N^o D'ORDRE 693

MANUSCRIT de THÈSE

présentée devant

L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE TOULOUSE

en vue de l'obtention du doctorat
Spécialité: Mathématiques Appliquées
par

M Julien Langou

Résolution de systèmes linéaires de grande taille avec plusieurs seconds membres.

Solving large linear systems with multiple right–hand sides.

---

soutenue publiquement à Saint Girons (09, France), le 10 juin 2003, devant Messieurs :

Guillaume Alléon	Directeur de groupe, EADS–CCR	France	*invité*
Åke Björck	Professeur, Linköping University	Suède	*rapporteur*
Iain S. Duff	Directeur de projet CERFACS et RAL	Royaume–Uni	*membre du jury*
Luc Giraud	Chercheur sénior, CERFACS	France	*directeur de thèse*
Gene H. Golub	Professeur, Stanford University	USA	*membre du jury*
Gérard Meurant	Directeur de Recherche au CEA	France	*président*
Chris C. Paige	Professeur, McGill University	Canada	*rapporteur*
Yousef Saad	Professeur, University of Minnesota	USA	*invité*

LANGOU Julien

Résolution de systèmes linéaires de grande taille avec plusieurs seconds membres.

nombre de pages : 235, thèse doctorat de mathématiques appliquées, soutenance à Saint Girons (09) le 10 juin 2003, numéro d'ordre: 693

RÉSUMÉ:
Le point de départ de cette thèse est un problème posé par le groupe électromagnétisme de EADS-CCR : comment résoudre plusieurs systèmes linéaires avec la même matrice mais différents seconds membres ? Pour l'application voulue, les matrices sont complexes, denses et de grande taille (de l'ordre de quelques millions). Comme de telles matrices ne peuvent être ni calculées, ni stockées dans un processus industriel, l'utilisation d'un produit matrice-vecteur approché est la seule alternative. En l'occurrence, le produit matrice-vecteur est effectué en utilisant la méthode multipôle rapide. Dans ce contexte, le but de cette thèse est d'adapter les méthodes itératives de type Krylov de telles sorte qu'elles traitent efficacement les nombreux seconds membres. Nous nous concentrons particulièrement sur l'algorithme GMRES et ses variantes. Les schémas d'orthogonalisation que nous avons implanté dans GMRES sont des variantes de l'algorithme de Gram-Schmidt. Dans une première partie, nous nous intéressons à l'influence des erreurs d'arrondi dans les algorithmes de Gram-Schmidt. Dans une deuxième partie, nous avons étudié des variantes de la méthode GMRES, notamment GMRES-DR, seed GMRES et block GMRES. La troisième partie est dédiée à l'amélioration de ces techniques standards dans le cadre des problèmes électromagnétiques.

MOTS CLÉS :
méthode d'orthogonalisation, méthode de Krylov avec plusieurs seconds membres, calcul de section équivallente radar monostatique

soutenue publiquement à Saint Girons (09, France), le 10 juin 2003, devant Messieurs :

Membres du jury:

Guillaume Alléon	Directeur de groupe, EADS–CCR	France	*invité*
Åke Björck	Professeur, Linköping University	Suède	*Rapporteur*
Iain S. Duff	Directeur de projet CERFACS et RAL	Royaume–Uni	*membre du jury*
Luc Giraud	Chercheur sénior, CERFACS	France	*Directeur de thèse*
Gene H. Golub	Professeur, Stanford University	USA	*membre du jury*
Gérard Meurant	Directeur de Recherche au CEA	France	*Président*
Chris C. Paige	Professor, McGill University	Canada	*Rapporteur*
Yousef Saad	Professeur, University of Minnesota	USA	*invité*

Dépôt à la bibliothèque universitaire en 4 exemplaires.

LISTE DES PUBLICATIONS (parues ou soumises)

1. Luc Giraud and Julien Langou (2002). When modified Gram–Schmidt generates a well–conditioned set of vectors. *IMA Journal of Numerical Analysis*, 22(4):521–528.

2. Luc Giraud, Julien Langou and Miroslav Rozložník (2002). On the loss of orthogonality in the Gram–Schmidt orthogonalization process. À paraître cette année dans *Computer and Mathematics with Applications (2003)*.

3. Luc Giraud and Julien Langou (2002). Robust selective Gram–Schmidt reorthogonalization. À paraître cette année dans *SIAM Journal of Scientific Computing (2003)*.

4. Luc Giraud, Serge Gratton and Julien Langou (2003). A reorthogonalization procedure for modified Gram–Schmidt algorithm based on a rank– k update. Soumis à *SIAM Journal of Matrix Analysis and Applications*.

5. Valérie Frayssé, Luc Giraud, Serge Gratton and Julien Langou (2003). A set of GMRES routines for real and complex arithmetics on high performance computers. Soumis à *ACM Transactions on Mathematical Software (TOMS)*.

Contents

I

Chapter 1

Study of the Gram–Schmidt algorithm and its variants

1.1 Presentation of the Gram–Schmidt algorithm

1.1.1 Projection

The first part of this manuscript is devoted to the study of the Gram–Schmidt algorithm. The basic operation of the Gram–Schmidt algorithm is the projection. We propose here a brief overview of what a projection is and what its properties are.
In $\mathbb{R}^m$, a *projection* M is a (square) matrix M such that $M^2 = M$. For geometrical reasons we say that the matrix M represents the oblique projection onto $\text{Range}(M)$ along $\text{Null}(M)$. This implies the following relation:

$$\text{Null}(M) \oplus \text{Range}(M) = \mathbb{R}^m.$$

If $x \in \text{Range}(M), Mx = x$ so that a projection is diagonalizable with $E_1 = \text{Range}(M)$ and $E_0 = \text{Null}(M)$, where E_1 is the invariant subspace associated with the eigenvalue one and E_0 is the eigenspace associated with the eigenvalue zero. From this, it follows that $\text{rank}(M) = \text{trace}(M)$.
For any nonzero vector a of size m, it is possible to define the projection M onto the orthogonal complement of a along a. The range of such a projection is $\text{Range}(M) = \{x \in \mathbb{R}^m \text{ such that } x \perp a\} = a^\perp$. The null space is $\text{Null}(M) = \text{Span}(\{a\})$. Since $\text{Range}(M) \perp \text{Null}(M)$, M is called an *orthogonal* projection. Moreover we call it *elementary* since the null space is of rank one. M is given via the formula

$$M = I_m - zz^T \quad \text{where } z = \frac{a}{\|a\|_2}. \tag{1.1}$$

Given an orthonormal basis $\{q_1, \ldots, q_n\}$ of a subspace $\mathcal{Q}$, the orthogonal projection M onto the orthogonal complement of $\mathcal{Q}$ is given by

$$M = I_m - QQ^T = I_m - q_1 q_1^T - \ldots - q_n q_n^T \,, \tag{1.2}$$

where $Q = (q_1, \ldots, q_n)$.
Regarding the orthogonal projections we have the following properties:

Property 1.1.1 *For all x, for M verifying equation (1.2)*

1. *$I_m - M$ is the orthogonal projection onto the range of $\mathcal{Q}$.*
2. *Theorem of Pythagoras: $\|Mx\|_2^2 + \|(I_m - M)x\|_2^2 = \|x\|_2^2$.*
3. *In particular we have $\|Mx\|_2 \leq \|x\|_2$. The reciprocal is also true: if a projection does not lengthen any distance, then it is an orthogonal projection.*
4. *The matrix M is symmetric: $M = M^T$. The reciprocal is also true: if a projection is symmetric then it is an orthogonal projection.*

1.1.2 The classical Gram–Schmidt orthogonalization process

It is intuitively plausible that a set of linearly independent vectors (which forms a basis for their span) may be replaced by an orthonormal basis to span the same space. The following theorem holds: *Given a Euclidean space, for any set of linearly independent vectors, there exists an orthonormal basis that spans this set.* There is existence, but note that, (if the set is not a singleton) there is an infinite number of orthonormal basis that comply with this assumption. The proof of this theorem is given by construction (i.e. when proving the existence, we construct a basis) and the algorithm used is in general the *Gram–Schmidt (orthogonalization) process.* It is a very simple and far–reaching algorithm for replacing the initial set of vectors. The Gram–Schmidt algorithm was originally given by Schmidt [120, Section 5]. The Gram–Schmidt algorithm holds in any Euclidean space. In order to highlight this fact, in this section, we consider a Euclidean space with the scalar product $< ., . >$. Let $\{a_1, \ldots, a_n\}$ be a set of n linearly independent vectors. We are interested in computing the orthonormal basis $\{q_1, \ldots, q_n\}$ such that $\text{Span}(\{q_1, \ldots, q_n\}) = \text{Span}(\{a_1, \ldots, a_n\})$. The strategy of the Gram–Schmidt process is to construct at each step $j = 1, \ldots, n$ the vectors q_j such that

$$\text{Span}(\{q_1, \ldots, q_j\}) = \text{Span}(\{a_1, \ldots, a_j\}). \tag{1.3}$$

For $j = 1$, we choose

$$q_1 = \frac{a_1}{< a_1, a_1 >^{\frac{1}{2}}}.$$

Let $w_2 = a_2 - q_1 < q_1, a_2 >$ so that w_2 is orthogonal to q_1 and belongs to $\text{Span}(\{q_1, a_2\}) = \text{Span}(\{a_1, a_2\})$. We choose

$$q_2 = \frac{w_2}{< w_2, w_2 >^{\frac{1}{2}}},$$

so that q_2 is normalized, q_2 is orthogonal to q_1 and equation (1.3) holds for $j = 2$. The process continues similarly. Assuming $q_1, \ldots, q_{j-1}$ have been computed, let

$$w_j = a_j - q_1 < q_1, a_j > - \ldots - q_{j-1} < q_{j-1}, a_j >, \tag{1.4}$$

so that w_j is orthogonal to $q_1, \ldots, q_{j-1}$ and belongs to $\text{Span}(\{q_1, \ldots, q_{j-1}, a_j\}) = \text{Span}(\{a_1, \ldots, a_{j-1}, a_j\})$. Again we normalize w_j to get

$$q_j = \frac{w_j}{< w_j, w_j >^{\frac{1}{2}}}. \tag{1.5}$$

We continue until the desired orthonormal vectors $q_1, \ldots, q_n$ have been produced. Note that an infinite orthonormal set of vectors could be produced from a countably infinite linearly independent set of vectors in an infinite–dimensional space in this way.
In this manuscript, we are interested in matrix analysis. Consequently, for the remainder of this document, unless clearly stated, the scalar product is the Euclidean scalar product, that is :

$$< x, y >= x^T y = \sum_{i=1}^{m} x_i^T y_i.$$

If we denote by R the n–by–n upper triangular matrix such that

$$\text{for all } j = 1, \ldots, n \quad r_{ij} = q_i^T a_j, \quad i < j \quad \text{and} \quad r_{jj} = \|w_j\|_2$$

and A (resp. Q) the m–by–n matrix with j^{th} column vector a_j (resp. q_j). We obtain $A = QR$, where $Q^T Q = I_n$ and R is upper triangular with positive diagonal entries; that is, the QR factorization of A. The upper triangularity of the R factor comes directly from equation (1.3). The QR factorization of a matrix is essentially unique in the sense that given a couple (Q, R) of QR factors for the QR factorization of A, the set of couples $\{(QD, -DR)$, where D are diagonal matrices with diagonal entries $d_{ii} = \pm 1\}$ describes all the QR factors for the QR factorization of A. In practice, in the Gram–Schmidt algorithm, we force the diagonal entries of R to be positive. This justifies the sign $+$ in the definition (1.5) of q_j.
Finally we note that the Gram–Schmidt orthogonalization process may be applied to a linearly dependent set of vectors as well. In this case, at least one j exists such that $w_j = 0$ so that (1.5) cannot be performed. Replacing a_j with a_{j+1} and continuing the process enables us to complete the algorithm. With this strategy, the algorithm ends up with p orthonormal vectors $q_1, \ldots, q_p$ such that $\text{Span}(\{q_1, \ldots, q_p\}) = \text{Span}(\{a_1, \ldots, a_n\})$. The value of the integer p corresponds to the rank of the set of vectors $\{a_1, \ldots, a_n\}$. This illustrates the fact that the Gram–Schmidt algorithm can be used to determine the rank of a matrix in theory.

Algorithm 1 Classical Gram–Schmidt algorithm – (CGS)

```
1.  Q = A
2.  for j = 1 to n do
3.      for i = 1 to j − 1 do
4.          r_ij = q_i^T a_j
5.          q_j = q_j − q_i r_ij
6.      end for
7.      r_jj = ‖q_j‖_2
8.      q_j = q_j / r_jj
9.  end for
```

The number of flops to perform this algorithm is about $2mn^2$ flops.
If the R factor is not needed, it is not necessary to store it during the Gram–Schmidt process. Once q_j is produced, a_j is not needed any longer. So if the matrix A is not needed at the end of the factorization, the matrix Q can be stored in place of A.

Note that the i–loop can be performed via

$$r(1:j,j) = Q(:,1:j)^T a_j \quad \text{and} \quad w = a_j - Q(:,1:j)r(1:j,j).$$

In the presence of rounding errors, the property of the QR factors might be lost. We call $(\bar{Q}, \bar{R})$ the computed QR factors for A. In this thesis, we are interested in quantifying

1. **The quality of the factorization.**
   does the factorization $\bar{Q}\bar{R}$ represent A *well* ?
2. **The quality of the Q factor.**
   is $\bar{Q}$ orthogonal to a *certain* level and does an upper triangular matrix $\hat{R}$ exist such that $\bar{Q}\hat{R}$ represents A *well* ?
3. **The quality of the R factor.**
   $\bar{R}$ is triangular by construction, but does a $\hat{Q}$ exist with orthonormal columns such that $\bar{Q}\hat{R}$ represents A *well* ?

These three questions have already been answered in the past for some variants of the Gram–Schmidt algorithm. In the first part of this manuscript, we establish new results that answer these questions for other variants.

1.1.3 The modified Gram–Schmidt orthogonalization process

At step j, let us define $Q_{j-1} = [q_1, \ldots, q_{j-1}]$. The projection $(I_m - Q_{j-1}Q_{j-1}^T)$ can also be computed via

$$(I_m - Q_{j-1}Q_{j-1}^T) = (I_m - q_{j-1}q_{j-1}^T)\ldots(I_m - q_1 q_1^T). \tag{1.6}$$

This is a consequence of the fact that the columns of $Q_{j-1} = [q_1, \ldots, q_{j-1}]$ are orthonormal. In other words, the orthogonal projection onto the orthogonal complements of $\{q_1, \ldots, q_{j-1}\}$ can be obtained by successively projecting onto the orthogonal complement of each individual $q_i, i = 1, \ldots, j-1$. These elementary projections commute. In equation (1.6), we have arbitrarily fixed the order from 1 to $j-1$. From equation (1.6), step (1.4) can also be computed via

$$w_j = (I_m - q_{j-1}q_{j-1}^T)\ldots(I_m - q_1 q_1^T)a_j. \tag{1.7}$$

This formulation gives rise to the modified Gram–Schmidt algorithm described in Algorithm 2.

There is not much difference between the classical Gram–Schmidt algorithm (often denoted CGS) (see Algorithm 1) and the modified Gram–Schmidt algorithm (often denoted MGS) from an algorithmic point of view. The only step that differs is step 4 where $r_{ij} = q_i^T a_j$, in the classical version, is replaced by $r_{ij} = q_i^T q_j$ in the modified version. Higham [75] quotes Wilkinson who admitted that "*I used the modified process for many years without even noticing explicitly that I was not performing the classical algorithm.*" In 1966, Rice [109] was the first to point out the difference between the two algorithms.

From the mathematical point of view, both algorithms provides exactly the same QR factorization, in the same numbers of flops and the same amount of memory. We

Algorithm 2 Modified Gram–Schmidt algorithm – (MGS)

```
1.  Q = A
2.  for j = 1 to n do
3.      for i = 1 to j − 1 do
4.          r_ij = q_i^T q_j
5.          q_j = q_j − q_i r_ij
6.      end for
7.      r_jj = ‖q_j‖_2
8.      q_j = q_j / r_jj
9.  end for
```

notice that in the modified Gram–Schmidt algorithm in the form presented in Algorithm 2, Level 2 BLAS cannot be used. This problem is addressed in Section 1.1.6.

Equation (1.6) explains why the classical projection and the modified projection are equal. This happens because the set of vectors $\{q_1, \ldots, q_{j-1}\}$ is orthonormal. If the set of vectors $\{q_1, \ldots, q_{j-1}\}$ is not orthonormal, equation (1.6) will not hold.

Rice [109] was the first to point out that the modified Gram–Schmidt method produces a more nearly orthonormal matrix than the classical Gram–Schmidt method in the presence of rounding errors.

1.1.4 The reorthogonalization versions

A solution to improve the orthogonality among the vectors Q computed by the Gram–Schmidt algorithm is to iterate the projection phase until a certain criterion is true. In Section 1.5, a historical description of this class of algorithms is given. The general method is described in Algorithm 3.

Algorithm 3 Modified Gram–Schmidt algorithm with reorthogonalization and selective reorthogonalization

```
1.  Q = A
2.  R = 0_{n,n}
3.  for j = 1 to n do
4.      repeat
5.          for i = 1 to j − 1 do
6.              α = q_i^T q_j
7.              r_ij = r_ij + α
8.              q_j = q_j − q_i α
9.          end for
10.     until (selective criterion is true)
11.     r_jj = ‖q_j‖_2
12.     q_j = q_j / r_jj
13. end for
```

1.1.5 The square–root free versions

In 1907, Schmidt [120, Section 5][1] originally gave the classical algorithm in its square–root free version. The square–root free name comes from the fact that the elementary projection (1.1) on the orthogonal complement of a can also be written as

$$M = I_m - \frac{aa^T}{a^T a}. \tag{1.8}$$

Given an orthogonal[2] set of vectors $\{q_1, \ldots, q_{j-1}\}$ the general orthogonal projection on the orthogonal complement of $\{q_1, \ldots, q_{j-1}\}$ is similarly given by

$$M = I_m - \frac{q_1 q_1^T}{q_1^T q_1} - \ldots - \frac{q_{j-1} q_{j-1}^T}{q_{j-1}^T q_{j-1}}. \tag{1.9}$$

This enables us to derive the classical Gram–Schmidt algorithm in its square–root free version (see Algorithm 4). At step j, the algorithm computes

$$q'_j = a_j - q'_1 \frac{(q'_1)^T a_j}{(q'_1)^T q'_1} - \ldots - q'_{j-1} \frac{(q'_{j-1})^T a_j}{(q'_{j-1})^T q'_{j-1}}, \tag{1.10}$$

q'_j is equal to w_j in equation (1.10). The vector q'_j is not normalized, so that no square–root is used. The vector q'_j is orthogonal to $(q'_1, \ldots, q'_{j-1})$ and we have $\mathrm{Span}(\{q'_1, \ldots, q'_j\}) = \mathrm{Span}(\{a_1, \ldots, a_j\})$.
If we denote by R' the upper triangular matrix such that

$$r'_{ij} = ((q'_i)^T) a_j, \quad i < j \quad \text{and} \quad r'_{jj} = 1 \,,$$

we obtain

$$A = Q'R', \quad \text{with } (Q')^T Q \text{ diagonal and } R' \text{ unit upper triangular.}$$

The link between (Q, R), the QR factors from Gram–Schmidt algorithm, and (Q', R'), the factors from the square–root free Gram–Schmidt algorithm, is

$$\text{for all } j = 1, \ldots, n \text{ and } i = 1, \ldots, j \quad q_j = q'_j / \|q'_j\|_2 \text{ and } r_{jj} = \|q'_j\|_2 r'_{jj}.$$

In matrix format, we obtain $Q = Q'D^{-1}$ and $R = DR'$ where D is the diagonal matrix with entries $d_{jj} = \|q'_j\|_2$.
The modified square–root free version can also be derived.
Note that the square–root free version saves the scaling operation $q_j = q_j / r_{jj}$. In the context where n is very small $n = 2, 3, \ldots$, this might be an interesting gain.

1.1.6 The row oriented modified Gram–Schmidt algorithm

The version of the modified Gram–Schmidt algorithm presented in Algorithm 2 is referred to as the column oriented modified Gram–Schmidt algorithm. In this form,

[1] He quotes Chapter 3 of his own dissertation and J. P. Gram, *Ueber die Entwickelung reeler Functionen in reihen mittelst der Methods der kleinsten Quadrate* [Journal für die reine and angewandte Mathematik, Bd, XCIV (1883), S. 41–73].

[2] orthonormal is fine but orthogonal is enough.

Algorithm 4 The classical Gram–Schmidt algorithm – square–root free version

1. $Q' = A$
2. **for** $j = 1$ to n **do**
3. **for** $i = 1$ to $j - 1$ **do**
4. $r_{ij} = ((q'_i)^T q'_j)/d_i$
5. $q'_j = q'_j - q'_i r_{ij}$
6. **end for**
7. $r_{jj} = 1$
8. $d_j = (q'_j)^T q'_j$
9. **end for**

the two loops perform $n(n-1)/2$ dot operations and $n(n-1)/2$ axpy operations in a completely sequential way. MGS uses only Level 1 BLAS operations whereas CGS uses Level 2 BLAS operations.
To remedy this drawback, we exchange the i–loop with the j–loop, and the resulting algorithm is known as the row oriented modified Gram–Schmidt algorithm (MGSR). This variant is described in Algorithm 5.

Algorithm 5 The row oriented modified Gram–Schmidt algorithm – (MGSR)

1. $Q = A$
2. **for** $i = 1$ to n **do**
3. $r_{ii} = \|q_i\|_2$
4. $q_i = q_i / r_{ii}$
5. **for** $j = i + 1$ to n **do**
6. $r_{ij} = q_i^T q_j$
7. $q_j = q_j - q_i r_{ij}$
8. **end for**
9. **end for**

This algorithm requires exactly the same number of flops as MGS but the j–loop can be rewritten as $r(i, i+1:n) = Q(:,i)^T Q(:, i+1:n)$ and $Q(:, i+1:n) = Q(:, i+1:n) - Q(:,i) r(i, i+1:n)$. This allows us to use Level 2 BLAS operations.
At each loop i, the i^{th} row of the R factor is produced. This is in contrast with the column oriented version where, at each loop i, the i^{th} column of the R factor is produced.
An important remark is that the row oriented modified Gram–Schmidt algorithm requires the knowledge of all the columns of A in advance. This is a nontrivial assumption that may prevent us from using this algorithm. For instance, in the Arnoldi process, the column j of A comes from the previous column $(j-1)$ of Q. In such a case only the column oriented version of the modified Gram–Schmidt algorithm is applicable.

1.1.7 Others variants

Jalby and Philippe [79] studied the stability of the block modified Gram–Schmidt algorithm. The vectors are gathered by blocks and the modified Gram–Schmidt algorithm is performed on the blocks of vectors. This variant appears naturally in the context of the block–Arnoldi method [119].

Since the Gram–Schmidt algorithm is defined for any scalar product, it is particularly useful in many more situations than the preceding analysis may let us think. Considering a symmetric positive definite matrix A of order m, a natural scalar product is the A–scalar product:

$$< x, y >_A = x^T A y.$$

A variant of the Gram–Schmidt algorithm is easily derived with this scalar product. For any m–by–n matrix B, it enables us to find Q and R such that

$$B = QR \quad Q^T A Q = I_n \quad \text{where } R \text{ is upper triangular.}$$

Finally, we shall also cite Nilsson [96] who gives a variant of the modified Gram–Schmidt algorithm to find a biorthogonal basis in the context of the Lanczos algorithm.

1.1.8 Other candidates to perform a QR factorization

Classically to perform a QR factorization, two other candidates exist:

QR factorization via Givens rotations,

QR factorization via Householder transformations.

The idea of both methods is to use unitary transformations to triangularize the matrix. First of all, these methods are much more stable and Wilkinson [136] showed that these factorizations were backward stable.
The Givens rotations are particularly efficient when the R factor of a sparse matrix (in fact only the lower triangular part needs to be sparse) is needed. For example, when one needs the QR factorization of a Hessenberg matrix, then it is recommended to use Givens rotations.
To obtain the R factor, the Householder algorithm requires $2n^2(m - n/3)$ flops (see e.g. [63]); to obtain the Q factor, it requires $2mn^2$ flops more. The MGS algorithm requires $2n^2m$ flops for Q and R.
We also notice that, when the scalar product is not the Euclidean scalar product but the A–scalar product, then if the matrix G is unitary in the A–unitary sense (i.e. $G^T A G = A$), its columns (and rows) are not A-orthogonal. Given a matrix W such that $W^T A W = I$ (W has A–orthogonal columns), unitary transformations with the A–scalar product provide Q and R such that (a) $B = QR$, (b) $W^T AR$ is triangular and (c) $Q^T AQ = A$. One may also write (a) $B = QR$, (b) R is (A, W)–triangular and (c) Q is A–unitary. When $A = I$ (Euclidean scalar product) and $W = I$, the QR factorization of a matrix can be obtained via unitary transformations.

1.2 Model of error analysis for projections

1.2.1 Floating–point arithmetic

To carry out the rounding error analysis of an algorithm, we need to make some assumptions about the arithmetic we use. The arithmetic studied here is based on the IEEE[3] 754 standard for binary floating–point arithmetic [1]. This standard holds on almost all computers. In this section, we give some points about this arithmetic. The motivation is to explain where and why roundoff errors take place and how they could be controlled. Many more details can be found in Higham [75] or Overton [98]. The latter book is devoted solely to the description of the IEEE floating–point arithmetic.

1.2.1.1 Binary floating–point representation of numbers

Every real number has a decimal representation and a binary representation (indeed a representation in a base equal to any integer greater than 1). This result comes from Euclid and the Euclidean division. For example, for the number[4] $25/8 = 3.125$ we have the decimal representation $(3.125)_{10}$ since

$$25/8 = (3.125)_{10} = 3 \times 10^0 + 1 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}$$

and its binary equivalent is $(11.001)_2$ since

$$25/8 = (11.001)_2 = 2^1 + 2^0 + 2^{-3}.$$

If every real number has a binary representation, it may not terminate (e.g. $8/25$). A computer using binary floating–point arithmetic represents a set of numbers with a certain number of bits, a bit being a figure that takes the value 0 or 1. In our case, we consider that 32 bits are used.
First of all we are interested in representing a *signed* integer using a 32–bit word. Since $2003 = 1024+512+256+128+64+16+2+1$, we have the binary representation

$$(2003)_2 = +11111010011\ ,$$

and so using a 32–bit word we may represent 2003 with

$$\boxed{0}\boxed{00000000000000000000011111010011},$$

where the first bit represents the sign of the number with the convention $+ \leftrightarrow 0$ and $- \leftrightarrow 1$, and the last 31 bits represent 2003 in binary representation. A first remark is that this 32–bit integer format is able to represent only integers that range from $-(2^{31}-1)$ to $2^{31}-1$. One may notice that 0 has two representations ($\boxed{0}\boxed{0\ldots}$ and $\boxed{1}\boxed{0\ldots}$). Therefore we can use the second representation of 0 to represent -2^{31} (say). This finally enables us to represent all signed numbers ranging from -2^{31} to $2^{31}-1$. This representation of integer is not the one usually used but is enough for the presentation made here. To avoid this problem while

[3] Institute for Electrical and Electronic Engineers
[4] when nothing is said, the decimal representation is used

being rigorous, we denote by $\text{ebits}_{32}(E)$ the representation of an integer E with a 32–bit word.
The binary floating–point representation is based on the *exponential* (or *scientific*) notation. In exponentional notation a nonzero number x is expressed in binary from as

$$x = \pm S \times 2^E, \quad \text{where } 1 \leq S < 2.$$

S is called the significand and the binary expansion of the significand is

$$S = (b_0.b_1b_2b_3\ldots)_2 \quad \text{with } b_0 = 1.$$

E is an integer called the exponent. For example, the number $25/8$ is expressed as

$$\frac{25}{8} = +(1.1001)_2 \times 2^1.$$

In order to represent numbers, the IEEE 754 single precision format uses the binary floating–point representation and a 32-bit word. In this format, the first bit of the word is used to store the sign with the convention $+ \leftrightarrow 0$ and $- \leftrightarrow 1$, the 8 bits thereafter are dedicated to store the exponent in an integer format rather similar to the one explained previously; finally the 23 remaining bits represent the significand. Since the first digit of the significand b_0 is necessarily a 1, we do not store it, b_0 is implicitly set to 1. We call $t = 24$ the precision of the arithmetic, it corresponds to the 24 digits of the significand. Therefore the IEEE 754 single format represents $25/8$ with

0	$\text{ebits}_8(1)$	10010000000000000000000

.

All single precision format floating–point numbers describe a finite subset of the real numbers that we call F.
Since the exponent is coded with an 8–bit word, the smallest exponent represented should be -128 and the largest is 127. In practice, the exponents 11111110 and 11111111 are reserved for other purposes than representing the integer -128 and -127. Consequently the exponent E ranges from -126 to 127. In this case, the smallest positive floating–point number in magnitude is

$$N_{min} = (1.000\ldots0)_2 \times 2^{-126} = 2^{-126} \approx 1.2 \times 10^{-38},$$

and the maximum positive floating–point number in magnitude is

$$N_{max} = (1.111\ldots1)_2 \times 2^{127} \approx 2^{128} \approx 3.4 \times 10^{38}.$$

Let x be a real number such that $N_{min} \leq |x| \leq N_{max}$, since the significand is coded with a 23–bit word x may not have a representation in floating–point arithmetic. Either its binary expansion is not finite or its binary expansion has more than 24 significant digits. For example, let us consider $x = 8/25 = 0.32$. The binary expansion of x is not finite, we have

$$\frac{8}{25} = (0.00\underline{10100011110101110000})_2$$

where the period is underlined.
For any real number x, we call $fl(x)$ the nearest element to x of F. for example for $x = 8/25 = 0.32$ the nearest floating point number $fl(x)$ is

0	ebits(−2)	01000111101011100001010

which in decimal representation gives 0.319999992847442626953125.
For any real number x such that $N_{min} \leq |x| \leq N_{max}$, $fl(x)$ represents x well in the following sense:

Theorem 1.2.1 *If $x \in \mathbb{R}$ is such that $N_{min} \leq |x| \leq N_{max}$ then there exists a real δ such that*

$$fl(x) = x(1+\delta), \quad |\delta| < u = 2^{-t}.$$

There also exists a real η such that

$$fl(x) = \frac{x}{1+\eta}, \quad |\eta| < u = 2^{-t}.$$

Proofs can be found in [75, pp. 42-43]. The quantity u is called the unit roundoff. Note that, for base β, the exact formula is $u = \frac{1}{2}\beta^{1-t}$ which gives $u = 2^{-t}$ when binary floating–point arithmetic is used (i.e. when $\beta = 2$). If a real number x does not comply with the assumptions of the theorem because $0 < |x| < N_{min}$, we say that $fl(x)$ *underflows*. If a real number x does not comply with the assumptions of the theorem because $|x| > N_{max}$, we say that $fl(x)$ *overflows*[5].
The limitations of the single precision format to represent the set of real numbers are the consequence of the fact that the size of the word used to represent a number is finite. A simple solution to improve the field of our representation is to increase the number of bits in a word. Another standard exists in this sense, it is the double precision format. The double precision binary format uses 64–bit words to represent floating–point numbers, the first bit of the word is used to store the sign, the 11 bits thereafter are dedicated to store the exponent, and the 52 last bits are used to represent the significand. In this case, $N_{min} \approx 2.2 \times 10^{-308}$, $N_{max} \approx 1.8 \times 10^{308}$ and $u = 2^{-53}$.
This presentation of the IEEE 754 binary floating–point representation omitted some important features of this standard. For example, the representation of 0, $\pm\infty$, the special number NaN[6] and subnormal numbers. We do not talk about the rounding modes either.

1.2.1.2 A model of arithmetic.

Given two floating–point numbers x and y belonging to F, we are now concerned with the operation x op y, where 'op' designates any of the four arithmetic operators $+ \ - \ * \ /$. The most common assumptions (e.g. [13, 75, 137]) are embodied in the following model

STANDARD MODEL

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\delta), \quad |\delta| < u. \tag{1.11}$$

[5] the subnormals numbers are not taken into account in this description also they are described in the IEEE 754 standard
[6] which stands for "Not a Number"

A floating–point arithmetic verifying this model is called a *well-designed arithmetic* ([13]). Note that the following modified version of (1.11) can also be used

$$fl(x \text{ op } y) = \frac{x \text{ op } y}{1+\delta}, \quad |\delta| < u. \tag{1.12}$$

Equation (1.12) can also be rewritten

$$fl(x \text{ op } y) = (x \text{ op } y)\frac{(1+\delta-\delta)}{1+\delta} = (x \text{ op } y)(1 - \frac{\delta}{1+\delta}).$$

Equation (1.12) is used for example by Daniel, Gragg, Kaufman and Stewart [34]. It leads naturally to a bound

$$|fl(x \text{ op } y) - (x \text{ op } y)| \leq \frac{u}{1+u}|x \text{ op } y|$$

that is better than

$$|fl(x \text{ op } y) - (x \text{ op } y)| \leq u|x \text{ op } y|$$

coming from Equation (1.11). In the remainder of this manuscript, we use the two formulations indifferently.
The IEEE 754 standard imposes (1.11) and (1.12). Other features concerning IEEE 754 arithmetic exist and are omitted in this brief description, for example the treatment of exceptional situations such as the division by zero.

1.2.1.3 A choice of model for the square root operation.

Our matrix algorithms use mainly the four arithmetic operators $+ \ - \ * \ /$. Another operator that we also often need is the square root operator. The bound for the error made in extracting a square root naturally depends to some extent on the algorithm which is used. We do not wishe to enter into a detailed discussion of such algorithms. Following Higham [75, p. 44], *it is normal to assume that (1.11) and (1.12) hold also for the square root operation.* This is in agreement to what is proposed by Lawson and Hanson [87]. The IEEE 754 standard imposes this property (e.g. [75, p. 45] and [98, p.38]). For information and comparison, in [137], Wilkinson considers that

$$\begin{aligned} fl(\sqrt{x}) &= \sqrt{x} + \varepsilon \qquad \text{where} |\varepsilon| < (1.00001)2^{-t-1}, \\ fl(\sqrt{x}) &= \sqrt{x}(1+\varepsilon) \qquad \text{where} |\varepsilon| < (1.00001)2^{-t}, \end{aligned}$$

where t is the precision (for single precision format we recall that $t = 24$, double precision format gives $t = 53$). Wilkinson notice that: *in most matrix algorithms the number of square roots is small compared with the number of other operations and even a larger error would make little significant difference to the overall error bounds.*

1.2.2 Error analysis for basic linear algebra operations

1.2.2.1 Simplified expression for error bounds

A direct consequence of the models of arithmetic is that they frequently lead in the first instance to bounds of the form $\varepsilon \leq (1+2^{-t})^m$, where m is an integer. Following the technique introduced by Wilkinson [137] and sucessfully used after him, we simplify this expression by using the following result:

Lemma 1.2.2 *If* $|\delta_i| \leq u$ *and* $\rho_i = \pm 1$ *for* $i = 1, \ldots, m$, *and* $mu < 1$, *then*

$$\prod_{i=1}^{m}(1+\delta_i)^{\rho_i} = 1+\theta_m,$$

where

$$|\theta_m| \leq \frac{mu}{1-mu}$$

Proof : see Higham [75, Lemma 3.1].

Lemma 1.2.3 *If* $|\delta_i| \leq u$ *for* $i = 1, \ldots, m$, *and* $mu < 2$, *then*

$$\prod_{i=1}^{m}(1+\delta_i) = 1+\theta_m,$$

where

$$|\theta_m| \leq \frac{mu}{1-mu/2}$$

Proof : see Kiełbasiński and Schwetlick [83, 84] or Higham [75, Lemma 3.4]. Wilkinson [137] defines t_1 such that

$$t_1 = t - \log_2(1.06). \tag{1.13}$$

Then the following corollary holds

Corollary 1.2.4 *If* $|\delta_i| \leq 2^{-t}$ *for* $i = 1, \ldots, m$ *and* $m2^{-t} \leq 0.1$ *then*

$$\prod_{i=1}^{m}(1+\delta_i) = 1+\theta_m \quad \text{where } |\theta_m| \leq 1.06 \cdot 2^{-t} = 2^{-t_1}. \tag{1.14}$$

Proof : Notice that if $m \cdot 2^{-t} \leq 0.1$ then $(1-m\cdot 2^{-t}/2)^{-1} \leq 1.06$ and use Lemma 1.2.3. ♡

1.2.2.2 Error analysis of an inner–product

Theorem 1.2.5 *Let* $x = (x_i)_{i=1,\ldots,m}$ *and* $y = (y_i)_{i=1,\ldots,m}$ *be floating–point vectors of dimension* m *where*

$$m \cdot 2^{-t} \leq 0.1. \tag{1.15}$$

Then the following error bound for the computed inner–product of x *and* y *is valid [137, pp. 114–117]:*

$$|fl(x^T y) - x^T y| \leq m \cdot 2^{-t_1}|x|^T|y| \leq m \cdot 2^{-t_1}\|x\|_2\|y\|_2. \tag{1.16}$$

Proof : The order in which the additions take place in the summation is not at all trivial. A deep study of the summation algorithm can be found in Higham [75, chap 4.]. In this manuscript, it is assumed that the operations take place in the order

$$\begin{aligned} &p_1 = x_1 \times y_1, \quad && s_1 = p_1, \\ &p_2 = x_2 \times y_2, \quad && s_2 = s_1 + p_2, \\ &\vdots \\ &p_m = x_m \times y_m, \quad && s_m = s_{m-1} + p_m. \end{aligned}$$

This is known as *recursive summation* and it is the model used for instance by Wilkinson [137, p. 114]. Using floating–point arithmetic, we have

$$\begin{aligned} fl(x^T y) &= fl(\ldots fl(fl(fl(x_1y_1) + fl(x_2y_2)) + fl(x_3y_3))\ldots), \\ &= fl(\ldots fl((x_1y_1(1+\varepsilon_1^{(+)}) + x_2y_2(1+\varepsilon_2^{(+)}))(1+\varepsilon_2^{(\times)}) + fl(x_3y_3))\ldots), \\ &= x_1y_1(1+\varepsilon_1^{(+)})(1+\varepsilon_2^{(\times)})\ldots(1+\varepsilon_m^{(\times)}) \\ &\quad +x_2y_2(1+\varepsilon_2^{(+)})(1+\varepsilon_2^{(\times)})\ldots(1+\varepsilon_m^{(\times)}) \\ &\quad +x_3y_3(1+\varepsilon_3^{(+)})(1+\varepsilon_3^{(\times)})\ldots(1+\varepsilon_m^{(\times)}) \\ &\quad +x_my_m(1+\varepsilon_m^{(+)})(1+\varepsilon_m^{(\times)}), \end{aligned}$$

where $|\varepsilon_i^{(+,\times)}| \leq 2^{-t}$, $i = 1, \ldots, m$. Using Corollary 1.2.4, one can write

$$fl(x^T y) = x_1y_1(1+\theta_1) + x_2y_2(1+\theta_2) + \ldots + x_my_m(1+\theta_m).$$

where

$$|\theta_1| \leq m \cdot 2^{-t_1}, \quad |\theta_2| \leq m \cdot 2^{-t_1}, \quad |\theta_3| \leq (m-1) \cdot 2^{-t_1}, \quad \ldots \quad |\theta_3| \leq 2 \cdot 2^{-t_1}.$$

so we guess that

$$|fl(x^T y) - x^T y| \leq \sum_{i=1}^{m} |\theta_i| \cdot |x_i| \cdot |y_i| \leq m \cdot 2^{-t_1} \sum_{i=1}^{m} |x_i| \cdot |y_i| = m \cdot 2^{-t_1} |x|^T |y|.$$

♡

1.2.2.3 Error analysis of a matrix–vector multiplication

The matrix–vector product of the m–by–n matrix A by the n–vector x gives

$$\tilde{y} = Ax = \sum_{k=1}^{n} a_k x_k,$$

where a_k denotes the k–th column of A. Starting from $y^{(0)} = 0$, the matrix–vector operation results in n axpy operations

$$\tilde{y}^{(k)} = \tilde{y}^{(k-1)} + a_k x_k, \quad k = 1, \ldots, n$$

to eventually give $\tilde{y} = \tilde{y}^{(n)} = Ax$. Using a well designed floating–point arithmetic, Daniel, Gragg, Kaufman and Stewart [34] considered that each of these axpy operations are performed with an error vector $e^{(k)}$, so that

$$y^{(k)} = y^{(k-1)} + a_k x_k + e^{(k)}, \quad k = 1, \ldots, n.$$

The error vector is controlled in each axpy operation by the relation

$$\|e^{(k)}\|_2 \leq \alpha \|y^{(k-1)}\|_2 + \beta \|a_k\|_2 |x_k, | \quad k = 1, \ldots, n,$$

where α and β are constants depending on the machine precision and the values of m and n. They are set in a second step. First of all, we give the following Theorem. It gives an upper bound for the 2–norm of e, the error vector that appears in computing $y = fl(Ax)$:

$$y = fl(Ax) = Ax + e.$$

Theorem 1.2.6 *Let $A \in \mathbb{R}^{m\times n}, x \in \mathbb{R}^n$ and let $y \in \mathbb{R}^m$ be the results of the algorithm*

$y^{(0)} = 0,$
for $k = 1, 2, \ldots, n$
$\quad y^{(k)} = y^{(k-1)} + a_k x_k + e^{(k)}$
end
$y = y^{(n)}$

in which the (error) vectors $e^{(k)}, k = 1, \ldots, n$ satisfy

$$\|e^{(k)}\|_2 \leq \alpha \|y^{(k-1)}\|_2 + \beta \|a_k\|_2 |x_k|. \tag{1.17}$$

Then $y = Ax + e$ with

$$\|e\|_2 \leq \left[(n-1)\alpha + \min(m^{1/2}, n^{1/2})\beta\right] (1+\alpha)^{n-1} \|A\|_2 \|x\|_2. \tag{1.18}$$

Proof : see Daniel, Gragg, Kaufman and Stewart [34].
In order to use Theorem 1.2.6, it remains for us to set the value for the quantity α and β in (1.17). If we consider that the operations are performed in floating–point arithmetic with the unit round-off u, we have

$$y_i^{(k)} = fl(y_i^{(k-1)} + a_{ik}x_k) = \left(y_i^{(k-1)} + a_{ik}x_k(1+\delta)\right)(1+\delta')$$

where $|\delta| \leq u/(1+u)$ and $|\delta'| < u$. So that

$$y_i^{(k)} = y_i^{(k-1)} + a_{ik}x_k + \left(y_i^{(k-1)}\delta' + a_{ik}x_k(\delta + \delta' + \delta\delta')\right).$$

Since, from equation (1.17), we have

$$y_i^{(k)} = y_i^{(k-1)} + a_{ik}x_k + e_i^{(k)},$$

we set

$$e_i^{(k)} = y_i^{(k-1)}\delta' + a_{ik}x_k(\delta + \delta' + \delta\delta').$$

Therefore

$$\begin{aligned} |e_i^{(k)}| &\leq |y_i^{(k-1)}|u + |a_{ik}||x_k|(u + \frac{u}{1+u} + \frac{u^2}{1+u}), \\ &\leq |y_i^{(k-1)}|u + |a_{ik}||x_k|(u + \frac{u(1+u-u)}{1+u} + \frac{u^2}{1+u}), \\ &\leq |y_i^{(k-1)}|u + |a_{ik}||x_k|2u. \end{aligned} \tag{1.19}$$

This last inequality is true for $i = 1$ to n so we have

$$\|e^{(k)}\|_2 \leq \|y^{(k-1)}\|_2 u + \|a_k\|_2 |x_k| 2u.$$

If we set

$$\alpha = u \quad \text{and} \quad \beta = 2u$$

then we get the hypothesis of the Theorem $\|e^{(k)}\|_2 \leq \alpha\|y^{(k-1)}\|_2 + \beta\|a_k\|_2|x_k|$ and so, using that Theorem, we have $y = Ax + e$ with

$$\|e\|_2 \leq \left[(n-1) + 2\min(m^{1/2}, n^{1/2})\right] u(1+u)^{n-1}\|A\|_2\|x\|_2 \quad , \tag{1.20}$$

and using Corollary 1.2.4, we have

$$\|e\|_2 \leq \left[(n-1) + 2\min(m^{1/2}, n^{1/2})\right] (n-1)2^{-t_1}\|A\|_2\|x\|_2 \quad . \tag{1.21}$$

Daniel, Gragg, Kaufman and Stewart [34] did not obtain equation (1.21) exactly since they considered an arithmetic such that

$$fl(a+b) = (a+b)(1+\delta) \quad \text{where } |\delta| \leq \frac{3}{2}u,$$
$$\text{and} \qquad fl(ab) = ab(1+\delta') \qquad \text{where } |\delta'| \leq \frac{u}{1+u}.$$

Note that their arithmetic is weaker than the one we actually use. With this arithmetic, the error for $y_i^{(k)}$ becomes

$$y_i^{(k)} = fl(y_i^{(k-1)} + a_{ik}x_k) = y_i^{(k-1)}(1+\delta') + a_{ik}x_k(1+\delta)(1+\delta'),$$

with $|\delta| \leq 3u/2$ and $|\delta'| < u/(1+u)$. One may observe that

$$\begin{aligned} |\delta + \delta' + \delta\delta'| &\leq \frac{3}{2}u + \frac{3}{2}\frac{u^2}{1+u} + \frac{u}{1+u} \\ &= \frac{3}{2}u + \frac{3}{2}\frac{u^2}{1+u} + u - \frac{u^2}{1+u} \\ &= \frac{5}{2}u + \frac{1}{2}\frac{u^2}{1+u} \\ &= \frac{5}{2}u(1 + \frac{1}{5}\frac{u}{1+u}) = \frac{5}{2}u' \end{aligned}$$

where u' is defined as $u' = u(1+u/5(1+u))$. The values for α and β given in [34] are consequently

$$\alpha = \frac{3}{2}u \quad \text{and} \quad \beta = \frac{5}{2}u'.$$

Thus

$$\|e\|_2 \leq \frac{1}{2}\left[3(n-1)+5\min(m^{1/2},n^{1/2})(1+\frac{1}{5}\frac{u}{1+u})\right]u(1+\frac{3}{2}u)^{n-1}\|A\|_2\|x\|_2.$$

Obviously $[5(1+u)]^{-1} \leq 3/2$ so

$$\begin{aligned}\|e\|_2 &\leq \frac{1}{2}\left[3(n-1)+5\min(m^{1/2},n^{1/2})(1+\frac{3}{2}u)\right]u(1+\frac{3}{2}u)^{n-1}\|A\|_2\|x\|_2 \\ &\leq \frac{1}{2}\left[3(n-1)+5\min(m^{1/2},n^{1/2})\right]u(1+\frac{3}{2}u)^{n}\|A\|_2\|x\|_2 \\ &\leq \frac{1}{2}\left[3(n-1)+5\min(m^{1/2},n^{1/2})\right]u_1\|A\|_2\|x\|_2 \qquad (1.22)\end{aligned}$$

with $u_1 = u(1+\frac{3}{2}u)^n$. Inequality (1.22) should be compared with (1.21). Both formulae are similar, only the constants are different.

1.2.2.4 Error analysis of the normalization of a vector

Theorem 1.2.7 *Let x be a floating–point vector of size m where m is such that*

$$(m+4)\cdot 2^{-t} \leq 0.1 \qquad (1.23)$$

then

$$|fl(\|x\|_2) - \|x\|_2| \leq \frac{m+2}{2}2^{-t_1}\|x\|_2. \qquad (1.24)$$

Moreover, if $x \neq 0$, we have

$$| \quad \|fl(x/\|x\|_2)\|_2^2 - 1 \quad | \leq (m+4)2^{-t_1}, \qquad (1.25)$$

$$\textit{and} \quad | \quad \|fl(x/\|x\|_2)\|_2 \quad | \leq 1 + \frac{m+4}{2}2^{-t_1}. \qquad (1.26)$$

Proof : In the first part of the proof, we study the dot product x^Tx in floating–point arithmetic. For the error analysis we choose to use equation (1.12). Following what is done in Section 1.2.2.2 for x^Ty, we get that

$$fl(x^Tx) = \sum_{i=1}^{m}\frac{x_i^2}{(1+\varepsilon_i^{(+)})(1+\varepsilon_i^{(\times)})(1+\varepsilon_{i+1}^{(\times)})\dots(1+\varepsilon_m^{(\times)})}, \qquad (1.27)$$

where $|\varepsilon_i^{(+,\times)}| \leq 2^{-t}$, $i = 1,\dots,m$. From equation (1.27), we get

$$\frac{x^Tx}{(1+2^{-t})^m} \leq fl(x^Tx) \leq \frac{x^Tx}{(1-2^{-t})^m}.$$

Since for $-2^{-t} \leq \alpha \leq 2^{-t}$, $(1+\alpha)^{-m}$ is a continuous function of α, there exists ε_1, $-2^{-t} \leq \varepsilon_1 \leq 2^{-t}$, such that

$$fl(x^Tx) = \frac{x^Tx}{(1+\varepsilon_1)^m}.$$

Using the model of square root described in Section 1.2.1.3, we get that

$$fl(\|x\|_2) = fl(\sqrt{fl(x^T x)}) = \sqrt{fl(x^T x)}\frac{1}{(1+\varepsilon_2)},$$

where $|\varepsilon_2| \leq 2^{-t}$. This gives

$$fl(\|x\|_2) = \frac{\sqrt{x^T x}}{(1+\varepsilon_1)^{m/2}(1+\varepsilon_2)} = \frac{\|x\|_2}{(1+\varepsilon_1)^{m/2}(1+\varepsilon_2)}. \tag{1.28}$$

If equation (1.11) is used instead of equation (1.12) during the proof we obtain, instead of equation (1.28),

$$fl(\|x\|_2) = \|x\|_2(1+\varepsilon_1')^{m/2}(1+\varepsilon_2'),$$

where $|\varepsilon_1'| \leq 2^{-t}$ and $|\varepsilon_2'| \leq 2^{-t}$. Using Corollary 1.2.4, since $(m+2)\cdot 2^{-t} \leq 0.1$, we find that a real θ_1 exists such that

$$fl(\|x\|_2) = \|x\|_2\sqrt{(1+\theta_1)} \quad \text{where } |\theta_1| \leq (m+2)\cdot 2^{-t_1}.$$

Since for all real α, $\alpha \geq -1$, we have $\sqrt{1+\alpha} \leq 1+\alpha/2$, this gives

$$fl(\|x\|_2) = \|x\|_2(1+\theta) \quad \text{where } |\theta| \leq \frac{m+2}{2}\cdot 2^{-t_1}.$$

From this latter equation we directly obtain equation (1.24).
From equation (1.28), we control the error made in the computation of $fl(\|x\|_2)$. Since underflows are not taken into account, we see that, if $x \neq 0$, then $fl(\|x\|_2) \neq 0$. It is therefore possible to divide x by $fl(\|x\|_2)$. We study this last step.
For all $i = 1, \ldots, m$, there exists $\varepsilon_3^{(i)}$ such that

$$fl(\frac{x_i}{\|x\|_2}) = \frac{x_i}{fl(\|x\|_2)}(1+\varepsilon_3^{(i)}), \quad \text{where } |\varepsilon_3^{(i)}| \leq 2^{-t}.$$

Using equation (1.28), this gives, for all $i = 1, \ldots, m$,

$$fl(\frac{x_i}{\|x\|_2}) = \frac{x_i}{\|x\|_2}(1+\varepsilon_1)^{m/2}(1+\varepsilon_2)(1+\varepsilon_3^{(i)}).$$

We obtain

$$(1-2^{-t})^{\frac{m+4}{2}} \leq \|fl(\frac{x}{\|x\|_2})\|_2 \leq (1+2^{-t})^{\frac{m+4}{2}}.$$

Since $(m+4)\cdot 2^{-t} \leq 0.1$, squaring the latter inequality, we obtain using Corollary 1.2.4

$$1-(m+4)\cdot 2^{-t_1} \leq \|fl(\frac{x}{\|x\|_2})\|_2^2 \leq 1+(m+4)\cdot 2^{-t_1}.$$

Equation (1.25) and equation (1.26) follow directly. ☺

1.2.3 Error analysis of an elementary projection

If one wants to project y on the orthogonal complement of a, then it is natural to use equation (1.2) directly. This is what we do with floating–point arithmetic. Let us consider a and y, two floating–point vectors of size m with $a \neq 0$. We define the computed quantity $x' = fl(a/\|a\|_2)$. We define $\bar{r}'$ and z the computed quantities

$$\bar{r}' = fl((x')^T y) \quad \text{and} \quad z = fl(y - x'\bar{r}'). \tag{1.29}$$

z is the computed projection of y on the orthogonal complement of a.
The study of orthogonal elementary projection was done by Björck [13] in 1967. However, he used a square–root free version. Instead of equation (1.2) Björck took

$$M = I_m - \frac{aa^T}{a^T a}. \tag{1.30}$$

In this section we adapt his work to the square–root version since it is nowadays the most widespread version. Björck [13] introduced the three quantities

$$x = x'/\|x'\|_2, \quad \bar{r} = \bar{r}'\|x'\|_2 \quad \text{and} \quad r = x^T y. \tag{1.31}$$

We are interested in computing error bounds for the norm of δ and η where δ and η are defined such that

$$z = y - \bar{r}x + \delta \quad \text{and} \quad z = y - xr + \eta. \tag{1.32}$$

We note that the reference vector in these relations is x, which is exactly normalized to one, not x' the vector that is actually computed. This is certainly an elegant way introduced by Björck to translate the problem to an exact projection. The quantity $y - xr$ indeed represents the *exact* projection of y on the orthogonal component of x.
We are now going to prove the following two bounds:

$$\|\eta\|_2 \leq (2.1m + 6)2^{-t_1}\|y\|_2. \tag{1.33}$$

$$\|\delta\|_2 \leq 2.06 \cdot 2^{-t}\|y\|_2. \tag{1.34}$$

To estimate the error in the multiplier $\bar{r}$ we first study the axpy operations of definition (1.32). Now we have

$$z_i \;=\; (y_i - x_i'\bar{r}'(1+\varepsilon_1^{(i)}))(1+\varepsilon_2^{(i)}) \quad \text{for all } i = 1, \ldots, m,$$

where $|\varepsilon_1^{(i)}| \leq 2^{-t}$ and $|\varepsilon_2^{(i)}| \leq 2^{-t}$. Using equation (1.31) leads to

$$z_i \;=\; (y_i - x_i\bar{r}(1+\varepsilon_1^{(i)}))(1+\varepsilon_2^{(i)})$$

so that

$$y_i = \frac{z_i}{1+\varepsilon_2^{(i)}} + x_i\bar{r}(1+\varepsilon_1^{(i)}).$$

Using this to eliminate y_i from the definition of δ, Björck got

$$\delta_i = \frac{\varepsilon_2^{(i)}}{1+\varepsilon_2^{(i)}} z_i - \varepsilon_1^{(i)}\bar{r}x_i,$$

and hence, since $\|x\|_2 = 1$,

$$\|\delta\|_2 \leq \frac{2^{-t}}{1-2^{-t}}\|z\|_2 + 2^{-t}|\bar{r}|. \tag{1.35}$$

Immediately from equation (1.32) he had

$$\|\eta\|_2 \leq \|\delta\|_2 + |\bar{r} - r|. \tag{1.36}$$

Now our analysis differs slightly from that of Björck since we use a square–root version. The first goal is to obtain an upper bound for the quantity $|\bar{r} - r|$.

$$\begin{aligned} \bar{r} - r &= \bar{r}'\|x'\|_2 - r \\ &= fl((x')^T y)\|x'\|_2 - r \\ &= (fl((x')^T y) - (x')^T y)\|x'\|_2 + (\|x'\|_2^2 - 1)r. \end{aligned}$$

We use equation (1.16) to bound $(fl((x')^T y) - (x')^T y)$ and equation (1.25) to bound $(\|x'\|_2^2 - 1)$, we obtain

$$|\bar{r} - r| \leq m \cdot 2^{-t_1}\|x'\|_2^2\|y\|_2 + (m+4)2^{-t_1} \cdot |r|.$$

From equation (1.26) and equation (1.23), $\|x'\|_2 \leq 1.1$ so that

$$|\bar{r} - r| \leq ((m+4)|r| + 1.1m\|y\|_2)2^{-t_1}. \tag{1.37}$$

For the square–root version of the elementary projection, equation (1.37) corresponds to equation [13, Eq. (4.11)]. For the sake of comparison, we recall equation [13, Eq. (4.11)], that is:

$$|\bar{r} - r| \leq ((m+1)|r| + m\|y\|_2)2^{-t_1}. \tag{1.38}$$

The bound (1.38) is better than (1.37), however the difference is slight and it enables us to use the square–root.
We have a bound for $|\bar{r} - r|$, as we can continue to develop Björck's proof [13]. Since $(z - \eta) = y - xr$ is orthogonal to x it follows from the theorem of Pythagoras that

$$\|z - \eta\|_2^2 + r^2 = \|y\|^2,$$

so that

$$\|z\|_2 \leq (\|y\|_2^2 - r^2)^{\frac{1}{2}} + \|\eta\|_2. \tag{1.39}$$

Substituting δ in equation (1.36) by using equation (1.35), Björck got

$$\|\eta\|_2 \leq \frac{2^{-t}}{1-2^{-t}}\|z\|_2 + 2^{-t}|\bar{r}| + |\bar{r} - r|,$$

that, multiplied by $(1 - 2^{-t})$ gives

$$(1-2^{-t})\|\eta\|_2 \leq 2^{-t}\|z\|_2 + 2^{-t}(1-2^{-t})|\bar{r}| + (1-2^{-t})|\bar{r} - r|.$$

Obviously

$$(1-2^{-t})\|\eta\|_2 \leq 2^{-t}\|z\|_2 + 2^{-t}|\bar{r}| + (1-2^{-t})|\bar{r} - r|,$$

and so

$$(1-2^{-t})\|\eta\|_2 \leq 2^{-t}\|z\|_2 + 2^{-t}|r| + |\bar{r}-r|.$$

Then, adding $2^{-t}\|\eta\|_2$ to both sides, this enables us to use equation (1.39) and gives

$$(1-2\cdot 2^{-t})\|\eta\|_2 \leq 2^{-t}\left((\|y\|_2^2 - r^2)^{\frac{1}{2}} + |r|\right) + |\bar{r}-r|.$$

Hence, using equation (1.37), we have

$$(1-2\cdot 2^{-t})\|\eta\|_2 \leq \left((\|y\|_2^2 - r^2)^{\frac{1}{2}} + (m+5)|r| + 1.1m\|y\|_2\right) 2^{-t_1}. \tag{1.40}$$

Maximizing $f(r) = (\|y\|_2^2 - r^2)^{\frac{1}{2}} + k|r|$ over r, where $0 \leq |r| \leq \|y\|_2$, the maximum is attained for $r_{\max} = \pm\|y\|_2 \frac{k}{(1+k^2)^{1/2}}$. Its value is $f(r_{\max}) = \|y\|_2(1+k^2)^{1/2}$ so we obtain, that for $0 \leq |r| \leq \|y\|_2$,

$$(\|y\|_2^2 - r^2)^{\frac{1}{2}} + (m+5)|r| \leq \|y\|_2(1+(m+5)^2)^{1/2}.$$

Going back to equation (1.40) gives

$$(1-2\cdot 2^{-t})\|\eta\|_2 \leq \left((1+(m+5)^2)^{1/2} + 1.1m\right) 2^{-t_1}\|y\|_2 ,$$

and finally, since we have assumed $(m+4)2^{-t} \leq 0.1$, we note that

$$\left((1+(m+5)^2)^{1/2} + 1.1m\right) \leq (2.1m+6)(1-2\cdot 2^{-t}),$$

and we get the bound (1.33)

$$\|\eta\|_2 \leq (2.1m+6)2^{-t_1}\|y\|_2.$$

Equation (1.35) gives us

$$\|\delta\|_2 \leq \frac{2^{-t}}{1-2^{-t}}\left((\|y\|_2^2 - r^2)^{\frac{1}{2}} + |r| + \|\eta\|_2 + |\bar{r}-r|\right).$$

Using for a second time the maximum of the function f with $k=1$ for the first two terms and using equation (1.33) and equation (1.37) for the last two norms, we get

$$\|\delta\|_2 \leq \frac{2^{-t}}{1-2^{-t}}\left((2)^{1/2} + (4.1m+7)2^{-t_1}\right)\|y\|_2 ,$$

that eventually enables us to get the bound (1.34):

$$\|\delta\|_2 \leq 2.06\cdot 2^{-t}\|y\|_2 .$$

We also have

$$\|z\|_2 < (1 + 1.01(m+2)\cdot 2^{-t_1})\|y\|_2. \tag{1.41}$$

Algorithm 6 Modified projection

1. $\bar{a}^{(1)} = a$
2. for $k = 1, \ldots, n$
3. $\quad \bar{r}'_k = fl((\bar{q}'_k)^T \bar{a}^k)$
4. $\quad \bar{a}^{k+1} = fl(\bar{a}^k - \bar{q}'_k \bar{r}'_k)$
5. endfor
6. $\bar{w} = \bar{a}^{n+1}$

1.2.4 Error analysis of a modified projection

Let us assume that we have a set of vectors $\bar{Q}' = (\bar{q}'_1, \bar{q}'_2, \ldots, \bar{q}'_n)$ with the property that

$$\text{for all } i = 1, \ldots, n, \quad \text{there exists } \bar{w}_i \neq 0 \text{ such that } \quad fl(\bar{w}_i / \|\bar{w}_i\|_2) = \bar{q}'_i.$$

Given an initial floating–point vector a, let us consider the Algorithm 6.
Then, given the error analysis of Section 1.2.3, we know that, if we define for all $k = 1, \ldots, n$,

$$\bar{q}_k = \bar{q}'_k / \|\bar{q}'_k\|_2, \quad \bar{r}_k = \bar{r}'_k \|\bar{q}'_k\|_2 \quad \text{and} \quad r_k = \bar{q}_k^T \bar{a}^{k-1},$$

then

$$\bar{a}^{k+1} = \bar{a}^k - \bar{q}_k \bar{r} + \delta^{(k)} , \tag{1.42}$$

$$\text{and} \quad \bar{a}^{k+1} = \bar{a}^k - \bar{q}_k r + \eta^{(k)} , \tag{1.43}$$

with

$$\|\delta^{(k)}\|_2 \leq 1.45 \cdot 2^{-t} \|\bar{a}^k\|_2 , \tag{1.44}$$

$$\text{and} \quad \|\eta^{(k)}\|_2 \leq (2m+3) \cdot 2^{-t} \|\bar{a}^k\|_2 . \tag{1.45}$$

Summing equation (1.42) for $k = 1, 2, \ldots, n$ we get

$$\bar{w} = \bar{a} - \sum_{k=1}^{n} \bar{q}_k \bar{r}_k = \sum_{k=1}^{n} \delta^{(k)} = \delta.$$

From (1.44), we have

$$\|\delta\|_2 \leq 1.45 \cdot 2^{-t} \sum_{k=1}^{n} \|\bar{a}^k\|_2 .$$

Using equation (1.41), we get

$$\|\bar{a}^k\|_2 < (1 + 1.01(m+2) \cdot 2^{-t_1})^{(k-1)} \|a\|_2 .$$

We assume that

$$2n(m+1)2^{-t_1} < 0.01 ,$$

and so Björck got

$$\|\bar{a}^k\|_2 < 1.006 \|a\|_2.$$

Finally,

$$\bar{w} = (I_m - \bar{q}_n\bar{q}_n^T)\dots(I_m - \bar{q}_1\bar{q}_1^T)a + \eta \ ,$$
$$\|\eta\|_2 \leq 3.25(n-1)(\frac{2}{3}m+1)2^{-t}\|a\|_2 \ ,$$
$$\bar{w} = a - \bar{Q}r + \delta \ ,$$
$$\text{and} \|\delta\|_2 \leq 1.5(n-1)2^{-t}\|a\|_2 \ .$$

1.2.5 Error analysis of a classical projection

Daniel, Gragg, Kaufman and Stewart [34] defined

$$r = fl(Q^Ta), \quad v = fl(Qr) \quad \text{and} \quad w = fl(a-v),$$

where w represents the computed orthogonal projection of a onto the orthogonal complement of Span(Q). We are typically interested in the norm of the three vectors c, g, η defined by

$$r = Q^Ta + c \ , \quad w = a - Qr + g \ , \quad \text{and} \quad w = (Id - QQ^T)a + \eta.$$

Daniel, Gragg, Kaufman and Stewart [34] introduce the intermediate quantities

$$v = Qr + e \quad \text{and} \quad w = a - v + f.$$

In other words,

$$c = fl(Q^Ta) - Q^Ta, \quad e = fl(Qr) - Qr \quad \text{and} \quad f = fl(a-v) - (a-v).$$

From equation (1.20), we know that

$$\begin{aligned} \|c\|_2 \leq & \left[m + n^{1/2} - 1\right] 2^{-t}(1+2^{-t})^{n-1}\|Q\|_2\|a\|_2 \ , \\ \text{and} \quad \|e\|_2 \leq & \left[n + n^{1/2} - 1\right] 2^{-t}(1+2^{-t})^{n-1}\|Q\|_2\|r\|_2 \ . \end{aligned}$$

From the axpy error analysis,

$$\|f\|_2 \leq 2^{-t}(\|v\|_2 + \|a\|_2) \leq 2^{-t}(\|Q\|_2\|r\|_2 + \|e\|_2 + \|a\|_2) \ .$$

Therefore, since $g = f - e$, we have

$$\begin{aligned} \|g\|_2 &\leq \|e\|_2 + \|f\|_2 \\ &\leq (1+2^{-t})\|e\|_2 + 2^{-t}\|Q\|_2\|r\|_2 + 2^{-t}\|a\|_2 \\ &\leq \left[n + 2n^{1/2}\right] 2^{-t}(1+2^{-t})^n\|Q\|_2\|r\|_2 + 2^{-t}\|a\|_2 \ . \end{aligned}$$

Daniel, Gragg, Kaufman and Stewart [34] also noticed that $\|r\|_2 \leq \|Q^Ta\|_2 + \|c\|_2$ so that

$$\begin{aligned} \|g\|_2 \leq & \left[n + 2n^{1/2}\right] 2^{-t}(1+2^{-t})^n\|Q\|_2\|Q^Ta\|_2 + 2^{-t}\|a\|_2 \\ & + \left[mn + 2n^{1/2}(m+n) + 4n\right] (1+2^{-t})^{2n-1}2^{-2t}\|Q\|_2^2\|a\|_2. \end{aligned}$$

For a simple expression, we get

$$\|g\|_2 \leq \phi_1(m,n)u\|Q\|_2^2\|a\|_2 \ ,$$

with $\phi_1(m,n) = \left[n + n^{1/2} + \zeta_1\right]\zeta_2$ where $\zeta_1 = 1 + 4mnu(1+u)^n$ and $\zeta_2 = (1+u)^{n+1}$.

We also have $\eta = Qc + g$

$$\|\eta\|_2 \leq \|Q\|_2\|c\|_2 + \|g\|_2 .$$

For a simple expression we get

$$\|\eta\|_2 \leq \psi(m,n)u\|Q\|_2^2\|a\|_2,$$

with $\psi(m,n) = \left[m + n + 2n^{1/2} + \zeta_1 - 1\right]\zeta_2$.

1.3 New insight into the Gram–Schmidt algorithm

1.3.1 When the modified Gram–Schmidt algorithm generates a well–conditioned set of vectors

When the modified Gram–Schmidt algorithm is run on an ill–conditioned set of vectors, despite the observed loss of orthogonality among the constructed set of vectors, it has been observed that the constructed set of vectors is well–conditioned. In Section 1.4, that corresponds to [61], we shall give a theoretical explanation of this observation. When the modified Gram–Schmidt algorithm is run on a "not too ill–conditioned" set of vectors, then the condition number of the computed set of vectors is around one. The proof is a direct consequence of a result of Björck [13] combined with a result of Higham [73]. When the initial matrix A is not "too ill–conditioned", Björck [13] has shown that the loss of orthogonality of the computed matrix $\bar{Q}$ is less than one. Higham [73] has shown that, if the loss of orthogonality of $\bar{Q}$ is less than one, then the distance from $\bar{Q}$ to a matrix with orthonormal columns is also less than one. We [61] add the fact that, if the distance from $\bar{Q}$ to a matrix with orthonormal columns is less than one, then $\bar{Q}$ is well–conditioned. In our analysis, the meaning of "not too ill–conditioned" and well–conditioned is rigorously defined.

In Section 1.4.3.3, we exhibit a 3–by–3 matrix called *CERFACS* that is on the border of our set of "not too ill–conditioned" matrices and gives a really ill–conditionned computed Q–factor. In a sense, one can consider the matrices "not too ill–conditioned" as numerically nonsingular and the matrices "too ill–conditioned" as numerically singular. The border between the two sets is clear as illustrated by our bound and the properties of the *CERFACS* matrix.

We highlight the fact that the *CERFACS* matrix has been built on purpose. It has a condition number, κ, of the order of the inverse of the machine precision u, and the condition number of its computed Q–factor, $\kappa(\bar{Q})$, is also of the order of the inverse of the machine precision u. In Figure 1.1, we plot the coordinates $(\kappa, \kappa(\bar{Q}))$ for several matrices. The experiments have been performed with MATLAB5, with $u = 1.1 \cdot 10^{-16}$. The *CERFACS* matrix is the only one from this set with $u\kappa \sim 1$ and $u\kappa(\bar{Q}) \sim 1$. The *CERFACS* matrix shows that the theoretical bound exhibited for the condition number of the initial set of vectors for which the condition number of $\bar{Q}$ is less than 1.3 is sharp.

Finally, we mention that the behaviour of *CERFACS* reported in Figure 1.1 was observed using MATLAB5. Surprisingly enough, with MATLAB6, we no longer observe a large $\kappa(\bar{Q})$ when the running modified Gram–Schmidt algorithm on *CERFACS*. This supports the fact that the pathological counter–example matrix is highly dependent on the algorithmic. Note also that the 17 digits (in base 10) of the nine entries of *CERFACS* were mandatory to fix correctly the floating–point numbers in double–precision arithmetic.

We briefly explain here how to construct a *CERFACS*–like matrix A. Given A_{n-1}, an m–by–$(n-1)$ ill–conditioned matrix (such that all but the smallest singular values are of the same order, while $u\kappa \sim 0.1$), we generate the first $(n-1)$ columns of $\bar{Q}_{n-1}$ with modified Gram–Schmidt applied to A_{n-1}. The $(n-2)$ left singular

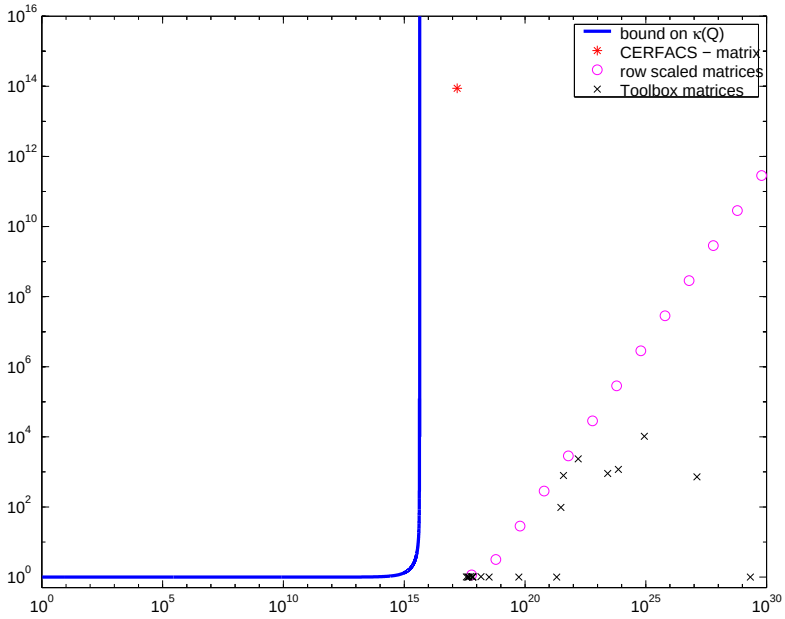


Figure 1.1: The x–axis represents the condition number of the initial matrix, the y–axis represents the condition number of the computed Q–factor using the modified Gram–Schmidt algorithm. The matrices from the Higham Toolbox are clement(94), dramadah(57,1), hilb(12), kahan(85), moler(24), pascal(16), prolate(23), and triw(47).

vectors of A, U_{n-2}, are computed. And we take u_1 as the left singular vector associated to the maximal singular value of

$$(I_m - U_{n-2}U_{n-2}^T)\bar{Q}_{n-1}.$$

Roughly speaking, u_1 represents the direction that is not in A_{n-1} and that "appears" in $\bar{Q}_{n-1}$. Finally, we set the n–th column of A to $a_n = u_1$.

1.3.2 The Gram–Schmidt algorithm with reorthogonalization at runtime

In most circumstances, the Gram–Schmidt algorithm is unable to provide a set of vectors orthogonal up to machine precision. In Section 1.5, we investigate the Gram–Schmidt algorithm with reorthogonalization. The goal of such algorithms is to provide a Q–factor orthogonal up to machine precision. Their main drawback is that they are more demanding in term of computational effort. Before going into the details of the algorithms, we first show some situtations where this extra computational expense is worthwhile to be afforded.
Let us call $\bar{Q}$ and $\bar{R}$ the computed QR–factor of A. In [15], Björck and Paige have shown that an m–by–n matrix $\hat{Q}$ exists such that $\hat{Q}$ has its columns orthonormal and $\hat{Q}\bar{R}$ represents A up to the machine precision. It means that the triangular factor computed by MGS is as good as that obtained using backward stable transformations such as Givens rotations or Householder reflections. This property of MGS explains why this algorithm can be safely used in applications where only the

triangular factors are needed. This is the case in the solution of linear least–squares problems of the form

$$\min_{x\in\mathbb{R}^n} \|b - Ax\|_2, \tag{1.46}$$

where $\|.\|_2$ denotes the spectral norm. In this case, only the R–factor of the QR factorization of $[A, b]$ is required [13, 15]. A general way to compensate for the lack of orthogonality of the Q–factor given by the modified Gram–Schmidt algorithm is derived in [15] for a wide class of problems. Such an approach is appealing since it enables us to use $\bar{Q}$ given by the modified Gram–Schmidt algorithm directly. However, the cost of using $\bar{Q}$ correctly is – in most cases – much higher than if Q was orthogonal up to machine precision. For example, when solving equation (1.46) via a QR–factorization of $[A, b]$, the cost for computing $(r_{i,n+1})_{i=1,\ldots,n}$, the updates of b during the calculation of $\bar{Q}$, is about twice as expensive as the simple calculation $(q_i^T b)_{i=1,\ldots,n}$. This latter approach could be used if the QR–factorization on A provided an orthogonal factor with orthogonality at the level of machine precision. When we want to use the Q–factor several times (e.g. a linear least–squares problem with multiple right–hand sides), these extra costs are accumulated and a method like the one proposed in [15] becomes less attractive than a strategy that aims at enhancing the orthogonality of the columns of the Q–factor at a moderate computational cost.
In the same spirit, the classical Gram–Schmidt algorithm enables the use of Level 3 BLAS operations whereas the modified Gram–Schmidt (row oriented version) uses Level 2 BLAS. There are situtations and computing platforms where, even if the classical Gram–Schmidt algorithm with reorthogonalizations performs twice as many operations as the modified Gram–Schmidt algorithm, the first algorithm is faster. Such situations are described for instance in the GMRES context [52], in a GCR context [49] or in an eigensolver context [88].
The variants of the Gram–Schmidt algorithm with reorthogonalization are often monitored through two parameters:

(a) the maximum number of reorthogonalization iterations performed for a given vector,

(b) the criterion used for deciding when the reorthogonalization iterations have to be stopped.

Our contribution in this context was first conducted in a joint ongoing work with Miroslav Rozložník [62]. We have shown that, for numerically nonsingular matrices (in a sense that is clearly defined), two loops of the Gram–Schmidt algorithm with reorthogonalization are enough to obtain a Q–factor orthogonal up to machine precision. We clearly relate the number of loops with the condition number which made this approach different (and complementary) from those of Abdelmaleck [2] or Daniel, Gragg, Kaufman and Stewart [34]. The main arguments of [62] are developed in Section 1.5.1.7 in the context of modified Gram–Schmidt with reorthogonalization. Denoting by $a_j^{(1)}$ the vector obtained after the projection of a_j onto Q_{j-1}, we bound the quantity $\|a_j\|/\|a_j^{(1)}\|$ by the condition number of A times a constant close to one.

Daniel, Gragg, Kaufman and Stewart [34] studied an infinite loop approach based on classical projections. Starting from Q_j an m–by–j matrix and the vector $a_{j+1}^{(0)} = a_{j+1}$, they performed for $\ell = 1, \ldots,$

$$a_{j+1}^{(\ell)} \longleftarrow (I - Q_j Q_j^T) a_{j+1}^{(\ell-1)}.$$

At each instance ℓ, they bound the quantity $\|Q_j^T a_{j+1}^{(\ell)}\|_2 / \|a_{j+1}^{(\ell)}\|_2$ via a real ζ_ℓ given by a reccurence formula $\zeta_\ell = \varphi(\zeta_{\ell-1})$. The function φ and the initial ζ_0 depend on Q_j and $a_{j+1}^{(0)}$. The assumption on Q_j is rather weak in term of orthogonality; roughly speaking it is $\|I - Q_j^T Q_j\|_2 < 1$. The formulae are given explicitly in [34]. In Figure 1.2, we plot the function φ for a given m–by–n matrix Q and a given machine precision (here $u = 1.1 \cdot 10^{-16}$); we also plot the iterates ζ_ℓ's that correspond to the bound on $\|Q^T\|_2 / \|a^{(\ell)}\|_2$ where $a^{(0)}$ is a random vector. This 40–by–30 matrix is such that $\|I_n - Q^T Q\|_2 = 10^{-3}$. At each step, the bound ζ_ℓ represents the effective orthogonality well. It can be observed that, even though $\|I_n - Q^T Q\|_2 = 10^{-3}$, the final orthogonality $\|Q_n^T a^{(\ell)}\|_2 / \|a^{(\ell)}\|_2$ is at the level of machine precision. The maximum level of accuracy given by Daniel, Gragg, Kaufman and Stewart [34] corresponds to $\zeta^\star$; it corresponds to the smallest x where the curves $y = \varphi(x)$ and $y = x$ intersect. In 5 steps, ζ_5 is close to $\zeta^\star$. We note that the final level obtained in finite precision is much lower than $\zeta^\star$.

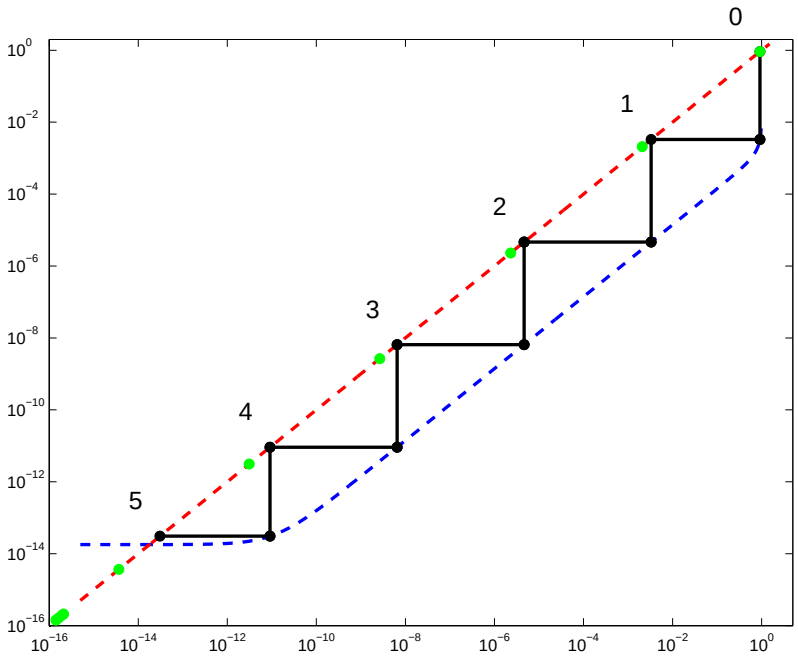


Figure 1.2: The function $y = x$ is plotted in red, the function $y = \varphi(x)$ is plotted in blue. Starting from ζ_0, we plot the iterate $\zeta_\ell = \varphi(\zeta_{\ell-1})$ with $\circ$. We verify that the ζ_ℓ's are bounds for $\|Q^T v^{(\ell)}\|_2 / \|v^{(\ell)}\|_2$ (green points). In this experiment, we have $\|I_n - Q^T Q\|_2 = 10^{-3}$.

This result can be related to the work of Ruhe [112]. Ruhe stated that Classical Gram–Schmidt Iterated algorithm corresponds to a Gauss–Jacobi iteration to solve

the linear system :

$$(Q_j^T Q_j) r = Q_j^T a_{j+1}. \tag{1.47}$$

The initial guess should be set to

$$r^{(0)} = 0. \tag{1.48}$$

We call r the exact solution of (1.47) and assume that $Q^T Q$ is nonsingular, then we can write

$$r = (Q_j^T Q_j)^{-1} Q_j^T a. \tag{1.49}$$

The splitting of the matrix $Q^T Q$ is

$$M = I, \qquad N = I - Q_j^T Q_j. \tag{1.50}$$

The Gauss–Jacobi iteration converges to the solution if $\|N\|_2 \leq 1$. We obtain that an infinite loop of projection gives an orthogonal vector, if $\|I - Q_j^T Q_j\|_2 < 1$. Note that Ruhe [112] also stated that the modified Gram-Schmidt iterated algorithm corresponds to a Gauss–Seidel iteration for solving the linear system (1.47).
In Section 1.5, we give a new reorthogonalization criterion for the modified Gram–Schmidt algorithm and investigate the numerical behaviour of the Gram–Schmidt algorithm with several reorthogonalization criteria.

1.3.3 A posteriori reorthogonalization in the Gram–Schmidt algorithm

Another idea to obtain a Q factor orthogonal to machine precision is to treat the Q factor not at runtime but a posteriori. This implies a good knowledge of the algorithm, in the sense that we voluntarily let the algorithm go wrong and correct it afterwards. This is fundammentally different from the philosophy of the reorthogonalization at runtime that does its best to enforce orthogonality at each step. A posteriori reorthogonalization technique have not been given much attention in the past and, in our bibliographical search, we have only found a posteriori reorthogonalization suggested in Mitchell and McCraith [91].
From Section 1.4.3.2, we obtain a straightforward but robust a posteriori procedure that consists, after a first run of modified Gram–Schmidt, in performing a second sweep systematically.
In Section 1.6, we give a reorthogonalization procedure for the modified Gram–Schmidt algorithm. In particular, we illustrate the efficiency of this new approach in the framework of the solution of linear systems arising in the application described in Chapter 3. This reorthogonalization algorithm is based on the new theoretical properties that are described in the first part of Section 1.6.

1.4 When the modified Gram–Schmidt algorithm generates a well–conditioned set of vectors

The title as well as the contents of this section corresponds to the following published paper:

IMA Journal on Numerical Analysis, 22(4):521–528, 2002.
joint work with Luc Giraud.

Abstract

In this paper, we show why the modified Gram–Schmidt algorithm generates a well-conditioned set of vectors. This result holds under the assumption that the initial matrix is not "too ill–conditioned" in a way that is quantified. As a consequence, we show that if two iterations of the algorithm are performed, the resulting algorithm produces a matrix whose columns are orthogonal up to machine precision. Finally, we illustrate through a numerical experiment the sharpness of our result.

Introduction

In this paper we study the condition number of the set of vectors generated by the Modified Gram–Schmidt (MGS) algorithm in floating–point arithmetic. After a quick review, in Section 1.4.1, of the fundamental results that we use, we devote Section 1.4.2 to our main theorem. Through this central theorem we give an upper bound close to one for the condition number of the set of vectors produced by MGS. This theorem applies to matrices that are not "too ill–conditioned". In Section 1.4.3 we give another way to prove a similar result. This other point of view throws light on the key points of the proof. In Section 4.2 we combine our theorem with a well known result from Björck to obtain that two iterations of MGS are indeed enough to get a matrix whose columns are orthogonal up to machine precision. We conclude Section 1.4.3 by exhibiting a counter example matrix. This matrix shows that if we relax the constraint on the condition number of the studied matrices, no pertinent information on the upper bound of the condition number of the set of vectors generated by MGS can be gained. For the sake of completeness, we give explicitly the constants that appear in our assumptions and formula: Appendix A details the calculus of those constants.

1.4.1 Previous results and notations

We consider the MGS algorithm applied to a matrix $A \in \mathbb{R}^{m\times n}$ with full rank $n \leq m$ and singular values: $\sigma_1 \geq \ldots \geq \sigma_n > 0$; we define the condition number of A as $\kappa(A) = \sigma_1/\sigma_n$.
Using results from Björck [13] and Björck and Paige [15], we know that, in floating–point arithmetic, MGS computes $\bar{Q} \in \mathbb{R}^{m\times n}$ and $\bar{R} \in \mathbb{R}^{n\times n}$ so that there exists $\bar{E} \in \mathbb{R}^{m\times n}$, $\hat{E} \in \mathbb{R}^{m\times n}$ and $\hat{Q} \in \mathbb{R}^{m\times n}$, where

$$A + \bar{E} = \bar{Q}\bar{R} \quad \text{and} \quad \|\bar{E}\|_2 \leq \bar{c}_1 u \|A\|_2, \tag{1.51}$$

$$\|I - \bar{Q}^T\bar{Q}\|_2 \leq \bar{c}_2 \kappa(A) u, \tag{1.52}$$

$$A + \hat{E} = \hat{Q}\bar{R} \quad , \quad \hat{Q}^T\hat{Q} = I \quad \text{and} \quad \|\hat{E}\|_2 \leq cu\|A\|_2. \tag{1.53}$$

$\bar{c}_i$ and c are constants depending on m, n and the details of the arithmetic, and $u = 2^{-t}$ is the unit round-off.
Result (1.51) shows that $\bar{Q}\bar{R}$ is a backward-stable factorization of A, that is the product $\bar{Q}\bar{R}$ represents accurately A up to machine precision.
Equation (1.53) says that $\bar{R}$ solves the QR-factorization problem in a backward-stable sense; that is, there exists an exact orthonormal matrix $\hat{Q}$ so that $\hat{Q}\bar{R}$ is a QR factorization of a slight perturbation of A.
We notice that results (1.51) from Björck [13] and (1.53) from Björck and Paige [15] are proved under assumptions

$$2.12 \cdot (m+1)u < 0.01, \tag{1.54}$$

$$cu\kappa(A) < 1. \tag{1.55}$$

For clarity, it is important to explicitly define the constants that are involved in the upper bounds of the inequalities. Complying with assumptions (1.54) and (1.55) we can set the constants c and $\bar{c}_1$ to

$$c = 18.53 \cdot n^{\frac{3}{2}} \quad \text{and} \quad \bar{c}_1 = 1.853 \cdot n^{\frac{3}{2}} = 0.1 \cdot c. \tag{1.56}$$

The value of $\bar{c}_1$ is given explicitly by Björck [13]. The details on the calculus of the constant c are given in Appendix A. It is worth noticing that the value of c depends only on n, the number of vectors to be orthogonalized, and not on m, the size of the vectors, since (1.54) holds.
Assumption (1.55) prevents $\bar{R}$ from being singular. Under this assumption and defining

$$\eta = \frac{1}{1 - cu\kappa(A)}, \tag{1.57}$$

Björck and Paige [15] obtain an upper bound for $\|\bar{R}^{-1}\|_2$ as

$$\|A\|_2 \|\bar{R}^{-1}\|_2 \leq \eta\kappa(A). \tag{1.58}$$

Assuming (1.55), we note that (1.51) and (1.53) are independent of $\kappa(A)$. This is not the case for inequality (1.52): the level of orthogonality in $\bar{Q}$ is dependent on $\kappa(A)$. If A is well-conditioned then $\bar{Q}$ is orthogonal to machine precision. But for an ill–conditioned matrix A, the set of vectors $\bar{Q}$ may lose orthogonality. An important question that arises then is whether MGS manages to preserve the full rank of $\bar{Q}$ or not. In order to investigate this, we study in the next section the condition number of $\bar{Q}$. For this purpose, we denote the singular values of $\bar{Q}$, $\sigma_1(\bar{Q}) \geq \ldots \geq \sigma_n(\bar{Q})$. When $\bar{Q}$ is nonsingular, $\sigma_n(\bar{Q}) > 0$, we also denote the condition number $\kappa(\bar{Q}) = \sigma_1(\bar{Q})/\sigma_n(\bar{Q})$.

1.4.2 Conditioning of the set of vectors $\bar{Q}$

This section is fully devoted to the key theorem of this paper and to its proof. For the sake of completeness, we establish a similar result using different arguments in the next section. The central theorem is the following.

Theorem 1.4.1

Let $A \in \mathbb{R}^{m\times n}$ be a matrix with full rank $n \leq m$ and condition number $\kappa(A)$ such that

$$2.12 \cdot (m+1)u < 0.01 \quad \textit{and} \quad cu\kappa(A) \leq 0.1, \tag{1.59}$$

where $c = 18.53 \cdot n^{\frac{3}{2}}$ and u is the unit round-off.
Then MGS in floating–point arithmetic computes $\bar{Q} \in \mathbb{R}^{m\times n}$ as

$$\boxed{\kappa(\bar{Q}) \leq 1.3.} \tag{1.60}$$

Note that assumption (1.59) is just slightly stronger than assumption (1.55) made by Björck and Paige [15].
Proof : On the one hand, MGS computes $\bar{Q}$, on the other hand, the matrix $\hat{Q}$ has exactly orthonormal columns. It seems natural to study the distance between $\bar{Q}$ and $\hat{Q}$. For that we define F as

$$F = \bar{Q} - \hat{Q}, \tag{1.61}$$

and look at its 2-norm. For this purpose, we subtract (1.53) from (1.51) to get

$$\begin{aligned} (\bar{Q} - \hat{Q})\bar{R} &= A + \bar{E} - A - \hat{E}, \\ F\bar{R} &= \bar{E} - \hat{E}. \end{aligned}$$

Assuming $cu\kappa(A) < 1$, $\bar{R}$ is nonsingular and we can write

$$F = (\bar{E} - \hat{E})\bar{R}^{-1}.$$

We bound, in terms of norms, this equality

$$\|F\|_2 \leq (\|\bar{E}\|_2 + \|\hat{E}\|_2)\|\bar{R}^{-1}\|_2.$$

Using inequality (1.51) on $\|\bar{E}\|_2$ and inequality (1.53) on $\|\hat{E}\|_2$, we obtain

$$\|F\|_2 \leq (c + \bar{c}_1)u\|A\|_2\|\bar{R}^{-1}\|_2.$$

Using inequality (1.58) on $\|A\|_2\|\bar{R}^{-1}\|_2$ and (1.56), we have

$$\boxed{\|F\|_2 \leq 1.1 \cdot cu\eta\kappa(A).} \tag{1.62}$$

This is the desired bound on $\|F\|_2$.
Since we are interested in an upper bound on $\kappa(\bar{Q})$, the condition number of $\bar{Q}$, we then look for an upper bound for the largest singular value of $\bar{Q}$ and a lower bound for its smallest singular value.
From Golub and van Loan [63, p. 449], we know that (1.61) implies

$$\sigma_1(\bar{Q}) \leq \sigma_1(\hat{Q}) + \|F\|_2 \quad \text{and} \quad \sigma_n(\bar{Q}) \geq \sigma_n(\hat{Q}) - \|F\|_2.$$

Since $\hat{Q}$ has exactly orthonormal columns, we have $\sigma_1(\hat{Q}) = \sigma_n(\hat{Q}) = 1$. Using the bound (1.62) on $\|F\|_2$, we get

$$\sigma_1(\bar{Q}) \leq 1 + 1.1 \cdot cu\eta\kappa(A) \quad \text{and} \quad \sigma_n(\bar{Q}) \geq 1 - 1.1 \cdot cu\eta\kappa(A).$$

With (1.57), these inequalities can be written as

$$\sigma_1(\bar{Q}) \leq \eta(1 - cu\kappa(A) + 1.1 \cdot cu\kappa(A)) = \eta(1 + 0.1 \cdot cu\kappa(A))$$

and

$$\sigma_n(\bar{Q}) \geq \eta(1 - cu\kappa(A) - 1.1 \cdot cu\kappa(A)) = \eta(1 - 2.1 \cdot cu\kappa(A)).$$

If we assume

$$2.1 \cdot cu\kappa(A) < 1, \tag{1.63}$$

$\sigma_n(\bar{Q}) > 0$ so $\bar{Q}$ is nonsingular.
Under this assumption, we have

$$\boxed{\kappa(\bar{Q}) \leq \tfrac{1+0.1\cdot cu\kappa(A)}{1-2.1\cdot cu\kappa(A)}.} \tag{1.64}$$

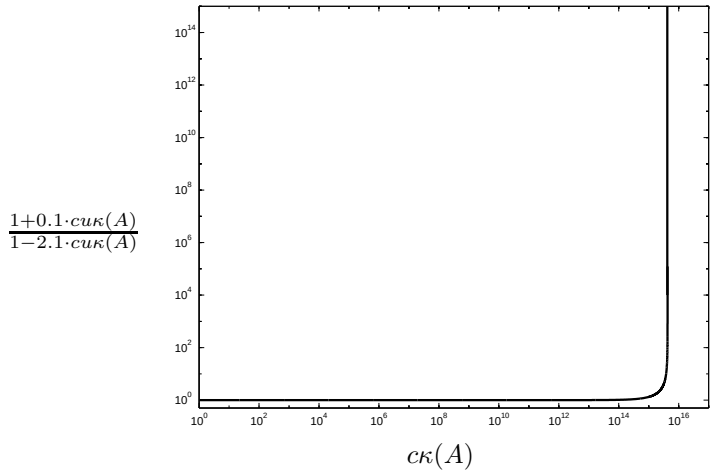


Figure 1.3: Behaviour of the upper bound on $\kappa(\bar{Q})$ as a function of $c\kappa(A)$.

To illustrate the behaviour of the upper bound of $\kappa(\bar{Q})$, we plot in Figure 1.3 the upper bound as a function of $c\kappa(A)$. We fix $u = 1.12 \cdot 10^{-16}$.
It can be seen that this upper bound explodes when $2.1 \cdot cu\kappa(A) \lesssim 1$ but in the main part of the domain where $2.1 \cdot cu\kappa(A) < 1$ it is small and very close to one. For instance, if we slightly increase the constraint (1.55) used by Björck and Paige [15] and assume that $cu\kappa(A) < 0.1$ then $\kappa(\bar{Q}) < 1.3$. ■

1.4.3 Some remarks

1.4.3.1 Another way to establish a result similar to Theorem 1.4.1

It is also possible to get a bound on $\kappa(\bar{Q})$ by using inequality (1.52). In this aim, we need explicitly the constant $\bar{c}_2$ given by Björck and Paige [15]. Using assumptions (1.54) and (1.59), $\bar{c}_2$ can be set to

$$\bar{c}_2 = 31.6863 \cdot n^{\frac{3}{2}} = 1.71 \cdot c. \tag{1.65}$$

The details on the calculus of the constant $\bar{c}_2$ are given in Appendix A.
Let $\bar{Q}$ have the polar decomposition $\bar{Q} = UH$. The matrix U is the closest orthonormal matrix to $\bar{Q}$ in any unitarily invariant norm. We define

$$G = \bar{Q} - U.$$

From Higham [75], we know that in 2-norm the distance from $\bar{Q}$ to U is bounded by $\|I - \bar{Q}^T\bar{Q}\|_2$. This means

$$\|G\|_2 = \|\bar{Q} - U\|_2 \leq \|I - \bar{Q}^T\bar{Q}\|_2$$

and using (1.52) we get

$$\|G\|_2 \leq \bar{c}_2 u\kappa(A) = 1.71 \cdot cu\kappa(A). \tag{1.66}$$

Using the same arguments as in Section 1.4.2 for the proof of Theorem 1.4.1, but replacing (1.62) with (1.66), we get a similar result: that is

$$\text{assuming (1.54) and (1.59)}, \quad \kappa(\bar{Q}) < 1.42.$$

This result should be compared with that of Theorem 1.4.1. With the same assumptions, we obtain a slightly weaker result.

1.4.3.2 Iterative modified Gram–Schmidt

If the assumption (1.59) on the condition number of A holds, then we obtain, after a first sweep of MGS, $\bar{Q}_1$ satisfying (1.64). If we run MGS a second time on $\bar{Q}_1$ to obtain $\bar{Q}_2$, we deduce using (1.52) that $\bar{Q}_2$ is such that

$$\|I - \bar{Q}_2^T\bar{Q}_2\|_2 \leq 1.71 \cdot c\kappa(\bar{Q}_1)u,$$

so we get

$$\boxed{\|I - \bar{Q}_2^T\bar{Q}_2\|_2 < 40.52 \cdot un^{\frac{3}{2}},} \tag{1.67}$$

meaning that $\bar{Q}_2$ has columns orthonormal to machine precision. Two MGS sweeps are indeed enough to have an orthonormal set of vectors Q.
We recover, in a slightly different framework, the famous sentence of Kahan

"Twice is enough."

Based on unpublished notes of Kahan, Parlett [101] shows that an iterative Gram–Schmidt process on two vectors with a selective criterion (optional) produces two vectors orthonormal up to machine precision. In this paper, inequality (1.67) show that *twice is enough* for n vectors under assumptions (1.54) and (1.59) with MGS and a complete *a posteriori* re-orthogonalization (i.e. no selective criterion).

1.4.3.3 What can be said on $\kappa(\bar{Q})$ when $cu\kappa(A) > 0.1$

For $2.1 \cdot cu\kappa(A) < 1$, the bound (1.64) on $\kappa(\bar{Q})$ is well defined but when $cu\kappa(A) > 0.1$, this bound explodes and very quickly nothing interesting can be said about the condition number of $\bar{Q}$. For $2.1 \cdot cu\kappa(A) > 1$, we even do not have any bound.

Here, we ask whether or not there can exist an interesting upper bound on $\bar{Q}$ when $cu\kappa(A) > 0.1$. In order to answer this problem, we consider the *CERFACS* matrix $\in \mathbb{R}^{3\times 3}$ (see Appendix B).
When we run MGS with Matlab on *CERFACS*, we obtain with $u = 1.12 \cdot 10^{-16}$

$$\kappa(A) = 3 \cdot 10^{15}, \quad cu\kappa(A) = 37 \quad \text{and} \quad \kappa(\bar{Q}) = 2 \cdot 10^{14} \ .$$

The *CERFACS* matrix generates a very ill–conditioned set of vectors $\bar{Q}$ with $cu\kappa(A)$ not too far from 0.1.
If we are looking for an upper bound of $\kappa(\bar{Q})$, we can take the value 1.3 up to $cu\kappa(A) = 0.1$ and then this upper bound has to be greater than $2 \cdot 10^{14}$ for $cu\kappa(A) = 37$.
The *CERFACS* matrix proves that it is not possible to increase by much the domain of validity (i.e. $cu\kappa(A) < 0.1$) of Theorem (1.4.1) in order to get a more interesting result.
One can also remark that with *CERFACS* two MGS sweeps are no longer enough since

$$\|I - \bar{Q}_2^T \bar{Q}_2\|_2 = 2 \cdot 10^{-3}.$$

Acknowledegment
We would like to thank Miroslav Rozložník for fruitful discussions on the Modified Gram–Schmidt algorithm and in particular for having highlighted that the sentence *twice is enough* required the assumption of a not "too ill–conditioned" matrix A. We also thank the anonymous referees for their comments that helped to improve the paper.

Appendix A: Details on the calculus of the constants

In this Appendix, we justify the values of the constants $\bar{c}_1$, $\bar{c}_2$ and c such as fixed in the paper. We state that
$\bar{c}_1 = 1.853 \cdot n^{\frac{3}{2}}$ verifies (1.51) under assumption (1.54),
$\bar{c}_2 = 31.6863 \cdot n^{\frac{3}{2}}$ verifies (1.52) under assumptions (1.54) and (1.59),
$c = 18.53 \cdot n^{\frac{3}{2}}$ verifies (1.53) under assumptions (1.54) and (1.55).

A value for $\bar{c}_1$ Under the assumption (1.54) Björck [13] has shown that

$$A + \bar{E} = \bar{Q}\bar{R} \quad \text{with} \quad \|\bar{E}\|_E \leq 1.5 \cdot (n-1)u\|A\|_E.$$

where $\|.\|_E$ denotes the Frobenius norm.
$\bar{c}_1 = 1.853 \cdot n^{\frac{3}{2}}$ verifies $\|\bar{E}\|_E \leq \bar{c}_1 u \|A\|_2$.

A value for c Björck and Paige [15] explained that the sequence of operations to obtain the R-factor with the MGS algorithm applied on A is exactly the same as the sequence of operations to obtain the R-factor with the Householder process applied on the augmented matrix $\begin{pmatrix} 0_n \\ A \end{pmatrix} \in \mathbb{R}^{(m+n)\times n}$. They deduced that the R-factor from the Householder process applied on the augmented matrix is equal

to $\bar{R}$. We first present the results from Wilkinson [137] related to the Householder process on the matrix $\begin{pmatrix} 0_n \\ A \end{pmatrix} \in \mathbb{R}^{(m+n)\times n}$. Wilkinson [137] works with a square matrix but in the case of a rectangular matrix, proofs and results remain the same. All the results of Wilkinson hold under the assumption $(m+n)\cdot u < 0.1$ which is true because of (1.54).
Defining $x = 12.36 \cdot u$, Wilkinson proves that there exists $P \in \mathbb{R}^{(m+n)\times n}$ with orthonormal columns such that

$$\|P\bar{R} - A\|_E \leq (n-1)(1+x)^{n-2} x \|A\|_E. \tag{1.68}$$

With assumption (1.54), we get $(1+x)^{n-2} \leq 1.060053$.
Let us define $E_1 \in \mathbb{R}^{n\times n}$ and $E_2 \in \mathbb{R}^{m\times n}$ by

$$\begin{pmatrix} E_1 \\ E_2 \end{pmatrix} = P\bar{R} - \begin{pmatrix} 0_n \\ A \end{pmatrix}.$$

We deduce with (1.68) that

$$\| \begin{pmatrix} E_1 \\ E_2 \end{pmatrix} \|_E \leq 13.1023 \cdot n^{\frac{3}{2}} u \|A\|_2. \tag{1.69}$$

If we set

$$c_1 = c_2 = 13.1023 \cdot n^{\frac{3}{2}}, \tag{1.70}$$

then we get $\|E_1\|_2 \leq c_1 u \|A\|_2$ and $\|E_2\|_2 \leq c_2 u \|A\|_2$.
Note that we also have

$$\|E_1\|_2 + \|E_2\|_2 \leq \sqrt{2} \| \begin{pmatrix} E_1 \\ E_2 \end{pmatrix} \|_E \leq \sqrt{2} c_1 u \|A\|_2. \tag{1.71}$$

With respect to MGS, Björck and Paige [15] have proved that there exists $\hat{E} \in \mathbb{R}^{m\times n}$ and $\hat{Q} \in \mathbb{R}^{m\times n}$ such that

$$A + \hat{E} = \hat{Q}\bar{R} \quad , \quad \hat{Q}^T\hat{Q} = I \quad \text{and} \quad \|\hat{E}\|_2 \leq \|E_1\|_2 + \|E_2\|_2.$$

With (1.71) we get

$$\|\hat{E}\|_2 \leq \|E_1\|_2 + \|E_2\|_2 \leq \sqrt{2} c_1 u \|A\|_2 \leq 18.53 \cdot n^{\frac{3}{2}},$$

and $c = 18.53 \cdot n^{\frac{3}{2}}$ verifies $\|\hat{E}\|_2 \leq cu\|A\|_2$.

A value for $\bar{c}_2$ Björck [13] defines a value for $\bar{c}_2$. In this paper, we do not consider this value because the assumptions on n and $\kappa(A)$ that we obtain are too restricted. The value of $\bar{c}_2$ from Björck and Paige [15] requires weaker assumptions that fit the context of this paper. From (1.59), we have $(c+c_1)u\kappa < 1$. Under this assumption, Björck and Paige [15] have proved that

$$\|I - \bar{Q}^T\bar{Q}\|_2 \leq \frac{2c_1}{1-(c+c_1)u\kappa} \kappa u. \tag{1.72}$$

With $\bar{c}_2 = 31.6863 \cdot n^{\frac{3}{2}}$ and using assumption (1.59), we have $\|I - \bar{Q}^T\bar{Q}\|_2 \leq \bar{c}_2 \kappa u$.

Appendix B: matrix *CERFACS*

We have developed a Matlab code that generates as many as desired matrices with relatively small $cu\kappa(A)$ and large $\kappa(\bar{Q})$. *CERFACS* is one of these:

$$CERFACS = \begin{pmatrix} 0.12100300219993308 & 2.09408775152625060 & 1.26139640819301024 \\ -0.10439395064078592 & -1.80665016070527140 & -1.08825526624380808 \\ 0.21661355806776747 & 0.49451660567698374 & -0.84174336538575500 \end{pmatrix}.$$

1.5 A robust criterion for modified Gram–Schmidt with selective reorthogonalization

The title as well as the contents of this section corresponds to the following paper accepted for publication in:

SIAM Journal on Scientific Computing, 2003.
joint work with Luc Giraud.

Abstract

A new criterion for selective reorthogonalization in the modified Gram–Schmidt algorithm is proposed. We study its behaviour in the presence of rounding errors. We give some counter–example matrices which prove that the standard criteria might fail. Through numerical experiments, we illustrate that our new criterion seems to be suitable also for the classical Gram–Schmidt algorithm with selective reorthogonalization.

AMS Subject Classification : 65F25, 65G50, 15A23.

Introduction

Let $A = (a_1, \ldots, a_n)$ be a real $m \times n$ matrix ($m > n$) whose columns are linearly independent. In many applications, it is required to have an orthonormal basis for the space spanned by the columns of A. This amounts to knowing a matrix $Q \in \mathbb{R}^{m \times n}$ with orthonormal columns such that $A = QR$, $R \in \mathbb{R}^{n \times n}$. Moreover, it is possible to require R to be triangular, we then end up with the so called QR–factorization. For all j, the first j columns of Q are an orthonormal basis for the space spanned by the first j columns of A.
Starting from A, there are many algorithms that build such a factorization. We focus, in this paper, on the Gram–Schmidt algorithm [120] that consists in projecting successively the columns of A on the space orthogonal to the space spanned by the already constructed columns of Q. Depending on how the projections are performed, there are two main versions of this algorithm [109]: the classical Gram–Schmidt algorithm (CGS) and the modified Gram–Schmidt algorithm (MGS). In exact arithmetic, both algorithms produce exactly the same results and the resulting matrix Q has orthonormal columns. In the presence of round–off errors, Q computed by CGS differs from that computed by MGS. In both cases, the columns of Q may be far from orthogonal. To remedy this problem, a solution is to iterate the procedure and to project each column of A several times instead of only once on the space orthogonal to the space spanned by the constructed columns of Q. Giraud, Langou and Rozložník [62] have shown that, when using floating–point arithmetic, either for CGS or MGS, two iterations were enough when the initial matrix A is numerically nonsingular. This confirms what was already experimentally well known and generalizes the result of Kahan and Parlett [101] for $n = 2$ vectors. In this paper, we focus mainly on the Gram–Schmidt algorithms where the number of projections for each column of A is either one or two. When the number of reorthogonalizations performed is exactly 2, we call the resulting algorithm the classical (resp. modified) Gram–Schmidt algorithm with reorthogonalization and

denote it by CGS2, (resp. MGS2); the MGS2 Algorithm is given in Algorithm 1. The use of either CGS2 or MGS2 guarantees a reliable result in term of orthogonality [62] but then the computational cost is twice as much as for CGS or MGS. In many applications, we observe that either CGS or MGS is good enough, the additional reorthogonalizations performed in CGS2 or MGS2 are then useless. A good compromise in term of orthogonality quality and time is to use a selective reorthogonalization criterion to check for each columns of A whether an extra reorthogonalization is needed or not. Historically, Rutishauser [113] introduced the first criterion in a Gram–Schmidt algorithm with reorthogonalization. We refer to it as the K–criterion. It is dependent on a single parameter $K \geq 1$. The resulting algorithms are called the classical or modified Gram–Schmidt algorithm with selective reorthogonalization and K–criterion; they are denoted by CGS2(K) and MGS2(K) respectively. We give below the modified Gram–Schmidt algorithm with selective reorthogonalization based on the K–criterion (MGS2(K)).

Algorithm 1 MGS2	**Algorithm 2** MGS2(K)
for $j = 1$ to n **do**	**for** $j = 1$ to n **do**
$a_j^{(1)(1)} = a_j$	$a_j^{(1)(1)} = a_j$
for $k = 1$ to $j-1$ **do**	**for** $k = 1$ to $j-1$ **do**
$r_{kj}^{(1)} = q_k^T a_j^{(k)(1)}$	$r_{kj}^{(1)} = q_k^T a_j^{(k)(1)}$
$a_j^{(k+1)(1)} = a_j^{(k)(1)} - q_k r_{kj}^{(1)}$	$a_j^{(k+1)(1)} = a_j^{(k)(1)} - q_k r_{kj}^{(1)}$
end for	**end for**
	if $\left(\frac{\|a_j\|_2}{\|a_j^{(j)(1)}\|_2} \leq K\right)$ **then**
	$r_{jj} = \|a_j^{(j)(1)}\|_2$
	$q_j = a_j^{(j)(1)}/r_{jj}$
	$r_{kj} = r_{kj}^{(1)}, 1 \leq k \leq j-1$
	else
$a_j^{(1)(2)} = a_j^{(j)(1)}$	$a_j^{(1)(2)} = a_j^{(j)(1)}$
for $k = 1$ to $j-1$ **do**	**for** $k = 1$ to $j-1$ **do**
$r_{kj}^{(2)} = q_k^T a_j^{(k)(2)}$	$r_{kj}^{(2)} = q_k^T a_j^{(k)(2)}$
$a_j^{(k+1)(2)} = a_j^{(k)(2)} - q_k r_{kj}^{(2)}$	$a_j^{(k+1)(2)} = a_j^{(k)(2)} - q_k r_{kj}^{(2)}$
end for	**end for**
$r_{jj} = \|a_j^{(j)(2)}\|_2$	$r_{jj} = \|a_j^{(j)(2)}\|_2$
$q_j = a_j^{(j)(2)}/r_{jj}$	$q_j = a_j^{(j)(2)}/r_{jj}$
$r_{kj} = r_{kj}^{(1)} + r_{kj}^{(2)}, 1 \leq k \leq j-1$	$r_{kj} = r_{kj}^{(1)} + r_{kj}^{(2)}, 1 \leq k \leq j-1$
	end if
end for	**end for**

Using floating–point arithmetic, Kahan and Parlett [101] have shown that for two vectors the orthogonality obtained (measured by $|q_1^T q_2|$) is bounded by a constant times $K\varepsilon$ where ε denotes the machine precision. This gives a way of computing K in order to ensure a satisfactory level of orthogonality. For n vectors, the choice of the parameter K is not so clear. Giraud and al. [62] show that if K is

greater than the condition number of A, $\kappa(A)$, then neither CGS2($K = \kappa(A)$) nor MGS2($K = \kappa(A)$) performs any reorthogonalization. Interesting values for K therefore range from 1 (this corresponds to CGS2 or MGS2) to $\kappa(A)$ (this corresponds to CGS or MGS). If K is high then we have few reorthogonalizations, so we could expect a lower level of orthogonality than if K is smaller where more reorthogonalizations are performed. In order to reach orthogonality at the machine precision level, Rutishauser [113] in 1967 chose the value $K = 10$. We find an explanation of this value in Gander [59, p. 12]: *"in particular one may state the rule of thumb that at least one decimal digit is lost by cancellation if* $10\|a_j^{(1)}\|_2 \leq \|a_j\|_2$. *This equation is the criterion used by Rutishauser to decide whether reorthogonalization is necessary."* The value $K = \sqrt{2}$ is also very often used since the publication of the paper of Daniel, Gragg, Kaufman and Stewart [34] (e.g. by Ruhe [112] or by Reichel and Gragg [108]). More exotic values like $K = 100.05$ [60] or $K = \sqrt{5}$ [50] have also been implemented. In 1989, Hoffmann [76] tested a wide range of values $K = 2, 10, \ldots, 10^{10}$. The conclusion of his experiments is that the K–criterion is always satisfied either at the first loop or at the second and the final level of orthogonality is proportional to the parameter K and to the machine precision, exactly as is the case for two vectors.

The goal of this paper is to give new ideas on the subject of selective reorthogonalization. In Section 1.5.1, we show that MGS2 applied to numerically nonsingular matrices gives a set of vectors orthogonal to machine precision. This is summarized in Theorem 1. The proof given in Section 1.5.1 is strongly related to the work of Björck [13]. In fact we extend his result for MGS to MGS2. Section 1.1 to Section 1.5 use his results directly with modifications adapted to a second loop of reorthogonalization. In Sections 1.5 to 1.11, we develop special results that aim to show that the R–factor corresponding to the second loop is well–conditioned. To work at step p of the algorithm, an assumption on the level of orthogonality at the previous step is necessary; this is done in Section 1.8 using an induction assumption. In Section 1.12, we adapt the work of Björck [13] to conclude that the level of orthogonality at step p is such that the induction assumption holds. During this proof, several assumptions are made, each of them are necessary during the proof, in Section 1.13, for sake of clarity, we encompass all these assumptions into one. Finally, in Section 1.14, we conclude the proof by induction. In Section 1.5.2.1, we give a new criterion for the modified Gram–Schmidt algorithm. This criterion is dependent on a single parameter L. We call this criterion the L–criterion and the resulting algorithm is named MGS2(L). This criterion appears naturally from the proof of Section 1.5.1 and the result of Theorem 1 for MGS2 holds also for MGS2(L) when $L < 1$. Therefore, we state that MGS2(L) with $L < 1$ applied to numerically nonsingular matrices gives a set of vectors orthogonal to machine precision. In Section 1.5.2.2, we give a counter–example matrix for which, if $L = 1.03$, then MGS2(L) provides a set of vectors that are far from orthogonal. Concerning the K–criterion, first of all we notice that the K–criterion makes sense for $K > 1$, otherwise MGS2(K) reduces to MGS2. In Section 1.5.3, we give counter–example matrices for which MGS2(K), K ranging from 1.43 down to 1.05, provides a set of vectors that are far from orthogonal. These examples illustrate that the K–criterion may not be robust.

The result established in Section 1.5.1 for MGS2 is similar to that given in [62]. Both papers establish with two different proofs that MGS2 gives a set of vectors orthogonal to machine precision. However the proof given in this paper is different and applies only to the modified Gram–Schmidt algorithm whereas the classical algorithm is covered by the proof in [62]. The advantage of this new proof is that it enables us to derive the L–criterion for the modified Gram–Schmidt algorithm. Moreover, this paper extends the work of Björck [13] directly from MGS to MGS2(L).
In the error analysis, we shall assume that floating–point arithmetic is used, and follow the technique and notations of Wilkinson [136] and Björck [13]. Let 'op' denote any of the four operators $+ \ - \ * \ /$. Then an equation of the form

$$z = \mathrm{fl}(x\text{`op'}y)$$

will imply that x, y and z are floating–point numbers and z is obtained from x and y using the appropriate floating–point operation. We assume that the rounding errors in these operations are such that

$$\mathrm{fl}(x\text{`op'}y) = (x\text{`op'}y)(1+\varepsilon), |\varepsilon| \leq 2^{-t}$$

where 2^{-t} is the unit roundoff.
In Section 1.5.1 and Section 1.5.2.1, in order to distinguish computed quantities from exact quantities, we use an overbar on the computed quantities. For the sake of readability in Section 1.5.2.2 and 1.5.3, that are dedicated to numerical experiments, the overbars are no longer used. Throughout this paper, the matrices are denoted with bold and capital characters, e.g. $\mathbf{A}$, the vectors with bold characters, e.g. $\mathbf{x}$, the scalars are in normal font, e.g. η. The entry (i,j) of A is denoted by a_{ij}. However, when there may be an ambiguity, we use a comma, e.g. the entry $(j-1,j)$ of A is denoted by $a_{j-1,j}$. The j^{th} column of A is the vector a_j. The paper is written for real matrices, the Euclidean scalar product is denoted by $\mathbf{x}^T\mathbf{y}$, $\| \ \|_2$ stands for the 2–norm for vectors and the induced norm for matrix, $\| \ \|_F$ stands for the Frobenius norm. $\sigma_{\min}(A)$ is the minimum singular value of A in the 2–norm. $\kappa(A)$ is the condition number of A in the 2–norm. I_p is the identity matrix of dimension p. Finally, we shall mention that our results also extend to complex arithmetic calculations.

1.5.1 Adaptation of the work by Björck (1967) for the modified Gram–Schmidt algorithm (MGS) to the modified Gram–Schmidt algorithm with one reorthogonalization step (MGS2)

1.5.1.1 Description of the algorithm MGS2 without square roots

In this section, we use the same approach as Björck in [13]. In his paper, he considers the MGS algorithm without square roots to study its numerical behaviour in floating–point arithmetic. In order to keep most of our work in agreement with his work we also study the MGS2 algorithm without square roots instead of the MGS2 algorithm (Algorithm 1). The MGS2 algorithm without square roots is described by Algorithm 3.

Algorithm 3 MGS2 without square roots

for $j = 1$ to n **do**
  $\mathbf{a}_j^{(1)(1)} = \mathbf{a}_j$
  for $k = 1$ to $j-1$ **do**
    $r_{kj}^{'(1)} = \mathbf{q}_k^{'T}\mathbf{a}_j^{(k)(1)}/d_k$
    $\mathbf{a}_j^{(k+1)(1)} = \mathbf{a}_j^{(k)(1)} - \mathbf{q}_k^{'} r_{kj}^{'(1)}$
  end for
  $\mathbf{a}_j^{(1)(2)} = \mathbf{a}_j^{(j)(1)}$
  for $k = 1$ to $j-1$ **do**
    $r_{kj}^{'(2)} = \mathbf{q}_k^{'T}\mathbf{a}_j^{(k)(2)}/d_k$
    $\mathbf{a}_j^{(k+1)(2)} = \mathbf{a}_j^{(k)(1)} - \mathbf{q}_k^{'} r_{kj}^{'(2)}$
  end for
  $\mathbf{q}_j^{'} = \mathbf{a}_j^{(j)(2)}$
  $d_j = \|\mathbf{q}_j^{'}\|_2^2$
  $r_{kj}^{'} = r_{kj}^{'(1)} + r_{kj}^{'(2)}, 1 \le k \le j-1$
  $r_{jj}^{'} = 1$
end for

The factorization resulting from MGS2 without square roots is denoted by

$$A = Q'R'$$

where R' is a unit upper triangular matrix and $(Q')^TQ'$ is diagonal. The main interest in that approach is to avoid the square root operation ($\sqrt{}$) in floating–point arithmetic. The associated algorithm only requires the four basic operations that are $+$, $-$, $*$ and $/$. In exact arithmetic, the link between the QR–factors Q' and R' of Algorithm 3 and the QR–factors Q and R of Algorithm 1 is

$$\mathbf{q}_j = \mathbf{q}_j^{'}/\|\mathbf{q}_j^{'}\|_2 \quad \text{and} \quad r_{kj} = r_{kj}^{'}\|\mathbf{q}_j^{'}\|_2 \quad k = 1, \ldots, j-1, \quad j = 1, \ldots, n.$$

1.5.1.2 Basic definitions for the error analysis

Following Björck [13], we define for $j = 1, \ldots, n$, the computed quantities for Algorithm 3

$$\begin{aligned}
&\bar{r}_{kj}^{'(r)} = \mathrm{fl}(\bar{q}_k^{'T}\bar{a}_j^{(k)(r)}/\bar{d}_k), && \text{for } k = 1, \ldots, j-1 \text{ and } r = 1, 2,\\
&\bar{a}_j^{(k+1)(r)} = \mathrm{fl}(\bar{a}_j^{(k)(r)} - \bar{q}_k^{'}\bar{r}_{kj}^{'(r)}), && \text{for } k = 1, \ldots, j-1 \text{ and } r = 1, 2,\\
&\bar{q}_j^{'} = \bar{a}_j^{(j)(2)},\\
&\bar{d}_j = \mathrm{fl}(\|\bar{q}_j^{'}\|_2^2),\\
&\bar{r}_{kj}^{'} = \mathrm{fl}(\bar{r}_{kj}^{'(1)} + \bar{r}_{kj}^{'(2)}), && \text{for } k = 1, \ldots, j-1,\\
&\bar{r}_{jj}^{'} = \mathrm{fl}(1).
\end{aligned}$$

The initialization is:

$$\bar{a}_j^{(1)(1)} = a_j,$$

at the end of the first loop (i.e $r = 1$) the following copy is performed before starting the next loop (i.e. $r = 2$)

$$\bar{a}_j^{(j)(2)} = \bar{a}_j^{(1)(1)}.$$

We also introduce the normalized quantities for $j = 1, \ldots, n$

$$\begin{array}{ll} \bar{q}_j = d_j^{-1/2}\bar{q}_j^{'}, & \bar{r}_{jj} = d_j^{1/2}, \\ \forall j = 1, \ldots, k-1, \quad \bar{r}_{kj}^{(r)} = d_j^{1/2}\bar{r}_{kj}^{'(r)}, & \bar{r}_{kj} = \bar{r}_{kj}^{(1)} + \bar{r}_{kj}^{(2)}, \end{array} \tag{1.73}$$

where

$$d_j^{1/2} = \begin{cases} \|\bar{q}_j^{'}\|_2, & \bar{q}_j^{'} \neq 0, \\ 1, & \bar{q}_j^{'} = 0. \end{cases}$$

Note that these latter quantities are never computed by Algorithm MGS2 without square roots, they are defined a posteriori. Thus expressions in (1.73) are exact relations.
From (1.73), the following relations also hold

$$\|\bar{q}_j\|_2 = 1, \quad \bar{r}_{jj} = \|\bar{a}_j^{(j)(2)}\|_2 \quad \text{and} \quad \bar{a}_j^{(j)(2)} = \bar{q}_j\bar{r}_{jj}.$$

The first relation implies that $I - \bar{q}_j\bar{q}_j^T$ is an orthogonal projection.
This section aims to prove the following Theorem.

Theorem 1
Let A be an m by n matrix on which MGS2 without square roots is run using a well designed floating–point arithmetic to obtain the computed Q–factor $\bar{Q}$.
Let 2^{-t} be the unit roundoff.
Let L be a real such that $0 < L < 1$. If

$$\frac{1}{L(1-L)} \times 10n^{\frac{5}{2}}(4.5m + 2)2^{-t} \cdot \kappa_2(A) \leq 1. \tag{1.74}$$

then $\bar{Q}$ is such that

$$\|I - \bar{Q}^T\bar{Q}\|_2 \leq \frac{2.61}{1-L} \cdot n^{\frac{3}{2}}(n + 1 + 2.5m)2^{-t}. \tag{1.75}$$

Notice that equation (1.75) indicates that the level of orthogonality reached with MGS2 is of the order of the machine precision and that assumption (1.74) implies that A is numerically nonsingular. In the remainder of this section, we make a series of assumptions on A that holds until the end of the section. In Paragraph 1.5.1.13, we combine all these assumption in one to finally obtain equation (1.74).

1.5.1.3 Errors in an elementary projection

The complete MGS2 algorithm is based on a sequence of elementary projections. In that respect, it is important to fully understand what is going on for each of them. In exact arithmetic, we have the following relations

$$\begin{array}{c} a_j^{(k+1)(r)} = a_j^{(k)(r)} - q_k r_{kj}^{(r)} \quad , \\ a_j^{(k+1)(r)} = (I - q_k q_k^T)a_j^{(k)(r)} \quad , \\ q_k^T a_j^{(k)(r)} = r_{kj}^{(r)} \\ \text{and} \quad \|a_j^{(k+1)(r)}\|_2 \leq \|a_j^{(k)(r)}\|_2. \end{array}$$

Björck [13], in his error analysis of an elementary projection, gives the equivalent of these four relations in floating–point arithmetic. We recall his results. In this section, the set of indices j for the column, r for the loop and k for the projection are frozen. Following Björck [13] we assume

$$m \geq 2 \quad \text{and} \quad 2n(m+2)2^{-t_1} < 0.01, \tag{1.76}$$

where $t_1 = t - \log_2(1.06)$.
If $\bar{q}_k^{'} \neq 0$, we define the related errors $\delta_j^{(k)(r)}$ and $\eta_j^{(k)(r)}$ by

$$\bar{a}_j^{(k+1)(r)} = \bar{a}_j^{(k)(r)} - \bar{q}_k \bar{r}_{kj}^{(r)} + \delta_j^{(k)(r)}, \tag{1.77}$$

$$\bar{a}_j^{(k+1)(r)} = (I - \bar{q}_k \bar{q}_k^T)\bar{a}_j^{(k)(r)} + \eta_j^{(k)(r)}. \tag{1.78}$$

In the singular situation, that is, when $\bar{q}_k^{'} = 0$ these relations are satisfied with

$$\bar{a}_j^{(k+1)(r)} = \bar{a}_j^{(k)(r)} \quad \text{and} \quad \delta_j^{(k)(r)} = \eta_j^{(k)(r)} = 0. \tag{1.79}$$

In the nonsingular case, Björck [13] shows that

$$\|\delta_j^{(k)(r)}\|_2 \leq 1.45 \cdot 2^{-t} \|\bar{a}_j^{(k)(r)}\|_2 \quad \text{and} \quad \|\eta_j^{(k)(r)}\|_2 \leq (2m+3) \cdot 2^{-t_1} \|\bar{a}_j^{(k)(r)}\|_2. \tag{1.80}$$

The error between $\bar{q}_k^T \bar{a}_j^{(k)(r)}$ and the computed value $\bar{r}_{kj}^{(r)}$ is given by

$$|\bar{q}_k^T \bar{a}_j^{(k)(r)} - \bar{r}_{kj}^{(r)}| < ((m+1) \cdot |\bar{q}_k^T \bar{a}_j^{(k)(r)}| + m\|\bar{a}_j^{(k)(r)}\|_2)2^{-t_1} \leq (2m+1)2^{-t_1} \cdot \|\bar{a}_j^{(k)(r)}\|_2. \tag{1.81}$$

In exact arithmetic, we have $a_j^{(k+1)(r)} = (I_m - q_k q_k^T)a_j^{(k)(r)}$ and so $\|a_j^{(k+1)(r)}\|_2 \leq \|a_j^{(k)(r)}\|_2$. In floating–point arithmetic, it can happen that the norm of the vector $a_j^{(k+1)(r)}$ gets larger than $a_j^{(k)(r)}$ due to the rounding errors. It is therefore important to have an upper bound to control $a_j^{(k+1)(r)}$. After k projections, $k < n$, Björck [13] shows that

$$\|\bar{a}_j^{(k)(r)}\|_2 < 1.006 \|a_j^{(1)(r)}\|_2 \tag{1.82}$$

The constant 1.006 comes from assumption (1.76). For more details, we refer directly to Björck [13]. Since $1.006^2 < 1.013$

$$\|\bar{a}_j^{(k)(r)}\|_2 < 1.013 \|a_j\|_2. \tag{1.83}$$

1.5.1.4 Errors in the factorization

We define

$$E = \bar{Q}\bar{R} - A. \tag{1.84}$$

We shall prove the following inequality

$$\boxed{\|E\|_F < 2.94(n-1) \cdot 2^{-t} \|A\|_F.} \tag{1.85}$$

Summing (1.77) for $k = 1, 2, \ldots, j-1$ and $r = 1, 2$ and using the relations

$$\bar{a}_j^{(1)(1)} = a_j, \quad \bar{a}_j^{(j)(1)} = \bar{a}_j^{(1)(2)}, \quad \bar{a}_j^{(j)(2)} = \bar{q}_j \bar{r}_{jj}, \quad \bar{r}_{kj} = \bar{r}_{kj}^{(1)} + \bar{r}_{kj}^{(2)},$$

we get

$$\sum_{k=1}^{j} \bar{q}_k \cdot \bar{r}_{kj} - a_j = \sum_{k=1}^{j-1} (\delta_j^{(k)(1)} + \delta_j^{(k)(2)}). \tag{1.86}$$

Let us define $\delta_j = \sum_{k=1}^{j-1} (\delta_j^{(k)(1)} + \delta_j^{(k)(2)})$. Then, from inequalities (1.80), we have

$$\|\delta_j\|_2 < 1.45 \cdot 2^{-t} \sum_{r=1}^{2} \sum_{k=1}^{j-1} \|\bar{a}_j^{(k)(r)}\|_2.$$

Using both inequality (1.83) and the fact that $1.013 \times 1.45 \times 2 < 2.94$, we have

$$\|\delta_j\|_2 < 2.94 \cdot 2^{-t} (j-1) \|a_j\|_2.$$

Finally, we obtain

$$\|E\|_F = (\sum_{j=1}^{n} \|\delta_j\|_2^2)^{1/2} < 2.94 \cdot 2^{-t} (n-1) (\sum_{j=1}^{n} \|a_j\|_2^2)^{1/2} = 2.94(n-1) \cdot 2^{-t} \|A\|_F.$$

1.5.1.5 Nonsingularity of $\bar{A}$

From equation (1.84), a sufficient condition for $\bar{A} = \bar{Q}\bar{R}$ to have full rank is given by Björck [13]. If the exact factorization of A is $A = QR$ then $\bar{A}$ has rank n if

$$2.94(n-1) \cdot 2^{-t} \|A\|_F \|R^{-1}\|_2 \leq \sqrt{2} - 1. \tag{1.87}$$

We assume in the following that inequality (1.87) is satisfied. This ensures that, for all $r = 1, 2$ and for all $j = 1, \ldots, n$,

$$\|\bar{a}_j^{(j)(r)}\|_2 \neq 0.$$

1.5.1.6 Theorem of Pythagoras

The purpose of this section is to exhibit an upper bound for

$$\sqrt{\sum_{i=1}^{j-1} (\bar{r}_{ij}^{(r)})^2}, \tag{1.88}$$

that will be used later in Sections 1.5.1.9, 1.5.1.10 and 1.5.1.11. In the sequel, we are interested in each step r individually. Therefore, for the sake of readability, we no longer use the subscript $^{(r)}$ to label the index loop.
In exact arithmetic, after the j^{th} step of the MGS algorithm, we have

$$a_j = \sum_{k=1}^{j-1} (q_k r_{kj}) + a_j^{(j)}$$

and as the vectors $q_k, k = 1, \ldots, j-1$ are orthonormal

$$\sum_{k=1}^{j-1} (r_{kj})^2 + \|a_j^{(j)}\|_2^2 = \|a_j\|_2^2. \tag{1.89}$$

Equation (1.89) is nothing but the theorem of Pythagoras. Still in exact arithmetic, let Q_{j-1} be such that $\|q_k\|_2 = 1, k = 1, \ldots, j-1$ without any additional assumption. Then, from the column a_j running step j of the MGS algorithm, we get

$$\begin{array}{rlcl}
a_j^{(1)} = (I - q_1 q_1^T)a_j, & \text{with } r_{1j} = q_1^T a_j & \Rightarrow & \|a_j\|_2^2 = (r_{1j})^2 + \|a_j^{(1)}\|_2^2, \\
\vdots & \vdots & & \vdots \\
a_j^{(j)} = (I - q_{j-1}q_{j-1}^T)a_j^{(j-1)}, & \text{with } r_{j-1,j} = q_{j-1}^T a_j^{(j-1)} & \Rightarrow & \|a_j^{(j-1)}\|_2^2 = (r_{j-1,j})^2 + \|a_j^{(j)}\|_2^2, \\
 & & \Rightarrow & \|a_j\|_2^2 = \sum_{k=1}^{j-1}(r_{kj})^2 + \|a_j^{(j)}\|_2^2.
\end{array}$$

We recover Property (1.89). Therefore we have the following statement: *when step j of MGS is performed in exact arithmetic with $\|q_k\|_2 = 1, k = 1, \ldots, j-1$, property (1.89) is true.* We apply the same idea in floating–point calculations. From equation (1.77),

$$\begin{aligned}
& \bar{a}_j^{(k+1)} = \bar{a}_j^{(k)} - \bar{q}_k \bar{r}_{kj} + \delta_j^{(k)}, \\
\Rightarrow \quad & \bar{a}_j^{(k)} + \delta_j^{(k)} = \bar{a}_j^{(k+1)} + \bar{q}_k \bar{r}_{kj}, \\
\Rightarrow \quad & \|\bar{a}_j^{(k)}\|_2^2 + \alpha_j^{(k)} = \|\bar{a}_j^{(k+1)}\|_2^2 + (\bar{r}_{kj})^2,
\end{aligned} \tag{1.90}$$

where

$$\alpha_j^{(k)} = (\delta_j^{(k)})^T \delta_j^{(k)} + 2(\delta_j^{(k)})^T \bar{a}_j^{(k)} - 2\bar{r}_{kj}(\bar{q}_k)^T \bar{a}_j^{(k+1)}.$$

Therefore we can get the following upper bound for $|\alpha_j^{(k)}|$

$$|\alpha_j^{(k)}| \leq \|\delta_j^{(k)}\|_2^2 + 2\|\delta_j^{(k)}\|_2 \|\bar{a}_j^{(k)}\|_2 + 2|\bar{r}_{kj}||\bar{q}_k^T \bar{a}_j^{(k+1)}|. \tag{1.91}$$

From equation (1.78) it follows that

$$(\bar{q}_k)^T \bar{a}_j^{(k+1)} = (\bar{q}_k)^T \eta_j^{(k)}, \tag{1.92}$$

and therefore

$$|\bar{q}_k^T \bar{a}_j^{(k+1)}| \leq \|\eta_j^{(k)}\|_2. \tag{1.93}$$

For $|\bar{r}_{kj}|$, equation (1.81) gives

$$|\bar{r}_{kj}| \leq (1 + (2m+1)2^{-t_1}) \cdot \|\bar{a}_j^{(k)(r)}\|_2 \leq 1.01 \cdot \|\bar{a}_j^{(k)(r)}\|_2. \tag{1.94}$$

Using equations (1.76), (1.80), (1.82), (1.93) and (1.94) in inequality (1.91), we get

$$\begin{aligned}
|\alpha_j^{(k)}| &\leq (1.006)^2 \times [1.45^2 \times 2^{-t} + 2 \times 1.45 + 2 \times 1.06 \times 1.01 \times (2m+3)] \cdot 2^{-t} \|\bar{a}_j\|_2^2, \\
&\leq (4.34m + 9.33)2^{-t} \cdot \|a_j\|_2^2,
\end{aligned} \tag{1.95}$$

where we use inequality (1.76) to bound 2^{-t} with 0.0016.
Summing equality (1.90) for $k = 1, \ldots, j-1$ gives

$$\|a_j\|_2^2 + \sum_{k=1}^{j-1} \alpha_j^{(k)} = \|\bar{a}_j^{(j)}\|_2^2 + \sum_{k=1}^{j-1} (\bar{r}_{kj})^2,$$

and then using inequality (1.95), we obtain

$$| \ (\|\bar{a}_j^{(j)}\|_2^2 + \sum_{k=1}^{j-1}(\bar{r}_{kj})^2) - \|a_j\|_2^2 \ | \leq (4.34m + 9.33)(j-1)2^{-t_1} \cdot \|a_j\|_2^2.$$

Using the fact that $\sqrt{1+x} \leq 1 + x/2$, for all $x \geq -1$, we have

$$\sqrt{\|\bar{a}_j^{(j)}\|_2^2 + \sum_{k=1}^{j-1}(\bar{r}_{kj})^2} \leq [1 + (2.17m + 4.67)(j-1)2^{-t_1}] \cdot \|a_j\|_2. \tag{1.96}$$

Let us assume that

$$(2.04m + 4.43)(j-1)2^{-t_1} \leq 0.01, \tag{1.97}$$

then we get

$$\boxed{\sqrt{\|\bar{a}_j^{(j)}\|_2^2 + \textstyle\sum_{k=1}^{j-1}(\bar{r}_{kj})^2} \leq 1.01 \cdot \|a_j\|_2.} \tag{1.98}$$

We remark that equation (1.98) and assumption (1.97) are satisfied without any assumption on the orthogonality of the columns of $\bar{Q}_{j-1}$.

1.5.1.7 Condition number of A and maximum value of $K_j^{(1)} = \frac{\|a_j\|_2}{\|\bar{a}_j^{(j)(1)}\|_2}$, for $j = 1, \ldots, n$

We define

$$K_j^{(1)} = \frac{\|a_j\|_2}{\|\bar{a}_j^{(j)(1)}\|_2} \quad \text{and} \quad K_j^{(2)} = \frac{\bar{a}_j^{(1)(2)}}{\|\bar{a}_j^{(j)(2)}\|_2}. \tag{1.99}$$

Notice that $\|\bar{a}_j^{(j)(1)}\|_2 \neq 0$ and $\|\bar{a}_j^{(j)(2)}\|_2 \neq 0$ because we make the assumption (1.87) on the numerical nonsingularity of A. We have seen in the introduction that the quantity $K_j^{(1)}$ plays an important role for checking the quality of the orthogonality for the computed vector $\bar{q}_j$ with respect to the previous $\bar{q}_i$, $i = 1, \ldots, n$. In this section, we derive an upper bound for $K_j^{(1)}$.
In exact arithmetic, if MGS is run on A to obtain the QR–factors Q and R then

$$\sigma_{\min}(A) = \sigma_{\min}(R) \leq |r_{jj}| = \|a_j^{(j)(1)}\|_2 \quad \text{and} \quad \|A\|_2 \geq \|a_j\|_2$$

so

$$K_j^{(1)} = \frac{\|a_j\|_2}{\|a_j^{(j)(1)}\|_2} \leq \kappa(A). \tag{1.100}$$

Inequality (1.100) indicates that, in exact arithmetic, $K_j^{(1)}$ is always less than the condition number of A, $\kappa_2(A)$. With rounding errors, we can establish a bound similar to inequality (1.100).
We recall equation (1.86) that is

$$a_k = \sum_{i=1}^{k} \bar{q}_k \cdot \bar{r}_{ik} - \delta_k, \quad k = 1, \ldots, j-1.$$

For $k = j$, we just consider the first loop (i.e. $r = 1$). This gives

$$a_j = \sum_{i=1}^{j} \bar{q}_i \cdot \bar{r}_{i,j}^{(1)} + \bar{a}_j^{(j)(1)} - \delta_j^{(1)}$$

with $\delta_j^{(1)} = \sum_{k=1}^{j-1} \delta_j^{(k)(1)}$. In matrix form, this can be written as

$$A_j = \bar{Q}_{j-1}\bar{R}_{(j-1,j)} - \Delta_j$$

with $\bar{Q}_{j-1} \in \mathbb{R}^{m\times j-1}$

$$\bar{Q}_{j-1} = [\bar{q}_1, \ldots, \bar{q}_{j-1}]$$

and $\bar{R}_{(j-1,j)} \in \mathbb{R}^{j-1\times j}$ such that

$$\bar{R}_{(j-1,j)} = \begin{pmatrix} \bar{r}_{1,1} & \ldots & \bar{r}_{1,j-1} & \bar{r}_{1,j}^{(1)} \\ & \ddots & \vdots & \vdots \\ & & \bar{r}_{j-1,j-1} & \bar{r}_{j-1,j}^{(1)} \end{pmatrix}.$$

Finally $\Delta_j \in \mathbb{R}^{m\times j}$ is defined by

$$\Delta_j = [\delta_1, \ldots, \delta_{j-1}, \delta_j^{(1)} - \bar{a}_j^{(j)(1)}],$$

with

$$0 < \|\Delta_j\|_F \leq 2.94(j-1)\cdot 2^{-t}\|A_j\|_F + \|\bar{a}_j^{(j)(1)}\|_2.$$

Notice that, by construction, the matrix $\bar{Q}_{j-1}\bar{R}_{(j-1,j)}$ is of rank $j-1$. Therefore the matrix $A_j + \Delta_j$ is singular, whereas we assume that the matrix A_j is nonsingular. The distance to singularity for a matrix A_j can be related to its minimum singular value. Some theorems on relative distance to singularity can be found in many books (e.g. [63, p. 73] or [75, p. 123]). Although the textbooks usually assume the matrices are square, this statement is also true for rectangular matrices. In our case, we have

$$\sigma_{\min}(A_j) = \min\{\|\Delta\|_2, \Delta \in \mathbb{R}^{m\times j} \text{ so that } A_j + \Delta \text{ is singular } \} \leq \|\Delta_j\|_2.$$

Dividing by $\|A_j\|_2$, we get

$$\frac{1}{\kappa_2(A_j)} \leq \frac{\|\Delta_j\|_2}{\|A_j\|_2} \leq \frac{\|\Delta_j\|_F}{\|A_j\|_2},$$

and since we know that $\|\Delta_j\|_F \neq 0$, this gives

$$\kappa_2(A) \geq \kappa_2(A_j) \geq \frac{\|A_j\|_2}{\|\Delta_j\|_F} \geq \frac{1}{2.94(n-1)\cdot 2^{-t}\frac{\|A_j\|_F}{\|A_j\|_2} + \frac{\|\bar{a}_j^{(j)(1)}\|_2}{\|A_j\|_2}},$$

Let us write

$$\kappa_2(A) \geq \frac{1}{2.94(j-1)\cdot 2^{-t}\frac{\|A_j\|_F}{\|A_j\|_2} + \frac{\|\bar{a}_j^{(j)(1)}\|_2}{\|a_j\|_2}\frac{\|a_j\|_2}{\|A_j\|_2}},$$

since

$$\frac{\|\bar{a}_j^{(j)(1)}\|_2}{\|a_j\|_2} = K_j^{(1)} \quad , \quad \frac{\|a_j\|_2}{\|A_j\|_2} \leq 1 \quad \text{and} \quad \frac{\|A_j\|_F}{\|A_j\|_2} < \sqrt{j},$$

we get

$$\kappa_2(A) \geq \frac{1}{2.94(j-1)\sqrt{j} \cdot 2^{-t} + \frac{1}{K_j^{(1)}}}.$$

For instance, if we assume that

$$2.94(n-1)n^{\frac{1}{2}} \cdot 2^{-t}.\kappa_2(A) < 0.09, \tag{1.101}$$

where the value 0.09 is taken arbitrarily but another value leads to a final similar result, then we have the following inequality

$$K_j^{(1)} \leq \frac{1}{1 - 2.94(j-1)j^{\frac{1}{2}}2^{-t} \cdot \kappa_2(A)}\kappa_2(A).$$

Using assumption (1.101) we get

$$\boxed{K_j^{(1)} \leq 1.1 \cdot \kappa_2(A).} \tag{1.102}$$

We remark that equation (1.102) and assumption (1.101) are independent of the orthogonality of the previously computed $\bar{Q}_{j-1}$; it is just a consequence of equation (1.84).
Note that the value 0.09 of the right–hand side in equation 1.101 is arbitrary. We point out that since the numerical properties of Gram–Schmidt algorithm are invariant under column scaling (without consideration on underflow), instead of the condition number $\kappa(\mathbf{A})$ one can use

$$\kappa_D(\mathbf{A}) = \min_{\mathbf{D} \text{ diagonal matrix}} \kappa(\mathbf{AD}).$$

1.5.1.8 Induction assumption

We want to show that the orthogonality of the computed vectors $\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_n$ is of the order of the machine precision.
In exact arithmetic, at step j, to show that the vector q_j generated by the MGS algorithm is orthogonal to the previous ones, we use the fact that the previous q_i, $i = 1, \ldots, j-1$ are already orthogonal to each other. Therefore to show the orthogonality at step j in floating–point arithmetic, we make an assumption on the orthogonality at step $j-1$.
The orthogonality of the computed vectors $\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_n$ can be measured by the norm of the matrix $(I - \bar{Q}^T\bar{Q})$. Let U_p, $p = 1, \ldots, n$ be the strictly upper triangular matrix of size (p, p) with entries:

$$u_{ij} = \bar{q}_i^T \bar{q}_j, \quad 1 \leq i < j \leq p \quad \text{and } u_{ij} = 0, \quad 1 \leq j \leq i \leq p.$$

We note $U = U_n$ and have:

$$I - \bar{Q}^T\bar{Q} = -(U + U^T). \tag{1.103}$$

We make a proof by induction to show that $\|U\|_2$ is small at step n. Therefore, we assume that at step $p-1$:

$$\boxed{\|U_{p-1}\|_2 \leq \lambda.} \tag{1.104}$$

Our aim is to show that at step p, we still have $\|U_p\|_2 \leq \lambda$. The value of λ is exhibited during the proof.
In the following, the index variables i, j, k and p are such that

$$1 \leq j \leq p \leq n, \quad 1 \leq i \leq j \quad \text{and} \quad 1 \leq k \leq j$$

1.5.1.9 Bound for $|\bar{q}_k^T \bar{a}_j^{(j)(1)}|$, for $k = 1, \ldots, j-1$ and for $j = 1, \ldots, p$

$|\bar{q}_k^T \bar{a}_j^{(j)(1)}|$ represents the orthogonality between $\bar{q}_k$, $k = 1, \ldots, j-1$, and the vector $\bar{a}_j^{(j)(1)}$ given by the first step of MGS ($r = 1$). In exact arithmetic, this quantity is zero. Following Björck [13], we sum equation (1.77) for $i = k+1, k+2, \ldots, j-1$ and $r = 1$, to get

$$\bar{a}_j^{(j)(1)} = \bar{a}_j^{(k+1)(1)} - \sum_{i=k+1}^{j-1} \bar{q}_i \bar{r}_{ij}^{(1)} + \sum_{i=k+1}^{j-1} \delta_j^{(i)(1)}.$$

Hence, multiplying this relation by $\bar{q}_k^T$ and using (1.92), we get

$$\bar{q}_k^T \bar{a}_j^{(j)(1)} = - \sum_{i=k+1}^{j-1} (\bar{q}_k^T \bar{q}_i) \bar{r}_{ij}^{(1)} + \bar{q}_k^T (\bar{\eta}_j^{(k)(1)} + \sum_{i=k+1}^{j-1} \delta_j^{(i)(1)}).$$

Therefore :

$$|\bar{q}_k^T \bar{a}_j^{(j)(1)}| \leq \sqrt{\sum_{i=k+1}^{j-1} (\bar{r}_{ij}^{(1)})^2} \sqrt{\sum_{i=k+1}^{j-1} (\bar{q}_k^T \bar{q}_i)^2} + \|\bar{\eta}_j^{(k)(1)}\|_2 + \sum_{i=k+1}^{j-1} \|\delta_j^{(i)(1)}\|_2.$$

We can interpret the terms of the right–hand side.

1. the orthogonalization of $\bar{a}_j^{(k)(1)}$ against $\bar{q}_k$ is not performed exactly; this corresponds to the second term,
2. the resulting vector $\bar{a}_j^{(k+1)(1)}$ is orthogonalized on $\bar{q}_i$, $i = k+1, \ldots, j-1$, and, since $\bar{Q}$ is not orthogonal, we also lose orthogonality here; this corresponds to the first term,
3. moreover, all these projections $i = k+1, \ldots, j-1$ are also done inaccurately; this corresponds to the third term.

Using inequalities (1.80) and (1.82), we have

$$\|\bar{\eta}_j^{(k)(1)}\|_2 + \sum_{i=k+1}^{j-1} \|\delta_j^{(i)(1)}\|_2 \leq (2.14m + 3.20 + 1.46(j-k-1)) 2^{-t} \cdot \|a_j\|_2.$$

Finally, using inequalities (1.98) and (1.104), we get

$$|\bar{q}_k^T \bar{a}_j^{(j)(1)}| \leq [1.01\lambda + (2.14m + 1.46(j-k-1) + 3.20) 2^{-t}] \cdot \|a_j\|_2.$$

1.5.1.10 Bound for $|\bar{r}_{kj}^{(2)}|$, for $k = 1, \ldots, j-1$ and for $j = 1, \ldots, p$

Having a bound for the orthogonality of the first step, we now study its influence in the second step by computing $|\bar{r}_{kj}^{(2)}|$. Again summing equation (1.77) for $i = 1, 2, \ldots, k-1$ and $r = 2$ we get :

$$\bar{a}_j^{(k)(2)} = \bar{a}_j^{(j)(1)} - \sum_{i=1}^{k-1} \bar{q}_i \bar{r}_{ij}^{(2)} + \sum_{i=1}^{k-1} \delta_j^{(i)(2)}.$$

Hence multiplying by $\bar{q}_k^T$, we get

$$\bar{q}_k^T \bar{a}_j^{(k)(2)} = \bar{q}_k^T \bar{a}_j^{(j)(1)} - \sum_{i=1}^{k-1} (\bar{q}_k^T \bar{q}_i) \bar{r}_{ij}^{(2)} + \bar{q}_k^T \sum_{i=1}^{k-1} \delta_j^{(i)(2)}.$$

Taking moduli, we have

$$|\bar{q}_k^T \bar{a}_j^{(k)(2)}| \leq |\bar{q}_k^T \bar{a}_j^{(j)(1)}| + \sqrt{\sum_{i=1}^{k-1} (\bar{r}_{ij}^{(2)})^2} \sqrt{\sum_{i=1}^{k-1} (\bar{q}_k^T \bar{q}_i)^2} + \sum_{i=1}^{k-1} \|\delta_j^{(i)(2)}\|_2.$$

Similarly, as in Section 1.9, we bound each term in the right-hand side and get

$$|\bar{q}_k^T \bar{a}_j^{(k)(2)}| \leq [2.02\lambda + (2.14m + 1.46(j-2) + 3.20)2^{-t}] \cdot \|a_j\|_2.$$

Using inequalities (1.81) and (1.82), we know that $|\bar{q}_k^T \bar{a}_j^{(k)(2)} - \bar{r}_{kj}^{(2)}| \leq (2.15m + 1.08) \cdot 2^{-t} \|a_j\|_2$, therefore

$$|\bar{r}_{kj}^{(2)}| \leq [2.02\lambda + (4.29m + 1.46(j-2) + 4.28)2^{-t}] \cdot \|a_j\|_2,$$

This expression can be simplified to obtain

$$\boxed{|\bar{r}_{kj}^{(2)}| \leq [2.02\lambda + 5.75(m+1)2^{-t}] \cdot \|a_j\|_2.} \tag{1.105}$$

1.5.1.11 Bound for $K_j^{(2)} = \frac{\|\bar{a}_j^{(1)(2)}\|_2}{\|\bar{a}_j^{(j)(2)}\|_2}$, for $j = 1, \ldots, p$

While the quantity $K_j^{(1)}$ is important for the level of orthogonality after the first orthogonalization loop, the quantity $K_j^{(2)}$ is important for the level of orthogonality after the second orthogonalization loop. In exact arithmetic, we have $a_j^{(1)(2)} = a_j^{(j)(2)}$ and therefore $K_j^{(2)} = 1$. In this section, we show that $K_j^{(2)}$, in floating–point arithmetic, is close to one.
Let us again sum equation (1.77) for $r = 2$, $k = 1, 2, \ldots, j-1$, to get

$$\bar{a}_j^{(j)(2)} = \bar{a}_j^{(j)(1)} - \sum_{k=1}^{j-1} \bar{q}_k \bar{r}_{kj}^{(2)} + \sum_{k=1}^{j-1} \delta_j^{(k)(2)},$$

then

$$\|\bar{a}_j^{(j)(2)}\|_2 \geq \|\bar{a}_j^{(j)(1)}\|_2 - \|\sum_{k=1}^{j-1} \bar{q}_k \bar{r}_{kj}^{(2)}\|_2 - \sum_{k=1}^{j-1} \|\delta_j^{(k)(2)}\|_2. \tag{1.106}$$

The induction assumption (1.103) implies that $\|\bar{Q}\|_2 \leq \sqrt{1+\lambda^2}$. From this, we can get an upper bound for $\|\sum_{k=1}^{j-1} \bar{q}_k \bar{r}_{kj}^{(2)}\|_2$, that is

$$\|\sum_{k=1}^{j-1} \bar{q}_k \bar{r}_{kj}^{(2)}\|_2 \leq \|\bar{Q}\|_2 \| \begin{pmatrix} \bar{r}_{1j}^{(2)} \\ \vdots \\ \bar{r}_{j-1,j}^{(2)} \end{pmatrix} \|_2 \leq \sqrt{1+\lambda^2} \| \begin{pmatrix} \bar{r}_{1j}^{(2)} \\ \vdots \\ \bar{r}_{j-1,j}^{(2)} \end{pmatrix} \|_2.$$

Using inequality (1.105) we get

$$\|\sum_{k=1}^{j-1} \bar{q}_k \bar{r}_{kj}^{(2)}\|_2 \leq \sqrt{1+\lambda^2} \cdot \sqrt{j-1}[2.02\lambda + 5.75(m+1)2^{-t}] \cdot \|a_j\|_2.$$

With inequalities (1.80) and (1.106) we have

$$\|\bar{a}_j^{(j)(2)}\|_2 \geq \|\bar{a}_j^{(j)(1)}\|_2 - \left[\sqrt{1+\lambda^2} \cdot \sqrt{j-1}(2.02\lambda + 5.75(m+1)2^{-t}) + 1.47(j-1)2^{-t}\right] \|a_j\|_2.$$

Dividing by $\|\bar{a}_j^{(j)(1)}\|_2$ we have

$$1/K_j^{(2)} \geq 1 - \left[\sqrt{1+\lambda^2} \cdot \sqrt{j-1}[2.02\lambda + 5.75(m+1)2^{-t}] - 1.47(j-1)2^{-t}\right] K_j^{(1)}.$$

Let us assume that

$$1.1\kappa_2(A) \left[\sqrt{1+\lambda^2} \cdot \sqrt{j-1}[2.02\lambda + 5.75(m+1)2^{-t}] + 1.47(j-1)2^{-t}\right] \leq 0.67 < 1, \tag{1.107}$$

where the value 0.67 is taken arbitrarily, but another value leads to a final similar result. We obtain

$$K_j^{(2)} \leq \frac{1}{1 - K_j^{(1)}\sqrt{1+\lambda^2} \cdot \sqrt{j-1}[2.02\lambda + 5.75(m+1)2^{-t}]} \leq \frac{1}{0.67}.$$

This gives

$$\boxed{K_j^{(2)} \leq 1.5.} \tag{1.108}$$

We remark that assumption (1.107) is dependent on the parameter λ that is still not yet known.

1.5.1.12 Bound for the orthogonality of the vectors

Summing (1.77) from $k = i+1, i+2, \ldots, j-1$ and $r = 2$ we get

$$\bar{a}_j^{(j)(2)} = \bar{a}_j^{(i+1)(2)} - \sum_{k=i+1}^{j-1} \bar{q}_k \bar{r}_{kj}^{(2)} + \sum_{k=i+1}^{j-1} \delta_j^{(k)(2)}. \tag{1.109}$$

From equation (1.92), we have $\bar{q}_i^T \bar{a}_j^{(i+1)(2)} = \bar{q}_i^T \eta_j^{(i)(2)}$ and $\bar{a}_j^{(j)(2)} = \bar{q}_j \bar{r}_{jj}^{(2)}$. Therefore multiplying (1.109) by $\bar{q}_i^T$ we get

$$\sum_{k=i+1}^{j} \bar{r}_{kj}^{(2)} (\bar{q}_i^T \bar{q}_k) = \bar{q}_i^T (\eta_j^{(i)(2)} + \sum_{k=i+1}^{j-1} \delta_j^{(k)(2)}).$$

We divide this by $|\bar{r}_{jj}^{(2)}|$ (which is different from 0) to get

$$\sum_{k=i+1}^{j} \frac{\bar{r}_{kj}^{(2)}}{|\bar{r}_{jj}^{(2)}|} (\bar{q}_i^T \bar{q}_k) = \frac{\bar{q}_i^T (\eta_j^{(i)(2)} + \sum_{k=i+1}^{j-1} \delta_j^{(k)(2)})}{|\bar{r}_{jj}^{(2)}|}. \tag{1.110}$$

We recall that this equality is true for all $j = 1, \ldots, p$ and $i = 1, \ldots, j-1$.
Define M_p as the unit upper triangular matrix with the (k, j) entry, m_{kj}, given by

$$m_{kj} = \frac{\bar{r}_{kj}^{(2)}}{|\bar{r}_{jj}^{(2)}|}, \quad \text{for } k < j, \tag{1.111}$$

and let S_p be the strictly upper triangular matrix where the (i, j) entry, s_{ij}, is

$$s_{ij} = \frac{\bar{q}_i^T (\eta_j^{(i)(2)} + \sum_{k=i+1}^{j-1} \delta_j^{(k)(2)})}{|\bar{r}_{jj}^{(2)}|}, \quad \text{for } i < j.$$

Since the entry (i, k) of U_p is $u_{ik} = \bar{q}_i^T \bar{q}_k$, equation (1.110) can be rewritten as

$$\forall j = 1, \ldots, p, \quad \forall i = 1, \ldots, j-1, \qquad s_{ij} = \sum_{k=i+1}^{j} u_{ik} m_{kj}.$$

Taking into account the facts that U_p and S_p are strictly upper triangular and M_p is upper triangular, we obtain

$$S_p = U_p M_p. \tag{1.112}$$

In [13], Björck gives an upper bound for the 2–norm of each column of S_p as

$$\|s_j\|_2 \leq 0.87 \cdot n^{\frac{1}{2}} (n + 1 + 2.5m) 2^{-t} \frac{\|\bar{a}_j^{(j)(1)}\|_2}{|\bar{r}_{jj}^{(2)}|}.$$

Since $|\bar{r}_{jj}^{(2)}| = \|\bar{a}_j^{(j)(2)}\|_2$, we obtain

$$\|s_j\|_2 \leq 0.87 K_j^{(2)} \cdot n^{\frac{1}{2}} (n + 1 + 2.5m) 2^{-t}.$$

Using inequality (1.108) and the fact that $0.87 \times 1.5 = 1.305$, we get

$$\|S_p\|_2 \leq 1.305 n (n + 1 + 2.5m) 2^{-t}. \tag{1.113}$$

M_p is nonsingular. Therefore from equation (1.112) we have

$$\|U_p\|_2 \leq \|M_p^{-1}\|_2 \|S_p\|_2. \tag{1.114}$$

At this stage, the quantity of interest is $\|M_p^{-1}\|_2$.
It is interesting to relate this proof to that of Björck [13]. Björck [13] shows an inequality similar to inequality (1.114) for MGS, with $\|S_p\|_2$ of the order of the machine precision, U_p as defined in Section 1.8 but with the $\bar{q}$ coming from MGS and M_p as defined in equation (1.111) but with the $\bar{r}_{kj}$ coming from MGS. Since he proves that, for MGS, $\|M_p^{-1}\|_2$ is of the order of $\kappa(A)$, he obtains the result: the final orthogonality obtained with MGS is of the order of $\kappa(A)2^{-t}$. Our goal is to show that $\|M_p^{-1}\|_2$ is independent of $\kappa(A)$ and is of the order of 1.
An idea for controlling the 2–norm of M_p is to show that M_p is diagonally dominant by columns. Following Varah [94], we say that M_p is diagonally dominant by columns if

$$\forall j = 1, \ldots, n, \quad |m_{jj}| > \sum_{k \neq j} |m_{kj}|. \tag{1.115}$$

In our case, since M_p is unit triangular it would be diagonally dominant by columns if

$$\forall j = 1, \ldots, n, \quad 1 > \sum_{k=1}^{j-1} |m_{kj}|.$$

It becomes then natural to look for an upper bound for $\sum_{k=1}^{j-1} |m_{kj}|$ that is lower than one.
From (1.105) we have

$$\sum_{k=1}^{j-1} |m_{kj}| \leq (j-1)[2.02\lambda + 5.75(m+1)2^{-t}] \frac{\|a_j\|_2}{|\bar{r}_{jj}^{(2)}|},$$

therefore

$$\sum_{k=1}^{j-1} |m_{kj}| \leq (j-1)[2.02\lambda + 5.75(m+1)2^{-t}] K_j^{(1)} K_j^{(2)}.$$

Using equations (1.102) and (1.108), we get as $1.1 \times 1.5 = 1.65$

$$\sum_{k=1}^{j-1} |m_{kj}| \leq 1.65(j-1)[2.02\lambda + 5.75(m+1)2^{-t}] \kappa_2(A).$$

We assume that

$$1.65(n-1)[2.02\lambda + 5.75(m+1)2^{-t}] \kappa_2(A) \leq L, \tag{1.116}$$

where L is a real number such that $0 < L < 1$. With inequality (1.116), we obtain

$$\sum_{k=1}^{j-1} |m_{kj}| \leq L. \tag{1.117}$$

This means that M_p is diagonally dominant by columns.
Let us decompose M_p as

$$M_p = I_p + C_p$$

where C_p is strictly upper triangular. Inequality (1.117) means that

$$\|C_p\|_1 = \max_{j=1,\ldots,p} \sum_{k=1}^{j-1} |m_{kj}| \leq L. \tag{1.118}$$

In addition, we also have

$$(I_p + C_p)(I_p - C_p + \ldots + (-1)^n C_p^{n-1}) = I_p + (-1)^n C_p^n.$$

Since C_p is strictly upper triangular, it is nilpotent (i.e. we have $C_p^n = 0$) so that

$$M_p(I_p - C_p + \ldots + (-1)^n C_p^{n-1}) = I_p.$$

Therefore

$$M_p^{-1} = I_p - C_p + \ldots + (-1)^n C_p^{n-1}.$$

In norm this implies that

$$\begin{aligned} \|M_p^{-1}\|_2 &\leq 1 + \|C_p\|_1 + \|C_p\|_1^2 + \ldots + \|C_p\|_1^{n-1}, \\ &\leq 1 + L + L^2 + \ldots + L^{n-1} = \frac{1 - L^n}{1 - L}. \end{aligned}$$

Finally we get

$$\|M_p^{-1}\|_1 \leq \frac{1}{1 - L}, \tag{1.119}$$

which implies that

$$\|M_p^{-1}\|_2 \leq \frac{\sqrt{n}}{1 - L}. \tag{1.120}$$

Notice that inequality (1.119) is nothing else than the result of Corollary 1 of Varah [94] applied to matrices with unit diagonal. The parameter L has to be chosen between 0 and 1. It should neither be too close to 0, so that assumption (1.116) does not become too strong, nor be too close to 1, so that the bound (1.120) on $\|M_p^{-1}\|_1$ does not become too large. With inequalities (1.113), (1.114) and (1.120), we get

$$\boxed{\|U_p\|_2 \leq \tfrac{1.305}{1-L} \cdot n^{\frac{3}{2}}(n + 1 + 2.5m)2^{-t}.} \tag{1.121}$$

A natural choice for λ is then

$$\lambda = \frac{1.305}{1 - L} \cdot n^{\frac{3}{2}}(n + 1 + 2.5m)2^{-t}, \tag{1.122}$$

so that the induction assumption (1.104) is verified at step p.

1.5.1.13 Assumptions on A

Since λ is defined, it is possible to explicitly state the assumptions made on A. The assumptions made are equations (1.76), (1.87), (1.97), (1.101), (1.107) and (1.116). We focus here on the main assumption that is (1.116). We replace λ by its value and get

$$\frac{1}{L} \times 1.65(n-1)[2.02\frac{1.305}{1 - L} \times n^{\frac{3}{2}}(n + 1 + 2.5m) + 5.75(m + 1)]2^{-t} \cdot \kappa_2(A) \leq 1.$$

For the sake of simplicity we replace it with

$$\frac{1}{L(1-L)} \times 10n^{\frac{5}{2}}(4.5m+2)2^{-t} \cdot \kappa_2(A) \leq 1.$$

1.5.1.14 Conclusion of the proof by induction

We have shown that, if we assume (1.74) and define λ with (1.122), then

if at step $(p-1)$, we have $\|U_{p-1}\|_2 \leq \lambda$ then at step p we also have $\|U_p\|_2 \leq \lambda$.

At step $n=1$, U_1 is defined as $\|U_1\|_2 = 0$ and so $\|U_1\|_2 \leq \lambda$. From this, we conclude that at step n, we have

$$\|I - Q^T Q\|_2 \leq \frac{2.61}{1-L} \cdot n(n+1+2.5m)2^{-t}.$$

This completes the proof of Theorem 1.
Theorem 1 involves a parameter L while MGS2 is parameter free. We can nevertheless use the result of that theorem to assess the quality of the orthogonality of the set of vectors generated by MGS2 by setting $L = 0.5$. The value 0.5 is chosen in order to relax the most the assumption (1.74) on the nonsingularity of A.

1.5.2 Link with selective reorthogonalization

1.5.2.1 Sufficiency of the condition $L < 1$ for robust reorthogonalization

The key property of the matrix M is given by inequality (1.117). The main effort in the proof of Section 1 consists in showing that for all $j = 1, \ldots, n$ we have, after the reorthogonalization loop,

$$L_j^{(2)} = \sum_{k=1}^{j-1} \frac{|r_{kj}^{(2)}|}{r_{jj}^{(2)}} \leq L < 1.$$

However, this property may already occur after the first orthogonalization, that is

$$L_j^{(1)} = \sum_{k=1}^{j-1} \frac{|r_{kj}^{(1)}|}{\|a_j^{(1)}\|_2} \leq L < 1. \tag{1.123}$$

In this case, we do not need to reorthogonalize $a_j^{(1)}$, to comply with inequality (1.117) at the second loop since it is already satisfied at the first loop. From this, we propose a new algorithm that checks whether the inequality (1.123) is satisfied or not at step j, $r = 1$. We call the resulting criterion the L–criterion and the corresponding algorithm MGS2(L). MGS2(L) is the same as Algorithm MGS2(K) except that line 7 is replaced by

$$\textbf{if} \quad \frac{\sum_{k=1}^{j-1} |r_{kj}^{(1)}|}{\|a_j^{(1)}\|_2} \leq L \quad \textbf{then}.$$

Since we have derived MGS2 without square roots from MGS2, we derive MGS2(L) without square roots from MGS2(L). The proof established in Section 1 for MGS2 without square roots needs basically inequality (1.85) to be satisfied and $\|U_p\|_2 \leq \lambda$ assuming $\|U_{p-1}\|_2 \leq \lambda$, $p \geq 1$. Whether one loop or two are performed, inequality (1.85) holds. If the L–criterion is satisfied at step p for the first loop then we can state that $\|U_p\|_2 \leq \lambda$. If not, at the second loop, we have anyway $\|U_p\|_2 \leq \lambda$. Therefore Theorem 1 holds also for MGS2(L) without square roots. We recall that Theorem 1 is true for $0 < L < 1$.
From the Theorem 1 point of view, the optimal value of L for having the weaker assumption on A is 0.5. With respect to the orthogonality; the lower L is, the better the orthogonality. To minimize the computational cost of the algorithm, a large value of L would imply performing only a few reorthogonalizations. Therefore, in Theorem 1, the value for L between 0 and 1 is a trade-off between the computational cost and the expected orthogonality quality. In our experiments, we choose the value $L = 0.99$.

1.5.2.2 Necessity of the condition $L < 1$ to ensure the robustness of the selective reorthogonalization

In this section we exhibit some counter–example matrices A such that for, any given value $L > 1$, the orthogonality obtained by MGS2(L) may be very poor. Our strategy is to find a matrix such that:

Property 1. the matrix is numerically nonsingular but ill–conditioned.

Property 2. MGS2(L) applied to this matrix performs no reorthogonalization and so it reduces to MGS.

Let us define the matrix $A(n, \alpha) \in \mathbb{R}^{n \times n}$ as

$$A(n, \alpha) = UT_A(n, \alpha) = U \begin{pmatrix} \alpha & 1 & & \\ & \ddots & \ddots & \\ & & \alpha & 1 \\ & & & \alpha \end{pmatrix} \tag{1.124}$$

where $U \in \mathbb{R}^{n \times n}$ is such that $U^T U = I$.
Matrices $A(n, \alpha)$ have the property that if we apply MGS2(L) (in exact arithmetic), we get

$$L_j^{(1)} = \sum_{k=1}^{j-1} \frac{|r_{kj}^{(1)}|}{\|a_j^{(1)}\|_2} = \frac{1}{\alpha}. \tag{1.125}$$

If we set α such that $L_j^{(1)} > L$ that is $1/\alpha > L$, then the L–criterion is always satisfied. In this case, no reorthogonalization is performed, and then Property 2 is satisfied.
Moreover for all α, $0 < \alpha < 1$,the condition number of the matrix $\kappa(A(n, \alpha))$ can be made arbitrarily large by choosing an appropriate n. We justify this claim by

studying the matrix $T_A(n,\alpha)$. First of all, we have

$$T_A(n,\alpha)x_1 = \begin{pmatrix} \alpha & 1 & & & & \\ & \alpha & 1 & & & \\ & & \alpha & 1 & & \\ & & & \ddots & \ddots & \\ & & & & \alpha & 1 \\ & & & & & \alpha \end{pmatrix} \begin{pmatrix} 1 \\ -\alpha \\ \alpha^2 \\ \vdots \\ (-1)^{n-2}\alpha^{n-2} \\ (-1)^{n-1}\alpha^{n-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ (-1)^{n-1}\alpha^n \end{pmatrix}$$

therefore

$$\sigma_{\min}(T_A(n,\alpha)) \leq \frac{\|T_A(n,\alpha)x_1\|_2}{\|x_1\|_2} \leq \alpha^n \sqrt{\frac{1-\alpha^{2n}}{1-\alpha^2}}.$$

On the other hand, we also have

$$T_A(n,\alpha)x_2 = \begin{pmatrix} \alpha & 1 & & & & \\ & \alpha & 1 & & & \\ & & \alpha & 1 & & \\ & & & \ddots & \ddots & \\ & & & & \alpha & 1 \\ & & & & & \alpha \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ \alpha \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

and therefore

$$\sigma_{\max}(T_A(n,\alpha)) \geq \frac{\|T_A(n,\alpha)x_2\|_2}{\|x_2\|_2} = \sqrt{1+\alpha^2}.$$

From equation (1.124), the condition number of $A(n,\alpha)$ is the same as that of $T_A(n,\alpha)$ and so can be bounded by

$$\kappa(A(n,\alpha)) \geq \alpha^{-n} \sqrt{\frac{1-\alpha^4}{1-\alpha^{2n}}}. \tag{1.126}$$

For a given $L > 1$, the parameter α is set by using equation (1.125) so that $\alpha < 1/L < 1$ (Property 2). Using equation (1.126), we increase n, the size of the matrix $A(n,\alpha)$, in order to have a sufficiently ill-conditioned matrix to comply with Property 1.

We have performed some numerical experiments with these matrices using MATLAB. The machine precision is $\varepsilon = 1.12 \cdot 10^{-16}$. We set $\alpha = 0.98$ and $n = 1500$ with a random unitary matrix U to obtain $A(n,\alpha)$. The condition number of the matrix is: $\kappa(A(n,\alpha)) = 7.28 \cdot 10^{14}$. We should point out that even though the theoretical result was proved for the square root free MGS algorithm, we consider in our experiments the classical implementation that involves the square root calculation. In Table 1.1, we display the numerical experiments. When $L = 1.03$, a few reorthogonalizations are performed and the algorithm is in fact very close to MGS applied to $A(n,\alpha)$. $\|I - Q^TQ\|_2$ is far from machine precision. When $L = 0.99$, the criterion permits all the reorthogonalizations, the algorithm is in fact exactly MGS2 and gives rise to a matrix Q that is orthogonal up to machine precision.

We show how to construct matrices such that the L–criterion with $L > 1$ fails, this strategy permits us to construct matrices such that $L = 1.03$ is not a good

MGS2($L = 1.03$)	$5.44 \cdot 10^{-1}$
MGS2($L = 0.99$)	$4.57 \cdot 10^{-14}$

Table 1.1: $\|I - Q^T Q\|_2$ for Q obtained by MGS2(L) for different values of L applied on $A(n = 1500, \alpha = 0.98)$.

criterion. We have been limited by the size of the matrices used and we conjectured that increasing the size of the matrices would enable us to decrease the value of L . Furthermore we remark that in our experiments, we do not observe the influence of the terms in n and m either in the assumption (1.74) on A or in the final orthogonality given by inequality (1.75).

1.5.3 Lack of robustness of the K–criterion

Assuming that $\sum_{k=1}^{j-1}(r_{kj}^{(1)})^2 + \|a_j^{(1)}\|_2^2 = \|a_j\|_2^2$ (which corresponds to the theorem of Pythagoras if Q_{j-1} has orthogonal columns), we can rewrite the K–criterion as

$$\frac{\sqrt{\sum_{k=1}^{j-1}(r_{kj}^{(1)})^2}}{\|a_j^{(1)}\|_2} \leq \sqrt{K^2 - 1}. \tag{1.127}$$

Formula (1.127) means that the K–criterion compares the 2–norm of the non-diagonal entries $r_{kj}^{(1)}$, $k < j$, to the diagonal entry $\|a_j^{(1)}\|_2$. We recall that the L–criterion consists in comparing the 1–norm of the non-diagonal entries $r_{kj}^{(1)}$, $k < j$, to the diagonal entry $\|a_j^{(1)}\|_2$.
By analogy with inequality (1.115) we call a *diagonally dominant matrix by columns in the* 2*–norm* a matrix A such that for all j ,

$$|a_{jj}| > \sqrt{\sum_{i \neq j} a_{ij}^2}. \tag{1.128}$$

The value $L = 1$ for the L–criterion, which means that the matrix is diagonally dominant by columns, can be related with the value $K = \sqrt{2}$ for the K–criterion, which means that the matrix is diagonally dominant by columns in the 2–norm. Therefore, our point of view is that the K–criterion forced R to be diagonally dominant by columns in the 2–norm whereas the L–criterion forced R to be diagonally dominant by columns.
We also notice that, if the K–criterion is satisfied, we have

$$\begin{aligned}
& \frac{\|a_j\|_2}{\|a_j^{(1)}\|_2} < K \\
\Rightarrow \quad & \frac{\sqrt{\|a_j^{(1)}\|_2^2 + \sum_{k=1}^{j-1} r_{kj}^{(1)2}}}{\|a_j^{(1)}\|_2} < K \\
\Rightarrow \quad & \frac{\sqrt{\sum_{k=1}^{j-1} r_{kj}^{(1)2}}}{\|a_j^{(1)}\|_2} < \sqrt{K^2 - 1} \\
\Rightarrow \quad & \frac{\sum_{k=1}^{j-1} |r_{kj}^{(1)}|}{\|a_j^{(1)}\|_2} < \sqrt{K^2 - 1}
\end{aligned}$$

So if the K–criterion is satisfied with $K = \sqrt{2}$, this implies that

$$\frac{\sum_{k=1}^{j-1} |r_{kj}^{(1)}|}{\|a_j^{(1)}\|_2} < 1$$

and the L–criterion with $L = 1$ is also satisfied. In other words, MGS2($L = 1$) reorthogonalizes more often than MGS2($K = \sqrt{2}$). In term of diagonal dominance, we get that a matrix that is diagonally dominant by columns in 2–norm is diagonally dominant by columns.
We have compared MGS2($K = \sqrt{2}$) and MGS2($L = 1$) on several numerically nonsingular matrices from the Matrix Market and also on the set of matrices of Hoffmann [76]. From our experiments, it appears that the K–criterion with $K = \sqrt{2}$ gives us as good results as the L–criterion with $L = 1$ in term of orthogonality on all these matrices. However, the L–criterion with $L = 1$ may perform a few extra useless reorthogonalizations. Therefore, on these cases, the K–criterion is to be preferred.
In this section, we look for matrices such that the K–criterion performs poorly. A first idea is to simply take the matrix $A(n, \alpha)$, $\alpha < 1$. For those matrices, in exact arithmetic, MGS2(K) does not perform any reorthogonalization for any

$$K \geq \sqrt{1 + (\frac{1}{\alpha})^2}.$$

If we consider $A(n = 1500, \alpha = 0.98)$, MGS2($K = 1.43$) performs no reorthogonalization and therefore reduces to MGS (Cf. Table 1.2). With the $A(n, \alpha)$ matrices,

MGS2($K = 1.43$)	$1.82 \cdot 10^0$

Table 1.2: $\|I - Q^T Q\|_2$ for Q obtained MGS2(K) applied on $A(n = 1500, \alpha = 0.98)$.

the smallest value of K for which MGS2(K) may fail is $K = \sqrt{2}$ which corresponds to diagonally dominant columns in the 2–norm criterion.
However we can find better counter–example matrices by considering the matrices $B(n, \alpha) \in \mathbb{R}^{n \times n}$ such that

$$B(n,\alpha) = UT(n,\alpha) = U \begin{pmatrix} 1 & -\alpha & -\alpha/\sqrt{2} & -\alpha/\sqrt{3} & & -\alpha/\sqrt{n-1} \\ & 1 & -\alpha/\sqrt{2} & -\alpha/\sqrt{3} & & -\alpha/\sqrt{n-1} \\ & & 1 & -\alpha/\sqrt{3} & & -\alpha/\sqrt{n-1} \\ & & & 1 & & \vdots \\ & & & & \ddots & \vdots \\ & & & & & -\alpha/\sqrt{n-1} \\ & & & & & 1 \end{pmatrix}$$

where $U \in \mathbb{R}^{n \times n}$ such that $U^T U = I$.
For $\alpha < 1$, the unit triangular matrix $T(n, \alpha)$ is a *diagonally dominant matrix by columns in the 2–norm* but is not a *diagonally dominant matrix* in the

usual sense. For the reorthogonalization criterion, this means that if we apply MGS2($K \geq \sqrt{1+\alpha^2}$) to $B(n,\alpha)$, no reorthogonalization is performed, whereas for MGS2($L = 1$) nearly all the reorthogonalizations are performed. With $\alpha < 1$ and matrix $B(n,\alpha)$, Property 2 is verified for MGS2($K \geq \sqrt{1+\alpha^2}$).
Moreover for $\alpha < 1$, the numerical experiments show that when n increases, $T(n,\alpha)$ becomes ill-conditioned. Property 1 is also verified. It seems therefore that matrices $B(n,\alpha)$ are good counter-examples for the K–criterion.
The experimental results are in Table 1.3. We run different versions of Gram–Schmidt with reorthogonalization on a set of matrices $B(n,\alpha)$. The experiments are carried out using MATLAB.

(L,K)	$L=0.99 \quad K=1.40$	$L=0.99 \quad K=1.30$	$L=0.99 \quad K=1.17$	$L=0.99 \quad K=1.05$
matrix B	$B(n=400,\alpha=0.97)$	$B(n=500,\alpha=0.82)$	$B(n=1000,\alpha=0.50)$	$B(n=2500,\alpha=0.30)$
$\kappa(B)$	$3.4 \cdot 10^{15}$	$8.6 \cdot 10^{14}$	$1.8 \cdot 10^{13}$	$5.9 \cdot 10^{12}$
MGS2(K)	$7.2 \cdot 10^{-1}$	$1.1 \cdot 10^{0}$	$1.0 \cdot 10^{-2}$	$7.6 \cdot 10^{-3}$
MGS2(L)	$1.5 \cdot 10^{-14}$	$1.9 \cdot 10^{-14}$	$3.5 \cdot 10^{-14}$	$8.0 \cdot 10^{-14}$

Table 1.3: $\|I - Q^T Q\|_2$ for Q obtained with the MGS2(L) and MGS2(K) algorithms applied to four matrices $B(n,\alpha)$.

With $B(n=2500,\alpha=0.30)$, the MGS2($K = 1.05$) algorithm gives a matrix Q that is far from orthogonal. This means that to guarantee good accuracy K has to be set to a value lower than 1.05. We recall that the value $K = 1$ implies that the algorithm reduces to MGS2. By diminishing α and increasing n, we expect that it is possible to exhibit smaller values for K.We notice that the algorithm MGS2($L = 0.99$) behaves well.

1.5.4 What about classical Gram–Schmidt?

The main focus of this paper is the modified Gram–Schmidt algorithm and its selective reorthogonalization variant. A natural question is whether the results extend or not to the classical Gram–Schmidt variant CGS2(L). In [62], the behaviour of CGS2 is analyzed. However, to our knowledge, no study exists either for the $CGS2(K)$ algorithm or for the $CGS2(L)$ algorithm. For that latter variant, we notice that the proof proposed in this paper for $MGS2(L)$ does not apply. Even though the theoretical behaviour is still an open question, we want to present some numerical experiments that tend to indicate that a similar behaviour might exist for $CGS2(L)$. In Table 1.4, we display the orthogonality quality produced by $CGS2(L)$ and $CGS2(K)$ on the same test matrix as used in Tables 1.1 and 1.2. We observe

CGS2($L=1.03$)	$6.67 \cdot 10^{0}$
CGS2($L=0.99$)	$3.56 \cdot 10^{-14}$
CGS2($K=1.43$)	$1.82 \cdot 10^{0}$

Table 1.4: $\|I - Q^T Q\|_2$ for Q obtained by CGS2(L) and CGS2(K) for different values of L and K applied on $A(n=1500,\alpha=0.98)$.

that, on that matrix, CGS2(L) with $L = 1.03$ does not produce an orthogonal ma-

trix while for $L = 0.99$, the computed Q factor is orthogonal to machine precision. Similarly to MGS2(K), CGS2(K) for K slightly larger than $\sqrt{2}$ cannot compute an orthogonal set of vectors.
Similar experiments to those displayed in Table 1.3 are reported in Table 1.5 and similar comments can be made. That is that the CGS2($K = 1.05$) algorithm gives a matrix Q that is far from being orthogonal. This means that, to guarantee good accuracy, K has to be set to a value lower than 1.05. We recall that the value $K = 1$ implies that the algorithm reduces to CGS2. On the other hand, the algorithm CGS2($L = 0.99$) behaves well. This is a clue suggesting that a theoretical analysis might be done to show that CGS2(L) with $L < 1$ generates an orthogonal set of vectors. This latter study might be the focus of future work that would require developping a completely different proof to the one exposed in this paper which does not apply.

(L, K)	$L = 0.99 \quad K = 1.40$	$L = 0.99 \quad K = 1.30$	$L = 0.99 \quad K = 1.17$	$L = 0.99 \quad K = 1.05$
matrix B	$B(n = 400, \alpha = 0.97)$	$B(n = 500, \alpha = 0.82)$	$B(n = 1000, \alpha = 0.50)$	$B(n = 2500, \alpha = 0.30)$
$\kappa(B)$	$3.4 \cdot 10^{15}$	$8.6 \cdot 10^{14}$	$1.8 \cdot 10^{13}$	$5.9 \cdot 10^{12}$
CGS2(K)	$1.6 \cdot 10^{0}$	$1.6 \cdot 10^{0}$	$1.6 \cdot 10^{0}$	$1.6 \cdot 10^{0}$
CGS2(L)	$1.2 \cdot 10^{-14}$	$1.5 \cdot 10^{-14}$	$2.8 \cdot 10^{-14}$	$6.0 \cdot 10^{-14}$

Table 1.5: $\|I - Q^T Q\|_2$ for Q obtained by different CGS algorithms applied to four matrices $B(n, \alpha)$.

Conclusion

In this paper, we give a new reorthogonalization criterion for the modified Gram–Schmidt algorithm with selective reorthogonalization that is referred to as the L–criterion. This criterion depends on a single parameter L. When L is chosen smaller than 1 (e.g. $L = 0.99$), for numerically nonsingular matrices, this criterion is able to realize the compromise between saving useless reorthogonalizations and giving a set of vectors Q orthogonal up to machine precision level. On the other hand if we set $L > 1$, we exhibit some matrices for which the modified Gram–Schmidt algorithm with selective reorthogonalization based on the L–criterion (GS2(L)) performs very badly. The condition $L < 1$ is therefore necessary to ensure the robustness of MGS2(L).
In order to justify the need of a new criterion, we also show counter–example matrices for which a standard criterion, the K–criterion, gives a final set of vectors far from being orthogonal for any value of the parameter K (at least for all $K > 1.05$). On all these counter–example matrices, we have verified the theory and observe that MGS2($L < 1$) behaves well.
Moreover, we have compared the K–criterion with $K = \sqrt{2}$ and the L–criterion with $L = 1$ on a wide class of standard test matrices. It appears that the K–criterion with $K = \sqrt{2}$ works fine in term of orthogonality of the computed set of vectors for all these matrices but it also saves more reorthogonalizations than the L–criterion with $L = 1$. Note that both criterions do save reorthogonalizations on standard test matrices. Therefore in many cases, the K–criterion with $K = \sqrt{2}$ may nevertheless be preferred to the L–criterion with $L = 1$.

Finally, even though no theory exists yet, we give some numerical evidence indicating that a similar analysis might exist for the classical Gram–Schmidt algorithm with selective orthogonalization based on the L–criterion. Furthermore, these numerical experiments show that neither MGS2(K) nor CGS2(K) succeed in generating a set of orthogonal vectors. This also illustrates the lack of robustness of this criterion when implementing a classical Gram–Schmidt algorithm with selective reorthogonalization.

Acknowledgments

The authors would like to thank the referees for their fruitful comments that helped to improve the readibility of the paper.

Below are three small annexes closely related to the section's topic.

Annex A: the Daniel, Gragg, Kaufman and Stewart criterion

Daniel, Gragg, Kaufman and Stewart [34] have shown that classical Gram–Schmidt used with a selective reorthogonalization criterion gives a set of vectors orthogonal up to machine precision. The selective reorthogonalization criterion they used is called the J–criterion and is

$$\frac{\|a_j^{(\ell)}\|_2}{\|a_j^{(\ell-1)}\|_2} + \omega_\ell \frac{\|Q^T a_j^{(\ell)}\|_2}{\|a_j^{(\ell-1)}\|_2} \leq \theta. \tag{1.129}$$

With θ, it is possible to control the level of orthogonality of the constructed Q–factor. The parameter ω_ℓ is set at run–time. In Figure 1.4, we give the level of orthogonality obtained for three matrices: $A(400, 0.97)$, $A(500, 0.82)$ and $A(1000, 0.5)$. These matrices are used in Section 1.5.2.2.

If the term ω is omitted from equation 1.129, then the J–criterion reduces to the K–criterion with $K = \theta$. As we have seen, this latter criterion might fail. The parameter ω_ℓ is needed for the stability of the method. It adapts itself to the level of orthogonality requested (given with θ) and the level of orthogonality of the columns $Q_{\ell-1}$.

Annex B: Link with the work of Abdelmaleck (1971)

Abdelmaleck [2] have shown that for the classical Gram–Schmidt algorithm iterated twice, under the assumption that for all j,

$$(j+2)^{\frac{1}{2}} \|a_j^{(1)}\|_2 / \|a_j^{(2)}\|_2 \leq 1, \tag{1.130}$$

then the level of orthogonality of the computed Q–factor is of the level of the machine precision.

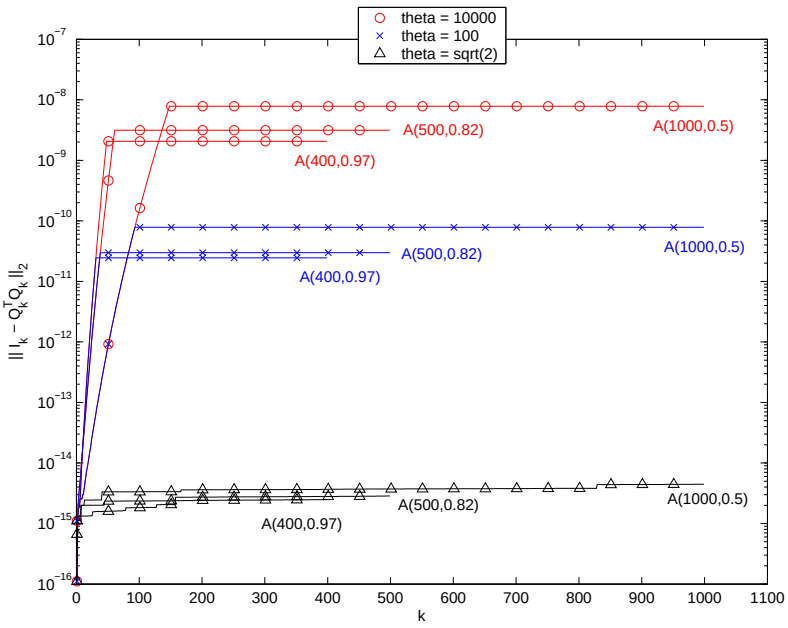


Figure 1.4: The Gram–Schmidt algorithm with reorthogonalization and selective reorthogonalization J–criterion is run on three matrices: $A(400, 0.97)$, $A(500, 0.82)$ and $A(1000, 0.5)$ (see Section 1.5.2.2). For each k, we compute the level of orthogonality of the Q–factor, Q_k. J We also relaxed the parameter θ of the J–criterion to see that it effectively controll well the final level of orthogonality.

With our analysis, we have shown that equation (1.130) is true under numerical nonsingularity assumption of the initial matrix. Moreover the equation (1.130) is verified at the first loop, then the second loop is not needed. Equation (1.130) is nothing but a reorthogonalization criterion. We call this reorthogonalization criterion the I–criterion. Finally note that this criterion –to our knowledge– appears the first time as a criterion in Kiełbasiński [82, 84].

Annex C: experimental comparison of the K–criterion and the L–criterion

In Section 1.5.2.1, we have shown that the L–criterion is able to provide a Q–factor orthogonal to the machine precision level. In Section 1.5.3, we have shown that the K–criterion is unable to provide a Q–factor orthogonal to the machine precision level. In this part, we want to verify whether or not this is a general trend among more *general* matrices.
Following Hoffman [76], the matrices are constructed by multiplying a given diagonal matrix (singular values) from both sides by random orthogonal matrices. The maximum singular value is always equal to 1 and the smallest varies between 0.1 and 10^{-12} so that the condition number κ of the matrices is between 10 and 10^{12}. In Table 1 we show a representative selection of the results of Hoffmann; it shows the typical behaviour of algorithms CGS2(K) and MGS2(K) for various values of the parameter κ. The average number of iterations per column is denoted by ν; the departure from orthogonality is measured in the 1-norm, and given by $\|Q^TQ-I\|_1$. In Table 2 the same experiments are run with CGS2(L) and MGS2(L). All matrices

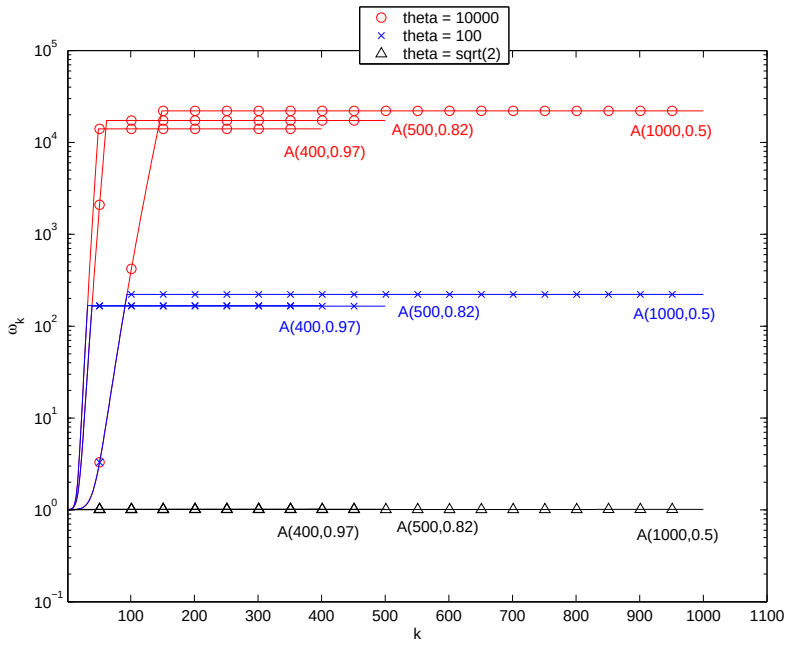


Figure 1.5: The Gram–Schmidt algorithm with reorthogonalization and selective reorthogonalization J–criterion is run on three matrices: $A(400, 0.97)$, $A(500, 0.82)$ and $A(1000, 0.5)$ (see Section 1.5.2.2 and Figure 1.4). For each k, we plot the corresponding ω_k used in the selective reorthogonalization J–criterion (1.129).

used in the selection described in Table 1.6 and 1.7 have $m = 210$ and $n = 100$. Some differences appear with the results of Hoffmann. First of all the precision machine Hoffmann used is $5 \cdot 10^{-14}$ whereas ours is $1.16 \cdot 10^{-16}$. Then we do not have exactly the same matrices. We have tried to recover his distribution of singular values. We know that "*the singular values are distributed equally over the interval* $[\kappa^{-1}, 1]$". As Hoffmann, "*we have observed that the distribution of the singular values within the interval* $[\sigma_{min}, \ldots, \sigma_{max}]$ *is of little importance for the resulting orthogonality of* Q*.*" But the distribution of the singular values influences the number of reorthogonalizations performed. Finally, we mention that we sucessfully reproduce the experiments of Hoffmann by considering a logarithmic distribution of singular values.
Few conclusions can be drawn from Table 1.6 and Table 1.7.

- As Hoffmann, we observe in Table 1.6 that the resulting orthogonality is proportional to K. This is a very interesting observation, despite the counter–example matrices exhibited in the previous section, for these matrices, we can adapt K with a trade–off between ν, the computational cost and $\|I - Q^T Q\|_1$, the level of orthogonality prescribed. In Table 1.7, the same conclusion holds for the L–criterion.

- Even if it is hard to compare the K–criterion and the L–criterion, for well–conditioned matrix, we see that the L–criterion enforces too many reorthogonalizations.

cond. nr.	K	ν(=avg. nr. iter. per col.)		$\|I - Q^TQ\|_1$	
		CGS2(K)	MGS2(K)	CGS2(K)	MGS2(K)
10^1	1.42	1.54	1.54	$1.16 \cdot 10^{-14}$	$1.15 \cdot 10^{-14}$
	2	1.10	1.10	$2.60 \cdot 10^{-14}$	$2.30 \cdot 10^{-14}$
	10^1	1.00	1.00	$3.60 \cdot 10^{-14}$	$2.92 \cdot 10^{-14}$
10^4	1.42	1.94	1.94	$6.25 \cdot 10^{-15}$	$6.62 \cdot 10^{-15}$
	2	1.87	1.87	$1.13 \cdot 10^{-14}$	$1.33 \cdot 10^{-14}$
	10^1	1.64	1.64	$5.55 \cdot 10^{-13}$	$7.35 \cdot 10^{-14}$
	10^2	1.27	1.27	$9.74 \cdot 10^{-11}$	$9.08 \cdot 10^{-13}$
	10^3	1.00	1.00	$6.84 \cdot 10^{-9}$	$6.09 \cdot 10^{-12}$
	10^4	1.00	1.00	$6.84 \cdot 10^{-9}$	$6.09 \cdot 10^{-12}$
10^7	1.42	1.96	1.96	$4.91 \cdot 10^{-15}$	$5.24 \cdot 10^{-15}$
	2	1.92	1.92	$8.00 \cdot 10^{-15}$	$5.91 \cdot 10^{-15}$
	10^1	1.79	1.79	$2.58 \cdot 10^{-13}$	$4.55 \cdot 10^{-14}$
	10^2	1.62	1.62	$9.90 \cdot 10^{-11}$	$5.36 \cdot 10^{-13}$
	10^3	1.46	1.46	$2.11 \cdot 10^{-8}$	$8.92 \cdot 10^{-12}$
	10^4	1.28	1.28	$4.88 \cdot 10^{-6}$	$1.14 \cdot 10^{-10}$
	10^5	1.13	1.13	$7.49 \cdot 10^{-4}$	$9.52 \cdot 10^{-10}$
	10^6	1.00	1.00	$1.83 \cdot 10^{-2}$	$2.27 \cdot 10^{-9}$
	10^7	1.00	1.00	$1.83 \cdot 10^{-2}$	$2.27 \cdot 10^{-9}$
10^{10}	1.42	1.96	1.96	$5.24 \cdot 10^{-15}$	$6.76 \cdot 10^{-15}$
	2	1.95	1.95	$5.77 \cdot 10^{-15}$	$7.10 \cdot 10^{-15}$
	10^1	1.86	1.86	$1.89 \cdot 10^{-13}$	$3.80 \cdot 10^{-14}$
	10^2	1.73	1.73	$3.19 \cdot 10^{-10}$	$7.80 \cdot 10^{-13}$
	10^3	1.63	1.63	$1.60 \cdot 10^{-8}$	$1.35 \cdot 10^{-11}$
	10^4	1.52	1.52	$4.72 \cdot 10^{-6}$	$7.03 \cdot 10^{-11}$
	10^5	1.42	1.42	$5.99 \cdot 10^{-4}$	$4.99 \cdot 10^{-10}$
	10^6	1.15	1.30	$2.16 \cdot 10^{1}$	$1.05 \cdot 10^{-8}$
	10^7	1.00	1.19	$1.89 \cdot 10^{1}$	$6.02 \cdot 10^{-8}$
	10^8	1.00	1.06	$1.89 \cdot 10^{1}$	$1.52 \cdot 10^{-6}$
	10^9	1.00	1.00	$1.89 \cdot 10^{1}$	$2.21 \cdot 10^{-6}$
	10^{10}	1.00	1.00	$1.89 \cdot 10^{1}$	$2.21 \cdot 10^{-6}$

Table 1.6: Average number of reorthogonalization (ν) and orthogonality observed ($\|I - Q^TQ\|_1$) for CGS2(K) and MGS2(K) applied to various matrices $\kappa = 10$ to 10^{10} with various $K = 1.42$ to κ .

cond. nr.	L	ν(=avg. nr. iter. per col.)		$\|I - Q^TQ\|_1$	
		CGS2(L)	MGS2(L)	CGS2(L)	MGS2(L)
10^1	0.99	1.91	1.91	$4.60 \cdot 10^{-15}$	$5.67 \cdot 10^{-15}$
	2	1.82	1.82	$4.88 \cdot 10^{-15}$	$5.58 \cdot 10^{-15}$
	10^1	1.24	1.24	$2.17 \cdot 10^{-14}$	$1.82 \cdot 10^{-14}$
10^4	0.99	1.97	1.97	$4.99 \cdot 10^{-15}$	$5.62 \cdot 10^{-15}$
	2	1.92	1.92	$6.74 \cdot 10^{-15}$	$7.83 \cdot 10^{-15}$
	10^1	1.80	1.80	$1.85 \cdot 10^{-14}$	$2.09 \cdot 10^{-14}$
	10^2	1.51	1.51	$4.02 \cdot 10^{-12}$	$1.83 \cdot 10^{-13}$
	10^3	1.21	1.21	$4.04 \cdot 10^{-10}$	$2.15 \cdot 10^{-12}$
	10^4	1.00	1.00	$6.84 \cdot 10^{-9}$	$6.09 \cdot 10^{-12}$
10^7	0.99	1.98	1.98	$5.89 \cdot 10^{-15}$	$5.39 \cdot 10^{-15}$
	2	1.96	1.96	$4.91 \cdot 10^{-15}$	$5.24 \cdot 10^{-15}$
	10^1	1.89	1.89	$1.34 \cdot 10^{-14}$	$9.18 \cdot 10^{-15}$
	10^2	1.73	1.73	$5.13 \cdot 10^{-12}$	$1.77 \cdot 10^{-13}$
	10^3	1.56	1.56	$4.23 \cdot 10^{-10}$	$1.61 \cdot 10^{-12}$
	10^4	1.39	1.39	$1.48 \cdot 10^{-7}$	$2.04 \cdot 10^{-11}$
	10^5	1.22	1.22	$2.18 \cdot 10^{-5}$	$1.76 \cdot 10^{-10}$
	10^6	1.02	1.02	$7.49 \cdot 10^{-3}$	$2.19 \cdot 10^{-9}$
	10^7	1.00	1.00	$1.83 \cdot 10^{-2}$	$2.27 \cdot 10^{-9}$
10^{10}	0.99	1.98	1.98	$5.43 \cdot 10^{-15}$	$5.80 \cdot 10^{-15}$
	2	1.96	1.96	$5.24 \cdot 10^{-15}$	$6.76 \cdot 10^{-15}$
	10^1	1.90	1.90	$3.22 \cdot 10^{-14}$	$1.69 \cdot 10^{-14}$
	10^2	1.81	1.81	$3.06 \cdot 10^{-12}$	$9.67 \cdot 10^{-14}$
	10^3	1.69	1.69	$1.42 \cdot 10^{-9}$	$2.72 \cdot 10^{-12}$
	10^4	1.56	1.56	$7.08 \cdot 10^{-7}$	$3.38 \cdot 10^{-11}$
	10^5	1.44	1.44	$1.84 \cdot 10^{-4}$	$2.02 \cdot 10^{-10}$
	10^6	1.28	1.34	$1.34 \cdot 10^{1}$	$5.45 \cdot 10^{-9}$
	10^7	1.04	1.24	$1.83 \cdot 10^{1}$	$2.06 \cdot 10^{-8}$
	10^8	1.00	1.13	$1.89 \cdot 10^{1}$	$2.60 \cdot 10^{-7}$
	10^9	1.00	1.00	$1.89 \cdot 10^{1}$	$2.21 \cdot 10^{-6}$
	10^{10}	1.00	1.00	$1.89 \cdot 10^{1}$	$2.21 \cdot 10^{-6}$

Table 1.7: Average number of reorthogonalization (ν) and orthogonality observed ($\|I - Q^TQ\|_1$) for CGS2(L) and MGS2(L) applied to various matrices $\kappa = 10$ to 10^{10} with various $L = 0.99$ to κ.

- As Hoffmann, we also observe that MGS works better than CGS.
- In Section 1.6.2.2.3, the K–criterion is used with $K = 2$ and $K = \sqrt{2}$ on matrices arising from the Arnoldi process. We observe that the K–criterion with $K = 2$ fails on these matrices while the K–criterion with $K = \sqrt{2}$ does not fail.

1.6 A reorthogonalization procedure for the modified Gram–Schmidt algorithm based on a rank k update

The title as well as the contents of this section corresponds to the following technical report: CERFACS Technical Report (2003) – TR/PA/03/11
joint work with Luc Giraud and Serge Gratton.

Abstract

The modified Gram–Schmidt algorithm is a well–known and widely used procedure to orthogonalize the column vectors of a given matrix. When applied to ill–conditioned matrices in floating point arithmetic, the orthogonality among the computed vectors may be lost. In this work, we propose an *a posteriori* reorthogonalization technique based on a rank– k update of the computed vectors. The level of orthogonality of the set of vectors built gets better when k increases and finally reaches the machine precision level for a large enough k. The rank of the update can be tuned in advance to monitor the orthogonality quality. We illustrate the efficiency of this approach in the framework of the seed–GMRES technique for the solution of an unsymmetric linear system with multiple right–hand sides. In particular, we report experiments on numerical simulations in electromagnetic applications where a rank–one update is sufficient to recover a set of vectors orthogonal to machine precision level.

Introduction

Let A be an $m \times n$ real matrix, $m \geq n$ of full rank n. In exact arithmetic, the Modified Gram–Schmidt algorithm (MGS) computes an $m \times n$ matrix Q with orthonormal columns and an $n \times n$ upper triangular matrix R such that $A = QR$. The framework of this paper is the study of the MGS algorithm in the presence of rounding errors. We call *computed quantities* quantities that are computed using a well–designed floating point arithmetic [13]. We denote by $\bar{Q}$ and $\bar{R}$ the computed quantities obtained by running MGS in the presence of rounding errors.

In [15], Björck and Paige show that $\bar{R}$ is as good as the triangular factor obtained using backward stable transformations such as Givens rotations or Householder reflections. This property of MGS explains why this algorithm can be safely used in applications where only the factor $\bar{R}$ is needed. This is namely the case in the solution of linear least squares problems where the R–factor of the QR–factorization of $[A, b]$ is needed [13, 15]. Another important feature of MGS is that the number of operations required to explicitly compute the Q–factor (problem known as the orthogonal basis problem) is approximatively half that of the methods using Givens rotations or Householder reflections [63, p. 232]. However the computed factor $\bar{Q}$ has less satisfactory properties, since for an ill–conditioned matrix A, it may exhibit a very poor orthogonality as measured by the quantity $\|\bar{Q}^T\bar{Q} - I_n\|$, where $\|.\|$ denotes the spectral 2–norm, and I_n the identity matrix of order n [109]. This has stimulated much work on various modifications of MGS that aim at enhancing the orthogonality of $\bar{Q}$ at a low computational cost. One of these strategies consists in performing reorthogonalizations during the algorithm when a prescribed criterion is satisfied. This has given rise to the family of iterated modified Gram–Schmidt

algorithms, which differ in the criterion they use to enforce the reorthogonalization (see e.g. [34, 76, 113]). An alternative way to compensate for the lack of orthogonality in $\bar{Q}$ is derived in [15] for a wide class of problems, including the linear least squares problem and computation of the minimum 2–norm solution of an underdetermined linear system and the projection of a vector onto a subspace. A careful use of $\bar{Q}$ and $\bar{R}$, based on an equivalence of MGS on A and Householder QR on an augmented matrix obtained by putting a matrix of zeros on top of A, leads to a backward stable algorithm. Such a strategy implies – in general – that the use of $\bar{Q}$ is computationally more expensive than would be the use of a Q–factor with orthonormal columns.
The error analyses related to the loss of orthogonality, that are used to derive the successful methods mentioned above, are based on the study of the quantity $\|\bar{Q}^T\bar{Q} - I_n\|$. We propose here to adopt a different approach by inspecting not only the largest singular value, as actually done in the related literature, but each singular value of the matrices involved in MGS. We denote by σ_i, $i = 1, \ldots n$ the singular values of A, $\sigma_1 \geq \cdots \geq \sigma_n > 0$, by $\kappa = \sigma_1/\sigma_n$ the spectral condition number of A. Let κ_i, the reduced condition number, be defined by $\kappa_i = \sigma_1/\sigma_{n-i+1}, i = 1 \ldots n$. Finally let $\widetilde{Q}$ be the matrix obtained from $\bar{Q}$ by normalizing its columns. In this paper, we exhibit a series of low rank matrices F_k, $k = 0, \ldots, n-1$ that enables us to update the factor $\widetilde{Q}$ such that

- $\mathrm{rank}(F_k) \leq k$,
- the columns of $\widetilde{Q} + F_k$ are orthonormal up to machine precision times κ_k, if $k = n - 1$, then the columns of $\widetilde{Q} + F_{n-1}$ are exactly orthonormal,
- $(\widetilde{Q} + F_k)\bar{R}$ represents A up to machine precision.

In the case $k = 0$, $F_0 = 0$ so $(\widetilde{Q} + F_0) = \widetilde{Q}$ and the results obtained are of the same essence as the ones by Björck (1967) [13]. Namely MGS generates a Q–factor such that the columns of $\widetilde{Q}$ are orthonormal up to machine precision times $\kappa = \kappa_0$ and $\widetilde{Q}\bar{R}$ represents A up to machine precision. In the case $k = n - 1$, $(\widetilde{Q} + F_{n-1})$ is indeed the same matrix as $\hat{Q}$, the matrix exhibited by Björck and Paige [15]. That is $\hat{Q}$ has orthonormal columns and $\hat{Q}\bar{R}$ represents A up to machine precision. Our result can be seen as a theoretical bridge that links the result of Björck (1967) [13] to the result of Björck and Paige (1992) [15]. An algorithm to compute F_k, $k = 0, \ldots, n-1$, is also derived. In our experiments this algorithm behaves well in the presence of rounding errors. For example when κ_k is close to one, the update of $\bar{Q}$ with F_k produces a Q–factor with columns orthonormal up to machine precision. The complexity of this algorithm increases with k. For small k, its complexity is competitive with other standard reorthogonalization techniques. We conclude our study with an application of this algorithm in the framework of the solution of unsymmetric linear systems with multiple right–hand sides where a seed–variant of GMRES can be successfully used (we refer to Section 2.5 for a detailed description of this solver).
In the remainder of this paper, for any $m \times n$ matrix X, we denote by $\sigma_i(X)$, $i = 1, \ldots n$ the singular values of X ordered such that $\sigma_1(X) \geq \cdots \geq \sigma_n(X)$. We

note that the work of this paper can be extended to complex arithmetic as well.

1.6.1 Rank considerations related to the loss of orthogonality in MGS

1.6.1.1 Introduction

A rigorous measure of the orthogonality of an $m \times n$ matrix $\bar{Q}$ can be defined to be the distance, in the spectral 2–norm, to the set $\mathcal{O}(m,n)$ of $m \times n$ matrices with orthonormal columns

$$\min_{V \in \mathcal{O}(m,n)} \|\bar{Q} - V\|.$$

Fan and Hoffman in [48] for the case $m = n$, and Higham in [73] for the general case $n \leq m$ proved that the minimum is attained for U being the unitary polar factor of $\bar{Q}$. The easily computed quantity $\|I_n - \bar{Q}^T\bar{Q}\|$ is often preferred to measure the orthogonality, because, as shown in Lemma 1.6.1, it has the same order of magnitude as $\min_{V \in \mathcal{O}(m,n)} \|\bar{Q} - V\|$ when $\|\bar{Q}\|$ is close to one.

Lemma 1.6.1 *[73] Let $\bar{Q} \in \mathbb{R}^{m \times n}$, $n \leq m$,*

$$\frac{\|I_n - \bar{Q}^T\bar{Q}\|}{1 + \|\bar{Q}\|} \leq \min_{V \in \mathcal{O}(m,n)} \|\bar{Q} - V\| \leq \|I_n - \bar{Q}^T\bar{Q}\|.$$

Lemma 1.6.1 can be easily generalized into Lemma 1.6.2.

Lemma 1.6.2 *Let $\bar{Q} \in \mathbb{R}^{m \times n}$, $n \leq m$,*

$$\frac{\sigma_i(\bar{Q}^T\bar{Q} - I_n)}{1 + \|\bar{Q}\|} \leq \sigma_i(\bar{Q} - U) \leq \sigma_i(\bar{Q}^T\bar{Q} - I_n),$$

where $i = 1, \ldots n$ and U is the unitary polar factor associated with $\bar{Q}$

Proof of Lemma 1.6.2 Let $\bar{Q} = UH$ be the polar decomposition of $\bar{Q}$. $U \in \mathbb{R}^{m \times n}$ has orthonormal columns and $H \in \mathbb{R}^{n \times n}$ symmetric positive definite. From $U^TU = I_n$ and $H^T = H$, it follows that

$$(\bar{Q} - U)^T(\bar{Q} + U) = \bar{Q}^T\bar{Q} - U^T\bar{Q} + \bar{Q}^TU - U^TU = \bar{Q}^T\bar{Q} - I_n. \tag{1.131}$$

We also have $\bar{Q} + U = U(I_n + H)$ and H being symmetric positive definite, $I_n + H$ has full rank n, and

$$\begin{aligned}
&\bar{Q}^T\bar{Q} - I_n = H^2 - I_n = (H - I_n)(H + I_n),\\
&\bar{Q} - U = U(H - I_n) = U(\bar{Q}^T\bar{Q} - I_n)(H + I_n)^{-1},\\
&\sigma_i(\bar{Q} - U) \leq \sigma_i(\bar{Q}^T\bar{Q} - I_n)\|(H + I_n)^{-1}\| \leq \sigma_i(\bar{Q}^T\bar{Q} - I_n)\\
&\leq \sigma_i(H - I_n)\|H + I_n\| = \sigma_i(\bar{Q} - U)\|H + I_n\| \leq \sigma_i(\bar{Q} - U)(1 + \|\bar{Q}\|).
\end{aligned}$$

♡

An important consequence of Lemma 1.6.2 is that if $\bar{Q}$ has not orthonormal columns, but if $\bar{Q}^T\bar{Q} - I_n$ has only k nonzero singular values, $\bar{Q}$ is at most a rank–k modification of a matrix with orthonormal columns (namely U).

In Section 1.6.1.3, we derive a result for MGS that is similar in essence to Lemma 1.6.2. However, for any $k \leq n$, the MGS context will enable us to find explicitly a rank–k matrix F_k such that $\bar{Q} + F_k$ has an improved orthogonality compared with $\bar{Q}$ and such that the product $(\bar{Q} + F_k)\bar{R}$ still accurately represents A.

1.6.1.2 Some useful background related to MGS in floating point arithmetic

A key result to understand the loss of orthogonality in MGS in floating point arithmetic, is that MGS on A can be interpreted as an Householder QR–factorization on $A_{aug} = \begin{bmatrix} O_n \\ A \end{bmatrix}$, where O_n is the square zero matrix of order n [15]. Since we elaborate our work on results and techniques presented in [15] we briefly outline them below.
The use of Wilkinson's analysis of Householder transformations [137, pp. 153–162] on A_{aug} enables Björck and Paige [15, Eq.(3.3)] to give an orthogonal transformation $\widetilde{P}$ such that

$$\begin{pmatrix} E_1 \\ A + E_2 \end{pmatrix} = \widetilde{P} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix} = \begin{pmatrix} \widetilde{P}_{11} \\ \widetilde{P}_{21} \end{pmatrix} \bar{R}, \qquad \|E_i\| \leq \bar{c}_i u \|A\|, \quad i = 1, 2, \tag{1.132}$$

where $\bar{c}_i$ are constant depending on m, n and the detail of the arithmetic, and u is the unit roundoff. Here $\widetilde{P}_{11}$ is strictly upper triangular, see [15, §4 & (4.1)].
Let $\widetilde{Q} = [\widetilde{q}_1, \ldots, \widetilde{q}_n]$ be the matrix obtained from $\bar{Q} = [\bar{q}_1, \ldots, \bar{q}_n]$ by normalizing its columns ($\widetilde{q}_i = \bar{q}_i / \|\bar{q}_i\|$). The equality $\widetilde{P}_{21} = \widetilde{Q}(I_n - \widetilde{P}_{11})$ holds [15, Eq.(4.5)] and the residual error of the polar factor $\hat{Q}$ of $\widetilde{P}_{21}$ can be bounded as follows,

$$\|A - \hat{Q}\bar{R}\| \leq \bar{c} u \|A\|, \tag{1.133}$$

where $\bar{c} = \bar{c}_1 + \bar{c}_2$, provided that $\bar{c} u \kappa < 1$ [15, Eq.(3.7)]. Finally, let $\bar{\sigma}_1 \geq \cdots \geq \bar{\sigma}_n$ be the singular values of $\bar{R}$. The singular values of $\bar{R}$ approximate those of A in the following sense $|\bar{\sigma}_i - \sigma_i| \leq \bar{c} u \sigma_1$ [15, Eq.(3.8)]. This implies that under the assumption $\bar{c} u \kappa < 1$, $\bar{R}$ has full rank n.

1.6.1.3 Recapture of the orthogonality in MGS

Since $\begin{pmatrix} \widetilde{P}_{11} \\ \tilde{P}_{21} \end{pmatrix}$ has orthonormal columns and $n \leq m$, we consider its CS decomposition [63, p. 77] defined by

$$\begin{aligned} \tilde{P}_{11} &= UCW^T, \\ \tilde{P}_{21} &= VSW^T, \end{aligned} \tag{1.134}$$

where C is singular since $\tilde{P}_{11}$ is strictly upper triangular, the entries of S are in increasing order ($0 \leq s_1 \leq .. \leq s_n = 1$), the entries of C are in decreasing order ($1 \geq c_1 \geq .. \geq c_n = 0$) and $C^2 + S^2 = I_n$. The three matrices U , V , W have orthonormal columns. C , S , U and W are $n \times n$, V is $m \times n$. Similarly as in

[15], we suppose that A is not too ill–conditioned, by assuming that $(\bar{c}_1+\bar{c})u\kappa < 1$ or equivalently (since this implies both $\bar{c}u\kappa < 1$ and $\bar{c}_1 u\kappa < 1-\bar{c}u\kappa$)

$$\bar{c}_1 u\eta\kappa < 1, \tag{1.135}$$

where $\eta = (1-\bar{c}u\kappa)^{-1}$. This has the following consequence. Since the leading element of C is (using (1.132)) $c_1 = \|\tilde{P}_{11}\| = \|E_1\bar{R}^{-1}\| \leq \bar{c}_1 u\sigma_1/\bar{\sigma}_n$, and since from (1.133) $|\sigma_n - \bar{\sigma}_n| \leq \bar{c}u\sigma_1$, we see that $\bar{\sigma}_n \geq \sigma_n - \leq \bar{c}u\sigma_1 = \sigma_n\eta$, and it follows $c_1 \leq \bar{c}_1 u\eta\kappa < 1$ (see [15, Eq.(3.11)]), all the s_i's are non zero, thus S has full rank n.

Our goal is to improve the orthogonality of the Q–factor while maintaining the residual error, $\|A - Q\bar{R}\|/\|A\|$, at the level of the machine precision. Since $\hat{Q}$ has orthonormal columns and (1.133) holds, $\hat{Q}$ answers our question. Therefore a straightforward but expensive way to achieve our goal would be to compute $\hat{Q}$ with $\hat{Q} = VW^T$ ([63, p. 149]). Let us evaluate $F = \hat{Q} - \widetilde{Q}$ to find matrices that approximate the difference between $\hat{Q}$ and $\widetilde{Q}$ at low computational cost. Since $\hat{Q} = VW^T$, using $\widetilde{P}_{21} = \widetilde{Q}(I_n - \widetilde{P}_{11})$ (see Section 1.6.1.3) and the CS decomposition (1.134), we get

$$\begin{aligned} F &= \widetilde{Q}\left((I_n - \widetilde{P}_{11})WS^{-1}(I_n - S)W^T - \widetilde{P}_{11}\right), \\ &= \widetilde{Q}\left(W(S^{-1} - I_n) - UCS^{-1}\right)W^T. \end{aligned} \tag{1.136}$$

We define the truncated matrices U_k, V_k and W_k by retaining the first k columns in their counterparts U, V and W. In Matlab style notation, it reads $U_k = U(:, 1:k)$. We also denote by C_k (resp. S_k) the diagonal matrix of order k whose diagonal entries are the $c_i, i = 1\ldots k$ (resp. $s_i, i = 1\ldots k$).

We define the matrix F_k obtained by setting the c_l and the s_l, $l > k$, to zero and one respectively in (1.136), this gives

$$F_k = \widetilde{Q}(W_k(S_k^{-1} - I_k) - U_k C_k S_k^{-1})W_k^T, \tag{1.137}$$

so that $F_0 = 0$, $F_{n-1} = F_n = F$, since $s_n = 1$ and $c_n = 0$. The matrix $\widetilde{Q} + F$ has orthonormal columns and accurately represents A when multiplied on the right by $\bar{R}$. Theorem 1.6.3 shows how these properties are modified when the matrix $\widetilde{Q} + F_k$ is considered instead. The matrices Q_k are then a sequence of matrices going from the matrix of normalized vectors from MGS $Q_0 = \widetilde{Q}$, to the matrix of orthogonal vectors $Q_{n-1} = \hat{Q}$.

Theorem 1.6.3 *Assume that $\bar{c}_1 u\eta\kappa < 1$, for $k = 0, \ldots, n-1$, the matrix Q_k defined by*

$$Q_k = \widetilde{Q} + F_k \tag{1.138}$$

enjoys the following properties

a)

$$rank(Q_k - \widetilde{Q}) \leq k,$$

b)

$$\|A - Q_k\bar{R}\| \leq \left[\bar{c}_2 + \bar{c}_1 \frac{2 - \bar{c}_1 u\eta\kappa}{(1 - \bar{c}_1 u\eta\kappa)^2}\right] u\|A\|,$$

c) for $k = 0, \ldots, n-2$,

$$\|\hat{Q} - Q_k\| \leq \frac{\bar{c}_1 u\eta}{(1 - \bar{c}_1 u\eta\kappa)^2}\kappa_{k+1}$$

for $k = n-1$, $Q_{n-1} = \hat{Q}$

d) for $k = 0, \ldots, n-2$,

$$\|I_n - Q_k^T Q_k\| \leq \|I_n - Q_k^T Q_k\| \leq \left[2 + \frac{\bar{c}_1 u\eta}{(1 - \bar{c}_1 u\eta\kappa)^2}\kappa_{k+1}\right] \frac{\bar{c}_1 u\eta}{(1 - \bar{c}_1 u\eta\kappa)^2}\kappa_{k+1} \leq \frac{2\bar{c}_1 u\eta}{(1 - \bar{c}_1 u\eta\kappa)^4}\kappa_{k+1},$$

for $k = n-1$, $Q_{n-1} = \hat{Q}$, *and so* $\|I_n - Q_{n-1}^T Q_{n-1}\| = 0$.

Proof of Theorem 1.6.3 Part a) is a consequence of the definition (1.137) of F_k. We then establish part b) of this theorem. From (1.132), $\widetilde{P}_{11}\bar{R} = E_1$, and multiplying to the left by U_k^T implies that $U_k^T U C W^T \bar{R} = U_k^T E_1$. Using the definition of the truncated matrices C_k and W_k, one gets $C_k W_k^T \bar{R} = U_k^T E_1$, and, taking norms, $\|C_k W_k^T \bar{R}\| = \|U_k^T E_1\| \leq \|E_1\|$. From (1.132), $\|E_1\| \leq \bar{c}_1 u\|A\|$, we obtain a first intermediate result

$$\|C_k W_k^T \bar{R}\| \leq \bar{c}_1 u\|A\|. \tag{1.139}$$

Let us bound the residual error $\|A - Q_k\bar{R}\|$. Using the triangular inequality, yields

$$\|A - Q_k\bar{R}\| \leq \|A - \widetilde{Q}\bar{R}\| + \|F_k\bar{R}\|. \tag{1.140}$$

The first term of the right–hand side can be bounded using Lemma 1.6.5 (see Appendix). We study the second term of the right–hand side: $\|F_k\bar{R}\|$. By definition (1.137) of F_k,

$$F_k\bar{R} = \widetilde{Q}(W_k(S_k^{-1} - I_k) - U_k C_k S_k^{-1})(W_k^T\bar{R}).$$

Let us use the facts that $C_k W_k^T \bar{R} = U_k^T E_1$ and $C^2 = I - S^2 = (I - S)(I + S)$, $(I - S) = C^2(I + S)^{-1}$ to give

$$\begin{aligned}
F_k\bar{R} &= \widetilde{Q}[W_k(S_k^{-1} - I_k) - U_k S_k^{-1} C_k] W_k^T \bar{R} \\
&= \widetilde{Q}[W_k C_k^2 (I_k + S_k)^{-1} S_k^{-1} - U_k S_k^{-1} C_k] W_k^T \bar{R} \\
&= \widetilde{Q}[W_k C_k (I_k + S_k)^{-1} S_k^{-1} - U_k S_k^{-1}] U_k^T E_1, \\
\|F_k\bar{R}\| &\leq \|\widetilde{Q}\|[\|C_k(I_k + S_k)^{-1} S_k^{-1}\| + \|S_k^{-1}\|]\bar{c}_1 u\|A\|.
\end{aligned}$$

But with $c = c_1$ and $s = s_1$ we can see from the ordering of the c_i and s_i that

$$\|C_k(I_k + S_k)^{-1} S_k^{-1}\| + \|S_k^{-1}\| = \frac{c}{s^2 + s} + \frac{1}{s} = \frac{c + s + 1}{s^2 + s} \cdot \frac{1 - c}{1 - c} = \frac{s^2 + s - cs}{(s^2 + s)(1 - c)} \leq \frac{1}{1 - c},$$

which, with the Lemma 1.6.4 gives

$$\|F_k\bar{R}\| \leq \frac{1}{(1 - \bar{c}_1 u\eta\kappa)^2}\bar{c}_1 u\|A\|.$$

The result (b) follows from (1.140) and the Lemma 1.6.5.

We now prove part c) of the Theorem. We define the matrices $U_{\bar{k}}$, $V_{\bar{k}}$, $W_{\bar{k}}$, so that $U = [U_k, U_{\bar{k}}]$, and similarly for V and W. In Matlab style notation, $U_{\bar{k}} = U(:, k+1:n)$. We also define the matrices $C_{\bar{k}}$ (resp. $S_{\bar{k}}$) the diagonal matrix of order $n-k+1$ whose diagonal elements are the c_i, $i = k+1, \ldots n$ (resp. s_i $i = k+1, \ldots n$). One has

$$\begin{aligned} \hat{Q} - Q_k &= F - F_k, && (1.141)\\ \hat{Q} - Q_k &= \widetilde{Q}\left(W_{\bar{k}}(S_{\bar{k}}^{-1} - I_{n-k+1}) - U_{\bar{k}}C_{\bar{k}}S_{\bar{k}}^{-1}\right)W_{\bar{k}}^T, && (1.142)\\ \|\hat{Q} - Q_k\| &\leq \|\widetilde{Q}\|\left(\|S_{\bar{k}}^{-1} - I_{n-k+1}\| + \|C_{\bar{k}}S_{\bar{k}}^{-1}\|\right). && (1.143) \end{aligned}$$

Since both the s_i's and c_i's belong to $[0,1]$, and the c_i (resp. the s_i) are sorted in decreasing (resp. increasing) order, one obtains

$$\|\hat{Q} - Q_k\| \leq \|\widetilde{Q}\|\left((s_{k+1}^{-1} - 1) + s_{k+1}^{-1}c_{k+1}\right).$$

But we can write

$$s_{k+1}^{-1} - 1 + s_{k+1}^{-1}c_{k+1} = \frac{1 - s_{k+1} + c_{k+1}}{s_{k+1}} \cdot \frac{1 - c_{k+1}}{1 - c_{k+1}} = \frac{s_{k+1}^2 - s_{k+1} + c_{k+1}s_{k+1}}{s_{k+1}(1 - c_{k+1})} = \frac{c_{k+1} + s_{k+1} - 1}{(1 - c_{k+1})} \leq \frac{c_{k+1}}{1 - c_{k+1}},$$

so

$$\|\hat{Q} - Q_k\| \leq \|\widetilde{Q}\|\ c_{k+1}\frac{1}{1 - c_{k+1}}.$$

From Lemma 1.6.4 and using the fact that $c_{k+1} \leq c_1 \leq \bar{c}_1 u\eta\kappa$, we get

$$\|\hat{Q} - Q_k\| \leq \frac{(1}{(1 - \bar{c}_1 u\eta\kappa)^2}c_{k+1}. \qquad (1.144)$$

Remember $\hat{Q}$ has orthonormal columns. The result for $k = n-1$ follows from $Q_{n-1} = \hat{Q}$ in part (c). For the unitary polar factor U_k of Q_k we see from part (c) that

$$\|U_k - Q_k\| \leq \|\hat{Q} - Q_k\| \leq \frac{\bar{c}_1 u\eta}{(1-\gamma)^2}\kappa_{k+1} = \delta \text{ say.}$$

Therefore from Lemma 1.6.2 we see that

$$\|I_n - Q_k^T Q_k\| \leq (1 + \|Q_k\|)\delta.$$

But

$$\|Q_k\| = \|\hat{Q} + Q_k - \hat{Q}\| \leq 1 + \|Q_k - \hat{Q}\| \leq 1 + \delta,$$

so $\|I_n - Q_k^T Q_k\| \leq (2+\delta)\delta$, proving the first inequality in (d). Now $\bar{c}_1 u\eta\kappa_{k+1} \leq \bar{c}_1 u\eta\kappa = \bar{c}_1 u\eta\kappa$, so

$$(2+\delta)\delta \leq \bar{c}_1 u\eta\kappa_{k+1}[2/(1 - \bar{c}_1 u\eta\kappa)^2 + \bar{c}_1 u\eta\kappa/(1 - \bar{c}_1 u\eta\kappa)^4],$$

and the second inequality in (d) follows by using

$\frac{2}{(1-\bar{c}_1 u\eta\kappa)^2} + \frac{\bar{c}_1 u\eta\kappa}{(1-\bar{c}_1 u\eta\kappa)^4} = \frac{2(1-\bar{c}_1 u\eta\kappa)^2 + \bar{c}_1 u\eta\kappa}{(1-\bar{c}_1 u\eta\kappa)^4} = \frac{2 - 3\bar{c}_1 u\eta\kappa + 2\bar{c}_1 u\eta\kappa^2}{(1-\bar{c}_1 u\eta\kappa)^4} \leq \frac{2}{(1-\bar{c}_1 u\eta\kappa)^4}$. ♡

Several remarks can be made. First consistency, $\|A - Q_k\bar{R}\|/\|A\|$, is maintained close to machine precision independently of the rank–k of the update. In the

Theorem 1.6.3 Part b) $k=0$	Lemma 1.6.5 derived from Björck and Paige (1992) [15]
$\|A-\widetilde{Q}\bar{R}\| \leq \left(\bar{c}_2 + 2\bar{c}_1 \frac{(1+\bar{c}_1 u\eta\kappa)}{(1-\bar{c}_1 u\eta\kappa)^2}\right) u\|A\|$	$\|A-\widetilde{Q}\bar{R}\| \leq \left(\bar{c}_2 + \bar{c}_1 \frac{1+\bar{c}_1 u\eta\kappa}{1-\bar{c}_1 u\eta\kappa}\right) u\|A\|$
Theorem 1.6.3 Part b) $k=n-1$	Björck and Paige (1992) [15, Eq.(3.7)]
$\|A-\hat{Q}\bar{R}\| \leq \left(\bar{c}_2 + 2\bar{c}_1 \frac{(1+\bar{c}_1 u\eta\kappa)}{(1-\bar{c}_1 u\eta\kappa)^2}\right) u\|A\|$	$\|A-\hat{Q}\bar{R}\| \leq (c_1+c_2)u\|A\|$
Theorem 1.6.3 Part c) $k=0$	Björck and Paige (1992) [15, Eq.(5.3)]
$\|I_n-\widetilde{Q}^T\widetilde{Q}\| \leq \left(2\bar{c}_1\eta \frac{(1+\bar{c}_1 u\eta\kappa)^2}{(1-\bar{c}_1 u\eta\kappa)^3}\right) u\kappa$	$\|I_n-\widetilde{Q}^T\widetilde{Q}\| \leq \frac{2c_1}{1-(c+c_1)u\kappa} u\kappa$
Theorem 1.6.3 Part c) $k=n-1$	Björck and Paige (1992) [15, Eq.(3.7)]
$\|I_n-\hat{Q}^T\hat{Q}\| = 0$	$\|I_n-\hat{Q}^T\hat{Q}\| = 0$

Table 1.8: Correspondence between the bounds in Theorem 1.6.3 and the results of Björck and Paige[15].

Introduction, we explain that in the case $k=0$ and $k=n-1$, we recover the result of Björck [13] for $\widetilde{Q}=Q_0$ and Björck and Paige [15] for $\hat{Q}=Q_{n-1}$ respectively. A consequence of this unified framework is that the bounds given are larger than the original ones but remain very close. In Table 1.8, we summarize the relations to be compared. Note that the results of Björck [13] have been replaced by analogous results of Björck and Paige [15] in order to compare the same quantities.

1.6.2 Numerical illustrations and examples of applications

1.6.2.1 Numerical illustrations of the bounds in Theorem 1.6.3

The aims of this section are twofold. First, we give an algorithm to compute the approximations $\bar{F}_k$ (resp. $\bar{Q}_k$) of the matrices F_k (resp. Q_k), then we numerically verify that Theorem 1.6.3 is satisfied with these computed quantities up to machine precision.
In order to make sure that the rank–k property of the $m\times n$ matrix F_k is inherited by the computed matrix $\bar{F}_k$, we define $\bar{F}_k$ as the product of the $m\times k$ computed quantities $\bar{Q}(W_k(S_k^{-1}-I_k)-U_kC_kS_k^{-1})$ times the $k\times n$ rectangular matrix W_k^T. Then by construction, the first statement a) of Theorem 1.6.3 is satisfied and we can now focus on the last two statements and show that the bounds are sharp.
In the following, the notation F_k (resp. Q_k) stands for the the computed quantity $\bar{F}_k$ (resp. $\bar{Q}_k$). For the experiments, we proceed as follows. Starting from an initial matrix A, we run MGS to obtain $\bar{Q}$ and $\bar{R}$. Then for each k from $k=0$ to $n-1$, we compute the associated matrix Q_k using formulae (1.137) and (1.138). In that respect, we need to compute $\widetilde{P}_{11}$. In [15, Eq.(4.1)], Björck and Paige show that $\widetilde{P}_{11}$ is strictly upper triangular with element (i,j) equal to $\tilde{q}_i^T(I_m-\tilde{q}_1\tilde{q}_1^T)\ldots(I_m-\tilde{q}_{j-1}\tilde{q}_{j-1}^T)\tilde{q}_j$ for $i<j$. We define $\widetilde{T}$ such that $\widetilde{T}$ is strictly upper triangular with element (i,j), $\tilde{q}_i^T\tilde{q}_j$, $(i<j)$. Since $\|\tilde{q}_i\|=1$ for all i, one may notice that $(I_n+\widetilde{T})(I_n-\widetilde{P}_{11})=I_n$, that can also be written

$$\widetilde{P}_{11}=(I_n+\widetilde{T})^{-1}\widetilde{T}. \tag{1.145}$$

Note that in practice the mathematical quantities $\tilde{q}_i$ are replaced by the computed

1. run MGS on A to obtain $\bar{Q}$ and $\bar{R}$
2. compute $\bar{T}$, the strictly upper triangular matrix with entry (i,j), $\bar{q}_i^T \bar{q}_j$, $(i<j)$ then form $\bar{P}_{11} = (I_n + \bar{T})^{-1}\bar{T}$
3. compute the k largest singular values of $\bar{P}_{11}$, c_i, $i = 1,\ldots,n$, and the associated k right (resp. left) singular vectors U_k (resp. W_k) finally form $s_i = \sqrt{1-c_i^2}$, $i = 1,\ldots,k$. The matrix C_k (resp. S_k) is the $k \times k$ diagonal matrix with entry (i,i) equal to c_i (resp. s_i).
4. Form $Q_k = \bar{Q} + \bar{Q}(W_k(S_k^{-1} - I_k) - U_k C_k S_k^{-1})W_k^T$

Table 1.9: Algorithm 1 : MGS with an a–posteriori reorthogonalization by a rank–k update

quantities $\bar{q}_i$. Equation (1.145) is preferred to the original equation of Björck and Paige [15, Eq.(4.1)] since it enables us to compute $\widetilde{P}_{11}$ with significantly less flops when m is large compared to n. We summarize the corresponding algorithm in Table 1.9.

In this section, the numerical experiments are run with MATLAB 6 where the unit roundoff is $u = 1.1 \cdot 10^{-16}$. We consider two test matrices, that are the matrices $P(1500, 500, 1, 5)$ from Paige and Saunders [99] and GRE_216B from the Matrix Market[7]. The first one is a 1500×500 matrix with condition number 10^{16} while the latter is a 216×216 matrix with condition number $6 \cdot 10^{14}$. On those two matrices we investigate how sharp the bounds b) and c) in Theorem 1.6.3 are.

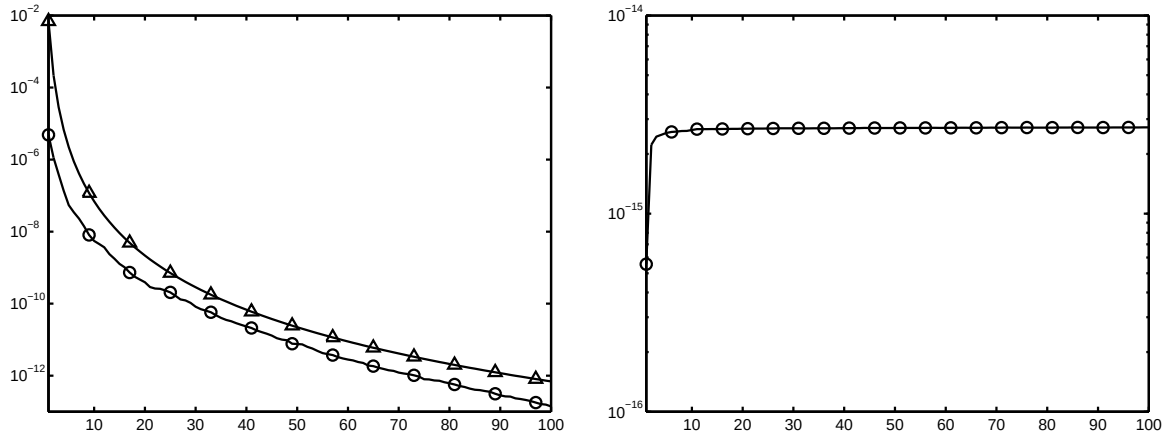
In order to quantify the orthogonality quality of the columns of different matrices, we define the level of orthogonality of Q as the quantity $\|I_n - Q^T Q\|$. In Figure 1.6 a), we plot the "recovered orthogonality" with $\circ$. For $k = 0$, we have $Q_0 = \bar{Q}$ therefore we simply plot the level of orthogonality obtained after the run of MGS on $P(1500, 500, 1, 5)$. For $k = 1$, we correct $\bar{Q}$ by the rank–one update F_1 to obtain Q_1 and then plot the level of orthogonality of Q_1. While k increases, we observe the benefit of adding F_k to $\bar{Q}$ on the orthogonality quality. We stop the plot at $k = 100$. At this step, the matrix Q_{100} has nearly reached its final level of orthogonality ($1.44 \cdot 10^{-14}$ for $k = n-1$). With $\vartriangle$, we plot the corresponding $u\kappa_{k+1}$, $k = 0,\ldots,n-1$. The theorem predicts that for each k, $\|I_n - Q_k^T Q_k\|$ is bounded above by $u\kappa_{k+1}$ times a constant. In this experiment we observe that both curves fit well. This indicates that the constant can be taken close to one for these experiments and that the bound c) of Theorem 1.6.3 is tight. In Figure 1.6 b), we illustrate that Property b) of Theorem 1.6.3 holds. In this case $\|A - Q_k \bar{R}\|$ is smaller than $u\|A\|$ times a constant where the constant is small.

Similar experiments are reported in Figure 1.7 for the matrix GRE_216B that also illustrates the tightness of the bounds.

Given the singular value distribution of A and the machine precision, Theorem 1.6.3 gives us a set of k for which all the associated matrices Q_k satisfy a prescribed level of orthogonality. Since the amount of work of Algorithm 1 increases with k, we can choose the lowest k of this set and update $\bar{Q}$ with the rank–k matrix F_k.

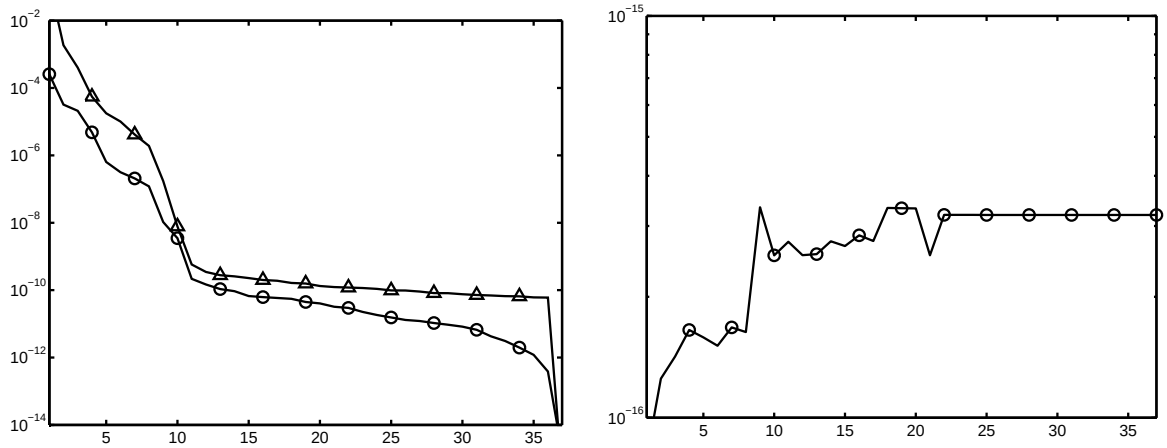
[7] http://math.nist.gov/MatrixMarket/

Therefore an interesting feature of Algorithm 1 is that it is able to adapt its amount of work with respect to the level of orthogonality expected. For example, if the level of orthogonality required for the Q–factor of matrix GRE_216B is 10^{-9}, with both Theorem 1.6.3 and the knowledge of $u\kappa_{k+1}$, we can choose $k = 10$. Meanwhile, if the level of orthogonality required is 10^{-14}, we can estimate the value a–priori $k = 37$. A–posteriori we observe in Figure 1.7 and curve $\|I_n - Q_k^T Q_k\|$ that these two choices are correct.



(a) $\triangle$ $u\kappa_{k+1}$, $\circ$ $\|I_n - Q_k^T Q_k\|$ (b) $\circ$ $\|A - Q_k \bar{R}\|/\|A\|$

Figure 1.6: Illustrations of bounds (b) and (c) of Theorem 1.6.3 for matrix $P(1500, 500, 1, 5)$



(a) $\triangle$ $u\kappa_{k+1}$, $\circ$ $\|I_n - Q_k^T Q_k\|$ (b) $\circ$ $\|A - Q_k \bar{R}\|/\|A\|$

Figure 1.7: Illustrations of bounds (b) and (c) of Theorem 1.6.3 for matrix GRE_216B

1.6.2.2 An application of choice: seed–GMRES

A practical framework where our algorithm fits perfectly is the seed–GMRES method for solving a sequence of linear systems with the same coefficient matrix but for a sequence of different right–hand sides. Roughly speaking one solves the linear

system for one right–hand side at a time but uses the Krylov space associated with the current right–hand side to compute a good initial guess for the next ones.
Let us now briefly describe the seed–GMRES method and the various alternatives we consider to compare with our algorithm. Let Z be a square matrix of order m with full rank. We want to solve the linear systems $Zx^{(i)} = b^{(i)}$ for $i = 0, \ldots, p$ by using seed–GMRES with MGS (see e.g. [101, 114]). For the sake of clarity, but without loss of generality, we describe the method assuming that the initial guesses for all the right–hand sides are zeros, and we only illustrate it when the first right–hand side has converged. For the next ones, the same algorithm applies but the initial guesses are no longer zero making the notation more complicated for a purpose that is out of the scope of this paper.
We first run GMRES with MGS to solve the linear system $Zx^{(0)} = b^{(0)}$. This amounts to solving the linear least squares problem

$$\min_{y \in \mathbb{R}^{n-1}} \|b^{(0)} - ZV_{n-1}^{(0)} y\|,$$

where $V_{n-1}^{(0)}$ is a set of $n-1$ vectors built with an Arnoldi process on Z using the starting vector $b^{(0)}$ and orthogonalization scheme MGS. In most applications, the computational burden lies in the matrix–vector products and the scalar products required to solve this linear least squares problem. In seed–GMRES, the subsequent right–hand sides benefit from this work. An effective initial guess $x^{(i)} = V_{n-1}^{(0)} y^{(i)}$ for the system i is obtained by solving the same linear least squares problem but with another right–hand side, namely

$$\min_{y \in \mathbb{R}^{n-1}} \|b^{(i)} - ZV_{n-1}^{(0)} y\|.$$

We first compare four approaches to solve this problem. In the first part, we present two standard algorithms and compare them in terms of floating point operations (flops) with an approach implementing Algorithm 1. In the second part, one aspect of our problem is examined in more details to show that – under reasonable assumptions – a rank–one update is enough to recover with Algorithm 1 a good level of orthogonality. In this particular case, a second algorithm is also derived based on an heuristic that enables us to save substantially computational work. Finally we illustrate the effectiveness of our approach when embedded in large electromagnetism applications.
In the sequel, the superscript $^{(0)}$ is omitted and the matrix A denotes the computed matrix $\left(b^{(0)}, ZV_{n-1}^{(0)}\right)$ similarly as in the first section, MGS is run on A of size $m \times n$ in a well designed floating point arithmetic to obtain $\bar{Q}$ and $\bar{R}$. Indeed, the Arnoldi process gives $\bar{Q} = V_n^{(0)}$ but for the sake of generality this property is not taken into account.

1.6.2.2.1 The three approaches Since we already have computed the QR factorization of (b, ZV_{n-1}) via MGS, an efficient way to solve the linear least squares with $b^{(i)}$ is to compute the R–factor of the QR factorization of $\left(b, ZV_{n-1}, b^{(i)}\right)$ via MGS.

In practice it remains to compute the last column of this R–factor that is $c_{\mathrm{MGS}}^{(i)}$ such that

$$c_{\mathrm{MGS}}^{(i)} = \begin{pmatrix} \bar{q}_1^T b^{(i)} \\ (I_m - \bar{q}_2 \bar{q}_2^T) \bar{q}_1^T b^{(i)} \\ \vdots \\ \bar{q}_n^T (I_m - \bar{q}_{n-1} \bar{q}_{n-1}^T) \dots (I_m - \bar{q}_1 \bar{q}_1^T) b^{(i)} \end{pmatrix}. \tag{1.146}$$

From a complexity point of view the MGS algorithm applied to A requires $2mn^2$ flops while the $(p-1)$ projections (1.146) for the remaining right–hand sides require $4mnp$ flops.
A second way is to reorthogonalize $\bar{Q}$, the Q–factor from MGS, before performing the set of projections. We reorthogonalize $\bar{Q}$ to obtain Q_k using formula (1.137), with Algorithm 1. The value of k is chosen large enough so that Q_k has columns orthonormal up to machine precision. Then we project the $(p-1)$ remaining right–hand sides with classical Gram–Schmidt type projections, that is

$$c_{\mathrm{CGS}}^{(i)} = \begin{pmatrix} \hat{q}_1^T b^{(i)} \\ \vdots \\ \hat{q}_n^T b^{(i)} \end{pmatrix}. \tag{1.147}$$

This latter approach still requires $2mn^2$ flops to get the QR–factorization of A but only $2mnp$ flops for the $(p-1)$ projections. However, we have to add the cost of the reorthogonalization that is mainly governed by the construction of T, that is mn^2 flops, plus the assembly of Q_k with Equation (1.137), that is $4mnk$ flops.
A third approach consists in not using MGS as orthogonalization scheme in GMRES but instead iterated modified Gram–Schmidt with a criterion denoted by MGS2(K) [76]. The extra costs compared with MGS comes from the reorthogonalizations. We call ν the quantity so that the cost of MGS2(K) is $2mn^2\nu$; ν ranges from 1 (if no reorthogonalization is performed) to 2 (if one reorthogonalization per column is performed). The parameter K defines the criterion used to decide whether the reorthogonalization has to be performed or not. According to [34], we choose the value $K = \sqrt{2}$ and justify this choice later through numerical experiments. The aim here is to obtain directly an orthogonal basis to machine precision and then use Equation (1.147) with the Q–factor obtained with MGS2(K).
We summarize the costs in flops of these three approaches in Table 1.10. From Table 1.10, for rather small p a good approach in term of flops seems to be MGS. However our interest is in large p. For large p, Algorithm 1 is interesting over MGS2(K) when $\frac{3}{2} + \frac{2k}{n} \leq \nu$. We have seen that the parameter k is determined a–priori by the level of orthogonality required by the user. In the sequel, we consider k small compared to n, the critical value is then $\nu = 1.50$. A larger value for ν would make our approach more efficient than MGS2(K) – and vice versa – since the construction of T which requires mn^2 is the main cost of Algorithm 1, therefore we compare $3mn^2$ (Algorithm 1) to $2mn^2\nu$ (MGS2(K)).

1.6.2.2.2 Special feature of $A = (b, ZV_{n-1})$ Greenbaum, Rozložník and Strakoš [69] have shown that for GMRES with orthogonalization schemes MGS, the quantity

MGS and $(p-1)$ projections (1.146)	$2mn^2 + 4mnp$
Algorithm 1 and $(p-1)$ projections (1.147)	$2mn^2 + 2mnp + mn(n+4k)$
MGS2(K) and $(p-1)$ projections (1.147)	$2mn^2\nu + 2mnp$

Table 1.10: Flops required for the three orthogonalization schemes and associated projection considering m large over k, n and p.

$\sigma_n((b, ZV_{n-1}))$ is of the order of the residual of GMRES obtained at step $n-1$. When the residual is small, we expect $A = (b, ZV_{n-1})$ to be ill–conditioned and so an important loss of orthogonality is expected with MGS.
Since $\sigma_{n-1}((b, ZV_{n-1})) \geq \sigma_{n-1}(ZV_{n-1}) \geq \sigma_{n-1}(Z)\sigma_n(V_{n-1})$, if we assume Z and V_{n-1} well–conditioned, we get that κ_2 is close to one. We note that if the matrix (b, ZV_{n-1}) is numerically nonsingular then in [61], it is stated that $\bar{Q}$ $(= V_n)$ is well–conditioned and we only restrict our study to well–conditioned matrix Z. From this analysis, the value $k = 1$ is enough for the reorthogonalization of $\bar{Q}$ with Algorithm 1 to obtain a Q–factor orthogonal up to machine precision. In the experimental part, we illustrate that $k = 1$ is indeed necessary and sufficient in the seed–GMRES context.
For small k compared to n, the cost of the a–posteriori reorthogonalization procedure of MGS performed with Algorithm 1 is mainly governed by the computation of the $n(n+1)/2$ entries of the matrix $\bar{T}$ (Section 1.6.2.2.1). We debase Algorithm 1 to get a second algorithm, this algorithm relies mainly on an heuristic that attempts to avoid the complete computation of $\bar{T}$. First of all we consider that the rank of $\bar{P}_{11}$ is one, – this is justified by the special feature of the problem: κ large and κ_2 close to one – and since P_{11} is strictly upper triangular therefore nilpotent (i.e. $P_{11}^n = 0$), we have $P_{11}^2 = 0$ and so Equation (1.145) reduces to $P_{11} = T$. Therefore in practice we just compute T and use it as P_{11}. But computing all the entries of a rank–one matrix may be considered as a waste of time. In theory, it is enough to build a row i and a column j so that the entry (i, j) is nonzero. With rounding errors, the best choice is to build the row i and the column j such that the entry (i, j) is the largest in magnitude. In practice, if the entry (i, j) is not the largest but of the order of the largest entry of $\bar{T}$, the procedure is still reliable. A good candidate to be of the order of the largest entry of $\bar{T}$ is $|\bar{q}_1^T \bar{q}_n|$ since the orthogonality given by MGS of $\bar{q}_n$ over $\bar{q}_1$ assumes in theory the orthogonality of all the previous vectors ; in practice, we expect the loss of orthogonality in V to be maximal between $\bar{q}_n$ and $\bar{q}_1$. This defines our heuristic:

Heuristic
$|\bar{q}_1^T \bar{q}_n|$ is of the order of the largest entry in magnitude of $\bar{T}$.

Thanks to this heuristic only the first row and the last column of $\bar{T}$ are computed. Algorithm 2 uses the reorthogonalization based on this heuristic, it is described in Table 1.11. The fourth approach to compute the orthogonalization and the projections in seed–GMRES is to use Algorithm 2 and then project the $(p-1)$ other right–hand sides with Equation (1.147). The whole algorithm is very cheap and only requires $2mn^2 + 2mnp + 8mn$ flops in which $8mn$ flops are necessary for the reorthogonalization. For comparison, $8mn$ corresponds to the extra cost of the

1. run MGS on $A = (b, ZV_{n-1})$ to obtain $\bar{Q}$ and $\bar{R}$,
2. compute $u^T = (\bar{q}_n^T \bar{q}_1, \ldots, \bar{q}_n^T \bar{q}_{n-1}, 0)$, $c = u(1)$
   and $w^T = (0, \bar{q}_1^T \bar{q}_2, \ldots, \bar{q}_1^T \bar{q}_n)$,
3. $c = u(1)$, $u = u/\|u\|$, $w = w/\|w\|$, $c = c/u(1)/w(n)$, $s = \sqrt{1-c^2}$,
4. compute $Q_1 = \bar{Q} + \bar{Q}(w(s^{-1}-1) - ucs^{-1})w^T$.

Table 1.11: Algorithm 2: MGS with an a–posteriori reorthogonalization by a rank–one update using the heuristic.

reorthogonalization of about 4 columns.

1.6.2.2.3 Numerical experiments in a large electromagnetism calculation Our case study arises from large calculations in electromagnetism. The boundary element method is used to discretize the 3D Maxwell's equations on the surface of an object. The formulation relies on the combined field integral equations and the preconditioner used is a sparse approximate inverse [130], this means that in practice the preconditioned matrix Z is well–conditioned. Moreover one can notice that the matrix Z is not explicitly known and is accessed through matrix–vector product done via the fast multipole method. All the calculations are performed using double precision arithmetic. There are several linear systems $Zx^{(i)} = b^{(i)}$ to be solved, for this typical calculation we have $p = 180$ but this value might be much larger if several radar cross sections have to be computed, as is often the case in engineering applications. For each right–hand side, GMRES is stopped at iteration l if the approximate solution $x_l^{(i)}$ verifies $\|b^{(i)} - Ax_l^{(i)}\|/\|b^{(i)}\| \leq 10^{-14}$. We remark that the problem is defined in complex arithmetic, however in order to be consistent with the whole paper the real notation is maintained.
Four geometries are considered, they represent standard test–cases for electromagnetism calculations, namely a cetaf, an Airbus airplane, a sphere and an almond [130]. In Table 1.12, we give the characteristics of the matrices (b, ZV_{n-1}) obtained by a GMRES–MGS run on these matrices. The values obtained with GMRES–MGS2(K) are the same. For more information on the method and the test–case, we refer to [130].
In Table 1.12, (# iter) represents the number of GMRES iterations required to converge. The number of columns of the matrix $A = (b, ZV_{n-1})$ is $n = \#$ iter $+1$, the number of rows is m. As expected (see Section 1.6.2.2.2), the condition number κ is such that $\kappa \cdot 10^{-14}$ is close to one, while κ_2 is of order $\mathcal{O}(1)$.
The fourth column of Table 1.12 corresponds to the average number of reorthogonalizations obtained with MGS2($\sqrt{2}$). In this cases, MGS2($\sqrt{2}$) systematically performs an extra reorthogonalization per matrix–vector product, which explains the constant value ($\nu = 2.00$).
In Table 1.13, we illustrate that all the residual errors $\|A - \bar{Q}\bar{R}\|$ – where $\bar{Q}$ and $\bar{R}$ designed the QR–factor given by one the four algorithms – are of the order of the

	m	# iter	κ	κ_2	ν
Cetaf	5391	31	$9.7 \cdot 10^{14}$	27	2.00
Airbus	23676	104	$3.6 \cdot 10^{14}$	14	2.00
Sphere	40368	59	$3.9 \cdot 10^{14}$	6.4	2.00
Almond	104973	71	$5.1 \cdot 10^{14}$	5.9	2.00

Table 1.12: Characteristics of $A = (b, ZV_{n-1})$

	MGS	Algorithm 1	MGS2($\sqrt{2}$)	Algorithm 2
Cetaf	$2.8 \cdot 10^{-17}$	$2.8 \cdot 10^{-16}$	$1.8 \cdot 10^{-16}$	$2.9 \cdot 10^{-16}$
Airbus	$4.0 \cdot 10^{-17}$	$4.4 \cdot 10^{-16}$	$2.7 \cdot 10^{-16}$	$4.4 \cdot 10^{-16}$
Sphere	$5.8 \cdot 10^{-17}$	$2.7 \cdot 10^{-16}$	$1.6 \cdot 10^{-16}$	$2.7 \cdot 10^{-16}$
Almond	$3.9 \cdot 10^{-17}$	$3.9 \cdot 10^{-16}$	$3.9 \cdot 10^{-16}$	$2.2 \cdot 10^{-16}$

Table 1.13: Residual errors for the four case–test and the different algorithms.

machine precision. In Table 1.14, the different levels of orthogonality characterized with $\|I_n - \bar{Q}^T\bar{Q}\|$ are given. As expected, MGS completely looses the orthogonality while the three other approaches give a set of vectors orthogonal up to machine precision. In the context of seed–GMRES, this enables us to use confidently Equation (1.147) to project the $(p-1)$ remaining right–hand sides.
A conclusion drawn from Table 1.14 is that in the case of GMRES–MGS applied to a not too ill–conditioned matrix the value $k = 1$ is satisfactory (Algorithm 1 with $k = 1$). Moreover from Table 1.13 and Table 1.14, we observe that Algorithm 2 relying on the heuristic works fine in practice.
One might question about the relevance of the choice $K = \sqrt{2}$ and its possible artificial high cost. In Table 1.15 we report on the sensitivity of the orthogonality quality with respect to the choice of the threshold. These experiments assess the choice of $K = \sqrt{2}$ for MGS2(K). This value gives a good orthogonality level for all the examples while the others tested ($K = 2$ and $K = \sqrt{5}$) fail. However $K = \sqrt{2}$ implies in these cases $\nu = 2.00$ meaning that the criterion is unable to save any reorthogonalization. This result is not satisfactory and highlights a weakness of the MGS2(K) procedure. Even if κ_2 is close to one, improving noticeably the condition number κ cannot be obtained in the general case by removing only a column of (b, ZV_{n-1}), it is a global phenomenon that needs a global treatment (e.g. to add the singular vector associated to the smallest singular value to all the columns). In the same way, the loss of orthogonality is global and affects all the columns of $\bar{Q}$. An algorithm like MGS2(K) that acts locally on each column performs poorly

	MGS	Algorithm 1	MGS2($\sqrt{2}$)	Algorithm 2
Cetaf	$1.6 \cdot 10^{-02}$	$1.9 \cdot 10^{-15}$	$2.8 \cdot 10^{-16}$	$2.4 \cdot 10^{-15}$
Airbus	$1.8 \cdot 10^{-02}$	$1.5 \cdot 10^{-15}$	$3.7 \cdot 10^{-16}$	$1.6 \cdot 10^{-15}$
Sphere	$3.9 \cdot 10^{-02}$	$5.4 \cdot 10^{-16}$	$3.0 \cdot 10^{-16}$	$7.8 \cdot 10^{-16}$
Almond	$4.1 \cdot 10^{-02}$	$6.8 \cdot 10^{-16}$	$2.8 \cdot 10^{-16}$	$7.9 \cdot 10^{-16}$

Table 1.14: $\|I_n - \bar{Q}^T\bar{Q}\|$ for the four case–test and the different algorithms.

	MGS2($\sqrt{2}$)	MGS2(2)	MGS2($\sqrt{5}$)
Cetaf	$2.8 \cdot 10^{-16}$ ($\nu = 2.00$)	$6.3 \cdot 10^{-16}$ ($\nu = 1.90$)	$1.2 \cdot 10^{-15}$ ($\nu = 1.87$)
Airbus	$3.7 \cdot 10^{-16}$ ($\nu = 2.00$)	$3.9 \cdot 10^{-03}$ ($\nu = 1.02$)	$8.8 \cdot 10^{-03}$ ($\nu = 1.01$)
Sphere	$3.0 \cdot 10^{-16}$ ($\nu = 2.00$)	$7.5 \cdot 10^{-15}$ ($\nu = 1.52$)	$4.9 \cdot 10^{-04}$ ($\nu = 1.07$)
Almond	$2.8 \cdot 10^{-16}$ ($\nu = 2.00$)	$1.7 \cdot 10^{-03}$ ($\nu = 1.06$)	$5.2 \cdot 10^{-03}$ ($\nu = 1.03$)

Table 1.15: $\|I_n - \bar{Q}^T \bar{Q}\|$.

in this case, whereas Algorithm 1 and 2 represent appealing strategies since the reorthogonalization – that has to be global – is expressed as a rank–one update.
Finally, there exists other examples where the value of $k > 1$ can be given a priori. Still for the solution of linear systems with multiple right–hand sides, we mention for instance the block(k)–seed–GMRES–MGS algorithm; that is one runs block GMRES on k vectors, when the convergence is observed, a rank–k update is performed to recover an orthogonal set of vectors, that we use to project the $p - k$ right–hand sides as in seed–GMRES (this method is shortly described in Section 2.6 and its QMR variant is detailed in [85]). In Table 1.16, we report on experiments on a sphere test problem of size 972 solved for 3 right–hand sides using a threshold for the stopping criterion equal to 10^{-14}. The 4 reduced condition numbers κ_i observed when running a classical block-GMRES with $MGS2(K = 2)$ for solving three right–hand sides are also displayed. It can first be observed that the first three reduced condition numbers are very large as it could have been expected. In

	m	# iter	κ	κ_2	κ_3	κ_4	ν
sphere	972	151	$4.5 \cdot 10^{15}$	$6.2 \cdot 10^{14}$	$4.2 \cdot 10^{14}$	26	2.00

Table 1.16: Characteristics of $A = (b, ZV_{n-1})$ Three right–hand sides corresponding to ($\theta = 0^o, \varphi = 0^o : 10^o : 20^o$) .

Table 1.17, we display the values of $\|I_n - \bar{Q}^T \bar{Q}\|$ and $\|A - QR\|$; it can be observed that the proposed re-orthogonalization techniques are still successful for k larger than one (i.e. three in this latter case).

	MGS	Algorithm 1	MGS2($\sqrt{2}$)	Algorithm 2
$\|I_n - Q^T Q\|$	$0.66 \cdot 10^{+00}$	$0.28 \cdot 10^{-14}$	$0.88 \cdot 10^{-15}$	$0.28 \cdot 10^{-14}$
residual errors	$0.12 \cdot 10^{-15}$	$0.15 \cdot 10^{-14}$	$0.63 \cdot 10^{-15}$	$0.15 \cdot 10^{-14}$

Table 1.17: Quality comparison of the algorithm.

Conclusion

In this paper we propose an *a posteriori* reorthogonalization technique based on a rank–k update to reorthogonalize a set of vectors built by the modified Gram–Schmidt algorithm. We show that for large enough k, we can fully recover the orthogonality. We illustrate the effectiveness of our technique in the framework of the iterative solution of linear systems based on the GMRES algorithm. On a

set of industrial test problems we demonstrate that our algorithm is efficient and outperforms classical approaches that also permit to remedy the loss of orthogonality observed when GMRES has converged. **Acknowledgment**
The authors would like to thank Chris C. Paige for his careful reading of several versions of the chapter and for his fruitful comments.

Appendix : Two technical Lemmas

In this appendix, we prove two technical lemmas that are used in the proof of Theorem 1.6.3. Lemma 1.6.4 relates the norm of the computed $\widetilde{Q}$ to the condition number of A. We prove that for a well–conditioned matrix A, $\|\widetilde{Q}\|$ is close to one. Lemma 1.6.5 gives an upper bound for the quantity $\|A - \widetilde{Q}\bar{R}\|$. Similar residual bounds have been derived in [13, 15], but they involve the computed $\bar{Q}$, instead of $\widetilde{Q}$. In these two lemmas, notations directly refers to the ones used in the article.

Lemma 1.6.4 *Suppose that* $\bar{c}_1 u\eta\kappa < 1$, *then*

$$\|\widetilde{Q}\| \leq \frac{1}{1 - \bar{c}_1 u\eta\kappa}.$$

Proof of Lemma 1.6.4 Since $\widetilde{P}_{21} = \widetilde{Q}(I_n - \widetilde{P}_{11})$ we obtain from (1.134)

$$\begin{aligned}
&\widetilde{P}_{21} = VSW^T = \widetilde{Q}(I_n - UCW^T)\\
&\widetilde{Q} = VSW^T + \widetilde{Q}UCW^T\\
&\|\widetilde{Q}\| \leq \|S\| + \|\widetilde{Q}\|.\|C\|\\
&\|\widetilde{Q}\| \leq \|S\|/(1 - \|C\|) \leq 1/(1 - \bar{c}_1 u\eta\kappa). \quad \heartsuit
\end{aligned}$$

Lemma 1.6.5 *Suppose that* $\bar{c}_1 u\eta\kappa < 1$, *then*

$$\|A - \widetilde{Q}\bar{R}\| \leq [\bar{c}_2 + \bar{c}_1/(1 - \bar{c}_1 u\eta\kappa)]u\|A\|.$$

Proof of Lemma 1.6.5 Since $\widetilde{P}_{21} = \widetilde{Q}(I_n - \widetilde{P}_{11})$, (1.132) gives $-E_2 = A - \widetilde{P}_{21}\bar{R} = A - \widetilde{Q}\bar{R} + \widetilde{Q}\widetilde{P}_{11}\bar{R}$, which with $E_1 = \widetilde{P}_{11}\bar{R}$ and the previous lemma shows

$$\begin{aligned}
&A - \widetilde{Q}\bar{R} = -E_2 - \widetilde{Q}E_1\\
&\|A - \widetilde{Q}\bar{R}\| \leq [\bar{c}_2 + \bar{c}_1/(1 - \bar{c}_1 u\eta\kappa)]u\|A\|. \quad \heartsuit
\end{aligned}$$

1.7 Miscellaneous topics on the Gram–Schmidt algorithm

1.7.1 The modified Gram–Schmidt algorithm is as the Householder algorithm ?

Björck and Paige [15] based all their analysis of the modified Gram–Schmidt algorithm on the following observation. The modified Gram–Schmidt algorithm performs the same operations on A as the Householder algorithm on the augmented A (see Section 1.6). Therefore the R–factors generated by these two algorithms are the same.

If the algorithms are those currently given in the literature (e.g. [14, 63]), this is not exactly true. Some operations differ slightly and eventually the computed the R–factors differ (at the machine precision level).

For the sake of completeness in Algorithm 7 and Algorithm 8, we give the modified Gram–Schmidt algorithm and the Householder algorithm on the augmented matrix respectively. These algorithms are slightly modified variants from the text-book versions. The standard version for MGS is given in Algorithm 2. In Algorithm 9, we give the modifications to make Algorithm 8 be able to generate the text-book Householder algorithm on the augmented matrix.

Both algorithms 7 and 9 give exactly the same floating–point numbers, since they perform exactly the same operations.

We should of course point out that the differences between the text-book versions and their variants are so slight that it fully justifies the assumption that the R–factors from the modified Gram–Schmidt algorithm and from the Householder on the augmented matrix are indeed the same. When wrote these few lines only to illustrate that, with small modifications, we can effectively have exactly the same computed digits.

Algorithm 7 The modified Gram–Schmidt algorithm with a slight modification so that the operations performed are exactly those of the Householder algorithm given in Algorithm 8.

1. **for** $j = 1$ to n **do**
2. $\quad w = a_j$
3. $\quad$ **for** $i = 1$ to $j - 1$ **do**
4. $\quad\quad r_{ij} = q_i^T w$
5. $\quad\quad w = w - q_i r_{ij}$
6. $\quad$ **end for**
7. $\quad q_j = w/\|w\|_2$
8. $\quad r_{jj} = w^T q_j$
9. **end for**

1.7.2 Blindy MGS: a model for the modified Gram–Schmidt in finite–precision arithmetic.

Let us assume that we run modified Gram–Schmidt algorithm in exact arithmetic on A where

Algorithm 8 Householder transformations on the augmented matrix for a QR factorization. A slight modification is added so that the operations performed are exactly those of the modified Gram–Schmidt algorithm given in Algorithm 7.

1. $\widetilde{A} = \begin{pmatrix} 0_{n,n} \\ A \end{pmatrix}$
2. $\widetilde{m} = m + n$
3. $z = \widetilde{a}_1$
4. **for** $j = 1$ to n **do**
5. $\quad w_j = 0_{\widetilde{m}}$
6. $\quad w_j(j+1:\widetilde{m}) = z(j+1:\widetilde{m})/\|z(j+1:\widetilde{m})\|_2$
7. $\quad w_{j,j} = -1$
8. $\quad r(1:j,j) = z(1:j) - w(1:j,j)(w(1:\widetilde{m},j)^T z)$
9. $\quad$ **if** $j < n$,
10. $\quad\quad z = \widetilde{a}_{j+1}$
11. $\quad\quad$ **for** $i = 1$ to j **do**
12. $\quad\quad\quad z = z - w_i(w_i^T z)$
13. $\quad\quad$ **end for**
14. $\quad$ **endif**
15. **end for**

Algorithm 9 Modifications to make to Algorithm 8 in order to recover a text-book Householder algorithm.

...
6. $\quad w_j(j:\widetilde{m}) = z(j:\widetilde{m})$
7a. $\quad \beta = \|z(j:\widetilde{m})\|_2 \cdot \mathrm{sign}(z(j,j))$
7b. $\quad w_{j,j} = w_{j,j} + \beta$
7c. $\quad w_j = w_j/\|w_j\|_2$
8. $\quad r(1:j,j) = z(1:j) - 2w(1:j,j)(w(1:\widetilde{m},j)^T z)$
...
12. $\quad\quad\quad z = z - 2w_i(w_i^T z)$
...

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{1.148}$$

Since $a_3 \in \mathrm{Span}(a_1, a_2)$, for $j = 3$ at step 7 of Algorithm 2, we have $r_{33} = \|q_3\|_2 = 0$, and so a breakdown occurs at step 8 when we want to compute $q_3 = q_3/r_{33}$. The singularity is of the form $0/0$.

The two standard ways to deal with this singularity are (a) to remove the third column of Q or (b) to take as q_3 a normal vector belonging to the orthogonal space of $\mathrm{Span}(q_1, q_2)$. Then the Gram–Schmidt process can go onto the fourth column. We suggest a third solution: (c) add a normalized vector. This variant is called blindy–MGS.

For the matrix A given in equation (1.148), we choose to add the vector $q_3 =$

$(\ 1/2\ \ 1/2\ \ \sqrt{2}/2\ \ 0)$ of norm 1. The resulting Q and R factor are respectively

$$Q = \begin{pmatrix} 1 & 0 & 1/2 & -1/2 \\ 0 & 1 & 1/2 & -1/2 \\ 0 & 0 & \sqrt{2}/2 & \sqrt{2}/2 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad R = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & \sqrt{2}/2 \\ 0 & 0 & 0 & \sqrt{2}/2 \end{pmatrix}$$

In Figure 1.8, a geometrical representation is given.

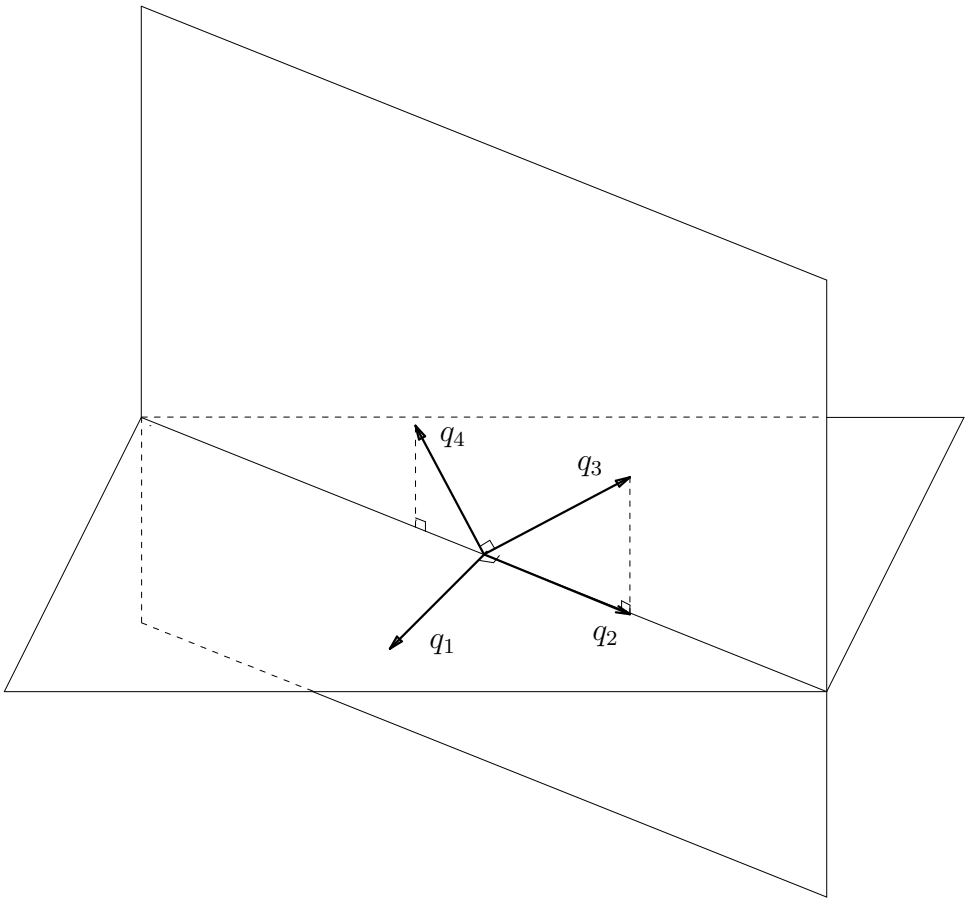


Figure 1.8: (q_1, q_2, q_3, q_4) given by blindy–MGS on A.

The properties of the set of vectors constructed by blindy–MGS are very close to what one can observe when using the modified Gram–Schmidt algorithm on a matrix in floating–point arithmetic. First of all, note that $A = QR$. If A is nonsingular, blindy–MGS reduces to MGS so that Q has orthonormal columns. If A is singular, say $\mathrm{Rank}(A) = (n-1)$, Q may be (a) orthogonal, (b) of full rank with two blocks Q_1 and Q_2 with orthonormal columns or (c) rank deficient $\mathrm{Rank}(Q) = (n-1)$. We depict, in our example with matrix A given by equation (1.148), the case (b).

We can enumerate some of the properties of the set of vectors Q.

$$(I_m - q_n q_n^T)(I_m - q_{n-1} q_{n-1}^T) \dots (I_m - q_1 q_1^T) A = 0 \tag{1.149}$$

if $\mathrm{Rank}(A) = n - k$, then

$$\mathrm{Rank}(\mathrm{triu}(I_n - Q^T Q))) \leq k \tag{1.150}$$

$$\text{there exists } \hat{Q} \text{ such that } \hat{Q}^T \hat{Q} = I_n \text{ and } \mathrm{Rank}(\hat{Q} - Q) \leq k \tag{1.151}$$

...

Equation (1.149) is *equivalent* to the relation shown by Björck [13, Eq. (5.4)] in the floating–point arithmetic case, that is

$$\|(I_m - q_n q_n^T)(I_m - q_{n-1} q_{n-1}^T) \dots (I_m - q_1 q_1^T) A\|_E \leq 3.25(\frac{2}{3}m + 1)(n-1) \cdot 2^{-t} \|A\|_E \tag{1.152}$$

For example, it enables him to derive a backward stable method based on the modified Gram–Schmidt algorithm for solving the least–squares problem. Equation (1.150) and Equation (1.151) are those that initiated the work presented in Section 1.6.

The purpose of this section is to provide a simple model, that quickly enables us to get an insight on the numerical behaviour of MGS in floating-point arithmetic.

1.7.3 Accurate eigencomputations using the modified Gram–Schmidt algorithm.

Braconnier, Langlois and Rioual [19] tested different orthogonalization schemes in the Arnoldi process to compute eigenvalues. Their experiments were on small matrices (of order 100) and the number of Arnoldi iterations performed is of the order of the matrix. In the following, we consider similar extreme situations.

Let us study the Toeplitz matrix of order $n = 100$ (see [19, 131])

$$\mathbb{Z} = \begin{pmatrix} 1 & \frac{3}{4} & \dots & \frac{3}{2^n} \\ 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \frac{3}{4} \\ 0 & \dots & \dots & 1 \end{pmatrix}$$

The Arnoldi method is performed with the starting vector $b = (1, \dots, 1)^T$. In Figure 1.9, we plot the normwise backward error for Householder–Arnoldi and MGS–Arnoldi. The backward error formulae are given in Braconnier et al. (also e.g. [28, p. 76]).

It is well known that Arnoldi computes accurate eigenvalues and eigenvectors of $\mathbb{Z}$ when the term $h_{j+1,j}$ of the Hessenberg matrix is of the order of the machine precision. In our case, this happens at step n of the Householder–Arnoldi method as shown in [19].

At step n of MGS–Arnoldi, the computed $h_{n+1,n}$ is not small at all. We note that the matrix $A_n = (b, \mathbb{Z}V_{n-1})$ has two small singular values, and so the the numerical rank of A_n can be considered equal to $(n-2)$. Our idea is to push the iteration j

further than n so that

$$\mathrm{Rank}(A_{j+1}) = \mathrm{Rank}\,(b, \mathbb{Z}V_j) = n.$$

In this last equation, Rank stands for *numerical rank*. This happens at step $n+2$: A *numerically* spans $\mathbb{R}^n$. In practice, the relation (1.152) holds at step $n+2$: necessarily this implies, since A spans $\mathbb{R}^n$ and Av_{n+2} belongs to $\mathbb{R}^n$, that

$$h_{n+3,n+2} = \|((I_m - q_n q_n^T)(I_m - q_{n-1}q_{n-1}^T)\ldots(I_m - q_1 q_1^T)(Av_{n+2}))\|_2$$

is of the order of the machine precision. We end up with the factorization

$$AV_{n+2} = V_{n+2}H_{n+2,n+2} + E_{n+2},$$

where E_{n+2} is of the order of the machine precision. The eigenvalues and eigenvectors of $H_{n+2,n+2}$ are good approximations to those of A. Note that $H_{n+2,n+2}$ is an $(n+2$–by–$(n+2)$ matrix and has $n+2$ eigencouples (in this particular case). We throw away the two smallest eigenvalues and corresponding eigenvectors. They represent the rank deficiency of V_{n+2} (n–by–$n+2$ matrix). The result is that the n eigencouples for $\mathbb{Z}$ are accurate (see Figure 1.9).

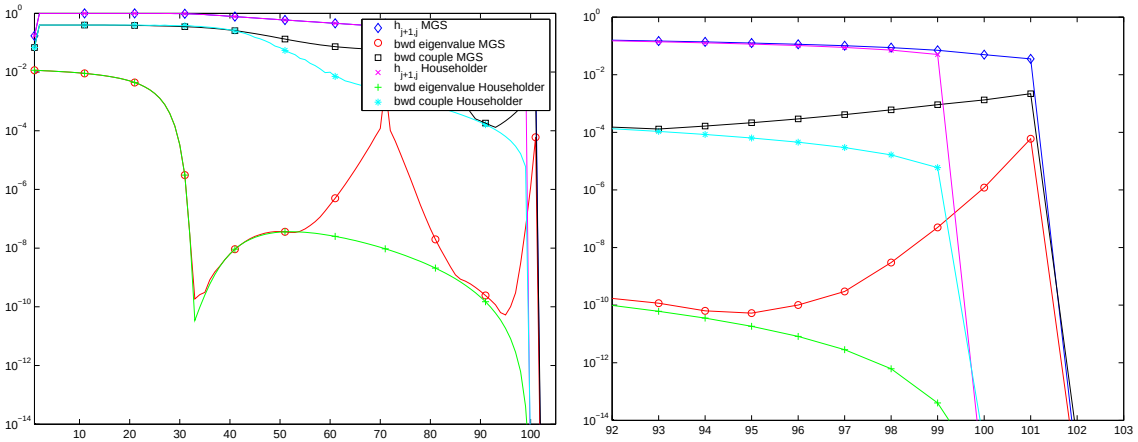


Figure 1.9: Backward error analysis of Arnoldi using Householder and modified Gram–Schmidt orthogonalization schemes. The order of the matrix $\mathbb{Z}$ is $n = 100$, the maximum number of iterations with the Householder orthogonalization scheme is 100, the maximum number of iterations with MGS orthogonalization scheme is 102. At step 102, the Arnoldi process returns 102 eigenvalues and we throw away the smallest 2.

1.8 Future work

In this first chapter, we have seen new results on the Gram–Schmidt algorithm that completes the existing framework.
The main effort in the future to complete this study is obtain theoretical concerning the classical Gram–Schmidt algorithm. What is the real bound on the orthogonality: experimentally, it is rather easy to obtain a loss of orthogonality of the order of the condition number of the initial matrix times the machine precision, it is hard to get much more. Theoretically it is rather easy to prove that the loss of orthogonality is bounded by the condition number of the initial matrix to the power the number of vectors orthogonalized times the machine precision and a constant.
Current work is devoted to the study of the Gram–Schmidt algorithm with other scalar product than the euclidean one.

II

Chapter 2

Implementation of iterative methods

2.1 Basics

Before describing in detail the different iterative methods we have considered, we first present their common features.

2.1.1 Preconditioning

The convergence of an iterative method for solving a linear system might be slow. To overcome this drawback, one often prefers to solve a transformed linear system that is referred to as the preconditioned linear system. More precisely, if $M_1 M_2 \approx A^{-1}$, we actually solve the linear system

$$M_1 A M_2 y = M_1 b \tag{2.1}$$

with $x = M_2 y$. In our implementation, we allow the user to select left and/or right preconditioning. The use of preconditioners has some consequences on the stopping criterion. We discuss these points in the next paragraph.

2.1.2 Stopping criteria

We have chosen to base our stopping criterion on the normwise backward error [28]. Backward error analysis, introduced by Givens and Wilkinson [136], is a powerful concept for analysing the quality of an approximate solution:

1. it is independent from the details of round-off propagation: the errors introduced during the computation are interpreted in terms of perturbations of the initial data, and the computed solution is considered as exact for the perturbed problem;

2. because round-off errors are seen as data perturbations, they can be compared with errors due to numerical approximations (consistency of numerical schemes) or to physical measurements (uncertainties on data coming from experiments for instance).

The backward error measures in fact the distance between the data of the initial problem and that of the perturbed problem; therefore it relies upon the choice of the data allowed to vary and the norm to measure these variations. In the context of linear systems, classical choices are the normwise and the componentwise perturbations [28]. These choices lead to explicit formulae for the backward error (often a normalized residual) which is then easily evaluated. For iterative methods, it is generally admitted that the normwise model of perturbation is appropriate [6].

Let x_j be an approximation of the solution $x = A^{-1}b$. Then

$$\begin{aligned} \eta_j &= \min\{\varepsilon > 0;\ \|\Delta A\|_2 \le \varepsilon\alpha,\ \|\Delta b\|_2 \le \varepsilon\beta \text{ and } (A+\Delta A)x_j = b + \Delta b\} \\ &= \frac{\|b - Ax_j\|_2}{\alpha\|x_j\|_2 + \beta} \end{aligned}$$

is called the *normwise backward error* associated with x_j. The best one can require from an algorithm is a backward error of the order of the machine precision. In practice, the approximation of the solution is acceptable when its backward error is lower than the uncertainty on the data. Therefore, there is no gain in iterating after the backward error has reached machine precision (or data accuracy).

In all our solvers, the evaluation of the norm of the residual $b - Ax_j$ is given directly from the algorithm so that it does not require an extra matix–vector product.

When the iterative method is used in conjunction with preconditioning, then our stopping criterion is based on the backward error for the preconditioned system (2.1):

$$\eta_j^P = \|M_1 A M_2 z_j - M_1 b\|_2 / (\alpha^P \|x_j\|_2 + \beta^P)$$

with $x_j = M_2 z_j$. We denote by

$$\eta_{A,j}^P = \frac{|r_{j+1,j+1}|}{\alpha^P \|x_j\|_2 + \beta^P}$$

the stopping criterion for the preconditioned iterative method. As previously, we stop the iterations when the computed values of $\eta_{A,j}^P$ and then η_j^P satisfy the prescribed tolerance. We prefer to stop the iterations on the preconditioned linear system and not on the original linear system because the residual which is readily available in the algorithm is that of the *preconditioned* system. It would be too expensive to compute the residual of the unpreconditioned system at each iteration. For the user's information, we also give the value of the backward error for the unpreconditioned system on return from the solver.

We should notice that, for a right preconditioner, $\eta = \eta^P$ (or $\eta_A = \eta_A^P$); this is the reason why right preconditioning is often preferred in many applications. Otherwise, there is a priori no relationship between the backward error of the preconditioned system and that of the unpreconditioned system. Nevertheless, we noticed in our experiments that η (or η_A) is usually smaller than η^P (or η_A^P). It is therefore recommended to use a larger tolerance for the preconditioned system than one would

α^P	β^P	Stopping criterion
0	0	$\frac{\|M_1AM_2z_j-M_1b\|_2}{\|M_1b\|_2}$
0	$\neq 0$	$\frac{\|M_1AM_2z_j-M_1b\|_2}{\beta^P}$
$\neq 0$	0	$\frac{\|M_1AM_2z_j-M_1b\|_2}{\alpha^P\|x_j\|_2}$
$\neq 0$	$\neq 0$	$\frac{\|M_1AM_2z_j-M_1b\|_2}{\alpha^P\|x_j\|_2+\beta^P}$

Table 2.1: Stopping criterion for the preconditioned iterative method.

α	β	Information on the unpreconditioned system
0	0	$\frac{\|Ax_j-b\|_2}{\|b\|_2}$
0	$\neq 0$	$\frac{\|Ax_j-b\|_2}{\beta}$
$\neq 0$	0	$\frac{\|Ax_j-b\|_2}{\alpha\|x_j\|_2}$
$\neq 0$	$\neq 0$	$\frac{\|Ax_j-b\|_2}{\alpha\|x_j\|_2+\beta}$

Table 2.2: Stopping criterion for the unpreconditioned iterative method.

have used on the unpreconditioned one.

How do we choose α, β, α^P and β^P ? Classical choices for α and β that appear in the literature are $\alpha = \|A\|_2$ and $\beta = \|b\|_2$. Similarly, α^P and β^P should be chosen such as $\alpha^P \sim \|M_1A\|_2$ and $\beta^P \sim \|M_1b\|_2$. Any other choice that reflects the possible uncertainty on the data can also be used. In our implementation, default values are used when the user's input is $\alpha = \beta = 0$ or $\alpha^P = \beta^P = 0$. Table 2.1 lists the stopping criteria for different choices of α^P and β^P. Similarly, Table 2.2 explains the output information given to the user on the unpreconditioned linear system on return from the solver.

2.1.3 Implementation details

For some given iterative methods, we basically have two versions of the code:

1. the first is freeware and distributed for non-commercial use only. The source codes is SOON available from the Web at the URL
   `http://www.cerfacs.fr/algor/`
   together with the software licence agreement and a set of example codes. Today, for unsymmetric solvers, only the GMRES and flexible GMRES packages are available.

2. the second version is a tuned implementation that complies with the out–of–core features of the EADS code.

For the sake of maintenance of the code, only one source file exists and is used to generate the source code for each of the four arithmetics. The final packages are written in Fortran 77 and make use of the BLAS routines.
For the sake of simplicity and portability, the implementations are based on the reverse communication mechanism

- for implementing the numerical kernels that depend on the data structure selected to represent the matrix A and the preconditioners,
- for performing the dot products.

This last point has been implemented to allow the use of the solvers in a parallel distributed memory environment, where only the user knows how the data have been distributed (we refer to [52] where some parallel distributed performance is reported).
The out–of–core version is also based on the reverse communication mechanism for all the operations using out–of–core vectors. For a complete description of the user interface, we refer to [51]. The description done for GMRES in this users' guide holds for the other solvers. In particular, we explain

1. the reverse communication management,
2. the control parameters and their default values,
3. the information parameters,
4. how invalid parameters are managed (i.e. automatic corrections and unrecoverable failures).

2.2 The GMRES method

2.2.1 Theoretical presentation

The Generalized Minimum RESidual (GMRES) method was proposed by Saad and Schultz in 1986 [118] in order to solve large, sparse and non Hermitian linear systems. GMRES belongs to the class of Krylov based iterative methods.

For the sake of generality, we describe this method for linear systems whose entries are complex, everything also extends to real arithmetic calculations. Let A be a square nonsingular $m \times m$ complex matrix, and b be a complex vector of length m, defining the linear system

$$Ax = b \tag{2.2}$$

to be solved. Let $x_0 \in \mathbb{C}^m$ be an initial guess for this linear system and $r_0 = b - Ax_0$ be its corresponding residual.

The GMRES algorithm builds an approximation of the solution of (2.2) in the form

$$x_n = x_0 + V_n y \tag{2.3}$$

where V_n is an orthonormal basis for the Krylov space of dimension n defined by

$$\mathcal{K}_n = \text{span}\left\{r_0, Ar_0, \ldots, A^{n-1}r_0\right\},$$

and where y belongs to $\mathbb{C}^n$. The vector y is determined so that the 2–norm of the residual $r_n = b - Ax_n$ is minimized over $\mathcal{K}_n$.

The basis of the columns of the matrix V_n for the Krylov subspace $\mathcal{K}_n$ is obtained via the well known Arnoldi process. The orthogonal projection of A onto $\mathcal{K}_n$ results in an upper Hessenberg matrix $H_n = V_n^H A V_n$ of order n. The Arnoldi process satisfies the relationship

$$AV_n = V_n H_n + h_{n+1,n} v_{n+1} e_n^H, \tag{2.4}$$

where e_n is the n^{th} canonical basis vector. Equation (2.4) can be rewritten as

$$AV_n = V_{n+1}\underline{H}_n$$

where

$$\underline{H}_n = \begin{bmatrix} H_n \\ 0 \cdots 0 \; h_{n+1,n} \end{bmatrix}$$

is an $(n+1) \times n$ matrix.

Let $v_1 = r_0/\beta$ where $\beta = \|r_0\|_2$. The residual r_n associated with the approximate solution (2.3) satisfies

$$\begin{aligned} r_n &= b - Ax_n = b - A(x_0 + V_n y) \\ &= r_0 - AV_n y = r_0 - V_{n+1}\underline{H}_n y \\ &= \beta v_1 - V_{n+1}\underline{H}_n y \\ &= V_{n+1}(\beta e_1 - \underline{H}_n y). \end{aligned}$$

Since V_{n+1} is a matrix with orthonormal columns, the residual norm $\|r_n\|_2 = \|\beta e_1 - \underline{H}_n y\|_2$ is minimized when y solves the linear least–squares problem

$$\min_{y \in \mathbb{C}^n} \|\beta e_1 - \underline{H}_n y\|_2. \tag{2.5}$$

We will denote by y_n the solution of (2.5). Therefore, $x_n = x_0 + V_n y_n$ is an approximate solution of (2.2) for which the residual is minimized over $\mathcal{K}_n$. GMRES owes its name to this minimization property that is its key feature as it ensures the decrease of the residual norm.

In exact arithmetic, GMRES converges in at most m steps. However, in practice, m can be very large and the storage of the orthonormal basis V_n may become prohibitive. Also the orthogonalization of v_n on the previous vectors V_{n-1} requires $2mn$ flops, for large n, the computational cost of the orthogonalization scheme may become too expensive. The restarted GMRES method is designed to cope with these two drawbacks. Given a fixed $n_{\max}$, the restarted GMRES method computes a sequence of approximate solutions x_n until x_n is acceptable or $n = n_{\max}$. If the solution is not found, then a new starting vector is chosen on which GMRES is again applied. Often, GMRES is restarted from the last computed approximation, i.e. $x_0 = x_{n_{\max}}$ to comply with the monotonicity property even when restarting. The process is iterated until a good enough approximation is found. We denote by GMRES($n_{\max}$) the restarted GMRES algorithm for a projection size of at most $n_{\max}$.
This concludes the theroretical background of the GMRES method. In the following paragraphs, we enlight the key–points for an efficient implementation of the GMRES method:

- the solution of the least-squares problem (2.5),
- the construction of the orthonormal basis V_n,
- the stopping criteria for the iterative scheme, and
- the calculation of the residual at the restart.

2.2.1.1 The least-squares problem

At each step n of GMRES, one needs to solve the least-squares problem (2.5). The matrix $\underline{H}_n$ involved in this least-squares problem is an $(n+1) \times n$ complex matrix which is upper Hessenberg. We wish to use an efficient algorithm for solving (2.5) which exploits the structure of $\underline{H}_n$.

First, we base the solution of (2.5) on the QR factorization of the matrix $[\underline{H}_n,\ \beta e_1]$: if $Q\Gamma = [\underline{H}_n,\ \beta e_1]$ where Q is a matrix with orthogonal columns and $\Gamma = (\gamma_{ik})$ is an $(n+1) \times (n+1)$ upper triangular matrix, then the solution y_n of (2.5) is given by

$$y_n = \Gamma(1:n, 1:n)^{-1} \Gamma(1:n, n+1). \tag{2.6}$$

Here, $\Gamma(1:n,1:n)$ denotes the first $n \times n$ submatrix of Γ and $\Gamma(1:n,n+1)$ stands for the last column of Γ. Moreover, it is easy to see that

$$\|r_n\|_2 = \|b - Ax_n\|_2 = \|\beta e_1 - \underline{H}_n y_n\|_2 = |r_{n+1,n+1}|. \tag{2.7}$$

Therefore, the norm of the residual of the linear system is a by product of the algorithm and can be obtained without explicitly evaluating the residual vector.

The QR factorization of upper Hessenberg matrices can be efficiently performed using Givens rotations, because they enable us to zero out all elements $\underline{H}_{k+1,k}$, $k = 1, \ldots n$, sequentially. However, since $[\underline{H}_{n+1},\ \beta e_1]$ is obtained from $[\underline{H}_n,\ \beta e_1]$ by adding one column c, the R–factor Γ_{n+1} of $[\underline{H}_{n+1},\ \beta e_1]$ is obtained by updating the R–factor Γ_n of $[\underline{H}_n,\ \beta e_1]$ using an algorithm that we briefly outline now, for $n = 3$ (see [12, 14, 16]):

1. Let
$$\Gamma_n = \begin{pmatrix} + & + & + & + \\ 0 & + & + & + \\ 0 & 0 & + & + \\ 0 & 0 & 0 & + \end{pmatrix}$$
and $Q_k \in C^{(n+1)\times(n+1)}$ be such that $[\underline{H}_n,\ \beta e_1] = Q_k \Gamma_k$. The matrix Q_k is not explicitly computed, only the sine and cosine of the Givens rotations are stored. The vector $w = Q_k^H c$ is then computed by applying the stored Givens rotations, and w is inserted in between the n and $n+1$ columns of Γ_k, to yield
$$\widetilde{\Gamma}_n = \begin{pmatrix} + & + & + & * & + \\ 0 & + & + & * & + \\ 0 & 0 & + & * & + \\ 0 & 0 & 0 & * & + \\ 0 & 0 & 0 & * & 0 \end{pmatrix}$$

2. A Givens rotation that zeros element $\widetilde{\Gamma}_n(n+2, n+1)$ is computed and applied to $\widetilde{\Gamma}_n$ to produce the matrix
$$\Gamma_{n+1} = \begin{pmatrix} + & + & + & + & + \\ 0 & + & + & + & + \\ 0 & 0 & + & + & + \\ 0 & 0 & 0 & + & + \\ 0 & 0 & 0 & 0 & + \end{pmatrix}$$

The computation of the sine and cosine involved in the givens QR factorization use the BLAS routines *ROTG, and we refer the reader to [12, 16] for questions related to the reliability of these transformations.

2.2.1.2 Evaluation of the norm of the residual

Thanks to equality (2.7), we see that the 2–norm of the residual is given directly in the algorithm during the solution of the least-squares problem. Therefore, the

backward error can be obtained at a low cost and we can use

$$\eta_{A,n} = \frac{|\gamma_{n+1,n+1}|}{\alpha\|x_n\|_2 + \beta}$$

as the stopping criterion of the GMRES iterations. However, it is well known that, in finite precision arithmetic, the computed residual (2.7) given from the Arnoldi process may differ significantly from the true residual. Therefore, it is not safe to use $\eta_{A,n}$ exclusively as the stopping criterion. Our strategy is the following: first we iterate until $\eta_{A,n}$ becomes lower than the tolerance, then afterwards, we iterate until η_n becomes itself lower than the tolerance. We hope in this way to minimize the number of explicit residual computations (involving the computation of matrix-vector products) necessary to evaluate η_n, while still having a reliable stopping criterion.

2.2.1.3 Computation of V_n

The quality of the orthogonality of the V_n plays a central role in GMRES as deteriorating it might slow down or delay the convergence. On the other hand, ensuring very good orthogonality might be expensive and useless for some applications. Consequently a trade-off has to be found to balance the numerical efficiency of the orthogonalization scheme and its inherent efficiency on a given target computer.
Most of the time, the Arnoldi algorithm is implemented through the modified Gram-Schmidt (MGS) process for the computation of V_n and H_n. However, in finite precision arithmetic, there might be a severe loss of orthogonality in the computed basis; this loss can be compensated by selectively iterating the orthogonalization scheme (see Section 1.5). The resulting algorithm is called iterative modified Gram-Schmidt (IMGS). The drawback of IMGS is the increased number of dot products. The classical Gram-Schmidt (CGS) algorithm can be implemented in an efficient manner by gathering the dot products into one matrix–vector product, but it is well known that CGS is numerically worse than MGS. However, CGS with selective reorthogonalization (ICGS) results in an algorithm of the same numerical quality as IMGS. Therefore, ICGS is particularly attractive in a parallel distributed environment, where the global reduction involved in the computation of the dot product is a well-known bottleneck [49, 52, 88, 122].
In our GMRES implementation, we have chosen to give the user the possibility of using any of the four different schemes quoted above : CGS, MGS, ICGS and IMGS. We follow [113] to define the criterion for the selective reorthogonalization and set $K = \sqrt{2}$ as suggested by [34] as the value for the threshold.

2.2.1.4 Computation of the residual at restart

In most of the applications, the computation of each matrix–vector product can be extremely expensive compared to the other operations of the GMRES process. In that case, one would like to avoid the explicit calculation of the residual at each restart of GMRES. Since we then set $x_0 = x_n$, we have $r_0 = b - Ax_n$ with

$x_n = x_0 + V_n y$. We can then observe that

$$\begin{aligned} r_0 &= b - A(x_0 + V_n y_n) \\ &= V_{n+1}(\beta e_1 - \underline{H} y_n) \\ &= V_{n+1} Q_n (Q_n^H \beta e_1 - \begin{bmatrix} \Gamma(1:n, 1:n) \\ 0 \end{bmatrix} y_n) \\ &= V_{n+1} Q_n \begin{bmatrix} 0 \\ \gamma_{n+1,n+1} \end{bmatrix}. \end{aligned}$$

It follows that the calculation of the residual amounts at computing a linear combination of the $(n+1)$ Arnoldi's vectors. The coefficients of the linear combination are computed by applying the Givens rotations in the reverse order to the vector which has all its entries equal to zero but the last that is equal to $r_{n+1,n+1}$. This non–zero value is a by product of the solution of the least–square problem. This calculation of the residual requires $m(2n+1)+2n$ floating–point operations (flops) and should be preferred to an explicit calculation if the matrix–vector product involving A implies more than $2m(n+1)$ flops. We should mention that in some circumstances, for instance when the required backward error is close to the machine precision, the use of this trick might slightly delay the convergence (while it might still enable us to get the solution in a shorter period of time). Notice that the implementation of this trick requires to store $(n+1)$ Arnoldi's vectors, while only n have to be stored otherwise. For the sake of robustness, even if this calculation of the residual is selected by the user, we enforce an explicit residual calculation if, in the previous restart, the convergence was detected by $\eta_{A,n}^P$ but not assessed by η_n^P.

2.3 The flexible GMRES method

In 1993, Saad [117] introduced a variant of the GMRES method with right preconditioning that enables the use of a different preconditioner at each step of the Arnoldi process.
In the sequel, we start by briefly describing the standard GMRES algorithm with right preconditioning and then show the modification which allows to use a different preconditioning at each GMRES iteration.
The GMRES algorithm with right preconditioning solves the modified system: $(AM)y = b$ and the solution of the system $Ax = b$ is set with $x = My$. The Arnoldi process constructs an orthonormal basis V_n of the preconditioned Krylov subspace:

$$\mathcal{K}_n = \text{span}\left\{r_0, AMr_0, \ldots, (AM)^{n-1}r_0\right\}.$$

and V_n is such that $AMV_n = V_{n+1}\underline{H}_{n+1,n}$. The approximate solution given by GMRES is in this case given via

$$x_n = x_0 + MV_n y_n,$$

where y_n is given by equation (2.5). The GMRES method with right preconditioning updates the solution using a linear combination of the preconditioned vectors $z_i = Mv_i$. When all these vectors are obtained by applying the same matrix M to the v_i we do not need to store them and the term $MV_n y_n$ is given using the associativity relation $(MV_n)y = M(V_n y)$. First we compute $V_n y$, then we apply the preconditioner.
In the flexible version of GMRES, referred to as FGMRES, the preconditioner *varies* at each step. The update of the solution x is done at the price of storing the sequence of vectors $z_i = M_i v_i$.
The only difference with the classical GMRES is that we have to store the preconditioned vectors z_i and perform the update of the solution using these vectors.
We do not pursue furthermore the description of this algorithm but refer to [117] for a complete exposition of the convergence theory; we only notice that contrary to the classical GMRES a general convergence theorem cannot be proved.

2.4 The GMRES method with Deflated Restarting

It is well known that the convergence of Krylov subspace methods for linear equations depends to a large degree on the distribution of eigenvalues. Some small eigenvalues in the spectrum can potentially slow down the convergence rate. Indeed a clustered spectrum is a highly desirable property for the rapid convergence of Krylov methods. In exact arithmetic the number of distinct eigenvalues would determine the maximum dimension of the Krylov subspace. If the diameters of the clusters are small enough, the eigenvalues within each cluster behave numerically like a single eigenvalue, and we would expect less iterations of a Krylov method to produce reasonably accurate approximations. Theoretical studies have related superlinear convergence of GMRES to the convergence of Ritz values [133]. Basically, convergence occurs as if, at each iteration of GMRES, the next smallest eigenvalue in magnitude is removed from the system. As the restarting procedure destroys information about the Ritz values at each restart, the superlinear convergence may be lost. Thus removing the effect of small eigenvalues in the preconditioned matrix can have a beneficial effect on the convergence. Note that there are exceptions [68, 95].

Kharchenko and Yeremin [81] built a preconditioner for the matrix using approximate eigenvectors. Their preconditioner is based on a sequence of rank–one updates that involve the left and right smallest eigenvectors. The method is based on the idea of translating isolated eigenvalues consecutively group by group into a vicinity of one using low–rank projections. After each restart of GMRES($n_{\max}$), approximations to the isolated eigenvalues to be translated are computed by the Arnoldi process. The isolated eigenvalues are translated towards one, and the next cycle of GMRES(m) is applied to the transformed matrix. The effectiveness of this method relies on the assumption that most of the eigenvalues of A are clustered close to one in the complex plane. Erhel, Burrage and Pohl [46] developed a different preconditioner. The preconditioner is based on a deflation technique such that the linear system is solved exactly in an invariant subspace of dimension r corresponding to the smallest r eigenvalues of A. This is improved upon by Burrage and Erhel [20] with a method that keep improving the quality of the approximate eigenvectors, this algorithm is called DEFLATION. A more general formulation of this preconditioner is given by Carpentieri, Duff and Giraud [24]; it is described in detail in Section 3.3.2.3. These three approaches use approximate eigenvectors generated during one GMRES cycle (or an eigensolver used in a preprocessing phase). Morgan [93] compared DEFLATION and GMRES–DR and concluded that, on his examples, DEFLATION in the best case does as well as GMRES–DR and in some cases GMRES–DR performs clearly better, this is explained by the fact that the approximate eigenvectors given by GMRES–DR are more accurate than the ones computed by DEFLATION.

Information from the invariant subspace associated with the smallest eigenvalues and its orthogonal complement are used to construct a preconditioner in the approach proposed by Baglama, Calvetti, Golub and Reichel [8]. The algorithm proposed uses the recursion formulae of the implicitly restarted Arnoldi (IRA) method described by Sorensen [127]. In this way, the first k columns V_k of the Krylov space spanned by V_n shall approximate the k eigenvectors associated with the k smallest eigenvalues.

In [8] the matrix

$$M = V_k(I_m + (V_k^H A V_k)^{-1})V_k^H.$$

is used as a left preconditioner. Note that this formulation for the preconditioner is the same as [46, 24], the difference is that in this latter situation the preconditioner is updated at each restart by extracting new eigenvalues which are the smallest in magnitude. Le Calvez and Molina [21] as well as Morgan [92] proposed algorithms that are also based on IRA. The method proposed by Morgan [92] is called GMRES–IR.

Wu and Simon [138] developed a method called thick–restart Lanczos for solving symmetric eigenvalue problem. Morgan [92] adapted this work for solving unsymmetric linear system, the resulting method is called GMRES–DR. In a sense, the GMRES–DR is to the thick–restart Lanczos what GMRES–IR is to IRA.

We present here some details in our implementation of the GMRES–DR method. A full description will be available in a Users's Guide that is under writing. The GMRES–DR algorithm is given in Algorithm 10.

Algorithm 10 The GMRES–DR algorithm.

1. *Start.* Choose n, the maximum size of the subspace, and k, the desired number of approximate eigenvectors. Choose an initial guess x_0 and compute $r_0 = b - Ax_0$. The recast problem is $A(x - x_0) = r_0$. Let $v_1 = r_0/\|r_0\|$ and $\beta = \|r_0\|$.

2. *First cycle.* Apply standard GMRES(n): generate V_{n+1} and $\underline{H}_{n+1,n}$ with the Arnoldi iteration, solve $\min \|c - \underline{H}_{n+1,n}d\|$ for d, where $c = \beta e_1$, and form the new approximate solution $x_n = x_0 + V_n d$. Let $\beta = h_{n+1,n}$, $x_0 = x_n$, and $r_0 = b - Ax_n$. Then compute the k smallest (or others, if desired) eigenpairs $(\tilde{\theta}_i, \tilde{g}_i)$ of $H_n + |\beta|^2 H_n^{-H} e_n e_n^H$. (The $\tilde{\theta}_i$ are the harmonic Ritz values.)

3. *Orthonormalization of the first k vectors.* Orthonormalize $\tilde{g}_i$'s, first separating into real and imaginary parts if complex, in order to form an n–by–k matrix P_k. (It may be necessary to adjust k in order to make sure that both real and imaginary parts of complex vectors are included.)

4. *Orthonormalization of the $(k+1)$-th vector.* First extend $p_1, \ldots, p_k$ (the columns of P_k) to length $(n+1)$ by appending a zero entry to each. Then orthonormalize the vector $c - \underline{H}_{n+1,n}d$ against them to form p_{k+1}. Note $c - \underline{H}_{n+1,n}d$ is the length $(n+1)$ vector corresponding to the GMRES residual vector. P_{k+1} is $(n+1)$–by–$(k+1)$.

5. *Form portions of new H and V using the old H and V.* Let $\underline{H}^{\text{new}}_{k+1,k} = P_{k+1}^H \underline{H}_{n+1,n} P_k$ and $V^{\text{new}}_{k+1} = V_{n+1} P_{k+1}$. Then let $\underline{H}_{k+1,k} = \underline{H}^{\text{new}}_{k+1,k}$ and $V_{k+1} = V^{\text{new}}_{k+1}$.

6. *Reorthogonalization of the $(k+1)$-th vector.* Orthogonalize v_{k+1} against the preceding columns of the new V_{k+1}.

7. *Arnoldi iteration.* Apply the Arnoldi iteration from v_{k+1} to form the rest of V_{n+1} and $\underline{H}_{n+1,n}$. Let $\beta = h_{n+1,n}$.

8. *Form the approximate solution.* Let $c = V^H r_0$ and solve $\min \|c - \underline{H}_{n+1,n}d\|$ for d. Let $x_n = x_0 + V_n d$. Compute the residual vector $r = b - Ax_n = V_{n+1}(c - \underline{H}_{n+1,n}d)$. Check $\|r\| = \|c - \underline{H}_{n+1,n}d\|$ for convergence and proceed if not satisfied.

9. *Eigenvalue computations.* Compute the k smallest (or others, if desired) eigenpairs $(\tilde{\theta}_i, \tilde{g}_i)$ of $H_n + |\beta|^2 H_n^{-H} e_n e_n^H$.

10. *Restart.* Let $x_0 = x_n$ and $r_0 = r$. Go to 3.

2.4.1 Use of the Givens rotations.

Classically, we use the Givens rotations to obtain the QR–factorization of the Hessenberg matrix $\underline{H}_{n+1,n}$ in the GMRES cycle at step 2 and 7. This is already described in Section 2.2.1.1. In particular the use of the Givens rotations enables the user to know the norm of the residual at a low cost during the iterations. In the GMRES–DR algorithm, the properties of the Givens rotations can be used to efficiently compute the matrix

$$H_n + |\beta|^2 H_n^{-H} e_n e_n^H \tag{2.8}$$

needed at step 2 and 9.

We recall that at step n, the QR–factorization of $\underline{H}_{n+1,n}$ writes

$$\underline{H}_{n+1,n} = \Theta_{n+1} \begin{pmatrix} T_n \\ 0_{1,n} \end{pmatrix}.$$

The matrix Θ_{n+1} is unitary of order $(n+1)$ and T_n is upper triangular of order n. The matrix Θ_{n+1} consists in the product of the n Givens rotations. Let us define the matrix U_n so that

$$U_n = \begin{pmatrix} I_n & 0_{n,1} \end{pmatrix} \Theta_{n+1} \begin{pmatrix} I_n \\ 0_{1,n} \end{pmatrix}.$$

U_n is the matrix of order n made with n first row and n first columns of Θ_{n+1} We recall that the line $n+1$ of of Θ_{n+1} has $n-1$ zeros and the n–th term is $-\sin_n$, so we can write

$$\Theta_{n+1} \begin{pmatrix} I_n \\ 0_{1,n} \end{pmatrix} = \begin{pmatrix} U_n \\ 0_{1,n-1} \quad -\sin_n \end{pmatrix}$$

Since $\begin{pmatrix} I_n & 0_{n,1} \end{pmatrix} \Theta_{n+1}^H \Theta_{n+1} \begin{pmatrix} I_n \\ 0_{1,n} \end{pmatrix} = I_n$, we have

$$\begin{pmatrix} U_n^H & 0_{n-1,1} \\ & \overline{\sin_n} \end{pmatrix} \begin{pmatrix} U_n \\ 0_{1,n-1} \quad -\sin_n \end{pmatrix} = I_n$$

So that

$$U_n^H U_n = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & \cos_n^2 \end{pmatrix} \tag{2.9}$$

Concerning the matrix H_n, we have

$$\begin{aligned} H_n &= \begin{pmatrix} I_n & 0_{n,1} \end{pmatrix} \underline{H}_{n+1,n} \\ &= \begin{pmatrix} I_n & 0_{n,1} \end{pmatrix} \Theta_{n+1} \begin{pmatrix} I_n \\ 0_{1,n} \end{pmatrix} T_n \\ &= U_n T_n \end{aligned} \tag{2.10}$$

Back to equation (2.8), and using results (2.9) and (2.10) we obtain

$$\begin{aligned} H_n + |\beta|^2 H_n^{-H} e_n e_n^H &= H_n + |\beta|^2 U_n^{-H} T_n^{-H} e_n e_n^H, \\ &= H_n + |\beta|^2 U_n \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & (\cos_n^2)^{-1} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ (\overline{t_{n,n}})^{-1} \end{pmatrix} e_n^H, \\ &= H_n + |\beta|^2 U_n \begin{pmatrix} 0 \\ \vdots \\ 0 \\ (\cos_n^2 \overline{t_{n,n}})^{-1} \end{pmatrix} e_n^H. \end{aligned}$$

Also note that the residual $c - \underline{H}_{n+1,n} d$ needed at step 4 can also be obtained via

$$c - \underline{H}_{n+1,n} d = \Theta_{n+1} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \gamma_{n+1,n+1} \end{pmatrix}.$$

2.4.2 Use of Householder transformations.

In order to obtain the estimate of the current residual at each step of the GMRES cycle in step 7. The minimization of step 8 is performed using one Givens rotation per column on the matrix $\underline{H}_{n+1,n}$ for $j = k+1, \ldots, n$. This requires to have the QR–factorization of the first columns $j = 1, \ldots, p$. Therefore before the GMRES cycle, we perform a QR–factorization of the first p columns of $\underline{H}_{n+1,n}$. We choose to use Householder transformations to triangularize the $(p+1)$–by–p non zeros block $\underline{H}_{p+1,p}$.
The i–th Householder transformation J_i writes

$$J_i = I_{p+1} - 2\beta_i y_i y_i^H,$$

and is fully determined by the vector y_i and the modulus of β_i. For the sake of simplicity of exposure, we choose β_i real. Let Q_k denote the orthogonal factor associated with the first k Householder transformations. We show below how to efficiently compute $Q_k V_{j+1}^H r_0$.
First of all, $r_0 \in V_{k+1}$ (see [93]). As the vectors V_{j+1} are orthonormal, $r_0 \perp V_{k+2,\ldots n}$, we do not need to compute the $k+2, \ldots, n$ entries of $c = V_{j+1}^H r_0$, they are set to zero. Consequently, the $k+2, \ldots, n$ entries of $Q_k V_{j+1}^H r_0$ are also set to zero.
It is also possible to show that the k first of $Q_k V_{j+1}^H r_0$ are 0. This requires all the results given by Morgan [93]. For a detailed description we refer to the forthcoming Users' Guide. Finally it appears that the only nonzero in $Q_k V_{j+1}^H r_0$ is its $(k+1)$–th entry, and its modulus is $\|r_0\|_2$, since it is real and positive, we set it directly from one cycle to the next

$$Q_k V_{j+1}^H r_0 = e_{k+1} \|r_0\|_2,$$

where e_k is the k–th vector of the canonical base.

2.4.3 The LU–matrix–matrix product

The GMRES–DR(n, k) method is always compared with GMRES(n) since both methods use a Krylov spaces of dimension n. However at step 5. of Algorithm 10, if the operation

$$V_{k+1}^{\text{new}} \longleftarrow V_{n+1} P_{k+1}, \tag{2.11}$$

is performed in a classical way (e.g. subroutine zgemm of the BLAS in double complex arithmetic) it requires $(n+k+2)$ vectors of size m. For that reason, we propose to use the LU–matrix–matrix product so that operation (2.11) only requires $(n+1)$ vectors of size m.
The standard matrix–matrix product is given in Algorithm 11. The cost is $(2n -$

Algorithm 11 standard matrix–matrix product algorithm.

```
1. for i = 1 : m,
2.     for j = 1 : k,
3.         w_ij = v_in h_nj
4.         for l = 1 : n − 1,
5.             w_ij = w_ij + v_il h_lj
6.         end for
7.     end for
8. end for
```

1)mk flops and the algorithm needs the workspace for V and W, that is $(n+k)$ vectors of size m.
In the case where $k < n$, the k first columns of V are not needed anymore after the matrix–matrix product, consequently another algorithm is possible. For the sake of generality and without loss of generality, we consider that the k first columns of V are not needed after the product. The method is as follows. First an LU–factorization of the matrix H is performed, then the product of V and L is performed and the solution is stored in the k first columns of V. This is again possible thanks to the triangular structure in L. Then the k first columns of V are multiplied with U and the resulting matrix is stored on V. This is possible thanks to the triangular structure in L. The algorithm is given in Algorithm 12.

Algorithm 12 LU–matrix–matrix product algorithm

```
1. LU-factorization
2. for p = 1 : k,
3.     if h_pp = 0 then stop
4.     for i = p + 1 : m,
5.         η = h_ip / h_pp
6.         for j = p + 1 : k,
7.             h_ij = h_ij − η h_pj
8.         endfor
9.     h_ip = η
10.    endfor
11. endfor
12. product V(1 : m, 1 : k) ← V(1 : m, 1 : n)L(1 : n, 1 : k)
13. for j = 1 : k,
14.     for i = j + 1 : n,
15.         v_j = v_j + v_j h_ij
16.     endfor
17. endfor
18. product V(1 : m, 1 : k) ← V(1 : m, 1 : k)L(1 : k, 1 : k)
19. for j = k : 1,
20.     v_j = v_j hjj
21.     for i = 1 : j − 1,
22.         v_j = v_j + v_j h_ij
23.     endfor
24. endfor
```

The total cost is $(2n-1)mk + k^2(n-k) + k(k-1)(\frac{2k}{3} + \frac{1}{6})$ flops, and the algorithm does not need any extra vector a part from the n columns of V. The extra costs

compare to Algorithm 11 are mainly governed by the LU–factorization of the n–by–k matrix H.
The LU–matrix–matrix product algorithm is particulary interesting when a matrix–matrix product like the one in (2.11) has to be performed with k and n small. Note that the particularity of operation (2.11) is that the first k columns of V are not needed anymore once the matrix–matrix product has been performed.

2.4.4 Preliminary experimental results

In order to validate our GMRES–DR implementation, we use the test matrices presented in Morgan [93] and cross-check our results. The test examples used here are real, we then use the real double precision version of the GMRES–DR package. In the next chapter, the matrices arising from electromagnetism are complex so that the (double) complex version of the solvers is experimented.
The first matrix we considered is the example 1 of [93]. The GMRES–DR(15,5) method is run on the matrix SAYLR4 from the Matrix Market[1]. It is described as a Saylor's petroleum engineering/reservoir simulation matrix arising from a 3D reservoir simulation on a grid (33x6x18). The matrix is of order 3564 with 22316 entries and an incomplete LU factorization with no fill–in is used as preconditioner. Note that the original matrix is symmetric but the preconditiner is not. The right–hand side b is random. In Figure 2.1, GMRES–DR(15,5) is compared with full GMRES and GMRES(15). We observe that, for the same amount of stored vectors and matrix–vector products, GMRES–DR(15,5) clearly outperforms GMRES(15) and is close to exhibit the same convergence behaviour as full GMRES.

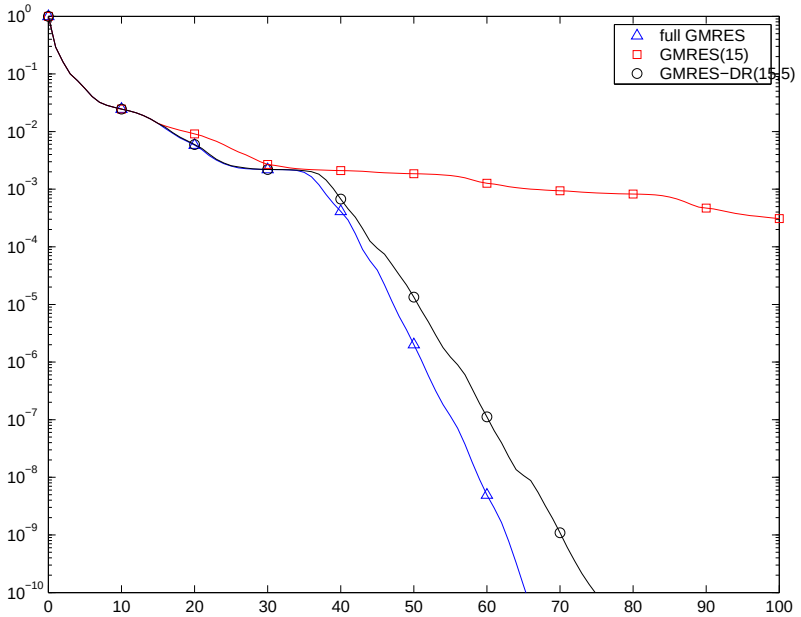


Figure 2.1: GMRES–DR(15,5), full GMRES and GMRES(15) are run on SAYLR4, a matrix of order 3564 from Matrix Market.

[1] http://math.nist.gov/MatrixMarket/

The second matrix we consider is the example 3 of [93]. The GMRES–DR(25,6) method is run on the bidiagonal matrix with entries 0.01, 0.1, 1, 2, $\ldots$, 997, 998 on the main diagonal and 1's on the super diagonal. The right–hand side has all 1's. No preconditioner is used. We shall point out that not only the GMRES–DR has been implemented in the four arithmetics but also the FOM–DR [93] algorithm. In Figure 2.2, we display the convergence history of six solvers applied to this matrix. We note that the peaks of FOM–DR(25,6) (resp. full FOM, FOM(25)) coincide with the plateaus of GMRES–DR(25,6) (resp. full GMRES, GMRES(25)) and that the FOM variants behave globally as their GMRES counterpart. On that matrix

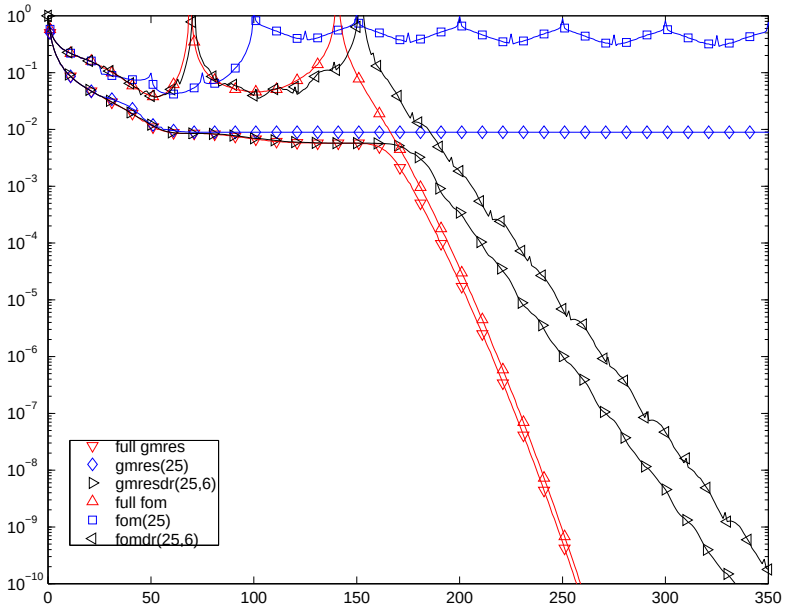


Figure 2.2: A comparison of MINRES solvers (GMRES's) and Galerkin projection solvers (FOM's) on the bidiagonal matrix with entries 0.01, 0.1, 1, 2, $\ldots$, 997, 998 on the main diagonal and 1's on the super diagonal.

the deflated restarting strategy improves the classical restarted method but is still outperformed by the full approach for the two solvers.

2.5 The seed–GMRES method

The principle of the Krylov subspace methods is to construct a Krylov subspace and then to search for an approximate solution within this subspace that complies with some optimality criterion. It is often admitted that the main effort resides in the construction of the Krylov subspace. Indeed to generate a subspace of size n one needs to perform at least $(n-1)$ matrix–vector products.
When dealing with multiple right–hand sides, the general governing idea is to use the Krylov subspace associated with one right–hand side to satisfy not only its associated optimum criterion but also the ones of the other right–hand sides. This process is referred to as the seed–variants of the Krylov subspace methods. We focus in this section on the seed–variant of GMRES, denoted by seed–GMRES. The algorithm with right preconditioning is given in Algorithm 13.

Algorithm 13 The seed–GMRES algorithm with right preconditioning and restart($n_{\max}$).

1. Choose n_{max} , the maximum size of the subspace. For each right–hand side, $b^{(\ell)}$, choose an initial guess $x_0^{(\ell)}$ and compute $r_0^{(\ell)} = b^{(\ell)} - Ax_0^{(\ell)}$. The recast problem is the error system $A(x^{(\ell)} - x_0^{(\ell)}) = r_0^{(\ell)}$, choose a right–hand side k , the first on which a cycle of GMRES is applied. Set $Z_p = (z_1, \ldots, z_p)$, p vectors of size m , to zero.

2. run one cycle of GMRES($n_{\max}$) with right preconditioning on $r_0^{(k)}$, either it converges in $n = n_1$ step or stops after $n = n_{\max}$ steps. This GMRES cycle provides us with V_{n+1} , that has orthonormal columns, and $\underline{H}_{n+1,n}$ such that $AV_n = V_{n+1}\underline{H}_{n+1,n}$. The approximate solution for the k-th system writes $x_n = x_0^{(k)} + MV_n y_n^{(k)}$ but is not computed as such. The vector z_k is updated via $z_k = z_k + V_n y_n^{(k)}$ and the residual can be formed via $r_n^{(k)} = r_0^{(k)} - V_{n+1}\underline{H}_{n+1,n} y_n^{(k)}$. If the k-th system has converged, form the approximate solution $x^{(k)} = x_0^{(k)} + Mz_k$. If all the systems have converged, stop.

3. For each right–hand side $\ell \neq k$ that has not converged, form $c^{(\ell)} = V_{n+1}^T r_0^{(\ell)}$ and compute $y_n^{(\ell)}$ the solution of the least squares problem $\min_y \|\underline{H}_{n+1,n} y - c^{(\ell)}\|_2$. Compute $z_\ell = z_\ell + V_n y_n^{(\ell)}$ and the residual $r_n^{(\ell)} = (I_m - V_{n+1}V_{n+1}^T) r_0^{(\ell)} + V_{n+1}(c^{(\ell)} - \underline{H}_{n+1,n} y_n^{(\ell)})$. If the system ℓ has converged, form the approximate solution $x^{(\ell)} = x_0^{(\ell)} + Mz_\ell$. If all the systems have converged, stop.

4. For each right–hand side ℓ that has not converged, set $r_0^{(\ell)} = r_n^{(\ell)}$. Choose a vector to run a cycle of GMRES. Traditionaly we take the first ℓ in the list $k, k+1, \ldots, p$ so that the system ℓ has not converged and go to step 2.

The design of Algorithm 13 has two particularities that aim at reducing the computational effort at a low extra storage cost. The right preconditioning is implemented via the storage of the vectors z_ℓ . The preconditioning operation that gives back the solution of the original system is performed only once at the end of the algorithm to form $x^{(\ell)} = x_0^{(\ell)} + Mz_\ell$. An alternative would be to compute the approximate solutions $x_n^{(\ell)} = x_0^{(\ell)} + MV_n y_n^{(\ell)}$ after each minimization. We have rather chosen to store the block vector z of size m–by–p rather than to perform a preconditioning step at each minimization.
At step 3, an updated residual $r_0^{(\ell)}$ is needed. This residual can be computed

either with the explicit formula $r_0^{(\ell)} = b - Ax_0^{(\ell)}$, or with the implicit computation $r_n^{(\ell)} = (I_m - V_{n+1}V_{n+1}^T)r_0^{(\ell)} + V_{n+1}(c^{(\ell)} - \underline{H}_{n+1,n}y_n^{(\ell)})$. We have chosen to store the block vector r_0 of size m–by–p and to perform the implicit computation rather than to perform a matrix–vector product at each minimization as for the explict computation.

From a memory point of view, we need a total storage of $n_{\max} + 4p$ vectors; namely p right–hand sides (b), p initial guesses (x_0), p residuals (r_0), p vectors (z) and the $n_{\max}$ Krylov vectors (v). The extra storage compared to a classical approach with the explicit updates is $2p$ vectors. In large calculation, this extra storage is largely compensated by the saved computational time.

2.6 The block–GMRES method

In many circumstances, it is desirable to work with a block of vectors instead of a single vector. This can be achieved by using the block generalizations of the Krylov subspaces methods, for which A always operates on a group of vectors instead of on a single vector. Moreover the block generalizations of Krylov subspaces methods enable the vectors to share their Krylov space with each other, consequently the convergence is expected to occur sooner.
Each Krylov methods has its block variant. O'Leary [97] gave the block variants for the Conjugate Gradient, Freund and Malhotra [55] derived the block–QMR algorithm, etc ... In this section, and in the manuscript in general, we only focus on the block–GMRES method. The origin of the block–GMRES method is often attributed to Vital [134]. Saad [119] also gives a description of the algorithm.

2.6.1 General overview of the block–Arnoldi method

The block generalization of the Arnoldi algorithm can be simply described as the Arnoldi algorithm for a single vector where all the single vectors are replaced by block vectors of size p. The entries of the Hessenberg matrix in the Arnoldi algorithm are replaced by blocks of size p–by–p, and the normalization in the Gram–Schmidt version is replaced by a factorization. At each step n, this factorization is in general either based on an SVD, in that case the Hessenberg matrix $H_{n+p,n}$ has a block structure with square block of size p and is block Hessenberg with a block bandwith of 1; or based on a QR–factorization, in which case the matrix $H_{n+p,n}$ is Hessenberg with a bandwidth p. Starting with an initial block V_p with orthonormal columns, the block–Arnoldi algorithm constructs in $s = n/p$ steps, the vectors V_{n+p} such that

$$\begin{aligned} V_{n+p}^H V_{n+p} &= I_{1:n+p}, \\ AV_n &= V_{n+p} H_{n+p,n}. \end{aligned}$$

The vectors V_n span the block–Krylov space

$$\mathcal{K}_n(A, V_p) = \text{span}(V_p, AV_p, A^2V_p, \ldots, A^{s-1}V_p).$$

2.6.2 Ruhe's variant of block–GMRES

We can also define the block–Krylov space $\mathcal{K}_n(A, V_p)$ when n is not a multiple of p. From the Euclidean division in $\mathbb{N}$, we define s_1 and s_2, the two unique integers such that $s_1 \geq 0$, $p > s_2 \geq 0$ and $n = s_1 p + s_2$. The block–Krylov space $\mathcal{K}_n(A, V_p)$ is

$$\text{span}(V_p, AV_p, A^2V_p, \ldots, A^{s_1-1}V_p, A^{s_1}V_{s_2}).$$

This can be constructed using the Ruhe's variant of block–Arnoldi algorithm [111] that considers the individual vector rather than a set of size p. In the Ruhe's variant, the factorization of the block that corresponds to the normalization step in single vector Arnoldi is necessarily a QR–factorization. In the remaining of our work, we take the notation of the Ruhe's variant. In particular, this means that

each step consists in increasing by one the dimension of the Krylov space instead of p in the general block–GMRES presentation [116].
We give a brief description of the block–GMRES algorithm. In this description, b, x, x_0, r_0 stands for the m–by–p matrices such that, for instance, $b = (b^{(\ell)})_{\ell=1,\ldots,p}$, where $b^{(\ell)}$ is the ℓ-th column of b.
If we have to solve the p linear systems,

$$Ax = b$$

with the initial guess x_0 then we set the initial residual to

$$r_0 = b - Ax_0.$$

The block–GMRES method is based on the block–Arnoldi algorithm as the GMRES method is based on the Arnoldi process. In a first step, we construct an orthonormal basis, V_p, for r_0 (we assume that r_0 has full rank) such that we have

$$r_0 = V_p\beta,$$

where β is a p–by–p matrix, $\beta = (\beta^{(\ell)})_{\ell=1,\ldots,p}$. Then the block–Arnoldi algorithm is developed with the starting block V_p. At each step n, the block–Krylov space $\mathcal{K}_n(A, b)$ is built and we minimize, for each ℓ, the least–squares problem:

$$\min_{x \in x_0^{(\ell)} + \mathcal{K}_n(A, r_0)} \|b^{(\ell)} - Ax\|_2. \tag{2.12}$$

The exact solutions of the problem are found in at most m steps. That requires at most m/p matrix–vector products per right–hand side. We mention that the p least–squares problems can also be writen in the following block form as a single least–squares problem

$$\min_{x \in x_0 + \mathcal{K}_n(A, r_0)} \|b - Ax\|_E, \tag{2.13}$$

where E denotes the Frobenius norm and $x_0 + \mathcal{K}_n(A, r_0)$ denotes the set of vectors $(x_0^{(\ell)} + \mathcal{K}_n(A, r_0))_{\ell=1,\ldots,p}$.
If p GMRES methods are performed simultaneously, then, after the step s, they have performed ps matrix–vector products and for each right–hand side ℓ, the residual is minimized on $\mathcal{K}_n(A, r_0^{(\ell)})$. At step $n = ps$ of the block–GMRES method, ps matrix–vector products have also been performed, for each right–hand side ℓ, the residual is minimized on $\oplus_s \mathcal{K}_s(A, r_0^{(\ell)}) = \mathcal{K}_n(A, r_0)$. Consequently the approximate solutions given by the block–GMRES method are expected to be better than the solutions given by the p individual GMRES since the residuals are minimized on a larger space.
When describing the block–GMRES algorithm, we assume that the p initial residuals r_0 are linearly independent. Given r_0, the p initial residuals, it may happen that they are linearly dependent. The remedy in that situtation is detailed in Section 3.6. For the remaining of this section we assume that r_0 has full rank.

2.6.3 The least–squares solution

In practice, the least squares problem (2.12) is solved as follows. The approximate solution at step n, $x_n \in x_0 + \mathcal{K}_n(A, r_0)$, writes $x_n = x_0 + V_n y_n$, where y_n is n-by–p, and we have

$$\begin{aligned} b - Ax_n &= r_0 - AV_n y_n, \\ &= V_{n+p}(\begin{pmatrix} \beta \\ 0_{n,p} \end{pmatrix} - H_{n+p,n} y_n). \end{aligned} \tag{2.14}$$

We perform p Givens rotations per column on the matrix $H_{n+p,n}$ to obtain its QR–factorization

$$H_{n+p,n} = \Theta_n \begin{pmatrix} T_n \\ 0_{p,n} \end{pmatrix}.$$

In equation (2.14), this gives

$$b - Ax_n = V_{n+p}\Theta_n(\begin{pmatrix} g_n \\ \tau_n \end{pmatrix} - \begin{pmatrix} T_n \\ 0_{p,n} \end{pmatrix} y_n),$$

where $\begin{pmatrix} g_n \\ \tau_n \end{pmatrix} = \Theta_n^H \begin{pmatrix} \beta \\ 0_{n,p} \end{pmatrix}$, g_n is an n–by–p matrix and τ_n is a p–by–p matrix. Since the matrices V_{n+p} and Θ_n have orthonomal columns, the p least squares problems (2.12) are solved by

$$x_n^{(\ell)} = x_0^{(\ell)} + V_n y_n^{(\ell)}$$

where $y_n^{(\ell)}$ corresponds to the ℓ-th solution of the triangular system $T_n y_n^{(\ell)} = g_n^{(\ell)}$. The block residual is $r_n = b - Ax_n$ and we have

$$r_n = b - Ax_n = V_{n+p}\Theta_n \begin{pmatrix} 0_{n,p} \\ \tau_n \end{pmatrix}.$$

Consequently, the norm of the residual associated with the system ℓ is

$$\|r_n^{(\ell)}\|_2 = \|\tau^{(\ell)}\|_2. \tag{2.15}$$

The p–by–p matrix τ is a by–product of the block–GMRES algorithm, and so equation (2.15) enables us to control the norm of the residuals without computing them explicitly. As we shall see, this is particulary useful to control the stopping criteria.
As GMRES, it may be necessary to restart the block–GMRES method. The generalization follows directly from the classical GMRES case.

2.6.4 $1/p$–happy breakdown in the block–GMRES algorithm.

The classical Arnoldi process may breakdown. The breakdown indeed occurs when $Av_n \in \mathcal{K}_{n-1}(A, r_0)$. This means that the Krylov space $\mathcal{K}_{n-1}(A, r_0)$ has reached its maximal size and is an invariant space. The name *breakdown* corresponds to the fact that, when the Gram–Schmidt orthogonalization process is used, the orthogonal

projection of Av_n on the orthogonal complement of $\mathcal{K}_{n-1}(A, r_0)$ is $w = 0$ and so the computation of the vector $v_{n+1} = w/\|w\|_2$ would result in a breakdown. However, this breakdown also implies that the solution of $Ax = b$ belongs to V_n, the solution is then found which gives rise to the name *happy breakdown*.
Similarly, the block–Arnoldi algorithm process may also breakdown. The consequences for the block–GMRES algorithm are not so heartening (indeed we show that they only are "$1/p$ happy").
As in the Arnoldi algorithm, the breakdown in the block–Arnoldi algorithm corresponds to the event

$$Av_n \in \mathcal{K}_{n-1}(A, r_0). \tag{2.16}$$

Let us examine the implications of statement (2.16) in the block–GMRES algorithm. We assume that no breakdown has occured until step n. This means that for all $j = 1, \ldots, n-1$, $h_{j+p,j} \neq 0$ and so

$$\text{Rank}\left(\begin{pmatrix} \beta \\ 0_{n-1,p} \end{pmatrix} H_{n+p-1,n-1}\right) = n + p.$$

At step n, the breakdown implies

$$AV_n = V_{n+p-1} H_{n+p-1,n}.$$

We also have

$$\text{span}\begin{pmatrix} \beta \\ 0_{n-1,p} \end{pmatrix} \cap \text{span}(H_{n+p-1,n}) = \text{span}(z^{(1)}),$$

where $z^{(1)}$ is a vector of size $(n + p - 1)$ that is defined by

$$z^{(1)} = \begin{pmatrix} \beta \\ 0_{n-1,p} \end{pmatrix} w^{(1)} = H_{n+p-1,n} t^{(1)}. \tag{2.17}$$

Multiplying equation (2.17) by V_{n+p-1} and denoting $u^{(1)} = V_n t^{(1)}$ we have

$$bw^{(1)} = Au^{(1)}.$$

When a breakdown occurs in the block GMRES algorithm, this implies that a linear combination $bw^{(1)}$ of the right–hand sides b has converged. Note that $t^{(1)} = y_n w^{(1)}$ so that the vectors $u^{(1)}$ verifies $u^{(1)} = V_n y_n w^{(1)} = x_n^{(1)} w^{(1)}$.
At step $n+1$, the vector v_{n+p} is generated using Av_{n+1}. Let p_n denotes the current bandwith at step n. We have $p_{n-1} = p$ and $p_n = p - 1$. Each time a breakdown occurs, the current bandwidth of the Hessenberg matrix decreases by one and a new linear combination of the right–hand sides has a solution in the Krylov space. A *new* linear combination has to be understood as a linear combination $w^{(n-p_n+1)}$ independent of the previous $(w^{(j)})_{j=1,\ldots,n-p_n}$. Eventually it happens that the current bandwdith at step n is 0; p independent linear combinations of the right–hand sides have a solution in the Krylov space, so all the sytems are solved.
The breakdown occurs at each n such that

$$h_{n+p_n,n} = 0. \tag{2.18}$$

In order to check whether a breakdown has occured, and to take into account the round–off errors, we implement the following criterion

$$|h_{n+p_n,n}| \leq \mathrm{tol}_{\mathrm{def}}/\|Av_n\|_2. \tag{2.19}$$

When this criterion is satisfied, we act as in the exact arithmetic case, the vector v_{n+p_n} is not generated from Av_n but from Av_{n+1}. The action of throwing away a vector in the Krylov sequence is called *deflation*.

2.6.5 Deflation in the residuals

The deflation of a Krylov vector in the block–GMRES algorithm implies that a linear combination of the right–hand sides has converged. Let us assume that the first deflation occurs at step n. This means that the rank of the columns of the residual $r_n = b - Ax_n$ is $(p-1)$ and we have

$$r_n w^{(1)} = (b - Ax_n)w^{(1)} = 0_m \quad \Leftrightarrow \quad \tau_n w^{(1)} = 0_p. \tag{2.20}$$

Another criterion for checking the breakdown in exact arithmetic is therefore

$$\sigma_{p_n}(\tau_n) = 0, \tag{2.21}$$

where $\sigma_p(\tau_n)$ denotes the largest singular value of τ_n. We recall that τ_n is a by–product of the algorithm.
When a deflation is detected in the residuals, for the block–QMR algorithm, Freund and Malhotra [55] extracted a residual from r_n (and τ_n). Let say the ℓ–th is extracted. They also store the solution $u^{(1)}$ that corresponds to the linear combination that has converged $u^{(1)} = V_n y_n w^{(1)}$ and the vectors $w^{(1)}$. Note that the vectors r_n and τ_n have now $p_n = (p-1)$ columns. Extracting a column of r_n is referred to as *deflation of residuals*. The process continues. Each time a breakdown occurs, a column is extracted. Eventually at step n, p breakdowns have occured and the process stops. Since

$$Au_n = bw_n,$$

the solution x_n is given via

$$x_n = u_n w^{-1}. \tag{2.22}$$

The p–by–p matrix w is triangular (up–to a permutation) and has full rank.
The deflation of the residuals introduced by Freund and Malhotra [55] for the block–QMR algorithm holds also for the block–GMRES algorithm. We have implemented and experimented it. It happens that the choice of ℓ, the residual to be deflated at step n, is important in order to have w well–conditioned. In that respect Freund and Nachtigal suggested to take ℓ such that $|w_\ell^{(n-p_n+1)}| = \max_j |w_j^{(n-p_n+1)}|$. In the block–QMR algorithm, the residual plays an important role. If the residuals are strongly linked this deteriorates considerably the biorthogonality relations and so affects directly the convergence of the method. In the block–GMRES algorithm, the block–Arnoldi process is completely decoupled from the minimization process. If the residuals are strongly linked, it does not influence the quality of the block–Arnoldi process. Equation (2.22) is potentially dangerous for the quality of the recovered

solutions and the suitable selection of the residual to be deflated is not clear. For stability reasons, we have therefore chosen in our implementation to only deflate the residuals upon the individual criterion

$$\frac{\|b - Ax_n^{(\ell)}\|_2}{\alpha^{(\ell)}\|x_n^{(\ell)}\|_2 + \beta^{(\ell)}} \leq \mathrm{tol}_\ell. \tag{2.23}$$

This criterion is not checked at each step but evaluated through

$$\frac{\|\tau_n^{(\ell)}\|_2}{\alpha^{(\ell)}\|x_n^{(\ell)}\|_2 + \beta^{(\ell)}} \leq \mathrm{tol}_\ell.$$

This is closely related to the standard stopping criterion defined in Section 2.1.2
A final remark is to note that, in exact arithmetic, the criterion (2.18) and the criterion (2.21) are equivalent; that is:

$$\sigma_{p_n}(\tau_n) = 0 \Leftrightarrow h_{n+p_n,n} = 0.$$

In Section 2.6.6.1, we show that , for $j = 1, \ldots, p$, $\sigma_j(\tau_n)$ is strongly related to $\sigma_{n-j+p+1}\left(\begin{pmatrix} \beta \\ 0_{n,p} \end{pmatrix} H_{n+p,n}\right)$. In exact arithmetic, the number of zero diagonal entries in a triangular matrix gives the number of zero singular values. However, it is well known that due to round–off and to the global ill–conditioning in the triangular matrices, the number of *small* singular values is greater than the number of *small* diagonal entries. In practice this holds for $\left(\begin{pmatrix} \beta \\ 0_{n,p} \end{pmatrix} H_{n+p,n}\right)$. Consequently, $\sigma_j(\tau_n)$ is poorly related to the j–th smallest diagonal entry of this matrix. The deflation in the residuals is decorrelated from the deflation in the Arnoldi procces. In Section 3.7.2.2, we experiment a deflation strategy on the Arnoldi vectors that is based on a criterion on the residual (namely criterion (2.23)).

2.6.6 Choice of the vectors in the Arnoldi sequence

In this paragraph, we describe how the Krylov space grows; that is, what is the strategy to select the next vector to be involved in the Arnoldi process. For the sake of clarity of exposure, we will ignore the problem related to the deflation and consider that deflation does not occur.
In paragraph 2.6.1, we have described the block–GMRES algorithm so that at step n, it generates v_{n+p} by normalizing w once it has been orthogonalized against V_{n+p-1}; w is computed by the matrix–vector product

$$w \longleftarrow Av_n.$$

From this scheme, we say that the Krylov vector v_{n+p} is the *son* of v_n. All the Krylov vectors are the sons of the p initial vectors that are the columns of V_p. In the example of the classic block–GMRES, any vector $v_{s_1p+s_2}$, with $s_1 \geq 0$ and $p > s_2 \geq 0$, is the son of the vector v_{s_2}. The vector v_{s_2} is called the root of $v_{s_1p+s_2}$. Each root vector ℓ has a succession of sons that eventually ends to what we call the

youngest son of ℓ. In that respect, we have p chains that link the Krylov vectors, each vector belongs to a unique chain and each chain has one root and one youngest son.
In the classic block–GMRES, the youngest sons of the root vectors are taken cyclically $1, 2, \ldots, p, 1, 2, \ldots$ to generate the successive Krylov vectors. However, we can choose any of the youngest sons as the next vector. If, at step n, the youngest son v_j is chosen to generate v_{n+p} then we set $\gamma(n) = j$. γ is an injective integer–valued function defined for $k = 1, \ldots, n$ that takes its values in $1, \ldots, n+p-1$. The p values in $1, \ldots, n+p-1$ that have no antecedent with γ correspond to the youngest sons. The function γ enables us to track the history of the run, thanks to this function we write

$$AV_{\gamma_n} = V_{n+p} H_{n+p,n},$$

where $V_{\gamma_n} = (V_{\gamma(1)}, \ldots, V_{\gamma(n)})$. Of course the relation $V_{n+p}^H V_{n+p} = I_{n+p}$ still holds. These variants of the block GMRES algorithm can, in some sense, be related with the flexible variant of GMRES, in this case $Z = V_{\gamma_n}$. In the block variant Z does not need to be stored explicitly, we only need γ to recover the solutions.
We consider three strategies to choose the next youngest son in the Arnoldi process. They give rise to three variants of block–GMRES, namely:

Classical block–GMRES: the youngest son is chosen cyclically as in the classic block–GMRES.

Depth-first block–GMRES: we always use the youngest son of a given root until the associated right–hand side converges, then we apply the same strategy to the next root. Using this terminology, the above classical block–GMRES could also be called breadth-first block–GMRES.

Largest-norm first block–GMRES: we select the youngest son of the root that is associated with the right–hand side that has the largest residual norm.

We can also introduce a fourth strategy that is somehow a continuum between the block and the seed GMRES, that is referred to as the continuum seed–block GMRES method. It somehow follows the deep first block–GMRES strategy. The complete block vector r_0 is not included in the Krylov space from the beginning but at run-time; one column at a time, once the linear system associated with the previous column has converged. The algorithm is as follows. Starting from the first initial residual $r_0^{(1)}$, the GMRES iterations built the Krylov space $V_{n_1+1}^{(1)}$, we stop at the n_1–th iteration when the stopping criterion threshold is achieved. The Arnoldi relation writes $\left(r_0^{(1)}, AV_{n_1}^{(1)}\right) = V_{n_1+1}^{(1)} R_{n_1+1}$ where $V_{n_1+1}^{(1)}$ has orthonormal columns and R_{n_1+1} is upper triangular. Then we insert the second initial residual $r_0^{(2)}$ to obtain $v_1^{(2)}$ and then for $j = 1, 2, \ldots$ computes the vectors $v_{j+1}^{(2)}$ via an Arnoldi–like process to obtain $\left(r_0^{(1)}, \mathbb{Z}V_{n_1}^{(1)}, r_0^{(2)}, AV_j^{(2)}\right) = \left(V_{n_1+1}^{(1)}, V_{j+1}^{(2)}\right) \begin{pmatrix} R_{n_1+1}^{(1)} & R_{j+1}^{(2)} \\ 0 & \end{pmatrix}$ where $\left(V_{n_1+1}^{(1)}, V_{j+1}^{(2)}\right)$ has orthonormal columns and $\begin{pmatrix} R_{n_1+1}^{(1)} & R_{j+1}^{(2)} \\ 0 & \end{pmatrix}$ is upper triangular. At each step j, we minimize $\|r_0^{(2)} - \mathbb{Z}J\|_2$ on $\left(V_{n_1}^{(1)}, V_j^{(2)}\right)$, the minimization is

performed via Givens rotations on the Hessenberg matrix given by $\begin{pmatrix} R^{(1)}_{n_1+1} & R^{(2)}_{j+1} \\ 0 & \end{pmatrix}$ where the first columns of $R^{(1)}_{n_1+1}$ and $R^{(2)}_{j+1}$ have been removed. Note that this Hessenberg matrix has a subdiagonal bandwidth of size 1 for the part associated with $R^{(1)}_{n_1+1}$ and a subdiagonal bandwidth of size 2 for the part associated with $R^{(2)}_{j+1}$. The process goes on until all the right–hand sides have converged. We recall that, in exact arithmetic, using this process, we compute the block–Krylov space for the block of vectors b. This algorithm may be viewed also as a variant of the seed-GMRES algorithm. We note that this algorithm is close to the algorithm given in [30, sec. 4.2]. We illustrate the numerical behaviour of this algorithm in Section 3.7.2.1. To conclude about the possible links between block variants and seed variants, we mention that it is possible to define the seed–block–GMRES method, the block methods is embedded in a seed process. For the QMR solver this method is studied in [85].
The strategy to select the next Arnoldi vector can symmetrically be used to define deflation strategy in the Arnoldi process. An example of a such a strategy is illustrated in Section 3.7.2.2. The strategy consists in stopping to use any youngest associated with a root right-hand sides that has converged.

2.6.6.1 An insight on the relation between the singular values of r_m and those of the Hessenberg matrix.

2.6.6.1.1 The classical GMRES context: for the solution of one right hand side the least squares problem in GMRES writes

$$\min_{y \in \mathbb{C}^n} \|e_1 \beta - H_{n+1,n} y\|_2.$$

If we denote by $U_{n+1} = \left(\begin{pmatrix} \beta \\ 0_n \end{pmatrix} H_{n+1,n} \right)$ it becomes

$$\min_{y \in \mathbb{C}^n} \|U_{n+1} \begin{pmatrix} 1 \\ y \end{pmatrix} \|_2. \tag{2.24}$$

It is interesting to relate this problem with

$$\begin{matrix} \min \\ \mu \in \mathbb{C}^n \\ \lambda \in \mathbb{C} \\ \|\mu\|_2^2 + |\lambda|^2 = 1 \end{matrix} \qquad \|U_{n+1} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \|_2. \tag{2.25}$$

Problem (2.25) amounts to find the singular vector associated with the smallest singular value of U_{n+1}. Problem (2.24) amounts to minimize U on the affine hyperplane x such that $e_1^T x = 1$, the solution is denoted $\begin{pmatrix} 1 \\ y \end{pmatrix}$. The value of the minimization is the norm of the residual of GMRES at step n, that is $\|r_n\|_2$. Problem (2.25) amounts to minimize U_{n+1} on the unit sphere, the solution is denoted

$\begin{pmatrix} \lambda \\ \mu \end{pmatrix}$, the value of the minimization is the smallest singular value of U_{n+1}, that is $\sigma_{n+1}(U_{n+1})$. In Figure 2.3, a geometrical interpretation of these two minimization problems is given. From the picture, if λ is not too small then the vectors $\begin{pmatrix} 1 \\ y \end{pmatrix}$ and $\begin{pmatrix} \lambda \\ \mu \end{pmatrix}$ are close, the two values of the norm of U_{n+1} applied to those vectors should also be close. Let assume $\lambda \neq 0$, on one hand we have

$$\sigma_{n+1}(U_{n+1}) = \|U_{n+1} \begin{pmatrix} \lambda \\ \mu \end{pmatrix} \|_2 = \lambda \|U_{n+1} \begin{pmatrix} 1 \\ \mu\lambda^{-1} \end{pmatrix} \|_2 \geq \lambda \|r_n\|_2.$$

On the other hand, we have

$$\|r_n\|_2 = \|U_{n+1} \begin{pmatrix} 1 \\ y \end{pmatrix} \|_2 \geq \sqrt{1 + \|y\|^2}\sigma_{n+1}(U_{n+1}).$$

Therefore we can write

$$\sigma_{n+1}(U_{n+1})\sqrt{1 + \|y\|^2} \leq \|r_n\|_2 \leq \sigma_{n+1}(U_{n+1})\lambda^{-1}.$$

This formula has to be related with the work of Strakŏs and Paige [100]. Indeed they have developed a more accurate study and manage to relate λ with the condition number of the matrix A. In this manuscript, we stick to this approach and assume $\lambda \neq 0$.

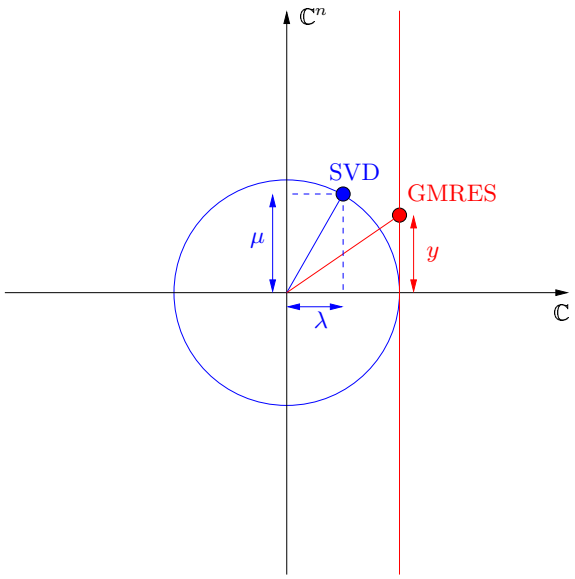


Figure 2.3: Link between the least squares problem and the smallest of singular value problem.

This result links the norm of the residual with the smallest singular value of the upper triangular matrix U. When the convergence occurs in GMRES ($\|r_n\|_2$ small), we expect U to be ill–conditioned. This result is useful to understand the GMRES algorithm with the modified Gram–Schmidt orthogonalization process. The QR–factorization of (b, AV_n) results in $V_{n+1}R_{n+1}$. Björck [13] shown that the loss of

orthogonality among the column of V_{n+1} is related, in the modified Gram–Schmidt algorithm, with the smallest singular value of R_{n+1}. Drkošová, Greenbaum, Rozložník and Strakoš [43] used these two results to show that the loss of orthogonality in the GMRES algorithm with the modified Gram–Schmidt orthogonalization only occurs at the convergence. Consequently, the effect of the loss of orthogonality is not problematic regarding the solution of the linear system.

2.6.6.1.2 The block-GMRES context: in that framework, we derive a similar result for the block–GMRES algorithm. At step n, we give a relation between the p singular values of r_n and the p smallest singular values of R_{n+p}
First of all,

$$r_n = V_{n+p} R_{n+p} \begin{pmatrix} 1 \\ -y_n \end{pmatrix}.$$

Consequently, for $k = 1, \ldots, p$ we have the following inequality on the singular value [77, p. 427]

$$\sigma_k(r_n) \geq \sigma_{n+k}(R_{n+p}) \sigma_p(\begin{pmatrix} 1 \\ -y_n \end{pmatrix}).$$

This gives

$$\sigma_k(r_n) \geq \sigma_{n+k}(R_{n+p}) \sqrt{1 + \sigma_p(y_n)^2}. \tag{2.26}$$

On the other hand, we recall that, using the givens rotations, Θ_n, we obtain

$$R_{n+p} = \Theta_n \begin{pmatrix} g_n & T_n \\ \tau_n & 0_{p,n} \end{pmatrix} \quad \text{and} \quad r_n = V_{n+p} \Theta_n \begin{pmatrix} 0_{n,p} \\ \tau_n \end{pmatrix}.$$

From the last expression, we recall that the p singular values of τ_n are the p singular values of r_n, for all $\ell = 1, \ldots, p$,

$$\sigma_\ell(r_n) = \sigma_\ell(\tau_n).$$

We denote the p singular vectors associated with the p smallest singular values of R_{n+p},

$$\begin{pmatrix} \lambda_n \\ \mu_n \end{pmatrix},$$

where λ_n is a p–by–p matrix, μ_n an n–by–p matrix and $\lambda_n^H \lambda_n + \mu_n^H \mu_n = I_p$. we can write

$$\Theta_n \begin{pmatrix} g_n & T_n \\ \tau_n & 0_{p,n} \end{pmatrix} \begin{pmatrix} \lambda_n \\ \mu_n \end{pmatrix} = u_n \Sigma_n,$$

where Σ_n is the diagonal matrix with the singular values $(\sigma_{n+1}(R_{n+p}), \ldots, \sigma_{n+p}(R_{n+p}))$ on the diagonal, and u_n corresponds to the left singular vectors. We obtain

$$\Theta_n \begin{pmatrix} g_n \lambda_n + T_n \mu_n \\ \tau_n \lambda_n \end{pmatrix} = u_n \Sigma_n.$$

Assuming λ_n nonsingular, we consider the matrix

$$\Theta_n \begin{pmatrix} g_n + T_n \mu_n \lambda_n^{-1} \\ \tau_n \end{pmatrix} = u_n \Sigma_n \lambda_n^{-1}.$$

Therefore the p singular values of τ_n satisfy, for all ℓ,

$$\sigma_k(\tau_n) \leq \sigma_k(u_n \Sigma_n \lambda_n^{-1}) = \sigma_k(\Sigma_n \lambda_n^{-1}) \leq \sigma_n(\lambda_n)^{-1} \sigma_{n+k}(R_{n+p}) \tag{2.27}$$

Finally equation (2.26) and equation (2.27) gives us

$$\sqrt{1 + \sigma_p(y_n)^2}\,\sigma_{n+k}(R_{n+p}) \leq \sigma_k(r_n) \leq \sigma_n(\lambda_n)^{-1} \sigma_{n+k}(R_{n+p}). \tag{2.28}$$

Assuming λ_n well–conditioned, we see that the convergence of the singular values of the block residual r_n are strongly linked with the convergence of the p smallest singular values of R_{n+p}. In Figure 2.4, we give a numerical illustration of this fact. The numerical experiment corresponds to the solution of a linear system with 5 right-hand sides, in each of the 5 sub-plots we display the 5 smallest singular values of r_n and R_{n+p} of same index when n, the iteration number, varies. It can effectively be seen that, in each sub-plot, the curves match and simmutaneously drop from a value close to 1.0 to a value close to machine precision; this happens when a convergence is observed. It can be seen that the first convergence occurs at the second iteration (bottom sub-plot), the second at iteration 5, and the last three at iteration 10.

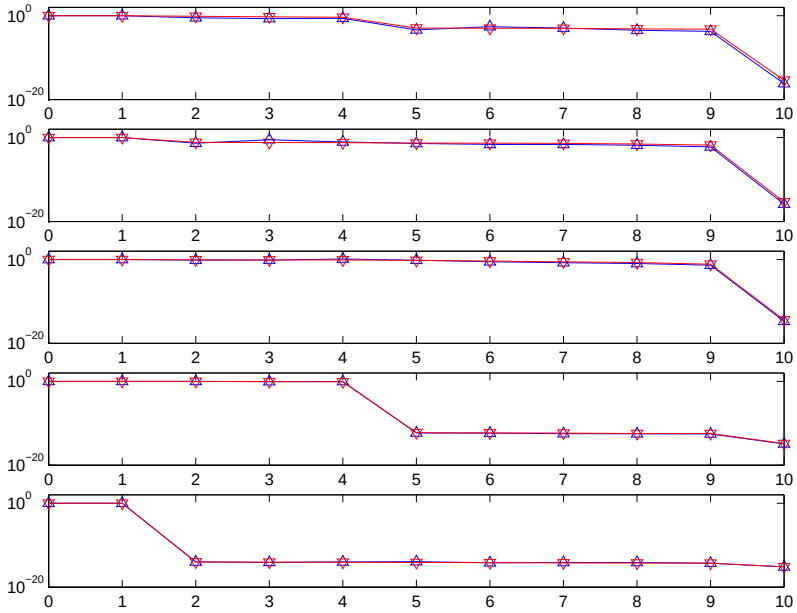


Figure 2.4: The convergence during the iterations of the block–GMRES method of the singular values of the block residual r_n ($\triangledown$) is strongly linked with the evolution of the p smallest singular values of R_{n+p} ($\triangle$).

When the first linear combination of the right–hand sides converges, this implies R_{n+1} is ill–conditioned. When the block–GMRES algorithm with modified Gram–Schmidt is run, we observe that an important loss of orthogonality appears in the set of vectors V_{n+p}. However, in the block case, $(p-1)$ linear combinations still have to converge and the consequences of the loss of orthogonality may be more

dangerous. In all our block implementation, we let the choice between the four orthogonalization scheme CGS, MGS, CGS2(K) and MGS2(K) but we highly recommend the reorthogonalization schemes.
A consequence of equation (2.28) is that at convergence we expect p small singular values in R_{n+p} ; this property is also exploited in Section 1.6.2.2.3.
Finally we note that the condition number of the matrix λ_n is related to the condition number of the matrix $\underline{H}_{n+p,n}$. In the single vector case, the condition number $\underline{H}_{n+1,n}$ is bounded by the one of A . In the block case, such a rule does not hold and even if A is well–conditioned it is possible for $\underline{H}_{n+p,n}$ to be ill–conditioned

III

Chapter 3

The Electromagnetism Application

3.1 Presentation of the electromagnetism problem

In the last decade, a significant amount of effort has been spent on the simulation of electromagnetic wave propagation phenomena to address various topics ranging from radar cross section, to electromagnetic compatibility, stealth, absorbing materials, and antenna design. In Figure 3.1, we illustrate one application of such a calculation that helps the car manufacturer to locate the best position for an antenna on a car. Two complementary approaches based on the solution of the Maxwell equations

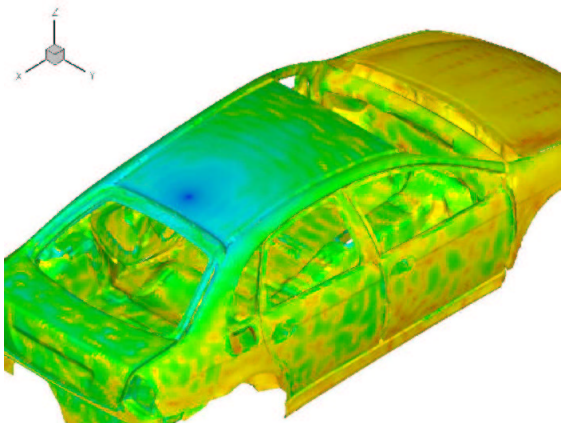


Figure 3.1: Representation of the electric current due to an antenna on the Citroën C5 car (courtesy of G. Sylvand, INRIA CERMICS).

are often adopted for tackling these problems. The first approach utilizes finite differencing in the time domain. The second one operates in the frequency domain. The latter approach offers two main advantages: (a) it does not require truncating the infinite spatial domain surrounding the scattering object which implies the use of approximate boundary conditions, and (b) it requires discretizing only the surface of the scattering object. On the other hand, the frequency domain approach leads to singular integral equations of the first kind, the discretization of which with boundary elements results in linear systems with complex and dense matrices which

are quite challenging to solve.
The *Boundary Element Method* (BEM) has been successfully employed in the numerical solution of this class of problems, proving to be an effective alternative to common discretization schemes like Finite Element Methods (FEM's), Finite Difference Methods (FDM's) or Finite Volume Methods (FVM's). The idea of BEM is to shift the focus from solving a partial differential equation defined on a closed or unbounded domain to solving a boundary integral equation over the finite part of the boundary. The discretization by BEM results in linear systems with dense complex matrices. The coefficient matrix can be symmetric non-Hermitian in the Electric Field Integral Equation formulation (EFIE), or unsymmetric in the Combined Field Integral Equation formulation (CFIE) (see [103] for further details). The unknowns are associated with the edges of an underlying mesh on the surface of the object. With the advent of parallel processing, this approach has become viable for large problems and the typical problem size in the electromagnetic industry is continually increasing. Nevertheless, nowadays, many problems can no longer be solved by parallel out-of-core direct solvers as they require too much memory, CPU and disk resources and iterative solvers appear as a viable alternative. In our work, we will mainly consider the EFIE formulation that usually gives rise to linear systems that are more difficult to solve with iterative methods. Another motivation to focus only on EFIE formulation is that it does not require any restriction on the geometry of the scattering obstacle as CFIE does, and, in this respect, is more general.

3.1.1 Background on the electric field–integral equation formulation

Let Γ be a metallic scatter. We suppose that the surface of Γ is discretized with triangles. The mesh has m_T triangles and m edges. We denote by r_Γ the radius of the smallest ball that completely encompasses the object. We set the origin of the coordinates at the centre of this ball. If λ is a wavelength, we denote the wave number by

$$k = \frac{2\pi}{\lambda}.$$

The frequency is $F = c/\lambda$ where c is the light velocity in the vacuum.
From Γ and its mesh, we construct a space of functions V_h. The dimension of this space is finite and equal to the number of edges,

$$V_h = \text{Span } \left\{ \vec{\Psi}_\ell(x); 1 \leq \ell \leq m \right\}.$$

The $\vec{\Psi}_\ell(x)$'s are the basis functions; we choose the standard basis functions for this type of problem: those of Raviart–Thomas [107]. Note that a few years later, they were rediscovered by Rao–Wilson–Glitton [106]. Each basis function $\vec{\Psi}_\ell(x)$ is associated with an edge e_ℓ and its value is zero everywhere except on the two triangles $T_{e_\ell^+}$ and $T_{e_\ell^-}$ that share the edge e_ℓ. On these two triangles, $\vec{\Psi}_\ell(x)$ is defined by

$$\vec{\Psi}_\ell(x) = \varepsilon \frac{\overrightarrow{x - A^{T^\varepsilon_{e_\ell}}}}{2 \cdot \text{area}(T^\varepsilon_{e_\ell})}, \quad x \in T^\varepsilon_{e_\ell},$$

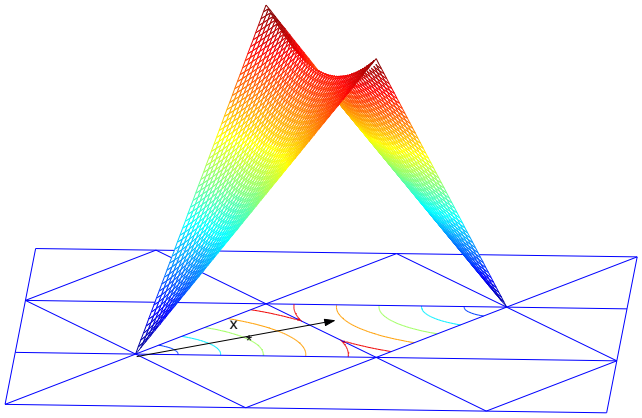


Figure 3.2: Representation of the vector basis function associated with the ℓ-th edges. The function is nonzero only on the two adjacent triangles $T_{e_\ell}^+$ and $T_{e_\ell}^-$ where the norm of its value is depicted.

where $\varepsilon = \pm$. Figure 3.2 is an illustration of one this basis function $\vec{\Psi}_\ell(x)$.
In the notation V_h, h represents a typical length of an edge, for example the largest one, this highlights the fact that V_h is an approximation subspace.
The EFIE can be written as follows. If $\vec{E}^{\text{inc}}(x)$ is some incident field, the problem is

Find $\vec{J}_h(x) \in V_h$ such that for each test function $\vec{J}_h^{\text{test}}(x) \in V_h$,

$$-\int\int_{\Gamma\times\Gamma} ikZ_0 \frac{e^{ik|y-x|}}{4\pi|y-x|}\left(\vec{J}_h(x)\cdot\vec{J}_h^{\text{test}}(y) - \frac{1}{k^2}\text{div}_\Gamma\vec{J}_h(x)\text{div}_\Gamma\vec{J}_h^{\text{test}}(y)\right)ds(x)ds(y)$$
$$= \int_\Gamma \vec{E}^{\text{inc}}(x)\cdot\vec{J}_h^{\text{test}}(x)ds(x). \tag{3.1}$$

Here, $\text{div}_\Gamma\vec{J}_h(y)$ denotes the surface divergence (it is a scalar number). $\cdot$ is the usual dot product in $\mathbb{R}^3$ and Z_0 is the impedance of the vacuum.
By writing $\vec{J}_h(x)$ in terms of the basis functions, we get

$$\vec{J}_h(x) = \sum_{\ell=1}^{m} J_\ell\vec{\Psi}_\ell(x).$$

The J_ℓ's are the components of $\vec{J}_h(x)$ in $\vec{\Psi}_\ell(x)$. They can be interpreted as the flux of the current $\vec{J}_h(x)$ across the edges.
TThe variational equation (3.1) holds for any basis function $\vec{\Psi}_\ell(x)$. The choice of taking the same basis for decomposing $\vec{J}_h(x)$ and for the test function is not

mandatory but simplifies the problem. This gives a linear system of order m where the unknowns are J_ℓ, $1 \leq \ell \leq m$. The m linear equations are

$$\sum_{\ell=1}^{m} \mathbb{Z}_{j,\ell} J_\ell = F_j, \quad 1 \leq j \leq m.$$

In matrix form, this gives

$$\mathbb{Z}J = F, \tag{3.2}$$

where the entry (j,ℓ) of $\mathbb{Z}$ is

$$\mathbb{Z}_{j,\ell} = -\int\int_{\Gamma\times\Gamma} ikZ_0 \frac{e^{ik|y-x|}}{4\pi|y-x|}\left(\vec{\Psi}_j(x)\cdot\vec{\Psi}_\ell(y) - \frac{1}{k^2}\mathrm{div}_\Gamma\vec{\Psi}_j(x)\mathrm{div}_\Gamma\vec{\Psi}_\ell(y)\right), \tag{3.3}$$

and the entry j of F is

$$F_j = \int_\Gamma \vec{E}^{\mathrm{inc}}(x)\cdot\vec{\Psi}_j(x)ds(x). \tag{3.4}$$

3.1.2 Plane wave scattering and monostatic calculation

In radar applications, the incident field is taken as a plane wave. The general expression of such a wave is

$$\vec{E}^{\mathrm{inc}}(x,\varphi,p_\theta,p_\varphi) = (p_\theta)\hat{u}_\theta e^{ikx\cdot\hat{u}_r(\varphi)} + (p_\varphi)\hat{u}_\varphi e^{ikx\cdot\hat{u}_r(\varphi)}, \tag{3.5}$$

where (p_θ, p_φ) are two complex numbers and $\hat{u}_r$, $\hat{u}_\theta$, $\hat{u}_\varphi$ are the classical unitary vectors:

$$\hat{u}_r = \begin{pmatrix} \cos\varphi\cos\theta \\ \sin\varphi\cos\theta \\ \sin\theta \end{pmatrix} \quad , \quad \hat{u}_\theta = \begin{pmatrix} -\cos\varphi\sin\theta \\ -\sin\varphi\sin\theta \\ \cos\theta \end{pmatrix} \quad , \quad \hat{u}_\varphi = \begin{pmatrix} -\sin\varphi\cos\theta \\ \cos\varphi\cos\theta \\ \sin\theta \end{pmatrix}.$$

In Figure 3.3, given the Cartesian coordinates $(O,\hat{x},\hat{y},\hat{z})$, we describe the spherical coordinates. From equation (3.5), the families of plane waves associated with the couple $(p_\theta,p_\varphi)=(1,0)$ and the families of plane waves associated with the couple $(p_\theta,p_\varphi)=(0,1)$ are independent. We call $(p_\theta,p_\varphi)=(1,0)$ the φ polarization and $(p_\theta,p_\varphi)=(0,1)$ the θ polarization.
In many applications, the wave is coming from different directions located all around a circle. At the price of a rotation of the coordinates, we can assume that $\theta = 0$ and φ varies from 0 to 2π. We get

$$\hat{u}_r = \begin{pmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{pmatrix} \quad , \quad \hat{u}_\theta = \hat{z} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad , \quad \hat{u}_\varphi = \begin{pmatrix} -\sin\varphi \\ \cos\varphi \\ 0 \end{pmatrix}.$$

Figure 3.4 illustrates this choice.
For the incident field we get

$$\begin{aligned} \vec{E}^{\mathrm{inc}}(x) = \vec{E}^{\mathrm{inc}}(x,\varphi) &= \hat{z}e^{ikx\cdot\hat{u}_r(\varphi)} \\ &= \hat{z}e^{ik(x_1\cos\varphi + x_2\sin\varphi)}. \end{aligned} \tag{3.6}$$

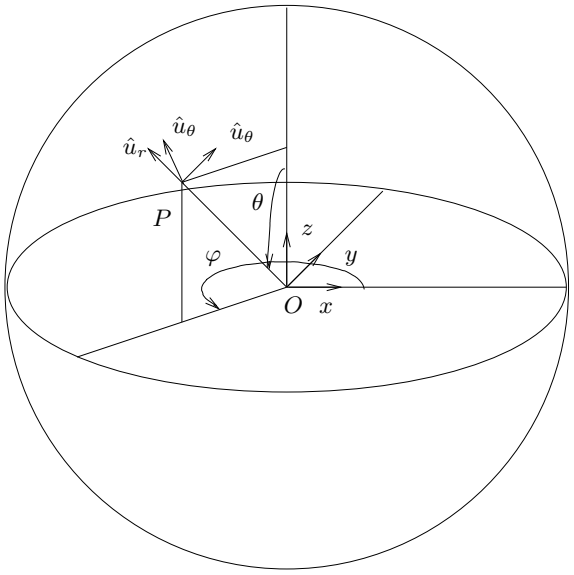


Figure 3.3: spherical coordinates.

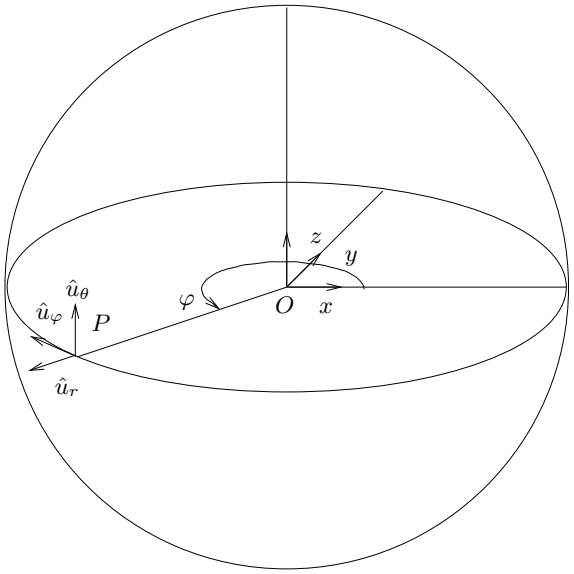


Figure 3.4: spherical coordinates in the plane $\hat{z} = 0$.

Definition 3.1.1 *The monostatic calculation with θ polarization is equivalent to finding the current $\vec{J}(x, \varphi)$ associated with the incident waves $\vec{E}^{inc}(x, \varphi)$ when φ varies from 0 to 2π .*

In practice we sample φ: we consider p equidistant angles $\varphi_1, \varphi_2, \ldots, \varphi_p$ and we have to solve p linear systems with the same coefficient matrix:

$$\mathbb{Z}J(\varphi_a) = F(\varphi_a), \quad 1 \leq a \leq p.$$

Once we have solved the a–th system to obtain $J(\varphi_a)$, we construct

$$\vec{J}_h(x,\varphi_a) = \sum_{\ell=1}^{n} J_\ell(\varphi_a)\vec{\Psi}_\ell(x)$$

then compute

$$a_\infty(\varphi_a) = \frac{ikZ_0}{4\pi}\int_\Gamma (\hat{u}_r(\varphi_a) \wedge (\vec{J}_h(x,\varphi_a) \wedge \hat{u}_r(\varphi_a)))ds(x),$$

$$a_\infty(\varphi_a) = \frac{ikZ_0}{4\pi}\int_\Gamma \hat{u}_r(\varphi_a) \wedge \vec{J}_h(x,\varphi_a)e^{ikx\cdot\hat{u}_r(\varphi_a)}ds(x),$$

and finally

$$\mathrm{RCS}(\varphi_a) = 20\log_{10}(|a_\infty(\varphi_a)|)$$

that is the back-scattered Radar Cross Section (RCS), the quantity of interest for instance for the aircraft manufacturers.
In Figure 3.5, we plot the computed monostatic radar cross section for the Airbus 23676 (see 3.2.4). In this graph, the black point on the curve has the following meaning: a plane wave is illuminating the Airbus coming from the direction $\varphi = 30^o$, we recover the energy of the associated back-scattered field in the same direction; depicted in decibel scale in the curve, the value of the energy $\mathrm{RCS}(\varphi = 30^o) = -4$.

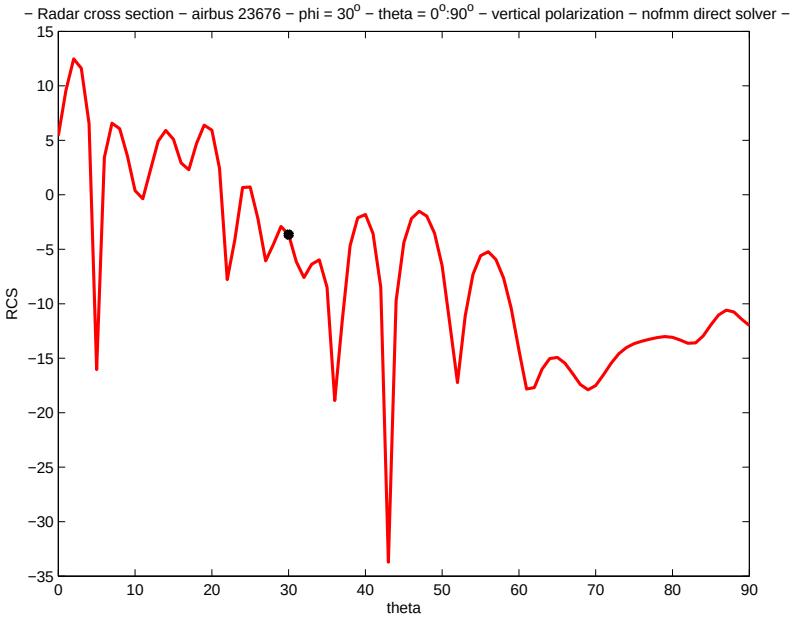


Figure 3.5: monostatic Radar Cross Section (RCS) for the Airbus 23676.

The current $J(\varphi_a)$ obtained from the solution of the system corresponding to $F(\varphi_a)$ may also be used to compute the bistatic RCS associated with the angle φ_a. For

the angle φ_b, it consists in computing

$$a_\infty(\varphi_a,\varphi_b)=\frac{ikZ_0}{4\pi}\int_\Gamma(\hat{u}_r(\varphi_b)\wedge(\vec{J}_h(x,\varphi_a)\wedge\hat{u}_r(\varphi_b)))ds(x).$$

The interpretation of $a_\infty(\varphi_a,\varphi_b)$ is the following: a plane wave illuminates the object coming from the direction φ_a, and we recover the energy of the associated back-scattered field in the direction φ_b.
The computation we are faced with is in general either the bistatic RCS associated with a fixed angle or the monostatic computation. The bistatic RCS requires the solution of a system with a single right–hand side whereas the monostatic RCS requires the solution of a system with multiple right–hand sides.

3.1.3 Properties of the EFIE matrix

From equation (3.3) that defines $\mathbb{Z}_{j,\ell}$, we have $\mathbb{Z}_{j,\ell}=\mathbb{Z}_{\ell,j}$; that is, with the EFIE formulation, $\mathbb{Z}$ is complex symmetric.
If we write $e^{ik|y-x|}=\cos(k|y-x|)+i\sin(k|y-x|)$ in equation (3.3), we obtain $\mathbb{Z}=\mathbb{B}+i\mathbb{D}$ where $\mathbb{B}$ and $\mathbb{D}$ are symmetric real such that

$$\begin{aligned}\mathbb{B}_{j,\ell} &= \int\int_{\Gamma\times\Gamma} kZ_0\frac{\sin(k|y-x|)}{4\pi|y-x|}\left(\vec{\Psi}_j(x)\cdot\vec{\Psi}_\ell(y)-\frac{1}{k^2}\mathrm{div}_\Gamma\vec{\Psi}_\ell(x)\mathrm{div}_\Gamma\vec{\Psi}_\ell(y)\right),\\ \mathbb{D}_{j,\ell} &= -\int\int_{\Gamma\times\Gamma} kZ_0\frac{\cos(k|y-x|)}{4\pi|y-x|}\left(\vec{\Psi}_j(x)\cdot\vec{\Psi}_\ell(y)-\frac{1}{k^2}\mathrm{div}_\Gamma\vec{\Psi}_j(x)\mathrm{div}_\Gamma\vec{\Psi}_\ell(y)\right).\end{aligned}$$

Since $\mathbb{B}$ (resp. $i\mathbb{D}$) is symmetric real (resp. symmetric imaginary), it has real (resp. imaginary) eigenvalues.
$\mathbb{B}$ comes from an integral operator with the following kernel

$$K_{\mathbb{B}}(x,y)=\frac{\sin(k|y-x|)}{4\pi|y-x|}.$$

This kernel is extremely smooth, consequently the matrix $\mathbb{B}$ is of low rank. Moreover Collino and Després [32] show that the matrix $\mathbb{B}$ may also be written

$$\mathbb{A}_\infty\mathbb{A}_\infty^H$$

where $\mathbb{A}_\infty$ corresponds to the operator that, given a current on the object, returns the diffracted field. We have the property that the eigenvalues of $\mathbb{A}_\infty\mathbb{A}_\infty^H$ are real positive with a large number of them equal to zero.
In the 2D case, equation (3.3) is multiplied by $-i$. So that in a sense we obtain

$$\mathbb{Z}=\mathbb{D}-i\mathbb{B}$$

and the properties of $\mathbb{B}$ and $\mathbb{D}$ from the 3D case hold in 2D.

3.2 Simulation codes and model problems

3.2.1 Presentation of the 2D code IE2M

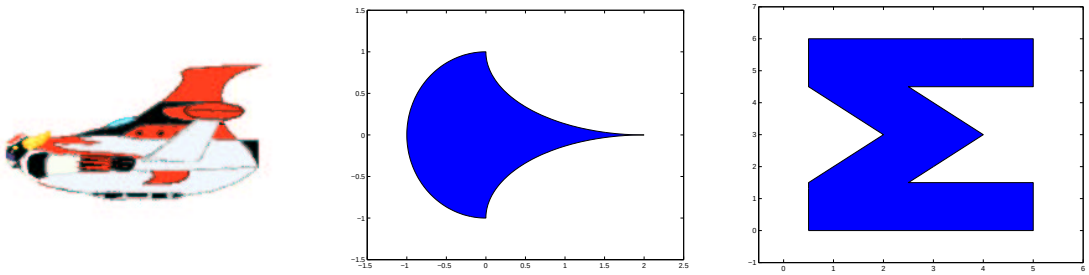
The 2D code IE2M has been developed by Abderrahmane Bendali from CERFACS. It is a MATLAB6 code based on the EFIE, where all the modules related to electromagnetics are implemented in single precision arithmetic Fortran code. The main interest of this code is that the difficulties encountered by the linear solvers in 2D are similar to those observed in 3D. Using MATLAB, we can easily use and quickly investigate the behaviour of various solvers and preconditioners.

3.2.2 Case study in 2D

In Table 3.1, we give the characteristics of the three 2D test cases we consider. The quantity p corresponds to the diameter in wavelengths of the smallest circle that encompasses the object. In Figure 3.6, the geometries of the three objects are depicted.

matrix	ddl	λ	p
Goldorak	715	0.5	0.32
CNSPH	310	0.5	3.0
CERFACS	312	0.5	0.75

Table 3.1: Characteristics of the 2D test cases.



(a) Goldorak (b) CNSPH (c) CERFACS

Figure 3.6: Different 2D test examples.

3.2.3 Presentation of the 3D code AS_ELFIP

The code AS_ELFIP is a joint effort of three partners. The first part is the work from EADS–CCR. They have developed the basic kernel operations for the electromagnetism equations and also the basic operations for the out–of–core linear algebra calculations. Dealing with the singularities arising in the integral equations is a key point that requires a knowledge that is based on a long study combined with a wide variety of test cases to assess the quality of the methods. Such work can only be done by a long established company namely EADS–CCR. Recently, the fast multipole method [71] has been extended to the integral equations of electromagnetism

[110, 126]. The second contribution to the work has been developed by Guillaume Sylvand [130] during his PhD thesis at CERMICS. He provided an implementation of the fast multipole method for sequential and parallel distributed platforms. His code is among the most advanced in terms of capabilities and efficiency. Finally the third component is the result of the work of the Parallel Algorithm Project at CERFACS. Bruno Carpentieri, Luc Giraud and Iain Duff provided the way to implement an efficient Frobenius norm minimization preconditioner; a detailed and rigorous study [23] shows the efficiency of this preconditioner in the electromagnetism context. In particular, to compensate for the lack of scalability of this preconditioner, they also adapted a flexible variant of GMRES. These three components result in the AS_ELFIP code. With this code, the solution of the EFIE for one right–hand side was properly addressed. However, for the monostatic RCS calculations, many linear systems with the same coefficient matrix and different right–hand sides have to be solved. The number of right-hand sides varies depending on the geometry and the frequency of the illuminating waves, but it usually classically ranges from a few tens to a few hundreds. The iterative methods mentioned earlier are no longer appropriate. This motivates the third part of this thesis.
From a technical point of view, the code is parallel and suited for shared or distributed memory computers. Furthermore, it implements out–of–core capabilities to handle very large problems. During my PhD, I have been using this code on several computers ranging from linux laptops or Sun workstations to parallel architectures with shared or distributed memory like the Compaq Alpha Server at CERFACS and CEA, the PC cluster at CERFACS, the Origin 2000 at CINES and the IBM SP3 (and SP4) at CINES.
For the sake of consistency in this manuscript, when running time are displayed they have been observed on the Compaq Alpha Server at CERFACS.

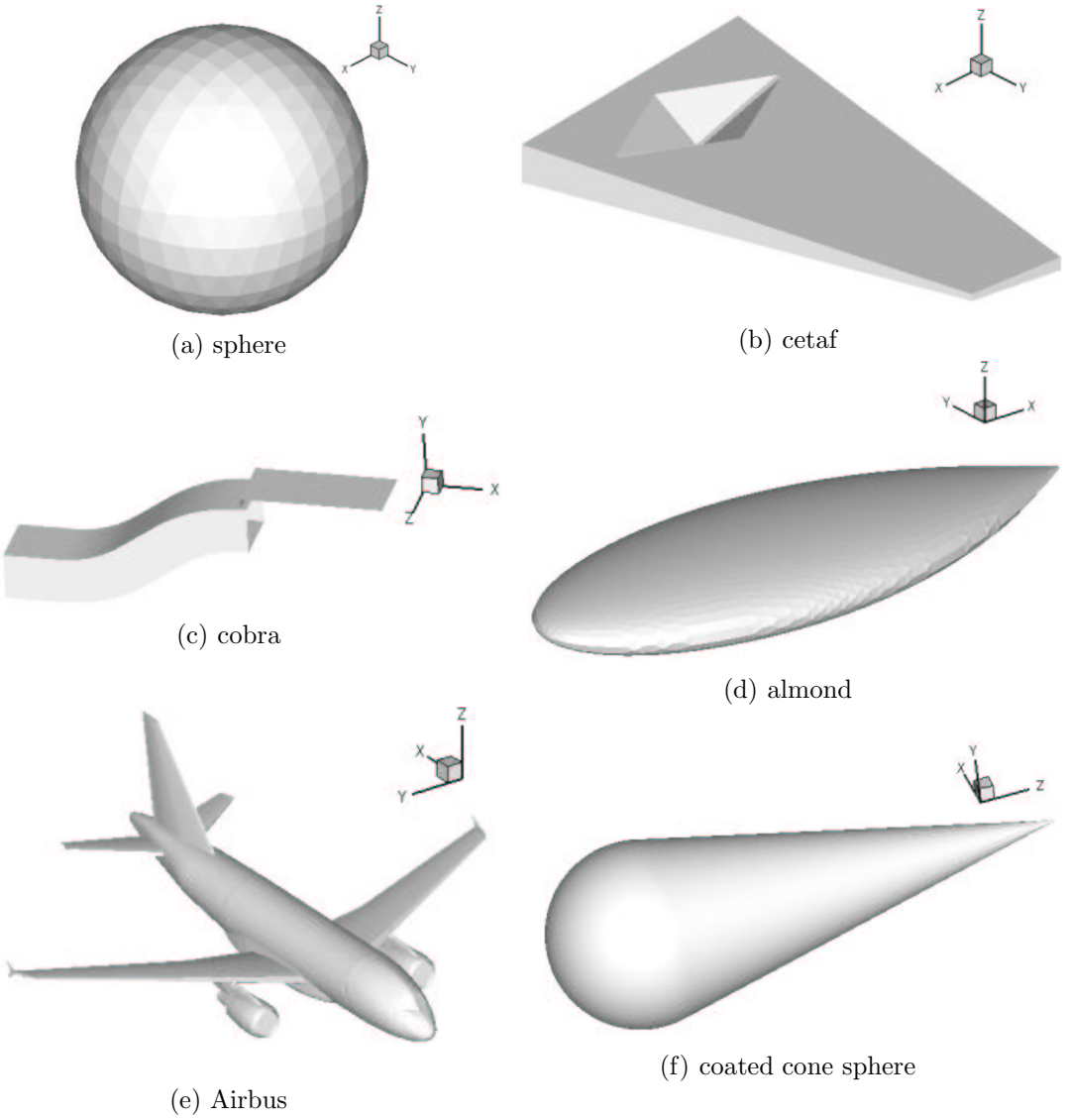
3.2.4 Case study in 3D

For the 3D calculations, we consider the six geometries that are depicted in Figure 3.7.
The sphere is the first test case, its mesh is easily constructed. It is homogeneous in the sense that all the edges have all the same size (i.e. with a $\pm 20\%$ variation). For all the spheres we consider a radius equal to one metre, the wavelength and the mesh size vary such that the average length of the edges is around $\lambda/10$. The quantity q is such that the average length of the edges is λ/q. In the case of the sphere, we thus have $q = 10$. An advantage of using this object is that the back-scattered far field is known analytically from the Mie series (see e.g. [130]). In order to study the scalability of our solvers, we have a collection of spheres of different mesh size. The ones used most frequently are given in Table 3.2. In that table, "dof" is the number of degrees of freedom (i.e. the number of edges) in the meshes, F the frequency of the incident fields, λ the wavelength and p the size of the sphere measured in number of wavelengths. For example, the wavelength of the incident field illuminating the sphere 40368 edges is $\lambda = 33.3$ cm since the radius of the sphere is one metre, the size of the sphere is $p = 6$ wavelengths. In fact, the value of p is defined for nonspherical objects as well. It corresponds to the diameter

(in wavelengths of the incident field) of the smallest sphere that encompasses the object. For most of the objects, we give a rough approximation of this value. One of the drawbacks is that the problem of the monostatic computation from an angle φ_l is the same for any $\varphi_l = 0, \ldots, 2\pi$. Therefore, there is practically no interest to perform runs with more than one right–hand side.
The second tested geometry is called a cetaf. This is a standard test case in computational electromagnetism. The object is depicted in Figure 3.7.b. It is a sort of wing with a slit. Its physical size is 50 cm $\times$ 30 cm $\times$ 5 cm and the associated value for q is 7.9. A typical monostatic computation is done for $\theta = 0^o : 1^o : 180^o$ and $\varphi = 0^o$ with a particular interest for the interval $\theta = 60^o : 1^o : 70^o$.
The third test case is called a cobra. It is also a standard test case. Its size is 67.9 cm $\times$ 23.3 cm $\times$ 11 cm. It represents a cavity and is considered from the electromagnetism point of view as a challenging problem. In this manuscript, we observe that it gives rise to a difficult linear algebra problem for the iterative solvers. For the radar cross section calculation, the interval of interest is $(\theta = 20^o : 1^o : 30^o, \varphi = 0^o)$. Among all the objects considered in this document, we notice that this is the only one that is an open surface and consequently can only be solved using the EFIE formulation.
The fourth test case is an almond. It was an official test–case for the JINA 2002 (12th International workshop on Antenna design, Nice 2002). Its size is 2.5 m. For this workshop, several codes were benchmarked on this problem to calculate the monostatic RCS on the interval $\theta = 90^o, \varphi = 0^o : 1^o : 180^o$.
The fifth object is a commercial airplane. It is an Airbus A318 of size (not realistic) about 1.8 m $\times$ 1.9 m $\times$ 0.65 m. In Table 3.2, we give the characteristics of the range of Airbus is used in the manuscript for the numerical experiments.
All the previous objects were perfectly conducting objects. Finally, we also consider a simulation with a dieletric. It consists in a coated cone sphere of size 24 cm in length and 9 cm in diameter. The number of degrees of freedom is 77604. The frequency of the incident field is 3 GHz.
We note that, in Table 3.2, the number of edges increases proportionally with the square of the frequency. This observation holds for all the geometries.

3.2.5 A remark on the mesh size versus the wavelength

In order to have a reliable representation of a sine function of wavelength λ on the mesh, the usual criterion is to take the edges so that the average of the length of the edges, e, is smaller than $\lambda/6$ or $\lambda/10$ sometimes (see e.g [130, p. 45]). In Figure 3.8, we show that if the mesh is too coarse then the sine function is not well approximated. Consequently, if the frequency of the illuminating field increases (i.e. the wavelength decreases proportionally), the size of the surface mesh increases in order to maintain the average length of the edges of the order of $\lambda/10$. Roughly speaking, if the frequency is multiplied by α, the size of the mesh is multiplied by α^2 (cf. Table 3.2).
When we study the aircrafts, we therefore study different physical problems. To illustrate this point, we consider a sphere of radius one metre. The first case is denoted by $(p, q) = (13, 10)$. We consider the wavelength so that $13\lambda = 1$ metre,



(a) sphere

(b) cetaf

(c) cobra

(d) almond

(e) Airbus

(f) coated cone sphere

Figure 3.7: Different 3D test examples.

and we mesh the sphere as that the average length of the edges is $\lambda/10$. The second case is $(p,q) = (10,13)$ ($10\lambda = 1$ metre, and the average length of the edges is $\lambda/13$), and the third $(p,q) = (10,10)$ ($10\lambda = 1$ metre, and the average length of the edges is $\lambda/10$). Doing so we obtain three cases such that the first two have exactly the same mesh but correspond to different problems, the last two represent the same physical problem but exploit two different meshes. The second mesh is very fine for the problem it is used for. In Table 3.3, we give the number of iterations for GMRES with the Frobenius preconditioner [1] to converge on these test cases. The stopping criterion is such that $\|r_n\|_2/\|b\|_2 \leq 2 \cdot 10^{-2}$. On the two cases representing the same physical problems (a sphere of one metre of radius with an illuminating field at 3.00 GHz), the number of GMRES iterations are the same even if the meshes are different. On the other hand, if the same meshes are taken (ddl

[1] see Section 3.3.2.1, for the description of this preconditioner

	dof	F	λ	p
cetaf	5391	3.0 GHz	10.0 cm	6
sphere	40368	0.9 GHz	33.3 cm	6
sphere	71148	1.2 GHz	25.0 cm	8
sphere	161472	1.8 GHz	16.7 cm	12
sphere	288300	2.4 GHz	12.5 cm	16
sphere	549552	3.3 GHz	9.1 cm	22
sphere	1023168	4.5 GHz	6.6 cm	30
Airbus	23676	2.3 GHz	13.0 cm	14
Airbus	94704	4.6 GHz	6.5 cm	29
Airbus	213084	6.9 GHz	4.3 cm	44
Airbus	591900	9.1 GHz	3.2 cm	59
Airbus	1160124	11.4 GHz	2.6 cm	73
cobra	3823	1.0 GHz	30.0 cm	2
cobra	14449	5.0 GHz	6.0 cm	12
cobra	60695	10.0 GHz	3.0 cm	24
almond	360	0.1 GHz	299.8 cm	1
almond	8112	0.7 GHz	42.8 cm	6
almond	104793	2.6 GHz	11.5 cm	22
coated cone	77604	3.0 GHz	10.0 cm	1.5

Table 3.2: Characteristic of the 3D test cases. All the object are perfectly conducting except the coated cone sphere (dielectric). All the object are closed except the cobra cavity.

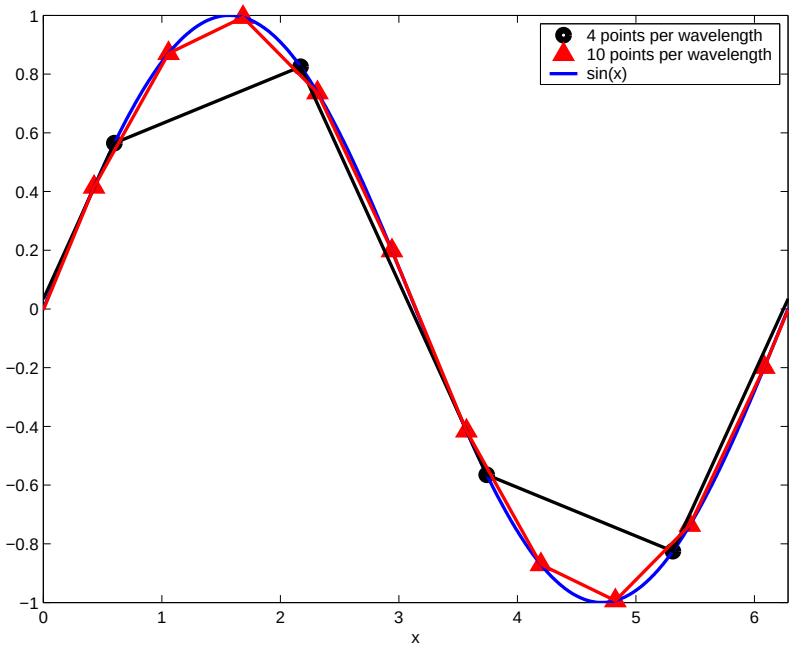


Figure 3.8: A sine function discretized with 4 points per wavelength and 10 points per wavelength.

$= 768108$) for different physical problems then the number of GMRES iterations differ. This experiment illustrates the fact that, the meshes of the same object differ not only in their number of edges but also in the physical problems they represent.

Finally, these experiments indicate that $q = 10$ is enough; increasing this value would only result in increasing the size of the linear systems to be solved but would not improve the physical meaning of the computed solution.

p	q	Frequency	dof	# iter
13	10	3.90 GHz	768108	142
10	13	3.00 GHz	768108	80
10	10	3.00 GHz	451632	80

Table 3.3: GMRES(30) with Frobenius preconditioner on three spheres of one metre radius.

3.2.6 On the properties of the linear systems

In Section 3.1, we explained the theoretical properties shared by the matrices and the right–hand sides involved in an electromagnetism calculation. In this section, some general observations are given; they correspond to the global trend observed in our test–cases.

The matrix arising from the boundary element method is dense, this is due to the fact that the value of the Green's function between one point x and another $y(\neq x)$ is never zero. However the decay of the magnitude of the function is quick and starting from a threshold level on the distance between x and y, the influence of x on y (and reciprocally) is small. Consequently, we expect the entry (j, ℓ) of $\mathbb{Z}$, depicting the influence between the edge e_j of the mesh and the edge e_ℓ, to be small. In Figure 3.9, we sparsify the matrix obtained from the Goldorak test case by removing the entries that are lower in modulus than a prescribed threshold (1/1000 of the largest magnitude).

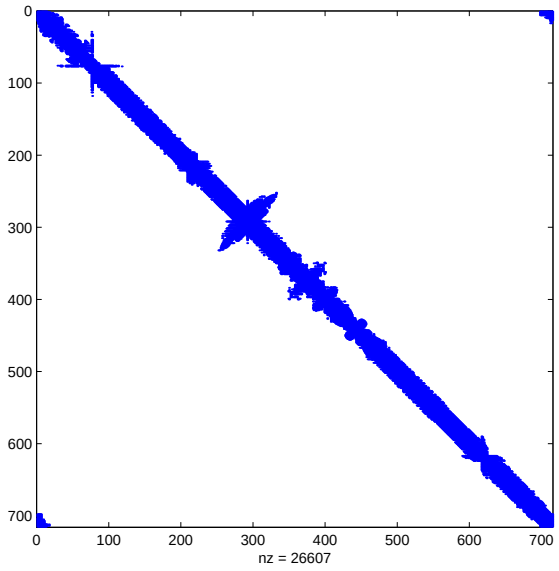
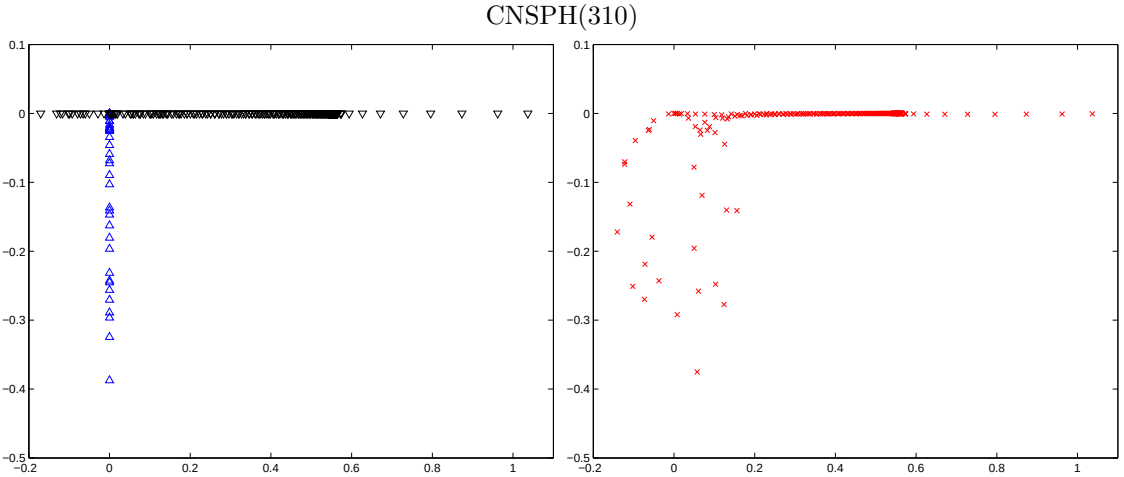


Figure 3.9: Entries of $\mathbb{Z}$ with modulus larger than 1/1000 of the largest entry in modulus for the test example Goldorak.

In Figure 3.10.a, we plot the eigenvalues of $-i\mathbb{B}$ and $\mathbb{D}$ computed with the IE2M toolbox. We may notice that, among the 310 eigenvalues of $\mathbb{B}$, 256 are smaller (in modulus) than $10^{-5}\|\mathbb{B}\|_2$. This is due to the low numerical rank of $\mathbb{B}$ (we recall that the IE2M toolbox is implemented in single precision arithmetic where the machine precision is about 10^{-6}). Moreover, all the eigenvalues of $\mathbb{B}$ are positive. In Figure 3.10.b, we plot the spectrum of $\mathbb{Z} = \mathbb{D} - i\mathbb{B}$.



(a) spectrum of $\mathbb{D}$ and $-i\mathbb{B}$,
∇ eigenvalues of $\mathbb{D}$,
$\triangle$ eigenvalues of $-i\mathbb{B}$.

(b) spectrum of $\mathbb{Z} = \mathbb{D} - i\mathbb{B}$.
$\times$ eigenvalues of $\mathbb{Z}$,

Figure 3.10: Spectrum of the various matrices.

Since the right–hand side F is in the range of $\mathbb{A}_\infty$, it belongs to the range of $\mathbb{B} = \mathbb{A}_\infty \mathbb{A}_\infty^H$. Then, if we decompose the right–hand sides using a basis of eigenvectors of $\mathbb{Z}$ (note that in practice we have observed that $\mathbb{Z}$ is diagonalisable and the condition number associated with the set of eigenvectors is reasonable), we expect that the largest components of this decomposition correspond to eigenvectors with a large real part. In Figure 3.11, we plot the spectrum of $\mathbb{Z}$ and we plot a circle on the eigenvalues for which the modulus of the components of the right–hand side corresponding to $\theta = 90^o$ is larger than 20% of the maximum modulus of the components in the eigenvectors basis. We observe that the real part of the spectrum is not excited.

Finally, in Figure 3.12, we plot the 100 largest singular values of $\mathbb{B}$ and of $[F(0^o), F(1^o), \ldots, F(359^o)]$. We observe that the rank of $\mathbb{B}$ is approximately the same as those of $[F(0^o), \ldots, F(359^o)]$. This is agreement with the theoretical result, that is, the right-hand sides belong to the range of $\mathbb{B}$.

The 3D–matrix $\mathbb{Z} = \mathbb{B} + i\mathbb{D}$ is obtained from the 2D–matrix $\mathbb{Z} = \mathbb{D} - i\mathbb{B}$ by a multiplication by $-i$. In a similar way, we observe that the 3D–spectrum is similar to a 2D–spectrum with a rotation of 90^o degrees clockwise. In Figure 3.13, we plot some spectrum for different 3D–test cases.

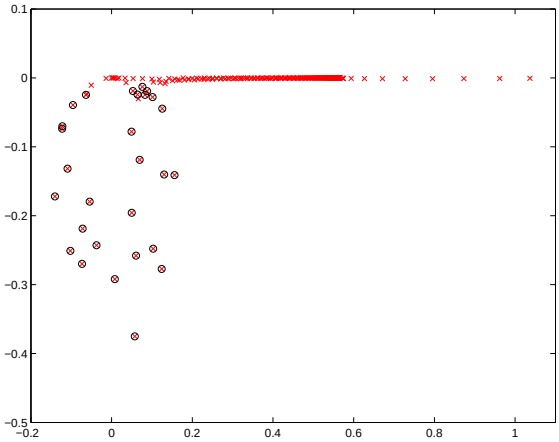


Figure 3.11: The test case is the CNSPH. The spectrum of $\mathbb{Z}$ is represented with $\times$. We decompose the right–hand side using the eigenvector basis and plot an $\circ$ on the eigenvalues when the components of the right–hand sides in the associated eigenvector is larger than 20% of the largest component.

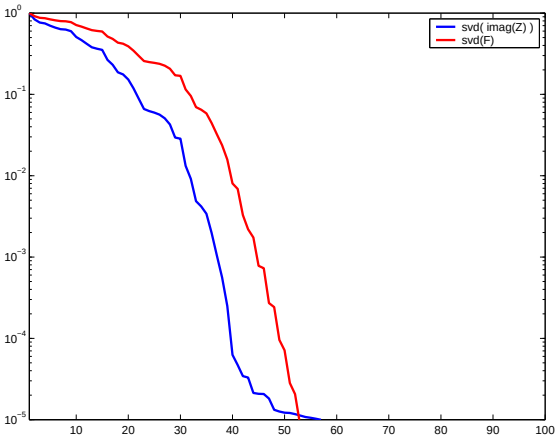
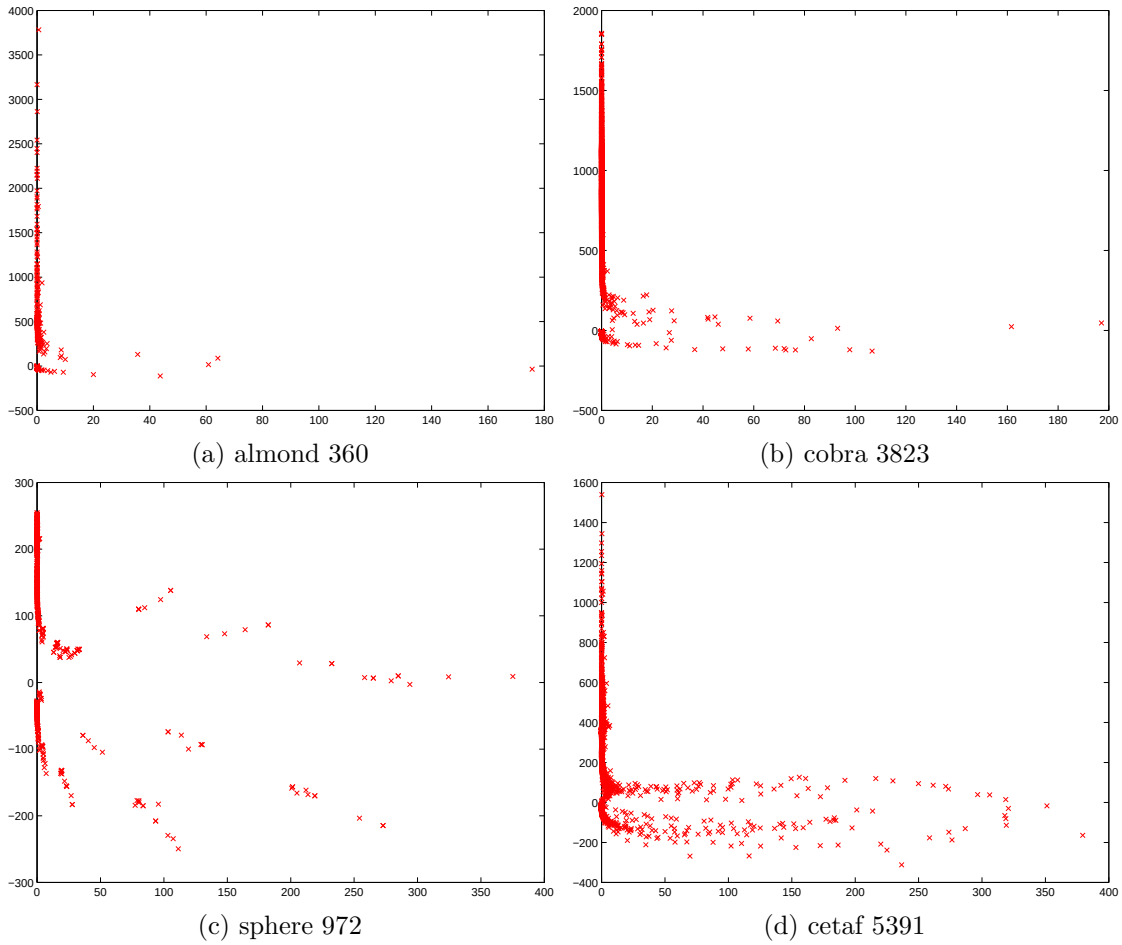


Figure 3.12: Singular values of $\mathbb{B}$ and of $F(0^o), F(1^o), \ldots, F(359^o)$.

We finally note that, in Table 1.12, we give the value $\kappa_2(F, \mathbb{Z}V)$, where $(F, \mathbb{Z}V)$ corresponds to the Krylov space generated by n iterations of the Arnoldi method applied to the starting vector F and the matrix $\mathbb{Z}$. This value is, in theory, an upper bound of the condition number of $\mathbb{Z}$ and is, in practice, a good approximation to the condition number of $\mathbb{Z}$. The value of $\kappa_2(F, \mathbb{Z}V)$ ranged from 27 for the cetaf to 5.9 for the almond. The matrices coming from the CFIE are therefore fairly well–conditioned.



(a) almond 360 (b) cobra 3823

(c) sphere 972 (d) cetaf 5391

Figure 3.13: Spectrum of the different matrices $\mathbb{Z}$ involved in an electromagnetic calculation.

3.3 A detailed presentation of the 3D code

3.3.1 The fast multipole method

3.3.1.1 Presentation of the fast multipole method

The Fast Multipole Method (FMM) introduced by Greengard and Rokhlin [71], provides an algorithm for computing approximate matrix–vector products for electromagnetic scattering problems. The method is fast in the sense that the computation of one matrix–vector product costs $\mathcal{O}(n \log n)$ arithmetic operations instead of the usual $\mathcal{O}(n^2)$ operations, and is approximate in the sense that the relative error with respect to the exact computation is around 10^{-3} [36, 130]. It is based on truncated series expansions of the Green's function for the electric–field integral equation (EFIE). The EFIE can be written as

$$E(x) = -\int_\Gamma \nabla G(x,x')\rho(x')d^3x' - \frac{ik}{c}\int_\Gamma G(x,x')\mathbf{J}(x')d^3x' + E^E(x), \qquad (3.7)$$

where E^E is the electric field due to external sources, $\mathbf{J}$(x) is the current density,

$\rho(x)$ is the charge density and the constants k and c are the *wavenumber* and the speed of light, respectively. The Green's function G can be expressed as

$$G(x, x') = \frac{e^{-ik|x-x'|}}{|x - x'|}. \tag{3.8}$$

The EFIE is converted into matrix equations by the Method of Moments [72]. The unknown current $\mathbf{J}(x)$ on the surface of the object is expanded into a set of basis functions $\mathbf{B}_i, i = 1, 2, ..., N$

$$\mathbf{J}(x) = \sum_{i=1}^{N} J_i \mathbf{B}_i(x).$$

This expansion is introduced in (3.7), and the discretized equation is applied to a set of test functions. A linear system is finally obtained. The entries in the coefficient matrix of the system are expressed in terms of surface integrals, and have the form

$$A_{KL} = \int \int G(x, y)\mathbf{B}_K(x) \cdot \mathbf{B}_L(y) dL(y) dK(x). \tag{3.9}$$

When m–point Gauss quadrature formulae are used to compute the surface integrals in (3.9), the entries of the coefficient matrix assume the form

$$A_{KL} = \sum_{i=1}^{m} \sum_{j=1}^{m} \omega_i \omega_j G(x_{K_i}, y_{L_j}) \mathbf{B}_K(x_{K_i}) \cdot \mathbf{B}_L(y_{L_j}). \tag{3.10}$$

Single and multilevel variants of the FMM exist and, for the multilevel algorithm, there are adaptive variants that handle inhomogeneous discretizations efficiently. In the one–level algorithm, the 3D obstacle is entirely enclosed in a large rectangular domain, and the domain is divided into eight boxes (four in 2D). Each box is recursively divided until the length of the edges of the boxes of the current level is small enough compared with the wavelength. The neighbourhood of a box is defined by the box itself and its 26 adjacent neighbours (eight in 2D). The interactions of the degrees of freedom within nearby boxes are computed exactly from (3.10), where the Green's function is expressed via (3.8). The contributions of far away cubes are computed approximately. For each far away box, the effect of a large number of degrees of freedom is concentrated into one multipole coefficient, that is computed using truncated series expansion of the Green's function

$$G(x, y) = \sum_{p=1}^{P} \psi_p(x) \phi_p(y). \tag{3.11}$$

The expansion (3.11) separates the Green's function into two sets of terms, ψ_i and ϕ_i, that depend on the observation point x and the source (or evaluation) point y, respectively. In (3.11) the origin of the expansion is near the source point and the observation point x is far away. Local coefficients for the observation cubes are computed by summing together multipole coefficients of far–away boxes, and the total effect of the far field on each observation point is evaluated from the local

expansions (see Figure 3.14 for a 2D illustration). Local and multipole coefficients can be computed in a preprocessing step; the approximate computation of the far field enables us to reduce the computational cost of the matrix–vector product to $\mathcal{O}(n^{3/2})$ in the basic one–level algorithm.
In the hierarchical multilevel algorithm, the obstacle is enclosed in a cube, the cube is divided into eight subcubes and each subcube is recursively divided until the size of the smallest box is generally half of a wavelength. Tree–structured data is used at all levels. In particular only non–empty cubes are indexed and recorded in the data structure. The resulting tree is called an *oct–tree* (see Figure 3.15) and we refer to its leaves as the leaf–boxes. The oct–tree provides a hierarchical representation of the computational domain partitioned by boxes. Each box has one parent in the oct–tree, except for the largest cube which encloses the whole domain, and up to eight children. Obviously, the leaf–boxes have no children. Multipole coefficients are computed for all cubes in the lowest level of the oct–tree, that is for the leaf–boxes. Multipole coefficients of the parent cubes in the hierarchy are computed by summing together contributions from the multipole coefficients of their children. The process is repeated recursively until the coarsest possible level. For each observation cube, an interaction list is defined that consists of those cubes that are not neighbours of the cube itself but whose parent is a neighbour of the cube's parent. In Figure 3.16 we denote by dashed lines the interaction list for the observation cube in the 2D case. The interactions of the degrees of freedom within neighbouring boxes are computed exactly, while the interactions between cubes in the interaction list are computed using the FMM. All the other interactions are computed hierarchically traversing the oct–tree at a coarser level. Both the computational cost and the memory requirement of the algorithm are of order $\mathcal{O}(n \log n)$. For further details on the algorithmic steps see [37, 105, 121] and [36, 39, 40, 41] for recent theoretical investigations. Parallel implementations of hierarchical methods have been described in [65, 66, 67, 70, 125, 139].
In Table 3.4, the time for assembling and applying a dense matrix is compared with the time for assembling the components for the FMM and applying the EFIE operator via the FMM. With the cetaf 5391, it turns out that the method is already interesting. The larger the mesh is, the more interesting the fast multipole method is; as can be seen when going from the cetaf test example to the Airbus test example.

		assembly		solve		# procs
	dof	FMM	dense	FMM	dense	
cetaf	5391	9.6	14.7	0.25	0.51	4
Airbus	23676	114.8	274.1	1.84	133.24	4

Table 3.4: Elapsed time to perform a matrix-vector product using the FMM and a BLAS-2 routine using the dense matrix.

The FMM makes problems affordable by reducing not only the time to perform the matrix–vector product but also the memory required to store the information needed to perform the product. For example, it is not possible to store dense matrices of order 50000 on the 8 Gigabyte disk space available on a two node Compaq (8 processors). Using the FMM, problems ten times larger can easily be solved.

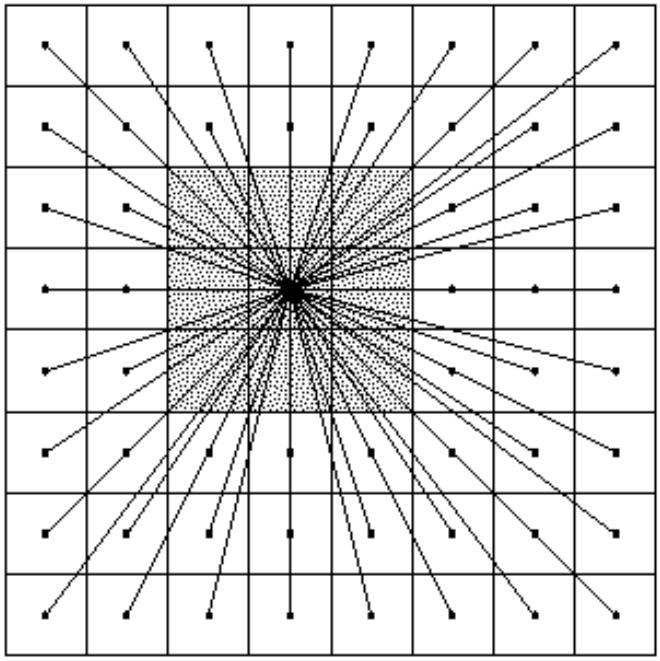


Figure 3.14: Interactions in the one–level FMM. For each leaf–box, the interactions with the gray neighbouring leaf–boxes are computed directly. The contribution of far away cubes are computed approximately. The multipole expansions of far away boxes are translated to local expansions for the leaf–box; these contributions are summed together and the total field induced by far away cubes is evaluated from local expansions.

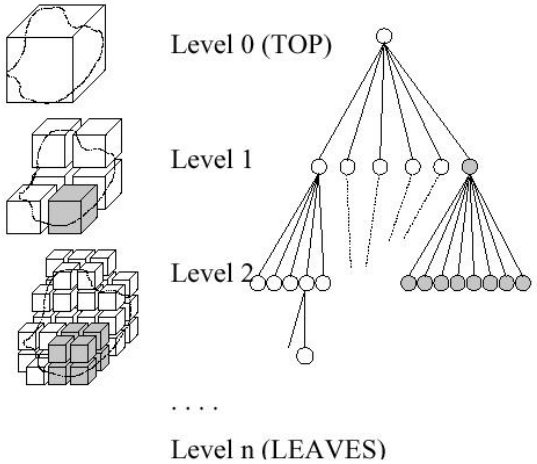


Figure 3.15: The oct–tree in the FMM algorithm. The maximum number of children is eight. The actual number corresponds to the subset of eight that intersect the object (courtesy of G. Sylvand, INRIA CERMICS).

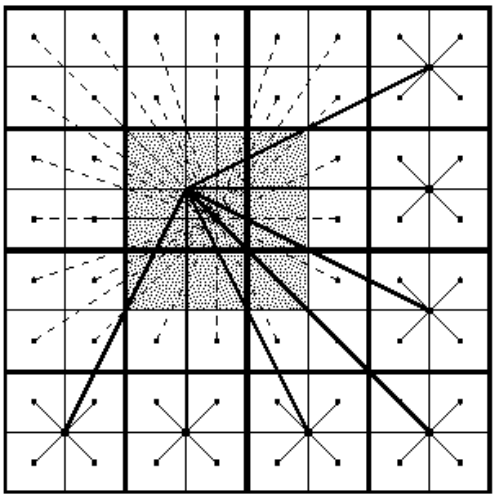


Figure 3.16: Interactions in the multilevel FMM. The interactions for the gray boxes are computed directly. We denote by dashed lines the interaction list for the observation box, that consists of those cubes that are not neighbours of the cube itself but whose parent is a neighbour of the cube's parent. The interactions of the cubes in the list are computed using the FMM. All the other interactions are computed hierarchically on a coarser level, denoted by solid lines.

3.3.1.2 The different accuracies

The code has some parameters in order to tune the features of the FMM. For instance, it is possible to change the size of the leaf boxes of the octree or to change the order of the expansion. Varying these parameters enables us to play with the FMM accuracy and the FMM computational cost; of course higher accuracy implies higher computational cost. This leads to define various FMM accuracies that correspond to different trade–off's between the computing cost and the accuracy of the matrix–vector multiplication. In that context, three FFM have been implemented, we call them *prec–1*, *prec–2* and *prec–3*. The less accurate and the fastest is *prec–1*, *prec–3* is the most accurate and the slowest. Sylvand [130, p.138] gives some estimates on the forward error associated with these FMM when solving electromagnetic problems using an iterative solver. The forward error is computed assuming that the exact solution is given by a direct solver. For the sphere of order 255792, he obtains relative errors of the order of 10^{-2}, 10^{-3} and $5 \cdot 10^{-4}$ for *prec–1*, *prec–2* and *prec–3* respectively.

As a default value, we use the *prec–3* accuracy. It should also be noticed that the default calculations are performed in single precision arithmetic (double precision is also available).

To illustrate the different computing costs associated with each accuracy, we give in Table 3.5 the average times observed for the FMM on different test cases.

	prec–1	*prec–2*	*prec–3*	*prec–3* (double)	# procs
Airbus 23676	1.3	1.7	1.8	2.0	4
Airbus 94704	3.5	4.3	4.6	5.1	8
Airbus 213084	6.2	9.1	11.1	13.0	8
Airbus 591900	17.0	25.5	29.8		8
sphere 40368	1.3	2.1	2.2	2.3	4
sphere 71148	2.3	3.4	3.8	4.4	4
sphere 161472	3.2	4.4	4.6	5.3	8
sphere 288300	4.7	7.5	8.3	9.5	8
sphere 549552	9.0	14.5	17.1	20.1	8
sphere 1023168		13.9	16.7		16

Table 3.5: Average elapsed time (s) for a matrix–vector product using the FMM with different level of accuracy.

3.3.1.3 The gathered fast multipole matrix–vector products

The code has been designed to take advantage of the memory hierarchy of the computer (cache memories) so that a Level 3 BLAS 3 effect can be observed if several matrix–vector products can be computed at the same time. We report in Table 3.6 on some computing times when the number of simultaneous matrix-vector products is varied. The times for the blocked method are roughly half than the time for a single FMM product. It can also be observed that this significant gain is already observed with relatively small blocks, about for 8 vectors. We show, in Section 3.3.3, a way of exploiting this when several right-hand sides have to be solved. We also remark that, for the simulations involving a dielectric, the corresponding FMM can exploit the data locality even more (as more floating–point operations have to be performed on the vectors); this results in a gain of about 13 on the coated cone sphere test example.

	1	2	4	8	16	32	# procs
Airbus 23676	1.8258	1.1794	0.9066	0.7949	0.7795	0.7874	4
Airbus 94704	4.6062	3.0609	2.3800	2.1287	2.0807	2.0938	8
Airbus 213084	11.0609	7.7233	6.2501	5.6018	5.4421	5.6690	8

Table 3.6: Average CPU time (s) for a matrix–vector product using the FMM where the block size is varied.

3.3.1.4 Consequence on the stopping criterion threshold

The problem we solve is an approximation of the original problem for several reasons (model error, discretization, use of the FMM, roundoff errors). In a backward error analysis framework, it seems therefore natural to set the tolerance of the iterative solver according to our estimate of the difference between the problem we solve and the problem we intend to solve.

In general, the stopping criterion is based on the normalized backward error

$$\frac{\|b - \mathbb{Z}x\|_2}{\|b\|_2} \leq \eta. \tag{3.12}$$

For example, in Figure 3.17, we plot two bistatic RCS computed for the sphere with $\eta = 10^{-1}$, $\eta = 10^{-2}$ and lastly the exact bistatic RCS computed using the analytic solution given by the Mie series. From these plots, the engineers advised us to select a stopping criterion threshold around 10^{-2}. Throughout this document, when convergence history are displayed, we always report the norm-wise backward error (residual norm normalized by the norm of the right-hand side) as a function of the iterations. In other words, we plot the quantity (or an approximation of this quantity)

$$\frac{\|F - \mathbb{Z}x_m\|_2}{\|F\|_2}$$

where x_m is the current iterate. This corresponds to the choice $\alpha^P = 0$ and $\beta^P = \|F\|_2$ for the stopping criterion of the preconditioned iterative method in Table (2.1).
The stopping criterion problem is further discussed in Section 3.8.2.

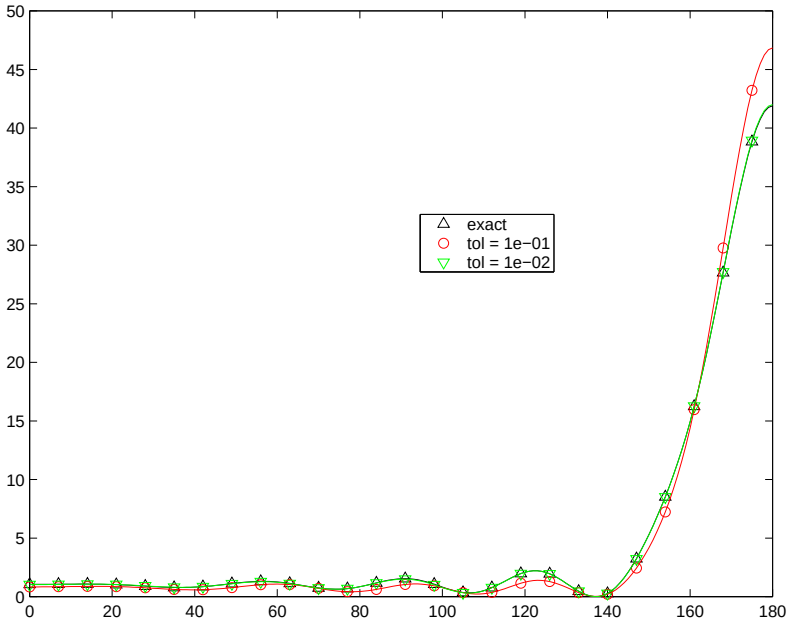


Figure 3.17: Bistatic RCS obtained for the sphere with different levels of backward error compared with the Mie series (exact solution).

3.3.2 Description of the preconditioners

In all the experiments we consider right preconditioning. The main reason is that we use GMRES quite extensively. Within this solver, the norm of the Arnoldi residual, used to evaluate our stopping criterion (see Section 2.2.1.2) is (in exact

arithmetic) the norm of the true residual associated with the linear system solved by GMRES. In that respect, the stopping criterion is related to the unpreconditioned linear system which is the one that is meaningful for the physical problem. On the contrary, with left preconditioning, the norm of the Arnoldi residual is the norm of the preconditioned residual; the stopping criterion is related to the preconditioned system and no longer to the original problem.

3.3.2.1 The Frobenius–norm minimization preconditioner

3.3.2.1.1 General presentation of the Frobenius norm minimization preconditioner

The preconditioner implemented in the code is a Frobenius norm minimization preconditioner. A complete description and study of the preconditioner is given in [23] in the context of our code.
Frobenius–norm minimization is a natural approach for building explicit preconditioners. This method computes a sparse approximate inverse as the matrix $M = \{m_{ij}\}$ which minimizes $\|I - MA\|_F$ (or $\|I - AM\|_F$ for right preconditioning) subject to certain sparsity constraints. Early references to this latter class can be found in [9, 10, 11, 53] and in [4] for some applications to boundary element matrices in electromagnetism. The Frobenius norm is usually chosen since it allows the decoupling of the constrained minimization problem into n independent linear least–squares problems, one for each column of M (when preconditioning from the right) or row of M (when preconditioning from the left).
The independence of these least–squares problems follows immediately from the identity:

$$\|I - MA\|_F^2 = \|I - AM^T\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_{j\bullet}\|_2^2 \tag{3.13}$$

where e_j is the j–th unit vector and $m_{j\bullet}$ is the column vector representing the j–th row of M.
In the case of right preconditioning, the analogous relation

$$\|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_{\bullet j}\|_2^2 \tag{3.14}$$

holds, where $m_{\bullet j}$ is the column vector representing the j–th column of M. Clearly, there is considerable scope for parallelism in this approach. The main issue for the computation of the sparse approximate inverse is the selection of the nonzero pattern of M, that is the set of indices

$$S = \{\ (i,j)\ \subseteq\ [1,n]^2 \mid m_{ij} = 0\ \}. \tag{3.15}$$

If the sparsity pattern of M is known, the nonzero structure for the j–th column of M is automatically determined, and defined as $J = \{i \in [1,n]\ s.t.\ (i,j) \in S\}$. The least–squares solution involves only the columns of A indexed by J; we indicate this subset by $A(:,J)$. When A is sparse, many rows in $A(:,J)$ are usually null, not affecting the solution of the least–squares problems (3.14). Thus if I is the set of indices corresponding to the nonzero rows in $A(:,J)$, and if we define by

$\hat{A} = A(I,J)$, by $\hat{m}_j = m_j(J)$, and by $\hat{e}_j = e_j(J)$, the actual "reduced" least–squares problems to solve are

$$min\|\hat{e}_j - \hat{A}\hat{m}_j\|_2, j = 1,..,n \tag{3.16}$$

Usually problems (3.16) have much smaller size than problems (3.14).
Two different approaches can be followed for the selection of the sparsity pattern of M: an adaptive technique that dynamically tries to identify the best structure for M; and a static technique, where the pattern of M is prescribed *a priori* based on some heuristics. The idea is to keep M reasonably sparse while trying to capture the "large" entries of the inverse, which are expected to contribute the most to the quality of the preconditioner. A static approach that requires an *a priori* nonzero pattern for the preconditioner, introduces significant scope for parallelism and has the advantage that the memory storage requirements and computational cost for the setup phase are known in advance. However, it can be very problem dependent.

3.3.2.1.2 Implementation of the Frobenius–norm minimization preconditioner in the fast multipole framework An efficient implementation of the Frobenius–norm minimization preconditioner in the FMM context exploits the box–wise partitioning of the domain. The subdivision into boxes of the computational domain uses geometric information from the obstacle, that is the spatial coordinates of its degrees of freedom. Carpentieri [23, Chapter 3] shown that this information can be profitably used to compute an effective *a priori* sparsity pattern for the approximate inverse. In the FMM implementation, we adopt the following criterion: the nonzero structure of each column of the preconditioner is defined by retaining all the edges within a given leaf–box and those in one level of neighbouring boxes. We recall that the neighbourhood of a box is defined by the box itself and its 26 adjacent neighbours (eight in 2D). The sparse approximation of the dense coefficient matrix is defined by retaining the entries associated with edges included in the given leaf–box as well as those belonging to the two levels of neighbours. The actual entries of the approximate inverse are computed column by column by solving independent least–squares problems. The main advantage of defining the pattern of the preconditioner and of the original sparsified matrix box–wise is that we only have to compute one QR factorization per leaf–box. Indeed the least–squares problems corresponding to edges within the same box are identical because they are defined using the same nonzero structure and the same entries of A. It means that the QR factorization can be performed once and reused many times, improving the efficiency of the computation significantly. The preconditioner has a sparse block structure; each block is dense and is associated with one leaf–box. Its construction can use a different partitioning from that used to approximate the dense coefficient matrix and represented by the oct–tree. The size of the smallest boxes in the partitioning associated with the preconditioner is a user–defined parameter that can be tuned to control the number of nonzeros computed per row, that is the density of the preconditioner. According to our criterion, the larger the size of the leaf–boxes, the larger the geometric neighbourhood that determines the sparsity structure of the columns of the preconditioner. Parallelism can be exploited by assigning disjoint subsets of leaf–boxes

to different processors and performing the least–squares solutions independently on each processor. Communication is required to get information on the entries of the coefficient matrix from neighbouring leaf–boxes.

3.3.2.1.3 Preliminary results In Figure 3.18, we study the influence of the size of the leaf-boxes on the density and on the efficiency of the preconditioner for cobra 3823. The density of the Frobenius preconditioner increases linearly with the size of the leaf-boxes while the number of iterations of GMRES to converge to a backward error of 10^{-3} eventually stagnates at around 30 iterations.

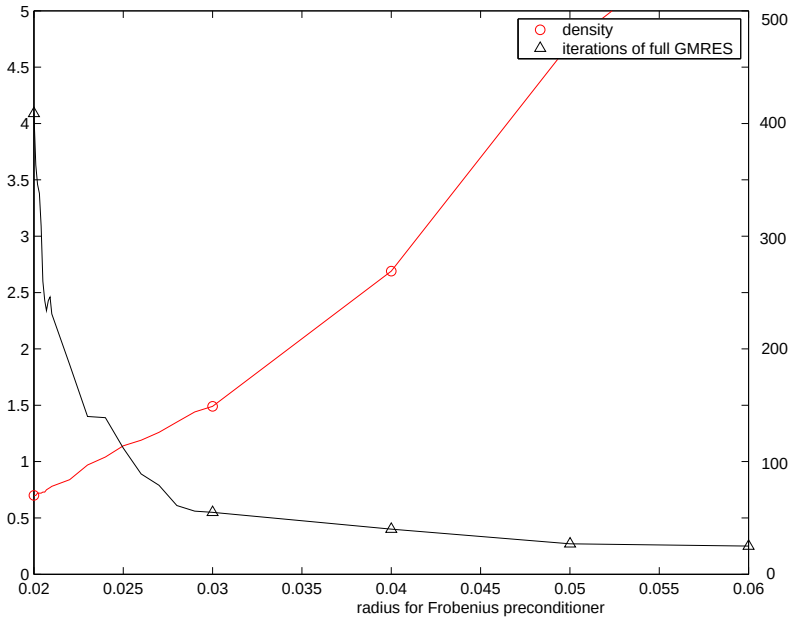


Figure 3.18: Number of iterations on the test example cobra 3823 with full GMRES for different sizes of the leaf-boxes used to define the sparsity pattern of the preconditioner. The corresponding density for the preconditioner is also given.

In the remainder of the manuscript, the size of the leaf-boxes of the oct-tree used to build the preconditioner is proportional to the wavelength. The ratio is set to 0.125. In Figure 3.19, we show the spectrum for each preconditioned matrix associated with four of our test examples. If we compare it with the spectrum of the matrices without a preconditioner given in Figure 3.13, we observe that the Frobenius norm minimization preconditioner manages to cluster nearly all the eigenvalues around one.

Since the size of the leaf-boxes of the oct-tree used to build the preconditioner is proportional to the wavelength, the number of nonzero entries per row in the preconditioner is constant (around 200) independently of the size of the problems. The amount of memory necessary to store the preconditioner, the time to assemble it and to apply it to a vector increase linearly with the size of the mesh. For example,

if we compare the last two rows in Table 3.7, we obtain

$$\frac{288300}{161472} = 1.79, \quad \frac{1108.1}{609.8} = 1.81 \quad \text{and} \quad \frac{1.09}{0.61} = 1.79.$$

The ratios are constant. This implies that the density of the preconditioner decreases when the size of the mesh increases. As a consequence, we expect the preconditioner to be less and less efficient when the size of the linear systems increases.

	nnz	nnz/row	density	assemble		apply	# procs
airbus 23676	5277284	223	0.94	183.8	(85.53%)	0.27	4
airbus 94704	24918048	263	0.28	605.3	(90.15%)	0.80	8
airbus 213084	58741250	276	0.13	1724.0	(92.65%)	1.78	8
sphere 40368	8694684	215	0.53	305.3	(89.35%)	0.29	4
sphere 71148	15210224	214	0.30	532.1	(87.25%)	0.54	4
sphere 161472	34899276	216	0.13	609.8	(88.05%)	0.61	8
sphere 288300	62580136	217	0.08	1108.1	(89.87%)	1.09	8

Table 3.7: Characteristics of the Frobenius norm minimizer preconditioner. For some of the test examples, we give the number of nonzeros (nnz), the number of nonzeros per row and the density of the preconditioner. We also give the elapsed time for assemblying it and, in parentheses, the percentage that it represents in the whole assembly phase (assembly of the FMM + assembly of the preconditioner). Finally, we give the elapsed time for applying the preconditioner to a single vector.

3.3.2.2 Flexible Krylov Variants

In this section, we describe some experiments on an inner-outer scheme introduced in [23, 26]. A preconditioner for $\mathbb{Z}$ is a matrix, M, that tries to approximate the inverse of $\mathbb{Z}$ at low computational and memory cost. Given x, it computes $y = Mx$ to obtain a the compromise between: (a) y is a good approximation to $\mathbb{Z}^{-1}x$ and (b) y is cheap to compute. In that context, a natural way to compute y is to approximatively solve the linear system $\mathbb{Z}x = y$ via an iterative method. Since at each step, the preconditioner *changes*, one variant of GMRES that allows this type of preconditioning is called flexible GMRES. For the preconditioner solver (also called the inner solver), we use GMRES since Carpentieri [23] observed that GMRES performs better in our case than SQMR [54] or transpose free QMR [58]. Therefore the solver results in an inner GMRES scheme embedded in an outer GMRES scheme. Let a be the value of the restart parameter of the outer GMRES and b be the value of the restart parameter of the inner GMRES, then the amount of memory required by the method is $(a + 2b)$ vectors.
The trade–off quality v.s. time of the preconditioning step is controlled via three parameters: (a) the stopping criterion of the inner scheme, that is set relatively high, (b) the maximum number of iterations of the inner scheme, that is set relatively low so that only a few steps of the method are performed, (c) the accuracy on the inner matrix–vector product; for that latter parameter we generally choose a less accurate FMM product to perform the matrix–vector product than that used for the outer iterations.

3.3.2.3 The spectral low rank update preconditioner

The construction of the Frobenius-norm minimization preconditioner is inherently local. Each degree of freedom in the approximate inverse is coupled to only a very few neighbours and this compact support does not allow an exchange of global information.
In [24], Carpentieri, Duff and Giraud proposed a refinement technique which enhances the robustness of the approximate inverse on large problems. Related previous work can also be found in [47, 8, 81, 115]. The method is based on the introduction of low-rank updates computed by exploiting spectral information of the preconditioned matrix. The Frobenius-norm minimization preconditioner succeeds in clustering most of the eigenvalues far from the origin, nevertheless eigenvalues near zero can potentially slow down convergence. The purpose of the preconditioner proposed in [24] is to shift the smallest eigenvalues in magnitude of the preconditioned matrix so they are close to one.
Most of the schemes exploiting spectral information are combined with the GMRES procedure as they derive spectral information directly from its internal Arnoldi process. In some preliminary work, we consider an explicit eigencomputation which makes the preconditioner independent of the Krylov solver used for the actual solution of the linear system.
We present the spectral low rank update in a general framework. Consider the solution of the linear system

$$\mathbb{Z}x = b, \tag{3.17}$$

where $\mathbb{Z}$ is a $m \times m$ complex unsymmetric nonsingular matrix, and x and b are vectors of size m.
We assume that the matrix $\mathbb{Z}$ is diagonalizable, that is:

$$\mathbb{Z} = V \Lambda V^{-1}, \tag{3.18}$$

with $\Lambda = diag(\lambda_i)$, where $|\lambda_1| \leq \ldots \leq |\lambda_m|$ are the eigenvalues and $V = (v_i)$ the associated right eigenvectors. We denote by $U = (u_i)$ the associated left eigenvectors. Let V_ε be the set of right eigenvectors associated with the set of eigenvalues λ_i with $|\lambda_i| \leq \varepsilon$. Similarly, we define by U_ε the corresponding subset of left eigenvectors.
Carpentieri et al [24] present the following theorem.

Theorem 3.3.1 *Let*

$$\mathbb{Z}_c = U_\varepsilon^H \mathbb{Z} V_\varepsilon,$$
$$M_c = V_\varepsilon \mathbb{Z}_c^{-1} U_\varepsilon^H$$

and

$$M = I_m + M_c.$$

Then $M\mathbb{Z}$ and $\mathbb{Z}M$ are diagonalisable and we have $M\mathbb{Z} = \mathbb{Z}M = V diag(\eta_i) V^{-1}$ with

$$\begin{cases} \eta_i = \lambda_i & \text{if} \quad |\lambda_i| > \varepsilon, \\ \eta_i = 1 + \lambda_i & \text{if} \quad |\lambda_i| \leq \varepsilon. \end{cases}$$

$\mathbb{Z}_c$ represents the projection of the matrix $\mathbb{Z}$ on the coarse space defined by the approximate eigenvectors associated with its smallest eigenvalues. The low rank spectral update is defined by M_c. Computing U_ε and V_ε can be quite expensive. In our case, the matrix–vector product with the transpose of the FMM operator is not available so that obtaining the left eigenvectors is absolutely not practicable. To address this situation, the following theorem is given in [24].

Theorem 3.3.2 *Let W be such that*

$$\tilde{\mathbb{Z}}_c = W^H \mathbb{Z} V_\varepsilon \text{ has full rank,}$$

$$\tilde{M}_c = V_\varepsilon \tilde{\mathbb{Z}}_c^{-1} W^H$$

and

$$\tilde{M} = I_m + \tilde{M}_c.$$

Then $\tilde{M}\mathbb{Z}$ and $\mathbb{Z}\tilde{M}$ are similar to a matrix whose eigenvalues are

$$\begin{cases} \eta_i = \lambda_i & \text{if} \quad |\lambda_i| > \varepsilon, \\ \eta_i = 1 + \lambda_i & \text{if} \quad |\lambda_i| \leq \varepsilon. \end{cases}$$

We should point out that, if the symmetry of the preconditioner is required (EFIE), then an obvious choice exists. For left preconditioning, we can set $W = \bar{V_\varepsilon}$. It can be noticed that, in the SPD case, this choice leads to a preconditioner that has a similar form to those proposed in [27] for two–level preconditioners in non–overlapping domain decomposition.
The spectral low rank updates that are described in Theorems 3.3.1 and 3.3.2 shift the smallest eigenvalues λ_i to $1+\lambda_i$. Indeed with a slight modification, it is possible to shift the eigenvalues to any desired value. We give here an alternative variant of the spectral low rank update (variant of Theorem 3.3.1) that shifts the k smallest eigenvalues to $|\lambda|$:

$$M_{SLRU} = I_m + U \left(|\lambda| D^{-1} - I_k \right) V^H, \tag{3.19}$$

where D is the k–by–k diagonal matrix with the k smallest eigenvalues on the diagonal. For the associated variant of Theorem 3.3.2, we obtain:

$$\tilde{M}_{SLRU} = I_m + U \left(|\lambda| I_k - D \right) (W^H \mathbb{Z} U)^{-1} W^H. \tag{3.20}$$

It is interesting to relate this work with [47, 8]. Given an invariant subspace P of $\mathbb{Z}$, Burrage, Erhel and Pohl [47] considered the orthonormal basis Z for P. The preconditioner they give has the following form:

$$M_{\text{BEP}} = I_m + Z \left(|\lambda| (Z^H \mathbb{Z} Z)^{-1} - I_k \right) Z^H. \tag{3.21}$$

Let consider $\mathbb{Z}$ spanning U so that there exists H a k–by–k matrix such that

$$Z = UH.$$

Z is an orthonormal basis of $\text{span}(U)$. Since Z is invariant under transformations by $\mathbb{Z}$ we have

$$\mathbb{Z} Z = Z (Z^H \mathbb{Z} Z).$$

Combining these two equations with $AU = UD$, we obtain

$$D = H(Z^H \mathbb{Z} U).$$

We then consider the preconditioner M_{BEP}:

$$\begin{aligned} M_{\text{BEP}} &= I_m + U\left(|\lambda|(Z^H\mathbb{Z}U)^{-1} - H\right)Z^H, \\ &= I_m + U\left(|\lambda| I_k - H(Z^H\mathbb{Z}U)\right)(Z^H\mathbb{Z}U)^{-1}Z^H, \\ &= I_m + U\left(|\lambda| I_k - D\right)(Z^H\mathbb{Z}U)^{-1}Z^H, \end{aligned} \tag{3.22}$$

That is

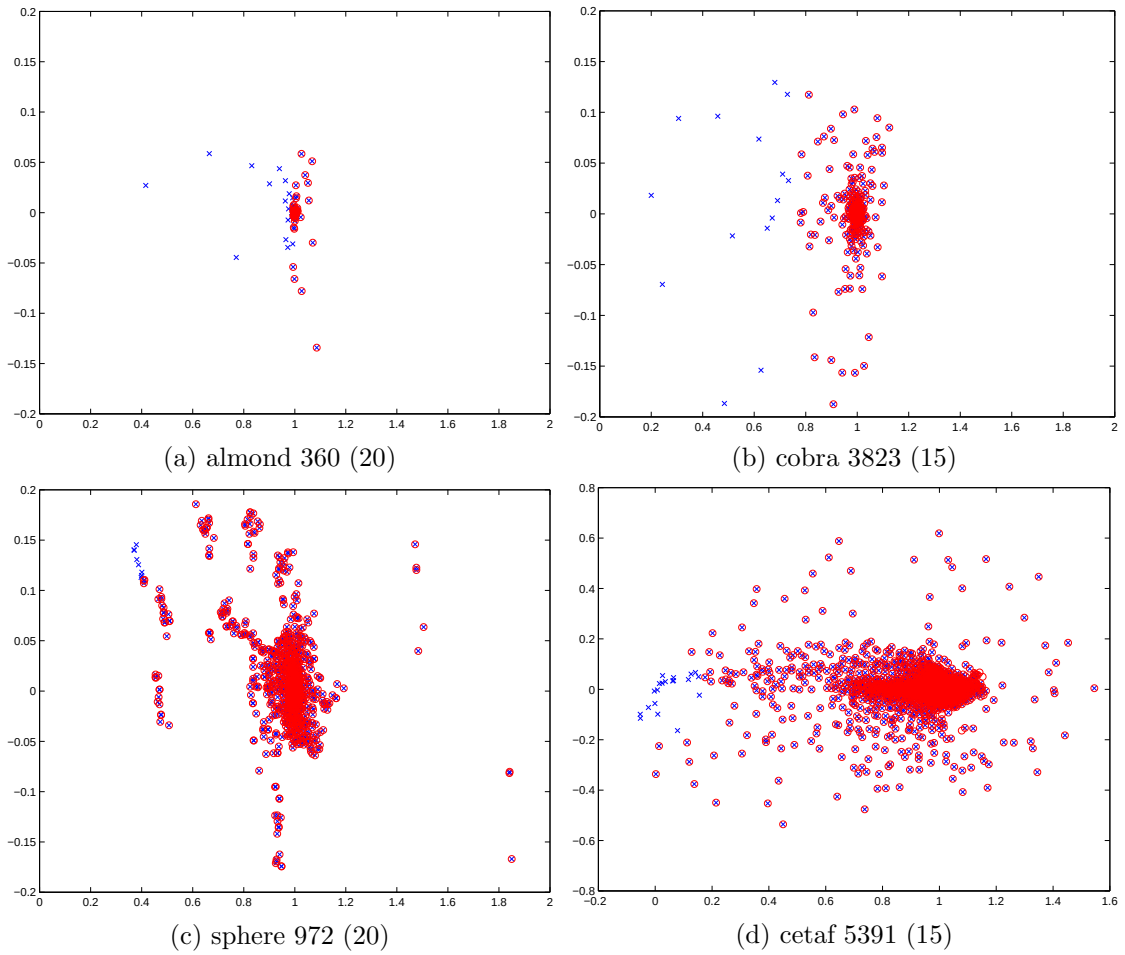
$$M_{\text{BEP}} = \tilde{M}_{SLRU}. \tag{3.23}$$

We recover the formulation of the spectral update given in [24] with $W = Z$.
In Figure 3.19, we plot with the symbol "x" the spectrum of the matrix preconditioned with the Frobenius preconditioner. For the sake of comparison, we recall that the spectra of the non–preconditioned matrices are given in Figure 3.13. The Frobenius-norm minimization preconditioner succeeds in clustering most of the eigenvalues far from the origin. We observe a big cluster near $(1.0, 0.0)$ in the spectrum of the preconditioned matrix. This kind of distribution is highly desirable to get fast convergence of Krylov solvers. Nevertheless the eigenvalues nearest to zero potentially can slow down the convergence. Using the symbol "o" we plot, in Figure 3.19, the spectrum of the matrix preconditioned with the Frobenius preconditioner and the spectral low rank update (equation (3.19)). We observe that the k smallest eigenvalues of the matrix preconditioned with the Frobenius preconditioner are shifted close to one, in agreement with Theorem 3.3.2. Consequently, we expect the Krylov solver to perform better with the spectral low rank update.
Several remarks are in order:

1. The choice of the shifts on $|\lambda_{\max}|$ made by Burrage, Erhel and Pohl [47] is not obvious. For example, let us consider a diagonalisable matrix with $(n-1)$ eigenvalues equal to -1 and one eigenvalue equal to -0.5; then the smallest eigenvalue in magnitude is -0.5. However, it seems more appropriate to shift it to minus one rather than to one. In our case, because we use this idea in combination with the Frobenius–norm minimization preconditioner, many eigenvalues are already clustered around one. It is therefore natural to shift the smallest eigenvalues close to one.

2. In equation (3.19), D may be ill–conditioned and so we can expect some trouble when using the spectral low rank update. In our experiments, D has always been rather well–conditioned.

The study of the spectral update preconditioner for large electromagnetism calculation is part of the ongoing PhD. thesis of Emeric Martin at CERFACS. It has been observed that the more eigenvalues that are shifted, the better the convergence of GMRES. However, the spectral low rank update has a cost at each iteration and also a preprocessing cost (currently the computation of the eigenvectors associated



(a) almond 360 (20) (b) cobra 3823 (15)

(c) sphere 972 (20) (d) cetaf 5391 (15)

Figure 3.19: Spectrum of the matrix preconditioned with Frobenius and the matrix preconditioned with Frobenius and Spectral Update. – x eigenvalues of the matrix preconditioned with Frobenius ; o spectrum of the matrix preconditioned with Frobenius and Spectral Update. The number of shifted eigenvalues is given in parentheses.

with the eigenvalues of smallest magnitude is performed in a pre-processing phase using ARPACK in forward mode). This cost grows with the number of eigenvectors obtained and a trade-off needs to be found. Following the choices of Emeric Martin, we take 15 eigenvectors for the cobra, 60 eigenvectors for the almond and 20 eigenvectors for the others. Note that these values are taken independently of the mesh size. The cost of the computation of 20 eigenvectors depends on the test problem and varies from 500 up to 1000 matrix-vector products. In Table 3.8, we give the times to assemble and apply the FMM, the Frobenius preconditioner and the spectral update (SLRU, the number of shifted eigenvalues is given in bracket). Note that, in the case of the spectral update, the eigenvectors are stored on disk and so the assembly of the spectral low rank update is rather fast and mainly consists in reading the eigenvectors from disk.

	assembly			application			# procs
	FMM	Frob	SLRU	FMM	Frob	SLRU	
cetaf 5391	9.6	631.8	3.7 (20)	0.25	0.11	0.02 (20)	4
Airbus 23676	114.8	184.8	36.7 (20)	1.86	0.29	0.03 (20)	4
cobra 60695	33.3	120.2	30.6 (15)	1.93	0.22	0.03 (15)	8
almond104793	53.1	345.8	182.8 (60)	2.92	0.41	0.10 (60)	8
almond104793	53.1	345.8	59.3 (20)	2.92	0.41	0.05 (20)	8

Table 3.8: Time of assembly and application of the FMM (prec-3), the Frobenius preconditioner and the spectral update (the number of shifted eigenvalues is given in bracket).

3.3.3 The remaining numerical kernels

In the code, the vectors are stored out–of–core and we do not want to be concerned with their storage. For that purpose, EADS-CCR has provided us with all the basic operations to handle the vectors. The basic kernel operations are listed in Table 3.9 using a BLAS-like notation.

zaxpy(α,x,y)	$x \leftarrow x + \alpha y$
zdscal(α,x,y)	$x \leftarrow \alpha x$
zcopy(x,y)	$x \leftarrow y$
zdot(x,y)	$x^H y$
precond(x)	$x \leftarrow Mx$
matvec(x)	$x \leftarrow \mathbb{Z}x$

Table 3.9: "kernel" operations needed by the iterative solvers.

The iterative solvers are implemented using a reverse communication mechanism (see Chapter 2). The time for an iteration is directly related to the time for each kernel operation and the number of calls to these operations. It is therefore important to know the average time consumed by each call to the kernel operations. We report these in Table 3.10. They are also given relative to the FMM time in Table 3.11.
First of all, we notice that the two most time consuming operations are of course the matrix–vector product and the preconditioning. In all our experiments, the number of iterations for a given solver is close to the number of times a matrix–vector product, and a preconditioning application, is performed. Consequently, the first goal is to reduce the number of iterations for each solver.
This last sentence should however be further discussed. The general principle of the design of iterative methods to handle several right–hand sides is to share, among all the right–hand sides and via an intensive use of vector–vector operations, the information obtained from a matrix–vector product. In general, the computational time of these operations is neglected and only the number of iterations are taken into account. In our case, the matrix–vector product is performed using the FMM. This numerical kernel is particularly well implemented and efficient. From Table 3.10, we see that, in most cases, 1000 dot products (say) are more expensive than one matrix–vector multiplication. In Section 3.7.1.1, we show that the overall time for the vector–vector operations is, for our iterative solvers with multiple right–hand sides, of the same order, and in general larger, than the overall time for the matrix–vector

products. To illustrate the importance of the vector–vector operations, we give the results in Table 3.16. For both the Airbus 23676 and sphere 71148 , the full GMRES method converges in less iterations than the GMRES(30) method. However, the elapsed time is larger for the full GMRES method than for the GMRES(30) method. Although, the full GMRES method performs less matrix–vector product than the GMRES(30) method, it performs more orthogonalization among vectors. Since the time spent in the orthogonalizations is not negligible, the full GMRES method takes more time to converge than the GMRES(30) method.

	zaxpy	zdscal	zcopy	zdot	precond	FMM	# proc.
Airbus 23676	0.0027	0.0026	0.0019	0.0024	0.27	1.83	4
Airbus 94704	0.0066	0.0063	0.0046	0.0051	0.80	4.61	8
Airbus 213084	0.0148	0.0142	0.0105	0.0106	1.78	11.13	8
sphere 40368	0.0044	0.0042	0.0030	0.0036	0.29	2.16	4
sphere 71148	0.0079	0.0059	0.0079	0.0061	0.54	3.84	4
sphere 161472	0.0131	0.0129	0.0093	0.0098	0.61	4.63	8
sphere 288300	0.0202	0.0194	0.0143	0.0140	1.09	8.25	8
cetaf 5391	0.0010	0.0001	0.0001	0.0013	0.13	0.25	4
cobra 60695	0.0045	0.0003	0.0004	0.0040	0.22	1.93	8
almond 104793	0.0082	0.0006	0.0007	0.0064	0.41	2.92	8

Table 3.10: Elapsed time (s) for each basic operation needed by the iterative solvers.

	100 zaxpy	100 zdscal	100 zcopy	100 zdot	precond	FMM	# proc.
Airbus 23676	14.75	14.21	10.38	13.11	14.75	100	4
Airbus 94704	14.32	13.67	9.98	11.06	17.35	100	8
Airbus 213084	13.30	12.76	9.43	9.52	15.99	100	8
sphere 40368	20.37	19.44	13.89	16.67	13.42	100	4
sphere 71148	20.57	15.36	20.57	15.89	14.06	100	4
sphere 161472	28.29	27.86	20.09	21.17	13.17	100	8
sphere 288300	24.48	23.52	17.33	16.97	13.21	100	8
cetaf 5391	40.80	2.40	2.40	50.80	52.00	100	4
cobra 60695	23.47	1.66	2.18	20.89	11.40	100	8
almond 104793	28.05	1.89	2.47	21.99	14.04	100	8

Table 3.11: Percentage of time required for 100 calls to each basic operations with respect to one call to the FMM.

An efficient way to reduce significantly the time of the zaxpy operations is to block them. The iterative solvers dealing with several right–hand sides use and abuse of this property. However, in our out–of–core implementation, this strategy does not at all improve the computational time. For instance, n consecutive zaxpy's performed simultaneously take the same time as n successive zaxpy's.
In Table 3.12, we compare the elapsed times for 1000 dot products performed simultaneously with The elapsed times for 1000 dot products performed sequentially. the benefit is clear. The implementation of the vector–vector operation for vectors held out–of–core deserves further study.

		successive ZDOT	gathered ZDOT	# procs
cetaf	5391	1.27(1)	0.13(20)	4
Airbus	23676	2.48(1)	0.46(20)	4
cobra	60695	4.03(1)	0.59(17)	8
almond	104793	6.42(1)	0.78(60)	8
almond	104793	6.42(1)	0.99(20)	8

Table 3.12: Elapsed time (s) for 1000 dot products performed one at a time and 1000 dot products gathered, the size of the set varies and is given in parentheses.

3.3.4 Parallel scalability: an insight

Thanks to the reverse communication, the parallelism of the code is implemented independently of the solver. All the operations needed by the solvers are parallel. A detailed study of the parallel implementation is out of the scope of this section. For the parallel implementation of the fast multipole method we refer to [130] and for the parallel implementation of the Frobenius–norm minimizer preconditioner we refer to [23, 130]. Nevertheless, we show, in Table 3.13, the results of some basic scalability experiments with the code. Going from four processors to eight, we observe how the basic parallel operations scale. The main two operations, the fast multipole method and the preconditioner, get a speedup of nearly two. Note that the dot product only has a speedup of 1.39 .

zaxpy	zdscal	zcopy	zdot	precond	FMM	assemb. FMM	assemb. precond	# proc.
0.0079	0.0059	0.0079	0.0061	0.54	3.84	532.1	590.6	4
0.0050	0.0048	0.0035	0.0044	0.28	2.11	270.4	309.9	8
1.58	1.23	2.25	1.39	1.93	1.82	1.97	1.91	

Table 3.13: Elapsed time (s) for each basic operation on 4 and 8 processors for the sphere 71148 test example. The corresponding speedups are given in the last row.

3.3.5 Preliminary results

3.3.5.1 Direct versus iterative solvers

The purpose of this chapter is the study of the iterative methods in the context of electromagnetism calculations. For the sake of completeness, we recall that direct methods might be highly suited to that framework as long as enough memory (RAM and disk) can be afforded. Direct solvers are particularly attractive when dealing with multiple right–hand sides since the factorization is performed only once for all the right–hand sides. Furthermore, since the matrices are dense and complex, the implementations of these solvers can benefit from the performance of the Level 3 BLAS and can reach high substained Megaflop rates; finally they can be parallelized in a ScaLAPACK fashion [3, 31]. In Table 3.14, we compare the elapsed times of the direct and the iterative solvers on the Airbus 23676 with 181 right–hand sides. On that example, the direct solver is clearly the fastest and should be preferred when

more than 45 right–hand sides have to be solved.

direct solver		iterative solver	
assembly of the dense matrix	57.1	assembly of the fmm matrix	8.93
factorization of the matrix	449.9	construction of the preconditionner	80.37
181 solves	15.9	GMRES solver	8982.5
elapsed time	522.9	elapsed time	9071.8

Table 3.14: Elapsed times on 8 processors for the direct and iterative (GMRES with Frobenius–norm minimization) solvers on the Airbus 23676 test example with 181 right–hand sides.

However, no definit conclusion can be reached for this comparison between direct and iterative solvers. While the cost of the direct method (when affordable) only depends on the size of the linear systems, the performance of the iterative techniques is problem dependent. For example, in the case of the coated cone sphere where the convergence is observed of 20 iterations for each right–hand side (20 is an average), the elapsed time for the 181 solves using GMRES is 10 hours (3617 cumulated iterations) whereas the direct solver needs more than 20 hours. (These times were observed on a cluster of 25 Pentium4 running at 2GHz by Guillaume Sylvand).
To conclude this short comparison, we shall mention that, thanks to the use of the fast multipole technique, iterative methods are the only alternative for the solution of problems with a few tens of thousand of unknowns. On this range of problems, the direct approaches are no longer affordable since they are too demanding in term of CPU and memory.

3.3.5.2 Other integral equation formulations

In Section 3.1, we have only described the EFIE formulation. Nevertheless, we mentioned that other formulations exist, for example the MFIE (magnetic–field integral equation) and the CFIE (combined field integral equation). These last two formulations have a primary limitation, that is, they are only defined on closed objects. Their main advantage is that they usually give rise to linear systems that are easier to solve with the iterative solvers. In Table 3.15, we illustrate this latter observation. It can be seen that the preconditioned EFIE requires 172 iterations whereas the unpreconditioned CFIE(0.2) only requires 30 iterations. Consequently the elapsed time is ten times larger for the EFIE than for the CFIE(0.2). For that geometry, we shall mention that all the three approaches enable us to get the correct bistatic RCS. Because all but the cobra example presented in Section 3.2.4

	No preconditioner			Frob preconditioner		
	EFIE	CFIE(0.2)	MFIE	EFIE	CFIE(0.2)	MFIE
# iterations	354	30	130	172	12	86
assembling time	49.5	76.8	67.9	399.1	461.1	425.4
solution time	1968.7	128.2	549.4	798.4	66.6	371.8
total time	2018.2	205	617.3	1197.5	527.7	797.2

Table 3.15: Elapsed time on 8 processor Compaq for solving the Airbus 23676 test problem using the three integral equation formulations.

are closed objects, CFIE(0.2) is applicable and then seems to be the formulation of choice. Nevertheless, we only consider the EFIE formulation in our work for many reasons. The first one is that, for realistic aircraft geometries, the windows are included; consequently the airplane can no longer be considered as a closed object. The second is that, even though on closed objects all the formulations give, in theory, the same solution, it appears in practice that the solution using the EFIE is sometimes better. This was revealed on several geometries during the JINA 2002 workshop and no explanation of that behaviour has been given to date. For the sake of reliability, the EFIE formulation is often the method of choice even on closed objects. These points justify the choice made in this study.
Finally, we mention that other integral equations exist that enable us to perform the same numerical simulation. For instance, Després [42] proposed a set of equations that also are widely used.

3.3.5.3 Combining the Frobenius–norm minimization preconditioner with flexible inner–outer iterations

When only a few right–hand sides have to be solved, the joint effort involving EADS–CCR, CERMICS and CERFACS has proposed a strategy to solve problems with upto a few tens of millions of unknowns [130]. The iterative solver used for those large simulations combines the fast multipole method with an inner–outer iteration scheme. The outer solver is the Flexible GMRES that uses, as its preconditioner, a few iterations of preconditioned GMRES. For the inner solver, the preconditioner is the Frobenius–norm minimization technique and the matrix–vector product is performed using a less accurate fast multipole method than that used for outer iterations.
To illustrate the attractive behaviour of the solver, we report in Table 3.16 on some experiments for the various discretizations of a sphere and the Airbus. For these experiments, we set the restart parameters of the various GMRES algorithms (classical and flexible) so that they all use the same amount of storage. For the inner-outer scheme, the inner GMRES consists in one restart of GMRES(20) and the value of the restart for the FGMRES is set to 5 . The total amount of vectors used by the resulting solver, denoted by flexible GMRES(20,5), is 30 (see Section 2.3). Regarding the accuracy of the inner fast multipole methods we consider *prec–2*, *prec–1*, to define FGMRES(20,5)(2), FGMRES(20,5)(1) respectively. The number of iterations for FGMRES(20,5) is given in a $x - y$ format where x denotes the number of outer iterations (using FMM *prec–3*) and y denotes the number of inner iterations (using FMM *prec–1* or *prec–2*).
For the *small* Airbus test examples, restarted GMRES or the full GMRES method may perform better in terms of iterations or elapsed time than the FGMRES(20,5) but the difference is slight. When the size becomes large (Airbus 213084 and Airbus 591900) the FGMRES(20,5) method outperforms the standard methods in terms of elapsed time. This is mainly due to three reasons. Firstly the cost of the orthogonalization scheme in flexible GMRES(20,5) is small compared to its cost in full GMRES; secondly, the inner iterations are performed with a less accurate but faster matrix–vector product. Finally, it appears that the flexible scheme manages

to converge on some large problems where the classical restarted scheme does not converge and the full GMRES runs out of memory (disk).
The same observation is true on the series of spheres. It should be noticed that on this academic example, the use of the less accurate FMM, *prec–1*, induces a significant delay in the convergence of the flexible GMRES(20,5) compared with the same scheme using the inner *prec–2* FMM. On the sphere 1023168, full GMRES runs out of memory, restarted GMRES runs out of iterations (and time limit on the computer batch queue), and FGMRES(20,5) is the only alternative that succeds in solving this one million degree of freedom problem. This confirms the results observed in [23, 26, 130].

			# iter	time (s)	(#*procs*)
Airbus	23676	full GMRES	71	207.0	(4)
Airbus	23676	GMRES(30)	81	184.7	(4)
Airbus	23676	FGMRES(20,5)(2)	5 – 100	222.7	(4)
Airbus	23676	FGMRES(20,5)(1)	5 – 100	184.2	(4)
Airbus	94704	full GMRES	100	678.8	(8)
Airbus	94704	GMRES(30)	131	772.8	(8)
Airbus	94704	FGMRES(20,5)(2)	8 – 160	930.3	(8)
Airbus	94704	FGMRES(20,5)(1)	8 – 160	736.8	(8)
Airbus	213084	full GMRES	123	1868.8	(8)
Airbus	213084	GMRES(30)	343	4556.8	(8)
Airbus	213084	FGMRES(20,5)(2)	9 – 180	2038.0	(8)
Airbus	213084	FGMRES(20,5)(1)	9 – 180	1524.1	(8)
Airbus	591900	full GMRES	146	6498.5	(8)
Airbus	591900	GMRES(30)	+×	+×	(8)
Airbus	591900	FGMRES(20,5)(2)	10 – 200	6412.6	(8)
Airbus	591900	FGMRES(20,5)(1)	10 – 200	4790.3	(8)
sphere	40368	full GMRES	61	214.2	(4)
sphere	40368	GMRES(30)	80	222.2	(4)
sphere	40368	FGMRES(20,5)(2)	4 – 80	257.3	(4)
sphere	40368	FGMRES(20,5)(1)	6 – 120	241.2	(4)
sphere	71148	full GMRES	66	388.4	(4)
sphere	71148	GMRES(30)	76	379.6	(4)
sphere	71148	FGMRES(20,5)(2)	5 – 100	454.5	(4)
sphere	71148	FGMRES(20,5)(1)	6 – 120	410.3	(4)
sphere	161472	full GMRES	77	549.0	(8)
sphere	161472	GMRES(30)	126	817.6	(8)
sphere	161472	FGMRES(20,5)(2)	6 – 120	736.2	(8)
sphere	161472	FGMRES(20,5)(1)	8 – 160	744.2	(8)
sphere	288300	full GMRES	131	1649.0	(8)
sphere	288300	GMRES(30)	311	3695.2	(8)
sphere	288300	FGMRES(20,5)(2)	10 – 200	2214.1	(8)
sphere	288300	FGMRES(20,5)(1)	14 – 280	1972.9	(8)
sphere	549552	full GMRES	154	4365.1	(8)
sphere	549552	GMRES(30)	345	7977.4	(8)
sphere	549552	FGMRES(20,5)(2)	11 – 220	4530.9	(8)
sphere	549552	FGMRES(20,5)(1)	25 – 500	6769.3	(8)
sphere	1023168	full GMRES	–	–	(16)
sphere	1023168	GMRES(30)	+×	+×	(16)
sphere	1023168	FGMRES(20,5)(2)	11 – 220	4080.2	(16)

Table 3.16: Number of iterations and elapsed time (s) on different test cases for full GMRES, GMRES(30) and FGMRES(20,5). × means that the number of iterations exceeded 500, + means that the times exceeded 10000 seconds and − means that the code runs out of memory.

3.4 Numerical behaviour of the linear solvers for one right–hand side

3.4.1 The GMRES–DR solver

3.4.1.1 Analysis of the convergence of GMRES–DR

In Section 2.4, we present the theoretical background of the GMRES–DR method and its implementation. In this section, we investigate its numerical behaviour for

solving large linear systems from electromagnetism.
We first consider GMRES–DR(30,20) on the cetaf test example. The right–hand side corresponds to the direction $(\theta = 60^o, \varphi = 0^o)$. The convergence history is shown in Figure 3.20. For the sake of illustration, the stopping criterion threshold has been set to the value 10^{-5} which is much smaller than what is usually required for the RCS calculations. The convergence of GMRES(30) and of full GMRES are also shown for comparison purposes. The objective of GMRES–DR is to approach the behaviour of the full GMRES method at a lower memory cost; we recall that this goal was achieved in the numerical experiments reported in Section 2.4.
In Figure 3.20, the behaviour of GMRES–DR(30,20) is satisfactory. It manages to follow well the superlinear convergence rate of full GMRES and outperforms GMRES(30). For the sake of comparison, we also plot the convergence history of GMRES(10) with a spectral low rank update preconditioner of size 20 and the convergence history of GMRES–DR(20,10) with a spectral low rank update preconditioner of size 10. These two solvers share a feature with GMRES–DR(30,20): all three attempt to capture the spectral information related to the 20 smallest eigenvalues using 20 vectors and use 10 vectors for the Arnoldi basis. GMRES(10) exploits spectral information from the beginning while GMRES–DR(30,20) does not have any spectral information at the beginning but constructs it during the iterations. In that respect, the comparison in term of iteration count is not fair. However, it is interesting to note that, at the end, the three convergence curves have the same slope. GMRES–DR(30,20) starts to exhibit this slope at the 150-th iteration while GMRES(10) with a spectral low rank update of size 20 has this slope from the beginning. It indicates that GMRES–DR(30,20) manages to construct a good enough approximation of the harmonic Ritz vectors to improve its speed of convergence notably.
For the same example, we have recovered the harmonic Ritz values computed by GMRES–DR(30,20) at each restart (i.e. every other 10 iterations). In Figure 3.21, we plot their modulus as a function of the iterations; we also plot the modulus of the 10 smallest eigenvalues of the matrix. These latter eigenvalues are computed using ARPACK. As the 10 smallest harmonic Ritz values converge toward the 10 smallest eigenvalues, convergence is also observed for their moduli as can be seen in the figure. Moreover, we point out the fact that the four smallest eigenvalues of the matrix are well approximated by the four smallest harmonic Ritz values after the 150–th iteration; that is, when GMRES–DR(30,20) starts to exhibit its superlinear rate of convergence. Instead of observing the harmonic Ritz value, we could also have looked at the Rayleigh quotient associated with y, that is $\frac{y^T \mathbb{Z} y}{y^T y}$. Indeed they are slightly better approximations of the eigenvalues than the harmonic Ritz values. In Section 3.8.1, this topic is also further discussed.
To conclude with these experiments we plot, in the complex plane, the 20 smallest eigenvalues and the 20 harmonic Ritz values given by GMRES–DR(30,20) at the last iteration. It can be seen that the smallest harmonic Ritz values match the corresponding eigenvalues. We also draw in the complex plane the path of the smallest harmonic Ritz value obtained at each restart. Note that this path goes from one eigenvalue to another. This observation is quite general and has been observed on all the examples.

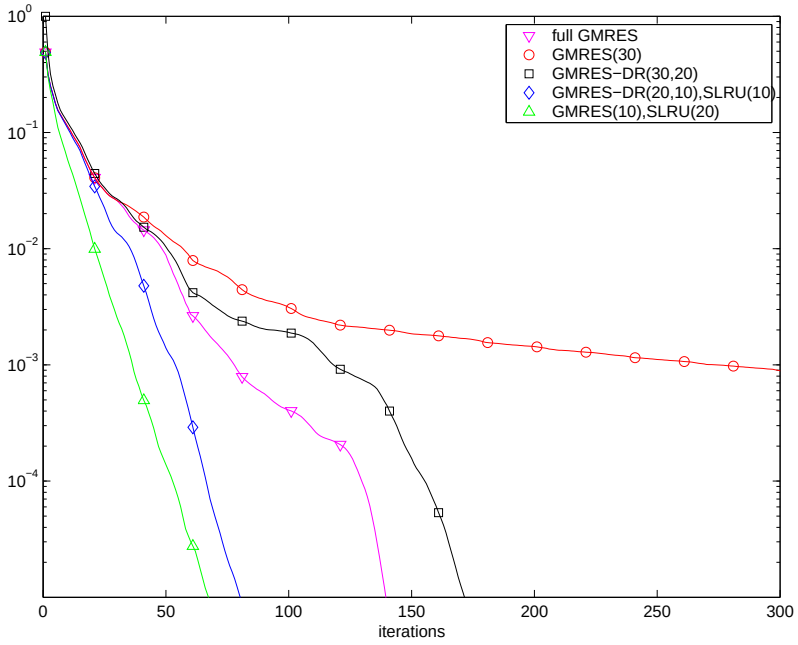


Figure 3.20: Convergence history for three solvers on the cetaf 5391 test example. The solvers are GMRES–DR(30,20), GMRES–DR(20,10) with a spectral low rank update of size 10 and GMRES(10) with a spectral low rank update of size 20. They share the same size of Krylov subspace (10) and the same number of spectral vectors (20). The full GMRES and GMRES(30) are also given for comparison.

In Figure 3.23, we report on another experiment with GMRES–DR(50,20) on the cobra 60695. The convergence history is plotted and compared with full GMRES and GMRES(50). As expected GMRES–DR(50,20) performs better than GMRES(50) and worse than full GMRES. In this case, we observe that the curve of GMRES–DR(50,20) does not fit the curve of full GMRES and that the final rate of convergence is half the one observed with full GMRES. At the end of the iterations, we observe that the smallest eigenvalue approximates the smallest eigenvalue well, so we think that the relatively poor rate of convergence of GMRES–DR(50,20) is mainly due to the fact that 20 vectors is not enough to properly represent the spectral information needed to obtain the slope of full GMRES.

In Figures 3.21 and 3.22, we illustrate that the GMRES–DR method finds a good approximation of the eigenvalues. This is not surprising since, in a sense, GMRES–DR is nothing but an eigensolver adapted for solving linear systems.

Finally in Table 3.17, we give the number of iterations of the GMRES–DR method, with full GMRES and restarted GMRES on the spheres and Airbus problems. The restarted GMRES and GMRES–DR use the same amount of memory to store their vectors. For GMRES–DR, we consider two different dimensions of the eigensubspaces, 10 and 20. In the case of the spheres, the GMRES–DR method succeeds in following the convergence curve of full GMRES. For the sphere with one million unknowns, the GMRES–DR(30,10) converges while the full GMRES method fails due to a lack of memory. In that respect the GMRES–DR(30,10) method and FGMRES

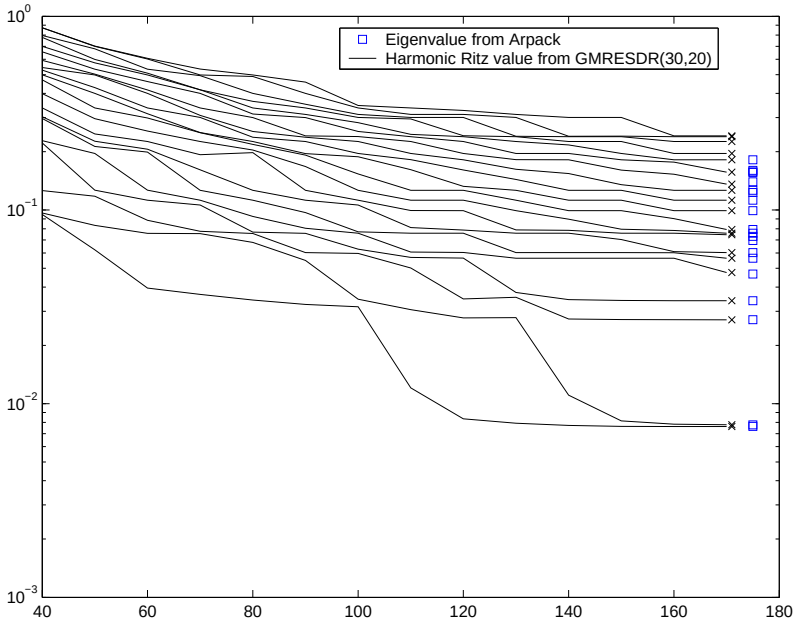


Figure 3.21: GMRES–DR(30,20) is run on the cetaf 5391 test example (see Figure 3.20). The modulus of the ten smallest harmonic Ritz values computed by GMRES–DR(30,20) during the iterations is plotted. We observe that they converged toward the modulus of the ten smallest eigenvalues that are also plotted.

(20,5)(2) are the only two methods that succeed in solving this very large problem. For the Airbus problems, the results are not as good, and the GMRES–DR solvers do not manage to follow the convergence curves of full GMRES exactly. The results are nevertheless satisfactory. We note that, for the Airbus 591900, GMRES–DR(30,10) and GMRES–DR(30,20) are close to convergence at the 500–th iteration whereas GMRES(30) stagnates at $2 \cdot 10^{-2}$ from the 300–th iteration.

3.4.2 The SQMR solver

In this section, we investigate the use of symmetric QMR [57] for the solution of symmetric dense linear systems arising from the EFIE formulation. Symmetric QMR (SQMR) is a hybrid version of QMR that benefits from the symmetry of the matrix so halving the memory and computing requirement compared with QMR. The advantage over solvers like GMRES is that SQMR uses a short term recurrence and therefore requires only a few vectors to be stored while the number of dot products is also considerably reduced. The main drawback is an observed delay in the convergence due, in general, to a loss of *orthogonality* among the computed vectors. In our experiments, the matrix-vector products are performed using a fast multipole code [130] with three different accuracies. Even though, in exact arithmetic, the dense matrix is symmetric, the use of floating–point arithmetic combined with the approximations made in the three implementations of the fast multipole method deteriorate this property. We therefore end up by using a non–symmetric matrix–vector product in a symmetric solver. In this section, we study the influence of this

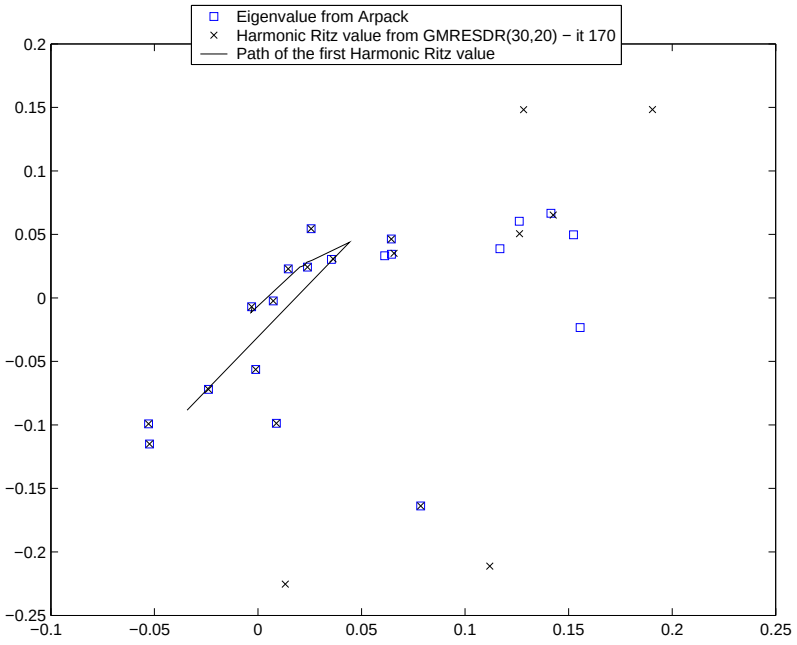


Figure 3.22: GMRES–DR(30,20) is run on the cetaf 5391 test example (see Figure 3.20). The twenty harmonic Ritz values obtained at the convergence from GMRES–DR(30,20) (iteration 170) with the twenty smallest eigenvalues of the matrix are plotted in the complex plane. The path during the iterations of the smallest harmonic Ritz value is also given.

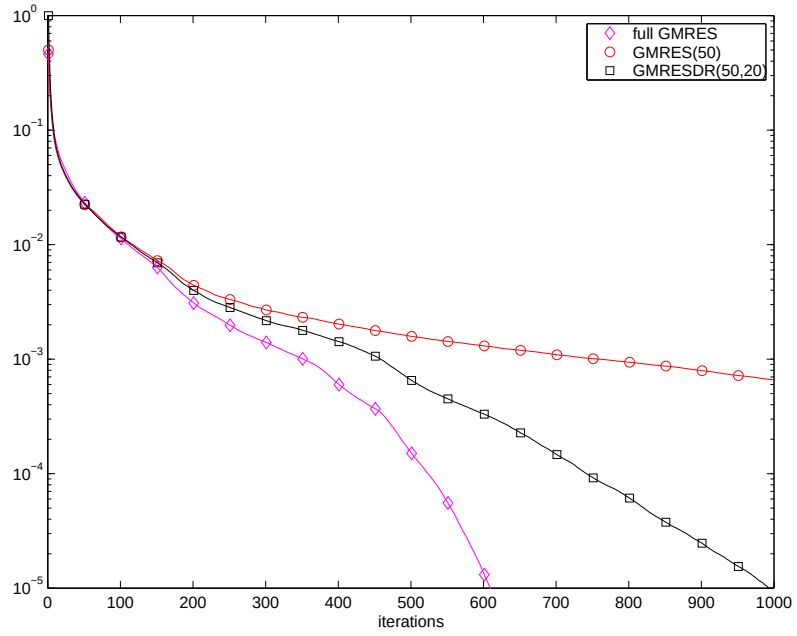


Figure 3.23: Convergence history for three solvers on the cobra 60695 test example. The solver GMRES–DR(50,20) is compared with full GMRES and GMRES(30).

			# iter
Airbus	23676	GMRES(30)	112
Airbus	23676	GMRES–DR(30,10)	95
Airbus	23676	GMRES–DR(30,20)	93
Airbus	23676	FULL GMRES	87
Airbus	94704	GMRES(30)	×
Airbus	94704	GMRES–DR(30,10)	170
Airbus	94704	GMRES–DR(30,20)	171
Airbus	94704	FULL GMRES	142
Airbus	213084	GMRES(30)	×
Airbus	213084	GMRES–DR(30,10)	266
Airbus	213084	GMRES–DR(30,20)	274
Airbus	213084	FULL GMRES	183
Airbus	591900	GMRES(30)	×
Airbus	591900	GMRES–DR(30,10)	×
Airbus	591900	GMRES–DR(30,20)	×
Airbus	591900	FULL GMRES	233
sphere	40368	GMRES(30)	138
sphere	40368	GMRES–DR(30,10)	81
sphere	40368	GMRES–DR(30,20)	80
sphere	40368	FULL GMRES	71
sphere	71148	GMRES(30)	87
sphere	71148	GMRES–DR(30,10)	84
sphere	71148	GMRES–DR(30,20)	80
sphere	71148	FULL GMRES	75
sphere	161472	GMRES(30)	139
sphere	161472	GMRES–DR(30,10)	104
sphere	161472	GMRES–DR(30,20)	98
sphere	161472	FULL GMRES	93
sphere	288300	GMRES(30)	335
sphere	288300	GMRES–DR(30,10)	187
sphere	288300	GMRES–DR(30,20)	166
sphere	288300	FULL GMRES	137
sphere	549552	GMRES(30)	475
sphere	549552	GMRES–DR(30,10)	198
sphere	549552	GMRES–DR(30,20)	209
sphere	549552	FULL GMRES	182
sphere	1023168	GMRES(30)	×
sphere	1023168	GMRES–DR(30,10)	197
sphere	1023168	FULL GMRES	–

Table 3.17: Comparison in term of iterations of GMRES–DR(30,10) and GMRES–DR(30,20) with full GMRES and GMRES(30). × means that the number of iterations exceeded 500, and – means that the code runs out-of disk space.

lack of symmetry on the behaviour of the linear solver.

3.4.2.1 A comparison study of solvers in the IE2M code.

We first report on experiments run with the IE2M code on the CNSPH test example (See Section 3.2 for their description). The main reason we use this problem is that

the corresponding matrix is small and explicitly available. That makes it possible to compute the level of symmetry for the matrix $\mathbb{Z}$, defined by

$$\frac{\|\mathbb{Z} - \mathbb{Z}^T\|_2}{2\|\mathbb{Z}\|_2}. \tag{3.23}$$

In equation (3.4.2.1), we use the 2–norm to represent the level of symmetry of the matrix $\mathbb{Z}$.From a mathematical point of view (see [74]), this quantity represents the distance in 2–norm from $\mathbb{Z}$ to the nearest symmetric matrix (that is $(\mathbb{Z} + \mathbb{Z}^T)/2$). We recall that the calculation of $\mathbb{Z}$ is performed in single precision. For this example, the code computes a matrix that has a level of symmetry equal to:

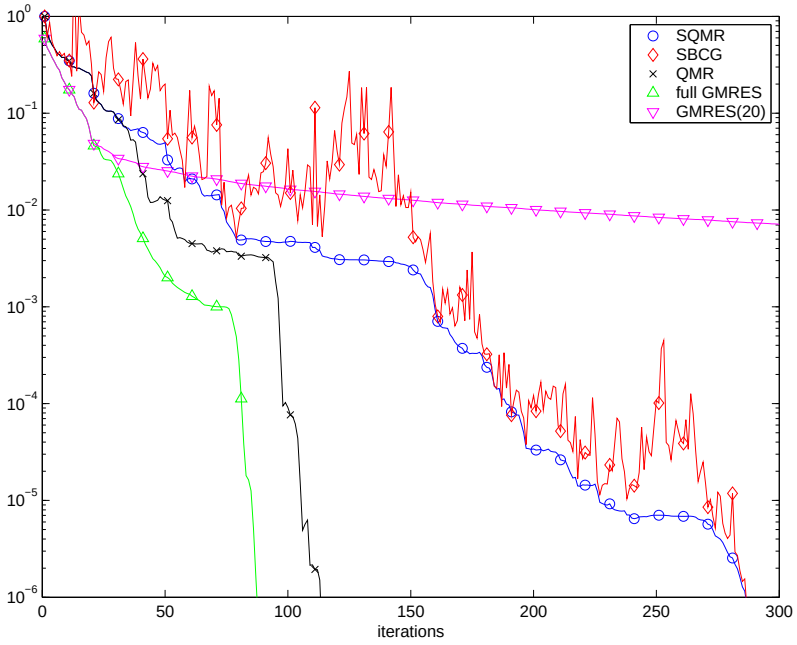
$$\frac{\|\mathbb{Z} - \mathbb{Z}^T\|_2}{2\|\mathbb{Z}\|_2} = 4 \cdot 10^{-5}.$$

This nonzero value is due to the fact that the code computes all the entries of the matrix without exploiting its symmetry; each entry z_{ij} is computed via a numerical integration, the round-off makes it different from z_{ji}. We recall that, in our experiments, this matrix is used in double precision arithmetic.
The first experiments consist in comparing the numerical behaviour of the solvers using both $\mathbb{Z}$, symmetric up to $4 \cdot 10^{-5}$, and $(\mathbb{Z} + \mathbb{Z}^T)/2$, that is symmetric by construction. As symmetric solvers we consider:

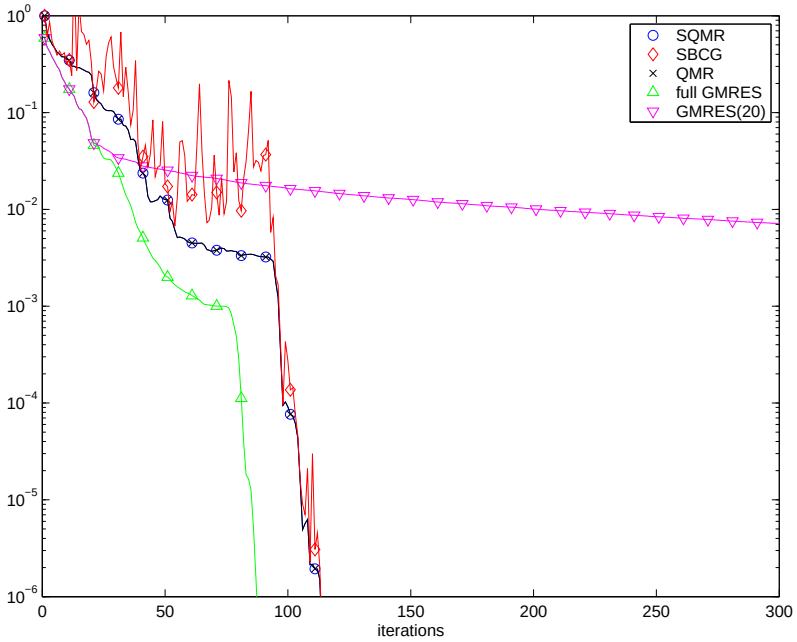
(a) Symmetric QMR (SQMR) with two three–term recurrences [54]. We have also played with the three two–term recurrence variants [56], but have observed the same behaviour and so do not report on these here.

(b) Biconjugate Gradient for a symmetric matrix (SBCG) [78, 80].

The convergence histories are displayed in Figure 3.24. For the purpose of comparison, we also plot the convergence of the non–symmetric solvers (c) QMR, (d) full GMRES and (e) GMRES(20). Theoretically the convergence histories of SQMR and QMR should perfectly overlap. It can be observed that it is certainly false when the solvers are applied to $\mathbb{Z}$ that is not symmetric (See Figure 3.24(a)). When the matrix is symmetrized, the SQMR curve follows exactly the QMR one. SQMR on $\mathbb{Z}$ needs 160 iterations to converge down to 10^{-3} whereas SQMR on $(\mathbb{Z} + \mathbb{Z}^T)/2$ only needs 97 iterations. We recall that although the level of symmetry of $\mathbb{Z}$ was not too far from the machine precision (the matrix was computed in single precision arithmetic), there is an already significant bad effect on the behaviour SQMR. We remark that SBCG oscillates a lot but eventually follows the curves of SQMR. This can be observed both with $\mathbb{Z}$ and with $(\mathbb{Z} + \mathbb{Z}^T)/2$. Note that if we run SQMR in single precision arithmetic on the explicitly symmetized matrix $(\mathbb{Z} + \mathbb{Z}^T)/2$, we would obtain a curve similar to that of SQMR in Figure 3.24(a). Finally we remark that we stop the iterations in the graphs of Figure 3.24 when the stopping criterion threshold 10^{-6} is verified by the backward error of the appoximate solution. The level of symmetry of $\mathbb{Z}$, $4 \cdot 10^{-5}$, corresponds to the nonsymmetric part of the perturbation that has deteriorated the system. Consequently, in practice, we are not interested to go lower than this value of the perturbation. However, if we continue after 10^{-6}, we note that all the curves of Figure 3.24, except that of GMRES(20),



(a) Experiments with $\mathbb{Z}$



(b) Experiments with $(\mathbb{Z}+\mathbb{Z}^T)/2$

Figure 3.24: Symmetric solvers (SBiCG, SQMR) are run on (a) $\mathbb{Z}$ for which the level of symmetry is $4{\cdot}10^{-5}$ and (b) $(\mathbb{Z}+\mathbb{Z}^T)/2$ for which is symmetric in floating–point arithmetic. For these solvers, one iteration requires one matrix–vector product with $\mathbb{Z}$. For the QMR solver, one matrix–vector product with $\mathbb{Z}$ and one matrix–vector product with $\mathbb{Z}^T$ are required per iteration.

reach the machine precision level. In particular, in our experiments, SQMR manages to obtain an approximate solution for $\mathbb{Z}J = F$ with a backward error of the order of the machine precision level; this happens even if the level of symmetry for $\mathbb{Z}$ is

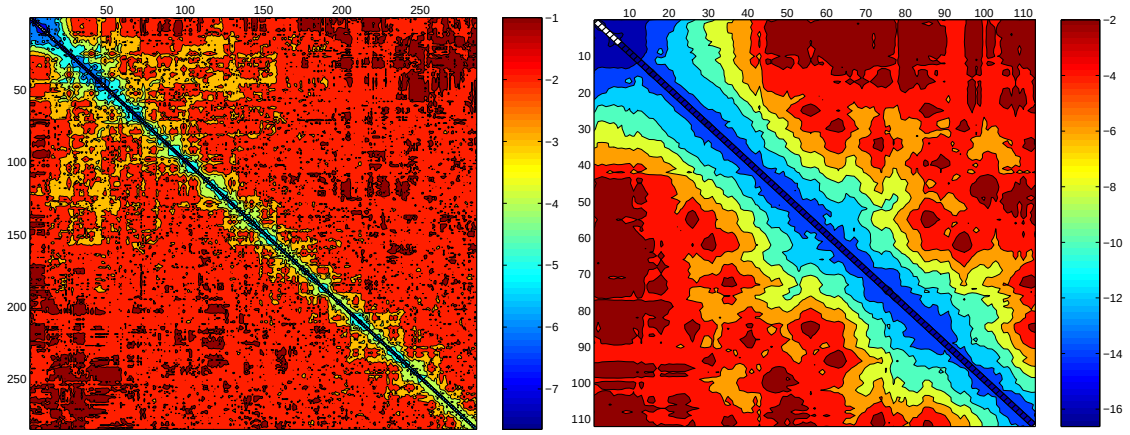
$4 \cdot 10^{-5}$.

3.4.2.2 On the loss of *t-orthogonality* due to the nonsymmetry

When the matrix is explicitly available the remedy is straighforward. It is enough to explicitly symmetrize the matrix, that is, to perform:

$$\text{for all } i \text{ and for all } j, \quad \mathbb{Z}_{ji} = (\mathbb{Z}_{ji} + \mathbb{Z}_{ij})/2 \text{ and } \mathbb{Z}_{ij} = \mathbb{Z}_{ji}.$$

In the case of AS_ELFIP, things are a bit more complicated. The matrix is not known explicitly and explicit symmetrization is no longer possible. One solution is to symmetrize the matrix implicitly. That is, each time we perform a matrix-vetor product, we enforce the relations that would hold if the matrix was symmetric. The relation imposed by the symmetry of $\mathbb{Z}$ is the *t-orthogonality* of the basis of the Krylov subspace constructed by SQMR. The *t-orthogonality* is the orthogonality defined with respect to the indefinite quadratic form associated with the tranpose (i.e. $x^T y = 0$). The matrix being implicitly symmetrized, is equivalent to enforcing the *t-orthogonality* among the Krylov basis in the SQMR method, eventually leading to a SQMR variant with full *t-reorthogonalization* (note that it requires storing all the Krylov basis as in GMRES).
It appears that there is a strong link between the level of symmetry of the matrix and the *t-orthogonality* obtained among the computed Krylov vectors. In Figure 3.25, we plot the loss of *t-orthogonality* among the 110 –first Krylov vectors on the CNSPH test example. In that figure, we plot the magnitudes of the entries of the matrix $I - Q^T Q$. We observe that larger the asymmetry of the matrix, the more the *t-*



(a) Experiments with $\mathbb{Z}$ (b) Experiments with $(\mathbb{Z} + \mathbb{Z}^T)/2$

Figure 3.25: Loss of *t-orthogonality* among the Krylov vectors in SQMR.

orthogonality is lost. Moreover, we observe that the loss of *t-orthogonality* among the Krylov vectors for SQMR applied to the explicitly symmetrized matrix can also grow up to 10^{-2} which is rather large. So, even with the explicitly symmetrized matrix, we can lose *t-orthogonality* (as the conjugate gradient algorithm might lose orthogonlity among the residuals).
This fact is not surprising and is in agreement with what we observe in Section 3.4.2.1. Since the matrix is known to the Krylov process only through a matrix–vector prod-

uct, even if it is exactly symmetric in floating–point arithmetic, we can consider that the matrix might have a loss of symmetry up-to the order of the machine precision. In Figure 3.24, we observe that a level of symmetry of $4 \cdot 10^{-5}$ results in an important delay in the convergence, even when the stopping criterion threshold is only set to 10^{-3}. Consequently, we can legitimally worry about the delay of convergence implied by a level of symmetry of the order of the machine precison. In Figure 3.26, we plot the curves of three solvers applied to $\mathbb{Z}$: (a) we run SQMR on the explicitly symmetrized matrix $\mathbb{Z}$, that is $(\mathbb{Z}+\mathbb{Z}^T)/2$; (b) we run SQMR on the implicitly and explicitly symmetrized matrix; that is, SQMR with *t-reorthogonalization* applied to $(\mathbb{Z}+\mathbb{Z}^T)/2$; (c) we run full GMRES on the explicitly symmetrized matrix $\mathbb{Z}$. In Fig-

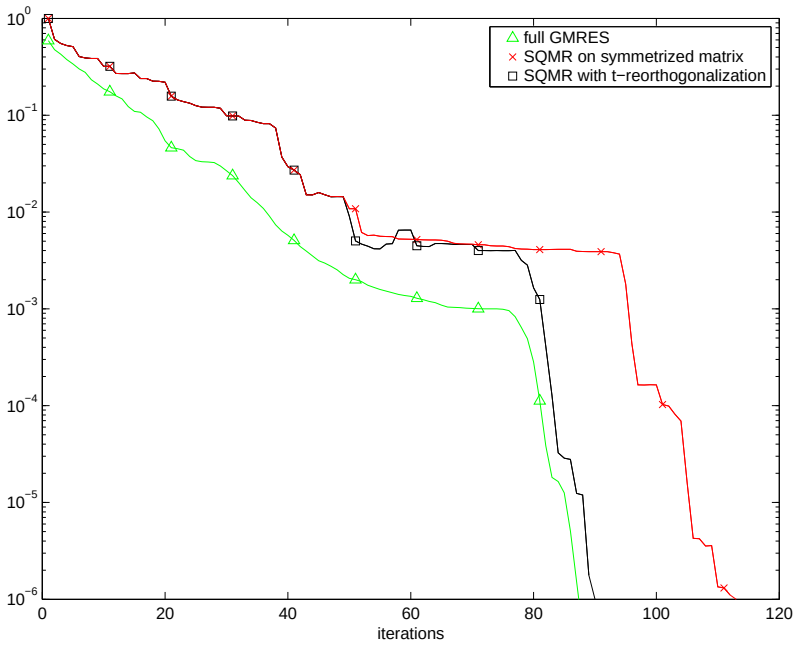


Figure 3.26: Comparison of the effect of two symmetrization strategies on the numerical behaviour of SQMR.

ure 3.26, SQMR with *t-reorthogonalization* clearly outperforms SQMR. The method benefits from the implicit symmetrization and almost behaves as GMRES.
In the context of the conjugate gradient algorithm, and more generally in the context of short term recurrence algorithms, such a phenomenon is well known and has led to extensive use and study of reorthogonalization algorithms. In Figure 3.27, we attempt to illustrate that our claims are also true for the conjugate gradient algorithm. We run the conjugate gradient algorithm on $\mathbb{Z}$ where $\mathbb{Z}$ is a 100–by–100 matrix. $\mathbb{Z}$ is the sum of a random Hermitian matrix $\mathbb{H}$ with eigenvalues logarithmically distributed between 1 and 10^{-3} and a random (nonsymmetric) perturbation $\mathbb{E}$ with $\|\mathbb{E}\|_2 = 10^{-7}$. The right–hand side b is random. Five solvers are tested. The backward error analysis is given with respect to $\mathbb{H}$ (i.e. not $\mathbb{Z} = \mathbb{H} + \mathbb{E}$) since the system we intend to solve is $\mathbb{H}x = b$. A direct solution using a LU factorization is performed to get a reference. We run the conjugate gradient algorithm on $\mathbb{Z}$. If the conjugate gradient algorithm is run on $(\mathbb{Z} + \mathbb{Z}^H)/2$ it performs better than if it

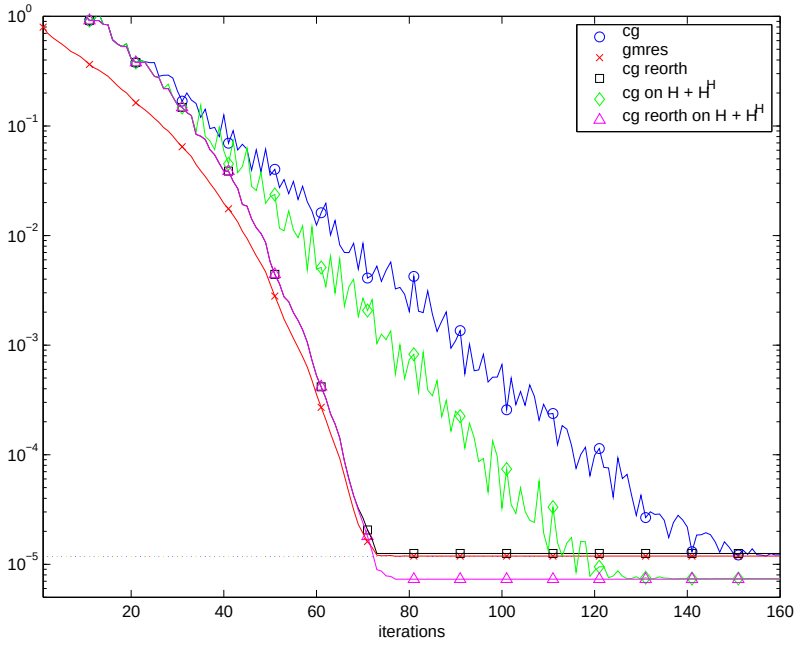


Figure 3.27: Convergence curves for the hermitian linear system $\mathbb{H}x = b$ of order 100 where $\mathbb{H}$ is a random Hermitian matrix with eigenvalues logarithmically distributed between 1 and 10^{-3}. The matrix-vector product is perturbed by a (nonsymmetric) perturbation $\mathbb{E}$ such that $\|\mathbb{E}\| = 10^{-7}$. The dotted line represents the backward error obtained with respect to $\mathbb{H}$ with a direct solver applied to $\mathbb{H} + \mathbb{E}$.

is performed on $\mathbb{Z}$ but these two strategies are outperformed by the conjugate gradient algorithm with reorthogonalization that nearly coincides with GMRES. We also note that the conjugate gradient algorithm with reorthogonalization on $(\mathbb{Z}+\mathbb{Z}^H)/2$ behaves exactly the same as the conjugate gradient algorithm on $\mathbb{Z}$ (this matrix has a level of symmetry of 10^{-7}) with also reorthogonalization. Finally, the same set of experiments have been performed with the QMR algorithm (also a short term recurrence algorithm) where we perturbed the matrix–vector products with $\mathbb{Z}$ and the matrix–vector products with $\mathbb{Z}^H$ by a random perturbation that changes at each product. The curves are similar to those of Figure 3.27 and the same conclusions can be drawn.

Another important remark that can be drawn from Figure 3.27 is that, even if the convergence of the conjugate gradient algorithm is severely damaged when the matrix $\mathbb{Z}$ is used, the algorithm still manages to provide a solution that is correct. This confirms the theoretical and experimental results given in [124, 132]. Indeed they even show that if the norm of the perturbation is increased proportionally to $1/\|r\|_2$ we should also reach this final accuracy level. However, their study does not predict any consequence on the convergence rate. Our experiments tend to show that if the perturbations are non–symmetric in a symmetric context then the convergence may be dramatically slower. This is also in agreement with all the work done in the past on various schemes of reorthogonalization to maintain the rate of convergence of the short term recurrence algorithms.

3.4.2.3 The AS_ELFIP code and the cetaf test example.

In order to have a symmetric preconditioner for SQMR, we take the Frobenius preconditioner, M, and use $M + M^T$ as preconditioner. Carpentieri, Duff, Giraud and Magolu monga Made [25] show that this strategy gives a suitable preconditioner. In Table 3.18, we give a simple illustration of this. We observe that full GMRES with the symmetrized preconditioner behaves the same as GMRES with the original preconditioner.

	# iter (10^{-3})		# iter (10^{-5})	
	Frob	symm–Frob	Frob	symm–Frob
without FMM	78	81	140	139
FMM *prec-2*	81	83	141	140

Table 3.18: Number of full GMRES iterations on the cetaf with either the Frobenius preconditioner or the symmetrized Frobenius preconditioner; two different values are considered for the stopping criterion threshold: 10^{-3} and 10^{-5}.

We first consider the cetaf. In Figure 3.28, we plot the backward error as a function of the iterations for the three accuracies of the FMM (denoted by *prec–1*, *prec–2* and *prec–3*) and two different arithmetics (i.e. single or double precision). Three

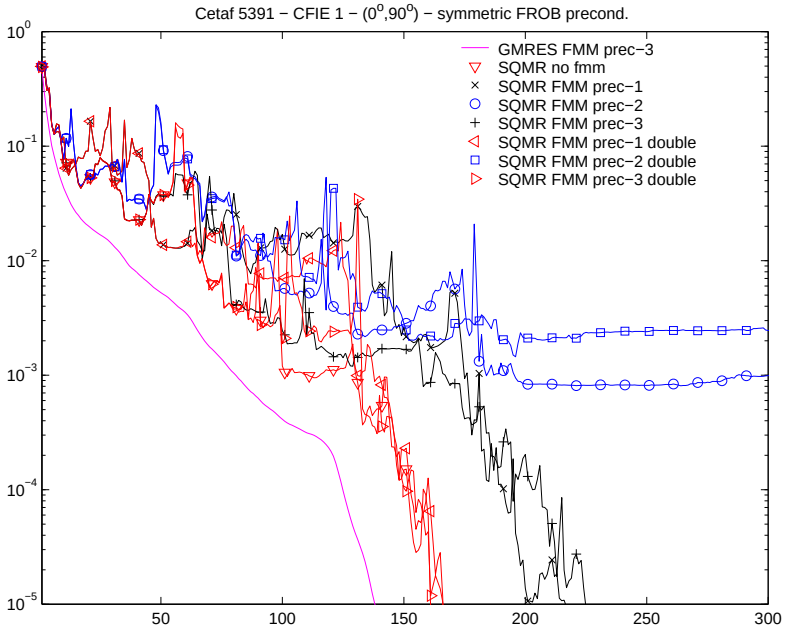


Figure 3.28: SQMR on the cetaf 5391 with different FMM implementation.

similar behaviours can be observed. The first is the non convergence of SQMR with the FMM *prec–2*. We checked the level of symmetry associated with this accuracy and observed that is was fairly bad. This explains the lack of convergence. The second is the similar behaviour oberved with both the FMM *prec–1* and *prec–3* when computed in single precision arithmetic; the corresponding matrices are symmetric

up to 10^{-6}. Finally, the last corresponds to the behaviour observed with the FMM *prec–1* and *prec–3* computed in double precision arithmetic. In that latter situation, the level of symmetry of these matrices is close to 10^{-15}. These experiments confirm that the rate of convergence of SQMR is greatly affected by the symmetry of the matrix involved in the construction of the Krylov vectors. The better the symmetry is, the faster the convergence. When the matrix is nearly symmetric, the behaviour of SQMR is fairly similar to that of GMRES. We have also compared SQMR with GMRES(30). The results are given in Table 3.19. It appears that SQMR, used

	# iter (10^{-3})	# iter (10^{-5})
Full GMRES	78	140
GMRES(30)	112	×
SQMR	133	167

Table 3.19: Number of iterations to obtain a backward error smaller than (10^{-3}) and (10^{-5}) for full GMRES, GMRES(30) and SQMR. The preconditioner used is the symmetrized Frobenius preconditioner for SQMR and the standard Frobenius preconditioner for GMRES. The multipole implementation uses *prec–3* in double precision arithmetic. × means that the stopping criterion is not satisfied in less than 500 iterations.

with the symmetric formulation of the multipole method, manages to converge to 10^{-5} whereas GMRES(30) fails. However, if a lower accuracy (10^{-3}) is requested, GMRES(30) performs better. GMRES(30) gets stuck between 10^{-3} and 10^{-5} and the convergence does not make any significant progress. More details on this work can also be found in [45].

3.4.2.4 Experimental study of SQMR

In this section, we investigate the numerical behaviour of SQMR on large electromagnetism problems, using the FMM *prec–3* in double precision that appears to be the most reliable. For that purpose we consider the Airbus and spheres set of test problems. The numerical experiments are reported in Table 3.20. It appears

		full GMRES	GMRES(30)	SQMR
Airbus	23676	71	81	324
Airbus	94704	100	131	440
Airbus	213084	123	343	×
sphere	40368	61	80	94
sphere	71148	66	76	111
sphere	161472	77	126	210
sphere	288300	131	311	370
sphere	549552	154	345	383

Table 3.20: Comparison of the number of iterations (equal to the number of matrix–vector product) needed for full GMRES, GMRES(30) and SQMR to converge. The matrix–vector product is performed using the FMM *prec–3 (double)*. × means that the convergence was not achieved in 500 iterations.

that SQMR gives satisfactory convergence behaviour that are nevertheless not fully

convincing. In practice, we have observed that, when the stopping criteria is set to 10^{-2}, then GMRES(30) exhibits a faster convergence than SQMR. When the stopping criterion threshold is set to 10^{-5} (very low for classical RCS calculations) then sometimes GMRES(30) does not converge whereas SQMR manages to converge.

3.4.2.5 Conclusion

SQMR, which is very appropriate for problems where the matrix is fully assembled [25], may also be applied with the multipole method but it requires a careful implementation to ensure the symmetry of the multipole expansion. From the experimental results using the AS_ELFIP code, we see that, even if the maximum symmetry affordable is obtained for using the FMM, SQMR does not give satifactory results for large systems when a low tolerance is requested. To decrease the number of iterations, a strategy would be to reorthogonalize the vectors. In this case, SQMR loses its computational interest and becomes as costly as full GMRES. For this reason, we do not investigate the use of SQMR in the AS_ELFIP code further; even if the use of local reorthogonalization techniques deserve to be studied.

3.5 Techniques to improve one right–hand–side solvers for multiple right–hand–side problems

The problem we face is to solve the linear systems (3.2) not only for one right-hand side but with several that are given simultaneously. In a classical RCS calculation, the number of right-hand sides is typically 360 if we are interested in observing the scattered waves in each direction in the plane of interest. If the solution for one right hand side requires a day of computation, the complete RCS will require a complete year; this is not acceptable in a design process. The purpose of this section is to indicate two strategies to significantly reduce the solution time for each right-hand side using a classical Krylov solver. The first strategy consists in exploiting the underlying physical problem to use suitable initial guesses. The second approach consists in solving simultaneously and independently several linear systems to take advantage of a Level 3 BLAS like efficiency of the FMM calculation.

3.5.1 Interpolation method

The solution of the linear system is the current $\vec{J}$ on the object. It depends continously on φ (and/or θ), the angle associated with the illuminating wave. If we assume that the system is solved for a given right–hand side $F(\varphi_0)$, giving the solution $J(\varphi_0)$, a natural idea is to use $J(\varphi_0)$ as the initial guess for the solution of the next linear system $\mathbb{Z}J = F(\varphi_0 + \delta\varphi)$ associated with the next illuminating angle. This leads to a simple but effective strategy, referred to as *strategy 1*. Some other strategies have been derived to find a more elaborate initial guess (e.g. Carayol [22]). In particular, Sylvand [130] has investigated several. Among these, we only use the two that appear to be the most effective. The first one is *strategy 1* described above. The second one further exploits the nature of the underlying equations. The right-hand side is defined by equations (3.4) and (3.5), the ℓ-th entry of F is defined by

$$F_\ell(\varphi) = \int_\Gamma e^{ikx\cdot\hat{u}_r(\varphi)}\hat{z}\cdot\vec{\Psi}_\ell(x)ds(x). \tag{3.23}$$

The entry F_ℓ only depends on φ. In a first approximation, we can assume that the dependency in φ is linear with respect to $e^{ikx\cdot\hat{u}_r(\varphi)}$. When going from an angle φ_0 to $\varphi_0 + \delta\varphi$ a natural strategy for the initial guess is to use the solution $J(\varphi_0)$ corrected with a phase term, giving

$$J_\ell(\varphi_0)e^{ikx_\ell\cdot\hat{u}_r(\delta\varphi)}.$$

For the third right–hand side, each entry of the initial guess is computed as the linear interpolation of the corresponding entries in the first and second solution, each of them corrected by the appropriate phase component. For the following initial guesses, the same strategy is applied that only used the two previous solutions. This second strategy is referred to as *strategy 2 for the initial guess*. Note that this strategy leads to an initial guess that is not in the span of the previous solution. This nonlinear way of getting the initial guess gives therefore a completely different intial guess than those obtain with standard linear algebra techniques.

These two strategies are very efficient and their numerical merits are reported throughout this document.

3.5.2 Gathering multiple GMRES iterationns

In Table 3.6, we observe that, when the FMM is used to perform several matrix-vector products at a time, significant gains can be expected due to a Level 3 BLAS effect. Since in a GMRES solve, the FMM is the main time consuming part, solving at the same time and independently several right-hand sides is an appealing strategy if the FMM product are gathered. We refer to this strategy as gathered GMRES. In Table 3.21, we give results on the elapsed times for gathered GMRES and for a sequence of classical GMRES. In our implementation of gathered GMRES, we have indeed synchronized and gathered all the kernel operations required by the p GMRES solvers. ight–hand sides have converged. We mention that this latter constraint can be relaxed by deflating the right–hand sides as soon the corresponding solution has converged. For the sake of simplicity of the implementation we have not considered this strategy in the preliminary implementation used for the experiments reported in this document. Under these four assumptions, each of the p solver performs, at each step, the same operation between the same vectors of its own. This enables the code to use the data locality in cache and memory efficiently as the code is out-of-core.
In Table 3.21, the total number of iterations reported for gathered GMRES is higher than for the sequence of classical GMRES and is indeed a multiple of the number of gathered right–hand sides. This is a direct consequence of the fact that we do not deflate a converged vector. This problem does not lead to important damage in the method since the convergence is rather uniform among the right–hand sides. On the coated cone sphere test example, the gap between gathered GMRES and a

	FMM (s)	Precond (s)	# FMM &Precond	total
gathered GMRES (10)	0.7	0.05	180	205.1
GMRES with zero as initial guess	1.8	0.27	177	401.4

(a) Airbus 23676.

	FMM (s)	Precond (s)	# FMM &Precond	total
GMRES gathered (19)	1.6	0.13	931	2340.8
GMRES with strategy 2 for the initial guess	7.1	0.42	856	7703.2

(b) coated cone sphere.

Table 3.21: Comparison in elapsed time (s) of gathered GMRES and a sequence of classical GMRES on two test examples. Gathered GMRES gathers the matrix–vector product (and precondition) by sets of size 19 for the coated cone sphere test example and sets of size 10 for the Airbus 23676 test example. The total elapsed time (s) for the solution is given in the last column.

sequence of classical GMRES is larger than for the other example. This is due to

the fact, that in the dielectric situation, the FMM performed more floating–point operations and exploiting the data locality induces larger gains.
Finally, we note that strategy 2 for the initial guess cannot be used within a set of right–hand sides, but could be implemented between the sets. That is, for instance for the RCS from $\theta = 0^o : 1^o : 179^o$ of the coated cone sphere, the first set will include the angles $0^o : 10^o : 170^o$, the next $1^o : 10^o : 171^o$, etc ... This strategy would deserve to be implemented in a future release of the code.

3.6 Linear dependency of the right–hand sides

A natural question to address when solving a linear system with p right–hand sides is whether these right–hand sides are linearly independent or not. If the right–hand sides are linearly dependent with $\text{rank}([F_1, \ldots, F_p]) = q < p$, there exists U, an n–by–q matrix, and S, a q–by–p matrix such that

$$F = US,$$

where $F = [F_1, \ldots, F_p]$. In such a case, a natural approach consists in solving the q systems associated with the right–hand sides U, that is

$$\mathbb{Z} J_U = U,$$

then recovering the unknowns of interest

$$J = J_U S.$$

This section is organized as follows. In the first part, we further investigate the nature of the right–hand sides arising in electromagnetism calculations. We show that any of these right-hand sides can be well approximated in a space spanned by q_{sh} spherical harmonic functions. In practical RCS calculations, the engineers usually provide a number of right-hand sides that is far larger than q_{sh}. In the second subsection, we compare the analytic value q_{sh} with the numerically computed value q. Finally, in a third part we illustrate the benefits of this approach and how it can be efficiently exploited on some practical application examples.

3.6.1 Features of the right–hand sides for plane waves with θ polarization

If we go back to the expression for the right–hand side, equation (3.4) and equation (3.6) give us

$$F_j(\varphi) = \int_\Gamma e^{ikx \cdot \hat{u}_r(\varphi)} \hat{z} \cdot \vec{\Psi}_j(x) ds(x). \tag{3.23}$$

Since F is only a function of φ, in equation (3.6.1) and in the remainder of this document we denote it by $F(\varphi)$, $F_j(\varphi)$ denotes its j–th entry and the associated solution of $\mathbb{Z}J = F(\varphi)$ is denoted by $J(\varphi)$. In this section, we recall some mathematical results on the spherical harmonics. In particular, we show that, for each $\varphi \in [0, 2\pi]$, $F(\varphi)$ can be expressed with a small error as a linear combination of the same finite set of vectors.

3.6.1.1 Use of Jacobi–Anger formula

3.6.1.1.1 Spherical harmonics If $\hat{s}$ is a direction on the unit sphere, we can associate it with the angles (θ_s, φ_s) so that

$$\hat{s} = \begin{pmatrix} \sin\theta_s \cos\varphi_s \\ \sin\theta_s \sin\varphi_s \\ \cos\theta_s \end{pmatrix}.$$

Let ℓ and n be two integers such that $|\ell| \leq n$ and $n \geq 0$. We define the spherical harmonics by

$$Y_n^\ell(\hat{s}) = \sqrt{\frac{2n+1}{4\pi}} S_n^{|\ell|}(\cos\theta_s) e^{i\ell\varphi_s}, \tag{3.23}$$

where $S_n^\ell(x)$ are the spherical functions of Legendre. They are defined by induction. If ℓ is positive then

$$\begin{aligned}
&S_n^\ell(x) = 0, \quad \text{for } n = 0, \ldots, n-1,\\
&S_\ell^\ell(x) = \frac{\sqrt{(2\ell)!}}{2^\ell \cdot \ell!}(1-x^2)^{\frac{\ell}{2}},\\
&S_n^\ell(x) = \left[(2n-1)xS_{n-1}^\ell(x) - ((n-1)^2 - \ell^2)^{\frac{1}{2}} S_{n-2}^\ell(x)\right] / \sqrt{n^2-\ell^2} \quad \text{for } n = \ell+1, \ell+2, \ldots
\end{aligned}$$

3.6.1.1.2 Jacobi–Anger formula This formula enables us to decompose a plane wave into spherical harmonics. If $x = |x|\hat{x}$, we have

$$e^{ikx\cdot\hat{u}_r} = 4\pi \sum_{n=0}^{\infty} \sum_{\ell=-n}^{n} i^n j_n(k|x|) \overline{Y_n^\ell(\hat{x})} Y_n^\ell(\hat{u}_r),$$

where $j_n(t)$ is the spherical Bessel function of order n (see [33]). Note that this series converges uniformly on any ball $\mathcal{B}_\Gamma = \{|x|; |x| \leq r_\Gamma\}$. Moreover, if

$$L_\varepsilon = kr_\Gamma + C_\varepsilon \log(kr_\Gamma + \pi), \tag{3.20}$$

and the series is truncated by keeping the first L_ε terms, it can be shown that the error is of the order $10^{-C_\varepsilon} = \varepsilon$ [86]. Therefore we can write with an error of order ε:

$$e^{ikx\cdot\hat{u}_r} = 4\pi \sum_{n=0}^{L_\varepsilon} \sum_{\ell=-n}^{n} i^n j_n(k|x|) \overline{Y_n^\ell(\hat{x})} Y_n^\ell(\hat{u}_r).$$

In our case, the corresponding angles are $\theta_s = \frac{\pi}{2}, \varphi_s = \varphi$, and $\hat{u}_r = \begin{pmatrix} \cos\varphi \\ \sin\varphi \\ 0 \end{pmatrix}$.

Equation (3.6.1.1.1) provides

$$e^{ikx\cdot\hat{u}_r} = \sqrt{4\pi} \sum_{n=0}^{L_\varepsilon} \sum_{\ell=-n}^{n} i^n \sqrt{2n+1} j_n(k|x|) \overline{Y_n^\ell(\hat{x})} S_n^{|\ell|}(0) e^{i\ell\varphi},$$

that can also be written

$$e^{ikx\cdot\hat{u}_r(\varphi)} = \sum_{n=-L_\varepsilon}^{L_\varepsilon} \sqrt{4\pi} \sum_{n\geq|\ell|} i^n \sqrt{2n+1} j_n(k|x|) \overline{Y_n^\ell(\hat{x})} S_n^{|\ell|}(0) e^{i\ell\varphi},$$

and finally

$$e^{ikx\cdot\hat{u}_r} = \sum_{\ell=-L_\varepsilon}^{\ell=L_\varepsilon} P_\ell(x) e^{i\ell\varphi} \tag{3.20}$$

where

$$P_\ell(x) = \sqrt{4\pi} \sum_{n \geq |\ell|} i^n \sqrt{2n+1} j_n(k|x|) \overline{Y_n^\ell(\hat{x})} S_n^{|\ell|}(0).$$

Equation (3.6.1.1.2) is the Fourier decomposition of the function

$$\varphi \in [0, 2\pi] \longrightarrow e^{ikx \cdot \hat{u}_r(\varphi)}.$$

If we replace L_ε by ∞ in equation (3.6.1.1.2), we recover the exact decomposition. If an accuracy ε is acceptable then, the first L_ε terms are enough.

3.6.1.2 Results for the right–hand sides.

We insert expression (3.6.1.1.2) in equation (3.6.1) to get

$$F_n(\varphi) = \sum_{\ell=-L_\varepsilon}^{L_\varepsilon} \left[\int_\Gamma P_\ell(x) \hat{z} \cdot \vec{\Psi}_n(x) ds(x) \right] e^{i\ell\varphi}.$$

If we define the vector ξ^ℓ of size n_e with its n^{th} entry equal to

$$\xi^\ell(n) = \left[\int_\Gamma P_\ell(x) \hat{z} \cdot \vec{\Psi}_n(x) ds(x) \right] \quad \text{where } 1 \leq n \leq n_e, \tag{3.20}$$

we have

$$F(\varphi) = \sum_{\ell=-L_\varepsilon}^{L_\varepsilon} \xi^\ell e^{i\ell\varphi}. \tag{3.20}$$

The vectors $F(\varphi)$ belong to a space of size at most $2L_\varepsilon+1$. Using equation (3.6.1.1.2), the dimension of the space spanned by the right–hand sides is bounded above by

$$M = 2\left(kr_\Gamma + C_\varepsilon \log(kr_\Gamma + \pi)\right) + 1. \tag{3.20}$$

Note that in equation (3.6.1.2), the term kr_Γ can be replaced by πp, where we recall that p is the size, in number of wavelengths, of the object. M is only a function of p and in a first approximation, we have $M \sim 2\pi p$. We observe that the number of right–hand sides increases proportionally with the frequency.

3.6.2 Numerical validation

The numerical validation of the theory presented in Section 3.6.1 is as follows. In our experiments we first build the $F(\varphi_l)$, $l = 1, \ldots, p$. Next, we compute the Singular Value Decomposition (SVD) of $F(\varphi_l)_{l=1,\ldots,p}$ in order to obtain

$$[F(\varphi_l)_{l=1,\ldots,p}] = U \Sigma V^H,$$

where U is an m–by–p matrix with orthonormal columns (i.e. $U^H U = I_p$), V is a p–by–p matrix with orthonormal columns (i.e. $V^H V = I_p$) and Σ is diagonal with positive real entries on the diagonal ordered by decreasing value. The entry (i, i) of Σ is denoted by σ_i. We consider the truncated SVD defined by:

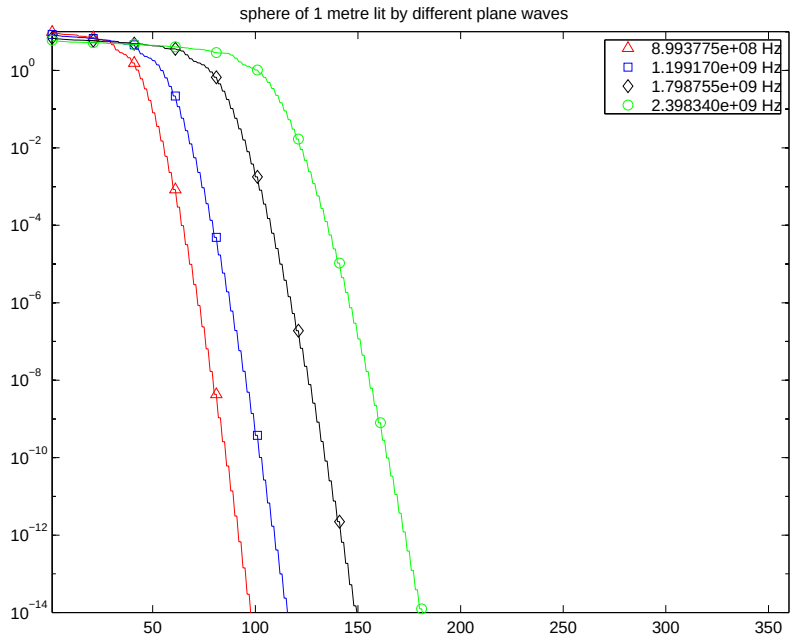
$$[F(\varphi_l)_{l=1,\ldots,p}] \approx U_q \Sigma_q V_q^H,$$

where $U_q = [(U_l)_{l=1,\ldots,q}]$, $V_q = [(V_l)_{l=1,\ldots,q}]$ and Σ_q is the q–by–q diagonal matrix with its entry (i,i) equal to σ_i.
The value of q is chosen such that $\sigma_q/\sigma_1 < \varepsilon$. Note that if it happens that $\sigma_p/\sigma_1 \geq \varepsilon$, we use a larger p. The basis U_q is not the set of vectors $(\xi^\ell)_{\ell=-L_\varepsilon,\ldots,L_\varepsilon}$ defined in equation (3.6.1.2), but it plays the same role; both span the space that contains the $(\varphi_l) l = 1,\ldots,p$.
In the remainder of this section we compare the computed number q with its theoretical counterpart q_{sh}. Because the number of columns in F is small compared with its number of rows, the algorithm of choice for computing the SVD of F is the R–SVD algorithm (see e.g. [63, pp. 152–254]), and we proceed as follows. First of all, we compute the QR factorization of F via modified Gram–Schmidt iterated twice to get $F = QR$, then we compute with LAPACK the SVD of R to get $R = U_R \Sigma W^H$. Finally, we compute $U = QU_R$, the left singular vectors. The SVD we seek is eventually given by $F = U\Sigma W^H$.

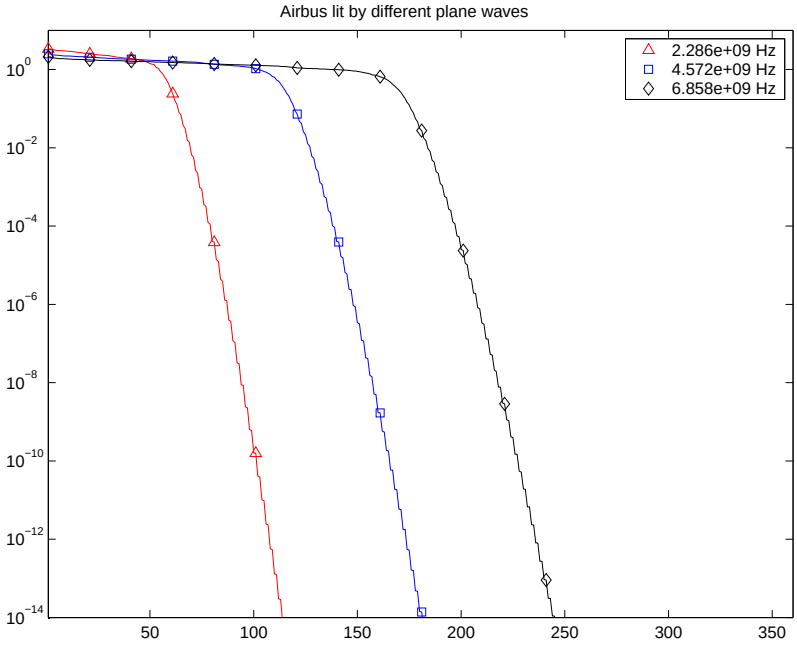
# dof	q	q_{sh}
40368	60	63
71148	76	78
161472	104	106
288300	130	133

Table 3.22: Comparison of q_{sh} computed with $C_\varepsilon = 4$ and q, the larger integer such that $\sigma_q/\sigma_1 < 10^{-4}$ for spheres with different numbers of degrees of freedom.

In Table 3.22, we compare the quantity q such that $\sigma_q/\sigma_1 < \varepsilon = 10^{-4}$ with the quantity q_{sh} obtained from equation (3.6.1.2). The parameter C_ε in equation (3.6.1.2) is set so that $\varepsilon = 10^{-4} = 10^{-C_\varepsilon}$; this gives $C_\varepsilon = 4$. This rule of thumb is given by the people working on electromagnetism applications [35]. For φ, the interval of angles is initially discretized at every degree from 0^o to 359^o. Therefore the parameter p is equal to 360. In Table 3.22, we observe that q_{sh} is close to q. Note that a perfect matching between q_{sh} and q is observed for the value $C_\varepsilon = 3.7$.
We should mention, that if we change the targeted accuracy for the right–hand sides from $\sigma_q/\sigma_1 < 10^{-4}$ to $\sigma_q/\sigma_1 < 10^{-6}$, the number of singular vectors q increases. Similarly, the corresponding value of C_ε increases but we still observe that q_{sh} and q remain close to each other. The model given in Section 3.6.1 and the assumption $\varepsilon \approx \sigma_q/\sigma_1$ seems therefore to enable us to get a rather sharp upper bound for q for any ε. Finally, we mention that q_{sh} increases with respect to the frequency. This seems to be in agreement with the fact that the RCS associated with high frequencies are highly oscillating. Classically, when the frequency is increased the engineers reduce the step used to discretize the interval of interest; this results in the solution of more linear systems. This behaviour, that is that the number of large singular values increases with respect to the frequency, is observed in Figure 3.29. In that figure, we plot the singular values larger than 10^{-14} for the sphere and the Airbus test example, when illuminated with waves of different frequencies.
It might be noticed that the value of q_{sh} only depends on the size of the object measured in wavelengths. The geometry of the object does not play any role on



(a) sphere



(b) Airbus

Figure 3.29: Largest singular values of F when the frequency of the illuminating waves is varied.

the value of q_{sh}. Finally, the above comparative study has been performed for a sample of plane waves that lie in the plane ($\theta = 0^o$). The gap between q and p, that is already significant, would be even larger if both φ and θ were varied. For that latter situation on the sphere 40368, we get $q = 549$ to be compared with

$p = 64800$ if every other degree is discretized in φ and θ.

3.6.3 Dealing with linearly dependent right–hand sides

Let us consider the system $\mathbb{Z}J = F$, where $F = [F_1, \ldots, F_p]$ and $J = [J_1, \ldots, J_p]$. Let tol_a denote the tolerance requested for the a-th right–hand side. We write the SVD of F in the following block form:

$$F = \begin{pmatrix} U & U_E \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & \Sigma_E \end{pmatrix} \begin{pmatrix} W^H \\ W_E^H \end{pmatrix},$$

where $\begin{pmatrix} U & U_E \end{pmatrix}$ is m–by–n and has orthonormal columns, U corresponds to the first q columns, U_E the last $n-q$; Σ is a q–by–q diagonal matrix with entries $\sigma_1, \ldots, \sigma_q$ on the diagonal, Σ_E is a $(n-q)$–by–$(n-q)$ diagonal matrix with entries $\sigma_{q+1}, \ldots \sigma_n$ on the diagonal, $\begin{pmatrix} W \\ W_E \end{pmatrix}$ is n–by–n and has orthonormal columns, W corresponds to the first q columns, W_E the last $n-q$.
If we note that $S = \Sigma W^H$ and $E = U_E \Sigma_E W_E^H$, we can write

$$F = US + E. \tag{3.20}$$

In this expression, U corresponds to the q linearly independent set of vectors chosen to represent F, S corresponds to the coefficients of F in U, and E the error made when approximating F by the product US. In that case we have

$$\|E\|_2 = \sigma_{q+1}.$$

What we would like to do at this stage is: to neglect the matrix E, to solve for the U (i.e. only solve q linear systems $\mathbb{Z}X_\ell = U_\ell$ using a backward error threshold equal to tol_ℓ^X), and to recover the solution to the system $\mathbb{Z}J = F$ by setting $J = XS$. For doing this, we need to know:

1. how to choose q,
2. how to choose the stopping criterion thresholds tol_ℓ^X, so that each reconstructed solution J_a, associated with the right–hand side F_a, has a backward error at most equal to tol_a.

To address these questions, let $R = U - \mathbb{Z}X$. Each column of R corresponds to the residual of the linear system associated with a singular vector. Consequently, R_ℓ satisfies the relation $\|R_\ell\|_2 \leq \mathrm{tol}_\ell^X \|U_\ell\|_2 = \mathrm{tol}_\ell^X$. From equation (3.6.3), the residuals $F - \mathbb{Z}J$ can also be written as

$$F - \mathbb{Z}J = RS + E, \tag{3.20}$$

which indicates that the final residuals, $F - \mathbb{Z}J$, are defined by E (part of the right–hand sides that we do not want to solve) and the residuals R multiplied by the matrix of the coefficients of F in U. Considering the a-th column of the

equality (3.6.3), we get

$$\begin{aligned}\frac{\|F_a - \mathbb{Z}J_a\|_2}{\|F_a\|_2} &= \frac{\|RS_a + E_a\|_2}{\|F_a\|_2} \\ &\leq \sum_{\ell=1}^{q}\left(\frac{|S_{\ell,a}|}{\|F_a\|_2}\|R_\ell\|_2\right) + \frac{\|E_a\|_2}{\|F_a\|_2} \\ &\leq \sum_{\ell=1}^{q}\left(\frac{|S_{\ell,a}|}{\|F_a\|_2}\mathrm{tol}_\ell^X\right) + \frac{\sigma_{q+1}}{\|F_a\|_2} \qquad (3.19)\end{aligned}$$

In order to ensure that, for any a, we have $\|F_a - \mathbb{Z}J_a\|_2/\|F_a\|_2 \leq \mathrm{tol}_a$. We should select q and tol_ℓ^X, $\ell = 1, \ldots, q$, such that

$$\sum_{\ell=1}^{q}\left(\frac{|S_{\ell,a}|}{\|F_a\|_2}\mathrm{tol}_\ell^X\right) + \frac{\sigma_{q+1}}{\|F_a\|_2} \leq \mathrm{tol}_a.$$

Among all the possibilities for (q, tol_ℓ^X), we select the one that has the smallest q such that

$$\sigma_{q+1} \leq \beta \min_a(\|F_a\|_2 \mathrm{tol}_a), \qquad (3.19)$$

where $\beta < 1$. We assume that $q < p$, then the stopping criterion for the q linear systems can be chosen such that

$$\mathrm{tol}_\ell^X = \alpha_\ell \min_a\left(\frac{\|F_a\|_2 \mathrm{tol}_a}{|S_{\ell,a}|}\right), \qquad (3.19)$$

where the α_ℓ's are such that

$$\beta + \sum_{\ell=1}^{q}\alpha_\ell < 1.$$

A natural choice for the α_ℓ's and β is $\alpha_\ell = \beta = (q+1)^{-1}$.

3.6.4 Heuristic for the choices of α and β

The parameters $\alpha_\ell = (q+1)^{-1}$ and $\beta = (q+1)^{-1}$ certainly ensure the prescribed backward error for each of the p systems $\mathbb{Z}J_a = F_a$. This choice takes into account the worst situation, when the triangle inequality is an equality; that it, the q columns of R and that of E_ℓ are colinear. This is possible but highly unlikely to happen. For large q, this gives fairly small values for α_ℓ and β. Such values request targeted accuracies that eventually lead to solutions that have much smaller backward errors that the ones prescribed. To overcome this drawback, we set in practice , $\alpha_\ell = \alpha = 0.5$ and $\beta = 0.7$; which enables us to compute RCS that are correct for all our test examples. Regarding the effect of this heuristic choice, we plot, in Figure 3.30, the backward error associated with each of the recovered solutions J_a for the cobra 14449 test example illuminated from 0^o to 90^o. On that example, only 21 linear systems have been effectively solved and 91 solutions are eventually reconstructed. The targeted accuracy was 10^{-3} for all the right–hand

sides. In that figure, it can be observed that the value $\alpha = 0.5$ does not manage to provide all backward errors at the targeted value, while the solution complies with the prescribed accuracies for many right–hand sides. It can also be observed that the value $\alpha = (q+1)^{-1} = 0.047$, referred to as the *secure strategy*, imposes a too strong constraint: most of the solutions are computed with a backward error around 10^{-4}.

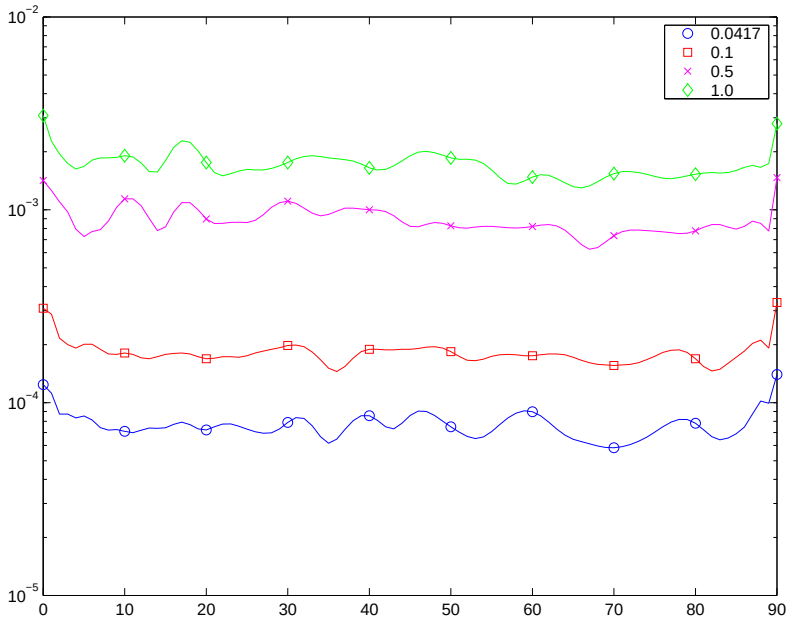


Figure 3.30: Backward error observed for different values of α in equation (3.6.3). For that example, $p = 90$ and $q = 21$. The value $\alpha = (q+1)^{-1} = 0.047$ corresponds to a secure strategy. The targeted backward error for all the right–hand sides is $\mathrm{tol}_a = 10^{-3}$, $a = 1, \ldots, p$. The other values correspond to relaxed strategies.

We further investigate the effect of the choices of α (β is constant and equal to 0.5) on the final observed accuracy of the reconstructed right–hand sides as well as on the computing cost required by the corresponding calculation. In Figure 3.31, we vary α and display the largest observed backward error associated with the reconstructed solutions and the cumulated number of iterations performed to solve the q linear systems. The largest value of α such that the targeted accuracy is observed for all the reconstructed right–hand sides is about 0.4; the corresponding calculation only requires 2100 cumulated iterations (to be compared with 2800 with the secure strategy). The optimal value of α is difficult to predict. We see in Section 3.6.7 that the problem simplifies if the linear solver is block-GMRES. Finally, note that a "bad" choice of α might lead to backward errors higher than the stopping criteria (e.g. $\alpha = 1$ in the investigated case). This latter situation can be easily tackled a posteriori with an iterative refinement type procedure. This would consist in performing a few GMRES steps using the recovered solutions as initial guesses for the right–hand sides that have not converged.

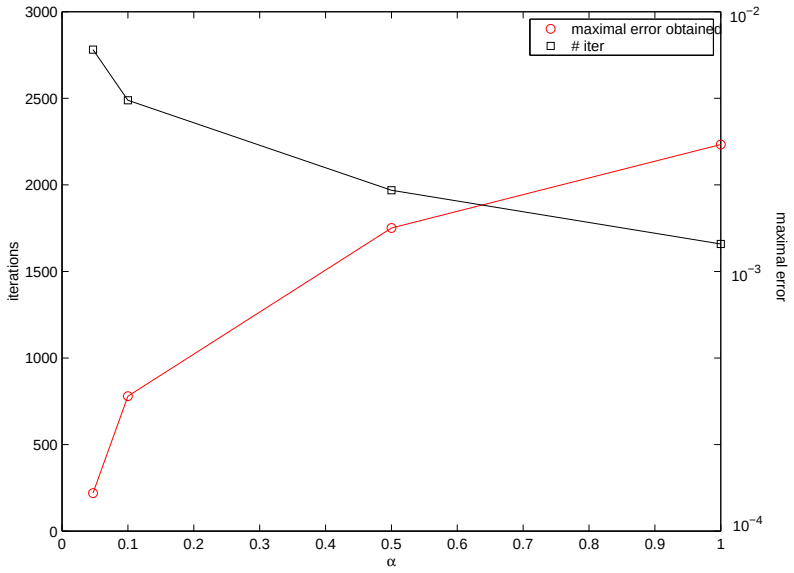


Figure 3.31: Largest backward error observed on the reconstructed solutions and computational cost expressed in cumulated number of iterations when α is varied.
The 21 linear systems are solved with seed–GMRES.

3.6.5 Relaxing the stopping criteria

In practical implementations, we use equation (3.6.3) to compute the stopping criterion thresholds, tol_ℓ^X, for the solution of the linear systems that have the singular vectors as right–hand sides. Since $S_{\ell,a} = \sigma_\ell (W_{\ell,a})^H$ and $\|W_\ell\|_2 = 1$ we get $|S_{\ell,a}| \leq \sigma_\ell$, that, combined with equation (3.6.3), leads to

$$\alpha \min_a (\|F_a\|_2 \mathrm{tol}_a) \frac{1}{\sigma_\ell} \geq \mathrm{tol}_\ell^X. \tag{3.19}$$

From this equation, it appears that the stopping criterion threshold for the ℓ–th singular vector is proportional to the inverse of ℓ–th singular value. The smaller the singular value is, the more the associated stopping criterion threshold is relaxed. Eventually, for the last $(n-q)$ linear systems, the stopping criterion thresholds are larger than one. For those linear systems, the solution $X = 0$ is fine; this is another illustration that they do not need to be solved. In Figure 3.32, we plot the values of stopping criterion thresholds corresponding to the right–hand sides solved when computing the RCS for the cobra 14449. For that calculation 91 angles are considered, and the following solvers are used:

(a) the GMRES method with zero initial guess, in this case the 91 stopping criterion thresholds are set to 10^{-3};

(b) the GMRES method with GMRES with strategy 2 for the initial guess. In that case, solving a linear system with a given initial guess and a prescribed accuracy is equivalent to solving the associated error equation with a zero initial guess

using a relaxed accuracy. More precisely,

$$\frac{\|ZJ_a - F_a\|}{\|F_a\|} = \frac{\|Z(J_a^{(0)} + e) - F_a\|}{\|F_a\|} = \frac{\|r_0\|}{\|F_a\|}\frac{\|Ze - r_0\|}{\|r_0\|}$$

where $r_0 = ZJ_a^{(0)} - F_a$. Consequently solving $ZJ_a = F_a$ with a backward error ε_a is equivalent to solving $Ze = r_0$ with a backward error $\frac{\|r_0\|}{\|F_a\|}\varepsilon_a$. This latter backward error can be considered as relaxed thanks to the use of a nonzero initial guess. These latter backward errors are the ones displayed.

(c) the GMRES method with the singular vectors as right–hand sides, the solution of the $p = 91$ systems reduces to the solution of the $q = 21$ linear systems on the 21 first singular vectors.

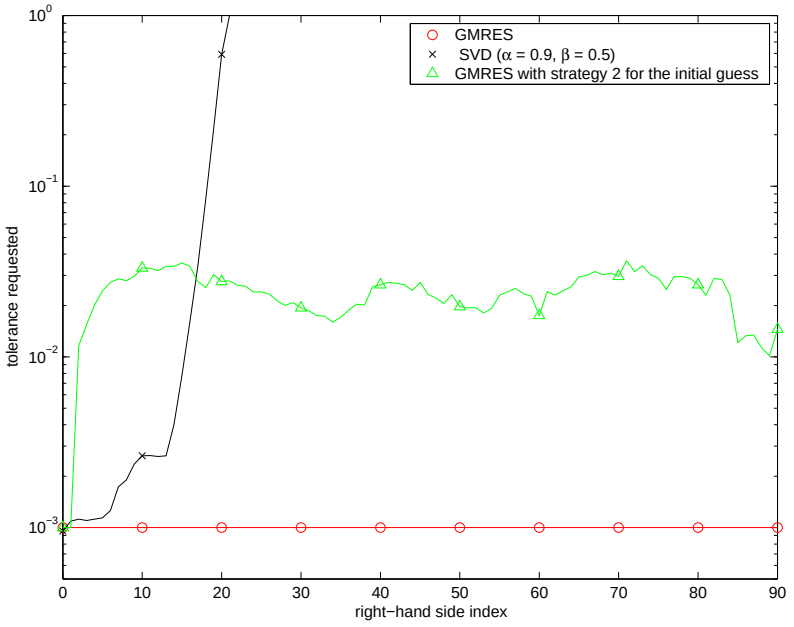


Figure 3.32: Stopping criterion thresholds on the right–hand sides for three different strategies.

In that figure, it can be seen that not only the number of linear systems to be solved is reduced with the SVD approach, but also the accuracies required for their solution.
Note that equation (3.6.5) is nice for interpretation purpose but should not be implemented. We illustrate our claim through this following extreme example. Let us assume that all the columns of F are the same vector (p times the same vector) and that the requested tolerances for each system are set to the same value tol. It is clear that $q = 1$ and this single system must be solved with the requested tolerance tol. Equation (3.6.3) performs well to set $\text{tol}_1^X = \text{tol}$. Equation (3.6.5) leads to $\text{tol}_1^X = \text{tol}/\sqrt{p}$ that is too low. Since our right–hand sides are highly dependent, this phenomenon might occur in many places. We take care to implement in our algorithm equation (3.6.3) and not its simplified version equation (3.6.5).

3.6.6 About the scaling among the $\|F_a\|_2 \mathrm{tol}_a$

Because of the term $\min_a(\|F_a\|_2 \mathrm{tol}_a)$ in equation (3.6.5), we should not have too large variations among each of the $\|F_a\|_2 \mathrm{tol}_a$ in order to avoid artificial unsuited imbalances between the stopping criterion thresholds. Let us illustrate this phenomenon through a small 2 by 2 example in real arithmetic.
Let us consider the QR factorization of $F = (F_1, F_2) = (Q_1, Q_2)R$ where

$$R = \begin{pmatrix} \nu_1 & \nu_2 \sin(\theta) \\ 0 & \nu_2 \cos(\theta) \end{pmatrix}.$$

where θ is set between 0 and $\pi/2$. In this case, the SVD of F is $F = (QU)\Sigma W^T$ where

$$\begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \mu \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{pmatrix}^T$$

where $\sigma_1 = \nu_1 \cos(\theta/2) + \nu_2 \sin(\theta/2)$ and $\sigma_2 = |\nu_1 \cos(\theta/2) - \nu_2 \sin(\theta/2)|$. We consider that the ν_a's are of the same order, and the angle θ is neither "too" close to $\pi/2\,\mathrm{x}$, (so that the vectors are not orthogonal), nor "too" close to 0 (such that the set of vectors is well conditioned). For example, we set $\nu_1 = \|F_1\| = \|F_2\| = \nu_2 = 1$ and $\theta = \pi/4$. The stopping criterion for the two systems are set so that $\nu_2 \mathrm{tol}_2 \gamma = \nu_1 \mathrm{tol}_1$, with $\gamma >> 1$. The accuracy requested on the first set of linear systems is high whereas the accuracy requested on the second is set low. In this case we expect our strategy to perform badly. Using equation (3.6.3), we obtain $q = 2$ since $\sigma_2 > \mathrm{tol}_1$. Using equation (3.6.3), we set the tol_ℓ^X, $\ell = 1, 2$ to

$$\frac{\sqrt{2}}{2}\mathrm{tol}_1.$$

Our strategy results in the solution of two sets of linear systems with right–hand sides (U_1, U_2), where the stopping criterion threshold is $\frac{\sqrt{2}}{2}\mathrm{tol}_1$. The initial problem was far simpler with the right-hand sides (F_1, F_2) and the requested accuracies tol_1 and $\mathrm{tol}_1\gamma$ respectively. The use of the SVD in a preprocessing phase is not relevant in that case where there is a bad scaling between the $\|F_a\|\mathrm{tol}_a$ (the ratio in our case is $\gamma >> 1$). Such a situation introduces undesirable difficulties for the block-GMRES that will attempt to solve the two sets of linear systems simultaneously and will not be able to realize that a better solution will be to solve them independently. This observation extends to the situation with several vectors. Possible remedies might exist but are out of the scope of this manuscript, as in our study, the norm of the right–hand sides are almost the same ($\pm 1\%$) and the stopping criterion thresholds are exactly the same.

3.6.7 SVD preprocessing in the block-GMRES method

The deflation technique described in Section 3.6.3 exactly corresponds to the deflation implemented in the first step of our block-GMRES implementation. For the sake of completeness, we have introduced the parameters α_ℓ and β so that each linear system associated with a particular singular vector can be solved by any solver,

including one right–hand side solvers. In that latter context, in order to alleviate the extra cost introduced by the lack of sharpness of our bound, we have introduced some relaxation heuristics. We show below how these heuristics can be removed in the framework of the block-GMRES solver.
We first recall that we have $F = US + E$. Starting from a zero initial guess for the solution of the linear systems that have U as right–hand sides, block-GMRES at step n, has

$$(U, \mathbb{Z}V_n) = V_{n+q} \begin{pmatrix} I_q & \\ 0_{n,q} & \bar{H}_{n+q,n} \end{pmatrix}.$$

Since U has orthonormal columns, we have $(V_0, \ldots, V_q) = U$. At that stage, if our wish was to solve the linear system $\mathbb{Z}X = U$, we would be interested in solving the least–squares problem

$$\min_{x \in \mathcal{K}} \|U - \mathbb{Z}x\|_2.$$

However, our initial goal is to solve $\mathbb{Z}J = US + E$ with $\|E\|_2$ small. A natural approach is to consider the least–squares problem

$$\min_{x \in \mathcal{K}} \|US - \mathbb{Z}x\|_2.$$

Since $x \in \mathcal{K}$, there exists y such that $x = V_n y$, this gives

$$\begin{aligned} US - \mathbb{Z}x &= US - \mathbb{Z}V_n y \\ &= V_{n+q} \begin{pmatrix} S \\ 0_{n,p} \end{pmatrix} - V_{n+q} \bar{H}_{n+q,n} y \\ &= V_{n+q} \left(\begin{pmatrix} S \\ 0_{n,p} \end{pmatrix} - \bar{H}_{n+q,n} y \right). \end{aligned}$$

Classically, we perform q Givens rotations on the n columns of $\bar{H}_{n+q,n}$ to obtain the n–by–n upper triangular matrix R_n such that $\bar{H}_{n+q,n} = \Theta_n \begin{pmatrix} R_n \\ 0_{q,n} \end{pmatrix}$. We obtain

$$US - \mathbb{Z}x = V_{n+q}\Theta_n \left(\Theta_n^H \begin{pmatrix} S \\ 0_{n,p} \end{pmatrix} - \begin{pmatrix} R_n \\ 0_{q,n} \end{pmatrix} \right).$$

With the notation,

$$\begin{pmatrix} g \\ \tau \end{pmatrix} = \Theta_n^H \begin{pmatrix} S \\ 0_{n,p} \end{pmatrix},$$

where g is n–by–p and τ is q–by–p, the approximate solution x is given by $V_n y$ where y is the solution of the triangular system $R_n y = g$. The norm of the residual of the a-th linear system is given by

$$\|F_a - \mathbb{Z}x_a\|_2 \leq \|US_a - \mathbb{Z}x_a\|_2 + \|E\|_2.$$

At each step of the Arnoldi iterations, the quantity $\|US_a - \mathbb{Z}x_a\|_2$ is given via

$$\|US_a - \mathbb{Z}x_a\|_2 \leq \|\tau_a\|_2.$$

The main part of the residual $\|US_a - \mathbb{Z}x_a\|_2$ is therefore given by the block-GMRES algorithm at a low computational cost. Consequently, the inequality that controls the residual associated with the a-th linear system is

$$\|F_a - \mathbb{Z}x_a\|_2 \leq \|\tau_a\|_2 + \|E\|_2.$$

This latter bound is sharper than equation (3.19). Indeed, rather than writing the triangle inequalities on a $(q+1)$–term sum, we write the triangular inequalities only for a two–term sum. The process is as follows. Fixing q so that equation (3.6.3) holds with $\beta < 1$, we set $\alpha = 1 - \beta$ and stop the block-GMRES iterations when $\|\tau_a\|_2 \leq \alpha$ for all $a = 1, \ldots, p$. In our block-GMRES implementation, we therefore choose this implementation to control the individual residuals.

3.6.8 Perspectives

From equation (3.6.1.2), we could compute explicitly the spherical harmonics $(\xi^\ell)_{\ell=-L_\varepsilon,\ldots,L_\varepsilon}$ since the coefficients of $F(\varphi)$ in this basis are known (the m–th coefficient is $e^{i\ell\varphi}$). An alternative strategy to solve the p linear systems is to

(a) compute the q_{sh} spherical harmonics (ξ^ℓ),

(b) solve the q_{sh} linear systems $\mathbb{Z}y = \xi$,

(c) finally, reconstruct the p solutions, $J(\varphi) = \sum_{\ell=-L_\varepsilon}^{L_\varepsilon} y_{q_{sh}} e^{i\ell\varphi}$.

We recall that the SVD strategy consists in

(a) computing the p right–hand sides $F(\varphi)$,

(b) computing the q first singular vectors U of the right–hand sides,

(c) solving the q linear systems $\mathbb{Z}x = U$,

(d) reconstructing the p solutions.

The computational work to construct the q_{sh} spherical harmonics is equivalent to computing q_{sh} columns of F. In the initial phase, the SVD approach requires us to compute $(p - q_{sh})$ extra right–hand sides. On the other hand, the cost of the SVD calculation is affordable as can be seen in Table 3.23. In a further study, it

Size of the problem	Elapsed time SVD	Elapsed time GMRES	# procs
40368	1063	214	4
71148	1903	388	4
161472	2992	550	8
288300	4923	1649	8

Table 3.23: Elapsed time to compute the SVD for 360 right–hand sides for the sphere. For comparison, we give the average elapsed time for the solution of one right–hand side using full GMRES (assembly of the preconditioner phase not included). The backward error is set to 10^{-2} for the GMRES solve.

would be interesting to compare the spherical harmonics and the SVD approaches

more deeply. In particular, we should look at the different costs for constructing the requested set of vectors and also the behaviour of the selected iterative solver on these right–hand sides, that are essentially different.
Finally, we should also point out the recent related work by Lötstedt and Nilsson [89]. In this work, the authors proposed a numerical scheme based on a dichotomy. Let us quickly describe their approach for a RCS computed on the interval $0^o : 1^o : 180^o$. In the first step, they solve the three systems corresponding to $0^o : 90^o : 180^o$; next they solve for the two remaining systems in the set $0^o : 45^o : 180^o$. At step s, they solve the 2^s remaining systems in the set $0^o : 90^o/2^s : 180^o$. Between each step, they decompose the right–hand sides, involved in the next step, into the set of right-hand sides already solved; this gives them an initial guess for the next linear systems to be solved. They eventually observe that after a certain number of steps, they do not need to solve a linear system anylonger as any new right–hand side can be written in the set of already solved right–hands sides. They theoretically end up with a bound for the number of steps in their method that can be compared to our bound given in equation (3.6.1.2). We should mention that our bound, based on the spherical harmonics analysis, gives a smaller estimate than their bound while requiring less assumptions.

3.7 Numerical behaviour of the multiple right–hand side solvers

In Section 3.6, we have illustrated the fact that the right–hand sides given by the engineers, when they compute a RCS, are strongly linked. In that case, the use of the SVD (see Section 3.6) is, in most cases, highly desirable to significantly reduce the number of linear systems to be effectively solved. However, we still have to solve the linear systems with the remaining right–hand sides. In that section, we investigate the numerical behaviour of two iterative methods designed to deal with multiple right–hand sides. More precisely, in Section 3.7.1 we consider the seed-GMRES method and in Section 3.7.2 the block-GMRES method. Both of these approaches have been described in detail in Chapter 2.

3.7.1 The seed–GMRES method

In this section, we aim at highlighting two numerical phenomena that have been observed when solving, using the seed-GMRES method, linear systems arising in electromagnetism applications. These numerical behaviours are representative but, in some sense, quite the opposite from each other. Firstly, we consider a test example where the seed approach is quite effective; secondly we show a different example where the seed approach is not too effective. In this latter situation, we show that the spectral low rank update preconditioner is a way to overcome the problem and makes seed-GMRES the most efficient solver. For the first example, described in detail in Section 3.7.1.1, we give a strategy to efficiently combine the numerical efficiency of the method while reducing the cost of its main time consuming numerical kernels. In the second case, we show, in Section 3.7.1.2, how the bad numerical behaviour can be fixed by using the spectral update preconditioner that nicely compensates for the weakness of the Krylov solver. We conclude with a comparative study on several variants of seed–GMRES.

3.7.1.1 The case of the almond 104793 and the restart seed–GMRES

The test example that we consider in this section, is the almond 104793 using the CFIE formulation for an RCS calculation on $\theta = 0^o : 0.5^o : 180^o$ and a stopping criterion threshold set to 10^{-3}. We should point out that this calculation exactly corresponds to the test case proposed for benchmarking purposes for the JINA 2002 workshop (see Section 3.2.4). In this case, strategy 2 for defining the initial guess for the GMRES method is very effective. The initial guess gives the correct answer on a spherical object, and so is very appropriate on any object that has a rather spherical shape, e.g. an almond. With this approach, the complete RCS calculation requires 1431 matrix–vector products, that have to be compared with about 4000 for the GMRES method with zero as the initial guess. This means that on average, each linear system is solved using only 6 iterations. For that calculation, seed-GMRES consumes 1185 matrix–vector products; that is, only 3 iterations on average per right-hand side. Unfortunately, if we look at the performance in elapsed time, the picture differs. GMRES with strategy 2 requires 6202 seconds, while seed-GMRES

takes 10460 seconds. In order to explain this behaviour, we give the profiling of these two runs in Table 3.24. This profiling reveals that the time spent by seed-

		seed–GMRES		GMRES with strategy 2	
		Solution phase (1185 iterations)		Solution phase (2152 iterations)	
operation	unitary time	# calls	total time	# calls	total time
ZDOT	0.0061	2575	16	3614	22
ZDSCAL	0.0006	1546	1	1792	1
ZCOPY	0.0005	722	0	722	0
ZAXPY	0.0082	2575	21	3614	30
MATVEC	2.9940	1185	3548	1431	4284
FROB	0.3636	1546	563	1792	651
total			4148		4989
		Projection		Initial Guess	
operation	unitary time	# calls	total time	# calls	total time
ZDOT	0.0061	28066	1703	360	3
ZDSCAL	0.0006				
ZCOPY	0.0005				
ZAXPY	0.0082	56132	4609	360	3
MATVEC	2.9940			360	1077
FROB	0.3636				
total			6312		1083
elapsed time			10460		6202

Table 3.24: Profiling details for the seed–GMRES method and GMRES with strategy 2 for the initial guess. The test example is the almond 104973 with $\theta = 0^o : 0.5^o : 180^o$ and the CFIE formulation.

GMRES in the ZAXPY and ZDOT kernels, involved in the minimization process on successive Krylov spaces (that are implemented to calculate the initial guesses), is larger than the elapsed time for the complete calculation using GMRES with strategy 2. In that minimization phase, the number of ZDOT operations is about $n_{\text{av}}p^2/2$, and the number of ZAXPY operations is about $n_{\text{av}}p^2$, where p is the number of right–hand sides and n_{av} the average size of the Krylov spaces (i.e. the average number of iterations to solve each right–hand side).
For instance, the residual corresponding to the last right–hand side is minimized $(p-1)$ times on the sequence of subspaces of size n_{av} generated for the solution of the first $(p-1)$ linear systems. In Figure 3.33, we plot the evolution of the backward error associated with the initial guess computed for the last right–hand side after each of the 359 minimizations. Starting from the value 1, the backward error decreases after each minimization but exhibits a long plateau before eventually converging faster, thanks to the influence of its few preceding right-hand sides. In Figure 3.34, we illustrate the same phenomenon in a different way. The curve K_ℓ represents the contribution of the minimization on the ℓ-th Krylov space, generated for solving the ℓ-th right–hand side, to the reduction of the residual norms of all the subsequent linear systems. The curve on the top left represents the norm of each residual after the minimization on the Krylov space generated for the first right–hand side. For the first right–hand side itself, its Krylov space gives an approximate

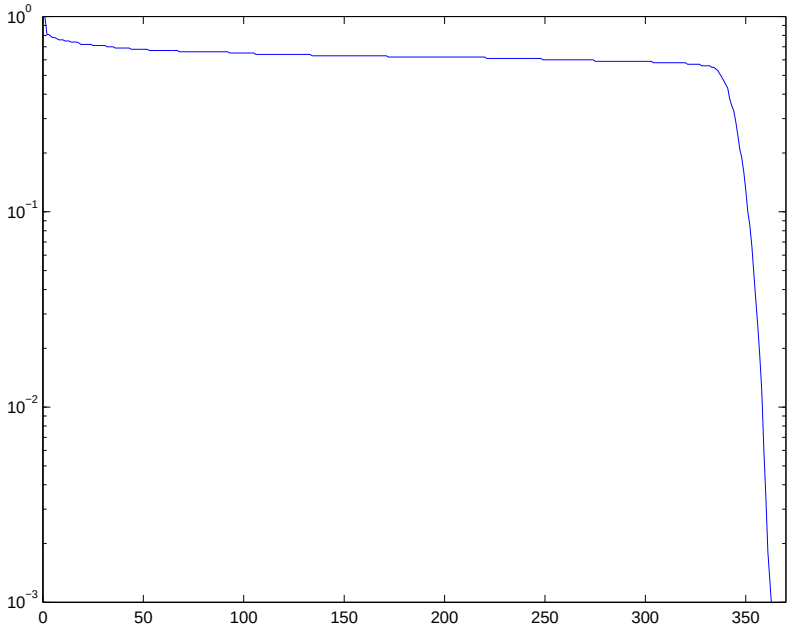


Figure 3.33: Evolution of the norm of the residual of the last right–hand sides at each minimization on the Krylov spaces generated for the solution of the previous right-hand sides. The test example is the almond 104973 with $\theta = 0^o : 0.5^o : 180^o$ and CFIE formulation.

solution with a backward error smaller than 10^{-3}, consequently the curve begins from 10^{-3}. For a given θ (abscissa), the values of the graph K_i at this abscissa indicate that its residual norm strictly reduces each time a linear system is solved. This is due to the fact that, at each time, its residual norm is minimized on the Krylov space generated for the right–hand side that has just been solved. It can also be seen that for $\theta \geq 2.5$, the norm of the updated residuals stagnates around $4 \cdot 10^{-3}$ before being eventually reduced to 10^{-3} by the Krylov space generated especially for it. Futhermore, the shape of K_1 shows that the Krylov space of the first right–hand side does not help much in minimizing the residual norm of the 10-th right–hand sides and the subsequent ones.
Based on the observation that

(a) using all the Krylov spaces to attempt to reduce the residual norm of all the remaining linear systems implies a very large number of minimizations;

(b) the influence of a given Krylov space is significant for reducing the residual norm of the very next right-hand sides, but negligible on those occuring later;

it appears natural to use the minimization phase for only a few vectors located close to the right–hand sides that have just been solved.
In oder to implement this idea, we first consider a sliding window of size w. This approach consists in exploiting the ℓ–th Krylov space to reduce the next w residuals. In Figure 3.35, we experiment with this sliding strategy for the almond 104973 (CFIE). For the first w right–hand sides, the seed–GMRES method and its sliding variant are the same method. The difference appears for the $(w+1)$–st right–

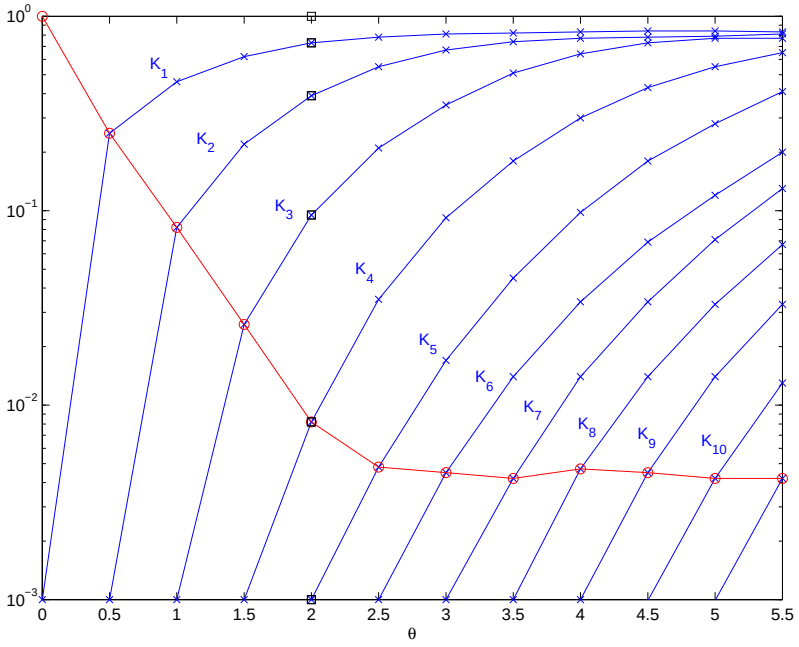


Figure 3.34: Norm of the residuals after each minimization on the Krylov spaces. The test example is the cobra 60695, $\theta = 0^o : 0.5^o : 5.5^o$. Each ($\times$) line corresponds to the minimization on a Krylov space, starting at the Krylov space associated with the first right–hand side for the top left curves, then the Krylov space for the second right–hand side, etc. The ($\bigcirc$) gives the norm of the residual just before the GMRES solver is used on it. The ($\square$) shows the residual for a fixed right–hand side ($\theta = 2^o$).

hand side where the sliding version induces a bad behaviour. For the seed–GMRES method, the $(w+1)$–st right–hand side is minimized on $\mathcal{K}_1$, then $\mathcal{K}_2$, ..., $\mathcal{K}_w$ and eventually on its own Krylov space. For the sliding version, the $(w+1)$–st vector is minimized on $\mathcal{K}_2$, ... , $\mathcal{K}_w$ and eventually on its own Krylov space.

In the seed–GMRES and the sliding version, F_2 is minimized on $\mathcal{K}_1$, so the Krylov space constructed from F_2, $\mathcal{K}_2$, is fully meaningful only for a vector that has been previously minimized on $\mathcal{K}_1$ as $\mathcal{K}_2$ cannot contain both itself and $\mathcal{K}_1$. The seed-approach implies a sonhood chain among the Krylov spaces: a Krylov space, $\mathcal{K}_j$, is the father of another, $\mathcal{K}_\ell$, if the vector used to generate $\mathcal{K}_\ell$ has been previously minimized on $\mathcal{K}_j$. To be efficient, the minimization on $\mathcal{K}_\ell$ requires that the same process has been applied to the father hierarchy of $\mathcal{K}_\ell$. The sliding version of seed–GMRES clearly breaks this hierarchy among the Krylov spaces. In Figure 3.35, the catastrophic effect of this break is visible starting from the $(w+1)$-st right–hand side. Several strategies are possible to preserve a sequence among the right–hand sides. The simplest is to gather consecutive right–hand sides in blocks of size r and run the seed–GMRES method on each block independently. We refer to this version as the consecutive seed–GMRES method. It only depends on the parameter r. One can easily imagine tuning this parameter at runtime through an auto-learning phase, based on the following simple model of the seed-GMRES behaviour. We often observed that the seed–GMRES behaviour consists of a transient phase followed by

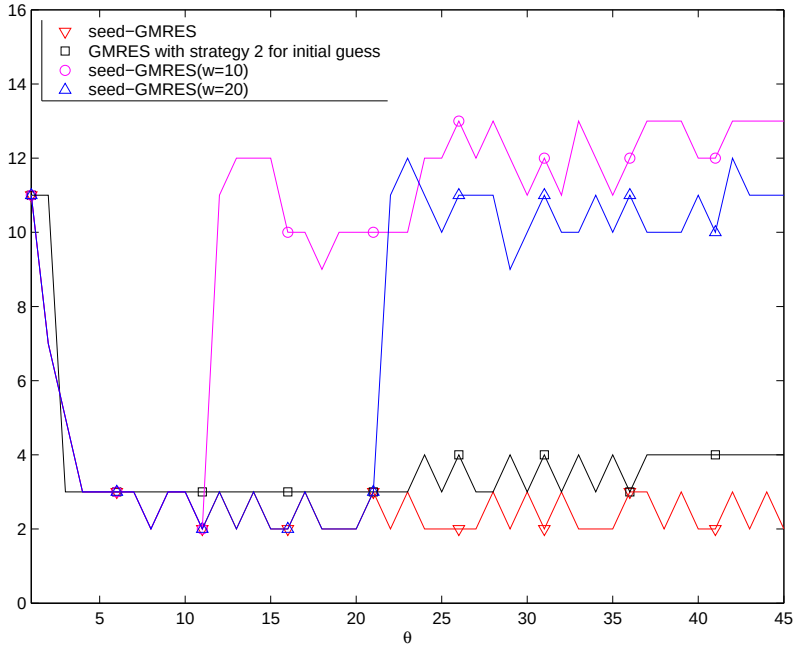


Figure 3.35: The seed–GMRES method is run on the almond 104973 (CFIE), we observe the number of iterations to reach the stopping criterion threshold for each right–hand sides. A variant of the seed–GMRES is also tested, the ℓ–th Krylov space is used to minimize the residuals of the next w residuals, we consider $w = 10$ and $w = 20$.

a steady state phase. Moreover, we assume that this model is independent of the initial right–hand side. We illustrate our approach on the almond. Starting the seed–GMRES method from the right–hand side $F(\theta = 0^o)$, we obtain the following sequence for the number of iterations for the first 15 right-hand sides:

$$\left[\begin{array}{cccccccccccccc} 11 & 7 & 5 & 3 & 3 & 3 & 3 & 2 & 3 & 3 & 2 & 3 & 2 & 3 & 2 \end{array}\right].$$

The transient phase is defined by the sequence 11, 7 and 5; the stationary phase is the remaining part where the number of iterations does not change much. It is well defined by a plateau at a value equal to the average number of iterations for the right–hand sides solved in this stationary phase. In the almond case, we obtain $(55 - 11 - 7 - 5)/12 \sim 2.67$. Finally we obtain the following model law:

$$\left[\begin{array}{cccccccccccccc} 11 & 7 & 5 & 2.67 & 2.67 & 2.67 & 2.67 & 2.67 & 2.67 & 2.67 & 2.67 & 2.67 & 2.67 & 2.67 & 2.67 \end{array}\right].$$

This law describes the number of iterations of the seed–GMRES method on 15 right–hand sides starting from $F(\theta = 0^o)$. We assume that this behaviour holds for any starting right–hand side $F(\theta)$. Using our model law and a value r for the block size, we are able to forecast the elapsed time and also the total number of iterations of the consecutive seed–GMRES method. In Figure 3.36, we plot the forecast elapsed time and number of iterations for all the values of r, ranging from 0, that corresponds to classical GMRES with a default guess, up to p, the classical seed–GMRES. For small values of r, the number of iterations quickly

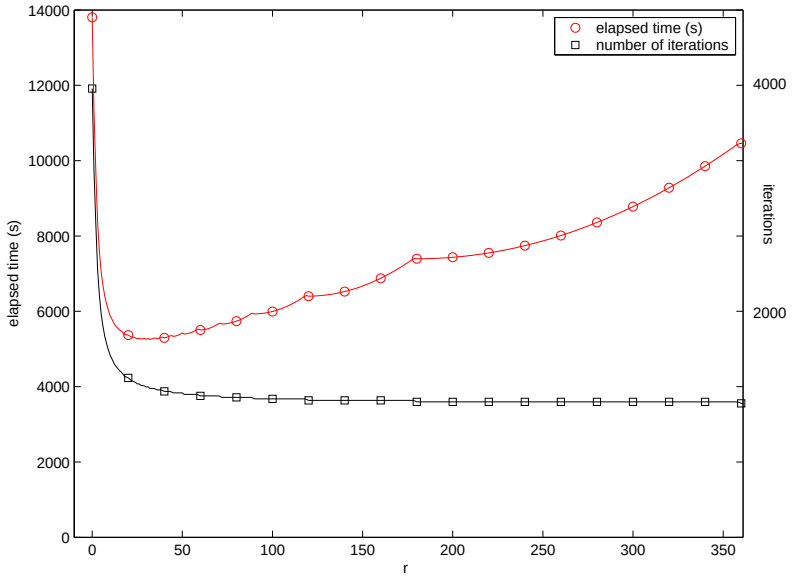


Figure 3.36: Elapsed time and number iterations versus the size of the restart r for restarted seed–GMRES. The test example is the almond 104973 with $\theta = 0^o : 0.5^o : 180^o$ and CFIE formulation.

decreases, so does the elapsed times. When r increases, the number of transient phases diminishes (e.g. 17 for $r = 20$); eventually, the number of extra iterations they induce is not significant compared with the total number of iterations. This implies that the total number of iterations becomes rather independent for a large enough size of the restarts. Consequently, the GMRES solves in the Seed GMRES methods take approximatively the same time; while the computational cost of the minimization phase increases with the restart size r, as does the elapsed time. From this analysis, we deduce that an optimum exists that minimizes the overall elapsed time. On the almond test example, we remark that the minimum number of total iterations is observed when the size of restart is 33. In this case, the total numbers of iterations is 1318 and the elapsed time is about 5300 seconds. With this technique, the seed–GMRES method becomes competitive with the classical GMRES method using strategy 2 for the initial guess (6200 seconds).
In this part, we have seen that, in the seed–GMRES, deteriorating the convergence (the number of iterations) may result in reducing the overall elapsed time. This can be related to the observed behaviour of the classical GMRES method where the full GMRES may exhibit better convergence properties than restarted GMRES but eventually takes more time.
Finally note that the number of singular values q such that $\sigma_q/\sigma_1 < \varepsilon = 10^{-4}$ (where σ_ℓ denotes the ℓ–th singular value of F) is equal to 67 for the almond 104973. The number of right–hand sides is $p = 360$. If we stored the 67 solutions associated with 67 equally spaced right–hand sides (say), we should recover the 360 solutions of the object. This is the strategy adopted by Lötstedt and Nilsson [89]. Since the seed–GMRES method minimize succesively and independently on the different Krylov subspaces, this global effect does not appear in this experiment.

3.7.1.2 Complementarity between the seed-GMRES and the spectral low rank update preconditioner

The example considered in this section is the cobra 60695. We first study the RCS that corresponds to the first twelve right–hand sides: $\theta = 0^o : 0.5^o : 5.5^o$. On these right–hand sides, seed-GMRES exhibits a very strange behaviour. To illustrate it, we first compare the seed-GMRES method with the GMRES method with either strategy 2 or zero for the initial guess. These three methods only differ in their choice of the initial guess J_0. In Table 3.25, we give the number of iterations per right–hand side for the three solvers and the initial backward error, $\|r_0\|_2/\|F\|_2 = \|F - \mathbb{Z}J_0\|_2/\|F\|_2$. Of course, we recall that the normalized initial residuals for the GMRES method with default initial guess are all equal to 1.0. In Table 3.25, we

	default guess	strat. 2		seed–GMRES	
(θ, φ)	# iter	# iter	$\|r_0\|_2/\|F\|_2$	# iter	$\|r_0\|_2/\|b\|_2$
(0.0,0.0)	338	338	1.000	338	1.000
(0.5,0.0)	339	339	1.000	190	0.230
(1.0,0.0)	340	63	0.071	210	0.074
(1.5,0.0)	341	56	0.048	130	0.024
(2.0,0.0)	341	81	0.039	197	0.007
(2.5,0.0)	341	116	0.037	215	0.004
(3.0,0.0)	342	139	0.028	237	0.004
(3.5,0.0)	342	151	0.028	226	0.004
(4.0,0.0)	343	159	0.024	254	0.004
(4.5,0.0)	344	180	0.022	258	0.004
(5.0,0.0)	345	179	0.027	265	0.004
(5.5,0.0)	346	192	0.022	299	0.004
# iterations	4102	2005		2819	
elapsed time (s)	15565.6	6765.6		9863.7	

Table 3.25: Number of iterations per right–hand side and initial residual norm for the GMRES method with default initial guess, strategy 2 and the seed–GMRES method. The test example is the cobra 60695.

see that the seed–GMRES method performs well in decreasing the initial residual norms. However, the final effect on the number of iterations performed is not what we might have expected. The initial residual norm provided by the seed–GMRES method is nearly always by far the smallest. Unfortunately, starting from the seed initial guess that is the closest (in the backward error sense) to the solution does not guarantee a fast convergence. From that initial guess, GMRES performs rather poorly. It performs only slightly better, if it starts from zero and is outperformed by the approach that starts from the initial guess provided by strategy 2. In other words and surprisingly enough, the approach that gives the smallest initial residual norm is not the method that gives the smallest number of iterations.

We have observed this behaviour of the seed–GMRES method on some other difficult problems. Intuitively, it seems to us that an analogy exists between this behaviour and the observed stagnation of the classical restarted GMRES method. In the two cases, an initial guess is extracted from a Krylov space to generate a new Krylov space. In the restarted GMRES method, one possible remedy is to use the spectral

low rank update preconditioner (see Section 3.3.2.3 and [44]).
In Figure 3.37, we investigate this possibility. For each right–hand side, $\theta = 60^o$:

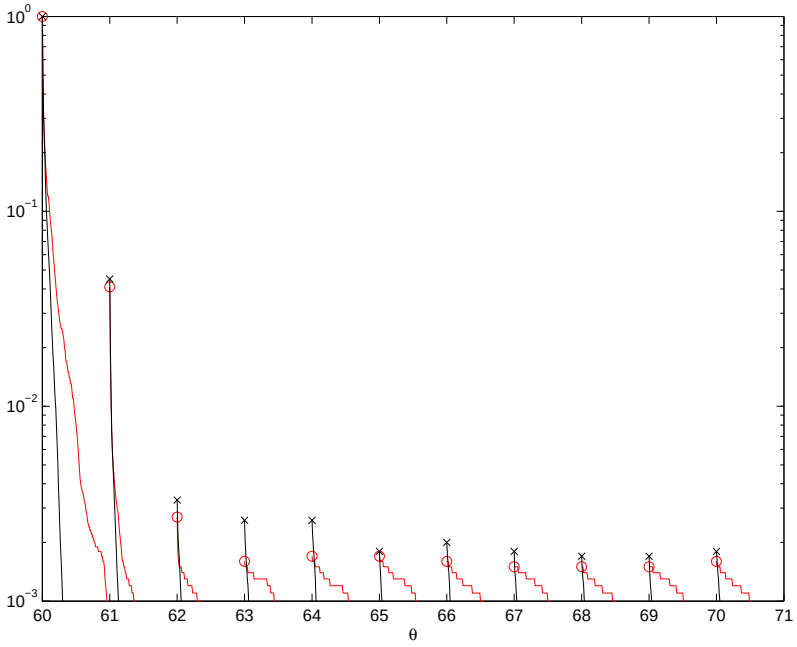


Figure 3.37: Comparison between the seed–GMRES method (○) and the seed–GMRES method with the spectral low rank update (×). The fifteenth smallest (in modulus) eigenvalues are shifted close to one with the spectral low rank update.

$1^o : 70^o$, we plot the convergence of the seed–GMRES method and the seed–GMRES method with the spectral low rank update preconditioner (both use the Frobenius preconditioner). While the norm of the initial residuals are about the same for the two methods, the rate of convergence is significantly improved by the spectral low rank update preconditioner. Finally, in Table 3.26, we make a comparative study to highlight the improvement introduced by the spectral low rank update in the context of the seed–GMRES method. Without the spectral low rank update, the

Solver	# iterations	
	M_{Frob}	M_{SLRU}
GMRES with default initial guess	1071	341
seed–GMRES	597	92
GMRES with strategy 1	485	111
GMRES with strategy 2	363	171
block–GMRES	249	153

Table 3.26: Number of iterations for various GMRES methods with or without the spectral low rank update preconditioner (exploiting the 20 eigenvectors). The test example is the cetaf 5391, $(\theta = 60^o : 1^o : 70^o, \varphi = 0^o)$.

seed–GMRES method performs poorly. It is less efficient than the GMRES method with either the first or the second initial guess strategy. The spectral low rank

update preconditioner improves the rate of convergence of the four methods that we have considered in that table. It is important to notice that the use of this preconditioner completly changes the performance ranking of the solvers and seed–GMRES becomes the most efficient. The complementarity of the seed–GMRES method and the spectral low rank update results in an excellent symbiosis. The solution offered by the seed–GMRES method is not acceptable, the solution offered by the spectral low rank update is correct (but block–GMRES performs better), the symbiosis of the two gives rise to the best method. The seed–GMRES method provides a small initial residual and the spectral low rank update ensure a good rate of convergence of GMRES iterations: in a race starting close from the arrival (seed strategy) and running fast (spectral low rank preconditioner) ensures to finish first!

3.7.1.3 Some variants of the seed–GMRES algorithm

In this section, we investigate three variants of the seed–GMRES methods. These three methods attempt to combine the appealing physical properties of the initial guess provided by strategy 2 with some linear algebra information. We briefly describe the methods and outline the underlying motivation for their implementations. The results are summarized in Table 3.27.

The seed–GMRES method is nothing but the GMRES method with a particular choice for the initial guess. The choice is governed by the seed strategy. Possible improvements would be to use different strategies to set up the initial guesses, aimed at improving the efficiency of the subsequent GMRES iterations. Using J_2, the initial guess provided by strategy 2, and J_{seed}, the initial guess built by the seed strategy, we compute the initial guess J_B that minimizes

$$\min_{J \in \mathrm{Span}(J_2, J_{seed})} \|F_\ell - \mathbb{Z}J\|_2.$$

We have $J_B = \mu J_2 + \nu J_{seed}$. With this approach, the initial residual norm is always better than those provided by any of the other two; we would expect that the GMRES iterations behaves better. Unfortunately, our experiments reveal the contrary. One possible explanation is that the resulting Krylov space breaks the sonhood chain described in Section 3.7.1.1. Consequently, the minimization on the successive Krylov spaces no longer makes much sense in the seed-philosophy. From our experiments, we observe that this strategy only succeeds in damaging the GMRES method with the second initial guess strategy, while it only slightly improves the seed-GMRES.

GMRES	3820
GMRES with initial guess J_2	1965
seed–GMRES	2373
GMRES with initial Guess Best of J_2 and J_{seed}	2132
seed–GMRES with $\mathbb{Z}(J_2, J_{seed})$ in the front of the Krylov space	2374
seed–GMRES with $\mathbb{Z}(J_2, J_{seed})$ at the end of the Krylov space	2373

Table 3.27: Three seed–GMRES variants on the cobra 14449 test example with $\theta = 0^o : 1^o : 35^o$.

The second approach consists in augmenting the spaces with $(\mathbb{Z}J_2, \mathbb{Z}J_{seed})$. At each step j of the Arnoldi iteration, we construct v_{j+1} and the $(j+1)$–st column of $R_{j+1,j+1}$ so that

$$(\mathbb{Z}J_2, \mathbb{Z}J_{seed}, b, \mathbb{Z}v_3, \mathbb{Z}v_4, \ldots, \mathbb{Z}v_j) = (v_1, v_2, v_3, \ldots, v_{j+1})R_{j+1,j+1}.$$

Then we minimize

$$\min_{J \in \mathrm{Span}(J_2, J_{seed}, v_2, \ldots, v_j)} \|F_\ell - \mathbb{Z}J\|_2.$$

Note that if J_2 and J_{seed} were eigenvectors, then this method would reduce to GMRES augmented with eigenvectors [30] and the subspace $(v_0, v_1, v_2, \ldots, v_{j+1})$ would be a Krylov space. The goal here is to enlarge the minimization space from $(v_2, \ldots, v_j)$ to $(J_2, J_{seed}, v_2, \ldots, v_j)$, and so enable us to add the information from J_2 and J_{seed}. With this initial guess strategy, this information is, in some sense, added with some prescribed constant coefficients (e.g. $(0,1)$ for the seed strategy and $(1,0)$ for initial guess with the second strategy, (μ, ν) for the first variants of seed GMRES). In this variant, the coefficients of J_2 and J_{seed} vary at each step so that the residuals are minimized. Following this argument, we would have expected to get a method that improves the performance of the other GMRES variants. Unfortunately, this is not the case. The total number of iterations of this method increases to 2374. The spaces constructed by this method are no longer Krylov spaces and the minimization on these spaces is not efficient; even though they were augmented with the vectors J_2 and J_{seed}, that are both good methods.

Augmenting the space with vectors that do not represent a stable space for $\mathbb{Z}$, prevents us to constructing a Krylov space and deteriorates the convergence behaviour of the iterative schemes. Another possibility is to enlarge the Krylov space by adding the vectors J_2 and J_{seed}, only for the minimization step of GMRES. That is, at each step j we still construct the standard Krylov space, $\mathcal{K}_{j+1}$, but perform the minimization on the augmented space as follows:

$$\min_{J \in \mathrm{Span}(J_2, J_{seed}, \mathcal{K}_j)} \|F_\ell - \mathbb{Z}J\|_2.$$

The motivation is to avoid the interference between the vectors J_2 and J_{seed} and the constructed Krylov space while minimizing on a larger space. We plot in Figure 3.38, the backward error. In Figure 3.38, we plot the convergence history, associated with the 11-th right hand side, for classical seed-GMRES and the variant described above (minimization on $\mathrm{Span}(J_2, J_{seed}, \mathcal{K}_j)$). In the very early iterations, the vectors J_2 and J_{seed} significantly contribute to reduce the backward error, unfortunately this positive effect quickly vanishes and both approaches eventually behave the same.

We conclude our dicussion of these three variants by mentioning that we did not manage to successfully combine physical and linear algebra information to design a linear solver that outperforms the classical seed–GMRES. Another approach that deserves to be investigated in this framework was proposed by Chapman and Saad [30]. It consists in running block–GMRES starting with the block composed by (F, J_2, J_{seed}).

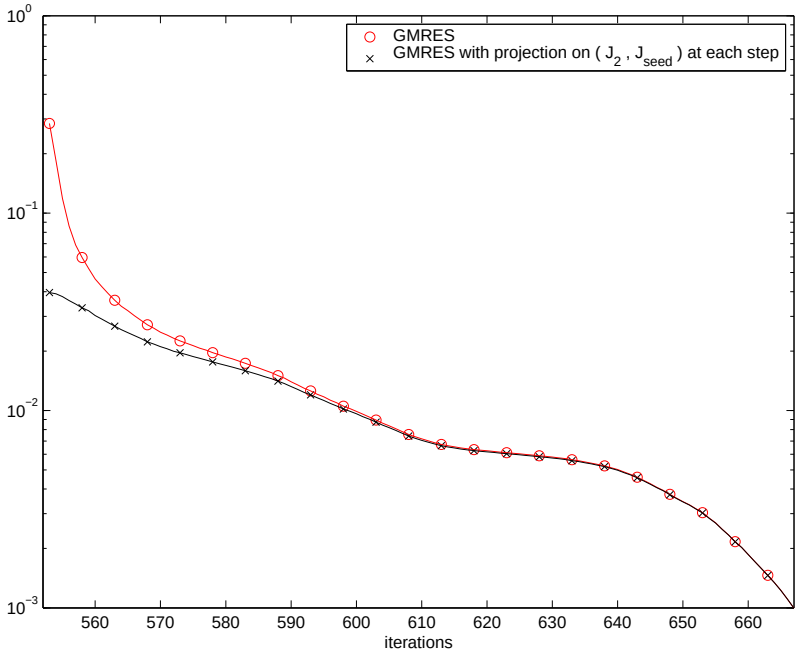


Figure 3.38: Backward errors of seed-GMRES (minimization on $\mathcal{K}_j$) and of seed-GMRES post–augmented (minimization on $\mathrm{Span}(J_2, J_{seed}, \mathcal{K}_j)$). The test example is the cobra 14449, the right–hand side corresponds to $\theta = 5^o$ and is solved starting from the 552–nd iteration of the process.

3.7.2 The block–GMRES method

The block strategies represent an alternative to the seed methods to deal efficiently with several right–hand sides. They are widely used in electromagnetism. For example, Paul Soudais [128] presents the multiGCR algorithm. This algorithm has been successfully used by Poirier [104] and by Simon [123]. In this manuscript, we focused on the block–GMRES method since the GMRES method has proved its efficiency for the solution of the linear systems arising from the EFIE formulation when only one right–hand side has to be solved.

3.7.2.1 About the continuum seed–block GMRES method

In Section 2.6, the block–GMRES method is presented as an Arnoldi process to construct the block–Krylov space of the right–hand sides associated with the matrix $\mathbb{Z}$. At each step n, we choose a vector among the p_n vectors from the Arnoldi process. In this section, we test a variant of the classic algorithm called block–seed–GMRES.

The test case is the cobra 14449, for $\theta = 0^o : 1^o : 35^o$. In Table 3.27, the number of total iterations requested for the convergence of the 36 right–hand sides is 960 for block–seed–GMRES, this is fairly good if we compare it with seed–GMRES and its variants. However the classic block–GMRES method, by itself requires 683 iterations. In our experiments, block–seed–GMRES does not perform better than

the classic block–GMRES. We do not pursue any further studies.

3.7.2.2 About deflation strategies in the block–GMRES method

In Section 2.6, deflation was described as a strategy to throw away a youngest son once the right-hand side associated with its root has converged. We propose to test in this section two strategies:

(a) block–GMRES without any deflation,

(b) block–GMRES with deflation of the converged right–hand sides.

We mention that implementing the deflation of the converged right–hand sides can also be interpreted as a strategy for selecting the next youngest son in the Arnoldi process, that is, a youngest son will never be selected if the right–hand side associated with its root has converged. In that respect, the following numerical experiment can be interpreted as an illustration for the effect of

either one deflation strategy,

or one selection strategy for the next youngest son,

on the convergence behaviour of the block–GMRES method.
The test example chosen is the cobra 14449 with $(\theta = 0^o : 1^o : 35^o, \varphi = 0^o)$. On this example, block–GMRES without any deflation performs better than block–GMRES with deflation of the converged right–hand sides. The total number of iterations for block–GMRES without any deflation is 683 and the total number of iterations for block–GMRES with deflation of the converged right–hand sides is 725.
In Figure 3.39, we describe the later iterations of both methods. For each right–hand side, we plot the iteration number where it has converged using both the block–GMRES method without deflation and the block–GMRES method with deflation of the converged right–hand sides. From the first iteration to the 662–nd, both algorithms behave exactly the same and build the same Krylov space by applying the Arnoldi process cyclically to the vectors within the previous block. It can be seen in the figure that the first residuals that converge are for $\theta = 29^o$ and $\theta = 30^o$ at iteration 663. However, until iteration 673 the two algorithms still construct the same Krylov space as the Arnoldi process still uses vectors associated with not-yet converged right–hand sides. The difference appears at iteration 674, where the next Arnoldi vector should be computed with the vector associated with the right–hand side ($\theta = 25^o$) that has converged at the previous block-iteration. The block-GMRES that deflates the converged right–hand sides, searches for the next vector that is associated with a not-yet converged right–hand side and deflates/skips those corresponding to right–hand sides sides that have converged. In that example, its next Arnoldi vector is computed with the vector associated with $\theta = 0^o$ (as right–hand sides $\theta = 25^o$ to $\theta = 35^o$ have converged in the previous block-iteration). The block-GMRES without deflation computes its next Arnoldi vector using the vector associated with $\theta = 25^o$. From this step, the two methods differ; they construct two different Krylov spaces and have a different convergence behaviour. As can be seen in this figure, the method without deflation enables the fastest convergence for

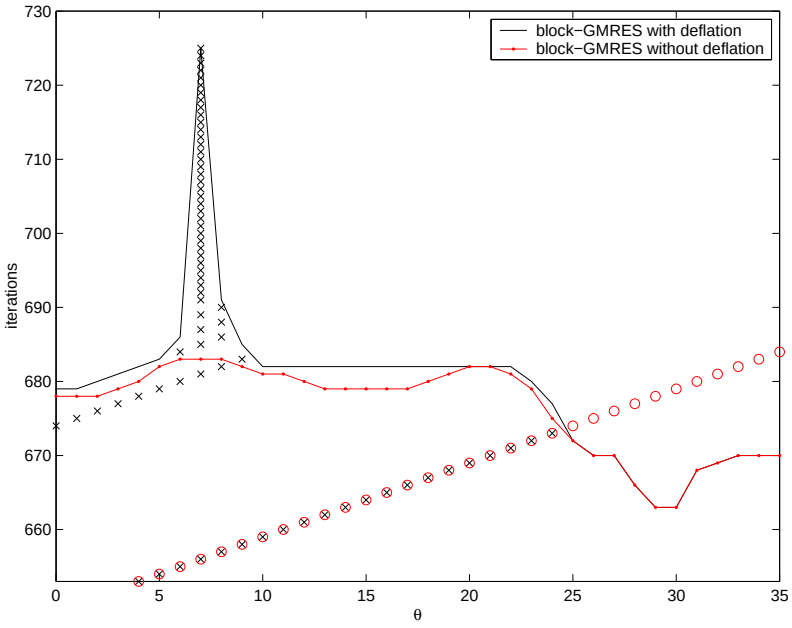


Figure 3.39: For each right–hand side, we plot the iteration number where it has converged using either the block–GMRES method without deflation or the block–GMRES method with deflation of the converged right–hand sides. ◯ indicates the index of the current vector used to build the next Arnoldi vector for the Krylov space generated by the block–GMRES method without any deflation. Similarly, we use × for the block–GMRES method with deflation of the converged right–hand sides. The test example is the cobra 14449 with $(\theta = 0^o : 1^o : 35^o, \varphi = 0^o)$

all the right-hand sides. The method with deflation seems to lose some information that results in noticeably slowing down its convergence. This bad behaviour has been observed on all the experiments we have performed with the block–GMRES with the deflation of the converged residuals. For this reason, we conclude that deflation should not be performed in the block–GMRES algorithm.

3.7.2.3 Restart and gathering of matrix–vector products

The restarting strategy can also be considered, and we implemented it in our block-GMRES algorithm. In Table 3.28, we report on the total number of iterations required by the block-GMRES(500) and full block-GMRES. We also report on the corresponding elapsed time. For that reason, we consider a block-GMRES implementation where the matrix-vector products have been gathered. In that table, it

	# iterations	elapsed time
gathered(11)–block–GMRES	1211	8135.0 s
block–GMRES	1211	9618.5 s
block–GMRES with restart 500	2958	13482.3 s

Table 3.28: Restart and gathering of matrix–vector products in the block–GMRES algorithm. The test example is the cobra 60695 for $(\theta = 20^o : 1^o : 30^o, \varphi = 0^o)$

can be seen that the restart significantly slows down the convergence and gathering the matrix-vector products speeds up the calculation.

3.7.2.4 The block–GMRES method combined with SVD preprocessing

Finally, we investigate the solution technique that appears to be the most efficient for large RCS calculations. It combines the SVD approach on the right–hand sides followed by a block-GMRES solver without deflation. We report in Table 3.29 the number of iterations and the elapsed time of the block-GMRES with and without the spectral low rank update preconditioner. For the sake of comparison, we also display preliminary results on the same RCS calculation using classical GMRES with and without the spectral low rank update preconditioner. The test example is the Airbus 23676 for $\theta = 0^o : 1^o : 180^o$. The combination of the SVD approach and

	# iterations	elapsed time
GMRES with default initial guess	86940	33 hours 52 minutes
GMRES with $M_{SLRU}(20)$ and default initial guess	47197	18 hours 21 minutes
SVD with block–GMRES	1501	4 hours 34 minutes
SVD with block–GMRES using M_{SLRU}	1357	4 hours 33 minutes

Table 3.29: Airbus 23676 with block–GMRES SVD (49) compared to blockgmres svd (49rhs) LRU(20)

the block–GMRES algorithm gives rise to a fairly efficient solver. Firstly, the SVD enables us to reduce the number of linear systems to be effectively solved from 181 to 49 and also permits us to reduce most of the stopping criterion thresholds used for these 49 right-hand sides. Secondly, the block–GMRES method solves these 49 right-hand sides efficiently. Finally, we mention that the benefit of using the spectral low rank preconditioner seems less important than for one right-hand sides method. The time saved by the reduction of the iterations is compensated by the extra cost within each iteration.
In Figure 3.40, we show the number of iterations required for each right–hand side to converge down to the requested accuracy for the four methods considered in Table 3.29.
In Figure 3.41, similar results are displayed using a different format. We depict the backward error level-curve for all the linear systems at different iteration numbers; that is, for a given iteration, the backward error associated with the current iterates are plotted.
Finally, in Table 3.30 we report on the elapsed time for each basic operation involved in the complete calculation based on the SVD preprocessing phase and the block-GMRES solution with the spectral low rank update preconditioner. The vector-vector operations clearly dominate and are far more expensive than the matrix-vector product involved in the FMM and the preconditioner. The orthogonalization scheme used is MGS2(K) and all the reorthogonalizations are performed.
We consider a second test example that is the coated cone sphere, $\theta = 0^o : 1^o : 180^o$. In Table 3.31, we report on the performance of four linear solvers. The seed–GMRES

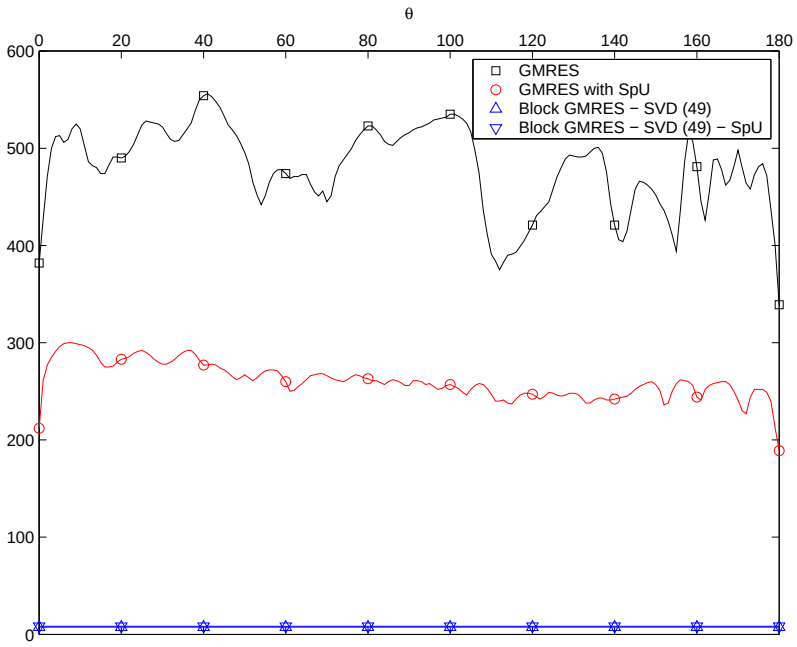


Figure 3.40: For each right–hand side, we plot the number of iterations (matrix–vector products) required to converge. The test example is the Airbus $\theta = 0^o : 1^o : 180^o$.

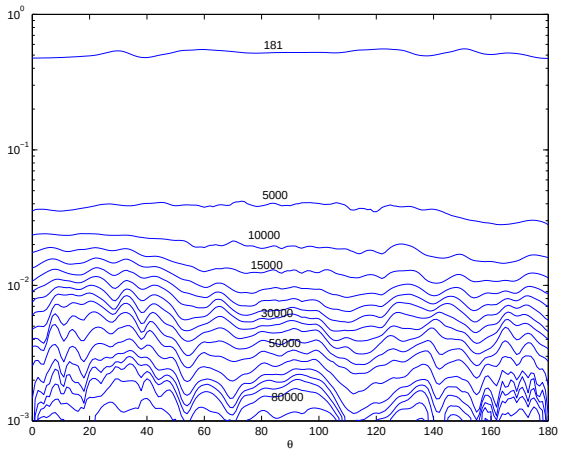
operations	#	elapsed time
FMM	1540	1773
frod	1540	401
slru	1540	46
dot	3000000	7751
zaxpy	3000000	6412
total	1	16383

Table 3.30: Detailed operation count in the block–GMRES method with SVD preprocessing. The test case is the Airbus 23676.

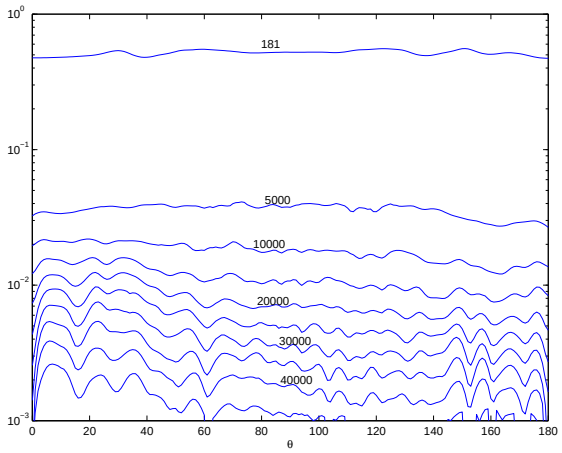
	# iterations	elapsed time (s)
GMRES with second strategy initial guess	3797	29726
seed–GMRES	784	10328
seed–GMRES SVD(16)	632	10930
block–GMRES SVD(19)	329	8894

Table 3.31: Four competitive methods on the coated cone sphere, $\theta = 0^o : 1^o : 180^o$.

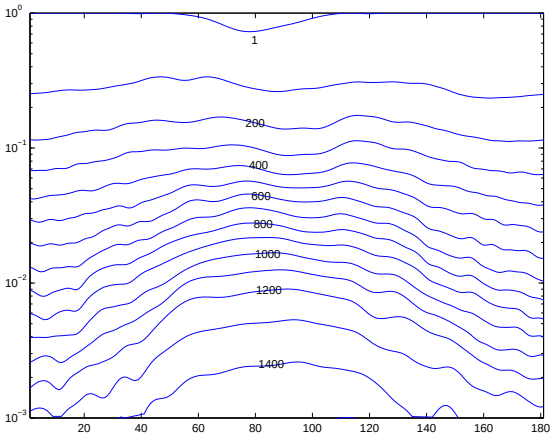
SVD performs poorly, the value of α is set to 0.1 in this experiment. Some stopping criterion threshold are very low and required too many iterations. This illustrates a drawback of the SVD preprocessing when it is followed by a sequence of one-right-hand side solvers; the tuning of the value for α is not easy. The SVD preprocessing followed by block–GMRES performs better.



(a) GMRES with default initial guess.



(b) GMRES with the spectral low rank update and default initial guess.



(c) block–GMRES with SVD preprocessing.

Figure 3.41: Level lines that represents the convergence of GMRES, GMRES with the spectral low rank update, and block–GMRES with SVD preprocessing. Each line gives the level of backward error for all the right–hand sides obtained at a given iteration. The test example is the Airbus with $\theta = 0^o : 1^o : 180^o$.

3.8 Prospectives

3.8.1 Using spectral information in the multiple right–hand sides context

In the single right–hand side case, at each restart, the GMRES–DR method enables us (a) to extract some spectral information from the Krylov space and (b) to use this information for building the next Krylov space aiming at speeding up its convergence. In Section 2.4, the implementation of the method is given and in Section 3.4.1 its behaviour in the electromagnetism context is examined. In the multiple right–hand sides case, we have shown that the use of the spectral low rank update preconditioner increases the efficiency of the solvers (GMRES and seed–GMRES in particular). In the experiments we have shown, the spectral information was calculated in a pre–processing phase using an eigensolver; namely ARPACK in forward mode. A more elegant possibility is to extract this spectral information from previous linear system solutions. Exactly as GMRES–DR does from one restart to the next, but, in our case, from one right–hand side to the next. Firstly, we consider an example where we extract from the GMRES–DR iterations the smallest eigenvalues and the associated eigenvectors. Secondly, we investigate different possible strategies to exploit this spectral information.

3.8.1.1 Spectral information from the GMRES–DR method

In Figures 3.21 and 3.22, we have illustrated that the GMRES–DR method succeeds in finding good approximations of the eigenvalues. This is not surprising since, in a sense, GMRES–DR is nothing but an eigensolver that solves linear systems. In Table 3.32, we report on the backward error associated with the ten harmonic Ritz vectors corresponding to the smallest (in modulus) harmonic Ritz values computed by GMRES–DR on the Airbus 213084. For the backward error on the eigenvectors, we consider the definition given in [29]; that is, let $y \neq 0$ be an approximation of an eigenvector so that the backward error η associated with y is

$$\eta = \frac{\|\mathbb{Z}y - y\frac{y^H \mathbb{Z} y}{y^H y}\|}{\|\mathbb{Z}\|_2 \|y\|_2}.$$

For estimating $\|\mathbb{Z}\|_2$, we use the approximation of the spectral radius given by the Arnoldi iterations. Finally, we mention that solving the linear system up to 10^{-2} requires 183 iterations, the extra cost of this run is 317 iterations $(500 - 183)$ and requires 10062.1 seconds on eight processors.

3.8.1.2 Strategies that use the spectral information.

If we know r eigenvectors $U = (u_1, u_2, \ldots, u_r)$ associated with the r smallest eigenvalues of $\mathbb{Z}$, there are at least three possibilities for using them to improve the iterative solution of the right–hand sides F:

(a) use them to build a preconditioner; for instance, the spectral low rank update preconditioner;

η_1	$3.78 \cdot 10^{-3}$
η_2	$4.63 \cdot 10^{-3}$
η_3	$7.94 \cdot 10^{-3}$
η_4	$9.54 \cdot 10^{-3}$
η_5	$1.03 \cdot 10^{-2}$
η_6	$1.23 \cdot 10^{-2}$
η_7	$1.26 \cdot 10^{-2}$
η_8	$1.62 \cdot 10^{-2}$
η_9	$1.57 \cdot 10^{-2}$
η_{10}	$4.03 \cdot 10^{-3}$

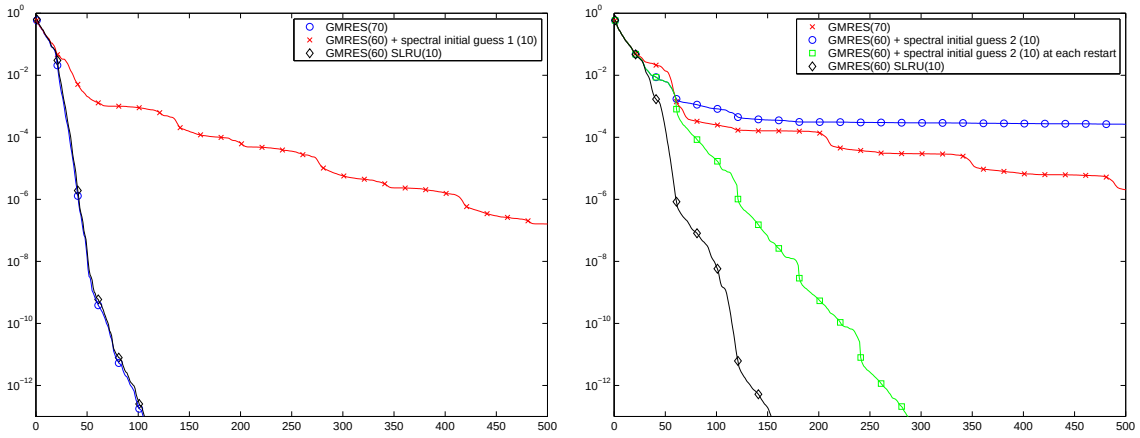
Table 3.32: Backward error of the harmonic Ritz eigenvectors associated with the 10 smallest harmonic Ritz values computed by GMRES–DR(500,10). The method performed 500 iterations (i.e. no restart). The test example is the Airbus 213084 .

(b) inject this information directly in the Krylov space so that we construct the block–Krylov space of $\mathcal{K}(\mathbb{Z}, u_1, u_2, \ldots, u_r, F) = \mathrm{Span}(u_1, u_2, \ldots, u_r, \mathcal{K}(\mathbb{Z}, F))$. This method is known as the GMRES method augmented with eigenvectors or often referred to as the augmented GMRES method.

(c) deflate the initial guess so that its residual has no components on these eigenvectors. If we also know the left eigenvectors $W = (w_1, w_2, \ldots, w_r)$ associated with the smallest eigenvalues, this is achieved by setting $x_0 = UD^{-1}W^TF$ and so $r_0 = (I_m - UW^H)F$. This initial guess strategy is referred to as the *spectral initial guess strategy 1.*
If the left eigenvectors are not known then we consider an alternative strategy, referred to as *the spectral initial guess strategy 2*, defined as follows. Given p vectors $\tilde{W}$ such that $(\tilde{W}^H\mathbb{Z}U)$ is of full rank, the second spectral initial guess strategy computes $x_0 = U(\tilde{W}^H\mathbb{Z}U)^{-1}\tilde{W}^HF$ and the initial residual is given by $r_0 = (I_m - \mathbb{Z}U(\tilde{W}^H\mathbb{Z}U)\tilde{W}^H)F$. The efficiency of this initial guess strategy depends on the choice of W . Classically, we take $\tilde{W} = U$. This choice corresponds to the first spectral initial guess strategy when the matrix is normal ($W = U$).

In Figure 3.42(a), we compare the spectral initial guess strategy with the spectral low rank update preconditioner. GMRES, with the first spectral initial guess strategy, behaves like GMRES preconditioned with the spectral low rank update preconditioner. This behaviour is in agreement with the theory. If the left eigenvectors are known (or the matrix is normal) then the first spectral initial guess strategy is attractive. It performs the deflations of the components only once while the spectral low rank update performs a deflation per iteration. Nevertheless, this requires us to have a very good accuracy on the eigenvectors, otherwise the first spectral initial guess strategy might become less effective while, GMRES preconditioned with the spectral low rank update preconditioner is not much affected [24].
Unfortunately, the left eigenvectors might be difficult to compute in some situations. For instance, in the context of the FMM preconditioned with the Frobenius–norm minimization, the matrix-vector product with the transpose conjugate matrix, $\mathbb{Z}^H$, is not available and cannot be implemented. Consequently, the left eigenvectors

cannot be computed via an Arnoldi process. In this situation, the second initial guess strategy represents an alternative. In Figure 3.42(b), we compare the second spectral initial guess strategy with the spectral low rank update. Because this is no longer a deflation, this strategy should be applied at each restart. Even though it succeeds in improving the convergence of GMRES, it is clearly outperformed by the spectral low rank update preconditioner.



(a) first spectral initial guess strategy (b) second spectral initial guess strategy

Figure 3.42: The convergence of the GMRES method with restart 60 and with spectral initial guess strategies using 10 eigenvectors is compared with the GMRES method with restart 60 and spectral low rank update with 10 eigenvectors. The test example is the CNSPH.

Morgan [93] proposed a variant of GMRES–DR for solving several right–hand sides; this variant is referred to as GMRES–Proj. From the run of GMRES–DR on the first right–hand side, the harmonic Ritz vectors associated with the r smallest harmonic Ritz values, $U_r^{\rm pr}$, are computed and the following relation holds

$$\mathbb{Z}U_r^{\rm pr} = U_{r+1}^{\rm pr}\bar{H}_r^{\rm pr}. \tag{3.15}$$

The GMRES–Proj algorithm implements cycles that alternate GMRES iterations with a projection phase. The projection phase may be performed via either a Galerkin projection or a MINRES projection. The MINRES projection consists in setting d at step 1 of the algorithm so that d solves the least–squares problem $\min \|(U_{r+1}^{\rm pr})^T F - \bar{H}_r^{\rm pr} d\|_2$. The GMRES–Proj with Galerkin projection corresponds to the second spectral initial guess strategy where W is an orthogonal basis of $\mathrm{Span}(U_r^{\rm pr})$. We consider this latter variant in our experiments.

Algorithm 14 GMRES–Proj with Galerkin projection between cycles

1. Solve $H_r^{\rm pr} d = (U_p^{\rm pr})^T F$.
2. The new approximate solution is $x_k = U_p^{\rm pr} d$.
3. The new residual vector is $r_k = r_0 - AU_k^{pr} d = r_0 - U_{r+1}^{\rm pr}\bar{H}_r^{\rm pr} d$.

Finally, we mention that the spectral information that is provided by GMRES–DR (or any other eigensolver) corresponds to approximate eigenvectors and approximate eigenvalues. Accurately computing the eigenvectors may be expensive and we

can wonder about the sensitivity of the convergence behaviour with respect to the accuracy of the computed eigenvectors. In Figure 3.43, we show the convergence histories of augmented GMRES and GMRES using the spectral low rank update preconditioner when the accuracy of the eigenvectors is varied. It can be seen in that figure that GMRES with the spectral low rank update preconditioner is much less sensitive than augmented GMRES to the accuracy of the eigenvectors.

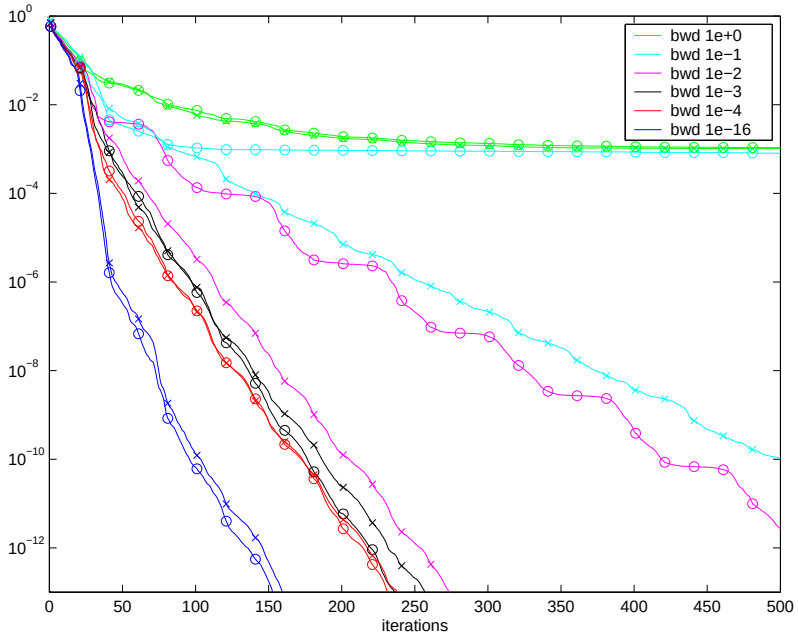


Figure 3.43: Convergence histories of augmented GMRES (10 eigenvectors, restart 40, plotted with a $\circ$) and GMRES with the spectral low rank update (10 eigenvectors, restart 40, plotted with a $\times$) when the accuracy of the eigenvectors is varied. The test example is the CNSPH.

3.8.2 Stopping criterion issue for the RCS calculations

When solving a linear systen is it often very useful to have some knowledge about the underlying application. For instance, it might help

(a) in the selection or design of the preconditioner (the Frobenius preconditioner for instance in our electromagnetism application) or in the selection of the Krylov solver; and

(b) in the stopping criterion to be implemented by the solver.

For instance, when solving the linear systems arising from the discretization of self–adjoint partial differential equations using either finite–element or finite–difference schemes, the matrix A of the linear system is symmetric and positive definite. The A–norm is also often called the energy norm, and the solution we are interested in complies with some optimal criterion in the energy norm. It seems therefore appropriate to use the conjugate gradient method (which minimizes the A–norm of the error) rather than a minimum residual method (which minimizes the 2–norm

of the residual).
In that situation, the choice of the appropriate stopping criterion is also crucial. The reason for having a suitable stopping criterion is to stop the iteration at the *right* time. In the case of the conjugate gradient method, we may want the A–norm of the error to be below a certain threshold provided by the physics and the discretization of the problem [5]. It is then important to use this criterion, that can be easily estimated at each iteration [64, 129], to stop the iterations. Similarly, in the case of non-symmetric problems one can also define an energy norm [7] in which to measure convergence. However, the stopping criterion in the GMRES method is often based on the 2–norm of the residual, this quantiy being easily accessed at each iteration. This may not necessarily be the quantity of interest.
As explained in Section 3.1, our main motivation when solving the linear system $\mathbb{Z}J = F$ is to eventually compute the RCS of the incident wave characterized by F. The currents at the surface of the object, that are represented by the solution J of our system, are post–processed to calculate one point of the RCS curve. In this thesis, the stopping criterion used is the one requested initially by the engineers: the successful solution has a backward error smaller than a given threshold parameter (10^{-2} for the Airbus, $5 \cdot 10^{-3}$ for the cobra, ...). In this section, we illustrate some weakness of this stopping criterion in that framework.
In order to investigate this question, we first try to look at the sensivity of the RCS operator. Given two currents J and $J+\Delta J$, we want to know if the associated RCS are close or not. In Figure 3.44, we give the RCS obtained from the approximate currents given by two different solvers; namely the GMRES method with the stopping criterion threshold 10^{-3}, and the seed–GMRES method with SVD preprocessing and stopping criterion threshold 10^{-3}. In this case, we have $\|\Delta J\|_2 \leq 2 \cdot 10^{-3} \cdot \kappa(\mathbb{Z})$ and the two RCS curves perfectly overlap. This seems to indicate that the RCS operator on that example is well conditioned. This also illustrates that the stopping criterion based on the backward error with a threshold set to 10^{-3} is satisfactory from an engineering point of view; it should be mentioned that for engineers the "quality" of an RCS is a qualitative rather than quantitative quantity.
In Figure 3.45, we plot the relative error of the RCS calculated for each iterate of GMRES for a given incident wave θ. For that example, we consider that the "exact" solution, denoted by x_{ref} is either computed by GMRES with a stopping criterion threshold set to 10^{-5} or is computed by a direct solver. The resulting RCS is denoted by $\text{rcs}_{\text{ref}}(\text{FMM})$ for the one calculated from the solution provided using GMRES-10^{-5}, and $\text{rcs}_{\text{ref}}(\text{no FMM})$ for the one calculated by the direct solver, respectively. The relative errors shown in Figure 3.45 are computed using

$$\frac{|\text{rcs}_n - \text{rcs}_{\text{ref}}|}{|\text{rcs}_{\text{ref}}|},$$

where rcs_n is the RCS computed for each iterate of GMRES. In that figure, it can first be seen that the relative error is not a strictly decreasing function and that the two relative error curves only overlap for the first iterations and then roughly exhibit the same behaviour. The complete RCS of the Airbus is given in Figure 3.5 (see Section 3.1). There it can be seen that the angle $\theta = 30^o, \varphi = 0^o$ does not correspond to a local minimum and in that respect is a "regular" point without

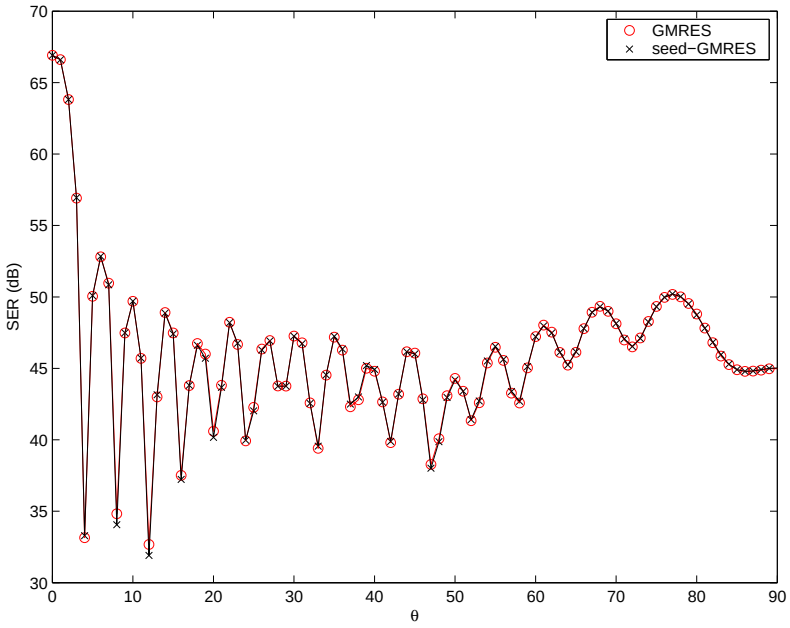


Figure 3.44: RCS computed using two different linear solvers on the cobra 14449, $(\theta = 0^o : 1^o : 90^o, \varphi = 0^o)$. For both solvers the stopping criterion is based on a backward error criterion and the threshold is set to 10^{-3}.

any specificity. Consequently, the same behaviour can be expected for many other points on the RCS curve.
In Figure 3.46, we report on the same experiment performed on the coated cone sphere for the angle $(\theta = 0^o, \varphi = 0^o)$. For that example the "exact" solution is computed using GMRES with the FMM *prec–3* using a backward error threshold equal to 10^{-5}. It can again be observed that the relative error is not a strictly decreasing function and consequently the RCS solution at some early iterates is better than at many other later iterates.
These two experiments illustrate that, if an appropriate stopping criterion based on the RCS existed, it would be possible to stop the iterations of the linear solvers earlier and consequently to save some computing time.
Finally, as mentioned in the introduction of this section, the nature of the underlying problem should also be exploited. So far in this work, we have only unsuccesfully tried to take advantage of the symmetry of the EFIE formulation when we used SQMR. We think that the structure of the matrix $\mathbb{Z}$ and the right–hand sides described in Section 3.2.6 should be further exploited.

3.8.3 Relaxing the matrix–vector accuracy during the convergence

In a series of CERFACS technical reports, Bouras, Frayssé and Giraud [17, 18] experimentally showed that the accuracy of the matrix–vector products can be relaxed during the iterations of Krylov linear solvers. In this context, it is possible to monitor the matrix-vector accuracy and relax it when the convergence proceeds. The model is the following. At step j, the matrix–vector product $w \leftarrow \mathbb{Z}v_j$ is performed

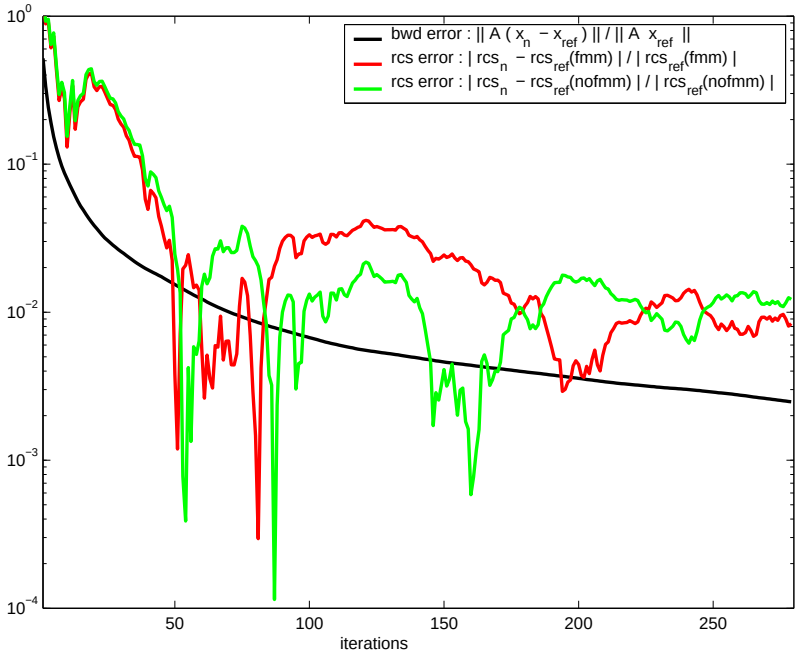


Figure 3.45: Relative error associated with each GMRES iterate for the monostatic RCS at the angle $\theta = 30^o, \varphi = 0^o$. The reference RCS value is given by either the value computed using the solution given by GMRES with the FMM *prec–3* and a backward error equal to 10^{-5} or the value computed using the solution provided by a direct solver. The convergence history of GMRES is also reported. The test example is the Airbus 23676.

with an error g_j

$$w \leftarrow \mathbb{Z}v_j + g_j, \tag{3.15}$$

where

$$\|g_j\|_2 \leq \eta_j \|\mathbb{Z}\|_2 \|v_j\|_2, \tag{3.15}$$

with $\|v_j\|_2 = 1$. When no relaxation is performed, the value of η_j remains constant for all j. For example, if the matrix–vector product (3.8.3) is performed in double precision arithmetic, η_j is of the order of the machine precision. Such a strategy that enables us to obtain the targeted backward error tol has been empirically proposed in [17, 18]. This so called relaxation strategy requires η_j to be bounded above by

$$\eta_j \leq \max\left\{\frac{\text{tol}}{\|r_j\|_2}, \text{tol}\right\}, \tag{3.15}$$

where tol is the threshold stopping criterion based on the backward error, and r_j is the residual of the system at step j.
Two years later and simultaneously, Simoncini and Szyld [124] on the one hand, and van den Eschoff and Sleijpen [132] on the other hand, shed some light on this phenomenon. They give a theoretical framework that attempts to explain why the relaxation strategy (3.8.3) works. Their studies rely on the observation that, due to

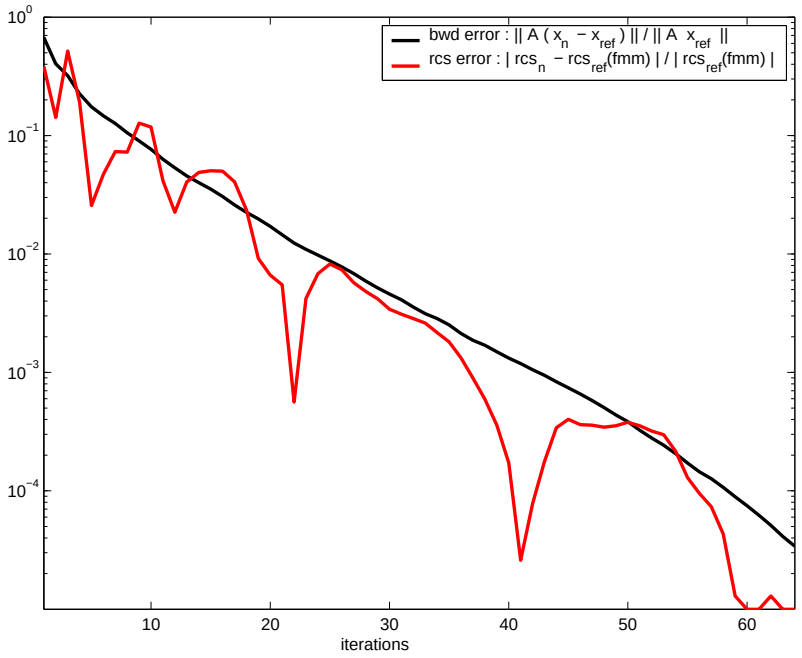


Figure 3.46: Evolution of the error of the monostatic RCS for the right–hand side corresponding to $\theta = 0^o, \varphi = 0^o$ during the iteration of GMRES. The reference point is given by the value obtained from the solution given by GMRES on the FMM *prec–3* (bwd error 10^{-5}). The convergence of the backward error of the approximate solution is also shown. The test case is the coated cone sphere.

equation (3.8.3), the Arnoldi relation at step ℓ becomes

$$\mathbb{Z}V_\ell + E_\ell = V_{\ell+1}\underline{H}_\ell,$$

where

$$E_\ell = (e_1, \ldots, e_\ell)\,. \tag{3.15}$$

Since $V_\ell^H V_\ell = I_\ell$, we have

$$(\mathbb{Z} + E_\ell V_\ell^H)V_\ell = V_{\ell+1}\underline{H}_\ell.$$

Their studies are based on the fact that inexact GMRES reduces to exact GMRES applied to the perturbed linear system

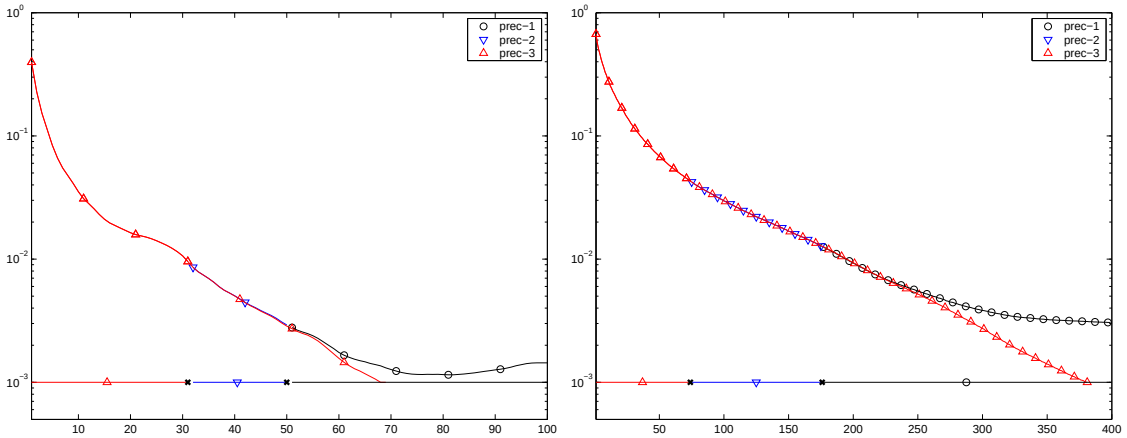
$$\hat{\mathbb{Z}}_\ell J = F,$$

where $\hat{\mathbb{Z}}_\ell = \mathbb{Z} + E_\ell V_\ell^H$.
In the AS_ELFIP code, the three FMM options that correspond to three levels of accuracy are available. From the least accurate to the most, we denote them by *prec–1*, *prec–2* and *prec–3*. When the accuracy is relaxed then we have a faster calculation of the matrix-vector product (see Section 3.3.1 for more details). We intend to solve the system $\mathbb{Z}J = F$ with $\mathbb{Z}$ being the matrix of the FMM *prec–3*. We have experimented the relaxation strategy for the matrix–vector products during

the GMRES iterations. The relaxation strategy chosen at step $(j+1)$ is slightly modified and is defined by

$$\eta_j \leq \max\left\{\frac{\text{tol}}{\omega \cdot \|r_j\|_2}, \text{tol}\right\}.$$

We mention that such a choice has also been considered in the numerical experiments in [124] and for industrial applications using inner-outer iterative solvers in [135], for radiation diffusion problems, and in [90], for the solution of the time-harmonic Maxwell's equations. In our experiments, we arbitrarily set the parameter $\omega = 10$ and $\text{tol} = 10^{-3}$. The numerical experiments are reported in Figure 3.47. We compare the relaxation strategy with the fixed strategy. At each step, the backward error is computed explicitly with the *prec–3* FMM (i.e. the Arnoldi estimate is not used since it holds for the matrix $\hat{\mathbb{Z}}$ and not the system $\mathbb{Z}$). In Figure 3.47, we observe that the relaxation strategy does not degrade the convergence while it enables us to perform faster and faster matrix–vector products. However we do not manage to attain the requested criterion tol; the parameter ω has probably been set too large. Even though some deeper investigations deserve to be developed



(a) with preconditioning, (b) without preconditioning.

Figure 3.47: The relaxation strategy of Bouras, Frayssé and Giraud [18] is tested on GMRES. The test example is the cetaf 5391 for the right–hand side $\theta = 0^o, \varphi = 90^o$.

to better understand the numerical behaviour of the inexact GMRES method, the optimal relaxation strategy as well as the tuning of the FMM parameters at run time, electromagnetism is clearly a topic that can fully take advantage of such numerical techniques.

3.9 Future work

The flexible variant of GMRES has been sucessfully used in the context of the electromagnetism calculations with a single right–hand side. An implementation of a flexible block–GMRES is straightforward and should be studied on this class of problems. Regarding the flexible GMRES, it would be a good thing also to look at even *worse* approximations of the matrix in the inner loop. Also, a full flexible GMRES (i.e. full GMRES in the outer loop) seems affordable and should be investigated on large examples.
Some ideas suggest using the FMM as a preconditioner. A main problem with the Frobenius–norm minimizer is that it is sparse. The use of the FMM as a preconditioner would enable us to use a dense preconditioner at low memory cost and low computational cost each time it is applied.
From a more practical point of view, we think that the storage of the vectors out–of–core is, in most of the cases, a bottleneck for the performance. An implementation with in–core vectors should be preferred. This should increase significantly the efficiency of our multiple right–hand side solvers by reducing the costs of the vector–vector operations. The drawback would be that the largest tractable problem would be smaller with the in–core storage. Nevertheless, for most of the problems, the RAM memory of high performance computers is large enough.
Morgan recently has worked on the block–GMRES–DR method, that, as its name indicates, is a block version of the GMRES–DR algorithm. We strongly believe that this method would be efficient in our case. Restarted block GMRES seems clearly less efficient than the full block GMRES (as it is in the single right–hand side case). However in the block case, the storage of the full approach quickly becomes a bottleneck.

Bibliography

[1] *IEEE Standard for binary Floating–Point Arithmetic, ANSI/IEEE Standard 754–1985*. Institute of Electrical and Electronics Engineers, New–York, 1985.

[2] Nabih N. Abdelmalek. Round off error analysis for Gram–Schmidt method and solution of linear least squares problems. *BIT*, 11:345–368, 1971.

[3] Guillaume Alléon, Sabine Amram, Nicolas Durante, Philippe Homsi, Denis Pogarieloff, and Charbel Farhat. Massively parallel processing boosts the solution of industrial electromagnetic problems: High performance out-of-core solution of complex dense systems. In M. Heath, V. Torczon, G. Astfalk, P. E. Bjørstad, A. H. Karp, C. H. Koebel, V. Kumar, R. F. Lucas, L. T. Watson, and D. E. Womble, editors, *Proceedings of the Eighth SIAM Conference on Parallel*. SIAM Book, Philadelphia, 1997. Conference held in Minneapolis, Minnesota, USA.

[4] Guillaume Alléon, Michele Benzi, and Luc Giraud. Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics. *Numerical Algorithms*, 16:1–15, 1997.

[5] Mario Arioli. A stopping criterion for the conjugate gradient algorithm in a finite element method framework. Technical Report #1179, IAN, 2000. Submitted to Numer. Math.

[6] Mario Arioli, Iain Duff, and Daniel Ruiz. Stopping criteria for iterative solvers. *SIAM Journal on Matrix Analysis and Applications*, 13(1):138–144, January 1992.

[7] Mario Arioli, Daniel Loghin, and Andy J. Wathen. Stopping criteria for iterations in finite element methods. Technical Report TR/PA/03/21, CERFACS, Toulouse, France, 2003.

[8] James Baglama, Daniela Calvetti, Gene H. Golub, and Lothar Reichel. Adaptively preconditioned GMRES algorithms. *SIAM Journal on Scientific Computing*, 20(1):243–269, 1999.

[9] Maurice W. Benson. Iterative solution of large scale linear systems. Master's thesis, Lakehead University, Thunder Bay, Canada, 1973.

[10] Maurice W. Benson and P. O. Frederickson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Utilitas Mathematica*, 22:127–140, 1982.

[11] Maurice W. Benson, J. Krettmann, and M. Wright. Parallel algorithms for the solution of certain large sparse linear systems. *Int J. of Computer Mathematics*, 16, 1984.

[12] David Bindel, Jim Demmel, William Kahan, and Osni Marques. On computing Givens rotations reliably and efficiently. *ACM Transactions on Mathematical Software (TOMS)*, 28(2):206–238, June 2002.

[13] Åke Björck. Solving linear least squares problems by Gram–Schmidt orthogonalization. *BIT*, 7::1–21, 1967.

[14] Åke Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[15] Åke Björck and Christopher C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM Journal on Matrix Analysis and Applications*, 13(1):176–190, 1992.

[16] L. Susan Blackford, Jim Demmel, Jack Dongarra, Iain S. Duff, Sven Hammarling, Greg Henry, Mike Heroux, Linda Kaufman, Andrew Lumsdaine, Antoine Petitet, Roldan Pozo, Karin Remington, and R. Clinton Whaley. An updated set of Basic Linear Algebra Subprograms (BLAS). *ACM Transactions on Mathematical Software (TOMS)*, 28(2):135–151, June 2002.

[17] Amina Bouras and Valérie Frayssé. A relaxation strategy for inexact matrix-vector products for Krylov methods. Technical Report TR/PA/00/15, CERFACS, Toulouse, France, 2000.

[18] Amina Bouras, Valérie Frayssé, and Luc Giraud. A relaxation strategy for inner-outer linear solvers in domain decomposition methods. Technical Report TR/PA/00/17, CERFACS, Toulouse, France, 2000.

[19] Thierry Braconnier, Philippe Langlois, and Jean-Christophe Rioual. The influence of orthogonality on the Arnoldi method. *Linear Algebra and its Applications*, 309(1–3):307–323, April 2000.

[20] Kevin Burrage and Jocelyne Erhel. On the performance of various adaptive preconditioned GMRES strategies. *Numerical Linear Algebra with Applications*, 5(2):101–121, March/April 1998.

[21] Caroline Le Calvez and B. Molina. Implicitly restarted an deflated GMRES. *Numerical Algorithms*, 21:261–285, 1999.

[22] Quentin Carayol. *Développement et analyse d'une méthode multipôle multiniveau pour l'électromagnétisme*. Ph.D. dissertation, Université Paris 6, 2001.

[23] Bruno Carpentieri. *Sparse preconditioners for dense complex linear systems in electromagnetic applications*. Ph.D. dissertation, INPT, April 2002. TH/PA/02/48.

[24] Bruno Carpentieri, Iain S. Duff, and Luc Giraud. A class of spectral two-level preconditioners. *SIAM Journal on Scientific Computing*, x(x):xx–xx, to appear. A preliminary version is available as CERFACS Technical Reports, TR/PA/02/55.

[25] Bruno Carpentieri, Iain S. Duff, Luc Giraud, and Mardochée Magolu monga Made. Sparse symmetric preconditioners for dense linear systems in electromagnetism. *Numerical Linear Algebra with Applications*, xx(x):xx–xx, 2003. A preliminary version is available as CERFACS Technical Reports, TR/PA/01/35.

[26] Bruno Carpentieri, Iain S. Duff, Luc Giraud, and Guillaume Sylvand. Combining fast multipole techniques and an approximate inverse preconditioner for large parallel electromagnetics calculations. Technical Report in preparation, CERFACS, Toulouse, France, 2003.

[27] Luiz M. Carvalho, Luc Giraud, and Patrick Le Tallec. Algebraic two-level preconditioners for the Schur complement method. *SIAM Journal on Scientific Computing*, 22(6):1987–2005, 2001.

[28] Françoise Chaitin-Chatelin and Valérie Frayssé. *Lectures on Finite Precision Computations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. SIAM series Software · Environments · Tools, Editor in chief Jack J. Dongarra.

[29] Françoise Chaitin-Chatelin, Vincent Toumazou, and Elisabeth Traviesas. Accuracy assessment for eigencomputations : variety of backward errors and pseudospectra. *Linear Algebra and its Applications*, 309:73–83, 2000.

[30] Andrew Chapman and Yousef Saad. Deflated and augmented Krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4(1):43–66, January/ February 1997.

[31] Jaeyoung Choi, Jim Demmel, Inderjiit Dhillon, Jack Dongarra, Susan Ostrouchov, Antoine Petitet, Ken Stanley, David Walker, and Clinton Whaley. ScaLAPACK: A portable linear algebra library for distributed memory computers - design issues and performance. *Computer Physics Communications*, 97:1–15, 1996. (also as LAPACK Working Note #95).

[32] Francis Collino and Bruno Després. Integral equations via saddle point problems for time–harmonic Maxwell's equations. *Journal of Computational and Applied Mathematics*, 150:157–192, 2003.

[33] David Colton and Rainer Kress. *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer–Verlag, Berlin Heidelberg New York, 1992.

[34] James W. Daniel, Walter Bill Gragg, Linda Kaufman, and G. W. (Pete) Stewart. Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. *Math. Comp.*, 30(136):772–795, 1976.

[35] Eric Darve. *Méthodes multipôles rapides : résolutions des équations de Maxwell par formulations intégrales*. PhD thesis, Université Paris 6, June 1999.

[36] Eric Darve. The fast multipole method (I) : Error analysis and asymptotic complexity. *SIAM Journal on Numerical Analysis*, 38(1):98–128, 2000.

[37] Eric Darve. The fast multipole method: Numerical implementation. *J. Comp. Phys.*, 160(1):195–240, 2000.

[38] Achiya Dax. A modified Gram–Schmidt algorithm with iterative orthogonalization and column pivoting. *Linear Algebra and its Applications*, 310(1–3):25–42, 2000.

[39] Ben Dembart and Michael A. Epton. A 3D fast multipole method for electromagnetics with multiple levels. Tech. Rep. ISSTECH-97-004, The Boeing Company, Seattle, WA, 1994.

[40] Ben Dembart and Michael A. Epton. Low frequency multipole translation theory for the Helmholtz equation. Tech. Rep. SSGTECH-98-013, The Boeing Company, Seattle, WA, 1998.

[41] Ben Dembart and Michael A. Epton. Spherical harmonic analysis and synthesis for the fast multipole method. Tech. Rep. SSGTECH-98-014, The Boeing Company, Seattle, WA, 1998.

[42] Bruno Després. Quadratic functional and integral equations for harmonic wave problems in exterior domain. *Mathematical Modelling and Numerical Analysis*, 31(6):679–732, 1997.

[43] Jitka Drkošová, Anne Greenbaum, Miroslav Rozložník, and Zdeněk Strakoš. Numerical stability of GMRES. *BIT*, 35(3):309–330, September 1995.

[44] Iain S. Duff, Luc Giraud, Julien Langou, and Émeric Martin. Exploiting spectral informations in large electromagnetic calculation. Tech. Rep. In preparation, CERFACS, Toulouse, France, 2003.

[45] Romain Durdos. Krylov solvers for large symmetric dense complex linear systems in electromagnetism: some numerical experiments. Working Notes WN/PA/02/97, CERFACS, Toulouse, France, 2002.

[46] Jocelyne Erhel, Kevin Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *Journal of Computational and Applied Mathematics*, 69:303–318, 1996.

[47] Jocelyne Erhel, Kevin Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *Journal of Computational and Applied Mathematics*, 69:303–318, 1996.

[48] Ky Fan and Alan J. Hoffman. Some metric inequalities in the space of matrices. *Proc. Amer. Math.*, 6:111–116, 1955.

[49] Jason Frank and C. (Kees)Vuik. Parallel implementation of a multiblock method with approximate subdomain solution. *Appl. Num. Math.*, 30:403–423, 1999.

[50] Valérie Frayssé, Luc Giraud, and Serge Gratton. A set of GMRES routines for real and complex arithmetics. Technical report TR/PA/97/49, CERFACS, Toulouse, France, 1997.

[51] Valérie Frayssé, Luc Giraud, Serge Gratton, and Julien Langou. A set of GMRES routines for real and complex arithmetics on high performance computers. Technical report TR/PA/03/03, CERFACS, Toulouse, France, 2003.

[52] Valérie Frayssé, Luc Giraud, and Hatim Kharraz–Aroussi. On the influence of the orthogonalization scheme on the parallel performance of GMRES. In *Proceedings of EuroPar'98*, volume 1470 of *Lecture Notes in Computer Science*, pages 751–762. Springer–Verlag, 1998. A preliminary version is available as CERFACS Technical Reports, TR/PA/98/07.

[53] Paul O. Frederickson. Fast approximate inversion of large sparse linear systems. Math. Report 7, Lakehead University, Thunder Bay, Canada, 1975.

[54] Roland W. Freund. Conjugate gradient–type methods for linear systems with complex symmetric coefficient matrices. *SIAM Journal on Scientific and Statistical Computing*, 13(1):425–448, January 1992.

[55] Roland W. Freund and Manish Malhotra. A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides. *Linear Algebra and its Applications*, 254(1–3):119–157, March 1997. Proceedings of the Fifth Conference of the International Linear Algebra Society (Atlanta, GA, 1995).

[56] Roland W. Freund and Noël Nachtigal. A new Krylov–subspace method for symmetric indefinite linear systems. Technical Report ORNL/TM-12754, ORNL, Oak Ridge, TN, US, May 1994.

[57] Roland W. Freund and Noël M. Nachtigal. An implementation of the QMR method based on coupled two-term recurrences. *SIAM Journal on Scientific Computing*, 15(2):313–337, 1994.

[58] Roland W. Freund and Noël M. Nachtigal. QMRPACK: a package of QMR algorithms. *ACM Transactions on Mathematical Software (TOMS)*, 22(1):46–77, March 1996.

[59] Walter Gander. Algorithms for the QR decomposition. Research report No. 80-02, Eidgenössische Technische Hochschule, Zürich, 1980.

[60] Walter Gander, Luciano Molinari, and Hana Švecová. In Birkäuser Verlag, editor, *Numerische Prozeduren aus Nachlass und Lehre von Prof. Heinz Rutishauser*, volume 33 of *International Series of Numerical Mathematics*. 1977.

[61] Luc Giraud and Julien Langou. When modified Gram–Schmidt generates a well–conditioned set of vectors. *IMA Journal on Numerical Analysis*, 22(4):521–528, 2002.

[62] Luc Giraud, Julien Langou, and Miroslav Rozložník. On the round-off error analysis of the Gram–Schmidt algorithm with reorthogonalization. Technical Report TR/PA/02/33, CERFACS, Toulouse, France, 2002.

[63] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins Series in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

[64] Gene. H. Golub and Gérard Meurant. Matrices, moments and quadrature II: How to compute the norm of the error in iterative methods. *BIT*, 37:687–705, 1997.

[65] Ananth Grama, Vipin Kumar, and Ahmed Sameh. On n-body simulations using message-passing parallel computers. In Sidney Karin, editor, *Proceedings of the 1995 SIAM Conference on Parallel Processing, San Francisco, CA, USA*, 1995.

[66] Ananth Grama, Vipin Kumar, and Ahmed Sameh. Parallel matrix-vector product using approximate hierarchical methods. In Sidney Karin, editor, *Proceedings of the 1995 ACM/IEEE Supercomputing Conference, December 3–8, 1995, San Diego Convention Center, San Diego, CA, USA*, New York, NY, USA, 1995. ACM Press and IEEE Computer Society Press.

[67] Ananth Grama, Vipin Kumar, and Ahmed Sameh. Scalable parallel formulations of the Barnes–Hut method for n-body simulations. *Parallel Computing*, 24(5–6):797–822, 1998.

[68] Anne Greenbaum, Vlastimil Pták, and Zdeněk Strakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM Journal on Matrix Analysis and Applications*, 17(3):465–469, July 1996.

[69] Anne Greenbaum, Miroslav Rozložník, and Zdeněk Strakoš. Numerical behaviour of the modified Gram–Schmidt GMRES implementation. *BIT*, 37:706–719, 1997.

[70] Leslie Greengard and William Gropp. A parallel version of the fast multipole method. *Comput. Math. Appl.*, 20:63–71, 1990.

[71] Leslie Greengard and Vladimir Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73:325–348, 1987.

[72] Roger F. Harrington. Origin and development of the method of moments for field computation. *IEEE Antennas and Propagation Magazine*, 1990.

[73] Nicholas J. Higham. Matrix nearness problems and applications. In M. J. C. Gover and S. Barnett, editors, *Applications of Matrix Theory*, pages 1–27. Oxford University Press, 1989.

[74] Nicholas J. Higham. Matrix nearness problems and applications. In M. J. C. Gover and S Barnett, editors, *Applications of Matrix Theory*, pages 1–27, Oxford, UK, 1989. Oxford University Press.

[75] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[76] Walter Hoffmann. Iterative algorithms for Gram–Schmidt orthogonalization. Computing, 41:335–348, 1989.

[77] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[78] D. A. H. Jacobs. A generalization of the conjugate–gradient method to solve complex systems. *IMA Journal on Numerical Analysis*, 6:447–452, 1986.

[79] William Jalby and Bernard Philippe. Stability analysis and improvement of the block Gram-Schmidt algorithm. *SIAM Journal on Scientific and Statistical Computing*, 12(5):1058–1073, September 1991.

[80] Pascal Joly and Gérard Meurant. Complex conjugate gradient methods. *Numerical Algorithms*, 4:379–406, 1993.

[81] Serge A. Kharchenko and Alex Yu. Yeremin. Eigenvalue translation based preconditioners for the GMRES(k) method. *Numerical Linear Algebra with Applications*, 2(1):51–77, 1995.

[82] Andrzej Kiełbasiński. Analiza numeryczna algorytmu ortogonalizacji Grama–Schmidta. *Seria III: Matematyka Stosowana II*, pages 15–35, 1974.

[83] Andrzej Kiełbasiński and Hubert Schwetlick. *Numerische Lineare Algebra: Eine Computerorientierte Einführung*. VEB Deutscher, Berlin, 1988.

[84] Andrzej Kiełbasiński and Hubert Schwetlick. *Numeryczna Algebra Liniowa: Wprowadzenie do Obliczeń Zautomatyzowanych*. Wydawnictwa Naukowo–Techniczne, Warszawa, 1992 edition, 1992.

[85] Misha Kilmer, Eric Miller, and Carey Rappaport. QMR-based projection techniques for the solution of non-Hermitian systems with multiple right-hand sides. *SIAM Journal on Scientific Computing*, 23(3):761–780, May 2002.

[86] Ronald Koifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method for the wave equation: a pedestrian prescription. *IEEE Antennas and Propagation Magazine*, 35(3):7–12, 1993.

[87] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995.

[88] Richard B. Lehoucq and Andrew G. Salinger. Large-scale eigenvalue calculations for stability analysis of steady flows on massively parallel computers. *Int. J. Numerical Methods in Fluids*, 36:309–327, 2001.

[89] Per Lötstedt and Martin Nilsson. A minimum residual interpolation method for linear equations with multiple right–hand sides. Technical report 2002-041, Uppsala University, 2002.

[90] Katherine Mer-Nkonga and Francis Collino. The fast multipole method applied to a mixed integral system for time-harmonic Maxwell's equations. In B. Michielsen and F. Decavèle, editors, *European symposium on numerical methods in electromagnetics*, pages 121–126, 2002.

[91] W. C. Mitchell and D. L. McCraith. Heuristic analysis of numerical variants of the Gram Schmidt orthonormalization process. Technical Report TR/CS/122, Computer Science Department, Stanford University, AD 687450, 1969.

[92] Ronald B. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1112–1135, October 2000.

[93] Ronald B. Morgan. GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24(1):20–37, 2002.

[94] Jim M.Varah. A lower bound for the smallest singular value of a matrix. *Linear Algebra and its Applications*, 11:3–5, 1975.

[95] Noël M. Nachtigal, Satish C. Reddy, and Lloyd N. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM Journal on Matrix Analysis and Applications*, 13(3):778–795, July 1992. Iterative methods in numerical linear algebra (Copper Mountain, CO, 1990).

[96] Martin Nilsson. *Iterative Solution of Maxwell's equations in Frequency Domain*. Ph.D. dissertation, UPPSALA University, June 2002.

[97] Dianne P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications*, 29:293–322, 1980.

[98] Michael L. Overton. *Numerical computing with IEEE foating point arithmetic. Including One Theorem, One Rule of Thumb and One Hundred One Exercises*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[99] Christopher C. Paige and Michael A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

[100] Christopher C. Paige and Zdeněk Strakoš. Residual and backward error bounds in minimum residual Krylov subspace methods. *SIAM Journal on Scientific Computing*, 23(6):1899–1924, November 2002.

[101] Beresford N. Parlett. A new look at the lanczos algorithm for solving systems of linear equations. *Linear Algebra and its Applications*, 29:323–346, 1980.

[102] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice–Hall, Englewood Cliffs, NJ, USA, 1980.

[103] Andrew F. Peterson, Scott L. Ray, and Raj Mittra. *Computational Methods for Electromagnetics.* IEEE Press, 1997.

[104] Jean-René Poirier. *Modélisation électromagnétique des effets de rugosité surfacique.* Ph.D. dissertation, Institut National des Sciences Appliquées de Toulouse, December 2000.

[105] Jussi Rahola. Experiments on iterative methods and the fast multipole method in electromagnetic scattering calculations. Technical Report TR/PA/98/49, CERFACS, Toulouse, France, 1998.

[106] Sadasiva M. Rao, Donald R. Wilton, and Allen W. Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Antennas and Propagation Magazine*, 30:409–418, 1982.

[107] Pierre-Arnaud Raviart and Jean-Marie Thomas. A mixed finite element method for 2nd order elliptic problems. In I. Galligani and E. Magenes, editors, *Mathematical aspects of finite element method*, volume 606 of *Lecture Notes in Mathematics*. Springer–Verlag, Berlin, 1975.

[108] Lothar Reichel and William B. Gragg. FORTRAN subroutines for updating the QR decomposition. *ACM Transactions on Mathematical Software (TOMS)*, 16:369–377, 1990.

[109] John R. Rice. Experiments on Gram–Schmidt orthogonalization. *Math. Comp.*, 20:325–328, 1966.

[110] Vladimir Rokhlin. Diagonal forms of translation operators for the Helmholtz equation in three dimensions. Technical report YALEU/DCS/RR-894, Department of Computer Science, Yale University, 1992.

[111] Axel Ruhe. Implementation aspects of band Lánczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Mathematics of Computation*, 33(146):680–687, April 1979.

[112] Axel Ruhe. Numerical aspects of Gram–Schmidt orthogonalization of vectors. *Linear Algebra and its Applications*, 52/53:591–60, 1983.

[113] Heinz Rutishauser. Description of algol 60. In F. L. Bauer, A. S. Householder, F. W. J. Olver, H. Rutishauser, K. Samelson, and E. Stiefel, editors, *Handbook for Automatic Computation*, volume 1, Part a. Springer Verlag, New York Inc., 1967.

[114] Youcef Saad. On the Lanczos method for solving symmetric linear systems with several right–hand sides. *Math. Comp.*, 48:651–662, 1987.

[115] Youcef Saad. Projection and deflation methods for partial pole assignment in linear state feedback. *IEEE Trans. Automat. Contr.*, 33(3):290–297, 1988.

[116] Youcef Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, UK, 1992.

[117] Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, March 1993.

[118] Youcef Saad and Martin H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.

[119] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, MA, US, 1996.

[120] Erhard Schmidt. Über die Auflösung linearer Gleichungen mit unendlich vielen Unbekannten. *Rend. Circ. Mat. Palermo. Ser. 1*, 25:53–77, 1908.

[121] Kevin E. Schmidt and Mickaël A. Lee. Implementing the fast multipole method in three dimensions. *J. Statist. Phys.*, 63:1120, 1991.

[122] John N. Shadid and Ray S. Tuminaro. A comparison of preconditioned nonsymmetric Krylov methods on a large-scale MIMD machine. *SIAM Journal on Scientific Computing*, 14(2):440–459, 1994.

[123] Jérôme Simon. *Extension des Méthodes Multipôles Rapides : Résolution pour des seconds membres multiples et applications aux objets diélectriques*. Ph.D. dissertation, Université de Versailles Saint–Quentin en Yvelines, 2003.

[124] Valeria Simoncini and Daniel B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM Journal on Scientific Computing*, x(x):xx–xx, 2003.

[125] Jaswinder P. Singh, Chris Holt, Takashi Totsuka, Anoop Gupta, and John L. Hennessy. Load Balancing and Data Locality in Adaptive Hierarchical N-body Methods: Barnes-Hut, Fast Multiple, and Radiosity. *Journal of Parallel and Distributed Computing*, 27:118–141, 1995.

[126] Jiming M. Song, Caicheng C. Lu, and Weng Cho Chew. Multilevel fast multipole algorithm for electromagnetic scattering. *IEEE Antennas and Propagation Magazine*, 45:1488–1493, 1997.

[127] Danny C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, 13:357–385, 1992.

[128] Paul Soudais. Iterative solution of a 3–d scattering problem from arbitrary shaped multidielectric and multiconducting bodies. *IEEE Antennas and Propagation Magazine*, 42(7):954–959, 1994.

[129] Zdeněk Strakoš and Petr Tichý. On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electronic Transactions on Numerical Analysis (ETNA)*, 13:56–80, 2002. The original publication is available on link at http://etna.mcs.kent.edu/ ©ETNA.

[130] Guillaume Sylvand. *La Méthode Multipôle Rapide en Electromagnétisme : Performances, Parallélisation, Applications*. Ph.D. dissertation, Ecole Nationale des Ponts et Chaussées, 2002.

[131] Lloyd N. Trefethen. Pseudospectra of matrices. In G. A. Watson D. F. Griffiths, editor, *14th Dundee Biennial Conference on Numerical Analysis*, 1991.

[132] Jasper van den Eshof and Gerard L. G. Sleijpen. Inexact Krylov subspace methods for linear systems. Preprint nr. 1224, Dep. of Mathematics, Utrecht University, The Netherlands, 2002.

[133] Henk A. van der Vorst and C. (Kees) Vuik. The superlinear convergence behaviour of GMRES. *Journal of Computational and Applied Mathematics*, 48:327–341, 1993.

[134] Brigitte Vital. *Étude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multiprocesseur*. Ph.D. dissertation, Université de Rennes, November 1990.

[135] James S. Warsa, Michele Benzi, Todid A. Wareing, and Jim E. Morel. Preconditioning a mixed discontinuous finite element method for radiation diffusion. *Numerical Linear Algebra with Applications*, 2003. To appear.

[136] James H. Wilkinson. *Rounding Errors in Algebraic Processes*. London, 1963.

[137] James H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, UK, 1965, 1965.

[138] Kesheng Wu and Horst Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22(2):602–616, April 2001.

[139] Feng Zhao and S. Lennart Johnsson. The parallel multipole method on the connection machine. *SIAM Journal on Scientific and Statistical Computing*, 12:1420–1437, 1991.