

# BLOCK KRYLOV SPACE METHODS FOR LINEAR SYSTEMS WITH MULTIPLE RIGHT-HAND SIDES: AN INTRODUCTION

MARTIN H. GUTKNECHT / VERSION OF February 28, 2006\*

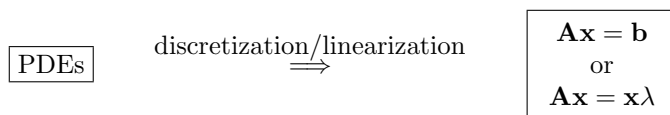
**Abstract.** In a number of applications in scientific computing and engineering one has to solve huge sparse linear systems of equations with several right-hand sides that are given at once. Block Krylov space solvers are iterative methods that are especially designed for such problems and have fundamental advantages over the corresponding methods for systems with a single right-hand side: much larger search spaces and simultaneously computable matrix-vector products. But there are inherent difficulties that make their implementation and their application challenging: the possible linear dependence of the various residuals, the much larger memory requirements, and the drastic increase of the size of certain small (“scalar”) auxiliary problems that have to be solved in each step.

We start this paper with a quick introduction into sparse linear systems, their solution by sparse direct and iterative methods, Krylov subspaces, and preconditioning. Then we turn to systems with multiple right-hand sides, block Krylov space solvers, and the deflation of right-hand sides in the case where the residuals are linearly dependent. As a model we discuss in particular a block version of the General Minimum Residual (GMRES) algorithm.

**Key words.** sparse linear systems, multiple right-hand sides, several right-hand sides, block Krylov space method, generalized minimal residual method, GMRES, block size reduction, deflation

**AMS subject classifications.** 65F10

**1. Large sparse linear systems.** Large sparse linear systems of equations or sparse matrix eigenvalue problems appear in most applications of scientific computing. In particular, the discretization of partial differential equations with the finite element method (FEM) or with the (older) finite difference method (FDM) leads to such problems:



“Sparse” refers to  $\mathbf{A}$  being a **sparse matrix** and means that most of the elements of  $\mathbf{A}$  are 0. In a simple, physically one-dimensional (1D) problem, there may be as few as three nonzeros per row; in a complex, physically three-dimensional (3D) problem, there may be as many as a hundred. Often, when nonlinear PDEs are solved, most computer time is spent for repeatedly solving such linear systems or eigenvalue problems.

While in 1950 “large” meant that  $\mathbf{A}$  had order  $N$  with, say,  $30 < N < 100$ , nowadays it means something like  $1'000 \leq N \leq 100'000'000$ . These sparse matrices need to be stored in appropriate data formats that avoid to store the zero elements. There are many different such formats; see, e.g., [23], Sect. 3.4. Examples of sparse matrices have been collected for numerical experiments. The two best known such collections are:

- *Matrix Market:*  
<http://math.nist.gov/MatrixMarket>
- *University of Florida Sparse Matrix Collection (Tim Davis):*  
<http://www.cise.ufl.edu/research/sparse/matrices>

---

\*Seminar for Applied Mathematics, ETH-Zurich, CH-8092 Zurich, Switzerland; Email: [mhg@math.ethz.ch](mailto:mhg@math.ethz.ch), URL: <http://www.sam.math.ethz.ch/~mhg>.

In addition to PDE based problems one encounters nowadays many other sparse matrix problems, and the nonzero pattern of their matrices may differ considerably from those generated by the finite element or the finite difference method.

We will concentrate here on linear systems of equations, but some of the methods we mention have analogs that are used to compute eigenvalues.

“Small” linear systems are normally solved with some version of Gaussian elimination, which leads to a factorization of  $\mathbf{A}$  with permuted rows (equations) into a product of a lower and an upper tridiagonal matrix:  $\mathbf{PA} = \mathbf{LU}$ . In the last twenty years enormous efforts have been made to develop algorithms of this type that maintain the stability of the Gauss algorithm with pivoting, but are adapted to large sparse matrices by avoiding much **fill-in** (i.e.,  $\mathbf{L} + \mathbf{U}$  have not many more nonzeros than  $\mathbf{A}$ ) and are at the same time optimized for the architecture of modern high-performance computers, that is, vectorized or parallelized. These algorithms are classified as **sparse direct methods**, see, e.g., [11, 16].

**2. Krylov space solvers.** While direct methods provide us after  $N - 1$  elimination steps and backward substitution with the exact solution vector  $\mathbf{x}_*$  of  $\mathbf{Ax} = \mathbf{b}$  up to roundoff errors, iterative methods generate a sequence of **iterates**  $\mathbf{x}_n$  that are approximate solutions, and hopefully converge quickly towards the exact solution  $\mathbf{x}_*$ . Many methods actually produce in exact arithmetic the exact solution in at most  $N$  steps.

Iterative methods include such classics as the Gauss-Seidel method, but except for multigrid, most methods that are used nowadays are **Krylov space solvers**. This means that  $\mathbf{x}_n$  belongs to an affine space that grows with  $n$ :

$$\mathbf{x}_n - \mathbf{x}_0 \in \mathcal{K}_n := \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0) := \text{span}(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^{n-1}\mathbf{r}_0) \subset \mathbb{C}^N,$$

where  $\mathbf{r}_0 := \mathbf{b} - \mathbf{Ax}_0$  is the **initial residual** and  $\mathcal{K}_n$  is the  $n$ th **Krylov subspace** generated by  $\mathbf{A}$  from  $\mathbf{r}_0$ .

Among the most famous Krylov (sub)space solvers are

- for Hermitian positive definite (Hpd)  $\mathbf{A}$ : the conjugate gradient (CG) method,
- for Hermitian indefinite  $\mathbf{A}$ : the minimal residual (MINRES) method, certain versions of the conjugate residual (CR) method
- for non-Hermitian  $\mathbf{A}$ : the generalized minimal residual (GMRES) method, the biconjugate gradient (BiCG) method, the stabilized biconjugate gradient squared (BiCGSTAB) method, the quasi-minimal residual (QMR) method, and the transpose-free quasi-minimal residual (TFQMR) method.

A large number of Krylov space solvers have been proposed, in total perhaps more than 100. There are some, where  $\mathbf{x}_n$  may not exist for some exceptional values of  $n$ .

There are also related iterative methods for finding some of the eigenvalues of a large sparse matrix. In theory they allow us to find all the distinct eigenvalues and corresponding one-dimensional eigenspaces.

What can we say about the dimension of Krylov subspaces, and why are they so suitable for approximating the solution of the linear system? Clearly, by definition,  $\mathcal{K}_n$  can have at most dimension  $n$ . However, one can say more. The following two lemmas and their two corollaries are easy to prove. We leave the proofs to the readers.

LEMMA 1. *There is a positive integer  $\bar{\nu} := \bar{\nu}(\mathbf{y}, \mathbf{A})$  such that*

$$\dim \mathcal{K}_n(\mathbf{A}, \mathbf{y}) = \begin{cases} n & \text{if } n \leq \bar{\nu}, \\ \bar{\nu} & \text{if } n \geq \bar{\nu}. \end{cases}$$

*The inequalities  $1 \leq \bar{\nu} \leq N$  hold, and  $\bar{\nu} < N$  is possible if  $N > 1$ .*

DEFINITION. The positive integer  $\bar{\nu} := \bar{\nu}(\mathbf{y}, \mathbf{A})$  of Lemma 1 is called **grade of  $\mathbf{y}$  with respect to  $\mathbf{A}$** .  $\blacktriangle$

Lemma 1 provides us with a characterization of  $\bar{\nu}(\mathbf{y}, \mathbf{A})$ :

COROLLARY 2. *The grade  $\bar{\nu}(\mathbf{y}, \mathbf{A})$  satisfies*

$$\begin{aligned} \bar{\nu}(\mathbf{y}, \mathbf{A}) &= \min \{n \mid \dim \mathcal{K}_n(\mathbf{A}, \mathbf{y}) = \dim \mathcal{K}_{n+1}(\mathbf{A}, \mathbf{y})\} \\ &= \min \{n \mid \mathcal{K}_n(\mathbf{A}, \mathbf{y}) = \mathcal{K}_{n+1}(\mathbf{A}, \mathbf{y})\}. \end{aligned}$$

There are further characterizations of the grade: the grade is the dimension of the smallest  $\mathbf{A}$ -invariant subspace that contains  $\mathbf{y}$ , and the following Lemma holds.

LEMMA 3. *The grade  $\bar{\nu}(\mathbf{y}, \mathbf{A})$  satisfies*

$$\bar{\nu}(\mathbf{y}, \mathbf{A}) = \min \{n \mid \mathbf{A}^{-1}\mathbf{y} \in \mathcal{K}_n(\mathbf{A}, \mathbf{y})\} \leq \partial\hat{\chi}_{\mathbf{A}},$$

where  $\partial\hat{\chi}_{\mathbf{A}}$  denotes the degree of the minimal polynomial of  $\mathbf{A}$ .

For proving the validity of the  $\leq$ -sign one applies an enhanced form of the Cayley–Hamilton theorem which says that for the minimal polynomial  $\hat{\chi}_{\mathbf{A}}$  of  $\mathbf{A}$  holds  $\hat{\chi}_{\mathbf{A}}(\mathbf{A}) = \mathbf{O}$ .

Unfortunately, even for  $n \leq \bar{\nu}$  the vectors  $\mathbf{y}, \mathbf{A}\mathbf{y}, \dots, \mathbf{A}^{n-1}\mathbf{y}$  form typically a very ill-conditioned basis for  $\mathcal{K}_n$ , since they tend to be nearly linearly dependent. In fact, one can easily show that if  $\mathbf{A}$  has a unique eigenvalue of largest absolute value and of algebraic multiplicity one, and if  $\mathbf{y}$  is not orthogonal to a corresponding normalized eigenvector  $\mathbf{y}$ , then  $\mathbf{A}^k\mathbf{y}/\|\mathbf{A}^k\mathbf{y}\| \rightarrow \pm\mathbf{y}$  as  $k \rightarrow \infty$ . Therefore, in practice, we will never make use of this so-called **Krylov basis**.

From Lemma 3 it now follows quickly that once we have constructed a basis of  $\mathcal{K}_{\bar{\nu}}(\mathbf{A}, \mathbf{r}_0)$  we can find the exact solution of the linear system there.

COROLLARY 4. *Let  $\mathbf{x}_*$  be the solution of  $\mathbf{A}\mathbf{x} = \mathbf{b}$  and let  $\mathbf{x}_0$  be any initial approximation of it and  $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$  the corresponding residual. Moreover, let  $\bar{\nu} := \bar{\nu}(\mathbf{r}_0, \mathbf{A})$ . Then*

$$\mathbf{x}_* \in \mathbf{x}_0 + \mathcal{K}_{\bar{\nu}}(\mathbf{A}, \mathbf{r}_0).$$

However, this is a theoretical result that is of limited practical value. Normally  $\bar{\nu}$  is so large that we do not want to spend the  $\bar{\nu}$  matrix-vector products that are needed to construct a basis of  $\mathcal{K}_{\bar{\nu}}(\mathbf{A}, \mathbf{r}_0)$ . We want to find very good approximate solutions with much fewer matrix-vector products. Typically, Krylov space solvers provide that; but there are always exceptions of particularly hard problems. On the other hand, there exist situations where  $\bar{\nu}$  is very small compared to  $N$ , and then Krylov space solvers do particularly well. Moreover, if we replaced the target of finding the exact solution  $\mathbf{x}_*$  by the one of finding a good approximate solution, we could deduce from a correspondingly adapted notion of grade results that are more relevant in practice.

**3. Preconditioning.** When applied to large real-world problems Krylov space solvers often converge very slowly — if at all. In practice, Krylov space solvers are therefore nearly always applied with **preconditioning**. The basic idea behind it is to replace the given linear system  $\mathbf{Ax} = \mathbf{b}$  by an equivalent one whose matrix is more suitable for a treatment by the chosen Krylov space method. In particular, the new matrix will normally have a much smaller condition number. Ideal are matrices whose eigenvalues are clustered around one point except for a few outliers, and such that the cluster is well separated from the origin. Other properties, like the degree of nonnormality, also play a role, since highly nonnormal matrices often cause a delay of the convergence. Note that again, this new matrix needs not be available explicitly.

There are several ways of preconditioning. In the simplest case, called **left preconditioning** the system is just multiplied from the left by some matrix  $\mathbf{C}$  that is in some sense an approximation of the inverse of  $\mathbf{A}$ :

$$\boxed{\underbrace{\mathbf{CA}}_{\hat{\mathbf{A}}} \mathbf{x} = \underbrace{\mathbf{Cb}}_{\hat{\mathbf{b}}}.}$$

$\mathbf{C}$  is then called a **left preconditioner** or, more appropriately, an **approximate inverse** applied on the left-hand side. Of course,  $\mathbf{C}$  should be sparse or specially structured, so that the matrix-vector product  $\mathbf{Cy}$  can be computed quickly for any  $\mathbf{y}$ .

Often, not  $\mathbf{C}$  is given but its inverse  $\mathbf{M} := \mathbf{C}^{-1}$ , which is also called **left preconditioner**. In this case, we need to be able to solve  $\mathbf{Mz} = \mathbf{y}$  quickly for any right-hand side  $\mathbf{y}$ .

Analogously there is **right preconditioning**. Yet another option is the so-called **split preconditioning** by two matrices  $\mathbf{L}$  and  $\mathbf{U}$ , so that  $\hat{\mathbf{A}} := \mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}$  is in some sense close to the unit matrix. The given linear system is then replaced by

$$\boxed{\underbrace{\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}}_{\hat{\mathbf{A}}} \underbrace{\mathbf{U}\mathbf{x}}_{\hat{\mathbf{x}}} = \underbrace{\mathbf{L}^{-1}\mathbf{b}}_{\hat{\mathbf{b}}},} \quad (3.1)$$

An often very effective approach is to choose  $\mathbf{L}$  and  $\mathbf{U}$  as approximate factors of the Gaussian LU decomposition of  $\mathbf{A}$ . More specifically, these factors are chosen such that they have less nonzero elements (less fill-in) than the exact LU factors. There are several versions of such **incomplete LU decompositions**, in particular ILUT, where the deletion of computed nonzeros depends on a threshold for the absolute value of that element in relation to some of the other elements.

The split preconditioning is particularly useful if  $\mathbf{A}$  is real symmetric (or Hermitian) and we choose as the right preconditioner the (Hermitian) transposed of the left one, so that  $\hat{\mathbf{A}}$  is still symmetric (or Hermitian). In this case the incomplete LU decomposition becomes an **incomplete Cholesky decomposition**.

The incomplete LU and Cholesky decompositions are affected by row (equation) and column (variable) permutations. If these are chosen suitably, the effect of the preconditioning can be improved. The new software package ILS [19] that has been used to compute some of the numerical results of the next section applies

- an unsymmetric permutation; see Duff/Koster [12];
- a symmetric permutation such as nested dissection (ND), reverse Cuthill-McKee (RCM), or multiple minimum degree (MMD);
- various versions of incomplete LU (ILU) and sparse approximate inverse (SPAI) preconditioning, preferably ILUT;
- a Krylov space solver, preferably BiCGSTAB [33].

Table 4.1: The matrices used in Röllin’s numerical experiments [19]. (“Elements” refers to the number of structurally nonzero matrix elements)

Name	Unknowns	Elements	Dim.
Igbt-10	11’010	234’984	2D
Si-pbh-laser-21	14’086	511’484	2D
Nmos-10	18’627	387’457	2D
Barrier2-7	115’625	6’372’663	3D
Para-7	155’924	8’374’204	3D
With-dfb-36	174’272	8’625’700	3D
Ohne-9	183’038	11’170’886	3D
Resistivity-9	318’026	19’455’650	3D

Table 4.2: Conditioning and diagonal dominance of the matrices used in Röllin’s numerical experiments [19]:

Matrix	Original			Scaled & permuted with MPS		
	Condest	d.d.rows	d.d.cols	Condest	d.d.rows	d.d.cols
Igbt-10	4.73e+19	2’421	132	1.55e+08	2’397	5’032
Si-pbh-laser-21	7.11e+23	1’530	54	5.34e+08	1’488	3’456
Nmos-10	9.28e+20	2’951	57	6.09e+06	2’951	6’862
Barrier2-7	2.99e+19	30’073	5’486	1.15e+19	24’956	53’930
Para-7	1.48e+19	41’144	5’768	2.74e+19	39’386	76’920
With-dfb-36	1.25e+20	33’424	3’837	9.75e+06	32’582	75’299
Ohne-9	7.48e+19	45’567	3’975	1.11e+20	43’799	91’609
Resistivity-9	failed	105’980	768	1.08e+09	105’850	109’581

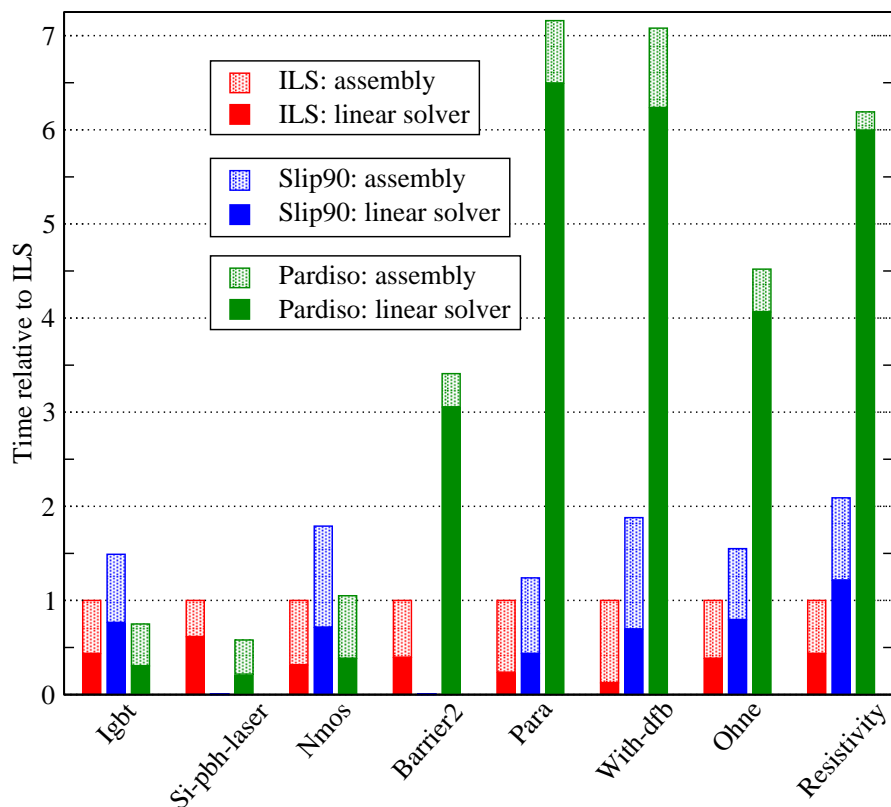
**4. Direct vs. iterative solvers.** There is a competition between direct and iterative methods for large sparse linear systems. Which approach is preferable? This depends very much on the particular problem: the sparsity and nonzero structure of the matrix, the efficiency of the preconditioner for the iterative method, and even a bit on the right hand side. But a general rule that holds in many examples based on PDEs is: direct for 1D and 2D problems — iterative for 3D problems.

Here are some new results by Stefan Röllin [19] which illustrate this rule. He solved a number of representative problems from semiconductor device simulation both by a sparse direct and by two preconditioned iterative solver packages, both allowing to use the BICGSTAB method. All three software packages are optimized packages that are used in or have been designed for commercial products for such applications. Slip90 was written by Diederik Fokkema and used to be the standard iterative solver in the semiconductor device simulator DESSIS. It uses BiCGStab(2) [31], a variant of BiCGStab2 [14]. The direct solver PARDISO is due to Schenk and Gärtner [26, 27]. The most recent package is Röllin’s own ILS, which applies BICGSTAB but includes new preconditioning features including the option to permute equations and unknowns suitably so that the ILU preconditioner, and in particular ILUT, becomes more effective.

Röllin’s results taken from [19] have been obtained on a sequential computer, although both PARDISO and ILS are best on shared memory multiprocessor computers running OPENMP.

The Tables 4.1 and 4.2 indicate the type, size, and condition number of the matrices that were used in the numerical experiments, and Figure 4.1 summarizes the

Fig. 4.1: Overall runtime for complete semiconductor device simulations for different linear solvers and simulations, as measured by Röllin [19]. All times are scaled to ILS. Dark bars show the time spent for the solution of the linear systems. Light bars indicate the time used to assemble the matrices and right hand sides; they also include the remaining time of the other parts in a simulation. Some of the results for PARDISO were found on faster computers with more memory.



run-time measurements for the corresponding complete semiconductor device simulations in which dozens of linear systems are solved. In Table 4.1 the column “Elements” shows the number of matrix elements that could be nonzero in view of the structure of the matrix.

Röllin’s measurements clearly reflect the rule that direct sparse solvers are competitive for (physically) two-dimensional problems, but not for three-dimensional ones. In addition to the enormous differences in the runtime, the memory requirements of the direct solver are much larger than those of the iterative solvers. The results also show that the new package ILS is more reliable than the older iterative package Slip90 in the sense that more simulations are successfully solved with ILS. But also the time spent to solve the linear system is reduced by up to a factor of three.

**5. Linear systems with multiple right-hand sides.** A nonsingular linear systems with  $s$  right-hand sides (RHSs) can be written as

$$\boxed{\mathbf{Ax} = \mathbf{b}} \quad \text{with} \quad \mathbf{A} \in \mathbb{C}^{N \times N}, \quad \mathbf{b} \in \mathbb{C}^{N \times s}, \quad \mathbf{x} \in \mathbb{C}^{N \times s}. \quad (5.1)$$

We allow here for complex systems because they can be covered with no extra effort. Most other authors choose the notation  $AX = B$ , but we want to use boldface lowercase letters for the “high and skinny”  $N \times s$  matrices of the unknowns and the RHSs. Generally, we will call such  $N \times s$  matrices **block vectors**. Their  $s$  columns will be distinguished by an upper index when they are used separately.

Using Gauss elimination we can solve such a system with  $s$  RHSs much more efficiently than  $s$  single linear systems with different matrices, since the LU decomposition of  $\mathbf{A}$  is needed only once. It does not matter if all the RHSs are known at the beginning or are produced one after another while the systems are solved.

If iterative methods are applied, it is harder to solve (5.1) much faster than  $s$  systems with a single RHS, and it does matter whether the RHSs are initially all known or not. There are two approaches:

- using the (iterative) solution of a **seed system** for subsequently solving the other systems faster,
- using **block Krylov space solvers**, which treat several RHSs at once.

In the second case, which we will treat in Sections 6–15 all RHSs are needed at once.

Most Krylov space solvers can be generalized easily to block Krylov space solvers, but the stability of block methods requires additional effort. Often this aspect is neglected. Moreover, such block methods may be, but need not be much faster than solving the  $s$  systems separately. Related block Krylov space methods for eigenvalues allow us to find multiple eigenvalues and corresponding eigenspaces. Here, accuracy aspects are even more important.

There are a variety of methods that can be viewed as seed methods. Here is a simplified view of one approach. We solve one system, say

$$\mathbf{A}x^{(1)} = b^{(1)},$$

by some Krylov space method generating the nested subspaces

$$\mathcal{K}_n := \mathcal{K}_n(\mathbf{A}, r_0^{(1)}) := \text{span}(r_0^{(1)}, \mathbf{A}r_0^{(1)}, \dots, \mathbf{A}^{n-1}r_0^{(1)}),$$

where

$$r_0^{(1)} := b^{(1)} - \mathbf{A}x_0^{(1)}.$$

For example, in GMRES,  $r_0^{(1)}$  is projected orthogonally onto  $\mathbf{A}\mathcal{K}_n$  to yield  $r_0^{(1)} - r_n^{(1)}$ .

We then project  $r_0^{(2)}, \dots, r_0^{(s)}$  onto  $\mathbf{A}\mathcal{K}_n$  to find approximate solutions  $x_n^{(2)}, \dots, x_n^{(s)}$  of the other  $s-1$  systems. Typically, these approximations are not yet sufficiently good, but they can be improved by applying the Krylov space solver to these  $s-1$  systems using  $x_n^{(2)}, \dots, x_n^{(s)}$  as initial approximations. Or we can consider the system  $\mathbf{A}x^{(2)} = b^{(2)}$  with initial approximation  $x_n^{(2)}$  as a new seed system.

References on the seed system approach include [18, 22, 32, 30, 7, 25]. Similar more recent work goes under the keywords “augmented basis” and “recycling Krylov spaces”. These techniques are not limited to systems with identical matrix but can also handle nearby matrices.

**6. Block Krylov spaces.** Let us return to the linear system (5.1) with the  $N \times N$  matrix  $\mathbf{A}$  and the block vectors  $\mathbf{b}, \mathbf{x} \in \mathbb{C}^{N \times s}$ . By defining an inner product we can readily turn the vector space  $\mathbb{C}^{N \times s}$  of block vectors into an Euclidean space, that is a finite-dimensional inner product space.

DEFINITION. For block vectors, the inner product  $\langle \cdot, \cdot \rangle_F$  and the norm  $\|\cdot\|_F$  it induces are defined by

$$\langle \mathbf{x}, \mathbf{y} \rangle_F := \text{trace } \mathbf{x}^* \mathbf{y}, \quad \|\mathbf{x}\|_F := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_F} := \sqrt{\text{trace } \mathbf{x}^* \mathbf{x}}. \quad (6.1)$$

▲

If

$$\mathbf{x} = \begin{pmatrix} x^{(1)} & \dots & x^{(s)} \end{pmatrix} = \begin{pmatrix} \xi_{11} & \dots & \xi_{1s} \\ \vdots & & \vdots \\ \xi_{N1} & \dots & \xi_{Ns} \end{pmatrix} \in \mathbb{C}^{N \times s},$$

and  $\mathbf{y}$  is structured analogously, then, with  $\langle \cdot, \cdot \rangle_2$  denoting the Euclidean inner product in  $\mathbb{C}^N$ ,

$$\langle \mathbf{x}, \mathbf{y} \rangle_F = \sum_{j=1}^s \langle x^{(j)}, y^{(j)} \rangle_2 = \sum_{j=1}^s \sum_{i=1}^N \xi_{i,j} \eta_{i,j}, \quad (6.2)$$

$$\|\mathbf{x}\|_F = \sqrt{\sum_{j=1}^s \|x^{(j)}\|_2^2} = \sqrt{\sum_{j=1}^s \sum_{i=1}^N |\xi_{i,j}|^2}, \quad (6.3)$$

whence it is customary to call  $\|\mathbf{x}\|_F$  the Frobenius norm, though it should not be considered as a matrix norm here, but as a (block) vector norm.

We want to anticipate, however, that the orthogonality induced by the inner product  $\langle \mathbf{x}, \mathbf{y} \rangle_F$  is not the one that we will use for the block Arnoldi process and block GMRES.

Amazingly, most publications on block Krylov (sub)space methods are a bit fuzzy when the basic approximation space is introduced. One assumes that some approximation  $\mathbf{x}_0 \in \mathbb{C}^{N \times s}$  is given and determines the **initial block residual**

$$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0 \in \mathbb{C}^{N \times s}. \quad (6.4)$$

Recall from Section 2 that in the case of a single system, that is when  $s = 1$ , the approximate solution  $\mathbf{x}_n$  is chosen such that the correction  $\mathbf{x}_n - \mathbf{x}_0$  lies in the **Krylov (sub)space**

$$\mathcal{K}_n := \mathcal{K}_n(\mathbf{A}\mathbf{r}_0) := \text{span}(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^{n-1}\mathbf{r}_0) \subset \mathbb{C}^N. \quad (6.5)$$

Many authors state that the same formula holds for the block case  $s > 1$ , but the usual definition of “span” would mean that

$$\mathbf{x}_n - \mathbf{x}_0 = \sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{r}_0 \gamma_k \quad (6.6)$$

for some scalars  $\gamma_0, \dots, \gamma_{n-1} \in \mathbb{C}$ , which is not correct for typical block methods. (It is, however, the choice made in the so-called global methods [15].) Instead, each of the  $s$  columns of  $\mathbf{x}_n - \mathbf{x}_0$  is approximated by a linear combination of all the  $s \times n$  columns in  $\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^{n-1}\mathbf{r}_0$ . To clarify this point we make the following definitions:

DEFINITION. Given  $\mathbf{A} \in \mathbb{C}^{N \times N}$  and  $\mathbf{y} \in \mathbb{C}^{N \times s}$ , the **block Krylov (sub) spaces**  $\mathcal{B}_n^\square$  ( $n \in \mathbb{N}^+$ ) generated by  $\mathbf{A}$  from  $\mathbf{y}$  are

$$\boxed{\mathcal{B}_n^\square := \mathcal{B}_n^\square(\mathbf{A}, \mathbf{y}) := \text{block span}(\mathbf{y}, \mathbf{A}\mathbf{y}, \dots, \mathbf{A}^{n-1}\mathbf{y}) \subset \mathbb{C}^{N \times s}}, \quad (6.7)$$

where ‘block span’ is defined such that

$$\boxed{\mathcal{B}_n^\square = \left\{ \sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{y} \gamma_k; \gamma_k \in \mathbb{C}^{s \times s} (k = 0, \dots, n-1) \right\}}. \quad (6.8)$$

A **block Krylov space method** for solving the  $s$  systems (5.1) is an iterative method that generates approximate solutions  $\mathbf{x}_n$  such that

$$\boxed{\mathbf{x}_n - \mathbf{x}_0 \in \mathcal{B}_n^\square(\mathbf{A}, \mathbf{r}_0)}, \quad (6.9)$$

where  $\mathbf{r}_0$  is the initial residual (6.4). ▲

The sum in (6.8) is not an ordinary linear combination, but it can be replaced by one: let

$$\boldsymbol{\epsilon}_{i,j} = \left( \begin{array}{c} \epsilon_{k,l}^{(i,j)} \end{array} \right)_{k,l=1}^s \in \mathbb{C}^{s \times s} \quad \text{with} \quad \epsilon_{k,l}^{(i,j)} := \begin{cases} 1 & \text{if } k = i \text{ and } l = j, \\ 0 & \text{otherwise.} \end{cases}$$

Then, if  $\boldsymbol{\gamma}_k = \left( \begin{array}{c} \gamma_{i,j}^{(k)} \end{array} \right)_{i,j=1}^s \in \mathbb{C}^{s \times s}$ ,

$$\mathbf{y} \boldsymbol{\gamma}_k = \sum_{i=1}^s \sum_{j=1}^s \gamma_{i,j}^{(k)} \mathbf{y} \boldsymbol{\epsilon}_{i,j}, \quad (6.10)$$

$$\sum_{k=0}^{n-1} \mathbf{A}^k \mathbf{y} \boldsymbol{\gamma}_k = \sum_{k=0}^{n-1} \sum_{i=1}^s \sum_{j=1}^s \gamma_{i,j}^{(k)} \mathbf{A}^k \mathbf{y} \boldsymbol{\epsilon}_{i,j}. \quad (6.11)$$

If the  $ns^2$  block vectors  $\mathbf{A}^k \mathbf{y} \boldsymbol{\epsilon}_{i,j} \in \mathbb{C}^{N \times s}$  ( $k = 0, \dots, n-1$ ) are linearly independent,

$$\dim \mathcal{B}_n^\square = ns^2. \quad (6.12)$$

EXAMPLE 6.1. *The following is a simple example for the construction in (6.10) and in (6.11) when  $k = 0$  and  $n = 1$ . We assume  $s = 2$  and let  $\mathbf{y} = \left( \begin{array}{cc} y^{(1)} & y^{(2)} \end{array} \right)$  to get*

$$\begin{aligned} \mathbf{y} \boldsymbol{\gamma}_0 &= \gamma_{11}^{(0)} \mathbf{y} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \gamma_{12}^{(0)} \mathbf{y} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \gamma_{21}^{(0)} \mathbf{y} \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + \gamma_{22}^{(0)} \mathbf{y} \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \\ &= \gamma_{11}^{(0)} \left( \begin{array}{cc} y^{(1)} & o \end{array} \right) + \gamma_{12}^{(0)} \left( \begin{array}{cc} o & y^{(1)} \end{array} \right) + \gamma_{21}^{(0)} \left( \begin{array}{cc} y^{(2)} & o \end{array} \right) + \gamma_{22}^{(0)} \left( \begin{array}{cc} o & y^{(2)} \end{array} \right). \end{aligned}$$

Here,  $o$  is the zero vector in  $\mathbb{C}^N$ . If  $y^{(1)}$  and  $y^{(2)}$  are linearly independent vectors, then  $\left( \begin{array}{cc} y^{(1)} & o \end{array} \right)$ ,  $\left( \begin{array}{cc} o & y^{(1)} \end{array} \right)$ ,  $\left( \begin{array}{cc} y^{(2)} & o \end{array} \right)$ , and  $\left( \begin{array}{cc} o & y^{(2)} \end{array} \right)$  are four linearly independent block vectors.

**7. The block grade.** The condition  $\mathbf{z} \in \mathcal{B}_n^\square$  can also be written as

$$\mathbf{z} =: \left( \begin{array}{ccc} z^{(1)} & \dots & z^{(s)} \end{array} \right) \quad \text{with} \quad z^{(j)} \in \mathcal{B}_n \quad (j = 1, \dots, s), \quad (7.1)$$

where

$$\mathcal{B}_n := \mathcal{B}_n(\mathbf{A}, \mathbf{y}) := \left\{ \sum_{i=1}^s \sum_{k=0}^{n-1} \mathbf{A}^k y^{(i)} \beta_{k,i}; \beta_{k,i} \in \mathbb{C} (\forall k, i) \right\}, \quad (7.2)$$

or, in other words,  $\mathcal{B}_n$  is the sum of the  $s$  Krylov subspaces  $\mathcal{K}(\mathbf{A}, y^{(i)})$ :

$$\boxed{\mathcal{B}_n = \mathcal{K}_n(\mathbf{A}, y^{(1)}) + \cdots + \mathcal{K}_n(\mathbf{A}, y^{(s)})}. \quad (7.3)$$

$\mathcal{B}_n^\square$  is just the Cartesian product of  $s$  copies of  $\mathcal{B}_n$ :

$$\boxed{\mathcal{B}_n^\square = \underbrace{\mathcal{B}_n \times \cdots \times \mathcal{B}_n}_{s \text{ times}}}. \quad (7.4)$$

So,  $x_0^{(i)} + \mathcal{B}_n$  is the affine space where the approximation  $x_n^{(i)}$  of the solution of the  $i$ th system  $\mathbf{A}x^{(i)} = b^{(i)}$  is constructed from:

$$\boxed{x_n^{(i)} \in x_0^{(i)} + \mathcal{B}_n}. \quad (7.5)$$

Clearly, if the  $ns$  vectors  $\mathbf{A}^k y^{(i)} \in \mathbb{C}^N$  in (7.2) are linearly independent,

$$\dim \mathcal{B}_n = ns. \quad (7.6)$$

But  $\dim \mathcal{B}_n$  can be less than  $ns$  because the sum (7.3) needs not be a direct sum and because  $\dim \mathcal{K}_n(\mathbf{A}, y^{(i)}) < n$  may hold for some  $i$ . This is where the difficulties but also some of the merits come from.

Like the Krylov subspaces, the subspaces  $\mathcal{B}_n$  and  $\mathcal{B}_n^\square$  are nested:

$$\mathcal{B}_n \subseteq \mathcal{B}_{n+1}, \quad \mathcal{B}_n^\square \subseteq \mathcal{B}_{n+1}^\square.$$

Again, for sufficiently large  $n$  equality holds. Schmelzer [28] introduced a generalization of the grade discussed in Section 2 to block Krylov spaces. It is based on a adaptation of Corollary 2 and allows us to establish a number of results.

DEFINITION. The positive integer  $\bar{\nu} := \bar{\nu}(\mathbf{y}, \mathbf{A})$  defined by

$$\boxed{\bar{\nu}(\mathbf{y}, \mathbf{A}) = \min \{n \mid \dim \mathcal{B}_n(\mathbf{A}, \mathbf{y}) = \dim \mathcal{B}_{n+1}(\mathbf{A}, \mathbf{y})\}}$$

is called **block grade of  $\mathbf{y}$  with respect to  $\mathbf{A}$** . ▲

In analogy to Lemma 1 we have then:

LEMMA 5. For  $n \geq \bar{\nu}(\mathbf{y}, \mathbf{A})$ ,

$$\boxed{\mathcal{B}_n(\mathbf{A}, \mathbf{y}) = \mathcal{B}_{n+1}(\mathbf{A}, \mathbf{y}), \quad \mathcal{B}_n^\square(\mathbf{A}, \mathbf{y}) = \mathcal{B}_{n+1}^\square(\mathbf{A}, \mathbf{y})}. \quad (7.7)$$

PROOF. By definition of  $\bar{\nu}(\mathbf{y}, \mathbf{A})$ , (7.7) holds for  $n = \bar{\nu}(\mathbf{y}, \mathbf{A})$ . Since for any of the individual Krylov spaces  $\mathcal{K}_n(\mathbf{A}, y^{(j)})$  in (7.3) we have clearly  $\mathcal{K}_{n+1}(\mathbf{A}, y^{(j)}) = \mathcal{K}_1(\mathbf{A}, y^{(j)}) + \mathbf{A}\mathcal{K}_n(\mathbf{A}, y^{(j)})$  it holds likewise that  $\mathcal{B}_{n+1}(\mathbf{A}, \mathbf{y}) = \mathcal{B}_1(\mathbf{A}, \mathbf{y}) + \mathbf{A}\mathcal{B}_n(\mathbf{A}, \mathbf{y})$ . So, in view of the nonsingularity of  $\mathbf{A}$  and the dimensions of the subspaces involved,  $\mathbf{A}\mathcal{B}_{\bar{\nu}}(\mathbf{A}, \mathbf{y}) = \mathcal{B}_{\bar{\nu}}(\mathbf{A}, \mathbf{y})$ , that is  $\mathcal{B}_{\bar{\nu}}(\mathbf{A}, \mathbf{y})$  is an invariant subspace of  $\mathbf{A}$ , and  $\mathcal{B}_{\bar{\nu}+1}(\mathbf{A}, \mathbf{y}) = \mathcal{B}_{\bar{\nu}}(\mathbf{A}, \mathbf{y})$ . So, applying  $\mathbf{A}$  to any element of  $\mathcal{B}_{\bar{\nu}}(\mathbf{A}, \mathbf{y})$  does not lead out of the space, i.e., the equality on left side of (7.7) holds. The one on the right side follows then from (7.4). □

The following lemma is another easy, but nontrivial result:

LEMMA 6. The block grade of the block Krylov space and the grades of the individual Krylov spaces contained in it are related by

$$\boxed{\mathcal{B}_{\bar{\nu}(\mathbf{y}, \mathbf{A})}(\mathbf{A}, \mathbf{y}) = \mathcal{K}_{\bar{\nu}(\mathbf{y}^{(1)}, \mathbf{A})}(\mathbf{A}, y^{(1)}) + \cdots + \mathcal{K}_{\bar{\nu}(\mathbf{y}^{(s)}, \mathbf{A})}(\mathbf{A}, y^{(s)})}. \quad (7.8)$$

PROOF. We choose in (7.3)  $n$  larger than the indices  $\bar{\nu}$  of all the spaces that appear in there and apply Lemma 5 on the left-hand side and  $s$  times Lemma 1 on the right-hand side.  $\square$

By definition of  $\bar{\nu} = \bar{\nu}(\mathbf{y}, \mathbf{A})$ , the columns of  $\mathbf{A}^{\bar{\nu}}\mathbf{y}$  are linear combinations of the columns of  $\mathbf{y}, \mathbf{A}\mathbf{y}, \dots, \mathbf{A}^{\bar{\nu}-1}\mathbf{y}$ , and this does not hold for all columns of  $\mathbf{A}^n\mathbf{y}$  for any  $n < \bar{\nu}$ . That means that there are matrices  $\gamma_0, \dots, \gamma_{\bar{\nu}-1} \in \mathbb{C}^{s \times s}$ , such that

$$\mathbf{A}^{\bar{\nu}}\mathbf{y} = \mathbf{y}\gamma_0 + \mathbf{A}\mathbf{y}\gamma_1 + \dots + \mathbf{A}^{\bar{\nu}-1}\mathbf{y}\gamma_{\bar{\nu}-1}. \quad (7.9)$$

Here,  $\gamma_0 \neq 0$ , because of the minimality of  $\bar{\nu}$ , but unfortunately we cannot be sure that  $\gamma_0$  is nonsingular. So, we cannot solve (7.9) easily for  $\mathbf{y}$  and then apply  $\mathbf{A}^{-1}$  to it. Nevertheless, by an alternative, more complicated argument we can still prove the following analog of Lemma 3.

LEMMA 7. *The block grade  $\bar{\nu}(\mathbf{y}, \mathbf{A})$  is characterized by*

$$\bar{\nu}(\mathbf{y}, \mathbf{A}) = \min \left\{ n \mid \mathbf{A}^{-1}\mathbf{y} \in \mathcal{B}_n^\square(\mathbf{A}, \mathbf{y}) \right\} \leq \partial\widehat{\chi}_{\mathbf{A}},$$

where  $\partial\widehat{\chi}_{\mathbf{A}}$  denotes the degree of the minimal polynomial of  $\mathbf{A}$ .

Next we are looking for an analog of Corollary 4.

COROLLARY 8. *Let  $\mathbf{x}_*$  be the block solution of  $\mathbf{A}\mathbf{x} = \mathbf{b}$  and let  $\mathbf{x}_0$  be any initial block approximation of it and  $\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$  the corresponding block residual. Then*

$$\mathbf{x}_* \in \mathbf{x}_0 + \mathcal{B}_{\bar{\nu}(\mathbf{r}_0, \mathbf{A})}^\square(\mathbf{A}, \mathbf{r}_0). \quad (7.10)$$

PROOF. We just combine Corollary 4 and the relations (7.8) and (7.4).  $\square$

**8. Deflation.** When block Krylov space solvers have to be worked out in detail, the extra challenge comes from the possible linear dependence of the residuals of the  $s$  systems. In most block methods such a dependence requires an explicit reduction of the number of RHSs. We call this **deflation**. (This term is also used with different meanings.) In the literature on block methods this deflation is only treated in a few papers, such as Cullum and Willoughby [10] (for symmetric Lanczos), Nikishin and Yeremin [17] (block CG), Aliaga, Boley, Freund, and Hernández [1] (nonsymmetric Lanczos), and Cullum and Zhang [9] (Arnoldi, nonsymmetric Lanczos).

Deflation may be possible at startup or in a later step. It is often said that it becomes necessary when “one of the systems converges”. However, deflation does not depend on an individual one of the  $s$  systems, but on the dimension of the space the  $s$  residuals span. So it becomes feasible when “a linear combination of the  $s$  systems converges”.

Let us consider some (very special) examples:

1. Let  $\mathbf{r}_0$  consist of  $s$  identical vectors  $r$ ,

$$\mathbf{r}_0 := \begin{pmatrix} r & r & r & \dots & r \end{pmatrix}.$$

These could come from different  $b^{(i)}$  and suitably chosen  $x_0^{(i)}$ :

$$r = b^{(i)} - \mathbf{A}x_0^{(i)} \quad (i = 1, \dots, s)$$

In this case it suffices to solve one system.

2. Assume that

$$\mathbf{r}_0 := \begin{pmatrix} r & \mathbf{A}r & \mathbf{A}^2r & \dots & \mathbf{A}^{s-1}r \end{pmatrix}.$$

Here, even if  $\text{rank } \mathbf{r}_0 = s$ , one extension of the block Krylov subspace leads to the column space of  $\begin{pmatrix} \mathbf{r}_0 & \mathbf{A}\mathbf{r}_0 \end{pmatrix}$  which has  $\text{rank} \begin{pmatrix} \mathbf{r}_0 & \mathbf{A}\mathbf{r}_0 \end{pmatrix} \leq s + 1$  only.

3. Let  $\mathbf{r}_0$  have  $s$  linearly independent columns that are linear combinations of  $s$  eigenvectors of  $\mathbf{A}$ . Then the column space of  $\mathbf{r}_0$  is an invariant subspace and thus  $\text{rank} \begin{pmatrix} \mathbf{r}_0 & \mathbf{A}\mathbf{r}_0 \end{pmatrix} \leq s$  and  $\bar{\nu}(\mathbf{r}_0, \mathbf{A}) = 1$ . Hence, one block iteration is enough to solve all systems. A non-block solver might require as many as  $s^2$  iterations.

Clearly exact deflation leads to a reduction of the number of MVS that are needed to solve the  $s$  linear systems. In fact, as we have seen, in the single RHS case, in exact arithmetic, computing the exact solution  $\mathbf{x}_*$  requires

$$\dim \mathcal{K}_{\bar{\nu}(\mathbf{r}_0, \mathbf{A})}(\mathbf{A}, r_0) = \bar{\nu}(\mathbf{r}_0, \mathbf{A}) \quad \text{MVS.}$$

In the multiple RHS case, in exact arithmetic, computing  $\mathbf{x}_*$  requires

$$\dim \mathcal{B}_{\bar{\nu}(\mathbf{r}_0, \mathbf{A})}(\mathbf{A}, \mathbf{r}_0) \in [\bar{\nu}(\mathbf{r}_0, \mathbf{A}), s\bar{\nu}(\mathbf{r}_0, \mathbf{A})] \quad \text{MVS.}$$

The latter is a big interval! From this point of view we can conclude that *block methods are most effective (compared to single RHS methods) if*

$$\dim \mathcal{B}_{\bar{\nu}(\mathbf{r}_0, \mathbf{A})}(\mathbf{A}, \mathbf{r}_0) \ll s\bar{\nu}(\mathbf{r}_0, \mathbf{A}).$$

*More exactly: block methods are most effective if*

$$\dim \mathcal{B}_{\bar{\nu}(\mathbf{r}_0, \mathbf{A})}(\mathbf{A}, \mathbf{r}_0) \ll \sum_{k=1}^s \dim \mathcal{K}_{\bar{\nu}(r_0^{(k)}, \mathbf{A})}(\mathbf{A}, r_0^{(k)}).$$

But this can be capitalized upon only if deflation is implemented, so *block methods are most effective (compared to single RHS methods) if deflation is possible and enforced!*

However, exact deflation is rare, and — as we will see below — in practice we need approximate deflation depending on a deflation tolerance. But approximate deflation introduces a deflation error, which may cause the convergence to slow down or may reduce the accuracy of the computed solution.

Restarting the iteration can be useful from this point of view.

**9. The block Arnoldi process without deflation.** Let us now look at a particular block Krylov space solver, the block GMRES (BLGMRES) method, which is a block version due to Vital [34] of the widely used GMRES algorithm of Saad and Schultz [24]. The latter method builds up an orthonormal basis for the nested Krylov spaces  $\mathcal{K}_n(\mathbf{A}, \mathbf{r}_0)$  and then solves (for a single system) the minimum problem

$$\|\mathbf{r}_n\|_2 = \min! \quad \text{subject to} \quad \mathbf{x}_n - \mathbf{x}_0 \in \mathcal{K}_n(\mathbf{A}, \mathbf{r}_0) \quad (9.1)$$

in coordinate space. (It makes use of the fact that in the 2-norms the coordinate map is isometric (length invariant), as we know from the famous Parseval formula. The orthonormal basis is created with the so-called Arnoldi process, so we first need to extend the latter to the block case.

One could define an Arnoldi process for block vectors by making use of the inner product  $\langle \cdot, \cdot \rangle_F$  and the norm  $\|\cdot\|_F$  defined by (6.1) (and this is what is done in the so-called global GMRES method [15]), but the aim here is a different one. We denote

the zero and unit matrix in  $\mathbb{C}^{s \times s}$  by  $\mathbf{o} = \mathbf{o}_s$  and  $\boldsymbol{\iota} = \boldsymbol{\iota}_s$ , respectively, and introduce the following notions:

DEFINITION. We call the block vectors  $\mathbf{x}$ ,  $\mathbf{y}$  **block-orthogonal** if  $\mathbf{x}^* \mathbf{y} = \mathbf{o}$ , and we call  $\mathbf{x}$  **block-normalized** if  $\mathbf{x}^* \mathbf{x} = \boldsymbol{\iota}$ . A set of block vectors  $\{\mathbf{y}_n\}$  is **block-orthonormal** if these block vectors are block-normalized and mutually block-orthogonal, that is if

$$\mathbf{y}_\ell^* \mathbf{y}_n = \begin{cases} \mathbf{o}, & \ell \neq n, \\ \boldsymbol{\iota}, & \ell = n. \end{cases} \quad (9.2)$$

Written in terms of the individual columns (9.2) means that

$$\left( \tilde{y}_\ell^{(i)} \right)^* \tilde{y}_n^{(j)} = \begin{cases} 0, & \ell \neq n \text{ or } i \neq j, \\ 1, & \ell = n \text{ and } i = j. \end{cases} \quad (9.3)$$

So, a set of block-orthonormal block vectors has the property that all the columns in this set are normalized  $N$ -vectors that are orthogonal to each other (even if they belong to the same block).

For given  $\mathbf{A} \in \mathbb{C}^{N \times N}$  and  $\tilde{\mathbf{y}}_0 \in \mathbb{C}^{N \times s}$ , we want now to generate nested block-orthonormal bases for the nested block Krylov spaces  $\mathcal{B}_n^\square(\mathbf{A}, \tilde{\mathbf{y}}_0)$ . The individual columns of such a basis of  $\mathcal{B}_n^\square(\mathbf{A}, \tilde{\mathbf{y}}_0)$  form at the same time an orthonormal basis of the at most  $ns$ -dimensional space  $\mathcal{B}_n(\mathbf{A}, \tilde{\mathbf{y}}_0)$ . As long as such block-orthonormal bases consisting of  $n$  block vectors of size  $N \times s$  exist, the following block Arnoldi process will allow us to construct them. We describe here first the version without deflation and assume that an initialization step yields a block-normalized  $\mathbf{y}_0$  whose columns form an orthonormal basis of  $\mathcal{B}_1$ . The subsequent  $m$  steps yield nested orthonormal bases for  $\mathcal{B}_2, \dots, \mathcal{B}_{m+1}$ . We apply the modified block Gram-Schmidt algorithm plus a QR factorization (constructed, e.g., with modified Gram-Schmidt) of each newly found block.

ALGORITHM 9.1 (*m* STEPS OF NON-DEFLATED BLOCK ARNOLDI ALGORITHM).  
*Start:* Given  $\tilde{\mathbf{y}}_0 \in \mathbb{C}^{N \times s}$  of full rank, let  
 $\mathbf{y}_0 \boldsymbol{\rho}_0 := \tilde{\mathbf{y}}_0$  (QR factorization:  $\boldsymbol{\rho}_0 \in \mathbb{C}^{s \times s}$ ,  
 $\mathbf{y}_0 \in \mathbb{C}^{N \times s}$ ,  $\mathbf{y}_0^* \mathbf{y}_0 = \boldsymbol{\iota}$ )  
*Loop:*  
**for**  $n = 1$  **to**  $m$  **do**  
 $\tilde{\mathbf{y}} := \mathbf{A} \mathbf{y}_{n-1}$  ( $s$  MVs in parallel)  
**for**  $k = 0$  **to**  $n - 1$  **do** (blockwise MGS)  
 $\boldsymbol{\eta}_{k,n-1} := \mathbf{y}_k^* \tilde{\mathbf{y}}$  ( $s^2$  SDOTs in parallel)  
 $\tilde{\mathbf{y}} := \tilde{\mathbf{y}} - \mathbf{y}_k \boldsymbol{\eta}_{k,n-1}$  ( $s^2$  SAXPYS in parallel)  
**end**  
 $\mathbf{y}_n \boldsymbol{\eta}_{n,n-1} := \tilde{\mathbf{y}}$  (QR factorization:  
 $\boldsymbol{\eta}_{n,n-1} \in \mathbb{C}^{s \times s}$ ,  $\mathbf{y}_n^* \mathbf{y}_n = \boldsymbol{\iota}$ )  
**end**

It is readily seen that when we define the  $N \times n$  matrices

$$\mathbf{Y}_n := ( \mathbf{y}_0 \quad \mathbf{y}_1 \quad \dots \quad \mathbf{y}_{n-1} )$$

and the  $(m+1)s \times ms$  matrix

$$\mathbf{H}_m := \begin{pmatrix} \boldsymbol{\eta}_{0,0} & \boldsymbol{\eta}_{0,1} & \cdots & \boldsymbol{\eta}_{0,m-1} \\ \boldsymbol{\eta}_{1,0} & \boldsymbol{\eta}_{1,1} & \cdots & \boldsymbol{\eta}_{1,m-1} \\ & \boldsymbol{\eta}_{2,1} & \ddots & \vdots \\ & & \ddots & \boldsymbol{\eta}_{m-1,m-1} \\ & & & \boldsymbol{\eta}_{m,m-1} \end{pmatrix}, \quad (9.4)$$

which due to the upper triangularity of  $\eta_{1,0}, \dots, \eta_{m,m-1}$  is banded below with lower bandwidth  $s$ , we obtain formally the ordinary Arnoldi relation

$$\boxed{\mathbf{A}\mathbf{Y}_m = \mathbf{Y}_{m+1}\underline{\mathbf{H}}_m.} \quad (9.5)$$

In the block Arnoldi algorithm the conditions  $\mathbf{y}_n^* \mathbf{y}_n = \boldsymbol{\iota}$  hold due to the QR factorizations, and the fact that  $\mathbf{y}_\ell^* \mathbf{y}_n = \mathbf{o}$  when  $\ell \neq n$  can be shown by induction. So (9.2) and (9.3) hold. Therefore,  $\left\{ y_k^{(i)} \right\}_{k=0, i=1}^{n,s}$  is an orthonormal basis of  $\mathcal{B}_{n+1}$ . This indicates that the whole process is equivalent to one where the block vectors are generated column by column using an ordinary modified Gram-Schmidt process. The related construction for the symmetric block Lanczos process was first proposed by Ruhe [20] and gave rise to the band Lanczos algorithm. We leave it to the reader to write down the code for the corresponding band Arnoldi algorithm. (Attention: some of the published pseudo-codes contain for-loop mistakes and use a not very elegant indexing.)

In the Gram-Schmidt part of the Arnoldi algorithm the projections of the columns of  $\mathbf{A}\mathbf{y}_{n-1}$  onto  $\mathcal{B}_n(\mathbf{A}, \tilde{\mathbf{y}}_0)$  are subtracted from these columns to yield the columns of  $\tilde{\mathbf{y}}$ , which is QR factorized at the end of a step. So the block Arnoldi algorithm is not at all an adaptation of the ordinary Arnoldi algorithm (for creating an orthonormal series of vectors) to the inner product space induced by  $\langle \cdot, \cdot \rangle_F$  on  $\mathcal{B}_n^\square$ , but it is rather an adaptation of the ordinary Arnoldi algorithm to the space  $\mathcal{B}_n$ .

**10. Block GMRES without deflation.** Block GMRES (BLGMRES) [34] is based on the block Arnoldi decomposition (9.5) and is formally fully analogous to the ordinary GMRES algorithm of Saad and Schultz [24]. Given an initial approximation  $\mathbf{x}_0 \in \mathbb{C}^{N \times s}$  of the solution of  $\mathbf{A}\mathbf{x} = \mathbf{b}$  (with  $\mathbf{b} \in \mathbb{C}^{N \times s}$ ), we let

$$\tilde{\mathbf{y}}_0 := \mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0 \quad (10.1)$$

and apply block Arnoldi. Then, in analogy to the unblocked case, assuming  $\mathbf{x}_n$  is of the form

$$\boxed{\mathbf{x}_n = \mathbf{x}_0 + \mathbf{Y}_n \mathbf{k}_n,} \quad (10.2)$$

we obtain for the  $n$ th block residual

$$\mathbf{r}_n := \mathbf{b} - \mathbf{A}\mathbf{x}_n = \mathbf{r}_0 - \mathbf{A}\mathbf{Y}_n \mathbf{k}_n \quad (10.3)$$

by inserting the Arnoldi relation (9.5) and  $\mathbf{r}_0 = \tilde{\mathbf{y}}_0 = \mathbf{y}_0 \boldsymbol{\rho}_0 = \mathbf{Y}_{n+1} \underline{\mathbf{e}}_1 \boldsymbol{\rho}_0$  the equation

$$\boxed{\mathbf{r}_n = \mathbf{Y}_{n+1} \underbrace{(\underline{\mathbf{e}}_1 \boldsymbol{\rho}_0 - \underline{\mathbf{H}}_n \mathbf{k}_n)}_{=: \mathbf{q}_n}} \quad (10.4)$$

( $n = 1, \dots, m$ ), with

- $\underline{\mathbf{e}}_1$  the first  $s$  columns of the  $(n+1)s \times (n+1)s$  unit matrix (so the size changes with  $n$ ),
- $\boldsymbol{\rho}_0 \in \mathbb{C}^{s \times s}$  upper triangular, obtained in Arnoldi's initialization step,
- $\underline{\mathbf{H}}_n$  the leading  $(n+1)s \times ns$  submatrix of  $\underline{\mathbf{H}}_m$ ,
- $\mathbf{k}_n \in \mathbb{C}^{ns \times s}$  the "block coordinates" of  $\mathbf{x}_n - \mathbf{x}_0$  with respect to the block Arnoldi basis,
- $\mathbf{q}_n \in \mathbb{C}^{(n+1)s \times s}$  the block residual in coordinate space (**block quasi-residual**).

Eq. (10.4) is the fundamental BLGMRES relation. The objective is to choose  $\mathbf{k}_n$  such that

$$\boxed{\|\mathbf{r}_n\|_F = \min! \quad \text{subject to} \quad \mathbf{x}_n - \mathbf{x}_0 \in \mathcal{B}_n^\square,} \quad (10.5)$$

which is equivalent to

$$\|r_n^{(i)}\|_2 = \min! \quad \text{subject to} \quad x_n^{(i)} - x_0^{(i)} \in \mathcal{B}_n \quad (i = 1, \dots, s), \quad (10.6)$$

and, since  $\mathbf{Y}_n$  has orthonormal columns, also to

$$\|q_n^{(i)}\|_2 = \min! \quad \text{subject to} \quad k_n^{(i)} \in \mathbb{C}^{ns} \quad (i = 1, \dots, s), \quad (10.7)$$

or

$$\boxed{\|\mathbf{q}_k\|_F = \min! \quad \text{subject to} \quad \mathbf{k}_n \in \mathbb{C}^{ns \times s}.} \quad (10.8)$$

In the last statement,  $\|\mathbf{q}_n\|_F$  is now the  $F$ -norm in  $\mathbb{C}^{(n+1)s \times s}$  defined by  $\|\mathbf{q}_n\|_F := \text{trace } \mathbf{q}_n^* \mathbf{q}_n$ .

The tasks (10.7), which are combined in (10.8), are  $s$  least squares problems with the same matrix  $\underline{\mathbf{H}}_n$ . They can be solved efficiently by recursively computing the QR factorizations

$$\underline{\mathbf{H}}_n = \mathbf{Q}_n \underline{\mathbf{R}}_n \quad \text{with} \quad \underline{\mathbf{R}}_n = \begin{pmatrix} \mathbf{R}_n \\ \mathbf{o}^\top \end{pmatrix}, \quad (10.9)$$

where  $\mathbf{Q}_n$  is unitary and of order  $(n+1)s$  and  $\underline{\mathbf{R}}_n$  is an upper triangular matrix of order  $ns$ , which is nonsingular under our assumptions. So,  $\mathbf{o}^\top$  is  $s \times ns$ . Since  $\underline{\mathbf{H}}_m$  has lower bandwidth  $s$  the standard update procedure for these QR factorizations is based on Givens (Jacobi) rotations [21]. In total  $m s^2$  Givens rotations are needed to make  $\underline{\mathbf{H}}_m$  upper triangular. Thanks to using Givens rotations for this QR factorization, we can store  $\mathbf{Q}_n$  (and  $\mathbf{Q}_n^*$ ) in a very sparse factored format. But there is an even more effective alternative [29]: instead of representing  $\mathbf{Q}_n$  as a product of  $m s^2$  Givens rotations we can compute and represent it as a product of  $m s$  suitably chosen Householder reflections. Then  $\mathbf{Q}_n$  is also stored in a compact factored format.

The solution in coordinate space is finally obtained as

$$\mathbf{k}_n := \mathbf{R}_n^{-1} \underline{\mathbf{I}}_{ns}^\top \mathbf{Q}_n^* \underline{\mathbf{e}}_1 \boldsymbol{\rho}_0, \quad \text{where} \quad \underline{\mathbf{I}}_{ns} := \begin{pmatrix} \mathbf{I}_{ns} \\ \mathbf{o}^\top \end{pmatrix}, \quad (10.10)$$

with  $\underline{\mathbf{I}}_{ns}$  the unit matrix of order  $ns$  and  $\mathbf{o}^\top$  the zero matrix of size  $s \times ns$ . The corresponding minimal block quasi-residual norm is

$$\|\mathbf{q}_n\|_F = \|\underline{\mathbf{e}}_n^* \mathbf{Q}_n^* \underline{\mathbf{e}}_1 \boldsymbol{\rho}_0\|_F, \quad \text{where} \quad \underline{\mathbf{e}}_n := \begin{pmatrix} \mathbf{o} \\ \boldsymbol{\nu}_s \end{pmatrix} \in \mathbb{R}^{(n+1)s \times s}. \quad (10.11)$$

So,  $\|\mathbf{q}_n\|_F$ , which equals  $\|\mathbf{r}_n\|_F$ , is obtained from the last  $s \times s$  block of  $\mathbf{Q}_n^* \underline{\mathbf{e}}_1 \boldsymbol{\rho}_0$ , and, like in ordinary GMRES, it can be computed even when  $\mathbf{k}_n$  is not known. If we had chosen to compute a QR decomposition  $\underline{\mathbf{H}}_n = \tilde{\mathbf{Q}}_n \underline{\mathbf{R}}_n$  with  $\tilde{\mathbf{Q}}_n$  of size  $(n+1)s \times s$  and with orthonormal columns, (10.10) could be adapted, but  $\|\mathbf{q}_n\|_F$  could not be found as in (10.11).

Of course, for determining the best approximation  $\mathbf{x}_{n_1}$  we need to compute  $\mathbf{k}_n$  by solving the triangular system (10.10) with the  $s$  RHSs  $\underline{\mathbf{I}}_{ns}^\top \mathbf{Q}_n^* \underline{\mathbf{e}}_1 \boldsymbol{\rho}_0$ , and, finally, we have to evaluate (10.2).

**11. Memory requirements and computational cost.** Let us look at the computational cost of the block Arnoldi process and BLGMRES, and compare it with the cost of solving each system separately.

Here is a table of the *cost* of  $m$  steps of block Arnoldi compared with  $s$  times the cost of  $m$  steps of (unblocked) Arnoldi:

Operations	block Arnoldi	$s$ times Arnoldi
MVS	$m s$	$m s$
SDOTS	$\frac{1}{2}m(m+1)s^2 + \frac{1}{2}m s(s+1)$	$\frac{1}{2}m(m+1)s$
SAXPYS	$\frac{1}{2}m(m+1)s^2 + \frac{1}{2}m s(s+1)$	$\frac{1}{2}m(m+1)s$

Clearly, block Arnoldi is more costly than  $s$  times Arnoldi, but the number of MVs is the same. This means that using expensive MVs and preconditioners (such as multigrid or other inner iterations) is fine for BLGMRES.

The next table shows the *storage requirements* of  $m$  steps of block Arnoldi compared with those of  $m$  steps of (unblocked) Arnoldi (applied once):

	block Arnoldi	Arnoldi
$\mathbf{y}_0, \dots, \mathbf{y}_m$	$(m+1)sN$	$(m+1)N$
$\boldsymbol{\rho}_0, \underline{\mathbf{H}}_m$	$\frac{1}{2}s(s+1) + \frac{1}{2}ms(ms+1) + ms^2$	$1 + \frac{1}{2}m(m+1) + m$

If we apply (unblocked) Arnoldi  $s$  times, we can always use the same memory if the resulting orthonormal basis and Hessenberg matrix need not be stored. However, if we distribute the  $s$  columns of  $\mathbf{y}_0$  on  $s$  processors with attached (distributed) memory, then block Arnoldi requires a lot of communication.

The *extra cost* of  $m$  steps of BLGMRES on top of block Arnoldi compared with  $s$  times the extra cost of  $m$  steps of GMRES is given in the following table:

Operations	BLGMRES	$s$ times GMRES
MVS	$s$	$s$
SAXPYS	$m s^2$	$m s$
scalar work	$\mathcal{O}(m^2 s^3)$	$\mathcal{O}(m^2 s)$

In particular, in the  $(k+1)$ th block step we have to apply first  $k s^2$  Givens rotations to the  $k$ th block column (of size  $(k+1)s \times s$ ) of  $\underline{\mathbf{H}}_m$ , which requires  $\mathcal{O}(k s^3)$  operations. Summing up over  $k$  yields  $\mathcal{O}(m^2 s^3)$  operations. Moreover,  $s$  times back substitution with a triangular  $m s \times m s$  matrix requires also  $\mathcal{O}(m^2 s^3)$  operations.

Let us summarize these numbers and give the *mean cost per iteration* of BLGMRES compared with  $s$  times the mean cost per iteration of (unblocked) GMRES:

Operations	BLGMRES	$s \times$ GMRES	factor
MVS	$(1 + \frac{1}{m})s$	$(1 + \frac{1}{m})s$	1
SDOTS	$\frac{1}{2}(m+1)s^2 + \frac{1}{2}s(s+1)$	$\frac{1}{2}(m+1)s$	$s + \frac{s+1}{m+1}$
SAXPYS	$\frac{1}{2}(m+3)s^2 + \frac{1}{2}s(s+1)$	$\frac{1}{2}(m+3)s$	$s + \frac{s+1}{m+3}$
scalar work	$\mathcal{O}(m s^3)$	$\mathcal{O}(m s)$	$\mathcal{O}(s^2)$

Recall that the most important point in the comparison of block and ordinary Krylov space solvers is that the dimensions of the search spaces  $\mathcal{B}_n$  and  $\mathcal{K}_n$  differ by a factor of up to  $s$ . This could mean that BLGMRES *may converge in roughly  $s$  times fewer iterations* than GMRES. Ideally, this might even be true if we choose

$$m_{\text{BLGMRES}} := \frac{1}{s} m_{\text{GMRES}}. \quad (11.1)$$

This assumption would make the memory requirement of both methods comparable. So, finally, let us compare the *costs till convergence* and the *storage requirements* assuming that either

- a) BLGMRES( $m$ ) converges in a total of  $s$  times fewer (inner) iterations than GMRES if both use the same  $m$ , or,  
b) BLGMRES( $m$ ) converges in a total of  $s$  times fewer (inner) iterations than GMRES even if (11.1) holds.

	cost factor per iter. (same $m$ )	cost factor till conv. ass. a)	cost factor till conv. ass. b)
MVS	1	$\frac{1}{s}$	$\approx \frac{1}{s}$
SDOTS	$s + \frac{s+1}{m+1}$	$\approx 1 + \frac{1}{m}$	$\approx \frac{1}{s} + \frac{1}{m}$
SAXPYS	$s + \frac{s+1}{m+1}$	$\approx 1 + \frac{1}{m}$	$\approx \frac{1}{s} + \frac{1}{m}$
scalar work	$\mathcal{O}(s^2)$	$\mathcal{O}(s)$	$\mathcal{O}(1)$
storage of $\mathbf{y}_0, \dots, \mathbf{y}_m$		$s$	1
storage of $\boldsymbol{\rho}_0, \underline{\mathbf{H}}_m$		$\approx s^2$	$\approx 1$

Unfortunately our numerical experiments indicate that it is rare to fully gain the cost factors of assumption a), and even more so to gain those of b).

**12. Initial deflation and the reconstruction of the full solution.** If the initial block residual  $\mathbf{r}_0$  is rank deficient, we can reduce from the beginning the number of systems that have to be solved. We call this reduction **initial deflation**. Instead of the given systems we will solve fewer other systems. They need not to be a subset of the given ones, but their initial residuals must span the same space as the original initial residuals did.

Detecting linear dependencies among the columns of  $\mathbf{r}_0$  is possible by QR factorization via a modified Gram-Schmidt process that allows for column pivoting. The QR factorization provides moreover an orthonormal basis of the column space of  $\mathbf{r}_0$ . In practice, we need to be able to handle near-dependencies. For treating them there are various more sophisticated rank-revealing QR factorizations, which naturally depend on some deflation tolerance  $\text{tol}$ ; see [2, 3, 4, 5, 6, 8, 13].

For conformity with possible later deflation steps we set here  $\tilde{\mathbf{y}}_0 := \mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$ , and we denote the numerical rank of  $\tilde{\mathbf{y}}$  by  $s_0 \leq s$ . If inequality holds, the columns of  $\boldsymbol{\rho}_0$  in (10.3) would be linearly dependent too, and, therefore also the RHSs  $\mathbf{e}_1 \boldsymbol{\rho}_0$  of the least-squares problems (10.8) for the block quasi-residual  $\mathbf{q}_n = \mathbf{e}_1 \boldsymbol{\rho}_0 - \underline{\mathbf{H}}_n \mathbf{k}_n$  defined in (10.3). Consequently, the columns of the solution  $\mathbf{k}_n$  would be as well linearly dependent, and so would be those of  $\mathbf{x}_n - \mathbf{x}_0 = \mathbf{Y}_n \mathbf{k}_n$ . We want to profit from this situation and only compute a linearly independent subset of these columns.

We write the QR decomposition with pivoting of  $\tilde{\mathbf{y}}_0$  as

$$\tilde{\mathbf{y}}_0 =: \begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_0^\Delta \end{pmatrix} \begin{pmatrix} \boldsymbol{\rho}_0 & \boldsymbol{\rho}_0^\square \\ \mathbf{o} & \boldsymbol{\rho}_0^\Delta \end{pmatrix} \boldsymbol{\pi}_0^\top =: \begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_0^\Delta \end{pmatrix} \begin{pmatrix} \boldsymbol{\eta}_0 \\ \boldsymbol{\eta}_0^\Delta \end{pmatrix}, \quad (12.1)$$

where

- $\boldsymbol{\pi}_0$  is an  $s \times s$  permutation matrix,
- $\begin{pmatrix} \mathbf{y}_0 & \mathbf{y}_0^\Delta \end{pmatrix}$  is  $N \times s$  with orthonormal columns,
- $\mathbf{y}_0$  is  $N \times s_0$  and will go into the basis,
- $\mathbf{y}_0^\Delta$  is  $N \times (s - s_0)$  and will be deflated,
- $\boldsymbol{\rho}_0$  is  $s_0 \times s_0$  upper triangular and nonsingular,
- $\boldsymbol{\rho}_0^\Delta$  is  $(s - s_0) \times (s - s_0)$  upper triangular and small:  $\|\boldsymbol{\rho}_0^\Delta\|_F < \sqrt{s - s_0} \text{tol}$ ,
- $\boldsymbol{\eta}_0 := \begin{pmatrix} \boldsymbol{\rho}_0 & \boldsymbol{\rho}_0^\square \end{pmatrix} \boldsymbol{\pi}_0^\top$  is  $s_0 \times s$  and of full rank  $s_0$ ,
- $\boldsymbol{\eta}_0^\Delta := \begin{pmatrix} \mathbf{o} & \boldsymbol{\rho}_0^\Delta \end{pmatrix} \boldsymbol{\pi}_0^\top$  is  $(s - s_0) \times s$  and small:  $\|\boldsymbol{\eta}_0^\Delta\|_F < \sqrt{s - s_0} \text{tol}$ .

We split the permutation matrix  $\pi_0$  of (12.1) up into an  $s \times s_0$  matrix  $\pi_0^\square$  and an  $s \times (s - s_0)$  matrix  $\pi_0^\Delta$ , so that

$$\begin{aligned}\tilde{\mathbf{y}}_0 \pi_0 &= \mathbf{r}_0 \left( \pi_0^\square \mid \pi_0^\Delta \right) = \left( \mathbf{r}_0 \pi_0^\square \mid \mathbf{r}_0 \pi_0^\Delta \right) \\ &= \left( \mathbf{b} \pi_0^\square \mid \mathbf{b} \pi_0^\Delta \right) - \mathbf{A} \left( \mathbf{x}_0 \pi_0^\square \mid \mathbf{x}_0 \pi_0^\Delta \right),\end{aligned}$$

while, by (12.1),

$$\tilde{\mathbf{y}}_0 \pi_0 = \left( \mathbf{y}_0 \boldsymbol{\rho}_0 \mid \mathbf{y}_0 \boldsymbol{\rho}_0^\square + \mathbf{y}_0^\Delta \boldsymbol{\rho}_0^\Delta \right).$$

So the second blocks always contain the quantities for the systems that have been deflated (that is, put aside) and the first those for the remaining ones. Equating the two expressions for  $\tilde{\mathbf{y}}_0 \pi_0$  and multiplying by  $\mathbf{A}^{-1}$  we obtain, separately for the left and the right part of the block vector,

$$\mathbf{A}^{-1} \mathbf{y}_0 \boldsymbol{\rho}_0 = \underbrace{\mathbf{A}^{-1} \mathbf{b} \pi_0^\square}_{=: \mathbf{x}_*^\square} - \underbrace{\mathbf{x}_0 \pi_0^\square}_{=: \mathbf{x}_0^\square} \quad (12.2)$$

and

$$\underbrace{\mathbf{A}^{-1} \mathbf{b} \pi_0^\Delta}_{=: \mathbf{x}_*^\Delta} = \underbrace{\mathbf{x}_0 \pi_0^\Delta}_{=: \mathbf{x}_0^\Delta} + (\mathbf{A}^{-1} \mathbf{y}_0 \boldsymbol{\rho}_0) \left( \boldsymbol{\rho}_0^{-1} \boldsymbol{\rho}_0^\square \right) + \mathbf{A}^{-1} \mathbf{y}_0^\Delta \boldsymbol{\rho}_0^\Delta. \quad (12.3)$$

Note that here the newly introduced quantities  $\mathbf{x}_0^\square$ ,  $\mathbf{x}_0^\Delta$ ,  $\mathbf{x}_*^\square$ , and  $\mathbf{x}_*^\Delta$  contain the non-deleted and the deleted columns of the initial approximation and the exact solution, respectively. Likewise,  $\mathbf{x}_n^\square$  and  $\mathbf{x}_n^\Delta$  will contain the undeleted and the deleted columns of the current approximation, the first being found directly by deflated BLGMRES, the other to be computed from the first, as explained next.

In the last term of (12.3)  $\boldsymbol{\rho}_0^\Delta = \mathbf{o}$  if  $\text{rank } \mathbf{r}_0 = s_0$  exactly. Otherwise, this term is small if  $\mathbf{A}$  is far from singular: in view of  $\|\mathbf{A}^{-1}x\|_2 \leq \|\mathbf{A}^{-1}\|_2 \|x\|_2$  for  $x \in \mathbb{C}^N$  we have  $\|\mathbf{A}^{-1}\mathbf{B}\|_F \leq \|\mathbf{A}^{-1}\|_2 \|\mathbf{B}\|_F$ ; moreover,

$$\|\mathbf{y}_0^\Delta \boldsymbol{\rho}_0^\Delta\|_F \leq \|\mathbf{y}_0^\Delta\|_F \|\boldsymbol{\rho}_0^\Delta\|_F, \quad \|\mathbf{y}_0^\Delta\|_F \leq \sqrt{s - s_0}, \quad \|\boldsymbol{\rho}_0^\Delta\|_F \leq \sqrt{s - s_0} \text{tol},$$

so in total we have

$$\|\mathbf{A}^{-1} \mathbf{y}_0^\Delta \boldsymbol{\rho}_0^\Delta\|_F \leq \|\mathbf{A}^{-1}\|_2 \|\mathbf{y}_0^\Delta\|_F \|\boldsymbol{\rho}_0^\Delta\|_F \leq \|\mathbf{A}^{-1}\|_2 (s - s_0) \text{tol}. \quad (12.4)$$

This term will be neglected. In the second term on the right-hand side of (12.3) we insert the expression for  $\mathbf{A}^{-1} \mathbf{y}_0 \boldsymbol{\rho}_0$  obtained from (12.2) after replacing there  $\mathbf{x}_*^\square$  by its current approximation  $\mathbf{x}_n^\square$ . Then we consider the resulting two terms as the current approximation  $\mathbf{x}_n^\Delta$  of  $\mathbf{x}_*^\Delta = \mathbf{A}^{-1} \mathbf{b} \pi_0^\Delta$ :

$$\boxed{\mathbf{x}_n^\Delta := \mathbf{x}_0^\Delta + \left( \mathbf{x}_n^\square - \mathbf{x}_0^\square \right) \left( \boldsymbol{\rho}_0^{-1} \boldsymbol{\rho}_0^\square \right)}. \quad (12.5)$$

So, in case of a (typically only approximate) linear dependence of columns of  $\mathbf{r}_0$  (either at the beginning or at a restart of BLGMRES), we can express some of the columns (stored in  $\mathbf{x}_n^\Delta$ ) of the  $n$ th approximant of the block solution in terms of the other columns (stored in  $\mathbf{x}_n^\square$ ) and the  $s_0 \times (s - s_0)$  matrix  $\boldsymbol{\rho}_0^{-1} \boldsymbol{\rho}_0^\Delta$ .

**13. The block Arnoldi process with deflation.** The  $m$  steps of the block Arnoldi process of Algorithm 9.1 are only fully correct as long as the columns of  $\tilde{\mathbf{y}}_0$  in the first QR factorization and those of  $\tilde{\mathbf{y}}$  in the second QR factorization remain linearly independent. Otherwise, we need to detect such linear dependencies and delete

columns so that only an independent subset that spans the same space is kept. This is again deflation. In fact, the case of a rank-deficient  $\tilde{\mathbf{y}}_0$  is exactly what we have treated in the previous section as initial deflation. The other case, when rank-deficiency is detected in the QR factorization of  $\tilde{\mathbf{y}}$  in the loop of Algorithm 9.1, may be called **Arnoldi deflation**. Since due to the limitations in memory space BLGMRES is typically restarted often (i.e.,  $m$  is small), initial deflation is more common than Arnoldi deflation. In fact, BLGMRES seems to work quite well without the implementation of Arnoldi deflation — but this is not true for comparable situations in other block Krylov space solvers.

Initial deflation reduces the size  $s \times s$  of all blocks of  $\underline{\mathbf{H}}_m$  to, say,  $s_0 \times s_0$  if rank  $\mathbf{r}_0 = s_0$ . If Arnoldi deflation is implemented too, the diagonal block  $\boldsymbol{\eta}_{n,n}$  may further reduce to  $s_n \times s_n$ , where  $s_{n-1} \geq s_n \geq s_{n+1}$ , so the size of  $\underline{\mathbf{H}}_m$  reduces to

$$t_{m+1} \times t_m, \quad \text{where } t_m := \sum_{k=0}^{m-1} s_k.$$

Implementing this Arnoldi deflation does not cause serious difficulties in BLGMRES, but, as mentioned, its benefits are limited when the restart period  $m$  is small.

So,  $\underline{\mathbf{H}}_m$  is defined as in (9.4), but the blocks are no longer all of the same size. Additionally, we let

$$\underline{\mathbf{H}}_m^\Delta := \begin{pmatrix} \mathbf{o} & \mathbf{o} & \cdots & \mathbf{o} \\ \boldsymbol{\eta}_1^\Delta & \mathbf{o} & \cdots & \mathbf{o} \\ & \boldsymbol{\eta}_2^\Delta & \ddots & \vdots \\ & & \ddots & \mathbf{o} \\ & & & \boldsymbol{\eta}_m^\Delta \end{pmatrix}, \quad (13.1)$$

where the entry  $\boldsymbol{\eta}_n^\Delta$  comes from a rank-revealing QR factorization analogous to (12.1) in the  $n$ th step of the loop in the block Arnoldi process, so reflects an Arnoldi deflation. Note that the blocks of  $\underline{\mathbf{H}}_m^\Delta$  are as wide as those of  $\underline{\mathbf{H}}_m$ , but typically much less high, as the height  $s_{n-1} - s_n$  of  $\boldsymbol{\eta}_n^\Delta$  is equal to the number of columns deflated in the  $n$ th step. The zero block row vector at the top is of height  $s - s_0$ , equal to the number of columns deflated in the initialization step. In total,  $\underline{\mathbf{H}}_m^\Delta$  has size  $(s - s_m) \times t_m$ . Note that  $s - s_m$  is the total number of deflations, including those in the initialization step. If we further let

$$\mathbf{Y}_n := (\mathbf{y}_0 \quad \mathbf{y}_1 \quad \cdots \quad \mathbf{y}_{n-1}), \quad \mathbf{Y}_n^\Delta := (\mathbf{y}_0^\Delta \quad \mathbf{y}_1^\Delta \quad \cdots \quad \mathbf{y}_{n-1}^\Delta),$$

we see that when deflation occurs the block Arnoldi relation (9.5) is replaced by

$$\boxed{\mathbf{A}\mathbf{Y}_m = \mathbf{Y}_{m+1}\underline{\mathbf{H}}_m + \mathbf{Y}_{m+1}^\Delta\underline{\mathbf{H}}_m^\Delta.} \quad (13.2)$$

Here,  $\mathbf{Y}_n$  is of size  $N \times t_n$  and  $\mathbf{Y}_{n+1}^\Delta$  is of size  $N \times (s - s_n)$ . Recall that  $\mathbf{Y}_{m+1}$  has orthonormal columns; but, in general, those of  $\mathbf{Y}_{m+1}^\Delta$  are not orthogonal if they belong to different blocks  $\mathbf{y}_k^\Delta$ ; they are just of 2-norm 1. If only exact deflations occur,  $\underline{\mathbf{H}}_m^\Delta = \mathbf{O}$ , but  $\underline{\mathbf{H}}_m$  has still blocks of different size.

**14. Block GMRES with deflation.** Deflation in BLGMRES is occurring in the block Arnoldi process, either as initial deflation (treated in Section 12) or as Arnoldi deflation (treated in Section 13). Neither the deflated (deleted) columns  $\mathbf{y}_k^\Delta$  of  $\mathbf{Y}_{m+1}^\Delta$  nor the matrix  $\underline{\mathbf{H}}_m^\Delta$  in (13.2) are used further, so the lower banded matrix  $\underline{\mathbf{H}}_m$  gets processed as before: while it is block-wise constructed it gets block-wise QR factorized by a sequence of Givens or Householder transformation. Up to a deflation error, the block residual norm is given and controlled by  $\|\mathbf{q}_n\|_F$  from (10.11), which

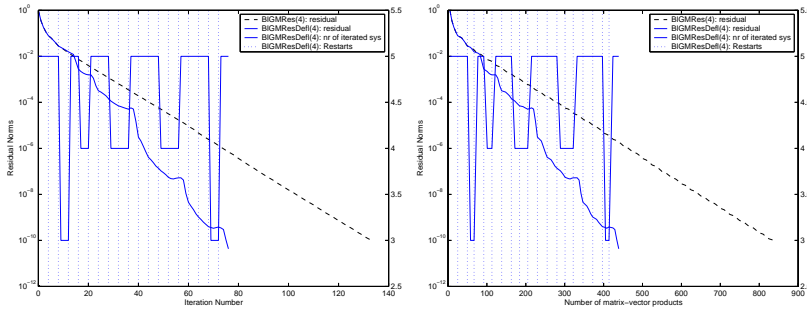


Fig. 15.1: Block GMRES: Laplace on  $10 \times 10$  grid ( $n = 100$ ), five RHSs chosen as first five unit vectors

now should be called  $\|\mathbf{q}_n^\square\|_F$ . Once it is small enough, or if  $n = m$  and a restart is due, the coordinates  $\mathbf{k}_n$  given by (10.10), which now should be denoted by  $\mathbf{k}_n^\square$ , are used to compute  $\mathbf{x}_n^\square$ , and then (12.5) is used to construct  $\mathbf{x}_n^\Delta$  and the current full block approximation

$$\mathbf{x}_n = \begin{pmatrix} \mathbf{x}_n^\square & \mathbf{x}_0^\Delta \end{pmatrix} \boldsymbol{\pi}_0^\top \quad (14.1)$$

of the block solution  $\mathbf{x}_* := \mathbf{A}^{-1}\mathbf{b}$ . We suggest to explicitly compute at this point the current block residual  $\mathbf{r}_n := \mathbf{b} - \mathbf{A}\mathbf{x}_n$  in order to check the size of the residual and, if necessary, to restart BLGMRES at the actual approximation  $\mathbf{x}_n$ . This means in particular to recompute the rank-revealing QR factorization for the tentative initial deflation. Of course, here we are free to check for the maximum of the 2-norms of the residuals instead of the F-norm of the block residual.

**15. A Numerical Experiment.** The following very simple numerical experiments illustrate the behavior of BLGMRES with optional initial deflation at each restart. Arnoldi deflation was also implemented, but never became active.

EXPERIMENT 1. The matrix  $\mathbf{A}$  comes from the 5-point stencil finite difference method discretization of the Laplace operator on a  $10 \times 10$  grid, so  $N = 100$  and  $s := 5$ . Moreover, we let the restart interval be just  $m := 4$ . As RHSs we choose either  $\mathbf{b} := (\mathbf{e}_1 \ \dots \ \mathbf{e}_s)$  or  $\mathbf{b} :=$  random rank-one matrix  $\mathbf{b}^1 + 10^{-3} \times$  random  $N \times s$  matrix  $\mathbf{b}^s$ .

We compare:

- **B1MRes(m)**: a straightforward BLGMRES implementation using MATLAB's `qr` without pivoting and with no deflation implemented.
- **B1MResDef1(m)**: BLGMRES using rank-revealing QR (the Chan/Foster high-rank-revealing QR implemented as `hrrqr` in Hansen's UTV package with tolerance set to 0.005) and deflation (both "initial" and "Arnoldi"); at each restart we check all  $s$  residuals for size and linear dependence.

The results for the first RHSs are shown in Figures 15.1, those for the second RHSs in Figures 15.2. The plots show

- the maximum of the 2-norms of all  $s$  residuals on the y-axis in log scale,
- the actual number of RHSs treated ( $s_n$ ) on the y-axis at right,
- either the iteration number  $n$  or the number of MVs on x-axis.

The dotted vertical lines indicate the restarts.

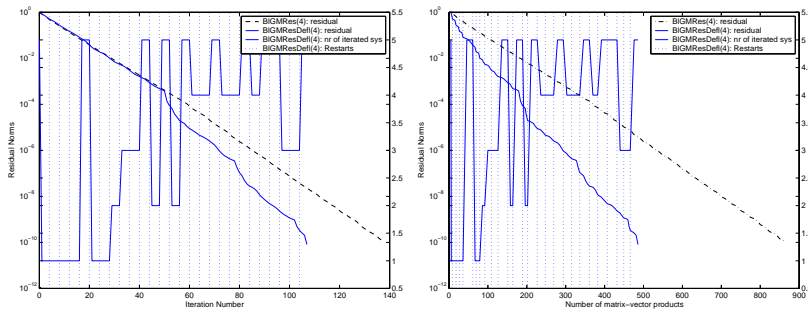


Fig. 15.2: Block GMRES: Laplace on  $10 \times 10$  grid ( $n = 100$ ), five RHSs chosen nearly linearly dependent

We see that with deflation we need in both examples only about half as many MVS as without deflation. Interestingly, this effect is not stronger in the second example, where the RHSs are very close to each other. In fact this closeness of the RHSs has a strong effect in the initial phase, where four of the five RHSs are deflated, but after iteration 56, where the residual norm is about  $10^{-5}$ , most iterations require again four or five RHSs. This is not so much of a surprise: the given RHSs are close to each other on a coarse scale, but not on a fine one. If we subtract the rank-one matrix  $\mathbf{b}^1$  from  $\mathbf{b}$ , what is left over is a small but full-rank matrix  $10^{-3}\mathbf{b}^s$ . If we write  $\mathbf{Ax} = \mathbf{b}$  as  $\mathbf{A}(\mathbf{x}^1 + 10^{-3}\mathbf{x}^s) = \mathbf{b}^1 + 10^{-3}\mathbf{b}^s$ , we see that once we have solved  $\mathbf{Ax}^1 = \mathbf{b}^1$ , there remains to solve  $\mathbf{Ax}^s = \mathbf{b}^s$ , and this is a linear system with  $s$  linearly independent RHSs.

*Acknowledgment.* The author would like to thank Stefan Röllin for the permission to reproduce Tables 4.1 and 4.2 as well as Figure 4.1, and for providing the background information on these numerical results. He is also indebted to Thomas Schmelzer for various suggestions and a proof of Lemma 7.

## REFERENCES

- [1] J. I. ALIAGA, D. L. BOLEY, R. W. FREUND, AND V. HERNÁNDEZ, *A Lanczos-type method for multiple starting vectors*, Math. Comp., 69 (2000), pp. 1577–1601.
- [2] C. H. BISCHOF AND P. C. HANSEN, *Structure-preserving and rank-revealing QR-factorizations*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1332–1350.
- [3] ———, *A block algorithm for computing rank-revealing QR factorizations*, Numer. Algorithms, 2 (1992), pp. 371–391.
- [4] C. H. BISCHOF AND G. QUINTANA-ORTÍ, *Computing rank-revealing QR factorizations of dense matrices*, ACM Trans. Math. Software, 24 (1998), pp. 226–253.
- [5] T. F. CHAN, *Rank revealing QR factorizations*, Linear Algebra Appl., 88/89 (1987), pp. 67–82.
- [6] T. F. CHAN AND P. C. HANSEN, *Some applications of the rank revealing QR factorization*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 727–741.
- [7] T. F. CHAN AND W. L. WAN, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM J. Sci. Comput., 18 (1997), pp. 1698–1721.
- [8] S. CHANDRASEKARAN AND I. C. F. IPSEN, *On rank-revealing factorisations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 592–622.
- [9] J. CULLUM AND T. ZHANG, *Two-sided Arnoldi and nonsymmetric Lanczos algorithms*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 303–319.
- [10] J. K. CULLUM AND R. A. WILLOUGHBY, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations (2 Vols.)*, Birkhäuser, Boston-Basel-Stuttgart, 1985.
- [11] I. S. DUFF, A. M. ERISMAN, AND J. K. REID, *Direct methods for sparse matrices*, Monographs on Numerical Analysis, The Clarendon Press Oxford University Press, New York, 1986.
- [12] I. S. DUFF AND J. KOSTER, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996 (electronic).
- [13] W. B. GRAGG AND L. REICHEL, *Algorithm 686: FORTRAN subroutines for updating the QR decomposition*, ACM Trans. Math. Software, 16 (1990), pp. 369–377.

- [14] M. H. GUTKNECHT, *Variants of BiCGStab for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993), pp. 1020–1033.
- [15] K. JBLOU, A. MESSAOUDI, AND H. SADOK, *Global FOM and GMRES algorithms for matrix equations*, Appl. Numer. Math., 31 (1999), pp. 49–63.
- [16] G. MEURANT, *Computer solution of large linear systems*, vol. 28 of Studies in Mathematics and its Applications, North-Holland, Amsterdam, 1999.
- [17] A. A. NIKISHIN AND A. Y. YEREMIN, *Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: General iterative scheme*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1135–1153.
- [18] B. N. PARLETT, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 323–346.
- [19] S. K. RÖLLIN, *Parallel iterative solvers in computational electronics*, PhD thesis, Diss. No. 15859, ETH Zurich, Zurich, Switzerland, 2005.
- [20] A. RUHE, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices*, Math. Comp., 33 (1979), pp. 680–687.
- [21] H. RUTISHAUSER, *On Jacobi rotation patterns*, Proc. Symposia in Appl. Math., 15 (1963), pp. 219–239.
- [22] Y. SAAD, *On the Lánczos method for solving symmetric linear systems with several right-hand sides*, Math. Comp., 48 (1987), pp. 651–662.
- [23] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
- [24] Y. SAAD AND M. H. SCHULTZ, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems*, Math. Comp., 44 (1985), pp. 417–424.
- [25] Y. SAAD, M. YEUNG, J. ERHEL, AND F. GUYOMARC’H, *A deflated version of the conjugate gradient algorithm*, SIAM J. Sci. Comput., 21 (2000), pp. 1909–1926 (electronic).
- [26] O. SCHENK, *Scalable Parallel Sparse LU Factorization Methods on Shared Memory Multiprocessors*, PhD thesis, Diss. No. 13515, ETH Zurich, Zurich, Switzerland, 2000.
- [27] O. SCHENK AND K. GÄRTNER, *Solving unsymmetric sparse systems of linear equations with PARDISO*, J. Future Generation Computer Systems, 20 (2004), pp. 475–487.
- [28] T. SCHMELZER, *Block Krylov methods for Hermitian linear systems*. Diploma thesis, Department of Mathematics, University of Kaiserslautern, Germany, 2004.
- [29] T. SCHMELZER AND M. H. GUTKNECHT, *A QR-decomposition of block tridiagonal matrices generated by the block Lanczos process*, in Proceedings of the 17th IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation, Paris, France, July 11–15, 2005, 2005. CD-ROM.
- [30] V. SIMONCINI AND E. GALLOPOULOS, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM J. Sci. Comput., 16 (1995), pp. 917–933.
- [31] G. L. G. SLEIJPEN AND D. R. FOKKEMA, *BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum*, Electronic Trans. Numer. Anal., 1 (1993), pp. 11–32.
- [32] C. SMITH, *The performance of preconditioned iterative methods in computational electromagnetics*, PhD thesis, Dept. of Electrical Engineering, Univ. of Illinois at Urbana-Champaign, 1987.
- [33] H. A. VAN DER VORST, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
- [34] B. VITAL, *Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur*, PhD thesis, Université de Rennes, 1990.