Resolution of sparse linear systems of equations: the RPK strategy

G .Montero, R.Montenegro, J.M.Escobar and E.Rodríguez

Institute of Intelligent Systems and Numerical Applications in Engineering
University of LasPalmas de Gran Canaria

Keywords: Sparse Linear systems of equations, iterative solvers, Krylov subspace methods,preconditioning,reordering.

Abstract

An over view of advanced techniques for solving large sparse linear systems of equations is presented. First, several reordering algorithms are introduced in order to improve the effect of preconditioning on a linear system. Next, we define the concept of preconditioning and formulate some of most popular preconditioners, especially those based in approximate inverse. On the other hand, some Krylov subspace methods for solving linear systems of equations are considered. For symmetric problems, the Conjugate Gradient method is proposed. However, for non-symmetric linear systems there exist several alternatives that may be classified into three family of methods: orthogonalisation, biorthogonalisation and normal equation methods. Nowadays, RPK strategy which combines those three techniques, reordering, preconditioning and Krylov subspace methods, seems to be the most efficient from the computational point of view. This is finally illustrated with some numerical experiments.

Keywords: Linear systems of equations, iterative methods, reordering, preconditioning, Krylov subspace methods.

1 Introduction

We are interested in the resolution of a linear system of equation,

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

where A is a *sparse*, large and non-singular matrix. The first question is if it is better a direct or an iterative resolution. The main disadvantage of direct methods compared with iterative ones is that the rounding errors are accumulated along the process of direct solving. Besides they require more memory requirements due to the fill-in effect. On the other hand, in non steady problems where there must be solved many

similar systems of equations, iterative solvers may use the solution obtained in the previous time step as initial guess. So, nowadays it is preferred to use iterative methods in front of direct ones for solving large scale sparse linear systems of equations.

The reordering techniques based on graph theory, that were initially applied in the resolution by using direct methods, provide matrices with smaller band width or a sparsity pattern with a lower number of nonzero inner entries. However, this reduction may be used in order to improve the effect of incomplete factorisation preconditioners on the rate of convergence of iterative methods. The effect several reordering techniques on different Krylov subspace methods may be seen in [1, 2, 3, 4, 5, 6, 7, 8].

Preconditioning techniques improve the convergence of iterative methods. Here, we study some standard preconditioners, in particular, Jacobi, SSOR, ILU and sparse approximate inverse. Other type of preconditioners may be found, e.g., in [9, 10, 11].

In addition, we study the three groups of Krylov subspace methods (see [12, 13]): orthogonalisation, biorthogonalisation and normal equation methods. For the case of symmetric linear systems of equations, the use of Conjugate Gradient method [14] is generally accepted as the best choice. It is based on the Lanczos orthogonalisation method which is a simplification of Arnoldi algorithm for symmetric linear systems

Among orthogonalisation methods for nonsymmetric linear systems that apply the Arnoldi algorithm [15], we study the Generalised Minimal Residual method (GMRES) [16]. Several versions of the latter may found in [17, 18, 19, 20, 21].

On the other hand, we study the Biconjugate Gradient Stabilised method (Bi-CGSTAB) [22] and its quasi-minimal residual version, the QMRCGSTAB algorithm [23].

Finally, we show some results from using the Least-square QR method (LSQR) [24] based on the normal equation.

2 Reordering

Reordering techniques were basically applied in the resolution of linear systems of equations by direct solvers. They are based on graph theory and produce matrices with a lower bandwidth, what reduces the fill-in effect due to factorisation. In addition, they do not affect the storage requirements of a matrix since the number of nonzero entries stands although they occupy different locations. Here, the object of applying these techniques before the above algorithms is to obtain incomplete factorisations closer to complete ones in order to improve the preconditioning. Another advantage of reordering is related to the use of sparse approximate inverse preconditioners (SPAI), of which it may reduce the amount of entries and improve their effect on the convergence of Krylov solvers [4]. In particular, the Minimum Degree algorithm (MDG, George and Liu [25]), Reverse Cuthill-Mckee algorithm (RCM, [26]) proposed by George in order to improve that proposed by Cuthill-Mckee [27], and the Minimum Neighboring algorithm (a variant of MDG, see, e.g., [3]) have been used in the numerical experiments of this paper.

2.1 Minimum Degree algorithm

This algorithm is used for matrices with symmetric sparsity pattern. It consists of developing a node numbering in an increasing order of their respective degrees (node connections in the graph). Nodes with higher degrees will produce a higher fill-in. Thus, the idea is to reorder them to the end in order to reduce the fill-in during the procedure.

MDG ALGORITHM

1 - Construct the graph related to matrix A, $g(x) = \langle V, E \rangle$, where V is the set of nodes and $E = \{\{a, b\} : a \neq b \ / \ a, b \in V\}$

2 - While $V \neq \emptyset$

2.1- Choose a node v of minimum degree in $g(x) = \langle V, E \rangle$ and reorder it as next node.

2.2 - Define

$V_v = V - \{v\}$,

$E_v = \{\{a, b\} \in E \ / \ a, b \in V_v\} \cup \{\{a, b\} \, a \neq b \ / \ a, b \in Adj_g(v)\}$

being, $Adj_g(v)$ the set of nodes connected to v in the graph $g(x)$ and do

$V = V_v, \ \ E = E_v$ y $g(x) = \langle V, E \rangle$

3 - End

2.2 Reverse Cuthill-Mckee algorithm

Cuthill-Mckee algorithm provides a simple way for reordering a sparse matrix in order to reduce the fill-in effect by transforming it in a band matrix. However, the reverse ordering of Cuthill-Mckee often results better that the former since it reduce the profile of the matrix for the same bandwidth.

RCM ALGORITHM

1 - Construct the graph related to matrix A, $g(x) = \langle V, E \rangle$, being V the set of nodes and $E = \{\{a, b\} : a \neq b \ / \ a, b \in V\}$

2 - Choose a node v of minimum degree in $g(x) = \langle V, E \rangle$

and reorder it as next node

2.1 - Define

$V = V - \{v\}$,

3 - While $V \neq \emptyset$

3.1 - Reorder all the nodes in $Adj_g(v)$ by increasing degree,

being, $Adj_g(v)$ the set of nodes connected to v in the graph $g(x)$

3.2 - Define

$V_v = V - \{a\}\,, \forall a \in Adj_g(v)$

and do

$V = V_v$, and $g(x) = \langle V, E\rangle$

3.3 - Set v as the next node of the reordered set of nodes

4 - Perform the reverse ordering

5 - End

2.3 Minimum Neighboring algorithm

This algorithm [25] is a variant of the Minimum Degree that works by eliminating the selected nodes from the graph, such that there is not defined nor inserted any new connection in the graph. The nodes with the lowest number of neighbours are selected consecutively. This algorithm is specially useful when we construct an incomplete factorisation with the same sparsity pattern as the matrix of the system, e.g., ILU(0) and SSOR preconditioners.

MN ALGORITHM

1 - Construct the graph related to matrix A, $g(x) = \langle V, E\rangle$, where V is the set of nodes and $E = \{\{a, b\} : a \neq b \ / \ a, b \in V\}$

2 - While $V \neq \emptyset$

2.1- Choose a node v of minimum degree in $g(x) = \langle V, E\rangle$

and reorder it as next node

2.2 - Define

$V_v = V - \{v\}\,, \ \ E_v = \{\{a, b\} \in E \ / \ a, b \in V_v\}$

and do

$V = V_v, \ \ E = E_v$ and $g(x) = \langle V, E\rangle$

3 - End

2.4 George's algorithm

The selection of the initial node in the above algorithms may be carried out by applying the George's algorithm [25] for searching a pseudo-peripheral node.

If we define the distance $d(x, y)$ between two nodes x and y in a graph $g(x) = \langle V, E\rangle$, as the length of the shortest trajectory which joins both nodes, and the eccentricity of a node x as $\varepsilon(x) = Max\,\{d(x, y)/x, y \in V\}$, the algorithm may be written as follows,

GEORGE'S ALGORITHM FOR SEARCHING PSEUDO-PERIPHERAL NODES

1- Choose any node r of V

2 . Generate a structure with levels rooted in r,

$$\left\{L_0(r), L_1(r), \ldots, L_{\varepsilon(r)}(r)\right\}.$$

being $L_i(r) = \{x/d(x,r) = i\}$

3 - Choose a node x of minimum degree in $L_{\varepsilon(r)}(r)$

4 . Generate a structure with levels rooted in x,

$$\left\{L_0(x), L_1(x), \ldots, L_{\varepsilon(x)}(x)\right\}$$

5 - If $\varepsilon(x) > \varepsilon(r)$, set $x \rightarrow r$ and return to step 3

6 - Else set x as initial node

7 - End

3 Preconditioning

The rate of convergence of Krylov subspace methods may be increased with the use of preconditioning techniques. They consist of replacing the original system of equations $Ax = b$ by another with identical solution, in such a way that the condition number of the matrix of the new system is lower that that of A.

The preconditioning may be carried out in three different ways,

$$\begin{array}{ll} M^{-1}Ax = M^{-1}b & \text{(Left preconditioning)} \\ AM^{-1}Mx = b & \text{(Right preconditioning)} \\ M_1^{-1}AM_2^{-1}M_2x = M_1^{-1}b & \text{(Both side preconditioning)} \end{array} \tag{2}$$

if M is factorised as $M = M_1M_2$.

A great number of preconditioners have been developed and widely used in several application fields, of which Jacobi, SSOR, ILU, SPAI and Optimal Diagonal are considered here.

3.1 Jacobi preconditioner

It can be derived from the comparison of Richardson iteration in the preconditioned system with Jacobi iteration in the unpreconditioned system. The application of Richardson iteration to the preconditioned system $M^{-1}Ax = M^{-1}b$ yields

$$Mx_{i+1} = Mx_i + \alpha\left(b - Ax_i\right) \tag{3}$$

On the other hand, the Jacobi iteration may be written as,

$$Dx_{i+1} = Dx_i + \left(b - Ax_i\right) \tag{4}$$

Thus, comparing equations (3) and (4), we notice that Jacobi iteration applied to the unpreconditioned system is equivalent to Richardson iteration, with $\alpha = 1$, applied to the preconditioned system using the diagonal of matrix A as preconditioner ($M = diag(A)$).

3.2 SSOR preconditioner

Similarly, if we apply the SSOR(ω) iteration to the unpreconditioned system, we have

$$\begin{aligned} & \tfrac{1}{\omega(2-\omega)} \quad (D-\omega E)\, D^{-1}\, (D-\omega F)\, x_{i+1} \\ = \; & \tfrac{1}{\omega(2-\omega)} \quad (D-\omega E)\, D^{-1}\, (D-\omega F)\, x_i + (b - Ax_i) \end{aligned} \tag{5}$$

resulting as preconditioner matrix,

$$M = \left(I - \omega E D^{-1}\right) \left(\frac{D-\omega F}{\omega(2-\omega)}\right) \tag{6}$$

When A is symmetric, it may be expressed as,

$$M = \left[\frac{(D-\omega E)\, D^{-1/2}}{\sqrt{\omega\,(2-\omega)}}\right] \left[\frac{(D-\omega E)\, D^{-1/2}}{\sqrt{\omega\,(2-\omega)}}\right]^T \tag{7}$$

3.3 ILU(0) preconditioner

It results from an incomplete LU factorisation of A, keeping the same zero entries in triangular matrices L and U,

$$A = LU \approx ILU(0) = M \tag{8}$$

where m_{ij} are the entries of M such that,

$$m_{ij} = 0 \qquad \text{if} \qquad a_{ij} = 0 \tag{9}$$

$$\{A - LU\}_{ij} = 0 \qquad \text{if} \qquad a_{ij} \neq 0 \tag{10}$$

If A is symmetric, the incomplete factorisation of Cholesky ILL$^t(0)$ may obtained instead.

3.4 Sparse approximate inverse

Nowadays, the use of the sparse approximate inverse preconditioner has become a good alternative to implicit preconditioners due to the possibilities of parallelisation of the former. For more details see [28, 29] and also a complete study comparing both type of preconditioners in [30]. Here, we will construct an approximate inverse using

the Frobenius inner product. Although we have used right preconditioning, the results with left preconditioning are straightforward.

Let $\mathcal{S} \subset \mathcal{M}_n$ be the subspace of matrices M where we search an explicit approximate inverse with an unknown sparsity pattern. The problem consists of finding $M_0 \in \mathcal{S}$ such that

$$M_0 = \underset{M \in \mathcal{S}}{\arg\min} \|AM - I\|_F \tag{11}$$

In addition, this initial matrix may be an approximate inverse of A strictly, i.e.,

$$\|AM_0 - I\|_F = \varepsilon < 1 \tag{12}$$

There are two reasons for this requirement. On the one hand, equation (12) allows to ensure that M_0 in non singular (Banach's lemma). On the other hand, equation (12) is the basis for constructing an explicit algorithm which allows to improve M_0 and solve (1).

In [29] Grote et al propose an efficient algorithm to obtain an approximate inverse as close to the inverse of a non singular matrix A as required. We have followed such technique but changing the selection method of entries in M_0 and the algorithm for solving problem (11). The construction of M_0 is carried out in parallel, computing each column independently. Although our algorithm allows to start from any entry of column k, it is commonly accepted the use of the diagonal as first approximation. In addition, the expression of the optimal diagonal preconditioner is well known. So, the next entry to be considered is selected in the set of candidates which is defined following the strategy proposed in [29].

Let r_k be the residual corresponding to column $k-th$,

$$r_k = Am_k - e_k \tag{13}$$

and let $\mathcal{I}_k$ be the set of indices of non zero entries in r_k, i.e., $\mathcal{I}_k = \{i \in \{1, 2, ..., n\} \ / \ r_{ik} \neq 0\}$. Si $\mathcal{L}_k = \{l \in \{1, 2, ..., n\} \ / \ m_{lk} \neq 0\}$, then the new entry is searched in the set $\mathcal{J}_k = \{j \in \mathcal{L}_k^c \ / \ a_{ij} \neq 0, \forall i \in \mathcal{I}_k\}$. Really, the only entries that are considered in m_k are those which involves the non zero entries of r_k. In the following, we assume that $\mathcal{L}_k \cup \{j\} = \left\{i_1^k, i_2^k, ..., i_{p_k}^k\right\}$ is not empty, being p_k the updated number of non zero entries of m_k, and $i_{p_k}^k = j$, for all $j \in \mathcal{J}_k$. For each j, we compute,

$$||Am_k - e_k||_2^2 = 1 - \sum_{l=1}^{p_k} \frac{\left[\det\left(D_l^k\right)\right]^2}{\det\left(G_{l-1}^k\right) \det\left(G_l^k\right)} \tag{14}$$

where, for all k, $\det\left(G_0^k\right) = 1$ and G_l^k is the Gram matrix of columns $i_1^k, i_2^k, ..., i_l^k$ of matrix A related to Euclidean inner product, D_l^k is the matrix which results from replacing the last row of matrix G_l^k by $a_{k\,i_1^k}, a_{k\,i_2^k}, ..., a_{k\,i_l^k}$, with $1 \leq l \leq p_k$. Then, the index j_k that minimises $\|Am_k - e_k\|_2$ is selected. This strategy (see [31]) defines the new selected index j_k only attending to the set $\mathcal{L}_k$, what leads us to new optimum where all the entries corresponding to the indices of $\mathcal{L}_k$ are updated. This improve the

criterion of [29] where the new index is selected and updated but the rest of entries are not updated.

Thus m_k is searched in the set $\mathcal{S}_k = \{m_k \in \mathbb{R}^n / m_{ik} = 0; \forall i \notin \mathcal{L}_k \cup \{j_k\}\}$,

$$m_k = \sum_{l=1}^{p_k} \frac{\det\left(D_l^k\right)}{\det\left(G_{l-1}^k\right)\det\left(G_l^k\right)} \widetilde{m}_l \tag{15}$$

where $\widetilde{m}_l$ is the vector with non zero entries i_h^k $(1 \leq h \leq l)$. Each of them are obtained computing the corresponding determinant which results from replacing the last row of $\det\left(G_l^k\right)$ by e_h^t, with $1 \leq l \leq p_k$.

Evidently, the computation of $||Am_k - e_k||_2^2$ and m_k may be updated by adding the contribution of the current entry $j \in \mathcal{J}_k$ to the previous sum from 1 to $p_k - 1$. In practice, $\det\left(G_l^k\right)$ is compute using the Cholesky factorisation since G_l^k is a symmetric positive definite matrix. Thus, only the factorisation of the last row and column is involved if we take advantage of G_{l-1}^k factorisation. On the other hand, $\det\left(D_l^k\right) / \det\left(G_l^k\right)$ is the value of the last unknown of the system $G_l^k\, d_l = \left(a_{k\, i_1^k}, a_{k\, i_2^k}, ..., a_{k\, i_l^k}\right)^t$ and, thus, only a backward substitution is required. Finally, to obtain $\widetilde{m}_l$ we must solve the system $G_l^k\, v_l = e_l$, with $\widetilde{m}_{i_h^k l} = v_{hl}$, $(1 \leq h \leq l)$.

3.5 Optimal diagonal preconditioner

In the particular case of $\mathcal{S}$ being the subspace of diagonal matrices of order n, the optimal diagonal left preconditioner is,

$$M = diag\left(\frac{a_{11}}{\|e_1^T A\|_2^2}, \frac{a_{22}}{\|e_2^T A\|_2^2}, \ldots, \frac{a_{nn}}{\|e_n^T A\|_2^2}\right) \tag{16}$$

$$\|MA - I\|_F^2 = n - \sum_{i=1}^{n} \frac{a_{ii}}{\|e_i^T A\|_2^2} \tag{17}$$

The right preconditioner may be obtained similarly.

4 Krylov subspace methods

4.1 Conjugate Gradient method (CG)

The Conjugate Gradient method is based on an orthogonal projection technique over the Krylov subspace $\mathcal{K}_k\left(A; r_0\right)$, where r_0 is the initial residual vector. It is derived from D-Lanczos algorithm as follows. An approximation x_{j+1} may be written as,

$$x_{j+1} = x_j + \alpha_j p_j \tag{18}$$

Then, the residual vectors must satisfy,

$$r_{j+1} = r_j - \alpha_j A p_j \tag{19}$$

and they are orthogonal,

$$\left\langle r_j - \alpha_j A p_j, r_j \right\rangle = 0$$

Thus,

$$\alpha_j = \frac{\langle r_j, r_j \rangle}{\left\langle A p_j, r_j \right\rangle} \tag{20}$$

As the following directions p_{j+1} are linear combinations of r_{j+1} and p_j, after a suitable scaling of vectors p, we obtain

$$p_{j+1} = r_{j+1} + \beta_j p_j \tag{21}$$

and then,

$$\left\langle A p_j, r_j \right\rangle = \left\langle A p_j, p_j - \beta_{j-1} p_{j-1} \right\rangle = \left\langle A p_j, p_j \right\rangle$$

since Ap_j is orthogonal to p_{j-1}. Thus, taking into account (20) and (21) we have,

$$\beta_j = -\frac{\left\langle r_{j+1}, A p_j \right\rangle}{\left\langle p_j, A p_j \right\rangle}$$

From (19), it yields,

$$A p_j = -\frac{1}{\alpha_j} \left(r_{j+1} - r_j \right) \tag{22}$$

and, finally,

$$\beta_j = -\frac{1}{\alpha_j} \frac{\langle r_{j+1}, (r_{j+1} - r_j) \rangle}{\left\langle A p_j, p_j \right\rangle} = \frac{\langle r_{j+1}, r_{j+1} \rangle}{\langle r_j, r_j \rangle}$$

PRECONDITIONED CONJUGATE GRADIENT ALGORITHM (PCG)

Initial guess x_0. $r_0 = b - A x_0$;

Solve $M z_0 = r_0$, $p_0 = z_0$;

While $\| r_j \| \, / \, \| r_0 \| \geq \varepsilon \quad (j = 0, 1, 2, 3, ...)$, do

$\alpha_j = \dfrac{\langle r_j, z_j \rangle}{\left\langle A p_j, p_j \right\rangle}$;

$x_{j+1} = x_j + \alpha_j p_j$;

$r_{j+1} = r_j - \alpha_j A p_j$;

Solve $M z_{j+1} = r_{j+1}$;

$\beta_j = \dfrac{\langle r_{j+1}, z_{j+1} \rangle}{\langle r_j, z_j \rangle}$;

$p_{j+1} = z_{j+1} + \beta_j p_j$;

End

4.2 Generalised Minimal Residual method (GMRES)

The GMRES algorithm [16] is a method of projection over $\mathcal{K} = \mathcal{K}_k$ with $\mathcal{L} = A\mathcal{K}_k$, where $\mathcal{K}_k$ is a Krylov subspace of dimension k. It minimises the residual norm. So, the development of the algorithm consists of finding a vector x of $x_0 + \mathcal{K}_k$ such that,

$$x = x_0 + V_k y$$

imposing the minimal condition for,

$$J(y) = \|b - Ax\| \tag{23}$$

As,

$$b - Ax = b - A\left(x_0 + V_k y\right) = r_0 - AV_k y$$

taking into account the Arnoldi algorithm [15],

$$AV_k = V_{k+1}\overline{H}_k \tag{24}$$

and denoting $v_1 = r_0 / \parallel r_0 \parallel$, being $\beta = \parallel r_0 \parallel$, then,

$$b - Ax = \beta v_1 - V_{k+1}\overline{H}_k y$$

But $v_1 = V_{k+1}e_1$, with $e_1 \in \Re^{k+1}$. Thus,

$$b - Ax = V_{k+1}\left(\beta e_1 - \overline{H}_k y\right) \tag{25}$$

and equation (23) becomes,

$$J(y) = \left\| V_{k+1}\left(\beta e_1 - \overline{H}_k y\right)\right\|$$

As columns of matrix V_{k+1} are orthonormal due to their construction, we can simplify the above equation,

$$J(y) = \left\| \left(\beta e_1 - \overline{H}_k y\right)\right\| \tag{26}$$

The GMRES algorithm searches the unique vector of $x_0 + \mathcal{K}_k$ which minimises the function $J(y)$.

GMRES ALGORITHM
Initial guess x_0. $r_0 = b - Ax_0$;
Define the $(k+1) \times k$ matrix $\overline{H}_k = \{H\}_{1 \le i \le k+1, 1 \le j \le k}$. Do $\overline{H}_k = 0$.
From $j = 1, ..., k$ do
    $w_j = Av_j$
    From $i = 1, ..., j$ do
        $\{H\}_{ij} = \langle w_j, v_i \rangle$;
        $w_j = w_j - \{H\}_{ij} v_i$;

End

$\{H\}_{j+1,j} = \| w_j \|$; If $\{H\}_{j+1,j} = 0$ do $k = j$ and stop

$v_{j+1} = \dfrac{1}{\{H\}_{j+1,j}} w_j$;

End

Obtain y_k which minimises $\left\| \left(\beta e_1 - \overline{H}_k y\right) \right\|$;

$x_k = x_0 + V_k y_k$ being $V_k = [v_1, v_2, ..., v_k]$;

$r_k = b - A x_k$

End

The GMRES algorithm may be unpracticable when k is very high due to the excessive computational cost. For this reason, restarted techniques are use with GMRES.

A variant of GMRES consists of using a direct solver for the least square problem that arises in each iteration [32], instead of the QR factorisation [16] or the Householder transformation [33]. Consider the orthogonal projection over the subspace of solutions, i.e., the product by matrix H_k^t,

$$H_k^t \, H_k \, u = H_k^t \, \beta_{i-1} e_1 \tag{27}$$

The structure of H_k suggests the decomposition of the product $H_k^t \, H_k$ in a sum. Indeed, H_k is a $(k+1) \times k$ matrix with the following structure,

$$H_k = \begin{pmatrix} \boxed{d_k^t} \\ \boxed{\begin{matrix} & U_k \\ & \\ 0 & \end{matrix}} \end{pmatrix}$$

The first row is defined by a vector of dimension k (d_k^t) and the rest form a square and upper triangular matrix U_k,

$$\{d_k\}_i = d_i = \{H\}_{1i} \qquad i = 1, ..., k \tag{28}$$

$$\{U_k\}_{ij} = u_{ij} = \begin{cases} \{H\}_{i+1\,j} & 1 \le i \le j \le k \\ 0 & \text{in the rest} \end{cases} \tag{29}$$

Theorem. Let d_k and U_k be defined by equations (28) and (29), and $\bar{p}_k$, p_k the solutions to triangular systems $U_k^t \bar{p}_k = d_k$ and $U_k p_k = \bar{p}_k$, respectively. If we define,

$$\lambda_i = \frac{\beta_{i-1}}{1 + \langle d_k, \, p_k \rangle}; \qquad\qquad u_k = \lambda_i \, p_k$$

then u_k minimises the quadratic function $J(u) = \|\beta_{i-1} e_1 - H_k \, u\|_2$ over R^k.

Proof. First observe that $1 + \langle d_k, \, p_k \rangle \neq 0$, since

$$\langle d_k, \, p_k \rangle = \left\langle U_k^t U_k p_k, \, p_k \right\rangle = \|U_k p_k\|_2^2 \geq 0$$

and thus λ_i may not degenerate.

Consider now the linear system given by (27). The (i,j) entry of $H_k^t\, H_k$ is,

$$\left\{H_k^t\, H_k\right\}_{ij} = d_i d_j + \sum_{m=1}^{k} u_{mi} u_{mj} \tag{30}$$

So we can write,

$$\left(d_k d_k^t + U_k^t U_k\right) u = H_k^t\, \beta_{i-1} e_1 \tag{31}$$

Taking into account that $H_k^t e_1 = d_k$, we obtain,

$$\left(d_k d_k^t + U_k^t U_k\right) u = \beta_{i-1} d_k \tag{32}$$

which can be expressed as

$$U_k^t U_k u = d_k \left(\beta_{i-1} - \langle d_k, u\rangle\right) \tag{33}$$

If we define,

$$\lambda_i = \beta_{i-1} - \langle d_k, u\rangle \tag{34}$$

$$u = \lambda_i p_k \tag{35}$$

then it yields,

$$U_k^t U_k p_k = d_k \tag{36}$$

This means to solve only two triangular systems since U_k^t and U_k are lower and upper triangular matrices, respectively. Once the systems are solved, we can compute λ_i in order to obtain u in equation (35). Substituting equation (35) in (34), it yields,

$$\lambda_i = \beta_{i-1} - \langle d_k, u\rangle = \beta_{i-1} - \lambda_i \langle d_k, p_k\rangle$$

and then, it proved that

$$\lambda_i = \frac{\beta_{i-1}}{1 + \langle d_k,\, p_k\rangle} \tag{37}$$

The computation of the new residual is based in the following result,

$$r_i = V_{k+1} \left(\beta_{i-1} e_1 - H_k\, u\right) \tag{38}$$

where we know that $V_{k+1} = [v_1, v_2, ..., v_{k+1}]$ and it is unitary. Thus,

$$r_i = V_{k+1} \hat{r}_i \tag{39}$$

being $\hat{r}_i$ the $(k+1)$ vector,

$$\hat{r}_i = \beta_{i-1} e_1 - H_k\, u \tag{40}$$

In addition, it is evident that,

$$\|r_i\|_2 = \|\hat{r}_i\|_2 \tag{41}$$

From the decomposition of H_k, we can see that the first entry of the $(k+1)$ vector $(H_k\, u)$ is the inner product $\langle d_k, u\rangle$, and the rest are given by the k vector $(U_k u)$. Then, the first entry of $\hat{r}_i$ is λ_i, and the rest,

$$-U_k u = -\lambda_i U_k p_k = -\lambda_i \bar{p}_k \tag{42}$$

being useful to keep the vector $\bar{p}_k$ after solving (36).

The Variable GMRES algorithm is a variant of GMRES. In short, the main idea is to use *full* GMRES in the first steps until a sub-accuracy δ, that may be a function of the required accuracy ε, is satisfied. So, the dimension k of the Krylov subspace is increased in each step while the residual norm is greater or equal to δ and k is lower than the maximal allowed dimension of the Krylov subspace k_{top} (e.g., due to memory requirements). From this step and with the current dimension k, the algorithm starts to work like a standard *restarted* GMRES until reaching the given accuracy ε [34].

PRECONDITIONED VGMRES ALGORITHM

Initial guess x_0. $r_0 = b - Ax_0$;

Choose k_{init}, k_{top}, $\delta \in [0,1]$, $k = k_{init}$

While $\| \widehat{r}_{i-1} \| \,/\, \| r_0 \| \geq \varepsilon \quad (i = 1, 2, 3, ...)$,

$\beta_{i-1} = \| r_{i-1} \|$, $v_i = r_{i-1}/\beta_{i-1}$;

If $\| r_{i-1} \| \,/\, \| r_0 \| \geq \delta$ y $k < k_{top}$ do $k = k + 1$;

From $j = 1, ..., k$ do

Solve $Mz_j = v_j$;

$w = Az_j$;

From $n = 1, ..., j$ do

$\{H\}_{nj} = w^t v_n$;

$w = w - \{H\}_{nj}\, v_n$;

End

$\{H\}_{j+1,j} = \| w \|$;

$v_{j+1} = w / \{H\}_{j+1,j}$:

End

Solve $U_k^t \bar{p} = d_k$ and $U_k p = \bar{p}$; with $\begin{matrix} \{d_k\}_m = \{H\}_{1m} \\ \{U_k\}_{lm} = \{H\}_{l+1,m} \end{matrix} \quad l, m = 1, ..., k$;

$$\lambda_i = \frac{\beta_{i-1}}{1 + d_k^t\, p};$$

$u_k = \lambda_i\, p$;

$x_i = x_{i-1} + Z_k u_k$; being $Z_k = [z_1, z_2, ..., z_k]$;

$r_i = Z_{k+1} \hat{r}_i$; with $\begin{matrix} \{\hat{r}_i\}_1 = \lambda_i \\ \{\hat{r}_i\}_{l+1} = -\lambda_i \{\bar{p}\}_l \end{matrix} \quad l = 1, ..., k$;

End

This algorithm allows to use different preconditioners in each iteration in the same way as FGMRES does.

4.3 Biconjugate Gradient Stabilised method (Bi-CGSTAB)

In CGS algorithm [35], residual vectors verify $\widehat{r}_j = \phi_j^2(A)r_0$. Van der Vorst in Bi-CGSTAB [22], proposes to obtain a residual vector by applying successively two different polynomial as follows,

$$r_j^{'} = \psi_j(A)\phi_j(A)r_0 \tag{43}$$

such that the recurrence relations of the algorithm where the polynomial $\psi_j(A)$ is involved must not be too complicated in order to optimise the residual in function of the parameters that appear in its definition. For this purpose, the next expression is suggested for $\psi_j(A)$,

$$\psi_{j+1}(A) = (I - w_j A)\psi_j(A) \tag{44}$$

obtaining w_j minimising r_j in the $j-th$ iteration. The recurrence relations are derived in the same way as in CGS algorithm. Thus,

$$\begin{aligned} \psi_{j+1}(A)\phi_{j+1}(A) &= (I - w_j A)\psi_j(A)\phi_{j+1}(A) \\ &= (I - w_j A)\left[\psi_j(A)\phi_j(A) - \alpha_j A\psi_j(A)\pi_j(A)\right] \end{aligned} \tag{45}$$

For the conjugate direction, we have

$$\begin{aligned} \psi_j(A)\pi_j(A) &= \psi_j(A)\left[\phi_j(A) + \beta_{j-1}\pi_{j-1}(A)\right] \\ &= \psi_j(A)\phi_j(A) + \beta_{j-1}(I - w_{j-1}A)\psi_{j-1}(A)\pi_{j-1}(A) \end{aligned}$$

and we define,

$$r_j = \psi_j(A)\phi_j(A)r_0,$$

$$p_j = \psi_j(A)\pi_j(A)r_0$$

Taking into account the above expressions, we can find the corresponding equations in function of parameters α_j y β_j,

$$r_{j+1} = (I - w_j A)\left(r_j - \alpha_j A p_j\right) \tag{46}$$

$$p_{j+1} = r_{j+1} + \beta_j(I - w_j A)p_j \tag{47}$$

To get the algorithm, we take as reference the BiCG method and carry out the necessary transformations to obtain the recurrence relations in function of the new residual vector.

In BiCG algorithm we have $\beta_j = \rho_{j+1}/\rho_j$ with,

$$\rho_j = \left\langle \phi_j(A)r_0, \phi_j(A^T)r_0^* \right\rangle = \left\langle \phi_j^2(A)r_0, r_0^* \right\rangle$$

As ρ_j is not computed since any vector $\phi_j(A)r_0, \phi_j(A^T)r_0^*$ or $\phi_j^2(A)r_0$ is available, we can relate it with the parameter,

$$\widetilde{\rho}_j = \left\langle \phi_j(A)r_0, \psi_j(A^T)r_0^* \right\rangle$$

which may be obtained as,

$$\widetilde{\rho}_j = \langle \psi_j(A)\phi_j(A)r_0, r_0^* \rangle = \langle r_j, r_0^* \rangle$$

Developing $\phi_j(A^T)r_0^*$ explicitly to relate ρ_j and $\widetilde{\rho}_j$, taking into account the orthogonality of $\phi_j(A)r_0$ with respect to all the vectors $(A^T)^i r_0^*$, with $i < j$, and since only the main coefficients of the polynomials are important in the above product,

$$\widetilde{\rho}_j = \left\langle \phi_j(A)r_0, \eta_1^j(A^T)^j r_0^* + \eta_2^j(A^T)^{j-1} r_0^* + ... \right\rangle$$

If γ_1^j is the main coefficient of the polynomial $\phi_j(A)$, then,

$$\widetilde{\rho}_j = \left\langle \phi_j(A)r_0, \frac{\eta_1^j}{\gamma_1^j}\phi_j(A^T)r_0^* \right\rangle = \frac{\eta_1^j}{\gamma_1^j}\rho_j$$

From the recurrence relations of $\phi_{j+1}(A)$ and $\psi_{j+1}(A)$, the main coefficients of these polynomials are obtained to satisfy,

$$\eta_1^{j+1} = -w_j\eta_1^j, \quad \gamma_1^{j+1} = -\alpha_j\gamma_1^j$$

Thus,

$$\frac{\widetilde{\rho}_{j+1}}{\widetilde{\rho}_j} = \frac{w_j}{\alpha_j}\frac{\rho_{j+1}}{\rho_j}$$

This allows us to find the next expression of β_j,

$$\beta_j = \frac{\widetilde{\rho}_{j+1}}{\widetilde{\rho}_j}\frac{\alpha_j}{w_j}$$

Similarly, we can define α_j as,

$$\alpha_j = \frac{\left\langle \phi_j(A)r_0, \phi_j(A^T)r_0^* \right\rangle}{\langle A\pi_j(A)r_0, \pi_j(A^T)r_0^* \rangle}$$

Considering only the main coefficients in the products of polynomials again, since these are identical for $\phi_j(A^T)r_0^*$ and $\pi_j(A^T)r_0^*$, we obtain,

$$\alpha_j = \frac{\left\langle \phi_j(A)r_0, \phi_j(A^T)r_0^* \right\rangle}{\langle A\pi_j(A)r_0, \phi_j(A^T)r_0^* \rangle} = \frac{\left\langle \phi_j(A)r_0, \psi_j(A^T)r_0^* \right\rangle}{\langle A\pi_j(A)r_0, \psi_j(A^T)r_0^* \rangle} = \frac{\langle \psi_j(A)\phi_j(A)r_0, r_0^* \rangle}{\langle A\psi_j(A)\pi_j(A)r_0, r_0^* \rangle}$$

In addition, $p_j = \psi_j(A)\pi_j(A)r_0$, thus,

$$\alpha_j = \frac{\widetilde{\rho}_j}{\left\langle Ap_j, r_0^* \right\rangle} \tag{48}$$

From equation (46), if we consider,

$$s_j = r_j - \alpha_j A p_j \tag{49}$$

the optimal value of w_j in the polynomial $\psi_j(A)$ is obtained by minimising the norm of the residual vector,

$$r_{j+1} = s_j - w_j A s_j$$

$$\|r_{j+1}\|^2 = \langle s_j - w_j A s_j, s_j - w_j A s_j \rangle = \langle s_j, s_j \rangle - 2w_j \langle s_j, A s_j \rangle + w_j^2 \langle A s_j, A s_j \rangle$$

$$\frac{\partial \|r_{j+1}\|^2}{\partial w_j} = -2 \langle s_j, A s_j \rangle + 2w_j \langle A s_j, A s_j \rangle = 0$$

and it yields,

$$w_j = \frac{\langle A s_j, s_j \rangle}{\langle A s_j, A s_j \rangle} \tag{50}$$

Equation (46) becomes now,

$$r_{j+1} = s_j - w_j A s_j = r_j - \alpha_j A p_j - w_j A s_j \tag{51}$$

and the recurrence relation for the unknown vector is given by,

$$x_{j+1} = x_j + \alpha_j p_j + w_j s_j \tag{52}$$

Once all the vector are expressed in function of the new residual vector and obtained their recurrence relations as well as the parameters involved, we can write an algorithm which contains two matrix-vector and four inner products. So Bi-CGSTAB carries out two inner products more than CGS. However, in general the convergence of the former is faster and smoother than the latter, what makes Bi-CGSTAB to be preferable to CGS.

For each preconditioning form [12], the expression of $\widetilde{\omega}$ is,

$$\left.\begin{array}{ll} \widetilde{\omega} = \dfrac{(As)^T s}{(As)^T As} & \text{right preconditioning} \\ \widetilde{\omega} = \dfrac{(M^{-1}As)^T (M^{-1}s)}{(M^{-1}As)^T (M^{-1}As)} & \text{left preconditioning} \\ \widetilde{\omega} = \dfrac{(L^{-1}As)^T (L^{-1}s)}{(L^{-1}As)^T (L^{-1}As)} & \text{both sides preconditioning} \end{array}\right\} \tag{53}$$

However, if $\widetilde{\omega}$ is obtained from the minimisation of the residual vector related to the unpreconditioned system, all the expressions in (53) coincide with that of right preconditioning, and thus, a unique algorithm is obtained (see Bi-CGSTABP in [22]).

PRECONDITIONED BICGSTAB ALGORITHM

Initial guess x_0. $r_0 = b - Ax_0$;

Choose any r_0^* such that $\langle r_0, r_0^* \rangle \neq 0$;

Solve $Mz_0 = r_0$;

$p_0 = z_0$;

While $\| r_{j-1} \| / \| r_0 \| \geq \varepsilon \quad (j = 1, 2, 3, ...)$, do

Solve $Mz_j = r_j$;

$y_j = Ap_j$;

Solve $Mv_j = y_j$;

$$\alpha_j = \frac{\langle z_j, r_0^* \rangle}{\langle v_j, r_0^* \rangle};$$

$s_j = r_j - \alpha_j y_j$;

$u_j = As_j$;

Solve $Mt_j = u_j$;

$$\widetilde{\omega}_j = \frac{\langle t_j, s_j \rangle}{\langle t_j, t_j \rangle};$$

$x_{j+1} = x_j + \alpha_j p_j + \widetilde{\omega}_j u_j$;

$z_{j+1} = s_j - \widetilde{\omega}_j t_j$;

$$\beta_j = \frac{\langle z_{j+1}, r_0^* \rangle}{\langle z_j, r_0^* \rangle} \frac{\alpha_j}{\widetilde{\omega}_j};$$

$p_{j+1} = z_{j+1} + \beta_j \left(p_j - \widetilde{\omega}_j v_j \right)$;

End

As a consequence, we can remark that any other selection of the initial residual vector will lead to a new preconditioning form (see [36]).

4.4 Quasi-Minimal Residual Stabilised method (QMRCGSTAB)

Proposed by Chan et al [23], it is based in the application of the minimisation principle used for Bi-CGSTAB algorithm to QMR method. Consider,

$$\begin{aligned} Y_k &= [y_1, y_2, \ldots y_k] \ , \\ W_{k+1} &= [w_0, w_1, \ldots w_k] \end{aligned}$$

such that,

$$\begin{cases} y_{2l-1} = p_l \quad \text{for} \ \ l = 1, \ldots, [(k+1)/2] \\ y_{2l} = s_l \quad \text{for} \ \ l = 1, \ldots, [k/2] \end{cases}$$

and,

$$\begin{cases} w_{2l-1} = s_l \quad \text{para} \ \ l = 1, \ldots, [(k+1)/2] \\ w_{2l} = r_l \quad \text{para} \ \ l = 0, 1, \ldots, [k/2] \end{cases}$$

where $[k/2]$ and $[(k+1)/2]$ mean the integer part of $k/2$ and $(k+1)/2$, respectively. Defining,

$$[\delta_1, \delta_2, \ldots \delta_k] \,/ \begin{cases} \delta_{2l} = \omega_l & \text{para } \; l = 1, \ldots, [(k+1)/2] \\ \delta_{2l-1} = \alpha_l & \text{para } \; l = 1, \ldots, [(k+1)/2] \end{cases}$$

then, for each column of W_{k+1} and Y_k, equations (49) and (51) yield,

$$Ay_m = (w_{m-1} - w_m)\, \delta_m^{-1}, \qquad m = 1, \ldots, k \tag{54}$$

i.e., in matrix notation,

$$AY_k = W_{k+1} E_{k+1}$$

where E_{k+1} is a $(k+1) \times k$ bi-diagonal matrix of which diagonal entries are δ_m^{-1} and lower diagonal entries $-\delta_m^{-1}$. This may be easily derived until the degrees of the polynomials corresponding to vectors r_j, s_j and p_j become $2j$, $2j-1$ and $2j-2$, respectively. Then, Y_k and W_k generate the same Krylov subspace as r_0 but with degree $k-1$.

The main idea in QMRCGSTAB is to find an approximation to the solution of the linear system (1) by using the Krylov subspace $\mathcal{K}_{k-1}$ as follows,

$$x_k = x_0 + Y_k g \qquad \text{con} \qquad g \in \Re^n$$

So, the equation for the residual vector results,

$$r_k = r_0 - AY_k g = r_0 - W_{k+1} E_{k+1} g$$

Taking into account that the first column of W_{k+1} is just r_0, then,

$$r_k = W_{k+1}\,(e_1 - E_{k+1} g)$$

Since the columns of W_{k+1} are not normalised, we shall use a scaling matrix $\Sigma_{k+1} = diag\,(\sigma_1, \ldots, \sigma_{k+1})$ con $\sigma_j = \|w_j\|$ to make unitary the columns of W_{k+1}. Thus,

$$r_k = W_{k+1} \Sigma_{k+1}^{-1} \Sigma_{k+1}\,(e_1 - E_{k+1} g) = W_{k+1} \Sigma_{k+1}^{-1}\,(\sigma_1 e_1 - H_{k+1} g)$$

with $H_{k+1} = \Sigma_{k+1} E_{k+1}$.

QMR approximation consists in the minimisation of $\|\sigma_1 e_1 - H_{k+1} g\|$ in order to obtain the optimum $g_k \in \Re^k$, where the least square problem is solved by using the QR decomposition of matrix H_{k+1}, step by step applying Givens rotations. As H_{k+1} is lower bi-diagonal, the rotation is only needed in each previous step.

PRECONDITIONED QMRCGSTAB ALGORITHM

Initial guess x_0. $r_0 = b - Ax_0$;

Solve $Mz_0 = r_0$;

Choose $\widetilde{r}_0$ such that $\langle z_0, \widetilde{r}_0 \rangle \neq 0$

$p_0 = v_0 = d_0 = 0$;
$\rho_0 = \alpha_0 = \widetilde{\omega}_0 = 1$, $\tau_0 = \|z_0\|$, $\theta_0 = 0$, $\eta_0 = 0$;
While $\sqrt{j+1}\,|\widetilde{\tau}| \,/\, \|r_0\| \geq \varepsilon (j = 1, 2, ...)$, do
$\rho_j = \langle z_{j-1}, \widetilde{r}_0 \rangle$, $\beta_j = (\rho_j / \rho_{j-1})(\alpha_{j-1} / \tilde{\omega}_{j-1})$;
$p_j = z_{j-1} + \beta_j (p_{j-1} - \tilde{\omega}_{j-1} v_{j-1})$;
$y_j = A p_j$;
Solve $M v_j = y_j$;
$\alpha_j = \rho_j / \langle v_j, \widetilde{r}_0 \rangle$;
$s_j = z_{j-1} - \alpha_j v_j$;
$\widetilde{\theta}_j = \|s_j\| / \tau$, $c = \dfrac{1}{\sqrt{1 + \widetilde{\theta}_j^2}}$; $\widetilde{\tau} = \tau\, \widetilde{\theta}_j c$;
$\widetilde{\eta}_j = c_j^2 \alpha_j$;
$\widetilde{d}_j = p_j + \dfrac{\theta_{j-1}^2 \eta_{j-1}}{\alpha_j} d_{j-1}$;
$\widetilde{x}_j = x_{j-1} + \widetilde{\eta}_j \widetilde{d}_j$;
$u_j = A s_j$;
Solve $M t_j = u_j$;
$\widetilde{\omega}_j = \dfrac{\langle s_j, t_j \rangle}{\langle t_j, t_j \rangle}$;
$z_j = s_j - \widetilde{\omega}_j t_j$;
$\theta_j = \|z_j\| / \widetilde{\tau}$, $c = \dfrac{1}{\sqrt{1 + \theta_j^2}}$; $\tau = \widetilde{\tau}\, \theta_j c$;
$\eta_j = c^2 \widetilde{\omega}_j$;
$d_j = s_j + \dfrac{\widetilde{\theta}_j^2 \widetilde{\eta}_j}{\widetilde{\omega}_j} \widetilde{d}_j$;
$x_j = \widetilde{x}_j + \eta_j d_j$;
End

4.5 Least-Square QR method (LSQR)

The resolution of (1), with nonsymmetric matrix A, is equivalent to solve the normal equation $A^T A x = A^T b$, where $A^T A$ is symmetric positive definite. As in GC algorithm, the so-called Normal Equation methods verify the conditions of residual minimisation and computational cost optimisation. However, they have the disadvantage that the condition number of the system is squared ($K_2\left(A^T A\right) = K_2\left(A\right)^2$). This may be a serious problem for ill-conditioned systems. In addition most these methods involve two matrix-vector products related to matrices A and A^T, increasing the computational cost.

The LSQR method, proposed by Paige and Saunders [24], was formulated in order to correct the increasing of the system condition number when the normal equation is applied. The main objective of LSQR algorithm is to obtain the solution of the symmetric system,

$$\begin{pmatrix} I & A \\ A^T & -\lambda^2 I \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \tag{55}$$

minimising,

$$\left\| \begin{pmatrix} A \\ \lambda I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2 \tag{56}$$

where λ is an arbitrary real number. The application of the Lanczos process to this system yields two ways for bidiagonalisation proposed by Golub and Kahan [37]. The preconditioned LSQR algorithm is presented below.

PRECONDITIONED LSQR ALGORITHM

Initial guess x_0. $r_0 = b - Ax_0$;

Solve $Mz_0 = r_0$;

$\beta_1 = \|z_0\|$, $u_1 = z_0/\beta_1$;

Solve $M^T s_1 = u_1$;

$\alpha_1 = \left\|A^T s_1\right\|$, $v_1 = A^T s_1/\alpha_1$, $w_1 = v_1$;

$\overline{\phi}_1 = \beta_1$, $\overline{\rho}_1 = \alpha_1$;

While $\overline{\phi}_j / \parallel r_0 \parallel \geq \varepsilon$ ($j = 1, ...$), do

$p_j = Av_j$;

Solve $Mq_j = p_j$;

$\beta_{j+1} = \|q_j - \alpha_j u_j\|$;

$u_{j+1} = \dfrac{q_j - \alpha_j u_j}{\beta_{j+1}}$;

Resolver $Ms_{j+1} = u_{j+1}$;

$\alpha_{j+1} = \left\|A^T s_{j+1} - \beta_{j+1} v_j\right\|$;

$v_{j+1} = \dfrac{A^T s_{j+1} - \beta_{j+1} v_j}{\alpha_{j+1}}$;

$\rho_j = \left(\overline{\rho}_j^2 + \beta_{j+1}^2\right)^{\frac{1}{2}}$;

$c_j = \dfrac{\overline{\rho}_j}{\rho_j}$;

$s_j = \dfrac{\beta_{j+1}}{\rho_j}$;

$\theta_{j+1} = s_j \alpha_{j+1}$;

$\overline{\rho}_{j+1} = -c_j \alpha_{j+1}$;

$\phi_j = c_j \overline{\phi}_j$;

$\overline{\phi}_{j+1} = s_j \overline{\phi}_j$;

$$x_j = x_{j-1} + \left(\frac{\phi_j}{\rho_j}\right) w_j;$$
$$w_{j+1} = v_{j+1} - \left(\frac{\theta_{j+1}}{\rho_j}\right) w_j;$$

End

5 Numerical experiments

In this section we consider some problems arising from scientific and industrial applications. All the computing was carried out in FORTRAN double precision. We always chose $x_0 = 0$, $\tilde{r}_0 = r_0 = b$, as initial guess and $\frac{\|b-Ax_i\|_2}{\|b\|_2} < 10^{-9}$ as stop criterion. Next we present a short description of the problems.

LightTruck problem comes from the numerical simulation of a canister in $3D$. The physical phenomenon which defines these devices is the transport of hydrocarbons. Huerta et al [38] have developed a convection-diffusion-reaction model,

$$\frac{\partial u}{\partial t} + v \cdot \nabla u - \nu \Delta u + \sigma(u)\, u = f(u)$$

where u is the concentration of hydrocarbons in the air, v represents the velocity field of the air which is previously computed by solving a potential flow problem, and ν is diffusivity coefficient. The reaction term $\sigma(u)\, u$ and the source $f(u)$ are strongly nonlinear. We have studied three linear systems of equations corresponding to the finite element discretization of three time step (4030, 10044 y 16802) of the transient problem. However, here we only show results from the last one since the others lead us to similar conclusions. The number of unknown is $n = 17914$, and the matrix is the same for the three systems. Figures 1 and 2 show that preconditioning is essential here. They also illustrate the comparison of computational cost and iteration, respectively, of PCG with several preconditioners. We can observe that PCG with Diagonal approximate inverse is the fastest. However SSOR and ILLT(0) produce less iterations. In this problem, since the preconditioner is constructed once, the latter may be competitive with the former and win at the end of the transient process.

Convdifhor example corresponds to a convection-diffusion problem defined in $[0,1]\times[0,1]$ by,

$$v_1 \frac{\partial u}{\partial x} - K\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = F$$

where $v_1 = 10^4\left(y - \frac{1}{2}\right)(x - x^2)\left(\frac{1}{2} - x\right)$, $K = 10^{-5} - 10^2$, y $F = 10^3 - 1$. The linear systems arise from an unstructured finite element mesh with $n = 441$ and $nz = 2927$, $n = 1960$ and $nz = 13412$, $n = 13190$ and $nz = 91227$, respectively, being n the number of unknowns and nz the amount of nonzero entries in the matrices. The convergence of Bi-CGSTAB improves when the sparse approximate inverse preconditioner is used. The number of iterations clearly decrease if we diminish ε_k. Nevertheless, to obtain an approximate for $n = 441$ the amount of entries per columns mast

be augmented to 200. Figure 5 shows that sparse approximate inverse may be competitive with implicit preconditioners like ILU(0) if parallel computing is available. In figures 3(a)-3(d) the sparsity pattern of A and M are represented for this case. Note that since the sparsity pattern of A and A^{-1} are completely different, the pattern of M is far from the one of A. In Figure 5 we compare the plots of convergence of Bi-CGSTAB-SPAI(0.2) for several reordering techniques in the case of $n = 1960$. The orderings by Minimum Degree and Reverse Cuthill-Mckee algorithms have beneficial effects on the rate of convergence of Bi-CGSTAB-SPAI. The reduction of the amount of entries in SPAI is about 40-50% using these reordering techniques. The Minimum Neighbouring algorithm does not affect to nz. Besides, the number of iterations was reduced from 60 to 70%. In Figures6(a)-6(d) the effect of reordering techniques on the sparsity pattern of SPAI with $\varepsilon_k = 0.3$. These patterns represent full matrices as expected, following the typical plots related to each reordering technique.

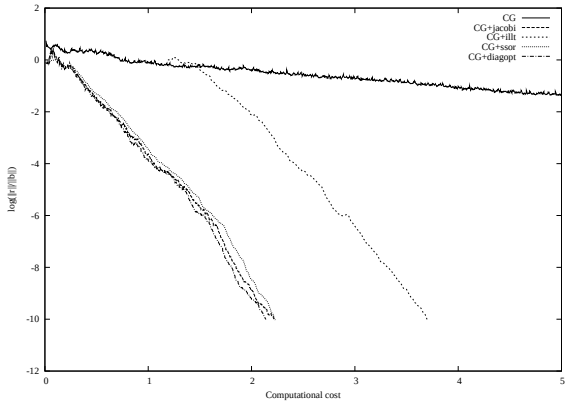


Figure 1: LightTruck problem: computational cost of PCG

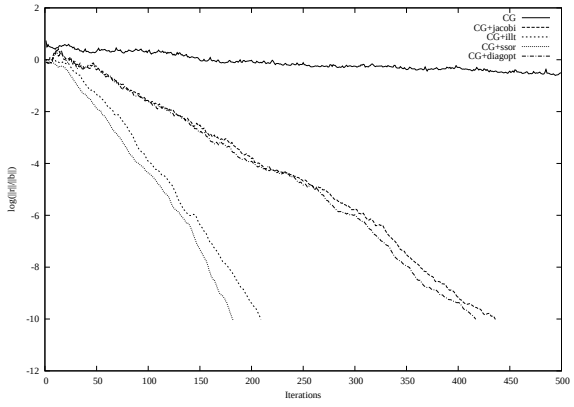
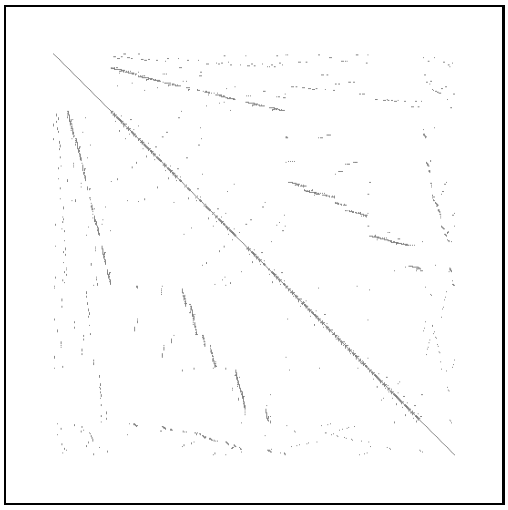
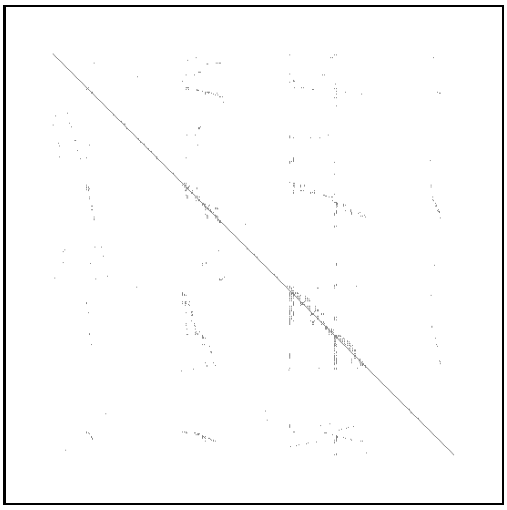


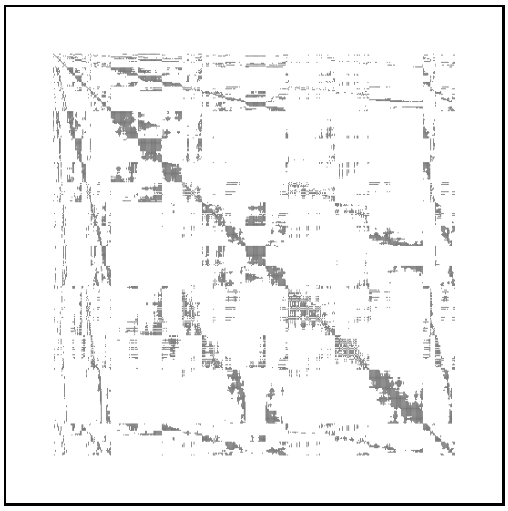
Figure 2: LightTruck problem: iterations of PCG



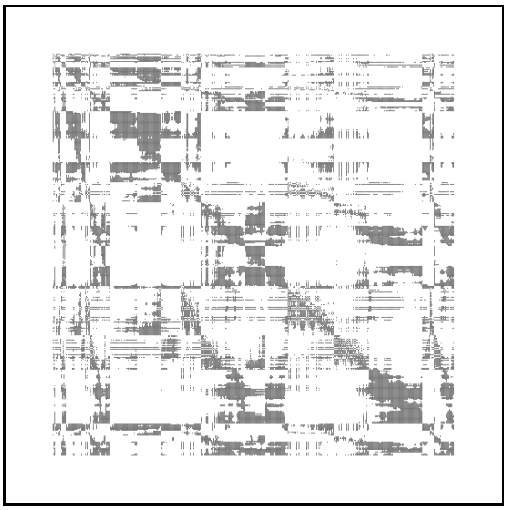
(a) Sparsity pattern of matrix A



(b) Sparsity pattern of SPAI with $\varepsilon_k = 0.5$ and $\max\, nz(m_{0k}) = 50$



(c) Sparsity pattern of SPAI with $\varepsilon_k = 0.5$ and $\max\, nz(m_{0k}) = 50$



(d) Sparsity pattern of SPAI with $\varepsilon_k = 0.05$ and $\max\, nz(m_{0k}) = 200$

Figure 3: Convdifhor problem ($n = 441$): sparsity pattern of A and SPAI.

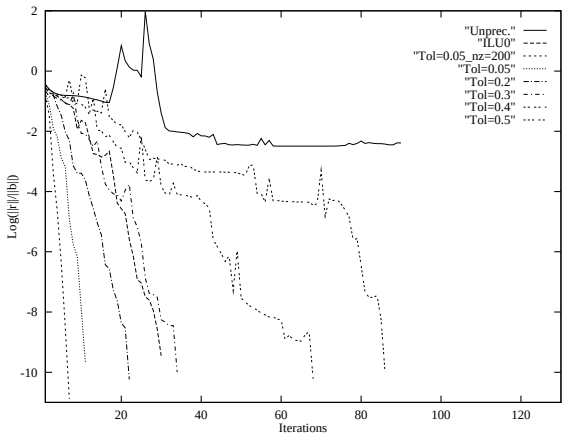


Figure 4: Convdifhor problem ($n = 441$):iterations of preconditioned Bi-CGSTAB.

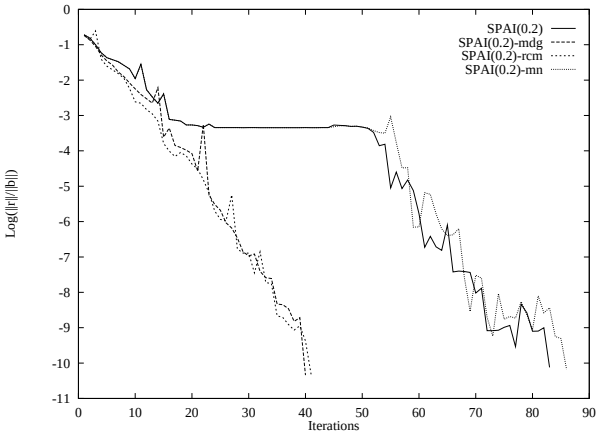
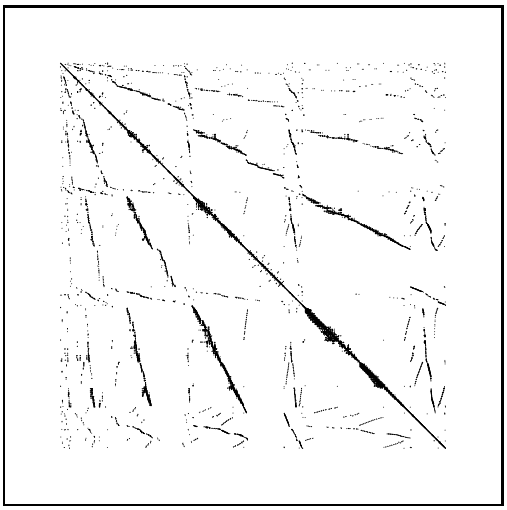


Figure 5: Convdifhor problem ($n = 1960$): effect of reordering on Bi-CGSTAB-SPAI convergence.

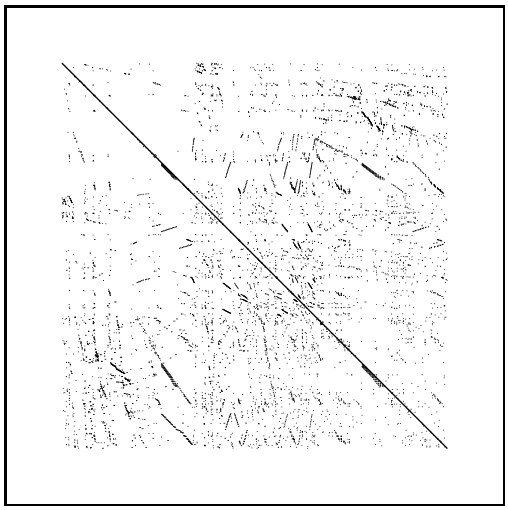
Figure 7 shows the convergence plots of Krylov subspace methods with ILU(0) preconditioner for $n = 13190$. The convergence of Bi-CGSTAB is fast but irregular. QMRCGSTAB avoids these irregularities but at slightly higher computational cost. In this case, ILU(0) is more efficient than the rest of preconditioners for QMRCGSTAB, as can be seen in Figure 8. Finally, reordering techniques reduce the cost under the 50% for reaching convergence with QMRCGSTAB-ILU(0).

6 Conclusions

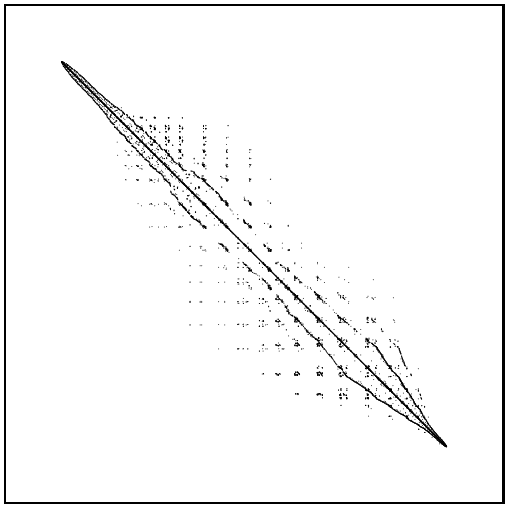
Krylov subspace methods provides a wide set of possibilities for solving linear systems of equations. Although, in the symmetric case the selection is clear (CG), for



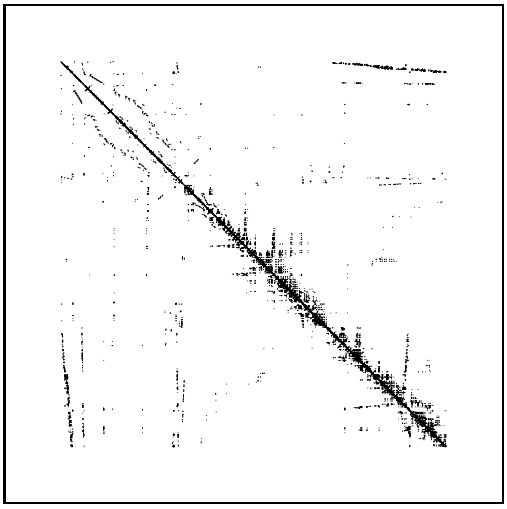
(a) Initial ordering



(b) Minimum Degree



(c) Reverse Cuthill-Mckee



(d) Minimum Neighbouring

Figure 6: Convdifhor problem ($n = 1960$): sparsity pattern of matrix SPAI(0.3) with several orderings.

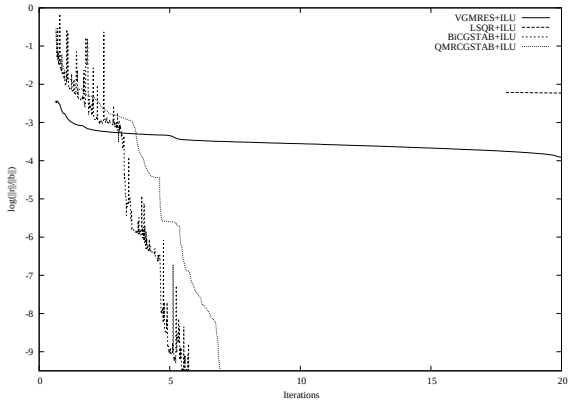


Figure 7: Convdifhor problem ($n = 13190$): iterations of Krylov subspace methods.

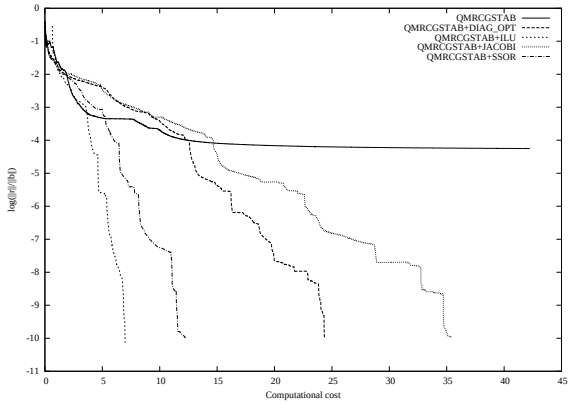


Figure 8: Convdifhor problem ($n = 13190$): computational cost of preconditioned QMRCGSTAB.

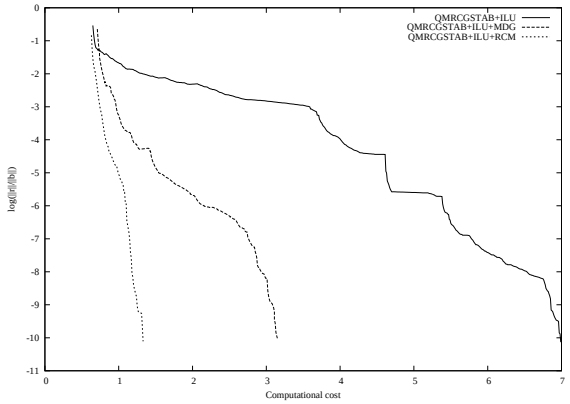


Figure 9: Convdifhor problem ($n = 13190$): Effect of reordering on the computational cost of QMRCGSTAB-ILU(0).

nonsymmetric problems the choice depends on several factors: transient process, available computer memory, parallel computation, very ill-conditioned problems, ...

The proposed algorithm for computing a sparse approximate inverse M_0 of a sparse nonsymmetric matrix A, may be implemented in parallel since the columns (or rows) of M_0 are obtained independently. The sparsity pattern of these preconditioners are dynamically built starting from the diagonal and augmenting the amount of nonzero entries. Evidently, this type of preconditioner may be competitive with implicit ones in a parallel environment. Nevertheless, the effectiveness of these preconditioners in the convergence of iterative solvers has been proved theoretically and practically.

We have experimentally proved that reordering techniques have beneficial effects on the preconditioners for the convergence of Krylov subspace methods. In particular, for sparse approximate inverses, the reduction of the amount of nonzero entries due to reordering allows to obtain SPAI with similar accuracy to those without reordering, but requiring less computer memory and cost. In addition, reordering produces better preconditioners since the allows to reach convergence with less iterations.

More research must be carried out on the effect of other reordering techniques which take into account the values of the entries in A and not only their positions (see [5], [8]). Although these techniques are expensive, in the case of solving many linear systems with the same invariable matrix, e.g. LightTruck problem, this techniques may compete in parallel machines.

Acknowledgement

The work has been partially supported by the Spanish Government (Ministerio de Ciencia y Tecnología) and FEDER, grant number REN2001-0925-C03-02/CLI.

References

[1] M. Benzi, D.B. Szyld, A. van Duin, *"Orderings for incomplete factorization preconditioning of nonsymmetric problems"*, SIAM J. Sci. Comput., 20, 5, 1652-1670, 1999.

[2] M. Benzi, M. Tůma,*" Orderings for factorized sparse approximate inverse preconditioners"*, SIAM J. Sci. Comput., 30, 5, 1851-1868,2000.

[3] L.C. Dutto, *"The Effect of Ordering on Preconditioned GMRES Algorithm"*, Int. Jour. Num, Meth. Eng., 36, 457-497, 1993.

[4] E. Flórez, M.D. García, L. González, G. Montero, *"The effect of orderings on sparse approximate inverse preconditioners for non-symmetric problems"*, Adv. Engng. Software, 33, 7-10, 611-619, 2002.

[5] R.R. Lewis, *"Simulated Annealing for Profile and Fill Reduction of Sparce Matrices"*, Int. Jour. Num, Meth. Eng., 37, 905-925, 1994.

[6] I.L. Lim, I.W. Johnston, S.K. Choi, *"A Comparison of Algorithms for Profile Reduction of Sparse Matrices"*, Computers & Structures, 57, 2, 297-302, 1995.

[7] M. Monga-Made, *"Ordering Strategies for Modified Block Incomplete Factorizations"*, SIAM J. Sci. Comput., 16, 2, 378-399, 1995.

[8] G. Montero, M. Galán, P. Cuesta, G. Winter, *"Effects of stochastic ordering on preconditioned GMRES algorithm"*, in B.H.V. Topping & M. Papadrakakis eds., Advances in Structural Optimization, 241-246, Civil-Comp Press, Edinburgh, 1994.

[9] T.F. Chan, T. Szeto, *"Composite Step Product Methods for Solving Nonsymmetric Linear Systems"*, SIAM J. Sci. Comput., 17, 6, 1491-1508, 1996.

[10] G. Montero, R. Montenegro, G. Winter, L. Ferragut, *Aplicacion de Esquemas EBE en Procesos Adaptativos*, Rev. Int. Met. Num. Cal. Dis. Ing., 6, 311-332, 1990.

[11] P.M. de Zeeuw, *Incomplete Line LU as Smoother and as Preconditioner*, in Eighth GAMM-Seminar, Kiel, 215-224, 1992.

[12] G. Montero, G. Winter, A. Suárez, M. Galán, D. García, *"Contribution to Iterative Methods for Nonsymmetric Linear Systems: GMRES, BCG and QMR Type Methods"*, in Computational Methods and Neural Networks. Part Two: Computational Methods, M. Sambandhamy & M.P. Bekakos, Eds., Dynamic Publishers, Inc., 97-128, 1999.

[13] N.M. Nachttigal, S.C. Reddy, L.N. Trefethen, *"How Fast Are Nonsymmetric Matrix Iterations?"*, SIAM J. Matr. Anal. Appl., 13, 3, 796-825, 1992.

[14] M.R. Hestenes, E. Stiefel, *"Methods of Conjugate Gradients for Solving Linear Systems"*, Jour. Res. Nat. Bur. Sta., 49, 6, 409-436, 1952.

[15] W.E. Arnoldi, *"The Principle of Minimized Iteration in the Solution of the Matrix Eingenvalue Problem"*, Quart. Appl. Math., 9, 17-29, 1951.

[16] Y. Saad, M. Schultz, *"GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems"*, SIAM J. Sci. Statist. Comput., 7, 856-869, 1986.

[17] A. Greenbaum, L.N. Trefethen, *"GMRES/CR and Arnoldi/Lanczos as Matrix Approximation Problems"*, SIAM J. Sci. Comput., 15, 2, 359-368, 1994.

[18] W. Joubert, *"A Robust GMRES-based Adaptive Polynomial Preconditioning Algorithm for Nonsymmetric Linear Systems"*, SIAM J. Sci. Comput., 15, 2, 427-439, 1994.

[19] E.M. Kasenally, *"GMBACK: A Generalised Minimum Backward Error Algorithm for Nonsymmetric Linear Systems"*, SIAM J. Sci. Comput., 16, 3, 698-719, 1995.

[20] N.M. Nachttigal, L. Reichel, L.N. Trefethen, *"A Hybrid GMRES Algorithm for Nonsymmetric Linear Systems"*, SIAM J. Matr. Anal. Appl., 13, 3, 778-795, 1992.

[21] H.A. van der Vorst, C. Vuik, *"GMRESR: A Family of Nested GMRES Methods"*, Tech, Rep. 91-80, Delft University of Technology, Mathematics and Informatics, Delft, The Netherlands, 1991.

[22] H.A. van der Vorst, *"Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems"*, SIAM J. Sci. Comput., 13, 631-644, 1992.

[23] T.F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, C.H. Tong, *"A Quasi-Minimal Residual Variant of the Bi-CGSTAB Algorithm for Nonsymmetric Systems"*,

SIAM J. Sci. Statist. Comput., 15, 338-247, 1994.
[24] C.C. Paige, M.A. Saunders, *"LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares"*, ACM Trans. Math. Soft., 8, 1, 43-71, 1982.
[25] A. George, J.W. Liu, *"The Evolution of the Minimum Degree Ordering Algorithms"*, SIAM Rev. 31, 1-19, 1989.
[26] A. George, *"Computer Implementation of the Finite Element Method"*, Report Stan CS-71-208, 1971.
[27] E.H. Cuthill, J.M. Mckee, *"Reducing the Bandwidth of Sparse Symmetric Matrices"*, Proc. 24th National Conference of the Association for Computing Machinery, Brondon Press, New Jersey, 157-172, 1969.
[28] M. Benzi, M. Tůma, *"A sparse approximate inverse preconditioner for nonsymmetric linear systems"*, SIAM J. Sci. Comput., 19, 3, 968-994, 1998.
[29] M. Grote, T. Huckle, *"Parallel preconditioning with sparse approximate inverses"*, SIAM J. Sci. Comput., 18, 3, 838-853, 1997.
[30] M. Benzi, M. Tůma, *"A comparative study of sparse approximate inverse preconditioners"*, Appl. Num. Math., 30, 305-340, 1999.
[31] G. Montero, L. González, E. Flórez, M.D. García, A. Suárez, *"Short Communication: Approximate inverse computation using Frobenius inner product"*, Num. Lin. Alg. Appl., 9, 239-247, 2002.
[32] M. Galán, G. Montero, G. Winter, *"A Direct Solver for the Least Square Problem Arising From GMRES(k)"*, Com. Num. Meth. Eng., 10, 743-749, 1994.
[33] H.F. Walker, *"Implementation of the GMRES Method Using Householder Transformations"*, SIAM J. Sci. Comput., 9, 152-163, 1988.
[34] M. Galán, G. Montero, G. Winter, *"Variable GMRES: an Optimizing Self-Configuring Implementation of GMRES(k) with Dynamic Memory Allocation"*, Tech. Rep. of CEANI, 1994.
[35] P. Sonneveld, *"CGS: a Fast Lanczos-Type Solver for Nonsymmetric Linear Systems"*, SIAM J. Sci. Statist. Comput., 10, 36-52, 1989.
[36] A. Suárez, G. Montero, D. García, E. Flórez, *"Efecto de la Elección del Vector Inicial en la Convergencia de los Algoritmos CGS y BICGSTAB"*, in Encuentro de Análisis Matricial y Aplicaciones (EAMA 97), Sevilla, 1997.
[37] G.H. Golub, Kahan W., *"Calculating the Singular Values and Pseudoinverse of a Matrix"*, SIAM J. Numer. Anal., 2, 205-224, 1965.
[38] A. Huerta, A. Rodríguez-Ferrán, J. Sarrate, P. Díez., S. Fernández-Méndez, *"Numerical Modelling, a Tool to Improve the Design of Active Carbon Canisters"*, in Abstracts of the Sixth U.S. National Congress on Computational Mechanics, Dearborn, Michigan, 2001.