Conjugate Gradients**) algorithm [19]. Now $\rho_k = (\mathbb{A}^{-1}\tilde{c}, r_k) = \|r_k\|^2$ and tracking the size of the residuals does not require an additional dot. Moreover, $\gamma_k = [c_k, c_k] = [\tilde{c}, c_k] = (r_k, c_k)$. If we compute $c_k = Au_k$ then we do not need $\tilde{c}$ and we can save an additional axpy.

5.2 CGNR (Conjugate gradient for normal equations)

If CG is applied to the problem with preconditioner K given by $K^{-1} = A^T$ then $\mathbb{A} = AA^T$ is symmetric positive definite and the method is called **CGNR** (**Conjugate Gradients for Normal Relations**; see ALGORITHM 4). Similarly, CRNR is preconditioned CR with preconditioner $K = (A^T)^{-1}$. CGNR is (equivalent to) unpreconditioned CG applied to the **normal equations** $A^TAx = A^Tb$.

If (r_k) is the sequence of residuals from the CGNR then one can show that[4]

$$\frac{\|r_k\|}{\|r_0\|} \leq \varepsilon_k\left(\sigma(A^TA)\right) \lesssim \left(\frac{1-\frac{1}{\mathcal{C}(A)}}{1+\frac{1}{\mathcal{C}(A)}}\right)^k \approx \exp\left(-2k\frac{1}{\mathcal{C}(A)}\right), \tag{52}$$

where $\mathcal{C}(A)$ is the condition number of A. The upper bound in (52) does not involve the condition number of the eigenvector basis of A (cf. 4.1). One can show that the convergence of CGNR depends only on the singular values of A (that is, on the square root of the eigenvalues of A^TA) and *not* on the conditioning of the eigenvector basis of A.

Example 1 (continued 3) If $K = I$ and A is symmetric then the condition number of A is the ratio of the largest eigenvalue λ_n and the smallest eigenvalue λ_1: λ_n/λ_1. In CGNR the condition number instead of its square root determines the speed of convergence (cf. (10) and (52)).

[4]The first estimate is sharp in some generic sense (see [16]).

In view of the above example, we expect that GCR converges much faster than CGNR specifically when $\mathcal{C}(\mathbb{A})$ is large, $\mathcal{C}_E$ is not too large, and $\mathbb{A}$ is not too far from being symmetric. A poor performance of CGNR should not be credited to the CG process. Usually a given large symmetric problem can best be solved by CG. However, our initial problem was not symmetric. We made it symmetric by multiplying by A^T, thus, risking the introduction of a matrix with an very large condition number. Still, there are situations, as the following example shows, for which CGNR performs much better than GCR.

Example 6 If A is the $n \times n$-circular matrix (i.e. $Ae_j = e_{j-1}$ for $j = 2, \ldots, n$ and $Ae_1 = e_n$) then GCR with $K = I$ needs n steps to solve the equation $Ax = e_1$ with $x_0 = 0$ (exactly, or with an accuracy $< \frac{1}{n}$) while CRNR only needs 1 step.

If A is the $2n \times 2n$-block diagonal matrix with consecutive diagonal blocks $\begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$ then GCR needs 2 steps to solve the equation $Ax = \mathbf{1}$ with $x_0 = 0$ while CGNR needs $\approx n$ steps.

6 Nested Krylov subspace methods

In the course of the computation, both the computational work and the memory requirements increase in any of the algorithmic representations of the Krylov subspace method with residuals of minimal norm. To limit the costs, we considered the possibility of truncation and assemblage in subsection 4.2.2. In this section, we place this approach in a more general setting to find more flexible and robust algorithms. We took our ideas from [46]. Related modifications can be found in [4] and [31].

As with GCR, suppose we have an orthogonal system $c_0, \ldots, c_{k-1}$ and for any of the c_j we also have the u_j for which $c_j = Au_j$. With P_k and Q_k as in (31) and (36) the vector r_k of the form $r_k = r_0 - \sum_{j=0}^{k-1} \gamma_j c_j$ that has minimal norm is given by $r_k = r_0 - P_k r_0$. This minimal vector is the residual of the approximation $x_k = x_0 + Q_k r_0$. By means of the stabilized recursion as in (30), x_k and its residual r_k can efficiently be computed. The recursion is particularly efficient if we extend the orthogonal system of c_j's recursively: then the consecutive x_k and r_k can be computed as in (38).

So, here as well as in the derivation of GCR, we have to answer the question how to compute the next u_k for which $c_k = Au_k$ is orthogonal to any of the c_j with $j < k$. The next c_k should also reduce the norm of the residual $r_{k+1} = r_k - \alpha_k c_k$ as much as possible. Clearly, if u solves the equation $Au = r_k$ then $c_k = Au$ gives the best reduction. Unfortunately, we can not obtain this solution u cheaply (otherwise we also would be able to solve the original equation $Ax = b$), but any good approximation $\tilde{u}$ of u may be worth considering. If $u_k \equiv \tilde{u} - Q_k A\tilde{u}$ then $c_k \equiv Au_k = (I - P_k)A\tilde{u}$ is orthogonal to the previous c_j and, since $r_k = (I - P_k)r_k$ we also have

$$\|r_k - \alpha c_k\| = \|r_k - \alpha A u_k\| = \|(I - P_k)(r_k - \alpha A\tilde{u})\| \leq \|r_k - \alpha A\tilde{u}\|.$$

Hence, if $\tilde{u}$ (or $\alpha\tilde{u}$) is a good approximate solution (i.e. $\|r_k - \alpha A\tilde{u}\|$ is "small"), then αu_k is not worse and c_k has the required orthogonality property.

So, we propose ALGORITHM 5 in which we have to specify how to

"`find an approximate` $\tilde{u}$ to the `solution` u `of` $Au = r_k$".

Since K^{-1} or the updated preconditioner H_k (see (47)) approximates A^{-1} in some sense, the choice $\tilde{u} = K^{-1}r_k$ or $\tilde{u} = H_k r_k$ yields a reasonable approximation of the solution u of the equation $Au = r_k$. With the choice $\tilde{u} = K^{-1}r_k$ we have precisely GCR. By taking $\tilde{u} = H_k r_k$ we have $u_k = \tilde{u}$ (see (48)) and the subsequent orthogonalization $u_k = \tilde{u} - Q_k\tilde{c}$, $c_k = \tilde{c} - P_k\tilde{c}$ is superfluous: here, we also have some form of GCR.

```
choose x_0,
k = -1,   r_0 = b - A x_0,
repeat until ||r_{k+1}|| is small enough:
   { k := k + 1,
   { find an approximate solution ũ of Au = r_k,   c̃ := Aũ,
   { compute by mod.G-S   u_k = ũ - Q_k c̃,   c_k = c̃ - P_k c̃,
   { γ_k = (c_k, c_k),  α_k = (r_k, c_k)/γ_k,
   { x_{k+1} = x_k + α_k u_k,  r_{k+1} = r_k - α_k c_k
```

ALGORITHM 5: The GMRESR algorithm.

The problem $Au = r_k$ can be solved approximately also by some iterative method. The algorithm that uses l steps of GMRES for this purpose is referred to as **GMRESR(l)** [46]. The vectors c_j can be used to speed-up the "local" iterative method [6]. Obviously, one can also restart methods as GMRESR(l) after m GCR iterations steps [46] or one can maintain only m GCR vectors c_j or some linear combinations of c_j's [6].

Numerical experiments show that GMRES(lm) (or GCR(lm)) and the restarted version GMRESR(m, l) of GMRESR(l) often require the same number of Mv's to obtain an equally good approximation (i.e. with a residual of comparable size; [46]). On average per Mv, GMRESR(m, l) is cheaper than GMRES(lm) by a factor $\frac{2}{l} + \frac{1}{m}$ with respect to the other computational costs (axpys and dots). We have a same factor in favor of GMRESR(m, l) for the amount of storage that is needed. Therefore, depending on the complexity of a matrix multiplication (on the sparseness of A and the choice for the preconditioner) GMRESR(m, l) can be very competitive. Since the optimal choice for l and m depends on the complexity of one Mv but also on the speed of convergence it is hard to make a good à priori choice.

From example 6 we learned that GCR may stagnate where CGNR performs extremely well. This experience may suggest a possibility to avoid stagnation in GCR: use ALGORITHM 5 and (once in a while) a few steps CGNR to solve approximately $Au = r_k$.

Put $A_k = (I - P_k)A$. Note that $r_k = (I - P_k)r_k$ and

$$A_k = (I - P_k)A = (I - P_k)(I - P_k)A = (I - P_k)A(I - Q_kA). \tag{53}$$

Moreover Q_kA is a (non-orthogonal) projection: $Q_kAQ_kA = Q_kP_kA = Q_kA$. Hence $I - Q_kA$ is a projection, say on the space $\mathbf{X}_k$. $I - P_k$ is a (orthogonal) projection, say on $\mathbf{Y}_k$, and A_k can considered to be a map from $\mathbf{X}_k$ onto $\mathbf{Y}_k$. As a map between these spaces, A_k is invertible. Therefore, instead of solving $Au = r_k$, one may as well solve the equation

$$A_k u = r_k \quad \text{with} \quad u \in \mathbf{X}_k. \tag{54}$$

Since $\mathbf{Y}_k$ and $\mathbf{X}_k$ will not coincide, we need a "preconditioner" H_k that maps $\mathbf{Y}_k$ bijectively onto $\mathbf{X}_k$. For instance, $H_k = (I - Q_kA)K^{-1}$ (see [6]).

If we use l steps of the GCR algorithm in the inner loop with preconditioning matrix $(I - Q_kA)K^{-1}$ to solve the equation $A_ku = r_k$ approximately (i.e. in the inner loop, we take $(I - Q_kA)K^{-1}$ instead of K^{-1} and in the matrix-vector multiplications we multiply by A_k) then the resulting algorithm is precisely our l-static assemblage GCR algorithm.
Recall that the GCR algorithm is 50% more expensive than the GMRES algorithm. We can save on GCR(m,l) by using l steps of the preconditioned GMRES algorithm to solve $A_ku = r_k$ approximately [6].

The method in [7] can be represented by the above algorithm as well: there, the authors take the solution $\tilde{u}$ of $Ku = (I - \mathbb{A})r_k$ as an "approximation" of the solution u of $Au = r_k$.

With $\tilde{u}$ such that $K\tilde{u} = (2 - \mathbb{A})r_k$, $\tilde{u}$ is the approximation of the solution u of the equation $Au = r_k$ that one obtains by two steps of the basic iterative method with start "x_0" equal to 0.

Observe also that $K^{-1}(2 - \mathbb{A})$ is the result of one Newton step applied to the matrix function $X \rightsquigarrow X^{-1} - A$ with initial approximation K^{-1}.

7 Quasi-optimal Krylov subspace methods

As experiments show, our strategy to limit the computational costs by assemblage or splitting the algorithm in inner- – and outer-loops is successful for a large class of problems. However, specifically for very large problems, the computational costs and the memory requirements for these adapted methods can still be unacceptable. The computational costs of methods like GCR(2,2) or the truncated version of GCR that uses only the last 3 or 4 vectors c_j and u_j in each step, do not become increasingly expensive when the number of steps increases. Unfortunately, these methods often converge slowly and need many steps to achieve the required accuracy. At the long run, these methods are still too expensive. In this section, we discuss methods that use in each step only a few vectors from previous steps and that still are the best in some adapted sense.

If $c_0, c_1, \ldots$ is a basis of the Krylov space of $\mathbb{A}$ generated by c_0 then the matrix B of $\mathbb{A}$ with respect to this basis is upper Hessenberg and $\mathbb{A}C = CB$, where C is the $n \times n$-matrix with $c_0, c_1, \ldots$ in its consecutive columns. If the c_j form an orthogonal system then we are back in the situation as discussed before. Now, we hope to find a (non-orthogonal) basis of c_j's by which we can project efficiently on the Krylov subspace $\mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)$ as in GCR or on $\mathcal{K}_k(\mathbb{A}; r_0)$ as in GMRES: our projections should involve short recursions as in (49). In methods as GCR and GMRES, the projection coefficients appear in the Hessenberg matrix (see 4.4 with $C = C_n$ and $B = B_n$. Concerning GCR, we observe that the upper-triangular matrix T_n is bi-diagonal if and only if $B_n = T_n S_n D_n^{-1}$ is tri-diagonal, where T_n, D_n and S_n are as in 4.4). Only a few of these coefficients are non-zero in each step if and only if the Hessenberg matrix B has only a few non-zero coefficients on each row. Therefore, we may hope to find short recursions if, for instance, B has only a few non-zero diagonals. For ease of argumentation, we consider the case where B is tri-diagonal, or equivalently, where B^T is upper Hessenberg as well. Since $\mathbb{A}^T(C^{-1})^T = (C^{-1})^T B^T$ we see that B^T is upper Hessenberg if and only if the columns of $(C^{-1})^T$ form a basis of the Krylov space of $\mathbb{A}^T$ generated by the first column of $(C^{-1})^T$.

Since $C^{-1}C = I$, the columns $\tilde{c}_0, \tilde{c}_1, \ldots$ of C^{-1} are characterized by bi-orthonormality with respect to the sequence $c_0, c_1, \ldots$, i.e.: $(c_j, \tilde{c}_i) = 0$ (bi-orthogonal) and $(c_j, \tilde{c}_j) = 1$ (bi-normal) $(i \neq j)$. Therefore, in order to obtain a basis for the Krylov space of $\mathbb{A}$ generated by c_0, for which the matrix B of $\mathbb{A}$ is tri-diagonal, we have to construct a pair of bi-orthonormal sequences that form a basis of the Krylov space of $\mathbb{A}$ and of $\mathbb{A}^T$, respectively. We will perform this construction by an analog of Gram-Schmidt, by the **Lanczos bi-orthogonalization** process starting with c_0 and some vector $\tilde{c}_0$ (in (59)). The construction may break down if, for the choice of the generator $\tilde{c}_0$, the Hessenberg matrix can not be tri-diagonal. In such a situation one can start again with another $\tilde{c}_0$ or one can try to cure the breakdown by aiming at, for instance, a locally penta-diagonal Hessenberg matrix (by **looking ahead**, [13]): here, for simplicity, we will assume that the processes do not break down (in finite precision arithmetic this is a rare event, indeed).

Again we consider residuals r_k of the form $r_k = q_k(\mathbb{A})r_0$ for some polynomial q_k in $\mathcal{P}_k^1$. By employing these bi-orthonormal basis of Krylov spaces, we find residuals that are optimal in the sense that

$$\begin{aligned}
\|C^{-1}r_k\| &= \min\{\|C^{-1}(r_0 - \mathbb{A}p(\mathbb{A})r_0)\| \,\big|\, p \in \mathcal{P}_{k-1}\} && (55)\\
&= \min\{\|C^{-1}(r_0 - \mathbb{A}u)\| \,\big|\, u \in \mathcal{K}_k(\mathbb{A}; r_0)\} && (56)\\
&= \min\{\|C^{-1}(r_0 - c)\| \,\big|\, c \in \mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)\} \quad (\Leftrightarrow\ C^{-1}r_k \perp C^{-1}\mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)\ ), && (57)
\end{aligned}$$

Since $\mathcal{K}_k(H; e_1) = \mathrm{span}\,(e_1, \ldots, e_k)$ for any Hessenberg matrix H and since both B and B^T are

Hessenberg, we have that

$$C^{-1}\mathcal{K}_k(\mathbb{A};c_0)=\mathcal{K}_k(B;e_1)=\mathcal{K}_k(B^T;e_1)=C^T\mathcal{K}_k(\mathbb{A}^T;\tilde{c}_0).$$

Hence, if c_0 is some scalar multiple of $\mathbb{A}r_0$ we have that

$$C^{-1}r_k\perp C^{-1}\mathcal{K}_k(\mathbb{A};c_0)\qquad\Leftrightarrow\qquad r_k\perp\mathcal{K}_k(\mathbb{A}^T;\tilde{c}_0).$$

Apparently, the "quasi-minimal residual" approach in (57) is equivalent to a "quasi-orthogonal residual" approach (cf. the introduction of section 4).

QMR chooses $c_0=r_0$ and solves (56) (similar to GMRES), Bi-CG chooses $c_0=\mathbb{A}r_0$ and solves (57) (similar to GCR), or equivalently, puts r_k orthogonal to $\mathcal{K}_k(\mathbb{A}^T;\tilde{r}_0)$ for some $\tilde{r}_0$ (similar to FOM).

7.1 Convergence

For the "quasi-minimal" residuals for which (55–57) holds we have (cf. (26))

$$\frac{\|r_k\|}{\|r_0\|}\leq\mathcal{C}(C)\min_{q\in\mathcal{P}_k^1}\|q(\mathbb{A})\|.\tag{58}$$

If $c_0=r_0$ (as in QMR) then $\mathcal{C}(C)$ may be replaced by the condition number of the matrix with columns $c_0,\ldots,c_k$, which is less than $k\max_{j\leq k}\|c_j\|\max_{j\leq k}\|\tilde{c}_j\|$. One can cheaply keep track of this quantity and, thus obtain some information on how well the method finds good Krylov subspace approximations: if $\mathcal{C}(C)$ is not too large then, in view of (7.1), we expect to have about the smallest residual one can get by any Krylov subspace method.

In exact arithmetic, these methods will exhibit accelerations much the same as for the optimal Krylov subspace methods (cf. 4.1). Although, here also, the accelerations depend on spectral properties of $\mathbb{A}$, the situation is obscured by the non-orthogonal basis transformation C.
When employing Lanczos bi-orthogonalization, C, and hence the speed of convergence, depends on the choice of $\tilde{c}_0$ ($\tilde{r}_0$ in the algorithms below). Unfortunately, there is no good strategy for selecting $\tilde{c}_0$. One often chooses $\tilde{c}_0=c_0$ and restarts with a new choice in the rare case of breakdown. Furthermore, the bi-orthogonality (cf. 5) will get lost in the course of the computation due to evaluation errors. This slows down the speed of convergence and may even lead to divergence. Nevertheless, for a large class of problems, these methods appear to do well. Compared to GCR, say, each step is so cheap that this often compensates less optimal convergence.

The C of Bi-CG (with $c_0=\mathbb{A}r_0$) differs from the C of QMR (where $c_0=r_0$). Nevertheless, these two methods exhibit roughly the same convergence behavior (when using the same $\tilde{c}_0$). The convergence history of the QMR residuals tends to be smoother than the one of Bi-CG and follows more or less the history of the local-best Bi-CG residuals.

7.2 Lanczos bi-orthogonalization

If $c'_0,c'_1,\ldots$ and $\tilde{c}'_0,\tilde{c}'_1,\ldots$ are basis of the Krylov space of $\mathbb{A}$ and of $\mathbb{A}^T$, respectively, and $c_0,\ldots,c_{k-1}$ and $\tilde{c}_0,\ldots,\tilde{c}_{k-1}$ are constructed from them by bi-orthogonalization, then we construct our next c_k and $\tilde{c}_k$ by means of the map $\tilde{P}_k$ given by

$$\tilde{P}_k\equiv\sum_{j=0}^{k-1}\frac{c_j\tilde{c}_j^T}{\tilde{c}_j^Tc_j}.$$

$\tilde{P}_k$ is a projection (non-orthogonal, unless $c_j=\tilde{c}_j$ for all j) onto the span of $c_0,\ldots,c_{k-1}$, hence onto $\mathcal{K}_k(\mathbb{A};c_0)$. Moreover, since $(I-\tilde{P}_k)c_j=0\quad(j<k)$, we see that $(I-\tilde{P}_k^T)y\perp c_j$ for any y and $j<k$. Similarly, $(I-\tilde{P}_k)y\perp\tilde{c}_j$. If we select some scaling factors τ_k and $\tilde{\tau}_k$ (i.e. non-zero

scalars) then $c_k \equiv \tau_k(I - \tilde{P}_k)c'_k$ and $\tilde{c}_k \equiv \tilde{\tau}_k(I - \tilde{P}_k^T)\tilde{c}'_k$ are our next bi-orthogonal vectors. Theoretically, any choice of non-zero scalars τ_k and $\tilde{\tau}_k$ will do. One often chooses $\tau_k = \tilde{\tau}_k = 1$ (in Bi-CG) or τ_k and $\tilde{\tau}_k$ such that $\|c_k\| = 1$ and $\tilde{c}_k^T c_k = 1$ (in QMR).

The sequences $c_0, c_1, \ldots$ and $\tilde{c}_0/\tilde{c}_0^T c_0, \tilde{c}_1/\tilde{c}_1^T c_1, \ldots$ are bi-orthonormal.

If, for all j, $c'_j = \mathbb{A}c_{j-1}$ and $\tilde{c}'_j = \mathbb{A}^T\tilde{c}_{j-1}$ then the coefficients in the projection of the c'_j constitute the matrix B of $\mathbb{A}$ with respect to $c_0, c_1, \ldots$. Since this **Lanczos matrix** is tri-diagonal, we have that

$$\tau_{k\,k-1}c_k = \mathbb{A}c_{k-1} - \sum_{j=1}^{2} \tau_{k-j\,k-1}c_{k-j} \quad \text{and} \quad \tilde{\tau}_{k\,k-1}\tilde{c}_k = \mathbb{A}^T\tilde{c}_{k-1} - \sum_{j=1}^{2} \tilde{\tau}_{k-j\,k-1}\tilde{c}_{k-j}, \tag{59}$$

where $c_{-1} = \tilde{c}_{-1} = 0$, $\tau_{k\,k-1} = \tau_k$, $\tilde{\tau}_{k\,k-1} = \tilde{\tau}_k$ and

$$\tau_{j\,k-1} = (\mathbb{A}c_{k-1}, \tilde{c}_j)/(c_j, \tilde{c}_j) \quad \text{and} \quad \tilde{\tau}_{j\,k-1} = (\mathbb{A}c_j, \tilde{c}_{k-1})/(c_j, \tilde{c}_j) \quad (j = k-1, k-2). \tag{60}$$

We apply these observations in a GCR type of algorithm and in a GMRES type of algorithm, yielding Bi-CG and QMR, respectively.

Lanczos [21] used this bi-orthogonalization procedure for finding eigenvalues of $\mathbb{A}$. We will briefly discuss his approach in subsection 7.5.

7.3 Bi-conjugate gradient

Choose some x_0 and let $r_0 = b - Ax_0$.

We choose some $\tilde{s}_0$, we take $c_0 = \mathbb{A}r_0$, $\tilde{u}_0 = \mathbb{A}^T\tilde{s}_0$ and we consider some basis $\tilde{u}_0, \ldots, \tilde{u}_{k-1}$ of $\mathcal{K}_k(\mathbb{A}^T; \tilde{u}_0)$. Suppose $u_0, \ldots, u_{k-1}$ are such that $c_0, \ldots, c_{k-1}$, with $c_j = Au_j$, form a basis of $\mathcal{K}_k(\mathbb{A}; \mathbb{A}r_0)$ that is **bi-orthogonal** with respect to the $\tilde{u}_j$. Consider the maps $\tilde{P}_k$ and $\tilde{Q}_k$ given by

$$\tilde{P}_k(y) \equiv \sum_{j=0}^{k-1} \frac{c_j\tilde{u}_j^T}{\tilde{u}_j^T c_j} \quad \text{and} \quad \tilde{Q}_k(y) \equiv \sum_{j=0}^{k-1} \frac{u_j\tilde{u}_j^T}{\tilde{u}_j^T c_j}. \tag{61}$$

$\tilde{P}_k$ is a projection that bi-orthogonalizes and $\tilde{Q}_k$ is the original of $\tilde{P}_k$ under A: $\tilde{P}_k = A\tilde{Q}_k$. Obviously, we can now copy the construction of the GCR algorithm by choosing $c'_k = \mathbb{A}r_k$ and $\tilde{u}'_k = \mathbb{A}^T\tilde{s}_k$, where $\tilde{s}_k = (I - \tilde{P}_k^T)\tilde{s}_0$. This leads to an algorithm that could be called Bi-CR (Bi-Conjugate Residuals). As in GCR, $r_k = r_{k-1} - \alpha_{k-1}c_{k-1}$ and $\tilde{s}_k = \tilde{s}_{k-1} - \tilde{\alpha}_{k-1}\tilde{u}_{k-1}$ for some appropriate scalars α_{k-1} and $\tilde{\alpha}_{k-1}$. As expected, we also have short recursions for c_k and $\tilde{u}_k$. We even have

$$c_k = \mathbb{A}r_k - \beta_k c_{k-1}, \quad \tilde{u}_k = \mathbb{A}^T\tilde{s}_k - \tilde{\beta}_k\tilde{u}_{k-1} \quad \text{with} \quad \tilde{\beta}_k = \beta_k \equiv \frac{(\mathbb{A}r_k, \tilde{u}_{k-1})}{(c_{k-1}, \tilde{u}_{k-1})}, \tag{62}$$

$$r_{k+1} = r_k - \alpha_k c_k, \quad \tilde{s}_{k+1} = \tilde{s}_k - \tilde{\alpha}_k\tilde{u}_k \quad \text{with} \quad \tilde{\alpha}_k = \alpha_k \equiv \frac{(r_k, \tilde{u}_k)}{(c_k, \tilde{u}_k)}. \tag{63}$$

In an ORTHODIR approach, we would choose $c'_k = \mathbb{A}c_{k-1}$ and $\tilde{u}'_k = \mathbb{A}\tilde{u}_{k-1}$ as in 7.2. This would lead to longer recursions for the c_k and $\tilde{u}_k$: $c_k = \mathbb{A}c_{k-1} - \tau_{k-1\,k-1}c_{k-1} - \tau_{k-2\,k-1}c_{k-2}$, etc..

To compute the scalars β_k and α_k, we do not need the $\tilde{s}_j$ nor the $\tilde{u}_j$ explicitly, we only need $\mathbb{A}^T\tilde{u}_j$. Neither do we need $c'_k = \mathbb{A}r_k$: since, for instance,

$$(\mathbb{A}r_k, \tilde{u}_{k-1}) = (r_k, \mathbb{A}^T\tilde{u}_{k-1}), \qquad (c_{k-1}, \tilde{u}_{k-1}) = (r_{k-1}, \mathbb{A}^T\tilde{u}_{k-1}),$$

and we can compute c_k by $c_k = Au_k$. We can compute $\mathbb{A}^T\tilde{u}_j$ from $\mathbb{A}^T\tilde{u}_k = \mathbb{A}^T\mathbb{A}^T\tilde{s}_k - \tilde{\beta}_k\mathbb{A}^T\tilde{u}_{k-1}$ and $\mathbb{A}^T\tilde{s}_{k+1} = \mathbb{A}^T\tilde{s}_k - \tilde{\alpha}_k\mathbb{A}^T\tilde{u}_k$. This saves one axpy as compared to Bi-CR.

```
choose x_0 and r̃_0,
k = -1,   r_0 = b - A x_0,   u_{-1} = 0,   ρ_{-1} = 1,
repeat until ||r_{k+1}|| is small enough:
    k := k + 1,
    solve u' from K u' := r_k,                          c̃' = 𝔸^T r̃_k
    ρ_k = (r_k, r̃_k),  β_k = -ρ_k/ρ_{k-1},
    u_k = u' - β_k u_{k-1},   c_k = A u_k,               c̃_k = c̃' - β_k c̃_{k-1}
    γ_k = (c_k, r̃_k),  α_k = ρ_k/γ_k,
    x_{k+1} = x_k + α_k u_k,   r_{k+1} = r_k - α_k c_k,   r̃_{k+1} = r̃_k - α_k c̃_k
```

Algorithm 6: The Bi-CG algorithm.

We write $\tilde{c}_j \equiv \mathbb{A}^T \tilde{u}_j$ and $\tilde{r}_j \equiv \mathbb{A}^T \tilde{s}_j$. In particular, we have that $\tilde{u}_0 = \tilde{r}_0$. Since we were free to select any initial $\tilde{r}_0$ we as may well select $\tilde{r}_0$ instead of $\tilde{s}_0$.

Now, note that

$$c_k,\, r_k = (I - \tilde{P}_k) r_0 \perp \mathcal{K}_k(\mathbb{A}^T; \tilde{r}_0) = \mathrm{span}\,(\tilde{u}_0, \ldots, \tilde{u}_{k-1}) \ni \tilde{r}_j \quad (j < k). \tag{64}$$

Therefore, for the scalar β_k we have that

$$\beta_k = \frac{(r_k, \tilde{c}_{k-1})}{(r_{k-1}, \tilde{c}_{k-1})} = \frac{(r_k, \tilde{r}_{k-1} - \tilde{r}_k)}{(r_{k-1}, \tilde{r}_{k-1} - \tilde{r}_k)} = -\frac{(r_k, \tilde{r}_k)}{(r_{k-1}, \tilde{r}_{k-1})}.$$

For the scalar α_k the following relations are useful. They follow from applying (64).

$$(r_k, \tilde{u}_k) = (r_k, \tilde{r}_k) \quad \text{and} \quad (r_k, \tilde{c}_k) = (c_k, \tilde{r}_k)$$

and this leads to **Bi-CG** (**Bi-Conjugate Gradients** [21, 11]) in Algorithm 6.

Each Bi-CG step is cheap. Estimate (58) may give some theoretical information on the speed of convergence. However, we don't have à priori information on $\mathcal{C}(C)$ and since we wish to keep k relatively small, we will never have this.

It was not our purpose to obtain the sequence of residuals r_k bi-orthogonal to the sequence of "shadow residuals" $\tilde{r}_k$. Nevertheless, according to (64), they have this bi-orthogonality relation.

We can easily recover Lanczos tri-diagonal matrices from the Bi-CG coefficients α_k and β_k (see subsection 7.5), and this fact can be used to find approximations for (the extremal) eigenvalues of $\mathbb{A}$ (see 7.5).

In fact, Bi-CG can be viewed as solving the projected tri-diagonal system

$$0 = \tilde{R}_k^T(\mathbb{A} R_k \vec{\gamma} - r_0) = \tilde{R}_k^T R_{k+1}(S_k D_k^{-1} T_k \vec{\gamma} - e_1) \quad \text{(cf. 7.5)},$$

where the solution process is done via LU decomposition of the tri-diagonal Lanczos matrix. This is made visible in Bi-CG by the two coupled two term recurrences for the residuals and search directions.

7.4 Quasi-Miminal Residuals

Choose some x_0 and $\tilde{r}_0$, $r_0 = b - Ax_0$.

In the GMRES version of this quasi-minimalization approach, the so-called **QMR** (**Quasi-Minimal Residuals**) algorithm [13], a pair of bi-orthogonal sequences (c_j) and $(\tilde{c}_j)$ is constructed by Lanczos bi-orthogonalization (59-60), starting with $c_0' = r_0$ and $\tilde{c}_0' = \tilde{r}_0$. In matrix notation, (59) reads as $\mathbb{A} C_k = C_{k+1} \underline{B}_k$, where $\underline{B}_k$ is the $(k+1) \times k$ Lanczos tri-diagonal matrix and C_k is the $n \times k$ matrix with columns $c_0, \ldots, c_{k-1}$, and (56) is equivalent to

$$\|C^{-1} r_k\| = \min_{\vec{\gamma}} \|\rho_0 e_1 - \underline{B}_k \vec{\gamma}\|, \tag{65}$$

where ρ_0 is such that $r_0 = \rho_0 c_0$. If $\vec{\gamma}^Q = (\gamma_0^Q, \ldots, \gamma_{k-1}^Q)^T \in \mathbb{R}^k$ is the solution of this minimization problem then

$$K(x_k - x_0) = \sum_{j=0}^{k-1} \gamma_j^Q c_j. \tag{66}$$

If we would compute x_k from this relation, we would need all previously computed c_j's which would be expensive on memory space. We can save in memory by updating the approximation x_k at each step, as we will see below.

If $\underline{B}_k = U_{k+1}\underline{R}_k$ is a QR-decomposition of $\underline{B}_k$ obtained by a repeated application of Givens rotations then U_{k+1} is a $(k+1)\times(k+1)$ orthonormal matrix and $\underline{R}_k$ is a $(k+1)\times k$ upper tri-diagonal matrix. Moreover, if the k-vector $\vec{\mu}_k$ and the scalar ρ_k is such that $(\vec{\mu}_k^T, \rho_k)^T = Q_{k+1}^T e_1$ then $\vec{\gamma}^Q = \rho_0 R_k^{-1}\vec{\mu}_k$, where R_k is the $k \times k$ upper block of $\underline{R}_k$. Since

$$|\rho_k| = \|\rho_0 e_1 - \underline{B}_k \vec{\gamma}^Q\|, \tag{67}$$

ρ_k can be used to estimate the accuracy (cf. (65)). We introduce the $n \times k$ matrix W_k with columns $w_0, \ldots w_{k-1}$ given by $W_k = C_k R_k^{-1}$. Then

$$K(x_k - x_0) = \sum_{j=0}^{k-1} \gamma_j^Q c_j = C_k \vec{\gamma}^Q = \rho_0 W_k \vec{\mu}_k. \tag{68}$$

Since R_k is upper tri-diagonal and $C_k = W_k R_k$ we see that

$$c_j = \eta_j w_{j-2} + \theta_j w_{j-1} + \zeta_j w_j \quad \text{for} \quad j = 2, \ldots, k-1. \tag{69}$$

In particular, this implies that the columns of W_{k-1} are precisely the first $k-1$ columns of W_k. By exploiting the fact that U_k arises from repeated application of Givens rotations one can show that $\vec{\mu}_k = (\vec{\mu}_{k-1}^T, 0)^T + \alpha_k e_k$ with $\alpha_k \equiv \kappa_k \rho_{k-1}$, where κ_k is some scalar (a cosinus) from the last Givens rotation (see [13] for details). A combination of this fact with (68) shows that

$$x_k = x_{k-1} + y_{k-1} \text{ where } y_{k-1} \text{ is the solution of } K y_{k-1} = \alpha_k w_{k-1}. \tag{70}$$

In one step, the QMR algorithm computes
– the new Lanczos vectors c_k and $\tilde{c}_k$ and new coefficients for the Lanczos matrix as in (59-60),
– the scalars for the new Givens rotation and the scalars η_k, θ_k, and ζ_k from the new coefficients of the Lanczos matrix and the new and the previous Givens rotation,
– $\rho_k = \sigma_k \rho_{k-1}$ and $\alpha_k = \kappa_k \rho_{k-1}$ from the new Givens rotation (with $\sigma_k^2 + \kappa_k^2 = 1$),
– w_{k-1} from $w_{k-1} = \frac{1}{\zeta_{k-1}}(c_{k-1} - \eta_{k-1} w_{k-3} - \theta_{k-1} w_{k-2})$ (see (69)),
– x_k as in (70).

On average, QMR and Bi-CG seem to have comparable speed of convergence. Actually, using the same initial residual and initial shadow residual for both methods, we have that

$$\left\| r_k^{\text{Bi-CG}} \right\| = \frac{|\rho_k|}{\sqrt{1 - |\sigma_k|^2}} \tag{71}$$

(see [13]), where $r_k^{\text{Bi-CG}}$ is the k-th residual generated by Bi-CG and $|\rho_k|$ estimates the norm $\|r_k^{\text{QMR}}\|$ of the k-th QMR residual (as in 7.1),

$$\sigma_- \, |\rho_k| \le \left\| r_k^{\text{QMR}} \right\| \le \sqrt{k+1}\, |\rho_k|,$$

with σ_- the smallest singular value of the matrix with columns $c_0, \ldots, c_k$ (cf. (65), (67)). Memory requirements and computational costs per step are about the same. However, QMR tends to converge a little more smoothly than Bi-CG and, as a consequence, QMR seems to be a little less sensitive to evaluation errors. In addition, QMR avoids one of the breakdown dangers of Bi-CG and allows more easily look-ahead strategies. The convergence history of Bi-CG shows a large peak in case Bi-CG nearly breaks down, while QMR stagnates (temporarily) at the same step (then $\sigma_k \approx 1$). FOM and GMRES show a comparable convergence relationship (cf. subsection 4.1, proposition 2).

7.5 Techniques to compute eigenvalues

After k steps, of the Lanczos process in (59-60), we have the $k \times k$ upper block of B. The eigenvalues of this block can easily be computed for a k of interest ($k \ll n$). They also approximate the eigenvalues of $\mathbb{A}$, the separated or isolated extremal eigenvalues first, and provide useful information (cf. 4.4 and 4.1).

Due to the fact that C is non-orthogonal, the convergence is much less smooth than for the cases we discussed in 4.4. Moreover, the construction in (59-60) is very sensitive to evalution errors: bi-orthogonality will be lost in a rather early stage of the process (i.e., even for modest k we will see that c_l and $\tilde{c}_{k+l}$ make large angles). This introduces "spurious" Ritz values stemming from evaluation errors. Surprisingly, these spurious Ritz values also often tend to converge to eigenvalues of $\mathbb{A}$ [26, 17]. A cluster of Ritz values consists of a "true" Ritz value that is close to some eigenvalue of $\mathbb{A}$ and a number of spurious Ritz values (also close to the same eigenvalue of $\mathbb{A}$).

In contrast to Bi-CG, QMR explicitly constructs and uses the Lanczos matrix. However, the Lanczos matrix can also easily be recovered from the Bi-CG scalars β_k and α_k. Because, as for GCR (cf. (39)), we have that $\mathbb{A}R_k = C_kT_k$ and $R_{k+1}S_k = C_kD_k$, where now T_k reduces to an upper triangular bi-diagonal matrix. Hence, $\mathbb{A}C_k = C_{k+1}T_{k+1}S_kD_k^{-1}$ (and $\mathbb{A}R_k = R_{k+1}S_kD_k^{-1}T_k$). Apparently, $T_{k+1}S_kD_k^{-1}$ (as well as $S_kD_k^{-1}T_k$) is the $(k+1)\times k$ right upper block of the Lanczos tri-diagonal matrix with respect to the basis of c_j (respectively, of r_j). We can exploit this fact to find approximations for (the extremal) eigenvalues of $\mathbb{A}$.

8 Mixed Krylov subspace methods

Although the computational costs of Bi-CG and QMR are low in terms of dots and axpys, each step requires two matrix multiplications, which is double the costs of GCR. Moreover, one Mv involves the multiplication by the transpose of $\mathbb{A}$ which is not always easily available. We will discuss modifications of Bi-CG that improve the speed of convergence and that use multiplications by $\mathbb{A}$ only; two multiplications per step.

We are not really interested in the Bi-CG vectors $\tilde{c}_k$ and $\tilde{r}_k$. Actually, we only need $\tilde{r}_k$ to compute β_k and α_k through the scalars ρ'_k and γ_k. However, we can compute these by means of any vector $\tilde{s}_k$ of the form $\tilde{s}_k = q_k(\mathbb{A}^T)\tilde{r}_0$ where q_k is some polynomial in $\mathcal{P}_k$ of which the leading coefficient is non-trivial and known.
To prove this we consider a $q_k \in \mathcal{P}_k$ with non-zero leading coefficient σ_k, that is $q_k(t) - \sigma_k t^k$ is a polynomial of degree $k-1$. $\tilde{r}_k = p_k(\mathbb{A}^T)\tilde{r}_0$ for some $p_k \in \mathcal{P}_k^1$ and $p_k(t) - \tau_k t^k$ is of degree $k-1$ for $\tau_k = (-\alpha_{k-1})(-\alpha_{k-1})\dots(-\alpha_0)$. Hence, both the vectors $\tilde{r}_k - \tau_k(\mathbb{A}^T)^k\tilde{r}_0$ and $\tilde{s}_k - \sigma_k(\mathbb{A}^T)^k\tilde{r}_0$ belong to $\mathcal{K}_{k-1}(\mathbb{A}^T;\tilde{r}_0)$. Since both the vectors r_k and c_k are orthogonal to this space (see (64)), we see that

$$\rho_k \equiv (r_k, \tilde{r}_k) = \frac{\tau_k}{\sigma_k}\rho'_k \quad \text{with} \quad \rho'_k \equiv (r_k, \tilde{s}_k)$$
$$\gamma_k \equiv (c_k, \tilde{r}_k) = \frac{\tau_k}{\sigma_k}\gamma'_k \quad \text{with} \quad \gamma'_k \equiv (c_k, \tilde{s}_k).$$

Hence

$$\beta_k = -\frac{\tau_k \sigma_{k-1}}{\sigma_k \tau_{k-1}} \frac{\rho'_k}{\rho'_{k-1}} \quad \text{and} \quad \alpha_k = \frac{\rho'_k}{\gamma'_k}. \tag{72}$$

This observation does not improve the algorithm significantly: in order to compute $\tilde{s}_k$ recursively, in each step, we will still have to multiply by $\mathbb{A}^T$. We might save two vector updates, by taking $\tilde{s}_k = \mathbb{A}^T \tilde{s}_{k-1}$, but this approach may affect the stability seriously. However, one might try to invest the effort of computing $\tilde{s}_k$ in an attempt to force an additional reduction of r_k. For this purpose, the following observation is important

$$\rho'_k = (r_k, q_k(\mathbb{A}^T)\tilde{r}_0) = (q_k(\mathbb{A})r_k, \tilde{r}_0) \quad \text{and} \quad \gamma'_k = (c_k, q_k(\mathbb{A}^T)\tilde{r}_0) = (q_k(\mathbb{A})c_k, \tilde{r}_0). \tag{73}$$

In the ideal case the operator $p_k(\mathbb{A})$ reduces r_0. One may try to select q_k such that $\mathbb{Q}_k \equiv q_k(\mathbb{A})$ again reduces r_k as much as possible. In such a case it would be an advantage to avoid computation of r_k and u_k, and one might try to compute immediately $\mathbf{r}_k \equiv \mathbb{Q}_k r_k$, the associated search direction $\mathbf{u}_k$ and approximation $\mathbf{x}_k$ by appropriate recursions. As (72) and (73) show, one can use these vectors to compute the Bi-CG scalars β_k and α_k. If the polynomials q_k are related by simple recursions, these recursions can be used to compute efficiently the iterates $\mathbf{r}_k$, $\mathbf{u}_k$ and $\mathbf{x}_k$ without computing r_k and u_k explicitly. Therefore, the computational effort of the Bi-CG algorithm to build the shadow Krylov subspace can also be used to obtain an additional reduction (by $\mathbb{Q}_k$) for the Bi-CG residual r_k. Since r_{2k} would have been constructed from the (quasi-) best polynomial in $\mathcal{P}^1_{2k}$, we may not expect that $\|\mathbf{r}_k\| \ll \|r_{2k}\|$ (in exact arithmetic): which says that a method based on $\mathbb{Q}_k$ can only converge twice as fast (in terms of costs). Since a Bi-CG step involves two Mv's it only makes sense to compute $\mathbf{r}_k$ instead of r_k if we can obtain $\mathbf{r}_k$ from $\mathbf{r}_{k-1}$ by 4 Mv at most and a few vector updates and if we can update simultaneously the corresponding approximation $\mathbf{x}_k$, where $\mathbf{r}_k = b - A\mathbf{x}_k$. This has been realized in CG-S, Bi-CGSTAB and Bi-CGstab(l) (these algorithms require 2 Mv's per step).

We give details on Bi-CGSTAB: it is a very attractive algorithm that can easily be presented now.

For a sequence of scalars ω_k we consider the polynomials q_k given by ($q_0 \equiv 1$)

$$q_k(t) = (1 - \omega_{k-1}t)q_{k-1}(t) \quad (t \in \mathbb{R}) \quad \text{for all} \quad k. \tag{74}$$

Since $\mathbb{A}\mathbb{Q}_k K u_k = \mathbb{Q}_k c_k$, the definition $\mathbf{u}_k \equiv K^{-1}\mathbb{Q}_k K u_k$ yields $A\mathbf{u}_k = \mathbf{c}_k$. A combination of the relation $\mathbb{Q}_k = \mathbb{Q}_{k-1} - \omega_{k-1}\mathbb{A}\mathbb{Q}_{k-1}$ (for k and $k+1$; from (74)) with the Bi-CG relations $K u_k = r_k - \beta_k K u_{k-1}$ and $r_{k+1} = r_k - \alpha_k c_k$ gives

$$\begin{aligned}
K\mathbf{u}_k = \mathbb{Q}_k K u_k &= \mathbb{Q}_k r_k - \beta_k \mathbb{Q}_{k-1} K u_{k-1} + \beta_k \omega_{k-1} \mathbb{A}\mathbb{Q}_{k-1} K u_{k-1} \\
&= \mathbf{r}_k - \beta_k K \mathbf{u}_{k-1} + \beta_k \omega_{k-1} \mathbf{c}_{k-1} \\
\mathbf{r}_{k+1} = Q_{k+1} r_{k+1} &= \mathbb{Q}_k r_k - \alpha_k \mathbb{Q}_k c_k - \omega_k \mathbb{A}(\mathbb{Q}_k r_k - \alpha_k \mathbb{Q}_k c_k) \\
&= \mathbf{r}_k - \alpha_k \mathbf{c}_k - \omega_k \mathbb{A}(\mathbf{r}_k - \alpha_k \mathbf{c}_k).
\end{aligned}$$

Since the updates $\mathbf{c}_k$ and $\mathbb{A}(\mathbf{r}_k - \alpha_k \mathbf{c}_k)$ of $\mathbf{r}_k$ have an original $\mathbf{u}_k$ and $K^{-1}(\mathbf{r}_k - \alpha_k \mathbf{c}_k)$, respectively, under A, it is clear how to update $\mathbf{x}_k$. Leaves us to specify how to choose the ω_k. If we minimize $\|\mathbf{r}_{k+1}\|$ as in LMR then we have **Bi-CGSTAB** (the stabilized version of CG-S) [44] in ALGORITHM 7.

Each step of the algorithm requires two multiplications by $\mathbb{A}$ and no multiplication by $\mathbb{A}^T$. Apart from these Mv's, each step is only slightly more expensive than a Bi-CG step. Moreover, Bi-CGSTAB produces a residual $\mathbf{r}_k$ in a Krylov subspace of double dimension and it often finds a very small residual in that larger space: as we learned from experiments, $\|\mathbf{r}_k\| \ll \|r_k\|$ for many problems.

Similarly, one can deduce algorithms for other choices of the polynomials q_k. If p_k is the Bi-CG polynomial then the choice $q_k = p_k$ yields **CG-S** (**Conjugate Gradient-Squared**)

$$
\boxed{\begin{array}{l}
\texttt{choose } \mathbf{x}_0 \texttt{ and } \tilde{r}_0, \\
k=-1,\ \mathbf{r}_0 = b - A\mathbf{x}_0,\ \mathbf{u}_{-1} = \mathbf{c}_{-1} = 0,\ \alpha_{-1} = 0,\ \omega_{-1} = \rho'_{-1} = 1, \\
\texttt{repeat until } \|\mathbf{r}_{k+1}\| \texttt{ is small enough:} \\
\qquad \left\{ \begin{array}{l}
k := k+1, \\
\rho'_k = (\mathbf{r}_k, \tilde{r}_0),\ \ \beta_k = -(\alpha_{k-1}/\omega_{k-1})(\rho'_k/\rho'_{k-1}), \\
\mathbf{u}_k = -\beta_k \mathbf{u}_{k-1} + K^{-1}(\mathbf{r}_k + \beta_k \omega_{k-1}\mathbf{c}_{k-1}), \quad \mathbf{c}_k = A\mathbf{u}_k, \\
\gamma'_k = (\mathbf{c}_k, \tilde{r}_0),\ \ \alpha_k = \rho'_k/\gamma'_k, \\
\mathbf{r}_{k+\frac{1}{2}} = \mathbf{r}_k - \alpha_k \mathbf{c}_k, \quad \mathbf{u}_{k+\frac{1}{2}} = K^{-1}\mathbf{r}_{k+\frac{1}{2}}, \quad \mathbf{c}_{k+\frac{1}{2}} = A\mathbf{u}_{k+\frac{1}{2}}, \\
\omega_k = (\mathbf{r}_{k+\frac{1}{2}}, \mathbf{c}_{k+\frac{1}{2}})/(\mathbf{c}_{k+\frac{1}{2}}, \mathbf{c}_{k+\frac{1}{2}}), \\
\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{u}_k + \omega_k \mathbf{u}_{k+\frac{1}{2}}, \quad \mathbf{r}_{k+1} = \mathbf{r}_{k+\frac{1}{2}} - \omega_k \mathbf{c}_{k+\frac{1}{2}}
\end{array} \right.
\end{array}}
$$

ALGORITHM 7: The Bi-CGSTAB algorithm.

[38]. Bi-CG gives simple recurrence relations for the p_k leading to simple recurrence relations for CG-S. $p_k(\mathbb{A})$ is the (quasi-)best reductor for r_0 of the given form. In cases where $p_k(\mathbb{A})$ also reduces r_k CG-S is an attractive method. But, since it is not constructed for r_k, we may not expect that it reduces r_k equally well. As experiments show there are situations in which $p_k(\mathbb{A})$ even amplifies r_k. This causes irregular convergence or even divergence and makes the method more sensitive to evaluation error [43] (if $\|\mathbf{r}_{k+1}\|$ is small while $\|\mathbf{r}_k\|$ is large then $\mathbf{r}_{k+1}$ must have been the difference of large vectors. Such a situation is a source of large evaluation errors). Bi-CGSTAB converges more smoothly. Consequently, it is less sensitive to evaluation errors and converges faster than Bi-CG and CG-S. Actually, Bi-CGSTAB was designed to "smooth" the erratic convergence behavior of CG-S.

BiCGstab(l) [33] takes for q_k products of MR polynomials of degree l, thus generalizing Bi-CGSTAB that uses degree one MR polynomials. For $l > 1$ (say $l = 2$ or $l = 4$) this is a competitive algorithm, especially for problems involving matrices with complex spectrum as arise from discretized advection-diffusion equations [33]. For these problems, Bi-CGSTAB often performs poorly [22]: eigenvalues with relatively large imaginary part slow down the convergence of Bi-CGSTAB or may lead to stagnation due the fact that the ω_k are real [33] (even in complex arithmetic).

These hybrid Bi-CG methods compute explicitly the Bi-CG coefficients α_k and β_k and, as in the Bi-CG case, we can cheaply find approximations for the extremal eigenvalues of $\mathbb{A}$ as a side product (see 7.5).

The QMR algorithm also has a "transpose free" version [14] (i.e. the algorithm does not use multiplications by A^T) and Bi-CGSTAB has a minimal residual version [12]; we will not discuss these algorithms here.

9 Numerical examples

So far we did not comment on the choice for the initial approximation x_0.
One often takes $x_0 = 0$. However, when solving time-dependent PDE's the solution from the previous time level may be a more educated guess. If the PDE is autonomous then the orthogonal basis vectors for a Krylov subspace (and the associated search directions) on one time level can be used to improve the initial guess for the next time level (in case GCR was used, take $x_0 + Q_k r_0$ instead of x_0, where the u_j and c_j that define Q_k stem from the previous time level and x_0 and r_0 are the present initial guess and associated residual). Actually, this comes down to an update of the preconditioner ($K^{-1} + Q_k(I - AK^{-1})$ instead of K^{-1}, see 4.3) that can also be used if the PDE is non-autonomous.

In this section we will discuss some numerical experiments. These experiments are intended to show some characteristic behavior of some of the methods discussed in this paper. We do not pretend that the problems are solved in the best possible way. For instance, in some experiments we used a preconditioner, whereas in others we did not. A suitable preconditioner can improve the speed of convergence of any of the methods, but this is not the point we would like to make.

All partial differential equations were discretized with finite volumes discretization. With exception of the first and the third example, we took $x_0 \equiv 0$ as an initial guess. However, in our experiments the convergence history was mainly determined by the preconditioned matrix $\mathbb{A}$ (in line with our observations in 4.1). The choice of x_0 (or, equivalently, of the solution or inhomogeneous term) did not have much influence. Other choices postponed phenomena by only a few iteration steps, but they gave approximately the same picture. The experiments were done in double precision on a SUN SPARC 670 MP. Since we are interested in the performance of the iterative solution methods we did not try to find the most optimal stopping criterion. For a discussion about more appropriate stopping criteria we refer to [1]. We stopped when the residual reduction was less than 10^{-9}, i.e. $\|r_k\|_2 \leq 10^{-9}\|r_0\|_2$, or when the number of matrix multiplications exceeded 1000: the size of the error in the final approximation may be much smaller than the size of the discretization error. The figures show the convergence behavior of the iterative methods. Horizontally the number of matrix multiplications is counted. At the end of this section we give in TABLE 1 an overview of the required CPU-time for the norm of the *true* residual $b - Ax_k$ for several iterative methods. The numbers between brackets () are the log of the norm of the final *true* residuals: ${}^{10}\log(\|b - Ax_k\|_2)$. The log of the norm of the computed updated residuals can be seen from the figures. A '*' in TABLE 1 indicates that the method did not meet the required tolerance before 1000 multiplications with the matrix A. We did our experiments for GMRES(25), GMRESR(10,5), Bi-CG, Bi-CGSTAB and BiCGstab(2). The first two algorithms have the same memory requirements.

9.1 Numerical example 1; the Molenkamp test problem

First we consider the Molenkamp problem:

$$\frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} + v\frac{\partial c}{\partial y} = 0, \qquad u = -2\pi y, \qquad v = 2\pi x$$

for $(x, y) \in [-1, +1] \times [-1, +1]$ and $t \geq 0$ with initial condition

$$c(x, y, 0) = 10^{-8\left((x+\frac{1}{2})^2+y^2\right)}$$

and boundary conditions

$$\begin{aligned} c(\pm 1, y, t) &= 10^{-8\left((\pm 1+\frac{1}{2}\cos 2\pi t)^2+(y+\frac{1}{2}\sin 2\pi t)^2\right)} \\ c(x, \pm 1, t) &= 10^{-8\left((x+\frac{1}{2}\cos 2\pi t)^2+(\pm 1+\frac{1}{2}\sin 2\pi t)^2\right)}. \end{aligned}$$

We discretized the spatial derivatives by finite volumes using (81×81) volumes. For time integration, we used the trapezoidal rule with $\Delta t = 1/500$ (leading to the Crank-Nicolson scheme).

Figure 1 shows the convergence histories to compute an accurate approximation on time level $t_1 = \Delta t$. For x_0 we took the vector given by the initial condition. Other time levels show comparable pictures if we take the solution of the preceding time level as initial guess.

GMRES(25) meets the required accuracy in 6 iteration steps. Hence it does not restart and its residuals are the best possible, i.e. they are the ones with minimal norm in the Krylov subspaces. From Figure 1 we see that the other methods find equally good approximations in the same Krylov subspaces (recall that Bi-CG needs two Mv's to increase the dimension of the Krylov space by one). Since Bi-CGSTAB is on average per Mv the cheapest method, it appears

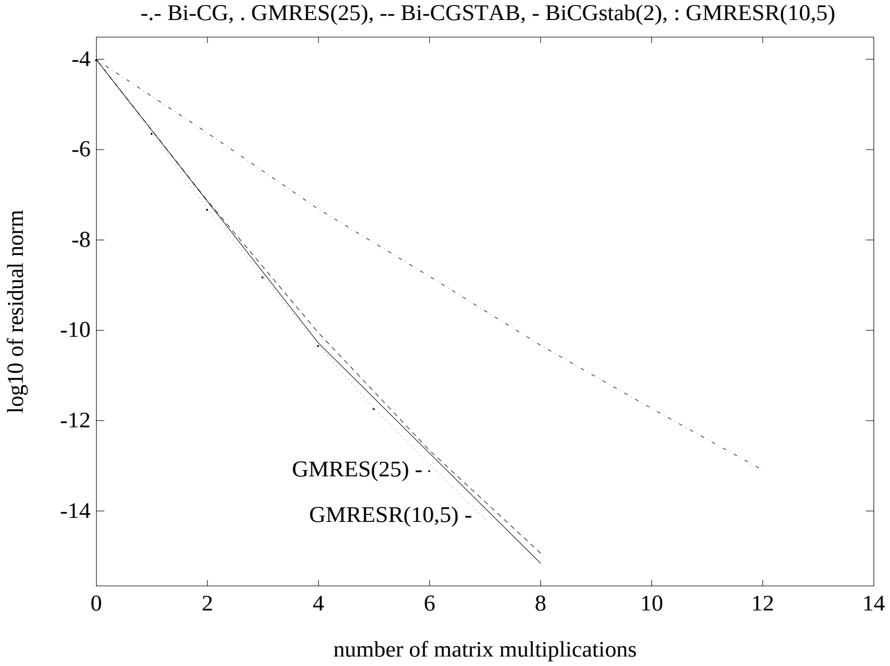

FIGURE 1: Convergence plot of numerical example 9.1.

to be the best method for this example. However, the differences are small since the additional costs (additional to the Mv's) can hardly grow in six iteration steps.

On the subsequential time levels, we did not try to use the basis of the Krylov subspace and the Hessenberg matrix as computed by GMRES(25), although this would make GMRES(25) very cheap.

Surprisingly as it may look, it often occurs that the Krylov subspace methods perform equally well in the first few iteration steps: the fact that Bi-CG uses skew projections and the BiCGstab methods combine such skew projections with local minimalization leads to deviation from the optimal approach only after a few iteration steps. But here, we only needed a few steps to find an accurate approximation. In the next examples, the methods converge slower, due to a less favorable eigenvalue distribution, need more steps, and show completely different convergence behaviors already after a few steps.

9.2 Numerical example 2

Next we give an example where Bi-CG and the BiCGstab methods perform better than the methods based on the MR approach.

The symmetric positive definite linear system stems from an (81×81) discretization of

$$-\frac{\partial}{\partial x} D \frac{\partial c}{\partial x} - \frac{\partial}{\partial y} D \frac{\partial c}{\partial y} = 1, \tag{75}$$

over the unit square, with Dirichlet boundary conditions along $y = 0$ and Neumann conditions along the other parts of the boundary. The function D is defined as

$$D = 1000 \;\text{ for }\; 0.1 \leq x, y \leq 0.9 \;\text{ and }\; D = 1 \;\text{ elsewhere.}$$

ILU [23] was used as preconditioner. This example was taken from [44]. A convergence plot is given in FIGURE 2.

In the first few iteration steps all the methods perform equally well. None of the methods makes much progress in the steps 30–75. For GMRES(25) this is fatal: due to the restart it does

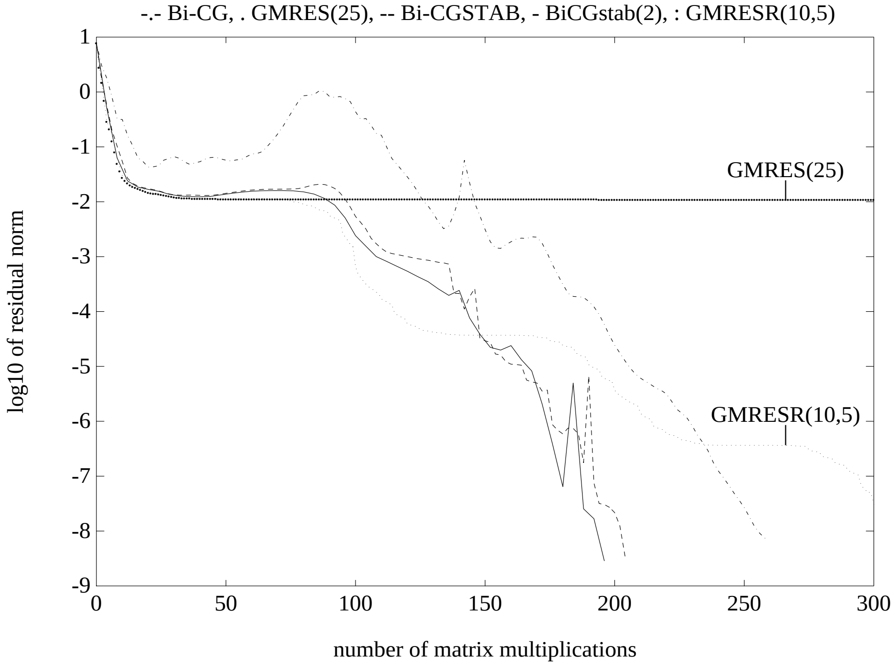

FIGURE 2: Convergence plot for numerical example 9.2.

not get beyond the point of stagnation. In view of the convergence history of GMRES(10,5), we expect GMRES(50) to do much better, although we did not try. In general, it is hard to select the proper m: for instance, GMRES(40) may stagnate while GMRES(41) may converge nicely. The selection of l and m for the best GMRESR(m, l) is equally difficult. Here, we see that GMRESR(10,5) performs well. The fact that the method is restarted at each 50-th matrix multiplication is reflected in the periodic stagnation phase.

The Bi-CG methods do not restart and show some acceleration. The BiCGstab methods are faster than Bi-CG (although not twice as fast). BiCGstab(2) is slightly faster than Bi-CGSTAB, but not fast enough to compensate for the more expensive steps (cf. Table 1). Bi-CGSTAB is the best method here, but BiCGstab(2) is comparable.

9.3 Numerical example 3

In our third example we consider an advection dominated 2-nd order partial differential equation, with Dirichlet boundary conditions, on the unit square:

$$-\frac{\partial^2 c}{\partial x^2} - \frac{\partial^2 c}{\partial y^2} + a\frac{\partial c}{\partial x} + 100\frac{\partial c}{\partial y} = F, \qquad (76)$$

where $a = 100$ for $x \in [0, \frac{1}{4}] \cup [\frac{1}{2}, \frac{3}{4}]$ and $a = -100$ otherwise. The function F is defined by the solution $c(x, y) = \sin(\pi x)\sin(\pi y)$. This equation was discretized using (81×81) volumes, resulting in a five-diagonal linear system of order 6600. We took a random initial guess x_0 and we did not use a preconditioner.

The convergence histories for this example show similarity to the ones in the previous example (though Bi-CG converges more irregularly). One often sees that Krylov subspace methods stagnate in the first phase of the iteration process when applied to problems arising from advection-diffusion PDE's with strongly varying diffusion coefficients or with large advection terms. Often GMRES(m) requires a large m to overcome the stagnation phase.

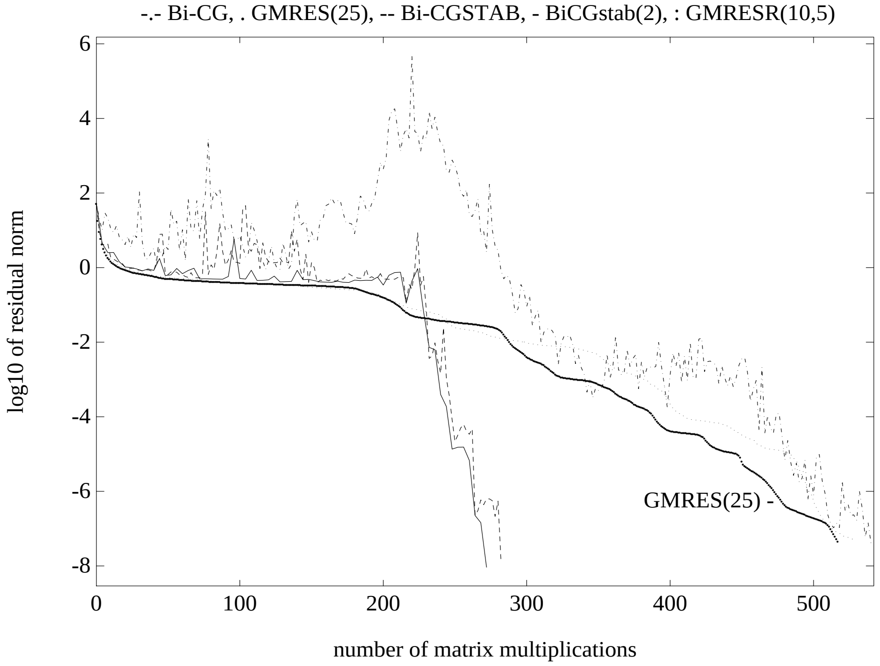

FIGURE 3: Convergence plot of numerical example 9.3.

9.4 Numerical example 4

In our fourth example we consider an advection dominated 2-nd order partial differential equation, with Dirichlet boundary conditions, on the unit cube (this equation was taken from [22]):

$$\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2} + 1000\frac{\partial c}{\partial x} = F. \tag{77}$$

The function F is defined by the solution $c(x, y, z) = xyz(1 - x)(1 - y)(1 - z)$. This equation was discretized using $(22 \times 22 \times 22)$ volumes, resulting in a seven-diagonal linear system of order 11000. No preconditioning was used.

In FIGURE 4 we see a plot of the convergence history. Bi-CGSTAB stagnates as might be anticipated from the fact that this linear system has large complex eigenpairs. Surprisingly, Bi-CGSTAB does even worse than Bi-CG. For this type of matrices this behavior of Bi-CGSTAB is not uncommon and might be explained by the poor first degree minimal residual reductions. In that case the Bi-CG iteration coefficients α_k and β_k are not recovered very well. BiCGstab(2) converges quite nicely and almost twice as fast as Bi-CG. Bi-CG loses bi-orthogonality among the residuals in a very early phase. This is also the case for the Bi-CG method that underlies BiCGstab(2). This explains why these methods do not accelerate (in contrast to what might be expected). But apparently BiCGstab(2) has less problems than Bi-CG.

GMRES(25) and GMRESR(10,5) converged nicely but not quite as fast as BiCGstab(2). Since their steps are also more expensive, BiCGstab(2) is the best method here (cf. Table 1).

9.5 Numerical example 5

Our last example shows that there is no guarantee that success is not ensured. For none of the five test methods we found an accurate approximation: Bi-CGSTAB broke down while the other four methods stagnated on a residual reduction level 10^{-1}.

The nonsymmetric linear system comes from a $(22 \times 22 \times 22)$ finite volume discretization of

$$\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} + \frac{\partial^2 c}{\partial z^2} + 10^5 x^2 (\frac{\partial c}{\partial x} + \frac{\partial c}{\partial y} + \frac{\partial c}{\partial z}) = F, \tag{78}$$

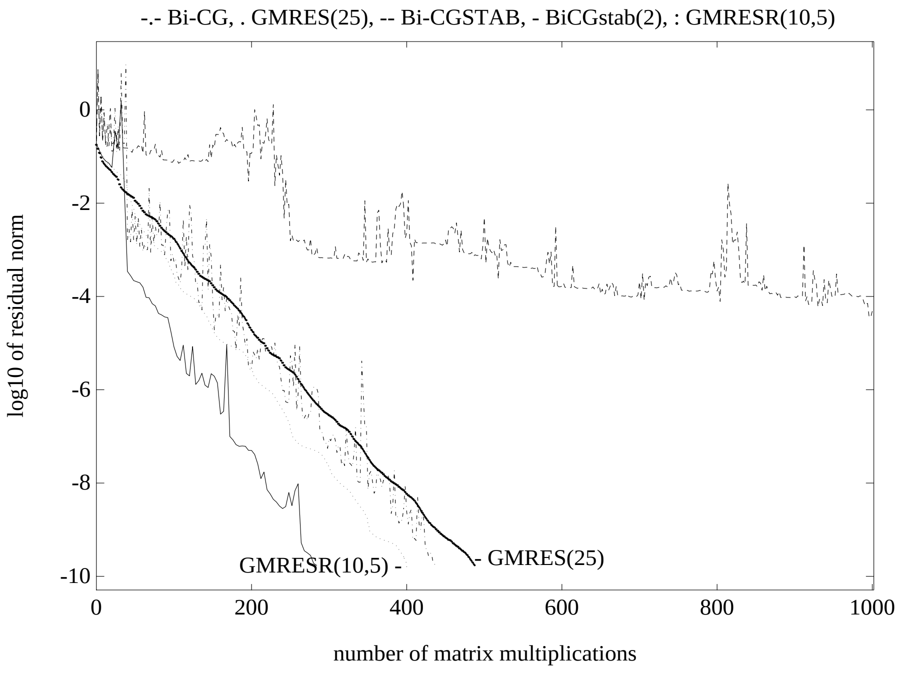

FIGURE 4: Convergence plot of numerical example 9.4.

METHOD	Example 1	Example 2	Example 3	Example 4
GMRES(25)	1.07 (-13.0)	stagnation	91.4 (-7.3)	78.9 (-9.8)
GMRESR(10,5)	1.11 (-14.2)	41.9 (-8.2)	53.2 (-7.3)	59.4 (-9.8)
Bi-CG	1.15 (-13.0)	22.1 (-8.1)	29.1 (-7.4)	34.8 (-9.7)
Bi-CGSTAB	0.82 (-14.9)	17.8 (-8.5)	15.6 (-7.8)	87.7 (-4.3) *
BiCGstab(2)	1.00 (-15.2)	19.4 (-8.5)	17.3 (-8.0)	28.8 (-9.8)

TABLE 1: CPU-time and log10 of the *true* residual norm (see the introduction of **9**).

on the unit cube with Dirichlet boundary conditions (this equation was taken from [22]). The function F is defined by the solution $c(x,y,z) = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$. We did not use any preconditioning.

We also tried other methods, other parameter values and a number of standard preconditioning strategies. None of our attempts was successful, but we did not try to find a preconditioner adjusted to the stream lines.

9.6 Conclusions

There is no method that solves (quickly and) accurately any linear equation $Ax = b$. However, the class of Krylov subspace methods and its hybrids contain very attractive algorithms. As a side product, they can provide useful information about the spectrum of $\mathbb{A}$. If a problem requires a large m to get GMRES(m) to converge and Bi-CG and CG-S stagnate, methods as restarted GMRESR(m,l) (with $l = 20$, $m = 10$?) or BiCGstab(l) (with $l = 1,2,4$) are often very attractive alternatives. Especially, BiCGstab(2) (and BiCGstab(4)) seems to be a very promising competitive algorithm to solve non-symmetric linear systems as arising from discretized advection diffusion PDE's.

Acknowledgement We are grateful to Diederik Fokkema for providing the numerical material.

References

[1] ARIOLI M., I. DUFF AND D. RUIZ: Stopping criteria for iterative solvers. *SIAM J. Matrix Anal. Appl.*, **13** (1992), pp. 138–144.

[2] AXELSSON O.: On preconditioning and convergence acceleration in sparse matrix problems. *Rep. CERN 74-10*, (Geneve, 1974).

[3] AXELSSON O.: Solution of linear systems of equations: iterative methods. In V. A. Barker, editor, *Sparse Matrix Techniques*, (Copenhagen 1976, Springer Verlag, Berlin, 1977).

[4] AXELSSON O. AND P.S. VASSILEVSKI: A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning. *SIAM J. Matrix Anal. Appl.*, **12** (1991), pp. 625–644.

[5] BAI Z., D. HU AND L. REICHEL: A Newton basis GMRES implementation. *Tech. Report 91–03* (Unviversity of Kentucky, 1991).

[6] DE STURLER E. AND D.R. FOKKEMA: Nested Krylov methods and preserving the orthogonality. *Preprint 796* (Dep. Math., University of Utrecht, 1993). *Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods* (NASA CP, 1993)

[7] EIROLA T. AND O. NEVANLINNA: Accelerating with rank-one updates. *Linear Algebra Appl.*, **121** (1989), pp. 511–520.

[8] EISENSTAT S.C., H.C. ELMAN AND M.H. SCHULTZ: Varational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, **20** (1983), pp. 245–357.

[9] ELMAN H.C.: Iterative methods for large, sparse, nonsymmetric systems of linear equations. *PhD thesis and Res. Rep. # 229* (Dept. of Comp. Sci., Yale U., 1982).

[10] FISCHER B. AND R.W. FREUND: On the constrained Chebyshev approximation problem on ellipses. *J. Approx. Theory*, **62** (1990), pp. 297–315.

[11] FLETCHER R.: Conjugate gradient methods for indefinite systems. In G. Watson, ed., *Proc. of the Dundee Biennial Conference on Numerical Analysis* (Springer-Verlag, New York, 1975).

[12] FREUND R.W.: A Transpose-Free Quasi-Minimal Residual Algorithm for non-Hermitian Linear Systems. *SIAM J. Sci. Stat. Comput.*, to appear.

[13] FREUND R.W. AND N.M. NACHTIGAL.: QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, **60** (1991), pp. 315–339.

[14] FREUND R.W. AND T. SZETO: A Quasi-Minimal Residual Squared Algorithm for non-Hermitian Linear Systems. In *Proc. of the Copper Mountain Conference on Iterative Methods, Vol I* (1992).

[15] GOLUB G.H. AND C.F. VAN LOAN: *Matrix Computations* (Second edition, The John Hopkins University Press, Baltimore and London, 1989).

[16] GREENBAUM A.: Comparison of splittings used with the conjugate gradient algorithm. *Numer. Math.*, **33** (1979), pp. 181–194.

[17] GREENBAUM. A: Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Linear Algebra Appl.*, 113:7–63, 1989.

[18] GREENBAUM A. AND Z. STRAKOS: Matrices that generate the same Krylov residual spaces. *IMA Preprint Series # 983* (IMA, University of Minnesota, Minneapolis, 1992).

[19] HESTENES M.R. AND E. STIEFEL: Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, **49** (1954), pp. 409–436.

[20] KHARCHENKO S.A. AND A.YU. YEREMIN: Eigenvalue translation based preconditioners for the GMRES(k) method. *Research Report EM-RR 2/92* (Elegant Math. Inc., 1992).

[21] LANCZOS C.: Solution of systems of linear equations by minimized iteration. *J. Res. Nat. Bur. Stand.*, **49** (1952), pp. 33–53.

[22] Meier Yang U.: Preconditioned Conjugate Gradient-Like methods for Nonsymmetric Linear Systems. *Preprint, Center for Research and Development* (University of Illinois at Urbana-Champaign, 1992).

[23] Meijerink J.A. and H.A. van der Vorst: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *ACCU Report*, (Utrecht, 1974), *Math. Comput.*, **31** (1977), pp. 148–162.

[24] Meijerink J.A. and H.A. van der Vorst: Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. *J. Comp. Phys.*, **44** (1981), pp. 134–155.

[25] Nachtigal N.M., S.C. Reddy and L.N. Trefethen: How fast are nonsymmetric matrix iterations? to appear *SIAM J. Sci. Stat. Comput.*.

[26] Paige C.C.: Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem. *Linear Algebra Appl.*, **33** (1980), pp. 235–258, 1980.

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

[28] Paige C.C. and M. A. Saunders: LSQR, an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, **8** (1982), pp. 43–71.

[29] Reid J.K.: On the method of conjugate gradients for the solution of large sparse systems of linear equations. *Proc. Conf. on large sparse sets of linear equations*, (Academic Press, New York, 1971).

[30] Saad Y.: Krylov subspace method for solving large unsymmetric linear systems. *Math. Comput.*, **37** (1981), pp. 105–126.

[31] Saad Y.: A flexible inner-outer preconditioned GMRES algorithm. *Tech. Report UMSI 91/279* (University of Minnesota Supercomputer Institute, Minneapolis, Minnesota, 1991).

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

[33] Sleijpen G.L.G. and D.R. Fokkema: BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum. *Preprint 772* (Dep. Math., University of Utrecht, 1993).

[34] Sleijpen G.L.G. and H.A. van der Vorst: Optimal iteration methods for large linear systems of equations. in *Proceedings of the seminar "Numerical Advection Difussion", Utrecht 1991–1992*, C.B. Vreugdenhil and B. Koren, eds., (Notes on Numerical Fluid Mechanics, Vieweg Verlag, Braunschweig, 1993).

[35] Sleijpen G.L.G. and A. van der Sluis: Further results on the convergence behaviour of CG and Ritz values. *Preprint 677* (Dep. Math., University of Utrecht, 1991).

[36] van der Sluis A.: The convergence behaviour of conjugate gradients and Ritz values in various circumstances. in *Iterative methods in linear algebra*, R. Beauwens and P. de Groen, eds. (North Holland, Amsterdam, London, New York, Tokyo, 1992).

[37] van der Sluis A. and H. A. van der Vorst: The rate of convergence of conjugate gradients. *Numer. Math.*, **48** (1986), pp. 543–560.

[38] Sonneveld P.: CGS, a fast Lanczos-type solver for nonsymetric linear systems. *SIAM J. Sci. Stat. Comput.*, **10** (1989), pp. 36–52.

[39] Trefethen L.N.: *Non-normal matrices and pseudo-eigenvalues in numerical analysis.* Book in preparation

[40] Varga R.S.: *Matrix Iterative Analysis* (Prentice-Hall, Englewood Cliffs N.J., 1962).

[41] Vinsome P.K.W.: Orthomin, an iterative method for solving sparse sets of simultaneous linear equations. Paper SPE 5729, 4th *Symposium of Numerical Simulation of Reservoir Performance of the Society of Petrolium Engineers of the AIME*, (Los Angeles, 1976).

[42] van der Vorst H.A. and G.L.G. Sleijpen: The effect of incomplete decomposition preconditioning on the convergence of conjugate gradients. *Incomplete Decompositions, Proceedings of the Eigth GAMM Seminar* (Vieweg Verlag, Braunschweig, 1992).

[43] van der Vorst H.A.: The convergence behavior of preconditioned CG and CG-S in the presence of rounding errors, *Lecture Notes in Math.*, **1457** (Springer, Berlin, 1990), 126–136.

[44] van der Vorst H.A.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, **13** (1992), pp. 631–644.

[45] van der Vorst H.A. and C. Vuik: The rate of convergence of GMRES. *Preprint* (Dep. Math., RUU, 1990).

[46] van der Vorst H.A. and C. Vuik: GMRESR: A family of nested GMRES methods. *Tech. Report 91–80* (Delft University of Technology, Faculty of Tech. Math., Delft, 1991).

[47] Young D.M. and K.C. Jea: Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods. *Linear Algebra Appl.*, **34** (1980), pp. 159–194.