



HAL
open science

Vary the s in Your s -step GMRES

David Imberti, Jocelyne Erhel

► **To cite this version:**

David Imberti, Jocelyne Erhel. Vary the s in Your s -step GMRES. Electronic Transactions on Numerical Analysis, In press. hal-01299652v2

HAL Id: hal-01299652

<https://inria.hal.science/hal-01299652v2>

Submitted on 7 Jul 2016 (v2), last revised 7 Dec 2017 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Vary the s in Your s -step GMRES

David Imberti ^{*}, Jocelyne Erhel [†]

July 7, 2016

Abstract

Krylov methods are commonly used iterative methods for solving large sparse linear systems, however they suffer communication bottlenecks on parallel computers. Therefore, Communication-Avoiding methods have been developed where the Krylov subspace is built block by block, so that s matrix-vector multiplications can be done before the orthonormalization operation.

This paper introduces a new variation on s -step GMRES in order to improve its stability and to reduce the number of iterations necessary to ensure convergence, with a small overhead in communications. Namely, we develop an adaptive s -step GMRES algorithm, where the block size is variable and increases gradually. Our numerical experiments show a good agreement with our stability analysis and demonstrate the efficiency of our variable s -step approach.

Key words. Communication-Avoiding, s -step Krylov method, GMRES algorithm, variable s -step.

AMS subject classifications. 65F10, 65N22

1 Introduction

Many computational problems need to solve a large linear system $Ax = b$. Because linear solvers can be quite time consuming, they require an efficient implementation for supercomputers. An important class of methods is based on Krylov subspace methods like GMRES [30]. In this paper, we aim to improve the parallelization of such methods for general sparse matrices. Parallel GMRES algorithms have been studied by several authors, for example [1, 4, 7, 10, 13, 17, 21, 24, 27].

In Figure 1, we plot a tree, each branch of which describes a further subset of GMRES algorithms. In the course of this paper, we will briefly describe each branch, and the one we eventually take, by discussing each level of this tree from the root node down (a breadth-first traversal through GMRES). Some parallelism can be found in the Arnoldi process of the usual restarted GMRES(m) algorithm, where m is the restarting parameter [3, 8, 12, 31]. However, global communications prevent from high performance with many processors. Another way to build an orthonormal basis is to compute first a Krylov basis of size m , where m matrix vector multiplications can be done in parallel, and to orthogonalize afterwards. We denote this algorithm as an m -step GMRES algorithm [1, 4, 7, 10, 13, 21]. However, if m is large, the Krylov basis may become ill-conditioned. Therefore, s -steps methods have been defined, where the Krylov basis is built block by block, with blocks of size s , allowing s parallel matrix vector multiplications and a better conditioned Krylov basis [17, 11].

We propose a new variation of m -step and s -step GMRES algorithm, where the subspace condition avoids solving a triangular system. We denote the approach with the inverse of a triangular matrix as the 'Traditional' approach in Figure 1, while we propose a new 'Non-Traditional' branch.

Both methods are mathematically equivalent and seem to have similar numerical behaviour, as illustrated in our numerical experiments.

^{*}INRIA, Campus universitaire de Beaulieu, 35042 Rennes Cedex, France

[†]INRIA, Campus universitaire de Beaulieu, 35042 Rennes Cedex, France

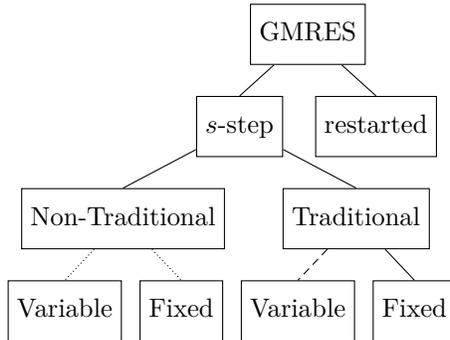


Figure 1: Variants of GMRES.

One key motivation to develop this 'Non-Traditional' branch is simplicity. From the 'Non-Traditional' s -step version, we easily derive a new algorithm, where the block size s is allowed to vary. It could be noted that a variable method could also be derived from the 'Traditional' branch, but we did not explore this potential branch for the sake of simplicity (therefore we use a dashed line in Figure 1). Indeed, it is much easier to vary the block size when there is no triangular system to solve. It can also be noted that variable s -steps methods have been designed independently for other Krylov methods [5].

The main novelty of this paper is thus the variable s -step GMRES algorithm, using a 'Non-Traditional' version of the subspace condition. We use dotted lines in Figure 1 to show the two algorithms described in this paper: 'Non-Traditional' fixed and variable s -step GMRES.

For both algorithms, we investigate numerical stability issues related to convergence rate, as well as communication issues. We suggest to use a Fibonacci sequence for the variable block size, in order to get a rapidly increasing block size, balancing the convergence rate and the communications overhead.

The paper is organized by first discussing each level of the tree in Figure 1. In section 2, we give a brief background to restarted GMRES and Traditional m -step GMRES (level 1). Then we introduce the Non-Traditional variant and compare both approaches for m -step GMRES (level 2).

In section 3, we define the Non-Traditional fixed s -step GMRES algorithm and derive our variable s -step algorithm (level 3).

Then we compare in section 4 convergence and communication issues of both algorithms. Since convergence is related to the condition numbers of the blocks used to build the Krylov basis, we prove lower bounds for these condition numbers. Regarding communication issues, we show that for a large ratio m/s , the numbers of messages are of the same order.

Our numerical experiments in section 5 demonstrate the efficiency of a variable block size compared to a fixed block size. We observe a faster convergence, which is closely related to the condition numbers of the blocks.

Finally, we conclude in section 6.

2 Traditional and Non-Traditional m -step GMRES

2.1 Restarted GMRES

We first recall the GMRES algorithm to build upon and recall m -step GMRES.

Let $Ax = b$ be a linear system, with A a large sparse nonsymmetric nonsingular matrix of size n .

We introduce the Krylov subspace

$$K_m = \text{span}\{r_0, Ar_0, A^2r_0, \dots, A^{m-1}r_0\}$$

with the residual vector defined as $r_0 = b - Ax_0$ for some chosen initial vector x_0 .

We also introduce $v_1 = r_0/\beta$, where $\beta = \|r_0\|$.

It is known that performing the Arnoldi process on A and r_0 generates an orthonormal basis,

$$V_{m+1} = \{v_1, \dots, v_{m+1}\}$$

of the Krylov subspace K_{m+1} and a Hessenberg matrix \bar{H}_m , of size $(m+1) \times m$, which satisfy the Arnoldi relation

$$AV_m = V_{m+1}\bar{H}_m. \quad (1)$$

The GMRES algorithm is based on the subspace condition $x_m \in x_0 + K_m$, which can be written

$$x_m = x_0 + V_m y.$$

Using the Arnoldi relation (1), we can say that $r_m = r_0 - AV_m y = V_{m+1}(\beta e_1 - \bar{H}_m y)$, where e_1 is the first column of the identity matrix.

The residual in GMRES satisfies the Galerkin condition $\min_{x \in x_0 + K_m} \|r\|$, which is equivalent to the linear least squares problem

$$\min_y \|\beta e_1 - \bar{H}_m y\|.$$

Restarted GMRES repeats this Arnoldi cycle for a new initial vector by setting $x_0 = x_m$ [18, 20, 23, 29, 30].

The GMRES algorithm may be modified to allow preconditioning, but this will not affect our discussion. We will also assume throughout the paper that the Krylov subspace K_m is of dimension m , so that the residual of minimal norm is unique.

In summary, and to provide a comparison to variable s -step GMRES later, restarted GMRES, denoted by GMRES(m), may be expressed by Algorithm 1.

Algorithm 1 GMRES(m)

```

1: while not converged do
2:    $r_0 = b - Ax_0$ 
3:    $\beta = \|r_0\|$ 
4:    $v_1 = r_0/\beta$ 
5:    $V_1 = \{v_1\}$ 
6:   for  $k=1,m$  do
7:      $w_k = Av_k$ 
8:      $AV_k = V_{k+1}H_k$  obtained from Arnoldi process
9:     solve the least squares problem  $\min_y \|\beta e_1 - \bar{H}_k y\| = \|\beta e_1 - \bar{H}_k \bar{y}_k\|$ 
10:    compute  $x_k = x_0 + V_k \bar{y}_k$ 
11:    test convergence
12:  end for
13:  if not converged then
14:     $x_0 = x_m$ 
15:  end if
16: end while

```

2.2 Traditional m -step GMRES

The purpose of m -step GMRES is to improve the communication efficiency of Algorithm 1. Most of the computational time in Algorithm 1 is spent inside of the Arnoldi loop. Indeed, the least squares problem

in step 9 of Algorithm 1 involves the small upper-Hessenberg matrix \overline{H}_k , therefore it may quickly be solved by a series of Givens rotations.

The Arnoldi process contains two key kernels: matrix-vector products and orthonormalization. To motivate m -step GMRES, we look at the problems involved in parallelizing these kernels.

The Arnoldi process builds an orthonormal basis of the Krylov subspace K_{m+1} with one matrix vector multiplication at a time and orthonormalizes it against the previous vectors as soon as it is added. The specific method for orthonormalization may vary. Regardless, all such methods lead to communication issues due to the global communication necessary in the dot product operation. A classical Gram-Schmidt orthonormalization reduces communication, compared to a modified Gram-Schmidt process, but then this procedure must be performed twice to ensure numerical stability [14]. Another possibility is to use Householder transformations, but this is more computationally intensive [35]. Parallelism in Gram-Schmidt process can be added by using a block Householder or QR method instead [19].

Similarly, a Krylov basis can be built before orthonormalizing. The first step becomes merely a succession of m matrix-vector products (thus the name m -step GMRES) and the second step a block orthonormalization process. These two steps provide much more parallelism by avoiding global communications.

Moreover, a Newton basis can improve stability issues [1, 13, 27, 24] and other basis can also be considered [26]. For the sake of simplicity, we do not consider here such a basis, and leave this for future work.

To detail the m -step GMRES procedure further we introduce

$$W_{m+1} = \{v_1, Av_1, \dots, A^m v_1\}, \quad (2)$$

which is a basis of the Krylov subspace K_{m+1} . We compute an orthonormal basis with a QR factorization:

$$W_{m+1} = V_{m+1}R_{m+1}. \quad (3)$$

The diagonal elements of the upper triangular matrix R_{m+1} are forced to be real positive, so that the QR factorization is unique [15].

Using this, we may further define

$$\begin{aligned} \{v_1, AW_m\} &= W_{m+1}, \\ &= V_{m+1}R_{m+1}, \\ &= V_{m+1}\{e_1, H_m\}, \end{aligned} \quad (4)$$

Note that in equation (4) that the first column of R_{m+1} is e_1 , so H_m may be defined as an upper Hessenberg matrix formed by removing this column. By removing the first column explicitly, we see that

$$AW_m = V_{m+1}H_m, \quad (5)$$

and that

$$AV_m = V_{m+1}(H_m R_m^{-1}). \quad (6)$$

Thus the subspace condition of GMRES can be written as before as $x_m = x_0 + V_m y$ and the Galerkin condition ends up with the least squares problem

$$\min_y \|\beta e_1 - \tilde{H}_m y\|, \quad (7)$$

where $\tilde{H}_m = H_m R_m^{-1}$.

If the Krylov subspace K_{m+1} is of dimension $m+1$, then the residual r_m is uniquely defined by the Galerkin condition, and we can conclude that m -step GMRES and GMRES(m) are mathematically equivalent.

We summarize these steps in Algorithm 2, which we call m -step GMRES.

Algorithm 2 Traditional m -step GMRES

```

1: while not converged do
2:    $r_0 = b - Ax_0$ 
3:    $\beta = \|r_0\|$ 
4:    $v_1 = r_0/\beta$ 
5:    $W_{m+1} = \{v_1, Av_1, \dots, A^{m-1}v_1, A^m v_1\}$ 
6:   QR factorization  $W_{m+1} = V_{m+1}R_{m+1}$ 
7:    $\tilde{H}_m = H_m R_m^{-1}$  where  $H_m$  is  $R_{m+1}$  removing the first column.
8:   solve the least squares problem  $\min_y \|\beta e_1 - \tilde{H}_m y\| = \|\beta e_1 - \tilde{H}_m \tilde{y}_m\|$ 
9:   compute  $x_m = x_0 + V_m \tilde{y}_m$ 
10:  test convergence
11:  if not converged then
12:     $x_0 = x_m$ 
13:  end if
14: end while

```

In practice, Algorithm 2 may overwrite W with V in order to save memory. It contains three main steps: the matrix-vector products in the computation of W , its QR factorization, and the multiplication of H_m by the R_m^{-1} factor. We now turn to consider this R_m^{-1} step, its necessity, and its influence on stability.

2.3 Non-Traditional m -step GMRES

We concern ourselves here with the R_m^{-1} factor, and we design a new version of m -step GMRES method without this factor, comparable with the treatment in [33]. We denote as 'Traditional' Algorithm 2 that uses this R_m^{-1} factor, and we propose a 'Non-Traditional' algorithm without R_m^{-1} .

Since W_m is a basis of the Krylov subspace K_m , the subspace condition can be equivalently written $x_m = x_0 + W_m y$ or, using the QR factorization $W_m = V_m R_m$,

$$x_m = x_0 + V_m R_m y. \tag{8}$$

If we use the relation (5), the Galerkin condition is equivalent to solving the least squares problem

$$\min_y \|\beta e_1 - H_m y\|. \tag{9}$$

Again, this algorithm is mathematically equivalent to GMRES(m), since it satisfies the subspace and Galerkin conditions. Here too, this approach may overwrite W with V in order to save memory. We summarize this new version of m -step GMRES in Algorithm 3.

Algorithm 3 Non-Traditional m -step GMRES

```
1: while not converged do
2:    $r_0 = b - Ax_0$ 
3:    $\beta = \|r_0\|$ 
4:    $v_1 = r_0/\beta$ 
5:    $W_{m+1} = \{v_1, Av_1, \dots, A^{m-1}v_1, A^m v_1\}$ 
6:    $QR$  factorization  $W_{m+1} = V_{m+1}R_{m+1}$ 
7:    $H_m = R_{m+1}$  without the first column.
8:   solve the least squares problem  $\min_y \|\beta e_1 - H_m y\| = \|\beta e_1 - H_m y_m\|$ 
9:   compute  $x_m = x_0 + V_m R_m y_m$ 
10:  test convergence
11:  if not converged then
12:     $x_0 = x_m$ 
13:  end if
14: end while
```

2.4 Comparison of the two m -step GMRES algorithms

We now compare our Non-Traditional approach to the Traditional one. In order to simplify our analysis, we focus on steps 8 and 9 of both algorithms.

Algorithm 2 creates the solution in the following way:

$$\text{solve } \min_y \|\beta e_1 - H_m R_m^{-1} y\|, x_m = x_0 + V_m \tilde{y}_m, \quad (10)$$

while Algorithm 3 performs the following:

$$\text{solve } \min_y \|\beta e_1 - H_m y\|, x_m = x_0 + V_m R_m y_m. \quad (11)$$

Then, in exact arithmetic, $\tilde{y}_m = R_m y_m$ (recall that R_m is uniquely defined) and the approximate solutions are the same in exact arithmetic.

What matters now is the numerical stability of the two algorithms. On the one hand, there is an inversion step in equation (10), and on the other hand there is a multiplication step in equation (11) (assuming that one uses an orthonormalization technique that is careful to preserve orthonormality [16]). It seems that numerical instabilities in the algorithm might simply swap places from the inversion of R_m to the matrix multiplication by R_m . However, it has been suggested that the inclusion of R_m^{-1} will eventually make the algorithm inherently unstable [33].

Here, we do not argue *a priori* that one method is more stable than the other. Therefore, we base our choice between the two algorithms on the ease of implementation. It will be more apparent as we generalize to a s -step GMRES formulation below, that our Non-Traditional approach is easier to implement. Indeed, we will not need to discuss the details of calculating and determining the R_m^{-1} blocks and the index challenges that occur as a result of that.

We now turn to such a s -step method, using both basis W and V in our Non-Traditional way.

3 Fixed and variable Non-Traditional s -step GMRES

3.1 Fixed s -step GMRES

In the m -step GMRES method, the size of the Krylov basis W_m is equal to the retarding parameter m . However, the condition number of AW_m and thus of H_m increases with m , limiting the restarting

parameter to small values. But quite often convergence could stall if the restarting parameter is too small [24].

Therefore, another approach can be taken where the Krylov orthonormal basis V_{m+1} is built by successively computing blocks B_j of size $s \leq m$ [17]. Each restarting cycle will then be composed of several steps j where a block B_j is added to the basis W and AB_j is orthonormalized, adding a new block to the orthonormal basis V . This should allow taking s sufficiently small for stability issues and m sufficiently large for convergence issues. If s is large enough, parallelism can occur by decoupling matrix-vector multiplications from orthonormalization.

For the sake of simplicity, we assume that m is a multiple of s so that $m = sJ$. We use the Non-Traditional way as described above, as opposed to the Traditional approach characterized by [17]. We compute the first block $B_1 = W_s$ as in equation (2) with the size s instead of m :

$$W_s = \{v_1, Av_1, \dots, A^{s-1}v_1\}, \quad (12)$$

and we perform a QR factorization as in equation (4):

$$\{v_1, AW_s\} = V_{s+1}R_{s+1}. \quad (13)$$

The last vector v_{s+1} of the orthonormal system V_{s+1} will serve as the initial vector of the next block B_2 . We may inductively give a more general definition of W , the orthonormal system V , and the triangular matrix R . At step j , with $2 \leq j \leq J$, we assume that we have built $W_{s(j-1)}$ of size $s(j-1)$ and $V_{s(j-1)+1}$ of size $s(j-1) + 1$, with the last vector $v_{s(j-1)+1}$. We define the block B_j of size s by

$$B_j = \{v_{s(j-1)+1}, Av_{s(j-1)+1}, \dots, A^{s-1}v_{s(j-1)+1}\}, \quad (14)$$

and describe a recursive relation for W_{sj} :

$$W_{sj} = \{W_{s(j-1)}, B_j\}. \quad (15)$$

We now perform a QR factorization

$$\{v_1, AW_{sj}\} = V_{sj+1}R_{sj+1}, \quad (16)$$

and we define the upper Hessenberg matrix H_{sj} as R_{sj+1} without the first column e_1 so that

$$AW_{sj} = V_{sj+1}H_{sj}. \quad (17)$$

In summary, we note that

$$\begin{aligned} W_{sj} &= \{B_1, B_2, \dots, B_j\}, \\ &= \{v_1, Av_1, \dots, A^{s-1}v_1, v_{s+1}, \dots, A^{s-1}v_{s+1}, \dots, v_{s(j-1)+1}, \dots, A^{s-1}v_{s(j-1)+1}\}. \end{aligned} \quad (18)$$

Before deriving the s -step method, we prove that the systems W_{sj} and V_{sj} span the Krylov subspace K_{sj} .

Theorem 1. *We assume that the dimension of K_{m+1} is equal to $m + 1$. Let W_{sj} defined by (15) and V_{sj+1} defined by (16). Then W_{sj} is a basis of K_{sj} and V_{sj+1} is an orthonormal basis of K_{sj+1} .*

Proof. The proof is by induction.

For $j = 1$, since $W_s = \{v_1, \dots, A^{s-1}v_1\}$, it is clear that W_s is a basis of K_s . Also $\{v_1, AW_s\}$ is a basis of K_{s+1} and by (13), V_{s+1} is an orthonormal basis of K_{s+1} .

Now, we assume that $W_{s(j-1)}$ is a basis of $K_{s(j-1)}$ and that $V_{s(j-1)+1}$ is a basis of $K_{s(j-1)+1}$.

Thus $u = v_{s(j-1)+1} \in K_{s(j-1)+1} \setminus K_{s(j-1)}$ and by construction $B_j \in K_{sj}$. Thanks to relation (15) and to the induction assumption, we conclude that $W_{sj} \in K_{sj}$.

Moreover, $u \notin K_{s(j-1)}$ thus the block B_j is linearly independent from the system $W_{s(j-1)}$ and W_{sj} is a basis of K_{sj} which is of dimension sj . Also $Aw_{sj} \in K_{sj+1}$ but $Aw_{sj} \notin K_{sj}$ thus $\{v_1, AW_{sj}\}$ is a basis of K_{sj+1} and by (16), V_{sj+1} is an orthonormal basis of K_{sj+1} . This completes the inductive step. \square

Therefore, the subspace condition at each step j can be written for the Krylov subspace K_{sj} as

$$x = x_0 + W_{sj}y. \quad (19)$$

And, with the Hessenberg matrix H_{sj} from equation (16), the Galerkin condition is again a least squares problem

$$\min_y \|\beta e_1 - H_{sj}y\|. \quad (20)$$

Now, we show how to compute at each step j the new block of s vectors in the orthonormal basis V_{sj+1} and the last s columns of the triangular matrix R_{sj+1} .

We write the QR factorization by blocks as follows:

$$\{v_1, AW_{sj}\} = V_{sj+1}R_{sj+1} = \{v_1, AW_{s(j-1)}, AB_j\}. \quad (21)$$

Since $\{v_1, AW_{s(j-1)}\} = V_{s(j-1)+1}R_{s(j-1)+1}$ from the previous step, we get

$$AB_j = V_{sj+1}S_j, \quad (22)$$

where the matrix S_j of size $(sj+1) \times s$ contains the last s columns of R_{sj+1} .

Therefore, at step j , we orthonormalize AB_j against $V_{s(j-1)+1}$ to get the last s vectors of V_{sj+1} and the last s columns of R_{sj+1} .

We summarize this fixed s -step GMRES, which we call SGMRES(m, s), in Algorithm 4.

Algorithm 4 Non-Traditional Fixed s -step SGMRES(m, s)

```

1: while not converged do
2:    $r_0 = b - Ax_0$ 
3:    $\beta = \|r_0\|$ 
4:    $v_1 = r_0/\beta$ 
5:    $W_0 = \emptyset$  and  $V_1 = \{v_1\}$ 
6:    $J = m/s$ 
7:   for  $j = 1, J$  do
8:      $B_j = \{v_{s(j-1)+1}, Av_{s(j-1)+1}, \dots, A^{s-1}v_{s(j-1)+1}\}$ 
9:      $W_{sj} = \{W_{s(j-1)}, B_j\}$ 
10:    orthonormalize  $AB_j$  against  $V_{s(j-1)+1}$  to obtain the last  $s$  vectors  $\{v_{s(j-1)+2}, \dots, v_{sj+1}\}$  and the
    last  $s$  columns of  $R_{sj+1}$ 
11:    solve the least squares problem  $\min_y \|\beta e_1 - H_{sj}y\| = \|\beta e_1 - H_{sj}y_F\|$ 
12:    compute  $x_{sj} = x_0 + W_{sj}y_F$ 
13:    test convergence
14:   end for
15:   if not converged then
16:      $x_0 = x_{sj}$ 
17:   end if
18: end while

```

In Algorithm 4, the basis W_{sj} is stored for the sake of clarity. However, it is possible to store only V_{sj+1} and R_{sj+1} since it is easy to compute W_{sj} from these matrices. Indeed, thanks to (14) and (22), we may write

$$B_j = V_{sj+1}\{e_{s(j-1)+1}, C_j\}, \quad (23)$$

where $e_{s(j-1)+1}$ is the vector defined by $v_{s(j-1)+1} = V_{sj+1}e_{s(j-1)+1}$ and C_j is the matrix of the first $s-1$ columns of S_j . Then, by induction, it is easy to get a relation between W_{sj} and V_{sj+1} .

One should note that if $s = 1$, one obtains restarted GMRES (Algorithm 1), and if $s = m$, one obtains m -step GMRES, or Algorithm 3. Thus, Algorithm 4 represents a generalization of GMRES algorithms. It is noteworthy to recall that a Traditional fixed s -step approach is possible [17], but requires intensive care in the appropriate block indices in order to generate and treat the R^{-1} factors.

3.2 Variable s -step GMRES

The fixed s -step GMRES method improves the condition number of the basis W compared with m -step GMRES, at the price of less parallelism. Nevertheless, if s is chosen too large, convergence may stagnate because of stability issues, as illustrated in the numerical experiments of section 5.

j	1	2	3	4	...
s_j	s	s	s	s	...
l_j	s	2s	3s	4s	...

Table 1: Fixed block size s

j	1	2	3	4	5	6 ...
s_j	1	2	3	5	8	13 ...
l_j	1	3	6	11	19	32 ...

Table 2: Fibonacci sequence of block size s_j .

To help balance these two concerns, we propose a new adaptive method, where the block size s is not fixed, but variable. Indeed, our numerical experiments of section 5 and theoretical results of section 4 suggest that taking a small block size in the first steps could improve the stability and convergence of the algorithm. The first steps would then go slower by performing fewer matrix-vector multiplications with less parallelism. The block size would increase quite fast up to a given size s , achieving the same parallelism as the fixed s -step method. Thanks to a faster convergence, the overall performance would be better.

In our variable approach, at each step j , we define a new block size s_j , and build a system W_{l_j} of size l_j , where $l_1 = s_1$ and $l_j = l_{j-1} + s_j$, $j \geq 2$. We assume that $m = l_J$ for some integer J .

In Algorithm SGMRES(m, s), we simply have $s_j = s$ and $l_j = sj$, as detailed in Table 1. Because of its rapid growth, we suggest to use the Fibonacci sequence to help define the block sizes s_j , as detailed in Table 2.

Because we do take a Non-Traditional approach, we may easily describe this variable approach. As such, all we need to do is to replace sj with l_j appropriately above in Algorithm 4 in order to properly describe a variable s -step method. We detail this more formally below.

As before, we define the first block $B_1 = W_{l_1}$ by

$$W_{l_1} = \{v_1, Av_1, \dots, A^{s_1-1}v_1\}, \quad (24)$$

and we perform a QR factorization

$$\{v_1, AW_{l_1}\} = V_{l_1+1}R_{l_1+1}. \quad (25)$$

As before, we give a recursive definition of W , the orthonormal system V and the triangular matrix R . At step j , with $2 \leq j \leq J$, we assume the existence of $W_{l_{j-1}}$, $V_{l_{j-1}+1}$ and we define the block B_j corresponding to a set of s_j multiplications performed between orthonormalizations.

We define the block B_j of size s_j by

$$B_j = \{u, Au, \dots, A^{s_j-1}u\}, \quad (26)$$

where $u = v_{l_{j-1}+1}$ the last vector of the orthonormal system $V_{l_{j-1}+1}$. We get a recursive relation for W_{l_j} :

$$W_{l_j} = \{W_{l_{j-1}}, B_j\}, \quad (27)$$

and perform a QR factorization

$$\{v_1, AW_{l_j}\} = V_{l_{j+1}}R_{l_{j+1}}. \quad (28)$$

Introducing the upper Hessenberg matrix H_{l_j} as $R_{l_{j+1}}$ minus the first column e_1 , we get

$$AW_{l_j} = V_{l_{j+1}}H_{l_j}. \quad (29)$$

In summary, we note that

$$\begin{aligned} W_{l_j} &= \{B_1, B_2, \dots, B_j\}, \\ &= \{v_1, Av_1, \dots, A^{s_1-1}v_1, v_{l_1+1}, \dots, A^{s_2-1}v_{l_1+1}, \dots, v_{l_{j-1}+1}, \dots, A^{s_j-1}v_{l_{j-1}+1}\}. \end{aligned} \quad (30)$$

We can reiterate the proof of Theorem 1 to get a very similar Theorem.

Theorem 2. *We assume that the dimension of K_{m+1} is equal to $m+1$. Let W_{l_j} defined by (27) and $V_{l_{j+1}}$ defined by (28). Then W_{l_j} is a basis of K_{l_j} and $V_{l_{j+1}}$ is a basis of $K_{l_{j+1}}$.*

Therefore, the subspace condition at each step j can be written for the Krylov subspace K_{l_j} as

$$x = x_0 + W_{l_j}y, \quad (31)$$

and the Galerkin condition is again a least squares problem

$$\min_y \|\beta e_1 - H_{l_j}y\|. \quad (32)$$

We summarize this variable s -step GMRES, which we call VGMRES(m,s), in Algorithm 5.

Algorithm 5 Non-Traditional Variable s -step VGMRES(m,s)

```

1: while not converged do
2:    $r_0 = b - Ax_0$ 
3:    $\beta = \|r_0\|$ 
4:    $v_1 = r_0/\beta$ 
5:    $W_0 = \emptyset$  and  $V_1 = \{v_1\}$ 
6:   for  $j = 1, J$  do
7:      $B_j = \{v_{l_{j-1}+1}, Av_{l_{j-1}+1}, \dots, A^{s_j-1}v_{l_{j-1}+1}\}$ 
8:      $W_{l_j} = \{W_{l_{j-1}}, B_j\}$ 
9:     orthonormalize  $AB_j$  against  $V_{l_{j-1}+1}$  to obtain  $\{v_{l_{j-1}+2}, \dots, v_{l_j+1}\}$  and the last columns of  $H_{l_j}$ 
10:    solve the least squares problem  $\min_y \|\beta e_1 - H_{l_j}y\| = \|\beta e_1 - H_{l_j}y_V\|$ 
11:    compute  $x_{l_j} = x_0 + W_{l_j}y_V$ 
12:    test convergence
13:   end for
14:   if not converged then
15:      $x_0 = x_m$ 
16:   end if
17: end while

```

Algorithm 5 is thus a generalization of Algorithm 4, with a fixed sequence $s_j = s$.

The actual conditions behind where to choose the block size s or the sequence s_j is a complex dynamical problem, which is dependent in a very complicated manner on the spectral decomposition of the matrix A and on the right-hand side b . As such, we only provide in the next section a few theorems to back up our claim that in general the first few steps should be small and then quickly grow.

4 Convergence and communication issues

4.1 Stability analysis

In both Algorithms 4 and 5, stability issues are directly related to the condition number $\kappa_2(H)$ to solve the least squares problem, and to the condition number $\kappa_2(W)$ to compute the approximate solution [16]. Since $AW = VH$, we get $\kappa_2(H) = \kappa_2(AW)$.

We start by obtaining a lower bound for the condition number of a matrix similar to W or AW . The general idea behind the following theorem should be compared to the power method [15], and is comparable to similar Krylov subspace results [2]. For some very large integer k , the two vectors $A^k u$ and $A^{k-1} u$ should be nearly in the direction of the dominant eigenvector of A . Because of this, isolating the influence of these two vectors shows the powerful effect they have on the condition.

Theorem 3. *Let λ_i be the eigenvalues of a nondegenerate nonsingular matrix A , such that $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n| > 0$.*

Let

$$D = \{D_1, A^{k-1}u, A^k u\} \quad (33)$$

for some integer k , some matrix D_1 with any given number of columns, and some vector u . Then there exists a constant c which depends on the vector u such that

$$\kappa_2(D) \geq c \left| \frac{\lambda_1}{\lambda_2} \right|^{k-1}$$

Proof. Note that

$$(\kappa_2(D))^{-1} = \min_{D+E \text{ is singular}} \frac{\|E\|}{\|D\|} \quad (34)$$

Let

$$E = \{0, 0, \lambda_1 A^{k-1}u - A^k u\}, \quad (35)$$

then $D + E$ is singular, so that

$$\kappa_2(D) \geq \frac{\|D\|}{\|E\|}.$$

First, we note that $\|D\| \geq \|A^k u\|$.

We consider the complex Schur factorization $A = QTQ^*$ with Q unitary and T upper triangular with the eigenvalues of A on the diagonal. We assume that the eigenvalues are ordered such that λ_1 is the last entry in the diagonal of T .

Let $\delta = Q^*u$ so that $u = Q\delta$ and $Au = QT\delta$. Then $A^k u = QT^k\delta$ and $\|A^k u\| = \|T^k\delta\|$. Since T is upper triangular with λ_1 in the bottom-right corner, we get $(T^k\delta)_n = \lambda_1^k \delta_n$, thus

$$\|D\| \geq |\lambda_1^k \delta_n|.$$

Second, $\|E\| = \|\lambda_1 A^{k-1}u - A^k u\| = \|\lambda_1 T^{k-1}\delta - T^k\delta\|$.

Let $X = (x_i)$ the eigenvectors of T and $\alpha = X^{-1}\delta$, such that

$$\lambda_1 T^{k-1}\delta - T^k\delta = \sum_{i=2}^n (\lambda_1 - \lambda_i)\lambda_i^{k-1}\alpha_i x_i.$$

Therefore

$$\|E\| \leq 2|\lambda_1||\lambda_2|^{k-1} \sum_{i=2}^n |\alpha_i| \|x_i\|.$$

Finally

$$\kappa_2(D) \geq c \left| \frac{\lambda_1}{\lambda_2} \right|^{k-1},$$

where

$$c = \frac{|\delta_n|}{2 \sum_{i=2}^n |\alpha_i| \|x_i\|}.$$

□

We may apply this theorem to obtain the following corollary to bound the condition numbers of W and H .

Corollary 3.1. *Let λ_i be the eigenvalues of a nondegenerate, nonsingular matrix A , ordered by decreasing modulus.*

Let W_{l_j} the Krylov basis defined by a variable s -step method, with a sequence s_j of block sizes.

Then there exist constants c and c_1 such that

$$\kappa_2(W_{l_j}) \geq c \left| \frac{\lambda_1}{\lambda_2} \right|^{s_j-2}, \kappa_2(H) \geq c_1 \left| \frac{\lambda_1}{\lambda_2} \right|^{s_j-1}.$$

Proof. Since $W_{l_j} = \{W_{l_{j-1}}, B_j\}$, and $B_j = \{u, Au, \dots, A^{s_j-1}u\}$, we can apply Theorem 3, with $k = s_j - 1$.

Similarly, we can apply Theorem 3 to AW_{l_j} with $k = s_j$ and use the equality $\kappa_2(H) = \kappa_2(AW)$.

□

These results show that with a fixed s -step method, the lower bound on $\kappa_2(H)$ is larger than $\left| \frac{\lambda_1}{\lambda_2} \right|^{s-1}$ right after the first step, and in the successive steps.

For a variable s -step method with a decreasing sequence s_j starting at $s_1 = s$, the conclusion is the same. Indeed, the condition number of W_{l_j} is larger than the condition number of $W_{l_{j-1}}$.

On the other hand, for a variable s -step method with an increasing sequence s_j , the lower bounds are much smaller. Thus we assume that $\kappa_2(H)$ is small in the first steps and that it increases gradually.

Therefore we expect better approximations in the first steps and faster convergence.

We propose to use a Fibonacci sequence as defined by Table 2, but capped at some maximal value s . We expect to get an increasing condition number, up to the order $\left| \frac{\lambda_1}{\lambda_2} \right|^{s-1}$ and to improve convergence, compared to a fixed s -step method.

This choice of a Fibonacci sequence is motivated by parallel issues and communication costs. Since the sequence increases rapidly, only a few steps are necessary to achieve the chosen value s . Then the algorithm is similar to the fixed s -step method, with the same parallel performances.

4.2 Communication Analysis

One last question remains. While we have studied the stability issues of Algorithm 5, we have yet to show that the parallel costs are not prohibitive.

The two main operations in a fixed or variable s -step method are the sequence of matrix-vector multiplications to compute the block B_j , and the orthonormalization of AB_j to compute the new basis vectors of V and the new columns of H .

The first operation can be done efficiently in parallel by using parallelism in each matrix-vector multiplication [24] or a power-matrix kernel [22], whereas the second operation can be parallelized by using various parallel algorithms [32] such as RODDEC [24] or CA-QR [11].

These communication-avoiding algorithms ensure a low communication overhead. We assume here that each step j requires a fixed number of messages, independent of s_j , so that the total number of messages is proportional to the total number of steps, which is m/s in a fixed s -step method and some J in a variable s -step method.

j	1	2	3	4	J_1	$J_1 + 1$...	$J_1 + J_2 = J$
s_j	1	2	3	5	$\leq s$	s	...	s
l_j	1	3	6	11	l_{J_1}	$l_{J_1} + s$...	m

Table 3: Capped Fibonacci s -step sequence.

We denote by $\text{FibGMRES}(m,s)$ the special case of Algorithm 5 where the sequence of block sizes s_j is given by Fibonacci numbers until the block size is capped at s , as shown in Table 3.

We compare it to Algorithm $\text{SGMRES}(m,s)$, where the block size s is fixed. To this end, we compare m/s with J :

Theorem 4. *Let J be the total number of steps in $\text{FibGMRES}(m,s)$ and m/s the total number of steps in $\text{SGMRES}(m,s)$. Then, if s is small compared to m , J is of the same order as m/s .*

Proof. Let J_1 be the step where the Fibonacci sequence is capped, and J_2 the number of remaining steps after this cap, so that $J = J_1 + J_2$.

Also, $m = l_{J_1} + sJ_2$ so that $J = m/s + J_1 - l_{J_1}/s$.

Assuming that we start here with $s_1 = 1, s_2 = 2$ as in Table 3, the block size s_j is the $(j + 1)$ th Fibonacci number. Using the properties of the Fibonacci sequence [28], we get $l_{J_1} = s_{J_1+2} - 2$, thus

$$J = m/s + J_1 - (s_{J_1+2} - 2)/s.$$

We now want to bound J_1 and to show that it is small compared to m/s .

Let $\phi = \frac{1+\sqrt{5}}{2}$, then $s_j = \frac{\phi^{j+1} - \phi^{-(j+1)}}{\sqrt{5}}$ for $j \leq J_1$.

Using that $s_{J_1} \leq s$, we get

$$\begin{aligned} \phi^{J_1+1} &= \sqrt{5}(s_{J_1} + \phi^{-(J_1+1)}) \\ &\leq \sqrt{5}(s + 1) \\ J_1 &\leq \log_{\phi}(\sqrt{5}(s + 1)) - 1 \\ J_1 &= O(\log_{\phi}(s)) \end{aligned} \tag{36}$$

Thus $J = O(m/s + \log_{\phi}(s)) = O(m/s)$. □

Therefore, the communication cost is at least of the same order regardless of using fixed s -step or FibGMRES due to the rapid growth of the s_j block size in FibGMRES .

It should also be noted here the important payoff between stability and communication issues. If s is kept too small, then this increases m/s , which is proportional to the number of messages. Therefore, decreasing s also decreases the level of parallelism in the algorithm. However, in the previous section we

showed that s being too large introduces stability problems. This implies that there is a balance that must be struck between stability and communication.

We now show some numerical experiments confirming this interplay, as well as the convergence properties of FibGMRES.

5 Numerical Results

In this section we concentrate on demonstrating three central results that follow chronologically with what we presented above.

The first will regard the performance of traditional and non-traditional Algorithms 2 and 3 respectively. Namely, in our tests, the differences in performance of these two methods is marginal, so that the simplicity of implementation of Algorithm 3 may be preferable in some situations compared to Algorithm 2.

In the second part we will analyze the impact of the block size s on convergence rate in the fixed s -step method. We show that the condition number of AW increases with s and affects convergence as predicted in Corollary 3.1.

After this we will focus on the heart of this paper, which is the performance of variable s -step GMRES methods, in particular the Fibonacci s -step FibGMRES(m, s) method, where the variable block size s_j follows a Fibonacci pattern. We compare the variable FibGMRES method to the fixed s -step SGMRES method, by looking at the convergence rate and at the condition number $\kappa_2(AW)$. Our tests demonstrate the behavior we analyzed in Corollary 3.1.

We developed a code in Octave and performed all experiments on a computer with a x86 architecture. In all our tests, the initial guess x_0 as well as the right-hand side b are vectors whose components are uniform random numbers between 0 and 1. The relative residual at iteration k is defined by $\|r_k\| / \|r_0\|$.

5.1 Experiments with m -step GMRES

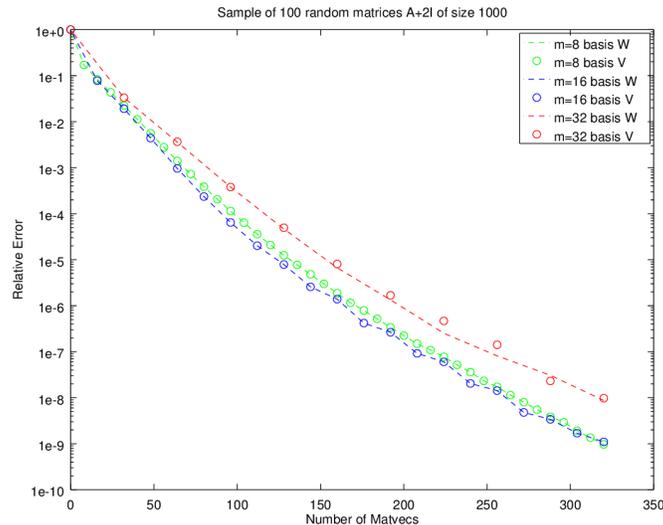


Figure 2: Results with samples of random matrices and algorithm m -step GMRES for three values of m . Mean error, using either V or W as a basis in the subspace condition.

The first question we have to answer is whether the use of V or W as a Krylov basis impacts the stability and convergence of m -step methods. To help answer this question, we rely on a series of tests, done with samples of sparse matrices of type $A + 2I$, where the matrix A is random and I is the identity matrix. These matrices are of size $n = 1000$ with 10000 nonzeros. They are scaled and preconditioned by Jacobi. We compare the two versions of m -step GMRES, with a block size equal to the restarting parameter, where the basis is either V or W . In Figure 2, we plot the mean absolute error $\|x_m - (A + 2I)^{-1}b\|$ after each restarting cycle, for three values of m .

The errors of the two versions, using either V or W , are very similar. This is in good agreement with our earlier remark that both approaches seem to have a similar numerical behavior. Instabilities in the algorithm occur either in the inversion of R_m or in the matrix multiplication by W_m .

Therefore, we use the basis W and Algorithm 3 for the sake of simplicity.

It should be noted that with the addition of better preconditioners or with a well-conditioned matrix, what would change is essentially the rate of convergence. A similar situation would occur with the addition of a Newton basis, which would in theory improve the condition numbers of W_m and R_m .

With classical restarted GMRES(m), convergence is quite often faster by increasing the restarting parameter m [30]. As we can see in Figure 2, $m = s = 16$ does marginally better than $m = s = 8$, but by the time $m = s = 32$, the stability issues inherent in m -step GMRES begin to surface.

Thus, on one hand, a large value of m is sometimes necessary to ensure convergence of restarted GMRES and on the other hand a small value of m is required to ensure a well-conditioned Krylov basis. In [24] for example, deflation is used to improve stability, whereas in [17] for example, the block size s is chosen smaller than m . We now investigate this solution.

5.2 Experiments with fixed s -step SGMRES(m,s)

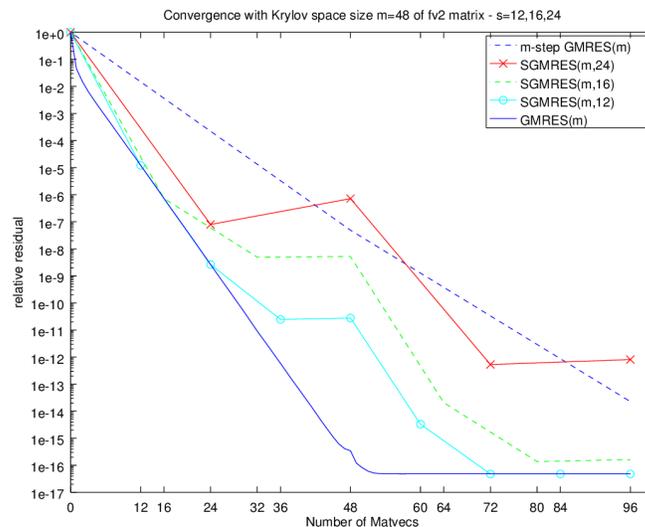


Figure 3: Results with matrix fv2 and fixed s -step SGMRES(m,s) with $m = 48$. Impact of block size s on convergence.

From now on, we use the basis W in all our experiments. Here, we analyze the convergence and the numerical stability of the fixed s -step method, with s smaller than m , denoted by SGMRES(m,s). We use the matrix fv2 from the University of Florida matrix collection, which is of size $n = 9801$ with

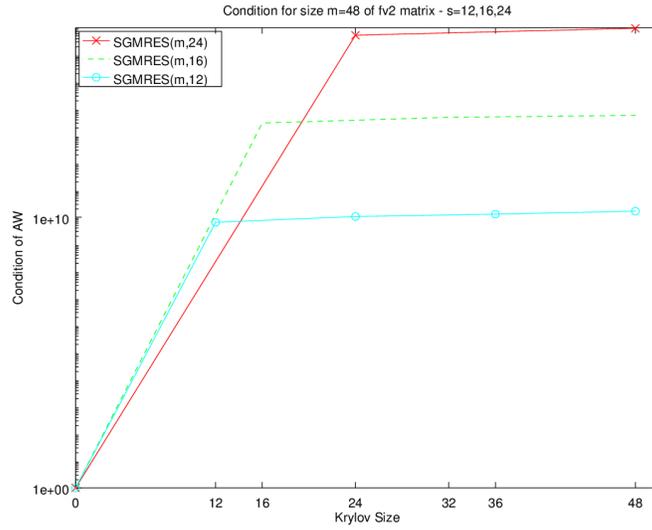


Figure 4: Results with matrix fv2 and fixed s -step SGMRES(m,s) with $m = 48$. Impact of block size s on condition number of AW .

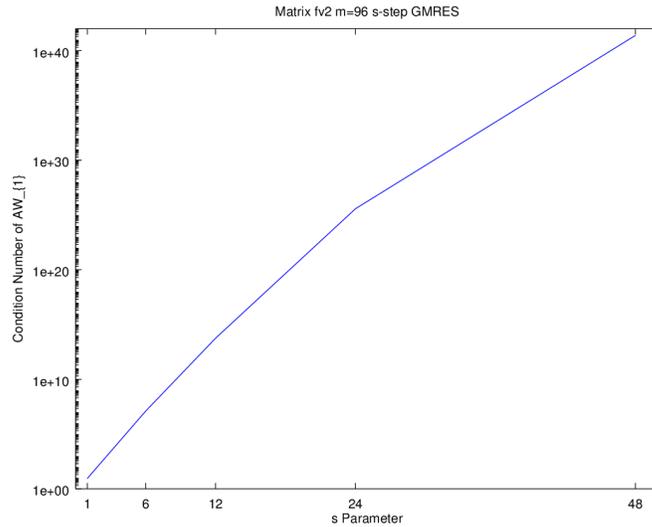


Figure 5: Results with matrix fv2 and fixed s -step SGMRES(m,s) with $m = 96$. Impact of block-size s on condition number of AW_1 .

$nz = 87025$ nonzero elements [9]. This matrix is symmetric and well-conditioned so that restarted GMRES(m) converges quickly, without preconditioning and with a relatively small value of m . We first analyze the impact of the block size s on stability and convergence, for a given restarting parameter $m = 48$. Figures 3 and 4 represent respectively the relative residual and the condition number of AW .

The primary feature to notice in Figure 3 is the bounding behavior by standard GMRES(m) and m -step GMRES, which correspond respectively to the bounding values $s = 1$ and $s = m$ of the block size. As expected, generally GMRES(m) is the best case, whereas m -step GMRES is the worst case.

A 'hockey-stick' pattern occurs for all s values during each restarting cycle, where just before restarting the residual norm has a mild plateau in the case of $s = 12$, but can even begin to increase slightly for $s = 16$ and $s = 24$. In this last case, the residual becomes even larger than for $s = m$.

Meanwhile, this behavior is well corroborated by Figure 4. The condition number of the first block is quite large and then plateaus. Whereas it does so only moderately for $s = 12$, it does so increasingly rapidly for higher s values.

That this happens has been described by our stability analysis. That larger value for s induces higher condition numbers by Corollary 3.1, which directly explains the condition number graph in Figure 4. Moreover, numerical instability occurs right after the first block, and convergence stagnates after a few steps.

In Figure 5, we have plotted the condition number of the first block AW_1 for $s = 1, 4, 8, 16, 32$, with $m = 96$, into a single line to show the growth of the condition number. We see that it increases with s , as expected by Corollary 3.1.

Therefore, due to the heightened instabilities, increasing s past a certain point is a waste of computational time. In a sense, 'The damage has already been done.' Due to this effect, we conclude that a large fixed block size can destroy convergence right after the first block.

5.3 Variable s -step FibGMRES(m, s) with symmetric matrices

Due to this principle of damage from large s being done quickly when introduced early on in the computation, we advocate a variable block size, where the block size increases gradually. This allows us to properly balance the considerations of the parallel computations that s -step affords with the stability enhancements that a small block size affords. A variable block size allows a much greater degree of control over these two competing concerns than with a fixed block size. Also this approach can be combined with adaptive techniques, where the block size could be chosen according to some indicator. We run experiments again with the matrix fv2, $m = 48$, and $s = 16$. We aim to compare the extreme cases $s = 1$ and $s = m = 48$ to the fixed case $s = 16$ and to the variable case FibGMRES(m, s) defined by Table 4. Here, we increase gradually the block size following a Fibonacci sequence capped at $s = 16$, and we recall that each new block B_j is of size s_j and that the Krylov size l_j is given by $l_j = l_{j-1} + s_j$. For the sake of comparison, we also gradually decrease the block size from $s = 16$ to $s = 1$ using the Fibonacci sequence in reverse order, as defined in Table 5.

In Figure 6, convergence of FibGMRES(m, s) is much better than convergence of a fixed s -step method or a variable s -step method with a decreasing block size. Indeed, we see that the method suggested by Table 5 behaves like SGMRES(m, s), with a hockey-stick pattern. On the other hand, FibGMRES curve closely follows the optimal standard GMRES curve.

As can be seen in Figure 7, the condition number of AW increases gradually for FibGMRES, whereas it is already large for the first block of the other methods. This emphasizes that the increased performance in Fibonacci s -step GMRES is really due to the order of the increasing step size, and the principle we introduced earlier that the damage can often already be done by the first step of the algorithm. Stability issues must be nipped in the bud early on before they spread.

j	1	2	3	4	5	6	7
s_j	1	2	3	5	8	13	16
l_j	1	3	6	11	19	32	48

Table 4: Variable increasing block size for $m = 48$ and $s = 16$.

j	1	2	3	4	5	6	7
s_j	16	13	8	5	3	2	1
l_j	16	29	37	42	45	47	48

Table 5: Variable decreasing block size for $m = 48$ and $s = 16$.

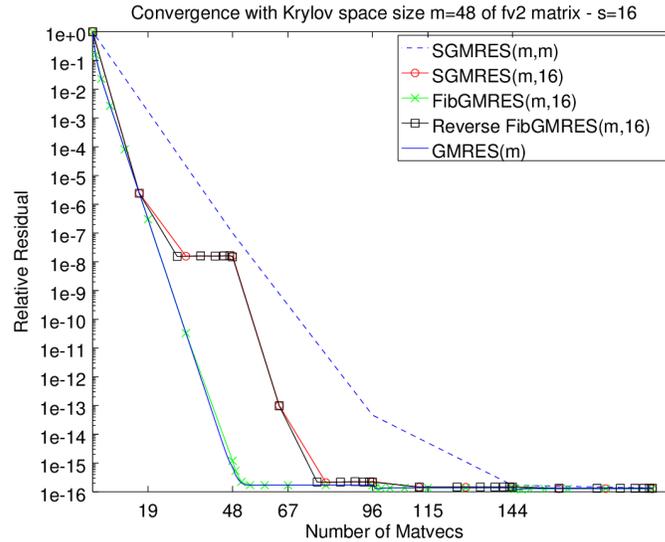


Figure 6: Convergence results with matrix $fv2$, with $m = 48$ and $s = 16$. Comparison of fixed and variable s -step GMRES.

We now run experiments with a prototypical test problem, where the matrix arises from a Poisson equation in a unit square, discretized by a finite difference scheme, using a 5-point stencil. A regular grid of size 150×150 results in a sparse matrix of order $n = 22500$. This Poisson matrix is symmetric but not as well-conditioned as the $fv2$ matrix, so we can expect a slower convergence. We choose a larger restarting parameter $m = 96$ and two maximal block sizes, namely $s = 16$ and $s = 32$. The sequences of block sizes are defined in Tables 6 and 7. As an aside, we should note that when $s = 32$, the truncated Fibonacci numbers do not neatly add up to $m = 96$. In order to make a fair comparison with fixed s -step GMRES, we have chosen to modify the Fibonacci sequence at step $j = 8$ in order to get $l_j = 96$ at step $j = 9$. In practice, this is not a concern because the user would choose the maximal block size s and the maximal number of steps J , in order to get l_j close to a chosen m value.

Convergence results are plotted in Figure 8. Whereas for $s = 16$, the fixed s -step GMRES and the variable Fibonacci GMRES converge as quickly as the classical restarted GMRES, the behaviors are very different for $s = 32$. Indeed, fixed SGMRES(96,32) shows an erratic convergence rate where the residual norms increase and are no longer bounded by those of m -step GMRES (where $s = m$). On the other hand, FibGMRES is still very efficient and converges almost as fast as restarted GMRES.

Again, these results can be explained by the condition number of AW and by our stability analysis. As can be seen in Figure 9, the condition number of the first block is quite high with the fixed s -step method, but increases progressively with FibGMRES, in both cases $s = 16$ and $s = 32$.

We now look at a larger Poisson matrix of size $n = 100489$ so that we may extend the Krylov size to $m = 400$, while keeping the block size at $s = 16$. These tests are important, because the number of messages is roughly equal to the quantity $\frac{m}{s}$ when it becomes large, as stated in Theorem 4. Thus

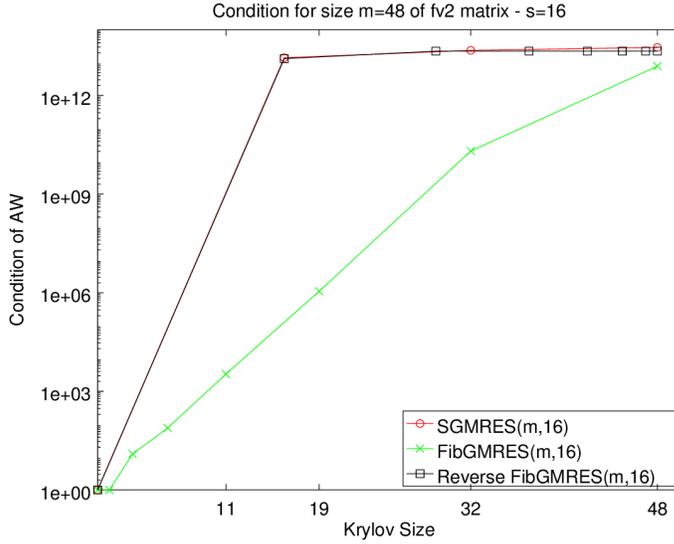


Figure 7: Condition results with matrix fv2, with $m = 48$ and $s = 16$. Comparison of fixed and variable s -step GMRES.

j	1	2	3	4	5	6	7	8	9	10
s_j	1	2	3	5	8	13	16	16	16	16
l_j	1	3	6	11	19	32	48	64	80	96

Table 6: Variable increasing block size for $m = 96$ and $s = 16$.

j	1	2	3	4	5	6	7	8	9
s_j	1	2	3	5	8	13	14	18	32
l_j	1	3	6	11	19	32	46	64	96

Table 7: Variable increasing block size for $m = 96$ and $s = 32$.

parallel performances are similar with fixed and variable block size in that case.

In Figure 10, we see that FibGMRES converges as fast as restarted GMRES, whereas SGMRES slows down just before restarting. In fact, it is interesting to compare Figure 10 with Figure 8, and to notice that it may take some time before a difference occurs between FibGMRES and SGMRES.

5.4 Experiments with variable FibGMRES(m,s) and nonsymmetric matrices

The previous experiments were done with symmetric matrices to illustrate various convergence behaviors. Nevertheless, GMRES is designed to solve nonsymmetric systems. Thus we now run experiments with nonsymmetric matrices.

The first matrix, called cage10, is from the Univeristy of Florida matrix collection [9]. It is of size 11397 with 150645 nonzero elements. Restarted GMRES converges quickly for this matrix, without preconditioning.

We repeat the same tests as for the matrix fv2 and plot the results in Figure 11 and 12. The conclusion is the same as for fv2: FibGMRES converges faster than SGMRES or variable s -step with a reverse

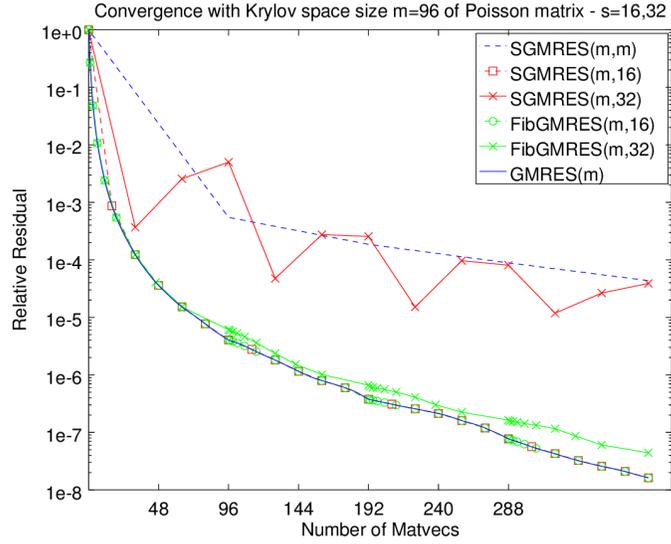


Figure 8: Convergence Results with the Poisson matrix, with $m = 96$ and $s = 16, 32$.

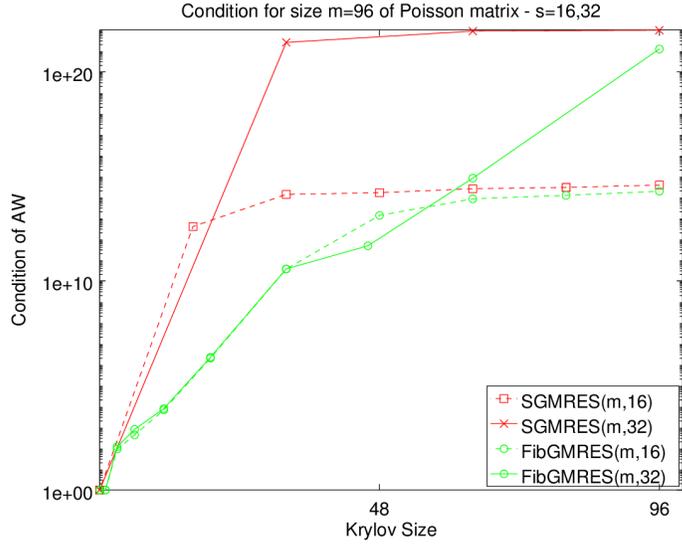


Figure 9: Results with the condition of the Poisson matrix, with $m = 96$ and $s = 16, 32$.

Fibonacci order. Also, condition numbers corroborate the theoretical results. The second matrix, PR02R, which is also pulled from the University of Florida matrix collection, represents a turbulence problem from the FLUOREM collection. It is of size 161070 with 8185136 nonzero elements. For this matrix and similar matrices from the FLUOREM collection, iterative methods converge very slowly or do not converge [34],[25]. Here, in order to get an easier problem with no preconditioning, we have added the matrix $1000 \times I$, where I is the Identity matrix. We repeat the same experiments as for the Poisson matrix and plot the results in Figures 13 and 14.

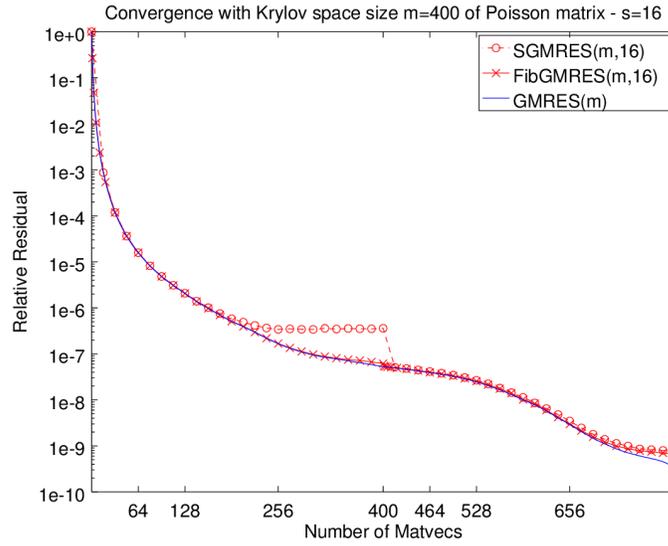


Figure 10: Convergence Results with a Poisson matrix of size $n = 100489$, $m = 400$ and $s = 16$.

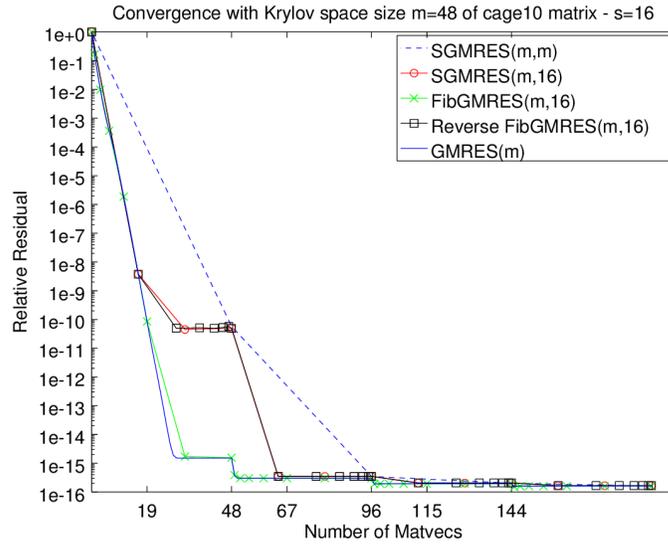


Figure 11: Convergence results with matrix `cage10`, with $m = 48$ and $s = 16$. Comparison of fixed and variable s -step GMRES.

This test case exhibits very similar features to previous ones. Indeed, variable s -step $\text{FibGMRES}(m,s)$ is more robust than fixed s -step $\text{SGMRES}(m,s)$ when increasing the block size: with $s = 16$ both methods are as fast as restarted GMRES, but with $s = 32$, $\text{FibGMRES}(s,m)$ converges almost as fast as restarted GMRES, whereas $\text{SGMRES}(m,s)$ is almost as slow as m -step GMRES. Also, the condition number of AW increases gradually with $\text{FibGMRES}(m,s)$, whereas it is quite large from the first block with $\text{SGMRES}(m,s)$: as already mentioned, the damage is done.

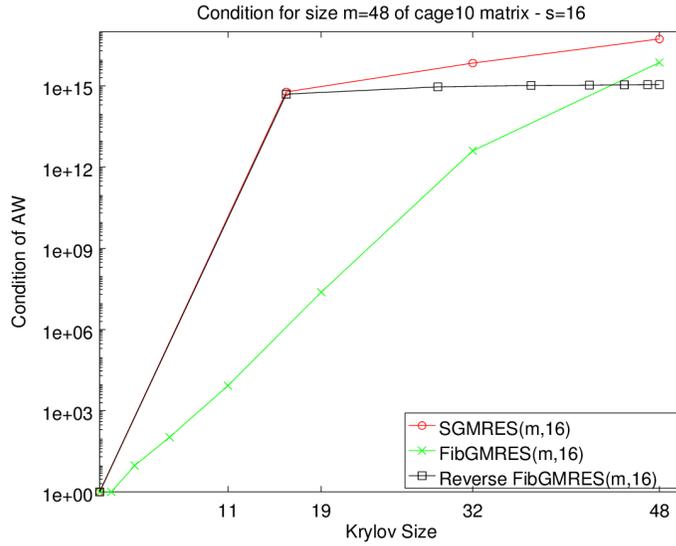


Figure 12: Condition results with matrix cage10, with $m = 48$ and $s = 16$. Comparison of fixed and variable s -step GMRES.

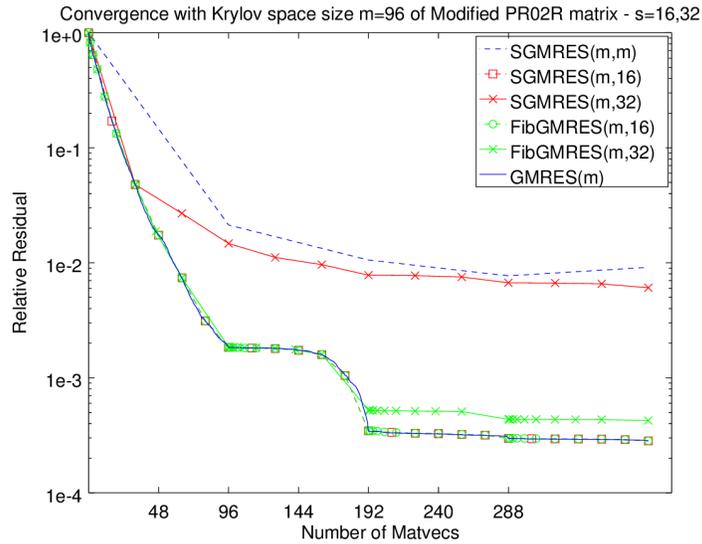


Figure 13: Convergence Results with the modified PR02R matrix, with $m = 96$ and $s = 16, 32$.

Then, we increase the Krylov size to $m = 192$ with the same block size $s = 16$ or $s = 32$. Here, the ratio m/s is large enough to ensure similar parallel behaviors in the fixed and variable s -step methods. Again, for $s = 16$, convergence is almost as fast as with restarted GMRES. But, for $s = 32$, convergence is damaged with either a fixed or a variable block size, with a worse behavior for a fixed block size.

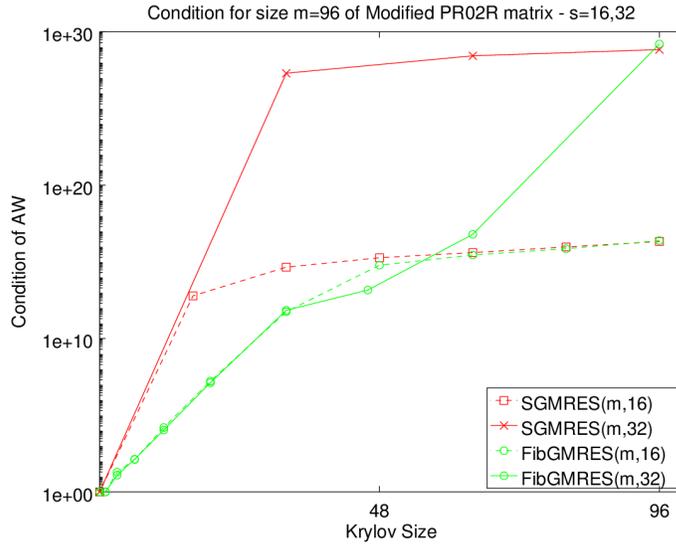


Figure 14: Results with the condition of the modified PR02R matrix, with $m = 96$ and $s = 16, 32$.

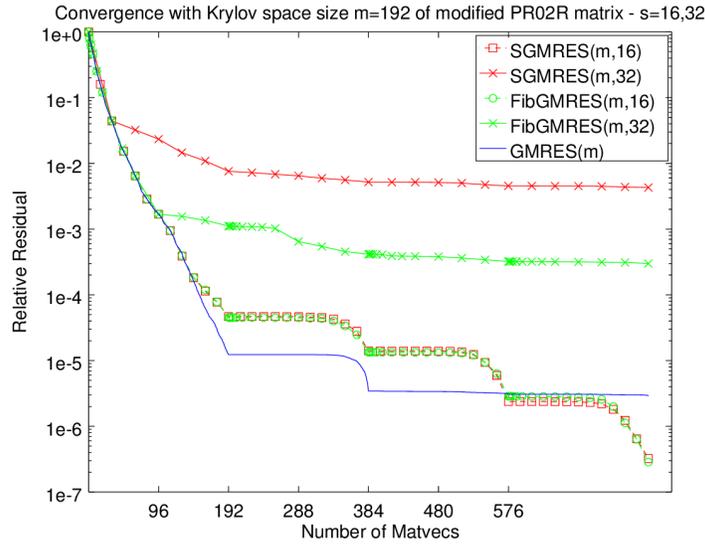


Figure 15: Convergence Results with the modified PR02R matrix, with $m = 192$ and $s = 16, 32$.

6 Conclusion

Communication-avoiding Krylov methods are very efficient to increase performances on parallel computers. Among them, s -step GMRES builds an orthonormal basis of a Krylov subspace by consecutive blocks of size s . In this paper, we described how to use two bases in order to simplify the algorithm. Then, we proposed to vary the s and defined an original variable s -step GMRES algorithm. Our stability analysis suggests to use an increasing sequence s_j of the block sizes.

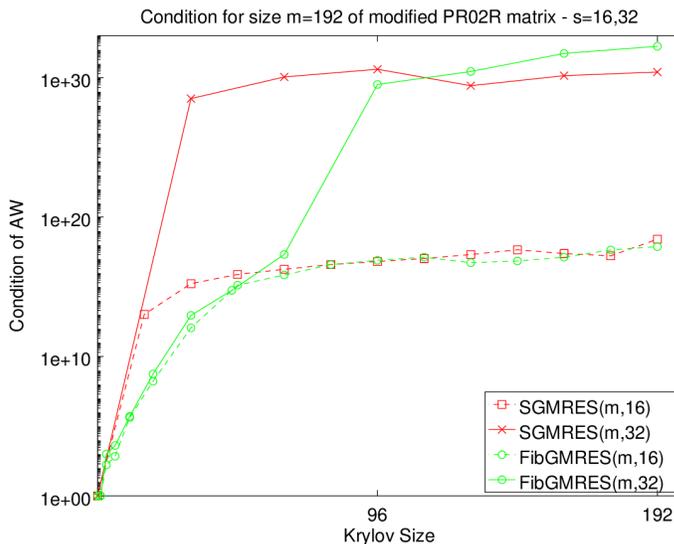


Figure 16: Results with the condition of the modified PR02R matrix, with $m = 192$ and $s = 16, 32$.

What we have shown, in essence, is a beneficial tradeoff. An additional cost occurs in the first few steps of the start-up phase of variable step GMRES, because communication overheads are induced by the fan-in fan-out problems associated to small blocks. However, the increased stability resulting from the adaptive nature of variable s -step GMRES in turn induces a significant reduction in the number of required iterations. Moreover, we propose to use a Fibonacci sequence capped at some value s in order to avoid communications by limiting the number of steps in the start-up phase. Our numerical experiments show that our new variable s -step GMRES, called $\text{FibGMRES}(m,s)$, converges often as quickly as restarted $\text{GMRES}(m)$ and faster than fixed s -step, called $\text{SGMRES}(m,s)$. Further, the performance is that of s -step methods once the sequence is capped.

As noted in the paper, we could save on memory by storing only the orthonormal basis. We could also improve the stability of the method by using a Newton basis for computing the block at each step [6]. In the future, we plan to implement our method on parallel computers, using an efficient matrix-vector product and a communication-avoiding orthonormalization algorithm. Parallel domain decomposition methods combined with deflation will be used for preconditioning the matrices. This parallel version will be tested with very large matrices, arising from various computational science problems.

References

- [1] Z. BAI, D. HU, AND L. REICHEL, *A Newton Basis GMRES Implementation*, IMA Journal of Numerical Analysis, (1994), pp. 563–581. 14(4).
- [2] BELLALIJ, M., MEURANT, G., AND SADOK, H., *The Distance of an Eigenvector to a Krylov Subspace and the Convergence of the Arnoldi Method for Eigenvalue Problems*, Linear Algebra and its Applications, (2016), pp. 387–405. 504.
- [3] R. H. BISSELING, T. M. DOUP, AND L. D. J. C. LOYENS, *A Parallel Interior Point Algorithm for Linear Programming on a Network of Transputers*, Annals of Operations Research, (1993), pp. 49–86. 43(2).
- [4] D. CALVETTI, J. PETERSEN, AND L. REICHEL, *A Parallel Implementation of the GMRES Method*, in Numerical Linear Algebra, L. Reichel, A. Ruttan, and R. S. Varga, eds., de Gruyter, Berlin, 1993, pp. 31–45.
- [5] E. CARSON, *Communication-Avoiding Krylov Subspace Methods in Theory and Practice*, PhD thesis, University of California at Berkeley, 2015.
- [6] E. CARSON AND J. DEMMEL, *An Efficient Deflation Technique for the Communication Avoiding Conjugate Gradient Method*, Electronic Transaction on Numerical Analysis, (2014), pp. 125–141. 43.
- [7] A. T. CHRONOPOULOS, *S-Step Iterative Methods for (non) Symmetric (in) Definite Linear Systems*, SIAM journal on numerical analysis, (1991), pp. 1776–1789. 28(6).
- [8] R. D. DA CUNHA AND T. HOPKINS, *A Parallel Implementation of the Restarted GMRES Iterative Algorithm for Nonsymmetric Systems of Linear Equations*, Advances in Computational Mathematics, (1994), pp. 261–277. 2(3).
- [9] T. A. DAVIS AND Y. HU, *The University of Florida Sparse Matrix Collection*, ACM Transactions on Mathematical Software (TOMS), (2011). 38(1).
- [10] E. DE STURLER AND H. A. VAN DER VORST, *Reducing the Effect of Global Communication in GMRES(m) and CG on Parallel Distributed Memory Computers*, Applied Numerical Mathematics, (1995), pp. 441–459. 18(4).
- [11] J. DEMMEL, L. GRIGORI, M. HOEMMEN, AND J. LANGOU, *Communication-optimal Parallel and Sequential QR and LU Factorizations*, SIAM Journal on Scientific Computing, (2012), pp. A206–A239. 34(1).
- [12] G. R. DI BROZOLO AND Y. ROBERT, *Parallel Conjugate Gradient-like Algorithms for Solving Sparse Nonsymmetric Linear Systems on a Vector Multiprocessor*, Parallel Computing, (1989), pp. 223–239. 11(2).
- [13] J. ERHEL, *A Parallel GMRES Version for General Sparse Matrices*, Electronic Transaction on Numerical Analysis, (1995), pp. 160–176. 3 (Dec).
- [14] L. GIRAUD, J. LANGOU, M. ROZLOZNIK, AND J. VAN DEN ESHOF, *Rounding Error Analysis of the Classical Gram-Schmidt Orthogonalization Process*, Numerische Mathematik, (2005), pp. 87–100. 101(1).
- [15] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations* (4th edition), John Hopkins University Press, 2013. (Vol. 4). JHUP.
- [16] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms* (second edition), SIAM, 2002.

- [17] M. F. HOEMMEN, Communication-avoiding Krylov subspace methods, *PhD thesis, EECS Department, University of California, Berkeley, Apr 2010.*
- [18] A. S. HOUSEHOLDER, The Theory of Matrices in Numerical Analysis, *Dover Publications.*, 1964.
- [19] W. JALBY AND B. PHILIPPE, Stability Analysis and Improvement of the Block Gram-Schmidt Algorithm, *SIAM journal on scientific and statistical computing*, (1991), pp. 1058–1073. 12(5).
- [20] C. T. KELLEY, Iterative Methods for Linear and Nonlinear Equations, *SIAM*, 1995.
- [21] S. KIM AND A. T. CHRONOPOULOS, An Efficient Nonsymmetric Lanczos Method on Parallel Vector Computers, *Journal of Computational and Applied Mathematics*, (1992), pp. 357–374. 42(3).
- [22] M. MOHIYUDDIN, M. HOEMMEN, J. DEMMEL, AND K. YELICK, Minimizing Communication in sparse matrix solvers, *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, (2009), p. 36.
- [23] N. NASSIF, J. ERHEL, AND B. PHILIPPE, Introduction to Computational Linear Algebra, *CRC Press*, 2015.
- [24] D. NUENTSA WAKAM AND J. ERHEL, Parallelism and Robustness in GMRES with the Newton Basis and the Deflated Restarting, *Electronic Transactions on Numerical Analysis*, 40 (2013), pp. 381–406.
- [25] D. NUENTSA WAKAM AND F. PACULL, Memory efficient hybrid algebraic solvers for linear systems arising from compressible flows, *Computers and Fluids*, 80 (2013), pp. 158–167.
- [26] B. PHILIPPE AND L. REICHEL, On the generation of Krylov subspace bases, *Applied Numerical Mathematics (APNUM)*, 62 (2012), pp. 1171–1186.
- [27] B. PHILLIPE AND R. SIDJE, Parallel Algorithms for the Arnoldi Procedure., *Iterative Methods in Linear Algebra, II, IMACS Ser. Comput. Appl. Math*, (1994), pp. 156–165. 3.
- [28] K. ROSEN, Elementary Number Theory, *Pearson Education.*, 2011.
- [29] Y. SAAD, Iterative Methods for Sparse Linear Systems, *SIAM.*, 2003.
- [30] Y. SAAD AND M. H. SCHULTZ, GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems, *SIAM Journal on scientific and statistical computing*, (1986), pp. 856–869. 7(3).
- [31] J. N. SHADID AND R. S. TUMINARO, Sparse Iterative Algorithm Software for Large Scale MIMD Machines: An initial discussion and implementation, *Concurrency: practice and experience*, (1992), pp. 481–497. 4(6).
- [32] R. B. SIDJE, Alternatives for Parallel Krylov Basis Computation, *Numerical Linear Algebra with Applications*, 4 (1997), pp. 305–331.
- [33] STRAKOS, Z., PAIGE, C.C., ROZIONIK, M., MEURANT, G., TICHY, P., AND LIESEN, J., Stability and Stopping Criteria in Iterative Methods for Linear Algebraic Systems, in *PANM 13 conference in honor of Professor Ivo Babuska, Prague, 2006.*
- [34] D. N. WAKAM, J. ERHEL, E. CANOT, AND G. ATENEKENG-KAHOU, A comparative study of some distributed linear solvers on systems arising from fluid dynamics simulations, in *Parallel Computing: from Multicores and GPU's to Petascale (proceedings of PARCO'09)*, B. Chapman, F. Desprez, G. Joubert, A. Lichnewsky, F. Peters, and T. Priol, eds., vol. 19 of *Advances in parallel computing*, IOS Press, 2010, pp. 51–58.

[35] H. F. WALKER, Implementation of the GMRES Method Using Householder Transformations, *SIAM Journal on Scientific and Statistical Computing*, (1988), pp. 152–163. 9(1).