# On the iterative methods for solving linear systems of equations

**Laura C. Dutto**

*Department of Mechanical Engineering*
*Concordia University*
*1455 de Maisonneuve Blvd. W. (C-424)*
*Montréal, Québec*
*Canada H3G 1M8*

ABSTRACT. *The solution of large systems of linear equations is a problem frequently met in numerical calculations, for example, from finite difference or finite element approximations to partial differential equations. The use of direct methods for solving linear systems of equations is limited by both storage requirement and computing time, and in practice, the only alternative is to use iterative algorithms. This paper presents some basic iterative methods, and also a short survey of recent research on this subject, with emphasis on linear systems whose coefficient matrix is non-Hermitian. We briefly discuss the notion of preconditioning, and we give some practical choices for the preconditioning matrix.*

RÉSUMÉ. *La résolution de systèmes d'équations linéaires de grande taille est un problème fréquemment rencontré dans le calcul numérique, par exemple, lors de l'approximation d'équations aux dérivées partielles par des méthodes d'éléments finis ou de différences finies. Les coûts en temps CPU et en place mémoire des méthodes directes de résolution limitent sensiblement leur utilisation en pratique et imposent l'emploi des méthodes itératives comme seule alternative. Le but de cet article est de présenter quelques méthodes itératives bien connues, ainsi que les progrès récents accomplis dans ce domaine. En particulier, nous concentrons notre attention sur la résolution de systèmes linéaires dont la matrice associée est non hermitienne. Nous esquissons la notion de préconditionnement et donnons quelques exemples.*

KEY WORDS : *large and sparse linear systems, non-Hermitian matrices, iterative methods, Petrov-Galerkin methods, Krylov subspace methods, conjugate gradient, like methods, preconditioning.*

MOTS-CLÉS : *systèmes linéaires creux et de grande taille, matrices non hermitiennes, méthodes itératives, méthodes de Petrov-Galerkin, méthodes des sous-espaces de Krylov, méthodes de type gradient conjugué, préconditionnement.*

Revue européenne des éléments finis. Volume 2 - n° 4/1993, pages 423 à 448

## 1. Introduction

One of the fundamental building blocks in the numerical approximation of partial differential equation is the ability to solve linear systems

$$A\mathbf{x} = \mathbf{b}. \qquad [1]$$

We can solve them by direct methods, such as Gaussian elimination. These algorithms give the solution of [1] on the basis of a factorization of the coefficient matrix $A$. If the order of $A$ is $N$, the LU factorization of $A$ needs, asymptotically, $O(N^3/3)$ multiplications, and the solution step, $O(N^2)$ multiplications. Concerning the storage requirements, if we stored $A$ like a dense matrix, we need the same amount of memory to store a copy of this one and of the initial right hand side, i.e. to store $N(N+1)$ coefficients. These estimates are good when *all the coefficients of the matrix are supposed nonzero*; they can be improved for a sparse matrix with a large number of zero entries (see Duff *et al.* (1986)).

In practice, the number of equations can be arbitrarily large; this is particularly true when solving partial differential equations. Fortunately, the resulting systems usually have some special structure, for example, sparsity, i.e. only few nonzero coefficients. Often, direct methods can be adapted to exploit the special structure of the matrix and then remain useful for large linear systems. However, in many cases, especially in three-dimensional calculations, direct approaches are prohibitive both in terms of storage requirements and computing time, and then the only alternative is to use iterative algorithms.

One of the big challenges of the last two decades is to develop robust iterative methods providing the solution for a great number of problems, with a minimal risk of break down. Users of these methods want to know which of them perform best. Unfortunately, this is not yet possible in the general case. A given method will perform better than the others on a particular problem, sometimes by a wide margin, but when applied to other problems, it can be disappointingly slow or may diverge. So, it is very important to know a large range of iterative methods in order to choose the best one for the particular problem we want to solve.

The outline of this paper is as follows. We review in Section 2 some basic iterative methods (see Young (1989)). In Section 3, we present the abstract framework which includes a number of more recently iterative methods, i.e. the Petrov–Galerkin methods (see Saad and Schultz (1985)). We consider Krylov subspace methods (see Saad (1989)), a particular case from the previous class. We recall the oustanding properties of conjugate gradient algorithm (CG) and discuss the issue of optimal extensions of CG to non-Hermitian matrices (see Freund, Golub and Nachtigal (1992)). Some well known algorithms developped to solve non-Hermitian linear systems are compared in Section 4. Finally, in Section 5, we briefly discuss the basic idea of preconditioning, and we present some particular choices for the preconditioning matrix (see Axelsson (1985) and Saad (1989)).

Today, the field of iterative methods is a rich and extremely active research area, and it becomes impossible to cover in a paper all recent advances. In addition to the survey papers mentioned above, we would like to point the reader to the following papers. Stoer (1983) reviews the state of CG–like algorithms up to the early eighties. An annoted bibliography on CG and CG–like methods covering the period up to 1976 was compiled by Golub and O'Leary (1989). Einsenstat *et al.* (1983) and Young and Jea (1980) present CG–like methods, describing some generalizations to nonsymmetrizable basic iterative methods. For numerical comparisons between CG–like methods, see Nachtigal *et al.* (1992). The survey paper of Ortega and Voigt (1985) gives an exhaustive bibliography for research done before 1985 in the general area of solution of partial differential equations on supercomputers. Finally, readers interested in direct methods for sparse linear systems are referred to the book by Duff *et al.* (1986) and, for the efficient use of these techniques on parallel machines, to Heath *et al.* (1991).

## 2. Some basic iterative methods

### 2.1. General study

Let us represent the matrix $A$ in the form

$$A = M - N, \tag{2}$$

where $M$ is a nonsingular, easily inverted matrix. The system [1] can be written:

$$M\mathbf{x} = N\mathbf{x} + \mathbf{b}.$$

The basic iterative method is defined by:

1. Initialization: Let $\mathbf{x}_0$ be the initial solution of [1].
2. For $k = 0, 1, \ldots$ until convergence, do:

$$\mathbf{x}_{k+1} = M^{-1}N\mathbf{x}_k + M^{-1}\mathbf{b}. \tag{3}$$

We can write [3] in some equivalent forms. If we define $G = M^{-1}N$ and $\mathbf{c} = M^{-1}\mathbf{b}$, the $(k+1)$th approximation of the solution defined by [3] could be written:

$$\mathbf{x}_{k+1} = G\mathbf{x}_k + \mathbf{c}. \tag{4}$$

If we denote by $\mathbf{r}_k$ the residual vector at the $k$th step

$$\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k, \tag{5}$$

we have:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + M^{-1}(\mathbf{b} - A\mathbf{x}_k) = \mathbf{x}_k + M^{-1}\mathbf{r}_k. \tag{6}$$

Let $\mathbf{e}_k$ be the error in the approximation of the solution at the $k$th step:

$$\mathbf{e}_k = \hat{\mathbf{x}} - \mathbf{x}_k, \qquad \text{with} \quad \hat{\mathbf{x}} = A^{-1}\mathbf{b} \in \mathcal{C}^N \quad \text{solution of [1].} \qquad [7]$$

By definition of the iterative method [4], $\hat{\mathbf{x}}$ is a fixed point of it ($\hat{\mathbf{x}} = G\hat{\mathbf{x}} + \mathbf{c}$), so we have:

$$\mathbf{e}_{k+1} = G\mathbf{e}_k = G^k\mathbf{e}_0. \qquad [8]$$

$G$ is known as the *iteration matrix.*

Let $\|\cdot\|$ be a given norm in $\mathcal{C}^N$. It is common to consider that the iterative method in use has converged at the $k$th step if

$$\|\mathbf{r}_k\| \leq \epsilon\|\mathbf{r}_0\| \qquad [9]$$

or if

$$\|\mathbf{e}_k\| \leq \epsilon\|\mathbf{e}_0\|, \qquad [10]$$

where $\epsilon$ is a small given parameter. The quantities involving the solution vector $\hat{\mathbf{x}}$ are not easy to handle, so in practice, the condition [10] is usually replaced by

$$\|\mathbf{x}_k - \mathbf{x}_{k-1}\| \leq \epsilon\|\mathbf{x}_k\|. \qquad [11]$$

Nevertheless, it is necessary to be careful with the inequality [11], because it could be satisfied without the iterate $\mathbf{x}_k$ being a good approximation of the solution $\hat{\mathbf{x}}$.

Let us note $\rho(G)$ the spectral radius of $G$:

$$\rho(G) = \max\{|\lambda_i| : \lambda_i \text{ is an eigenvalue of } G\}.$$

We can caracterize the converging basic iterative methods by the following result (see Lascaux and Théodor (1987) §7.2.1, thm. 8).

**Theorem 2.1.** *The iterative method defined by [4] converges to $\hat{\mathbf{x}} = G\hat{\mathbf{x}} + \mathbf{c}$ for each $\mathbf{x}_0 \in \mathcal{C}$ if and only if $\rho(G) < 1$.*

The rate of convergence of a basic iterative method depends on the spectral radius $\rho(G)$ of the iteration matrix $G$. Roughly speaking, from [8], the error is reduced after each iteration by a factor of $\rho(G)$. To reduce the error by a factor of $\epsilon$ we must use $n^*$ iterations, where

$$\Big(\rho(G)\Big)^{n^*} < \epsilon \qquad \Longleftrightarrow \qquad n^* \geq \frac{-\ln\epsilon}{-\ln\rho(G)}.$$

The quantity $-\ln\rho(G)$ is referred to as the *rate of convergence* of the iteration matrix $G$. If we have two iterative methods for the same problem with a similar computation cost per iteration, we will prefer the one with a largest rate of convergence.

## 2.2. Some examples

In this section, we review some well known basic iterative methods, outcoming from the above formulation. Let us represent $A$ in the following forms:

**Case (a)**: $A = D - C_L - C_U$, where $D = (d_{ij}) = (a_{ii}\delta_{ij})$ is a diagonal matrix and $C_L$ and $C_U$ are strictly lower and upper triangular matrices, respectively.
**Case (b)**: $A = H + V$, with $H$ and $V$ two easily invertible matrices.

**Case (a)**: $A = D - C_L - C_U$, where $D = (d_{ij}) = (a_{ii}\delta_{ij})$ is a diagonal matrix and $C_L$ and $C_U$ are strictly lower and upper triangular matrices, respectively.

(i) *Jacobi method or method of simultaneous displacements* (Jacobi (1845))

The method is defined by [3], with the splitting:

$$M = D, \qquad N = C_L + C_U.$$

It is common to denote by $B$ the iteration matrix of the Jacobi method, $B = D^{-1}(C_L + C_U)$.

(ii) *Gauss–Seidel method or method of successive displacements* (Gauss (1823), Seidel (1874))

This method is the same as the Jacobi method, except that one uses new values as soon as available. Thus, we have:

$$M = D - C_L, \qquad N = C_U.$$

**Remark 2.1.** If we reorder the equations and the unknowns by performing the same permutation of indices, the iterates given by the Jacobi method will be the same. For the Gauss–Seidel method, this result is no more true, i.e., some orderings will perform better than others.

(iii) *Successive overrelaxation method* (SOR) (Southwell (1946))

We relax the above methods using a real parameter $\omega$ (Southwell (1946)). If $\tilde{\mathbf{x}}_{k+1}$ is the $(k+1)$-th iterate obtained by one of the previous methods, we define the $(k+1)$-th iterate of the respective relaxation method by:

$$\mathbf{x}_{k+1} = \omega\tilde{\mathbf{x}}_{k+1} + (1-\omega)\mathbf{x}_k. \qquad [12]$$

The relaxed Jacobi method is not really used because it does not improve the Jacobi method itself. On the other hand, the relaxed Gauss–Seidel method is well known as the *successive overrelaxation method* (SOR). The splitting [2] of the matrix $A$ for the SOR method is given by:

$$M = \frac{1}{\omega}D - C_L, \qquad N = \frac{1-\omega}{\omega}D + C_U.$$

The parameter $\omega$ is called a *relaxation factor*. It is current to denote by $\mathcal{L}_\omega$ the iteration matrix of the SOR method, $\mathcal{L}_\omega = M^{-1}N$. If $\omega = 1$ we obtain the iteration matrix of the Gauss–Seidel method, $\mathcal{L}_1$.

Arms, Gates and Zondek (1956) developed *block SOR* methods. For block methods, the unknowns are grouped into blocks and all values within a block

are modified simultaneously using direct methods. Usually, the diagonal blocks are easily invertible. The SOR theory applies for the block SOR method, and faster convergence is obtained than for the point SOR method. Cuthill and Varga (1959) showed how the extra work per iteration for line SOR can be essentially eliminated.

(iv) *Symmetric successive overrelaxation method* (SSOR) (Sheldon (1955))

This method is defined as follows. Given the $k$th iterate $\mathbf{x}_k$, compute $\mathbf{x}_{k+1/2}$ using the SOR method, and then, compute $\mathbf{x}_{k+1}$ based on $\mathbf{x}_{k+1/2}$ using SOR again, taking the equations in reverse order (a backward sweep). The splitting of $A$ is given by:

$$M = \frac{1}{\omega(2-\omega)}(D-\omega C_L)D^{-1}(D-\omega C_U)$$

$$N = \frac{1}{\omega(2-\omega)}(\omega(\omega-1)(D-C_L-C_U)+(1-\omega)D+\omega^2 C_L D^{-1}C_U).$$

The above matrix $M$ is commonly used as a preconditioner.

The convergence properties of the SSOR method have been studied by number of researchers. Habetler and Wachspress (1961) showed that the key quantities for the SSOR method are the largest eigenvalue of the iteration matrix $B$ for the Jacobi method, and the spectral radius $\rho(LU)$ of the matrix $LU$, where $L$ is a strictly lower triangular matrix and $U$ is an upper triangular matrix, such that $B = L + U$. If $\rho(LU) \leq \frac{1}{4} + \epsilon$, where $\epsilon$ is "small", then the SSOR method is effective. For further discussion of the SSOR method, including the use of variable values of $\omega$ between the blocks of points see Axelsson and Barker (1984).

Before starting a summary of some known results, we recall some useful definitions:

- The matrix $A$ is Hermitian if and only if $A = A^*$, where $A^* = \overline{A^T}$ is the complex conjugate of the transpose of $A$, i.e., $(A^*)_{ij} = \overline{a_{ij}}$. This property implies that $(A\mathbf{x}, \mathbf{x}) \in I\!R \quad \forall \mathbf{x} \in \mathbb{C}^N$. In particular, each Hermitian matrix is diagonalizable. Its eigenvalues are real and its eigenvectors are orthogonal.
- The Hermitian matrix $A$ is positive definite if and only if $(A\mathbf{x}, \mathbf{x}) > 0$, $\forall \mathbf{x} \neq \mathbf{0}$. This is equivalent to say that all the eigenvalues of $A$ are strictly positive.
- The matrix $A = (a_{ij})$ is diagonally dominant (DD) if $|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \forall i$; $A$ is strictly DD if $|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \forall i$; $A$ is strongly DD if it is DD and if the previous inequality is verified at least for one row $i$.

Subsequently, the matrix $A$ is always assumed to be nonsingular. For the mentioned basic iterative methods, we can prove the following results (see Lascaux and Théodor (1987)):

- Let $A$ be an Hermitian matrix splitted in the form [2]. Let $M + M^* - A$ be positive definite. Then $\rho(M^{-1}N) < 1 \Leftrightarrow A$ is positive definite.
- Let $A$ be a strictly DD matrix or an irreductible and strongly DD ma-

trix. Then the Jacobi method converges. If $0 < \omega \leq 1$, the SOR method converges (in particular, the Gauss–Seidel method converges).

- Let $A$ be an Hermitian matrix, and let $2D - A$ be positive definite. Then, the point and the block Jacobi methods converge $\Leftrightarrow$ $A$ is positive definite.
- For each matrix $A$, we have $\rho(\mathcal{L}_\omega) \geq |\omega - 1|$, where $\rho(\mathcal{L}_\omega)$ is the spectral radius of the iteration matrix of the SOR method. In particular a necessary condition of convergence for the SOR method is $0 < \omega < 2$.
- Let $A$ be a positive definite Hermitian matrix. Then the SOR method converges $\Leftrightarrow$ $0 < \omega < 2$.
- Let $A$ be an Hermitian matrix splitted in the form $A = D - C_L - C_U$ (case (a)), with $D$ positive definite. Then, the point and the block SOR methods converge (with $0 < \omega < 2$) $\Leftrightarrow$ $A$ is positive definite.
- Let $A$ be a block tridiagonal matrix, with invertible diagonal blocks. Then the block Jacobi and Gauss–Seidel methods converge or diverge simultaneously. If they converge, the block Gauss–Seidel method converge asymptotically two times faster than the block Jacobi method.
- Let $A$ be a block tridiagonal matrix, with invertible diagonal blocks. If each eigenvalue of the iteration matrix $B$ of the Jacobi method is real, then the block Jacobi and SOR methods ($0 < \omega < 2$) converge or diverge simultaneously. If they converge, the optimum value of $\omega$ is given by

$$\omega^* = \frac{2}{1 + \sqrt{1 - \rho^2(B)}}$$

Moreover, the spectral radius of $\mathcal{L}_{\omega^*}$ is given by $\rho(\mathcal{L}_{\omega^*}) = \omega^* - 1$.

**Remark 2.2.** Considerable work has been done on the choice of the optimum $\omega$ in the SOR method for the case where $A$ is not symmetric positive definite, and where the eigenvalues of the matrix $B$ for the Jacobi method are complex. Ehrlich (1981) and others have considered the use of an "ad hoc SOR" procedure for solving linear systems corresponding to partial difference equations. Here $\omega$ is allowed to vary from point to point. The method works well in many cases, in spite of the lack of any rigorous theory.

**Case (b):** $A = H + V$, with $H$ and $V$ two easily invertible matrices.

*Alternating Direction Implicit* (ADI) methods were introduced in two companion papers which appeared in the mid 1950s. One paper by Peaceman and Rachford (1955) was concerned with an alternating direction implicit method for solving linear systems arising from the finite difference solution of elliptic partial differential equations. The other paper by Douglas (1955) was concerned with the solution of parabolic partial differential equations.

Let $\mathbf{x}_0$ be an initial guess for the solution of [1], and let $\mathbf{x}_k$ be the approximation of the solution at the $k$th step. The $(k+1)$th iterates given by the ADI method is defined by:

$$\begin{aligned} (H + \rho I)\mathbf{x}_{k+1/2} &= \mathbf{b} - (V - \rho I)\mathbf{x}_k \\ (V + \rho I)\mathbf{x}_{k+1} &= \mathbf{b} - (H - \rho I)\mathbf{x}_{k+1/2}. \end{aligned} \qquad [13]$$

Here $\rho \in I\!R$ is a parameter to define. Usually, the parameter $\rho$ is allowed to vary from iteration to iteration.

We can write this method in the form [4] defining:

$$G = (V + \rho I)^{-1}(H - \rho I)(H + \rho I)^{-1}(V - \rho I),$$
$$\mathbf{c} = [I - (H - \rho I)(H + \rho I)^{-1}]\mathbf{b}.$$

If $\rho > 0$, we can also write it in the form [3], with

$$M = \frac{1}{2\rho}(H + \rho I)(V + \rho I), \qquad N = \frac{1}{2\rho}(H - \rho I)(V - \rho I), \qquad \text{with} \quad \rho > 0.$$

It is possible to prove the following general result (see Lascaux and Théodor (1987) §7.8.3, thm. 48):

**Theorem 2.2.** *Let $A$ be given by $A = H + V$, where $H$ and $V$ are symmetric positive definite matrices. Then, the alternating direction implicit method defined by [13] converges for all $\rho > 0$. This method still converges if one of the two symmetric matrices $H$ or $V$ is only semi-positive definite.*

It was proved that the key condition is the commutativity condition $HV = VH$. Procedures for choosing the parameters, given the commutativity condition, and also bounds for the eigenvalues of $H$ and $V$ are given by Peaceman and Rachford (1955) and by other authors (see Young (1971)).

Unfortunately, attemps to extend the theory beyond the commutative case have been largely futile. The ADI method can also fail in some cases, when this condition is not verified.

## 3. Petrov–Galerkin methods

In this section, we present the framework of Petrov–Galerkin methods (see Saad and Schultz (1985)), also known as oblique projection methods (see Saad (1982)). We use them on Krylov subspaces, and we specify some common choices for them. We present also the conjugate gradient algorithm, because it is in the core of a number of recently developped iterative methods for solving non-Hermitian linear systems of equations.

### 3.1. Framework of the Petrov–Galerkin methods

Let $(\cdot,\cdot)$ be the euclidean inner–product in $\mathbb{C}^N$ and $\|\cdot\|$ the corresponding norm. If $A$ is Hermitian positive definite of order $N$, let $(\cdot,\cdot)_A$ be the inner-product induced by $A$: $(\cdot,\cdot)_A = (A\cdot,\cdot)$ and let $\|\cdot\|_A$ be the corresponding norm.

Let $\mathbf{x}_0$ be any initial approximate solution of problem [1] and $K_m$ and $L_m$ two subspaces of dimension $m$ of $\mathbb{C}^N$, $m \geq 1$. The Petrov–Galerkin method seeks an approximation of [1] of the form $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{z}_m$, where $\mathbf{z}_m$ belongs to the subspace $K_m$ by imposing the condition that the residual vector $\mathbf{r}_m = \mathbf{b} - A\mathbf{x}_m$

satisfies the Petrov–Galerkin condition, i.e., $\mathbf{r}_m$ is orthogonal to the subspace $L_m$:

$$\text{Find } \mathbf{x}_m = \mathbf{x}_0 + \mathbf{z}_m \text{ with } \mathbf{z}_m \in K_m \ : \ (\mathbf{b} - A\mathbf{x}_m, \mathbf{v}) = 0, \quad \forall \mathbf{v} \in L_m. \quad [14]$$

Clearly, if $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$ is the initial residual vector, the problem [14] is equivalent to:

$$\text{Find } \mathbf{z}_m \in K_m \ : \ (\mathbf{r}_0 - A\mathbf{z}_m, \mathbf{v}) = (\mathbf{r}_m, \mathbf{v}) = 0, \quad \forall \mathbf{v} \in L_m. \quad [15]$$

Suppose now that we have a basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ of $K_m$ and a basis $\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ of $L_m$. We denote by $V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ and $W_m = [\mathbf{w}_1, \ldots, \mathbf{w}_m]$ the matrices whose columns are the vectors of the respective bases. Then, in order to realize the Petrov–Galerkin approximation [15], we may write the unknown $\mathbf{z}_m \in K_m$ in the form:

$$\mathbf{z}_m = V_m \mathbf{y}_m, \qquad \mathbf{y}_m \in \mathbb{C}^m.$$

The problem [15] is then equivalent to:

$$\text{Find } \mathbf{y}_m \in \mathbb{C}^m \ : \ W_m^T(\mathbf{r}_0 - AV_m\mathbf{y}_m) = \mathbf{0}.$$

Assuming $W_m^T A V_m$ is nonsingular, this leads to the solution

$$\begin{aligned} \mathbf{y}_m &= [W_m^T A V_m]^{-1} W_m^T \mathbf{r}_0 \qquad \text{and} \\ \mathbf{x}_m &= \mathbf{x}_0 + V_m \mathbf{y}_m = \mathbf{x}_0 + [W_m^T A V_m]^{-1} W_m^T \mathbf{r}_0. \end{aligned} \quad [16]$$

The above Petrov–Galerkin approximation is well defined if and only if $W_m^T A V_m$ is nonsingular.

There are two important special cases. The first choice $L_m = K_m$ leads to the well known Galerkin method. The second choice $L_m = AK_m$ leads to the least-squares method which finds the approximate solution of [1] of the form $\mathbf{x}_0 + \mathbf{z}_m$, having the smallest possible residual in the euclidean norm. In fact, this observation may be formalized as follows.

**Theorem 3.1.** *Assume $L_m = AK_m$. Then, $\mathbf{x}_m$ is the approximate solution provided by the Petrov-Galerkin method if and only if*

$$\|\mathbf{b} - A\mathbf{x}_m\| = min_{\mathbf{x} \in \mathbf{x}_0 + K_m} \|\mathbf{b} - A\mathbf{x}\|. \quad [17]$$

An interesting question is whether a similar optimality property is also satisfied for the Galerkin method, i.e., when $L_m = K_m$. The answer is known only in the case of a positive definite Hermitian matrix $A$.

**Theorem 3.2.** *Let $A$ be Hermitian positive definite and $L_m = K_m$. The following results are equivalent:*

(i) *$\mathbf{x}_m$ is the approximate solution produced by the Petrov–Galerkin method.*

(ii) $\|\mathbf{b} - A\mathbf{x}_m\|_{A^{-1}} = \min_{\mathbf{x}\in\mathbf{x}_0+K_m}\|\mathbf{b} - A\mathbf{x}\|_{A^{-1}}$.
(iii) $\|\mathbf{x}_m - \hat{\mathbf{x}}\|_A = \min_{\mathbf{x}\in\mathbf{x}_0+K_m}\|\mathbf{x} - \hat{\mathbf{x}}\|_A$, *with* $\hat{\mathbf{x}} = A^{-1}\mathbf{b}$.

Among the projection methods used to solve a linear system of equations, an important class is the *Krylov subspace methods.* These are the Petrov–Galerkin methods where $K_m$ is the Krylov subspace associated with the matrix $A$ and the initial residual $\mathbf{r}_0$:

$$K_m(A, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}. \qquad [18]$$

The different versions of Krylov subspace methods arise from different choices of the subspaces $K_m$ and $L_m$ and from the ways in which the system is preconditioned. The most popular choices of $K_m$ and $L_m$ are the following (see Saad (1989)):

1. $L_m = K_m = K_m(A, \mathbf{r}_0)$. The gradient conjugate method (Hestenes and Stiefel (1952)) is a particular instance of this method when the matrix is Hermitian positive definite. Other methods in this class are the full orthogonalization method (FOM) (Saad (1981)), ORTHORES (Young and Jea (1980)) and ORTHOMIN (Vinsome (1976)), which are mathematically equivalent one to the other, defined to solve non-Hermitian systems.

2. $L_m = AK_m$; $K_m = K_m(A, \mathbf{r}_0)$. The GMRES algorithm, developed by Saad and Schultz (1986) to solve non-Hermitian systems of equations, is a good representative one in this class.

3. $L_m = K_m(A^T, \mathbf{r}_0)$; $K_m = K_m(A, \mathbf{r}_0)$. Clearly, in the Hermitian case this class of methods reduces to the first one. In the non-Hermitian case, the biconjugate gradient method (BCG) due to Lanczos (1952) and Fletcher (1976) is a good representative of this class. More recently, some efficient variations on this method have been proposed, in particular, CGS (Sonneveld (1989)), Bi-CGSTAB (van der Vorst (1992)) and QMR (Freund and Nachtigal (1991)), which we will describe in the next section.

4. $L_m = K_m = K_m(A^TA, A^T\mathbf{r}_0)$. This is nothing but the conjugate gradient method applied to the normal equations

$$A^TA\mathbf{x} = A^T\mathbf{b},$$

often referred to as CGNR (Hestenes and Stiefel (1952)). The condition number of the normal equations is likely to be too large for most problems to make this approach competitive with the previous approaches, except possibly to indefinite problems, i.e., problems for which the symmetric part of the matrix, $(A + A^T)/2$, is not positive definite. We put in this category also the method of conjugate gradient applied to

$$AA^T\mathbf{y} = \mathbf{b},$$

whose solution $\mathbf{y}$ is trivially related to $\mathbf{x}$, solution of system [1], by $\mathbf{x} = A^T\mathbf{y}$. This is often referred to as CGNE, or Craig's method (Craig (1955)). If

we express the Galerkin conditions in terms of the $\mathbf{y}$ variable, then, $K_m = K_m(AA^T, \mathbf{r}_0)$ and $L_m = K_m$. Using the relationship $\mathbf{x} = A^T\mathbf{y}$ between the $\mathbf{x}$ and $\mathbf{y}$ variables, we can verify that for the variable $\mathbf{x}$, Craig's method takes $K_m = K_m(A^TA, A^T\mathbf{r}_0)$ and $L_m = A^{-T}K_m$. Moreover, the main difference between CGNR and CGNE is that the first minimizes the residual norm over $K_m$ while the second minimizes the error norm over $K_m$.

### 3.2. The Conjugate Gradient algorithm

It was developed by Hestenes and Stiefel (1952) for solving Hermitian positive definite linear systems. The works of Reid (1971) and Concus *et al.* (1976) showed its true potential. Since then, a considerable part of the research in numerical linear algebra has been devoted to generalizations of CG to indefinite and non-Hermitian linear systems. The algorithm has a superlinear convergence behaviour (Winther (1980), van der Sluis and van der Vorst (1986)), allowing substantial gains if we compare it with other basic iterative methods to solve linear systems of equations (Section 2). In exact arithmetic, it finds the solution after a finite number of steps, so it is essentially a direct method. Let $A$ be an Hermitian positive definite matrix of dimension $N$. The algorithm can be stated as follows.

1. Initialization: Let $\mathbf{x}_0$ be an initial guess for the solution of [1]. Set $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$; $\mathbf{p}_{-1} = \mathbf{0}$; $\rho_{-1} = 1$; $m = 0$.
2. If $\mathbf{x}_m$ is a good approximation for the solution of [1], stop. Otherwise, compute:

$$\begin{aligned}
\rho_m &= (\mathbf{r}_m, \mathbf{r}_m) \\
\beta_m &= \rho_m/\rho_{m-1} \\
\mathbf{p}_m &= \mathbf{r}_m + \beta_m\mathbf{p}_{m-1} \\
\alpha_m &= \frac{\rho_m}{(A\mathbf{p}_m, \mathbf{p}_m)} \\
\mathbf{r}_{m+1} &= \mathbf{r}_m - \alpha_m A\mathbf{p}_m \\
\mathbf{x}_{m+1} &= \mathbf{x}_m + \alpha_m\mathbf{p}_m
\end{aligned}$$

3. Set $m = m + 1$ and go to 2.

The main results concerning the CG algorithm are summarized in the following theorem (see Lascaux and Théodor (1987) §8.3, thm. 11).

**Theorem 3.3.** *If* $\mathbf{r}_i \neq 0, \quad 0 \leq i \leq m$, *the following relations are verified in the CG method:*

$$(\mathbf{r}_m, \mathbf{p}_i) = 0, \qquad i \leq m-1, \tag{19}$$

$$K_{m+1}(A, \mathbf{r}_0) = span\{\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_m\} = span\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_m\}, \tag{20}$$

$$(\mathbf{p}_m, A\mathbf{p}_i) = (A\mathbf{p}_m, \mathbf{p}_i) = 0, \qquad i \leq m-1, \tag{21}$$

$$(\mathbf{r}_m, \mathbf{r}_i) = 0, \qquad i \leq m-1. \tag{22}$$

The condition $\mathbf{r}_i \neq \mathbf{0}$ from the previous theorem is not restrictive, because if $\mathbf{r}_i = \mathbf{0}$, then $\mathbf{x}_i = \hat{\mathbf{x}}$ is the solution of [1].

Two vectors $\mathbf{u}$ and $\mathbf{v}$ satisfying the condition $(A\mathbf{u}, \mathbf{v}) = 0$ are known as *A-conjugates*. From [21], the vectors $\{\mathbf{p}_i\}_{i=1}^m$ are $A$-conjugates by pairs, i.e., they are orthogonals for the inner product induced by $A$.

From the equality [20], it is clear that the iterate $\mathbf{x}_{m+1}$ given by the CG algorithm is of the form $\mathbf{x}_0 + \mathbf{z}_{m+1}$, with $\mathbf{z}_{m+1} \in K_{m+1}(A, \mathbf{r}_0)$. From [22], the residual vector $\mathbf{r}_{m+1}$ verifies the Petrov–Galerkin condition [15] with $L_{m+1} = K_{m+1}(A, \mathbf{r}_0)$. Thus, the CG algorithm is in class 1 of Krylov subspaces methods as we said below, and it verifies the theorem 3.2.

Because of the previous results, it is clear that the CG algorithm has outstanding properties (see Freund, Golub and Nachtigal (1992)):

- $\mathbf{x}_m$ satisfies a minimization property of the residual vector in the $A^{-1}$-norm (and also a minimization property of the error in the $A$-norm).
- $\mathbf{x}_m$ satisfies a Petrov–Galerkin orthogonality condition for the residual vector.
- $\mathbf{x}_m$ can be computed efficiently, based on simple three-term recurrences.

Thus, an ideal extension of CG to non-Hermitian matrices $A$ would have the following features:

(i) its iterates would be characterized by either a minimization property of the residual vector (MR) or a Petrov–Galerkin orthogonality condition for the residual vector (OR);

(ii) for any initial vector it could be implemented based on short vector recursions, so that work and storage requirements per iteration would be low and roughly constant.

Unfortunately, it turns out that, for general matrices, the conditions (i) and (ii) can not be fulfilled simultaneously:

**Theorem 3.4.** *(Faber and Manteuffel, 1984 and 1987). Except for a few anomalies, ideal CG–like methods that satisfy both requirements (i) and (ii) exist only for matrices of the special form*

$$A = e^{i\theta}(T + \sigma I), \qquad \text{where} \quad T = T^*, \quad \theta \in \mathbb{R}, \quad \sigma \in \mathbb{C}. \qquad [23]$$

The class [23] consists of just the shifted and rotated Hermitian matrices. Note that the important subclass of real nonsymmetric matrices

$$A = I - S, \qquad \text{where} \quad S = -S^T \quad \text{is real},$$

is contained in [23], with $e^{i\theta} = i$, $\sigma = -i$ and $T = iS$. For this class of matrices Freund (1990) has derived practical implementations of the ideal CG–like methods using (MR) and (OR) approaches.

Another special case that arises frequently in applications are complex symmetric matrices $A = A^T$. For example, the complex Helmholtz equations leads to complex symmetric systems. For an overview of CG–type methods and

further results for complex symmetric linear systems, we refer the reader to Freund (1992).

**Remark 3.1.** For non-Hermitian $A$ and even for Hermitian indefinite $A$, an iterate $\mathbf{x}_m$ given by a (OR) property need not exist at each step $m$. In contrast, there is always a $\mathbf{x}_m$ satisfying a (MR) property.

Because of theorem 3.4, clearly, it is not possible to have all the oustanding features of CG algorithm for an ordinary matrix. Thus, it is necessary to make a choice, and this one will be different according to the problem to solve, to the storage capacities of the computer and to the computing time available.

## 4. Non-Hermitian linear systems

In this section we present the principal features of some well known algorithms, defined for solving non-Hermitian linear systems of equations.

We do not intend to be exhaustive here, but only to present some algorithms used as a model or simply, more commonly used than others. The algorithms themselves are not described in detail, only their oustanding features are mentioned. For additional reading we refer to the original paper of each method.

### 4.1. The linear GMRES algorithm

- As mentioned in Section 3, the *Generalized Minimal Residual* (GMRES) algorithm is a Krylov subspace method with $L_m = AK_m$ and $K_m = K_m(A, \mathbf{r}_0)$, proposed by Saad and Schultz (1986).

- It generates an *orthonormal* basis $V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ of $K_m$, adding only one vector at each iteration, with $\mathbf{v}_1 = \mathbf{r}_0/\beta$, $\beta = \|\mathbf{r}_0\|$.

- The iterate $\mathbf{x}_m$ is determined imposing a (MR) condition. From remark 3.1, it is always possible to calculate the next iterate, i.e., $\mathbf{x}_m$ always exists. The point is that from the orthonormality of $V_{m+1}$, the condition (MR) is imposed solving a *m-dimensional* upper Hessenberg system. This one has always a unique solution because, if we have not yet find the exact solution, the matrix $\tilde{H}_m$ (of dimension $(m+1) \times m$) is of full column rank.

$$\begin{aligned}\|\mathbf{r}_0 - A\mathbf{z}_m\| &= \|\mathbf{r}_0 - AV_m\mathbf{y}_m\| = \min_{\mathbf{y}\in\mathbb{C}^m}\|\mathbf{r}_0 - AV_m\mathbf{y}\| \\ &= \min_{\mathbf{y}\in\mathbb{C}^m}\|V_{m+1}(\beta\mathbf{e}_1 - \tilde{H}_m\mathbf{y})\| = \min_{\mathbf{y}\in\mathbb{C}^m}\|\beta\mathbf{e}_1 - \tilde{H}_m\mathbf{y}\|\end{aligned}$$

- GMRES only involves matrix–vector products with $A$.

- It is possible to prove that in exact arithmetic the only possibility to break down is to obtain a residual vector equal to zero, i.e., to find the exact solution.

- Like the conjugate gradient algorithm, if we exclude the influence of rounding errors, the algorithm finds the exact solution in at most $N$ steps. Clearly, if $m = N$, we minimize the residual norm over the whole space $\mathbb{C}^N$!

- Let $\mathcal{P}_m$ be the set of polynomials of degree at most $m$, and $\lambda(A)$ the spectrum of $A$. Thus the convergence result for GMRES reads as follows.

**Theorem 4.1.** *(Saad and Schultz (1986)). Suppose that $A$ is diagonalizable so that $A = XDX^{-1}$ and let*

$$\epsilon^{(m)} = min_{p \in \mathcal{P}_m, p(0)=1} max_{\lambda_i \in \lambda(A)} |p(\lambda_i)|. \qquad [24]$$

*Then, the residual norm provided at the mth step of GMRES satisfies*

$$\|\mathbf{r}_{m+1}\| \leq \kappa(X)\epsilon^{(m)}\|\mathbf{r}_0\|, \qquad [25]$$

*where* $\kappa(X) = \|X\|\|X^{-1}\|$.

- A consequence of theorem 3.4 is that we can not find a short vector recursion to control work and storage requirements per iteration. Here, work and storage requirements increase with the iteration index $m$. To remedy this difficulty, we can use the algorithm iteratively, i.e., we can restart the algorithm every $m$ steps, where $m$ is some fixed integer parameter, using the approached solution $\mathbf{x}_m$ as the initial guess of the solution at the next time. This restarted version of GMRES is denoted by GMRES($m$).

- The GMRES($m$) method converges for all $m \geq 1$, under the condition that $A$ is positive real, as a simple consequence of theorem 4.1. Moreover, if $A$ is nearly positive real, i.e. when it has a small number of eigenvalues on the left half plane, then $m$ need not be too large for convergence of GMRES($m$) to take place. In fact, $m$ depends on the spectrum of $A$. Unfortunately, this result does not extend to indefinite problems. From the (MR) property, GMRES($m$) can not diverge, but it may be stationary.

### 4.2. The BCG algorithm

- The *Bi-Conjugate Gradient* (BCG) algorithm was first introduced by Lanczos (1952), for solving general nonsingular non-Hermitian linear systems. It is a modification of the well known Lanczos algorithm (Lanczos (1950)), originally introduced to approximate the eigenvalues of $A$. Nevertheless, BCG was ignored until the work of Fletcher (1976).

- It belongs to the Krylov subspace methods (Section 3), with $L_m = K_m(A^T, \tilde{\mathbf{r}}_0)$ $K_m = K_m(A, \mathbf{r}_0)$ and $\tilde{\mathbf{r}}_0$ a nonzero vector, nonorthogonal to $\mathbf{r}_0$, i.e., $(\tilde{\mathbf{r}}_0, \mathbf{r}_0) \neq 0$. In particular, we can take $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$. BCG generates iterates defined by a (OR) property. As we pointed out in remark 3.1, the existence of an iterate satisfying a (OR) property *is not guaranteed* at every step, and the algorithm can break down without satisfying the given convergence criterium.

- Unfortunately, BCG typically exhibits a rather erratic convergence behavior with wild oscillations in the residual norm, due to the lack of a (MR) property.

- The algorithm involves matrix–vector products with $A$ and also with $A^T$.

- BCG uses a short vector recursion, which limits work and storage requirements per iteration.

- Like the previous algorithms, in the absence of roundoff, the algorithm performs at most $N$ steps.

- BCG generates basis vectors $V_m = [\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{m-1}]$ of $K_m$ and $W_m = [\tilde{\mathbf{r}}_0, \tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_{m-1}]$ of $L_m$ such that the biorthogonality condition

$$(\tilde{\mathbf{r}}_j, \mathbf{r}_k) = \rho_k \delta_{jk}, \qquad 0 \leq j, k \leq m-1, \qquad [26]$$

holds. This condition could produce a serious breakdown of the algorithm.

### 4.3. The CGS algorithm

- The *Conjugate Gradient Squared* algorithm (CGS), proposed by Sonneveld (1989), is a transpose–free variant of the BCG method; it was developped to eliminate the matrix–vector products involving $A^T$. In fact, if $\mathbf{x}_m^{BCG}$ is the $m$th iterates computed by BCG, with

$$\mathbf{r}_m^{BCG} = \Psi_m^{BCG}(A)\mathbf{r}_0, \qquad \Psi_m^{BCG} \in \mathcal{P}_m,$$

the CGS algorithm gives as a new approximation of the solution the vector $\mathbf{x}_{2m} \in \mathbf{x}_0 + K_{2m}(A, \mathbf{r}_0)$, with

$$\mathbf{r}_{2m} = \mathbf{b} - A\mathbf{x}_{2m} = \left(\Psi_m^{BCG}(A)\right)^2 \mathbf{r}_0. \qquad [27]$$

- As is clear from [27], the erratic convergence behaviour of BCG is magnified in CGS, and CGS typically accelerates convergence as well as divergence of BCG. Moreover, there are cases for which CGS diverges, while BCG still converges.

- Like BCG, the problem of breakdowns is not solved.

### 4.4. The Bi-CGSTAB algorithm

- The *Bi-Conjugate Gradients Stabilized* (Bi-CGSTAB) algorithm was proposed by van der Vorst (1992) like a smoothly converging variant of BCG and CGS algorithms. It does not involve matrix–vector products with $A^T$. Let $\mathbf{x}_m^{BCG}$ be the $m$th iterates produced by BCG, and $\mathbf{r}_m^{BCG}$ the associated residual vector. The iterate $\mathbf{x}_{2m} \in \mathbf{x}_0 + K_{2m}(A, \mathbf{r}_0)$ given by Bi-CGSTAB algorithm verifies

$$\mathbf{r}_{2m} = \mathbf{b} - A\mathbf{x}_{2m} = \Psi_m^{BCG}(A)\chi_m(A)\mathbf{r}_0.$$

Here $\chi_m \in \mathcal{P}_m$, with $\chi_m(0) = 1$, is a polynomial that is updated from step to step by adding a new linear factor

$$\chi_m(\lambda) = (1 - \eta_m \lambda)\chi_{m-1}(\lambda). \qquad [28]$$

The free parameter $\eta_m$ in [28] is determined by a local steepest descent step:

$$\|(I - \eta_m A)\chi_{m-1}(A)\Psi_m^{BCG}(A)\mathbf{r}_0\| = \min_{\eta \in \mathbb{C}} \|(I - \eta A)\chi_{m-1}(A)\Psi_m^{BCG}(A)\mathbf{r}_0\|.$$

- Bi-CGSTAB can break down without a satisfactory approximation of the solution.

### 4.5. The QMR algorithm

- The *Quasi-Minimal Residual* (QMR) algorithm (Freund and Nachtigal (1991)) is again a Krylov subspace method, with $L_m = K_m(A^T, \tilde{\mathbf{r}}_0)$, $K_m = K_m(A, \mathbf{r}_0)$ and $\tilde{\mathbf{r}}_0$ an initial nonzero vector, nonorthogonal to $\mathbf{r}_0$, i.e., $(\tilde{\mathbf{r}}_0, \mathbf{r}_0) \neq 0$. In particular, we can take $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$. It overcomes the two sources of breakdown presented by BCG: the (OR) condition, and also the biorthogonality condition [26] between the basis $V_m$ of $K_m$ and $W_m$ of $L_m$.

- The biorthogonality condition [26] is relaxed when a breakdown is encountered. In this case, we start an internal loop generating the vectors in the basis of $L_m$ and $K_m$ such that they fulfill the biorthogonality condition [26] *by blocks*, i.e.

$$W_{m_j}^T V_{m_k} = \begin{cases} 0 & \text{if } j \neq k \\ D_k & \text{if } j = k, \end{cases} \qquad [29]$$

and $V_m = [V_{m_1}, \ldots, V_{m_l}]$, $W_m = [W_{m_1}, \ldots, W_{m_l}]$.

There are some situations where this procedure would build an infinite block, without never finding a nonsingular $D_k$. Fortunately, in practice, round-off errors will make an incurable breakdown highly unlikely. Moreover, the number of internal vectors, i.e., the size of the blocks, remains small.

- At the $m$th step, the QMR algorithm generates:

$$\mathbf{x}_m = \mathbf{x}_0 + V_m \mathbf{y}_m \in \mathbf{x}_0 + K_m(A, \mathbf{r}_0)$$
$$\mathbf{r}_m = \mathbf{r}_0 - A V_m \mathbf{y}_m = V_{m+1}(\beta \mathbf{e}_1 - H_m \mathbf{y}_m), \qquad \beta = \|\mathbf{r}_0\|.$$

As $V_{m+1}$ is not unitary it is not possible to minimize the euclidean norm of the residual with a near–constant work and storage requirements. Instead, $\mathbf{y}_m$ is chosen solving a quasi–minimization problem for the residual norm, i.e.,

$$\|\beta \mathbf{e}_1 - H_m \mathbf{y}_m\| = \min_{\mathbf{y} \in \mathbb{C}^m} \|\beta \mathbf{e}_1 - H_m \mathbf{y}\|. \qquad [30]$$

The least squares problem [30] has always a unique solution, so the algorithm will not break down because of the (OR) property, as BCG does.

- In addition, the matrix $H_m$ is block tridiagonal (while in GMRES it is an upper Hessenberg matrix). This allows us to compute the iterates using a short vector recursion.

- The quasi–minimization [30] is strong enough to enable us to prove a convergence result similar to theorem 4.1 for GMRES. This is in contrast to BCG and methods derived from BCG, for which no convergence results are known. Without giving the details, the known results are:

**Theorem 4.2.** *(Freund and Nachtigal (1991)). Suppose that the $L \times L$ matrix generated by the QMR algorithm is diagonalizable, and let $X \in \mathbb{C}^{L \times L}$ be a matrix of eigenvectors of $H_L$. If $\epsilon^{(m)}$ is given by [24] and $\kappa(X) = \|X\|\|X^{-1}\|$, for $n = 1, \ldots, L-1$ we have:*

$$\|\mathbf{r}_m^{QMR}\| \leq \|V_{m+1}\| \kappa(X) \epsilon^{(m)} \|\mathbf{r}_0\|. \qquad [31]$$

**Theorem 4.3.** *(Nachtigal (1991))*

$$\|\mathbf{r}_m^{QMR}\| \leq \kappa(V_{m+1})\|\mathbf{r}_m^{GMRES}\|.$$

- The algorithm involves matrix–vector products with $A$ and also with $A^T$.
- In principle, it is always possible to eliminate $A^T$ altogether, by choosing the starting vector $\tilde{\mathbf{r}}_0$ suitably. This observation is based on the fact that any square matrix is similar to its transpose. In particular, there always exists a nonsingular matrix $P$ such that

$$A^T P = PA. \qquad [32]$$

In choosing $\tilde{\mathbf{r}}_0 = P\mathbf{r}_0/\|P\mathbf{r}_0\|$, we can replace the operations involving $A^T$ by the matrix–vector multiplications with $P$. Therefore, this approach is only viable for special classes of matrices $A$, for which one can find a matrix $P$ satisfying [32] easily, and for which, at the same time, matrix–vector products with $P$ can be computed cheaply.

- There are some other transpose–free QMR methods, but in general they are modeled after CGS and they do not address the problem of breakdowns (see Freund, Golub and Nachtigal (1992)).

## 5. Preconditioning techniques for linear systems

For the solution of realistic problems, it is crucial to combine a good iterative method with an efficient preconditioning technique. In this section, we present the basic idea of preconditioning, and also some representative examples. For a careful revision of this subject, the reader could refer to the survey papers of Axelsson (1985) and Saad (1989).

### 5.1. A general study of preconditioning

Let $M$ be a given nonsingular $N \times N$ matrix, which approximates in some sense the coefficient matrix $A$ of the original linear system [1]. Moreover, assume that $M$ is decomposed in the form

$$M = M_1 M_2. \qquad [33]$$

The chosen iterative method of resolution is then used to solve the preconditioned linear system

$$A'\mathbf{x}' = \mathbf{b}', \qquad [34]$$

where

$$A' = M_1^{-1} A M_2^{-1}, \qquad \mathbf{x}' = M_2\mathbf{x} \qquad \text{and} \qquad \mathbf{b}' = M_1^{-1}\mathbf{b}. \qquad [35]$$

Clearly, the systems [1] and [34] are relied by the relations [35], and we can easily obtain $\mathbf{x}$ from $\mathbf{x}'$ (and viceversa) by a change of variables. Usually, one

avoids the explicit calculation of primed quantities, and instead one rewrites the resulting algorithm in terms of the corresponding quantities for the original system. If the matrix $A$ has some special structure, the decomposition [33] can often be chosen such that the structure is preserved for $A'$. For example, if the matrix $A$ is symmetric and positive definite, one can set $M_2 = M_1^T$ in [33]. This allows us to use the same range of iterative methods for solving both the systems [1] and [34], in particular, to use some methods specially adapted to the given structure of the matrix.

We remark that the special cases $M_1 = I$ or $M_2 = I$ on [33] are referred to as *right* or *left preconditioning*, respectively. The general case is known as *preconditioning from both sides.*

Obviously, there are two (in general conflicting) requirements for the choice of the preconditioning matrix $M$ for a given iterative method:

(i) $M_1^{-1}AM_2^{-1}$ should approximate the identity matrix. In other words, the condition number of the matrix $A'$ is smaller than the one of $A$, and the iterative method would present a better convergence behaviour solving [34] than [1].

(ii) The linear systems associated to matrices $M_i$, $i = 1, 2$, can be solved cheaply. Moreover, for algorithms that involve matrix–vector products with $A^T$, the same thing is required to the systems associated to matrices $M_i^T$, $i = 1, 2$.

When the iterative method belongs to the class of Krylov subspace methods (Section 3), the approximate solutions of [34] are of the form

$$\mathbf{x}'_m \in \mathbf{x}'_0 + K_m(A', \mathbf{r}'_0),$$

and the iterates and residual vectors for [1] and [34] are connected by

$$\mathbf{x}_m = M_2^{-1}\mathbf{x}'_m \qquad \text{and} \qquad \mathbf{r}_m = M_1\mathbf{r}'_m. \qquad [36]$$

For right preconditioning, the preconditioned residual vectors coincide, at each step, with their counterparts for the original system. For this reason, right preconditioning is usually preferred for Krylov subspace methods based on a property (MR). Moreover, it is very easy to compare the effect of different preconditioners used for solving a given linear system, because the convergence to zero of the preconditioned residual norm is the same of the residual norm itself.

### 5.2. Some common preconditioners

After the work of Meijerinck and van der Vorst (1977), a widespread preconditioning technique consists in defining $M$ as an incomplete $LU$ factorization of $A$, i.e.

$$M = LU = A + R,$$

where $L$ and $U$ are lower and upper triangular matrices respectively, and where the residual matrix $R$ is "small". We can distinguish two classes of incomplete factorizations:

(i) The zero–nonzero structure of $L + U$ coincides with the one of $A$, i.e., $m_{ij}$ is a nonzero coefficient of $L + U$ if and only if $a_{ij}$ itself is nonzero. This restriction is very advantageous from a programmer's point of view, because it allows us to use the same data structure of $A$ (the associated pointer arrays, for example) for the preconditioning matrix without (or with very few) supplementary work.

(ii) We allow a limited amount of fill-in to take place in the structure of $L + U$. A particular case of this class is when $R = 0$; the matrix $M$ is then a complete factorization of the matrix $A$, and the solution of the problem [34] is obtained in only one step.

Among the preconditioners having the similar structure of matrix $A$ (case (i)), the most popular is ILU(0). In the general sparse case, the incomplete factorization is obtained by performing the standard $LU$ factorization of $A$ and dropping all fill-in elements that are generated during the process (Meijerinck and van der Vorst (1977)). In other words, the imposed condition is:

$$r_{ij} = 0 \qquad \forall \quad (i,j) \quad \text{such that} \quad a_{ij} \neq 0. \tag{37}$$

See Dahl and Wille (1992) for an example of this preconditioner on 3D realistic calculations.

The *modified incomplete factorization* (Gustafsson(1978)) imposes a supplementary condition on the diagonal elements of $M$ at each step of the factorization procedure, that is:

$$\sum_{j=1}^{N} r_{ij} = 0, \qquad i = 1, \ldots, N. \tag{38}$$

Under certain conditions, it is possible to construct modified incomplete factorizations such that the order of convergence of the corresponding preconditioned conjugate gradient method is increased in relation to the use of ILU(0) as a preconditioner.

Axelsson (1986) generalized this idea. He wrote the condition [38] in a most general form

$$R\mathbf{e} = \mathbf{c}. \tag{39}$$

If $\mathbf{e}^T = (1, \ldots, 1) \in I\!R^N$ and $\mathbf{c} = \mathbf{0} \in I\!R^N$, the condition [39] is equivalent to [38] and all the two incomplete factorizations coincide.

Among the preconditioners allowing some fill-in in its matrix structure (case (ii)) we can distinguish three groupes: first, the criteria to calculate the coefficients of the preconditioning matrix is only based in their position. This is a generalization of the condition [37], allowing a larger choice on the index $(i, j)$. On the other hand, the criteria to save a calculation is the size of the coefficient. The typical test imposed here is

$$\text{if} \quad |m_{ij}| \leq \epsilon |m_{ii}| \qquad \Rightarrow \qquad \text{set} \quad m_{ij} = 0,$$

where $\epsilon$ is a prescribed small parameter. Finally, the last class consist of incomplete factorizations choosing their coefficients as a more or less complicated mixing of the two previous techniques.

The first class generalizes, as we have already said, the ILU(0) factorization, introducing the notion of *level of fill-in* (Watts (1981)): the level of a fill-in element is defined as the minimum between its own level and one plus the sum of the levels of the $L$ and $U$ elements from which it is spawned in the elimination process. Initially, all nonzero elements of matrix $A$ have level of fill-in equal to zero, and all the others, infinity. ILU($k$) is then defined as the incomplete factorization that is obtained by dropping all fill-in elements whose level exceeds $k$.

When the size of each element is checked in the factorization process, the resulting preconditioner approximates better, in general, the original matrix than in the previous case. Unfortunately, this process is extremely expensive in computing time, and it is not possible in general to give a priori accurate bounds for the storage requirements. For some applications of these kind of preconditioners to the Navier–Stokes equations, see for example, Bristeau *et al.* (1990), Dutto (1990), and D'Azevedo *et al.* (1992a).

Zlatev (1982) and Saad (1990) propose a preconditioner of the third class, obtained by "mixing" all the two tests mentioned above. Each row of $L$ and $U$ is constructed subject to the restriction that only a small amount of fill-in, for example, $k$ more elements for each one, is allowed beyond the number of elements of $A$ already presents in that row. Furthermore, elements which are deemed to make only an insignificant contribution to the decomposition are also dropped. This technique controls the storage requirements for the preconditioner, and it stays close to the problem to solve. See Freund and Nachtigal (1991) for some numerical results of this kind of preconditioner.

The ordering of the unknowns has a great influence in the efficacy of incomplete factorizations used as preconditioners. For some numerical examples, see Duff and Meurant (1989), D'Azevedo *et al.* (1992b) and Dutto (1993).

Another class of successful preconditioning techniques based on block factorizations was popularized independently by Axelsson, Brinkkemper and Il'in (1984) and by Concus, Golub and Meurant (1985). Given a block tridiagonal matrix $A = \text{tridiag}(B_i^T, A_i, B_{i+1})$, the basic idea is derived from the standard block Gaussian elimination process, in which $A$ is factored as

$$A = (D - L)D^{-1}(D - L)^T,$$

where $L$ is the negative of the strict lower triangular part of $A$, and $D$ is a block diagonal matrix $D = \text{diag}(D_i)$, defined through the recurrence

$$D_i = A_i - B_i D_{i-1}^{-1} B_i^T, \quad i = 2, 3, \ldots, N.$$

Even though the blocks $D_i$ are dense, a sparse approximation to them can easily found, for example, using a banded approximation $\Gamma_i$ to the inverse

$D_i^{-1}$. Numerical experiments on test problems for two dimensions indicate that using good strategies to approach the inverses $D_i^{-1}$, block preconditioning can be more efficient for the same computer storage than other preconditionings, including the corresponding point ones.

As mentioned in Section 2, many of the standard point of line relaxation techniques such as Gauss–Seidel, SOR, SSOR or ADI, can be used as preconditioners.

An interesting kind of preconditioner on vector computers is *polynomial preconditioning*. It consists of choosing a polynomial $s$ and replacing the original linear system by

$$s(A)A\mathbf{x} = s(A)\mathbf{b}. \qquad [40]$$

To be efficient, polynomial preconditioning require the determination of an optimum polynomial $s$. The preconditioned matrix $s(A)A$ should be as close as possible to the identity matrix in some sense. One possible criterion is to make the spectrum of the preconditioned matrix as close as possible from that of the identity. For example, denoting by $\lambda(A)$ the spectrum of $A$ and by $\mathcal{P}_k$ the space of polynomials of degree not exceeding $k$, we may wish to solve

$$\text{Find } s \in \mathcal{P}_k \quad \text{that minimizes} \quad \max_{\lambda\in\lambda(A)}|1 - \lambda s(\lambda)|. \qquad [41]$$

Unfortunately, this problem involves all the eigenvalues of $A$. If we know some continuous set $E$ that encloses $\lambda(A)$, we can replace the problem [41] by:

$$\text{Find } s \in \mathcal{P}_k \quad \text{that minimizes} \quad \max_{\lambda\in E}|1 - \lambda s(\lambda)|. \qquad [42]$$

Thus, it is necessary to construct a region $E$ in the complex plane which ideally would contain the eigenvalues of the matrix $A$. If $A$ is a symmetric positive definite matrix, the set $E$ can be taken to be the interval $[\lambda_{\min}, \lambda_{\max}]$, with $\lambda_{\min}$ and $\lambda_{\max}$ the smallest and the biggest eigenvalues of $A$, respectively. In the more general non-Hermitian case, it is possible to get eigenvalue estimates using a certain number of steps of an iterative algorithm like Lanczos or GMRES, and to use this information for computing the polynomial $s$ (see Elman, Saad and Saylor (1986) and Saad (1989)).

The main attraction of polynomial preconditioning is that the only operations involving the matrix are products with vectors. It can be combined with a subsidiary relaxation–type preconditioning such as SSOR. The main disadvantage of polynomial preconditioning is their poor performance on sequential machines or parallel machines with small number of processors. Axelsson (1985) has shown that polynomial preconditioning is rarely competitive with the nonpreconditioned conjugate gradient method, for a symmetric positive definite matrix. This conclusion is restricted to the simplest polynomial preconditioning, and it is not known whether a similar conclusion might be proved for the more sophisticated preconditioners.

We want also to mention the more recent efforts in deriving preconditioners from domain decomposition techniques. The attraction of this approach is that

they have been implemented naturally in many engineering applications in the past and, as a result, it is possible at present to benefit from this experience. A collection of recent papers on domain decomposition methods can be found in the proceedings edited by Glowinski *et al.* (1988) and Chan *et al.* (1989).

Finally, hierarchical multigrid techniques seem to be promising in the context of finite element discretizations. Briefly, these techniques amount to using hierarchical basis functions, i.e., a basis that consists not only of the nodal functions at the finest grid, but also of the coarser basis functions from which those fine grid functions have been obtained. Thus the function space is identical but its basis has changed. Yserentant (1986) has shown that for two-dimensional problems, the coefficient matrix arising from the discretization of elliptic partial differential equations with such bases has condition number smaller than usual. This alows us to use very simple preconditioners without losing a good performance of the iterative methods used.

## AKNOWLEDGMENTS

I would like to especially thank Professors Wagdi G. Habashi and Rabi Baliga for inviting me to give a conference at CERCA (Montréal, Canada) on this subject. I am also very greateful to Professor W. G. Habashi for his financial support and to Professor Michel Fortin for his advice.

## REFERENCES

[ARM 56] Arms R.J., Gates L.D. and Zondek B., "A method of block iteration", *J. Soc. Indust. Appl. Math.*, **4**, 1956, p. 220–229.

[AXE 85] Axelsson O., "A survey of preconditioned iterative methods for linear systems of algebraic equations", *BIT*, **25**, 1985, p. 166–187.

[AXE 86] Axelsson O., "A general incomplete block–matrix factorization method" *Linear Algebra Appl.*, **74**, 1986, p. 179–190.

[AXE 84] Axelsson O. and Barker V.A., *Finite Element Solution of Boundary Value Problems. Theory and Computation*, Academic Press, New York, 1984.

[ABI 84] Axelsson O., Brinkkemper S. and Il'in V.P., "On some versions of incomplete block-matrix factorization iterative methods", *Linear Algebra Appl.*, **58**, 1984, p. 3–15.

[BRI 90] Bristeau M.-O., Glowinski R., Dutto L., Périaux J. and Rogé G., "Compressible viscous flow calculations using compatible finite element approximations", *Int. J. Numer. Methods in Fluids*, **11**, 1990, p. 719–749.

[CHA 89] Chan T.F., Glowinski R., Périaux J. and Widlund O., eds., *Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, PA, 1989.

[CON 85] Concus P., Golub G.H. and Meurant G., "Block preconditioning for the conjugate gradient method", *SIAM J. Sci. Stat. Comput.*, **6**, 1985, p. 309–332.

[CON 76] Concus P., Golub G.H. and O'Leary D.P., "A generalized conjugate gradient method for the numerical solution of elliptic partial differ-

ential equations", in *Sparse Matrix Computations* (J. R. Bunch and D. J. Rose, eds.), Academic Press, New York (1976), p. 309–332.

[CRA 55] Craig E.J., "The $N$-step iteration procedures", *J. Math. Physics*, **34**, 1955, p. 64–73.

[CUT 59] Cuthill E.H. and Varga R.S., "A method of normalized block iteration", *J. Assoc. Comput. Mach.*, **6**, 1959, p. 236–244.

[DAH 92] Dahl O. and Wille S.Ø., "An ILU preconditioner with coupled node fill-in for iterative solution of the mixed finite element formulation of the 2D and 3D Navier–Stokes equations", *Int. J. Numer. Methods Fluids*, **15**, 1992, p. 525–544.

[DAZ 92a] D'Azevedo E.F., Forsyth P.A. and Tang W.-P., "Towards a cost–effective ILU preconditioner with high level fill", *BIT*, **32**, 1992, p. 442–463.

[DAZ 92b] D'Azevedo E.F.,Forsyth P.A. and Tang W.-P., "Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems", *SIAM J. Mat. Anal. and Appl.*, **13**, 1992, p. 944–961.

[DOU 55] Douglas J. Jr., *J. Soc. Indust. Appl. Math.*, **3**, 1955, p. 42.

[DUF 86] Duff I.S., Erisman A.M. and Reid J.K., *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, 1986.

[DUF 89] Duff I.S. and Meurant G.A., "The effect of ordering on preconditioned conjugate gradients", *BIT*, **29**, 1989, p. 635–657.

[DUT 90] Dutto L.C., Étude de préconditionnements pour la résolution, par la méthode des éléments finis, des équations de Navier–Stokes pour un fluide compressible, thèse de doctorat, université Pierre et Marie Curie (Paris VI), France, 1990.

[DUT 93] Dutto L.C., "The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier–Stokes equations", *Int. J. Numer. Methods Eng.*, **36**, 1993, p. 457–497.

[EHR 81] Ehrlich L.W., "An ad hoc SOR method", *J. Comput. Phys.*, **44**, 1981, p. 31–45.

[EIS 83] Eisentat S.C., Elman H.C. and Schultz M.H., "Variational iterative methods for nonsymmetric systems of linear equations", *SIAM J. Numer. Anal.*, **20**, 1983, p. 345–357.

[ELM 86] Elman H.C., Saad Y. and Saylor P., "A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations", *SIAM J. Sci. Stat. Comput.*, **7**, 1986, p. 840–855.

[FAB 84] Faber V. and Manteuffel T., "Necessary and sufficient conditions for the existence of a conjugate gradient method", *SIAM J. Numer. Anal.*, **21**, 1984, p. 352–362.

[FAB 87] Faber V. and Manteuffel T., "Orthogonal error methods", *SIAM J. Numer. Anal.*, **24**, 1987, p. 170–187.

[FLE 76] Fletcher R., "Conjugate gradient methods for indefinite systems", in *Numerical Analysis Dundee 1975* (G. A. Watson, ed.), Lecture Notes in Mathematics **506**, Springer, Berlin (1976), p. 73–89.

[FRE 90] Freund R.W., "On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices", *Numer. Math.*, **57**, 1990, p. 285–312.

[FRE 92] Freund R.W., "Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices", *SIAM J. Sci. Stat. Comput.*, **13**, 1992, p. 425–448.

[FGN 92] Freund R.W., Golub G.H. and Nachtigal N.M., "Iterative solution of linear systems", *Acta Numerica*, 1992.

[FRE 91] Freund R.W. and Nachtigal N.M., "QMR: a quasi-minimal residual method for non-Hermitian linear systems", *Numer. Math.*, **60**, 1991, p. 315–339.

[GAU 23] Gauss G.F., *Brief an Gerling Werke*, **9**, 1823, p. 278; translated by G. E. Forsythe under the title "Gauss to Gerling on Relaxation", *Math. Tables Aids Comput.*, **5**, 1954, p. 255.

[GLO 88] Glowinski R., Golub G.H., Meurant G.A. and Périaux J., eds., *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1988.

[GOL 89] Golub G.H. and O'Leary D.P., "Some history of de conjugate gradient and Lanczos algorithms: 1948–1976", *SIAM Review*, **31**, 1989, p. 50–102.

[GUS 78] Gustafsson I., "A class of first order factorization methods", *BIT*, **18**, 1978, p. 142–156.

[HAB 61] Habetler G.J. and Wachspress E.L., "Symmetric successive overrelaxation in solving diffusion difference equations", *Math. Comput.*, **15**, 1961, p. 356–362.

[HEA 91] Heath M.T., Ng E. and Peyton B.W., "Parallel algorithms for sparse linear systems", *SIAM Review*, **33**, 1991, p. 420–460.

[HES 52] Hestenes M.R. and Stiefel E., "Methods of conjugate gradients for solving linear systems", *J. Res. Nat. Bur. Stand.*, **49**, 1952, p. 409–436.

[JAC 45] Jacobi C.G.J., *Astr. Nachr.*, **22**, 1845, p. 297.

[LAN 50] Lanczos C., "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators", *J. Res. Math. Bur. Stand.*, **45**, 1950, p. 255–282.

[LAN 52] Lanczos C., "Solution of systems of linear equations by minimized iterations", *J. Res. Math. Bur. Stand.*, **49**, 1952, p. 33–53.

[LAS 87] Lascaux P. and Théodor R., *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, Tome 2, Masson, Paris, 1987.

[MEI 77] Meijerink J.A. and van der Vorst H.A., "An iterative solution method for linear systems of which the coefficient matrix is a symmetrix M-matrix", *Math. Comput.*, **31**, 1977, p. 148–162.

[NAC 91] Nachtigal N.M., A look-ahead variant of the Lanczos algorithm and its application to the quasi-minimal residual method for non-Hermitian linear systems, Ph.D dissertation, Massachusetts Institute of Technology, Cambridge, 1991.

[NAC 92] Nachtigal N.M., Reddy S.C. and Treffethen L.N., "How fast are non-symmetric matrix iterations?", *SIAM J. Matrix Anal. Appl.*, 1992; also in *Num. Anal. Rep.*, **90-2**, MIT, 1990.

[ORT 85] Ortega J.M. and Voigt R.G., "Solution of partial differential equatins on vector and parallel computers", *SIAM Review*, **27**, 1985, p. 149–240.

[PEA 55] Peaceman D.W. and Rachford H.H. Jr., "The numerical solution of parabolic and elliptic differential equations", *J. Soc. Indust. Appl. Math.*, **3**, 1955, p. 28–41.

[REI 71] Reid J.K., "On the method of conjugate gradients for the solution of large sparse systems of linear equations", in *Large Sparse Sets of Linear Equations* (J. K. Reid, ed.), Academic Press, New York (1971), p. 231–253.

[SAA 81] Saad Y., "Krylov subspace methods for solving large unsymmetric linear systems", *Math. Comput.*, **37**, 1981, p. 105–126.

[SAA 82] Saad Y., "The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems", *SIAM J. Numer. Anal.*, **19**, 1982, p. 485–506.

[SAA 89] Saad Y., "Krylov subspace methods on supercomputers", *SIAM J. Sci. Stat. Comput.*, **10**, 1989, p. 1200–1232.

[SAA 90] Saad Y., "SPARSKIT: a basic tool kit for sparse matrix computations", *Technical Report* **90.20**, RIACS, NASA Ames Research Center, 1990.

[SAA 85] Saad Y. and Schultz M.H., "Conjugate gradient-like algorithms for solving nonsymmetric linear systems", *Math. Comput.*, **44**, 1985, p. 417–424.

[SAA 86] Saad Y. and Schultz M.H., "GMRES: a Generalized Minimal Residual algorithm for solving nonsymmetric linear systems", *SIAM J. Sci. Stat. Comput.*, **7**, 1986, p. 856–869.

[SEI 74] Seidel L., *Abh. Math.-Phys. Kl. Bayerische Akad. Wiss. München*, **11** (III), 1874, p. 81.

[SHE 55] Sheldon J., *Math. Tables Aids Comput.*, **9**, 1955, p. 101.

[SON 89] Sonneveld P., "CGS, a fast Lanczos type solver for nonsymmetric linear systems", *SIAM J. Sci. Stat. Comput.*, **10**, 1989, p. 36–52.

[SOU 46] Southwell R.V., *Relaxation Methods in Theoretical Physics*, Oxford Univ. Press, New York, 1946.

[STO 83] Stoer J., "Solution of large systems of equations by conjugate gradient type methods", in *Mathematical Programming - The State of the Art* (A. Bachem, M. Grötschel and B. Korte, eds.), Springer, Berlin (1983), p. 540–565.

[VAN 86] van der Sluis A. and van der Vorst H.A., "The rate of convergence of conjugate gradients", *Numer. Math.*, **48**, 1986, p. 543–560.

[VAN 92] van der Vorst H.A., "Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems", *SIAM J. Sci. Stat. Comput.*, **13**, 1992, p. 631–644.

[VIN 76] Vinsome P.K.W., "Orthomin, an iterative method for solving sparse sets of simultaneous linear equations", *Proceeding of the Fourth Sym-*

*posium on Reservoir Simulation*, Society of Petroleum Engineers of AIME (1976), p. 149–159.

[WAT 81] Watts J.W. III, "A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation presure equation", *Society of Petrolum Engineer J.*, **21**, 1981, p. 345–353.

[WIN 80] Winther R., "Some superlinear convergence results for the conjugate gradient method' *SIAM J. Numer. Anal.*, **17**, 1980, p. 14–17.

[YOU 71] Young D.M., *Iterarive Solution of Large Linear Systems*, Academic Press, New York, 1971.

[YOU 89] Young D.M., "A historical overview of iterarive methods", *Comp. Phys. Comm.*, **53**, 1989, p. 1–17.

[YOU 80] Young D.M. and Jea K.C., "Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods", *Linear Algebra Appl.*, **34**, 1980, p. 159–194.

[YSE 86] Yserentant H., "On the multi-level splitting of finite element spaces", *Numer. Math.*, **49**, 1986, p. 379–412.

[ZLA 82] Zlatev Z., "Use of iterative refinement in the solution of sparse linear systems", *SIAM J. Numer. Anal.*, **19**, 1982, p. 381–399.