Abstract

We consider the preconditioning of linear systems with matrices depending on a parameter, i.e., $A_\varepsilon x_\varepsilon = b_\varepsilon$ with $A_\varepsilon = M + \varepsilon N$ symmetric positive definite. Instead of using extreme strategies as to apply the same preconditioner along the process, or on the contrary, to compute a different preconditioner for each value of ε, we propose an intermediate technique. It is based on the construction of one initial preconditioner from an incomplete factorization of M, which can be easily updated for each value of ε at a low computational cost.

Several numerical experiments are presented in order to show the efficiency of the proposed preconditioner.

Keywords: Incomplete Factorization, Shifted Linear Systems, Preconditioning, Conjugate Gradient, Iterative Methods, Wind Modelling.

1 Introduction

The resolution of several problems of science and engineering, such as parabolic partial differential equations, mass consistent models for wind field adjustment [1, 2], etc., with any discretization technique, yields linear systems of equations of the form,

$$(M + \varepsilon N)\, x_\varepsilon = b_\varepsilon \tag{1}$$

where M and N are constant for a given discretization. In these problems, the system (1) must be solved for different values of ε.
Iterative solvers based on Krylov subspaces are the most efficient methods for such large and sparse linear systems [3]. In our case, since M and N are symmetric positive definite matrices, the Conjugate Gradient (CG) provides the best results. In addition, the use of suitable preconditioning techniques [4] allows a faster convergence of CG.

For preconditioning these systems, we can build a different preconditioner for each value of ε. In general, this means to obtain good convergence behaviour but at a high computational cost related to each preconditioner. On the contrary, we can use a unique preconditioner, the first of the above list, for solving all the linear systems. However, this second strategy may lead to convergences as slow as the value of ε is far from the initial value ε_0 chosen for building the preconditioner.

In this work, an intermediate procedure is proposed. It consists of a preconditioner based on an incomplete Cholesky factorization that may be updated for each new system at a low computational cost. Thus, it provides better convergence than the latter strategy and is cheaper that the former. In a similar way, Meurant [5] proposes this preconditioner for the special case $(M + \varepsilon D)\, x_\varepsilon = b_\varepsilon$, with D being a diagonal matrix. In addition, Benzi [6] develops a preconditioner, based on a factorized approximate inverse [7], for shifted linear systems of the form $(M + \varepsilon I)\, x_\varepsilon = b_\varepsilon$, with I being the unit matrix. This preconditioner may be updated in function of ε.

The paper has been organized as follows. In section 2, the construction of the preconditioner is set in terms of an incomplete Cholesky factorization for the matrix $A_\varepsilon = M + \varepsilon N$, with the corresponding simplifications that allows its undating at a reasonable cost. Section 3 is devoted to describe the selected numerical experiments and the obtained results in order to show the performance of our updating procedure. Finally, the conclusions and some related future works are presented in section 4.

2 Updating of the incomplete Cholesky factorization

We will generalize the incomplete factorization proposed by Meurant [5] for the case of matrices $A_\varepsilon = M + \varepsilon D$, with D being diagonal, to matrices $A_\varepsilon = M + \varepsilon N$, with M and N being two $n \times n$ symmetric positive definite matrices. We can write A_ε as follows,

$$A_\varepsilon = (m_i j) + \varepsilon\, (n_i j) = \begin{pmatrix} m_{11} + \varepsilon n_{11} & (f_{1M} + \varepsilon f_{1N})^T \\ f_{1M} + \varepsilon f_{1N} & M_2 + \varepsilon N_2 \end{pmatrix}$$

where f_{1M}, f_{1N} represent $(n-1) \times 1$ column matrices and M_2, N_2, $(n-1) \times (n-1)$ matrices.

A factorization of the first row and column of A_ε is carried out,

$$A_\varepsilon =$$
$$\begin{pmatrix} m_{11} + \varepsilon n_{11} & \mathbf{0} \\ l_{1M} + \varepsilon l_{1N} & \mathbf{I} \end{pmatrix} \begin{pmatrix} (m_{11} + \varepsilon n_{11})^{-1} & \mathbf{0} \\ \mathbf{0} & C_2 \end{pmatrix} \begin{pmatrix} m_{11} + \varepsilon n_{11} & (l_{1M} + \varepsilon l_{1N})^T \\ \mathbf{0} & \mathbf{I} \end{pmatrix} =$$
$$L_1 Z_1 L_1^T$$

with $l_{1M} = f_{1M}$ and $l_{1N} = f_{1N}$.

Then, identifying term with term, we obtain for matrix C_2,

$$C_2 = M_2 + \varepsilon N_2 - \frac{1}{m_{11} + \varepsilon n_{11}} \left(l_{1M} + \varepsilon l_{1N}\right) \left(l_{1M} + \varepsilon l_{1N}\right)^T \tag{2}$$

If, in order to build the preconditioner, we consider only the diagonal entries of N as first approximation, equation (2) is simplified since $l_{1N} = 0$,

$$C_2 = \varepsilon D_2 + M_2 - \frac{1}{m_{11} + \varepsilon n_{11}} l_{1M} l_{1M}^T,$$

An order 0 algorithm is derived from,

$$C_2 = \varepsilon D_2 + M_2 - \frac{1}{m_{11}} l_{1M} l_{1M}^T$$

and the entries of C_2 are computed by adding εD_2 to what we would have obtained for the incomplete decomposition of M.
Another approximation consists of considering all the entries in N_2 and neglecting the products εl_{1N} in (2). So, the successive computations of matrices C_i do not involve ε and those may be obtained easily from the M decompositions,

$$C_2 = \varepsilon N_2 + M_2 - \frac{1}{m_{11}} l_{1M} l_{1M}^T$$

and thus, in matrix form,

$$C_2 = \varepsilon N_2 + \begin{pmatrix} m_{22}^{(2)} & f_{2M}^T \\ f_{2M} & M_3 \end{pmatrix} = \begin{pmatrix} m_{22}^{(2)} + \varepsilon n_{22} & \left(f_{2M} + \varepsilon l_{2N}\right) \\ f_{2M} + \varepsilon l_{2N} & M_3 + \varepsilon N_3 \end{pmatrix}$$

Only the entries of f_{2M} corresponding to non null entries of M are computed in order to avoid the fill-in, obtaining l_{2M}. So the decomposition of C_2 results,

$$C_2 \approx$$
$$\begin{pmatrix} m_{22}^{(2)} + \varepsilon n_{22} & \mathbf{0} \\ l_{2M} + \varepsilon l_{2N} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \left(m_{22}^{(2)} + \varepsilon n_{22}\right)^{-1} & \mathbf{0} \\ \mathbf{0} & C_3 \end{pmatrix} \begin{pmatrix} m_{22}^{(2)} + \varepsilon n_{22} & \left(l_{2M} + \varepsilon l_{2N}\right)^T \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$$

where, identifying,

$$C_3 = M_3 + \varepsilon N_3 - \frac{1}{m_{22}^{(2)} + \varepsilon n_{22}} \left(l_{2M} + \varepsilon l_{2N}\right) \left(l_{2M} + \varepsilon l_{2N}\right)^T$$

Similarly, with the same simplifications, we have,

$$C_3 = M_3 + \varepsilon N_3 - \frac{1}{m_{22}^{(2)}} l_{2M} l_{2M}^T = \begin{pmatrix} m_{33} + \varepsilon n_{33} & \left(f_{3M} + \varepsilon l_{3N}\right)^T \\ f_{3M} + \varepsilon l_{3N} & M_4 + \varepsilon N_4 \end{pmatrix}$$

that is constructed following the same procedure of C_2.

In this way, obtaining all the matrices C_i, the incomplete decomposition of A_ε results,

$$A_\varepsilon \approx L_1 Z_1 L_1^T = L_1 L_2 Z_2 L_2^T L_1^T = (L_1 L_2 \cdots L_n)\, Z_n\, (L_1 L_2 \cdots L_n)^T \tag{3}$$

being Z the diagonal matrix,

$$\begin{pmatrix} (m_{11} + \varepsilon n_{11})^{-1} & & & . & \\ & \left(m_{22}^{(2)} + \varepsilon n_{22}\right)^{-1} & & . & \\ & & \left(m_{33}^{(3)} + \varepsilon n_{33}\right)^{-1} & . & \\ . & . & . & . & . \\ & & & . & \left(m_{nn}^{(n)} + \varepsilon n_{nn}\right)^{-1} \end{pmatrix}$$

The diagonal entries of the lower triangular matrix $L_1 L_2 \cdots L_n$ are $m_{ii}^{(i)} + \varepsilon n_{ii}$. The respective columns below the diagonal entries are defined by $(n - i) \times 1$ matrices $l_{jM} + \varepsilon l_{jN}$.

3 Numerical experiments

In this section we present the results obtained using CG with the proposed preconditioners for solving the linear systems of equations arising from three PDE problems: a parabolic equation related to a 2-D heat transfer test problem with variable thermal conductivity, Vthcond, a convection-diffusion-reaction equation related to a 3-D active carbon filter problem, LightTruck [9], and an elliptic equation related to a 3-D mass consistent model for wind field adjustment, Windfield [1, 2]. All the experiments were carried out in a DELL Precision M60 computer using double precision Fortran. In the resolution, we always started from the null vector and stopped if $\|r_k\|_2 \leq 10^{-10} \|r_0\|_2$ or if the number of iterations was greater than 5000.

The results of different problems have been represented in several tables for a wide range of values of ε, including number of iterations and timings for reaching convergence. In tables, ICHOL$_D$ and ICHOL$_N$ represent the ICHOL preconditioners obtained with the approaches developed in section 2. These preconditioners are compared with Full-ICHOL of matrix A_ε, that is, computing a new ICHOL decomposition for each ε, and with the use of unique preconditioner, ICHOL(A_{ε_0}), along the whole process.

In all the experiment we start from the incomplete factorization of matrix M, i.e., $\varepsilon_0 = 0$. Thus in the following we consider. In all the experiments, the results are shown in several tables with the iterations and the computational cost of CG for different values of ε.

3.1 Example 1: Vthcond

This experiment involves a 2-D heat transfer problem modelled with a parabolic PDE,

$$\begin{aligned} \frac{\partial u}{\partial t} - (c+\delta c)\Delta u &= f && && (4) \\ u &= u_d && \text{on} \quad \Gamma \times (0,T] && (5) \\ u(x,0) &= u^0 && \text{in} \quad \Omega && (6) \end{aligned}$$

where u represents the temperature, c is the thermal conductivity, δc is a perturbation on the thermal conductivity and f is a constant source term.

We have solved this problem for a 2-D domain defined by vertices $A(0,0)$, $B(3,0,$ $C(3,3)$, $D(2,3)$, $E(2,2)$ and $F(0,2)$. We discretize in space with finite differences with a stepsize h and a time implicit scheme with a time step k. Then we obtain

$$\left(\frac{1}{k}I + \frac{c}{h^2}R + \frac{\delta c}{h^2}R\right) u^{n+1} = \frac{u^n}{k} + f^{n+1}$$

If we define

$$M = \frac{1}{k}I + \frac{c}{h^2}R, \qquad N = \frac{1}{h^2}R,$$

the matrix of the problem is,

$$A_{\delta c} = M + \delta c N$$

So in this experiment ε is identified with δc.

The stepsize was $h = 0.02$, the time step $k = 0.001$, the source $f = 1$ and $u_d = u^0 = 0$. We have work with a linear system corresponding to an intermediate time step of the process with 17201 unknowns, changing the perturbation of $c = 0.1$ from 10^{-6} to 10^6.

We remark that in this type of matrices, with a few non zero entries (here we have a maximum number of 5 non zero entries per row), the computational cost of the incomplete Cholesky factorization with the same sparsity pattern as the system matrix is very low. This effect is shown in table 1. In fact, all the strategies perform similarly for $\varepsilon < 1$ since the number of iterations of CG are exactly the same independently of the used preconditioner. For $\varepsilon \geq 1$, the Full-ICHOL preconditioner reaches convergence faster. Only from $\varepsilon = 1$ to $\varepsilon = 10$, CG-ICHOL$_N$ may compete with the CG-Full-ICHOL computational cost.

Another remarkable result is that ICHO$_D$ works the worst for $\varepsilon \geq 1$. Even the ICHOL(A_{ε_0}) reaches convergence faster than it. In this problem, the explanation is that adding εD with $\varepsilon \geq 1$ to the incomplete Cholesky factorization of M yields the loss of the diagonal dominant property of the preconditioned matrix, and thus, the quality of the preconditioning is decreased.

ε		ICHOL(A_{ε_0})	ICHOL$_D$	ICHOL$_N$	Full-ICHOL
0	n^{o}Iter.	–	–	–	6
	t(s)	–	–	–	0.03
10^{-6}	n^{o}Iter.	6	6	6	6
	t(s)	0.03	0.03	0.03	0.03
10^{-5}	n^{o}Iter.	6	6	6	6
	t(s)	0.03	0.03	0.03	0.03
10^{-4}	n^{o}Iter.	6	6	6	6
	t(s)	0.03	0.03	0.03	0.03
10^{-3}	n^{o}Iter.	6	6	6	6
	t(s)	0.03	0.03	0.03	0.03
10^{-2}	n^{o}Iter.	6	6	6	6
	t(s)	0.03	0.03	0.03	0.03
10^{-1}	n^{o}Iter.	7	7	7	7
	t(s)	0.03	0.03	0.03	0.03
1	n^{o}Iter.	11	15	9	8
	t(s)	0.05	0.06	**0.04**	**0.04**
10	n^{o}Iter.	31	49	19	17
	t(s)	0.11	0.16	**0.06**	0.07
10^2	n^{o}Iter.	95	159	57	48
	t(s)	0.32	0.57	0.19	**0.18**
10^3	n^{o}Iter.	234	395	141	118
	t(s)	0.78	1.31	0.48	**0.40**
10^4	n^{o}Iter.	302	512	181	152
	t(s)	1.00	1.70	0.62	**0.51**
10^5	n^{o}Iter.	313	529	188	158
	t(s)	1.04	1.75	0.63	**0.54**
10^6	n^{o}Iter.	314	535	189	159
	t(s)	1.04	1.78	0.64	**0.54**

Table 1: Example 1, 17201 equations: Number of iterations and computational cost (in s.) of Conjugate Gradient with different preconditioners

3.2 Example 2: LightTruck

This problem corresponds to a numerical simulation of a 3-D active carbon filter [9], which is formulated by the following convection-diffusion-reaction equation, of a 3-D active carbon filter [9], which is formulated by the following convection-diffusion-reaction equation,

$$\frac{\partial u}{\partial t} + v \cdot \nabla u - \nu \Delta u + \sigma(u)\, u = f(u) \tag{7}$$

$$u = u_{input} \quad \text{on} \quad \Gamma_a \times (0,T] \tag{8}$$

$$\nabla u \cdot n = 0 \quad \text{on} \quad \Gamma_b \times (0,T] \tag{9}$$

$$u(x,0) = u^0 \quad \text{in} \quad \Omega \tag{10}$$

where u is the concentration of hydrocarbons in the air, v is a constant non-uniform velocity field of the air which is previously computed by solving a potential flow problem (for instance, porous media flow combined with potential flow for active-carbon filters, see [9]), and $\nu > 0$ is the diffusivity coefficient. The reaction term $\sigma(u)\, u$ and source $f(u)$ are strongly nonlinear. T is the final time of analysis. This PDE is complemented with Dirichlet and Neumann boundary conditions (eqs. 8, 9) and initial conditions (eq. 10). In these equations, $c_{input}(x,t)$ is the prescribed concentration on the Dirichlet boundary Γ_a, n is the outward unit normal vector and $u^0(x)$ is the prescribed initial concentration.

With a finite element discretization of this problem, we obtain a linear system of 17914 equations with the same SPD matrix for each time step of the transient process. Among them, we have selected one system corresponding to an intermediate time. In order to obtain a shifted linear system, a perturbation of the matrix was carried out, such that, in our example M was equal to the original matrix of the convection-diffusion-reaction problem and N was constructed such that it is also SPD,

$$N = M \odot R \qquad \text{with} \qquad R \begin{cases} r_{ij} = r_{ji} & \text{if} \quad i \neq j \\[2ex] r_{ii} > \displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{n} r_{ij} & \end{cases}$$

where $\odot$ represents the Hadamard product [10] and each r_{ij} is randomly generated such that $0 < r_{ij} < 1$.

Table 2 shows the results obtained with CG for each preconditioning technique. In this case, the cost of the incomplete Cholesky factorization is still low but it may be appreciated in the timings (about 0.05 seconds). The first conclusion is that for small values of ε all the preconditioners lead to the same number of CG iterations. In such cases, ICHOL(A_{ε_0}) is preferable since it is the cheapest. From ε equal to 10^{-2}, the updated preconditioners perform better. More precisely, ICHOL$_N$ reaches the faster convergence. In addition, the number of CG iterations with ICHOL$_N$ are lower than those of Full-ICHOL. Thus, here we have an example where updating is more robust that re-computing. The results obtained with ICHOL$_D$ are also good but worse than ICHOL$_N$. Finally, the use of ICHOL(A_{ε_0}) is not a good choice for high values of ε.

3.3 Example 3: Windfield

This wind model [1] is based on the continuity equation for an incompressible flow with constant air density in the domain Ω and *no-flow-through* boundary conditions on the terrain Γ_b,

$$\vec{\nabla} \cdot \vec{u} = 0 \qquad \text{in } \Omega \tag{11}$$
$$\vec{n} \cdot \vec{u} = 0 \qquad \text{on } \Gamma_b \tag{12}$$

ε		ICHOL(A_{ε_0})	ICHOL$_D$	ICHOL$_N$	Full-ICHOL
0	n^{o}Iter.	–	–	–	75
	t(s)	–	–	–	1.02
10^{-6}	n^{o}Iter.	77	77	77	77
	t(s)	**0.96**	0.97	0.98	1.03
10^{-5}	n^{o}Iter.	82	82	82	82
	t(s)	1.02	**1.00**	1.03	1.09
10^{-4}	n^{o}Iter.	80	81	80	81
	t(s)	**0.99**	1.03	1.01	1.09
10^{-3}	n^{o}Iter.	62	62	62	62
	t(s)	**0.78**	0.80	0.79	0.86
10^{-2}	n^{o}Iter.	34	33	33	33
	t(s)	0.45	**0.43**	0.44	0.49
10^{-1}	n^{o}Iter.	40	15	15	15
	t(s)	0.51	**0.21**	0.22	0.27
1	n^{o}Iter.	125	11	9	8
	t(s)	1.54	0.16	**0.14**	0.18
10	n^{o}Iter.	362	11	6	7
	t(s)	4.41	0.16	**0.11**	0.16
10^2	n^{o}Iter.	546	11	6	8
	t(s)	6.71	0.16	**0.11**	0.19
10^3	n^{o}Iter.	566	11	6	8
	t(s)	6.92	0.16	**0.11**	0.19
10^4	n^{o}Iter.	585	11	6	8
	t(s)	7.13	0.17	**0.11**	0.18
10^5	n^{o}Iter.	560	13	6	8
	t(s)	6.82	0.21	**0.11**	0.18
10^6	n^{o}Iter.	620	14	7	9
	t(s)	7.54	0.22	**0.12**	0.19

Table 2: Example 2, 17914 equations: Number of iterations and computational cost (in s.) of Conjugate Gradient with different preconditioners

The problem is formulated as a least-square approach in Ω, with $\vec{u}(\widetilde{u},\widetilde{v},\widetilde{w})$ to be adjusted

$$E(\vec{u}) = \int_\Omega \left[\alpha_1^2\left((\widetilde{u}-u_0)^2+(\widetilde{v}-v_0)^2\right)+\alpha_2^2\left(\widetilde{w}-w_0\right)^2\right]\, d\Omega \qquad (13)$$

where the interpolated wind $\vec{v}_0 = (u_0, v_0, w_0)$ is obtained from experimental measurements and physical considerations, and α_1, α_2 are the Gauss precision moduli. In practice, we use the so called stability parameter of the wind model,

$$\alpha = \frac{\alpha_1}{\alpha_2} \qquad (14)$$

since the minimum of the functional given by (13) is the same if we divide it by α_2^2.

The variational approach results in the following elliptic problem,

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} + \alpha^2 \frac{\partial^2 \phi}{\partial z^2} = -2\alpha_1^2 \left(\frac{\partial u_0}{\partial x} + \frac{\partial v_0}{\partial y} + \frac{\partial w_0}{\partial z} \right) \text{ in } \Omega \tag{15}$$

We consider Dirichlet condition for *flow-through* boundaries and Neumann condition for terrain and top

$$\phi = 0 \text{ on } \Gamma_a \tag{16}$$

$$\vec{n} \cdot T \vec{\nabla} \mu = -\vec{n} \cdot \vec{v}_0 \text{ on } \Gamma_b \tag{17}$$

with $T = diag \left[\frac{1}{2\alpha_1^2}, \frac{1}{2\alpha_1^2}, \frac{1}{2\alpha_2^2} \right]$. Note that in this experiment $\varepsilon = \alpha^2$.

This example is related to a wind simulation in a region of La Palma Island. Matrices M and N are given from a numerical modelling with the above mass consistent model for wind field adjustment proposed in [2]. We have used three different meshes to produce linear systems of 17991, 43954 and 98999 equations, respectively, with M and N SPD matrices.

The results of the first set of linear system can be seen in table 3. Similar conclusions that for the previous example may be reached. From $\varepsilon = 10^{-6}$ to $\varepsilon = 10^{-2}$, the ICHOL(A_{ε_0}) seems to be sufficient to reach convergence at a lowest cost. From $\varepsilon = 10^{-1}$ to $\varepsilon = 1$, the faster strategy is Full-ICHOL, but for $\varepsilon > 1$, ICHOL$_N$ get the best results, since Full-ICHOL does not allow to reach convergence. This fact is due to the special structure of the matrices which makes the incomplete factorization algorithm not work properly. Also ICHOL(A_{ε_0}) leads to a slow convergence. So, for high values of ε, both preconditioners proposed here have the best performance, with ICHOL$_N$ being preferable.

The results obtained for 43954 and 98999 equations are shown in tables 4 and 5. The conclusions are the same again. For small values of ε it is not necessary to update the initial incomplete factorization since it reaches convergence at the lowest cost. However, when ε is high, the ICHOL$_N$ preconditioner have the best behaviour. Only in the surrounding of $\varepsilon = 1$, the re-computing of the incomplete factorization seems to be advisable. In the case of 98999 equations, none of the preconditioners allowed to reach convergence with CG, and thus these values have been eliminated in table 5.

ε		ICHOL(A_{ε_0})	ICHOL$_D$	ICHOL$_N$	Full-ICHOL
0	n^{o}Iter.	–	–	–	155
	t(s)	–	–	–	2.00
10^{-6}	n^{o}Iter.	155	157	157	155
	t(s)	**1.92**	1.94	1.95	1.99
10^{-5}	n^{o}Iter.	157	157	171	170
	t(s)	**1.93**	1.94	2.13	2.17
10^{-4}	n^{o}Iter.	157	155	155	157
	t(s)	1.93	**1.92**	1.93	2.01
10^{-3}	n^{o}Iter.	166	166	166	166
	t(s)	**2.04**	2.05	2.08	2.10
10^{-2}	n^{o}Iter.	127	128	127	127
	t(s)	**1.56**	**1.58**	1.59	1.63
10^{-1}	n^{o}Iter.	148	117	106	101
	t(s)	1.81	1.46	1.34	**1.33**
1	n^{o}Iter.	279	197	105	81
	t(s)	3.43	2.43	1.32	**1.09**
10	n^{o}Iter.	676	475	200	272
	t(s)	8.24	5.82	**2.48**	3.40
10^2	n^{o}Iter.	2004	1183	407	>5000
	t(s)	24.53	15.51	**5.03**	–
10^3	n^{o}Iter.	3056	1794	604	>5000
	t(s)	37.5	21.96	**7.42**	–
10^4	n^{o}Iter.	3337	1926	647	>5000
	t(s)	40.91	23.63	**7.96**	–
10^5	n^{o}Iter.	3363	1990	655	>5000
	t(s)	41.05	24.46	**8.07**	–
10^6	n^{o}Iter.	3473	1926	650	>5000
	t(s)	42.42	23.71	**7.99**	–

Table 3: Example 3, 17991 equations: Number of iterations and computational cost (in s.) of Conjugate Gradient with different preconditioners

ε		ICHOL(A_{ε_0})	ICHOL$_D$	ICHOL$_N$	Full-ICHOL
0	n^{o}Iter.	–	–	–	184
	t(s)	–	–	–	6.21
10^{-6}	n^{o}Iter.	184	184	184	184
	t(s)	**5.96**	5.99	6.00	6.21
10^{-5}	n^{o}Iter.	184	184	184	184
	t(s)	**5.96**	5.99	6.00	6.23
10^{-4}	n^{o}Iter.	184	184	184	184
	t(s)	**5.98**	5.99	6.01	6.21
10^{-3}	n^{o}Iter.	181	181	181	181
	t(s)	**5.89**	5.90	5.91	6.12
10^{-2}	n^{o}Iter.	170	170	170	169
	t(s)	**5.55**	5.56	5.56	5.73
10^{-1}	n^{o}Iter.	148	135	131	126
	t(s)	4.81	4.43	**4.29**	4.35
1	n^{o}Iter.	232	149	105	78
	t(s)	7.50	4.88	3.46	**2.79**
10	n^{o}Iter.	454	303	145	76
	t(s)	14.66	9.84	4.74	2.73
10^2	n^{o}Iter.	995	675	261	>5000
	t(s)	32.09	22.03	**8.46**	–
10^3	n^{o}Iter.	1452	965	354	>5000
	t(s)	46.58	31.29	**11.50**	–
10^4	n^{o}Iter.	1583	1049	384	>5000
	t(s)	50.73	33.98	**12.45**	–
10^5	n^{o}Iter.	1604	1059	388	>5000
	t(s)	51.40	34.25	**12.57**	–
10^6	n^{o}Iter.	1605	1060	388	>5000
	t(s)	51.43	34.29	**12.58**	–

Table 4: Example 3, 43954 equations: Number of iterations and computational cost (in s.) of Conjugate Gradient with different preconditioners

ε		ICHOL(A_{ε_0})	ICHOL$_D$	ICHOL$_N$	Full-ICHOL
0	n^{o}Iter.	–	–	–	201
	t(s)	–	–	–	16.81
10^{-6}	n^{o}Iter.	201	201	201	201
	t(s)	**16.14**	16.16	16.19	16.82
10^{-5}	n^{o}Iter.	201	201	201	201
	t(s)	**16.15**	16.16	16.19	16.83
10^{-4}	n^{o}Iter.	201	201	201	201
	t(s)	**16.15**	16.16	16.19	16.83
10^{-3}	n^{o}Iter.	201	201	200	200
	t(s)	16.14	16.16	**16.11**	16.76
10^{-2}	n^{o}Iter.	188	191	189	189
	t(s)	**15.22**	15.35	15.24	15.87
10^{-1}	n^{o}Iter.	225	157	155	151
	t(s)	18.08	12.65	**12.52**	12.85
1	n^{o}Iter.	483	211	148	132
	t(s)	38.63	16.94	11.97	**11.33**
10	n^{o}Iter.	1350	540	259	236
	t(s)	107.71	43.14	20.91	**19.64**
10^2	n^{o}Iter.	3973	1466	593	>5000
	t(s)	317.16	116.86	**47.62**	–
10^3	n^{o}Iter.	>5000	3468	1269	>5000
	t(s)	–	277.11	**101.60**	–

Table 5: Example 3, 98999 equations: Number of iterations and computational cost (in s.) of Conjugate Gradient with different preconditioners

4 Conclusions

The proposed updating of the incompleted Cholesky factorization seems to be an efficient tool for improving the convergence of conjugate gradient algorithm in the resolution of shifted linear systems of equations. At least, there is a wide range of the parameter ε for which the proposed preconditioners lead to the fastest convergence in front of the use of the initial incomplete factorization or even the re-computing of it for each ε. This was generally true for high values of ε. However, when ε is very small, the initial decomposition is enough to reach best results. This phenomenon was expected since, in these situations, the perturbation on the matrix M may be neglected. In the surrounding of 1, the experiments that we have carried out do not allow to obtain a definitive conclusion about the best strategy. Probably, the re-computing of the incomplete factorization is the most reliable choice in such cases.

Acknowledgements

This work has been partially supported by the MEC of the Spanish Government and FEDER, Grant number CGL2004-06171-C03-02/CLI.

References

[1] G. Montero, R. Montenegro, J.M. Escobar, *‘A 3-D diagnostic model for wind field adjustment’*, J. Wind Eng. Ind. Aer.,74-76, 249-261, 1998.

[2] G. Montero, E. Rodríguez, R. Montenegro, J.M. Escobar, J.M. González- Yuste, *‘Genetic algorithms for an improved parameter estimation with local refinement of tetrahedral meshes in a wind model’*, Adv. Engng. Soft., 36, 3-10, 2005.

[3] Y.Saad, *‘Iterative methods for sparse linear systems’*, PWE Publishing Company. Boston, 1996.

[4] M. Benzi, *‘Preconditioning techniques for large linear systems: a survey’*, J. Comput Physics, 182, 418-477, 2002.

[5] G. Meurant, *‘On the incomplete Cholesky decomposition of a class of perturbed matrices’*, SIAM J. Sci. Comput., 23(2), 419-429, 2001.

[6] M. Benzi and D. Bertaccini, *‘Approximate inverse preconditioning for shifted linear systems’*, BIT Num. Math., 43, 231-244, 2003.

[7] M. Benzi, J.K. Cullum and M. Tůma,*‘Robust approximate inverse preconditioning for the conjugate gradient method’*, SIAM J. Sci. Comput., 22, 1318-1332, 2000.

[8] M. Benzi, C.D. Meyer and M. Tůma, *‘A sparse approximate inverse preconditioner for the conjugate gradient method"’*, SIAM J. Sci. Comput., 17(5), 1135-1149, 1996.

[9] A. Rodríguez y M.L. Sandoval, *‘Incomplete Cholesky factorizations for transient convection-diffusion problems’*, in proceedings of The Fourth International Conference on Engineering Computational Technology, B.H.V. Topping and C.A. Mota Soares, Eds. Civil-Comp. Press, 2004.

[10] O. Axelsson, *‘Iterative solution methods’*, Cambridge University Press, 1996.