An Augmented Subspace Conjugate Gradient

Jocelyne Erhel, Frédéric Guyomarc'H

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An augmented subspace Conjugate Gradient

Jocelyne Erhel , Frédéric Guyomarc'h

N ° 3278

Octobre 1997

THÈME 4

ISSN 0249-6399

INRIA
RENNES

An augmented subspace Conjugate Gradient

Jocelyne Erhel *, Frédéric Guyomarc'h †

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet ALADIN

Rapport de recherche n3278 — Octobre 1997 — 20 pages

Abstract: Many scientific applications require to solve successively linear systems $Ax = b$ with different right-hand sides b and a symmetric positive definite matrix A. The Conjugate Gradient method applied to the first system generates a Krylov subspace which can be efficiently recycled thanks to orthogonal projections in subsequent systems. A modified Conjugate Gradient method is then applied with a specific initial guess and initial descent direction and a modified descent direction during the iterations. This paper gives new theoretical results for this method and proposes a new version which seems robust as far as loss of orthogonality is concerned. Numerical experiments show the efficacy of our method even for quite different right-hand sides.

Key-words: Conjugate Gradient, Krylov subspace, orthogonal projection

(Résumé : tsvp)

* INRIA, UR Rennes, Jocelyne.Erhel@inria.fr, http://www.irisa.fr/aladin/perso/erhel.html
† INRIA, UR Rennes, Frederic.Guyomarch@inria.fr, http://www.irisa.fr/aladin/perso/fguyomar.html

Unit´e de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
T´el´ephone : 02 99 84 71 00 - International : +33 2 99 84 71 00
T´el´ecopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Gradient Conjugué avec sous-espace augmenté

Résumé : Dans beaucoup d'applications scientifiques, il faut résoudre successivement des systèmes linéaires du type $Ax = b$ avec différents seconds membres b et une matrice symétrique définie positive A. Lorsqu'on applique la méthode du Gradient Conjugué au premier système, on génère un sous-espace de Krylov que l'on peut recycler efficacement dans les systèmes suivants grâce à des projections orthogonales. On applique alors une méthode de Gradient Conjugué modifiée, où on définit une solution et une direction de descente initiales spécifiques et où on modifie la direction de descente durant les itérations. Ce rapport présente de nouveaux résultats théoriques pour cette méthode et propose une nouvelle version qui semble robuste par rapport à la perte d'orthogonalité. Les essais numériques montrent l'efficacité de notre méthode même pour des seconds membres très différents.

Mots-clé : Gradient Conjugué, sous-espace de Krylov, projection orthogonale

1 Introduction

In this paper we are concerned with the solution of several symmetric linear systems of the form

$$Ax^{(i)} = b^{(i)}$$

where A is a $N \times N$ symmetric positive definite matrix. If A is not too large, a direct method is competitive since it only requires one factorization for several triangular solves. However, if A is large, an iterative method such as the conjugate gradient (CG) is usually necessary because of memory constraints. But iterative methods require quite often a preconditioning to converge fast enough [2, 20]. Classical preconditioners such as Incomplete Cholesky IC(k) are still memory consuming. Moreover, they do not vectorize or parallelize well on supercomputers because of short length vectors with indirect addressing. The aim of the paper is to speed-up the convergence of CG in order to reduce the CPU cost. The method should keep memory requirements low and vectorize or parallelize easily. The main idea is to solve the first system by CG and to recycle in the subsequent systems the Krylov subspace $\mathcal{K}_m(A, s_0)$ generated in the first system. Similar ideas have been recently applied to non symmetric problems using GMRES algorithm [14, 5, 6, 1, 8, 7].

When the different right-hand sides are known a priori, then block-conjugate-gradient methods are efficient [15]. In [4], block-CG is combined with a seed projection method which generates a Krylov subspace for the so-called seed system and projects the residuals of other systems onto this subspace. Similar ideas are developed in [12, 16, 23]. This method is also extended to varying matrices [3].

A particular case arises when using Generalized-Cross Validation (GCV) to compute a regularizing parameter for ill-posed problems. This method involves a sequence of linear systems of the form $(A + \sigma^{(i)} I)x^{(i)} = b$ with the same right-hand side. In this case, the Krylov subspace generated by a Lanczos method is the same for any system and it can be used very efficiently along with recurrences to compute the quantities required by GCV [11, 21].

Here we assume that the different right-hand sides $b^{(i)}$ are computed sequentially. This situation arises for instance when a new right-hand side depends upon previous solutions. A first idea is to use previous systems to derive an initial guess for the current system. In [10], the current right-hand side is A-projected onto the subspace spanned by the l previous approximate solutions, where l is a user-chosen parameter. When A is symmetric and possibly indefinite, a similar idea to the seed projection is introduced in [17] and further analyzed in [19] and [25]. In a first step, this method computes an initial approximation by using a projection onto the Krylov subspace $\mathcal{K}_m(A, s_0)$. Saad [19] shows that this so-called restarted Lanczos-Galerkin method is in some sense equivalent to a block-Lanczos method. In a second step, the modified Lanczos method starts with the same initial approximation and uses also $\mathcal{K}_m(A, s_0)$ in the iterations. More precisely, the classical Lanczos procedure is modified so that the current Lanczos vector is orthogonal to $\mathcal{K}_m(A, s_0)$ and it appears sufficient to orthogonalize it against the last vector in $\mathcal{K}_m(A, s_0)$. In [18, 9], a similar approach is applied to the Conjugate Gradient method. A domain decomposition method leads to

a small interface problem which is solved by CG. The so-called Krylov correction method also recycles the first Krylov subspace $\mathcal{K}_m(A, s_0)$ by computing an initial approximation and by forcing orthogonality against $\mathcal{K}_m(A, s_0)$. But here, the current direction is orthogonalized against all previous directions to avoid loss of orthogonality due to rounding errors. Numerical experiments show a significant improvement over the classical CG.

In this paper, we analyze the Krylov correction method from a theoretical point of view and we give some numerical results. We first design an algorithm where we compute an initial approximate solution and initial descent direction by projecting onto the first Krylov subspace $\mathcal{K}_m(A, s_0)$. As in [19] for Lanczos method, we show that this method is in some sense equivalent to a block-conjugate-gradient algorithm. We then use the framework defined in [13] to design a new Balanced Projection Method defined by a solution space condition and a Petrov-Galerkin condition, which we call an Augmented Conjugate Gradient method. The solution space at step k is here spanned by $\mathcal{K}_m(A, s_0)$ and the previous residuals r_0, r_1, r_{k-1}. This algorithm combines the modified Lanczos and the Krylov correction techniques.

We show several new theoretical results for this method. We first prove that this Balanced Projection Method possesses a four-term recurrence, so that the overhead introduced is very small. We then introduce a polynomial formalism using polynomials in the two variables A and P, where P is the matrix of the A-orthogonal projection onto $\mathcal{K}_m(A, s_0)^{\perp_A}$. Using this framework, we show that the solution space is a direct sum of the two Krylov subspaces $\mathcal{K}_m(A, s_0)$ and $\mathcal{K}_k(P^*AP, r_0)$. This allows us to derive an error bound involving the condition number of the matrix P^*AP.

The paper is organized as follows : section 2 defines our new Augmented Conjugate Gradient method. Section 3 gives a polynomial formalism and proves the main theoretical results. Section 4 deals with some practical considerations, such as complexity, memory requirements and numerical stability. Finally, section 5 presents numerical results.

We introduce some notation which will be useful throughout the paper. We note (x, y) the scalar product between the two vectors x and y of $\mathbb{R}^N$; the transpose of the matrix B is noted by B^*. $Span(B)$ denotes the subspace spanned by the column vectors of B. $\mathbb{R}[X]$ denotes the set of polynomials in one variable ; $\mathbb{R}_j[X]$ denotes the set of polynomials in one variable of degree at most j ; $\mathbb{R}\langle X, Y\rangle$ denotes the set of polynomials in two non commutative variables.

2 Augmented Conjugate Gradient

Let A a symmetric definite positive matrix of $\mathbb{R}^{N,N}$ and let

$$Ay = c, \tag{1}$$

$$Ax = b \tag{2}$$

two successive linear systems to solve.

We assume that a classical Conjugate Gradient algorithm (CG) is used to solve the first system (1). Below is the algorithm to set the notation :

ALGORITHM 1: CG1

* iterative solution of $Ay = c$;
* Initialisation ;
choose y_0 ;
$s_0 = c - Ay_0$;
$w_0 = s_0$;
* Iteration ;
for $j = 0, 1, \ldots$ **until** convergence **do**
  $\gamma_j = (s_j, s_j)/(w_j, Aw_j)$;
  $y_{j+1} = y_j + \gamma_j w_j$;
  $s_{j+1} = s_j - \gamma_j A w_j$;
  $\delta_{j+1} = (s_{j+1}, s_{j+1})/(s_j, s_j)$;
  $w_{j+1} = s_{j+1} + \delta_{j+1} w_j$;
endo

Let m be the number of iterations and

$$S = (s_0, s_1, \ldots, s_m), \quad W = (w_0, w_1, \ldots, w_m)$$

the set of residuals and descent directions generated. Recall that

$$S^*S = \Delta,\ W^*AW = D,\ span(S) = span(W) = \mathcal{K}_m(A, s_0), \tag{3}$$

where Δ and D are diagonal matrices and $\mathcal{K}_m(A, s_0)$ is the Krylov subspace of dimension $m+1$ generated by the initial residual s_0.

We now want to use this information to speed-up the solution of the second system (2).

2.1 Initial guess

A first idea, introduced in [17, 19, 25] for Lanczos method, is to choose an initial guess x_0 such that the initial residual $r_0 = b - Ax_0$ is orthogonal to the Krylov subspace $\mathcal{K}_m(A, s_0)$ and an initial descent direction p_0 conjugate to the descent directions W. We thus want to enforce the orthogonality conditions

$$W^*r_0 = 0, W^*Ap_0 = 0. \tag{4}$$

Let x_{-1} be an initial approximate solution and $r_{-1} = b - Ax_{-1}$. To guarantee (4), we choose

$$x_0 = x_{-1} + WD^{-1}W^*r_{-1}, \quad r_0 = b - Ax_0, \quad p_0 = (I - WD^{-1}(AW)^*)r_0. \tag{5}$$

We then use the Conjugate Gradient algorithm as usual. However, the initial choice of p_0 is not usual, so that the algorithm, which we call InitCG, is no longer, strictly speaking, a Conjugate Gradient method. It is in fact a non-polynomial projection method, as defined in [13]. However, we now prove that this algorithm is equivalent to a Block Conjugate Gradient method, following the proof of [19] for Lanczos method.

Theorem 2.1 *Let n the number of iterations in the second resolution. For $n \leq m/2$, the Conjugate Gradient algorithm applied to the linear system $Ax = b$, started with the initial guess r_0 and initial direction p_0 given by (5), is mathematically equivalent to the block-conjugate gradient algorithm with block size of 2 started with the block (s_0, r_0).*

Proof. We first prove that $W^* r_0 = 0$ and $W^* A p_0 = 0$.

Let $P = I - WD^{-1}(AW)^*$ be the matrix of the A-orthogonal projection onto $\mathcal{K}_m(A, s_0)^{\perp_A}$; $P^* = I - (AW)D^{-1}W^*$ is the A^{-1}-orthogonal projection onto $\mathcal{K}_m(A, s_0)^{\perp}$.

From the definitions (5), we get readily $r_0 = P^* r_{-1}$ and $p_0 = P r_0$ so that the result follows.

Let $S_j = (s_0, s_1, \ldots, s_j)$ and $R_i = (r_0, r_1, \ldots, r_i)$. We will show that $Span(S_j)$ is orthogonal to $Span(R_i) \quad \forall i, j$ such that $i + j \leq m$. The proof of the theorem immediately follows.

Let s and r be two vectors of $Span(S_j)$ and $Span(R_i)$ respectively. Thanks to (3), there exist a polynomial $Q \in \mathbb{R}_j[X]$ and a polynomial $P \in \mathbb{R}_i[X]$ such that $s = Q(A)s_0$ and $r = P(A)r_0$. Therefore

$$(s, r) = (Q(A)s_0, P(A)r_0)$$

and we only have to prove the orthogonality for monomials such as $A^j s_0$ and $A^i r_0$. This property comes from the choise of the initial residual r_0; indeed, since A is symmetric,

$$(A^j s_0 \ , \ A^i r_0) = (A^{j+i} s_0 \ , \ r_0).$$

Now, since $i + j \leq m$, $A^{j+i} s_0 \in \mathcal{K}_m(A, s_0) = Span(W)$ and we proved that $W^* r_0 = 0$ so we conclude that

$$(A^j s_0 \ , \ A^i r_0) = 0$$

and consequently $(s, r) = 0$. □

Therefore, for the first $m/2$ iterations, we get an accelerated scheme compared to a classical CG method.

2.2 Augmented CG

Another way to improve the resolution is to keep the orthogonality condition throughout the iterations. This method was developed for Lanczos method in first [17] and then [19]. It was also defined for Conjugate Gradient in [18, 9].

The idea here is to split the search for the minimum in two sub-spaces: The Krylov subspace $\mathcal{K}_m(A, s_0)$ already computed and another subspace which replaces the Krylov subspace we should have computed to solve the second system. A condition to satisfy is that both spaces are A-conjugate; this requirement guides our choice for the second subspace: at each iteration, we force the new descent direction to be A-conjugate with $\mathcal{K}_m(A, s_0)$. This orthogonality condition will appear to be satisfied by a recurrence relation, as in [19] and contrary to [18, 9], where the context required a full conjugaison with W.

We denote by x_k the current approximate solution, by r_k the current residual $r_k = b - Ax_k$, by $e_k = x - x_k$ the current error where x is the exact solution and by p_k the current descent direction.

We thus start with x_0, r_0, p_0 as defined by (5). The method is still a projection method onto a subspace which is no longer a Krylov subspace. This subspace, called the *solution space*, is defined by

$$\mathcal{K}_{m,k}(A, s_0, r_0) = span(s_0, s_1, \ldots, s_m, r_0, r_1, \ldots, r_k). \tag{6}$$

Our projection method is then defined by the solution space condition

$$x_{k+1} - x_k \in \mathcal{K}_{m,k}(A, s_0, r_0) \tag{7}$$

and by the Petrov-Galerkin condition

$$(r_{k+1}, z) = 0 \quad \forall z \in \mathcal{K}_{m,k}(A, s_0, r_0). \tag{8}$$

The algorithm, which we call "Augmented Conjugate Gradient (AugCG)", defined by (6) (7) and (8), is a Balanced Projection Method (BPM) based on the matrix $B = A$ as defined in [13]. Since A is HPD we therefore have immediately the following result.

Theorem 2.2 *Let A be a symmetric positive definite matrix. The algorithm AugCG applied to the linear system $Ax = b$ will not break down at any step. The approximate solution x_k is the unique minimizer of the error $\|e_k\|_A$ over the solution space $\mathcal{K}_{m,k}(A, s_0, r_0)$ and there exists $\epsilon > 0$ independent of e_0 such that for all k*

$$\|e_k\|_A \leq (1 - \epsilon) \ \|e_{k-1}\|_A.$$

Proof. see theorems 2.4, 2.6 and 2.7 in [13]. □

We will now prove that AugCG can be implemented with a four-term recurrence. The solution space condition (7) is satisfied as usual in the Conjugate Gradient. The Petrov-Galerkin condition (8) is satisfied simply with two orthogonality conditions : the current descent direction p_{k+1} must be orthogonal to the last descent direction p_k and to the last vector w_m respectively. This recurrence means that the complexity of the inner loop is similar to the classical Conjugate Gradient. Our new method adds merely one dot-product and one vector update at each iteration.

We have the following Augmented CG algorithm :

ALGORITHM 2: AugCG

* iterative solution of $Ax = b$;
* Initialisation ;
choose x_{-1} ;
$r_{-1} = b - Ax_{-1}$;
$x_0 = x_{-1} + WD^{-1}W^* r_{-1}$;

$r_0 = b - Ax_0$;
$p_0 = (I - WD^{-1}(AW)^*)r_0$;
* Iteration ;
for $k = 0, 1, \ldots$ **until** convergence **do**
$\alpha_k = (r_k, r_k)/(p_k, Ap_k)$;
$x_{k+1} = x_k + \alpha_k p_k$;
$r_{k+1} = r_k - \alpha_k Ap_k$;
$\beta_{k+1} = (r_{k+1}, r_{k+1})/(r_k, r_k)$;
$\mu_{k+1} = (r_{k+1}, Aw_m)/(w_m, Aw_m)$;
$p_{k+1} = r_{k+1} + \beta_{k+1} p_k - \mu_{k+1} w_m$;
endo

As shown in Algorithm 2, we introduce a new scalar μ_{k+1} and a new update for the descent direction p_{k+1}. Of course, x_0 and r_0 must satisfy (5), a condition which is guaranteed by the inintialisation step.

We now prove that this algorithm is equivalent with the method defined by (7) and (8).

Theorem 2.3 *Algorithm 2 is equivalent to the Balanced Projection Method AugCG defined by (7) and (8). More precisely,*

$$(z, r_{k+1}) = 0 \quad \textit{and} \quad (z, Ap_{k+1}) = 0, \quad \forall z \in \mathcal{K}_{m,k}(A, s_0, r_0).$$

Proof. Algorithm 2 readily satisfies the solution space condition (7). As in the classical Conjugate Gradient, it is easy to prove by induction that $(z, r_{k+1}) = 0$ and $(z, Ap_{k+1}) = 0, \quad \forall z \in Span(r_0, \ldots, r_k)$. So it is sufficient to prove the following assertions :

$$W^* r_{k+1} = 0, \; W^* Ap_{k+1} = 0. \tag{9}$$

We prove this by induction. For $k = 0$, we already proved in theorem 2.1 that $W^* r_0 = 0$ and $W^* Ap_0 = 0$.

Now we assume the assertion (9) true for k; So

$$W^* r_{k+1} = W^* r_k - \alpha_k W^* Ap_k = 0.$$

Now we prove that $Pr_{k+1} = r_{k+1} - \mu_{k+1} w_m$. We have

$$Pr_{k+1} = r_{k+1} - \mu_{k+1} w_m - \sum_{j=1}^{j=m-1} \frac{(r_{k+1}, Aw_j)}{(w_j, Aw_j)} w_j.$$

Now, for $j \leq m-1$, $Aw_j \in span(w_0, \ldots, w_{j+1}) \subset Span(W)$ so that $(r_{k+1}, Aw_j) = 0$, because $W^* r_{k+1} = 0$. Consequently, $Pr_{k+1} = r_{k+1} - \mu_{k+1} w_m$ and

$$W^* Ap_{k+1} = W^* APr_{k+1} + \beta_{k+1} W^* Ap_k = 0.$$

This completes the proof. □

3 Polynomial Formalism

We will now describe a polynomial version of the Augmented CG algorithm, which enables us to derive properties of the solution space $\mathcal{K}_{m,p}$ more easily. We use again the matrix $P = I - WD^{-1}(AW)^*$ of the A-orthogonal projection onto $\mathcal{K}_m(A, s_0)^{\perp_A}$ and the matrix $P^* = I - (AW)D^{-1}W^*$ of the A^{-1}-orthogonal projection onto $\mathcal{K}_m(A, s_0)^{\perp}$. We will express r_k and p_k with polynomials in two variables A and P. More precisely, we have the following result.

Theorem 3.1 *For $k \geq 0$, $r_k = \phi_k(A,P)r_0$ and $p_k = \psi_k(A,P)r_0$ with ϕ_k and $\psi_k \in \mathbb{R}\langle X, Y\rangle$. Furthermore $\phi_0 = 1$, $\psi_0 = Y$ and we have the following recurrence relations :*

$$\begin{aligned} \phi_{k+1} &= \phi_k - \alpha_k X\psi_k, \\ \psi_{k+1} &= Y\phi_{k+1} + \beta_{k+1}\psi_k. \end{aligned}$$

Proof. The values of ϕ_0 and ψ_0 come directly from (5). The recurrences are readily available from algorithm 2 ; it is easy to see that $\phi_{k+1} = \phi_k - \alpha_k X\psi_k$; recalling that $Pr_{k+1} = r_{k+1} - \mu_{k+1}w_m$ we get

$$\psi_{k+1} = Y\phi_{k+1} + \beta_{k+1}\psi_k.$$

□

We define a symetric bilinear form on $\mathbb{R}\langle X, Y\rangle \times \mathbb{R}\langle X, Y\rangle$ with :

$$(\phi|\psi) = (\phi(A,P)r_0, \psi(A,P)r_0)\,.$$

The coefficients α_k and β_k can be expressed by means of this bilinear form :

$$\alpha_k = \frac{(r_k, r_k)}{(p_k, Ap_k)} = \frac{(\phi_k|\phi_k)}{(\psi_k|X\psi_k)}; \tag{10}$$

$$\beta_k = \frac{(r_{k+1}, r_{k+1})}{(r_k, r_k)} = \frac{(\phi_{k+1}|\phi_{k+1})}{(\phi_k|\phi_k)}. \tag{11}$$

So we can rewrite the algorithm in a polynomial form :

ALGORITHM 3: AugCG polynomial version

* resolution of $Ax = b$;
* Initialisation ;
$\phi_0 = 1$;
$\psi_0 = Y$;
* Iterations ;
for $k = 0, 1, \ldots$ **until** convergence **do**
  $\alpha_k = (\phi_k|\phi_k)/(\psi_k|X\psi_k)$;
  $\phi_{k+1} = \phi_k - \alpha_k X\psi_k$;
  $\beta_{k+1} = (\phi_{k+1}|\phi_{k+1})/(\phi_k|\phi_k)$;
  $\psi_{k+1} = Y\phi_{k+1} + \beta_{k+1}\psi_k$.
endo

We can now simplify the polynomial expression by introducing polynomials in only one variable XY. Indeed, we have the following result.

Theorem 3.2 *For $k \geq 0$, there exists $P_k \in \mathbb{R}[X]$ and $Q_k \in \mathbb{R}[X]$ such that*

$$\phi_k(X,Y) = P_k(XY) \quad and \quad \psi_k(X,Y) = YQ_k(XY). \tag{12}$$

More precisely, ϕ_k is given by

$$\phi_{k+1} = \phi_k - \alpha_k \sum_{j=-1}^{k-1} \left(\prod_{l=k-j}^{k} \beta_l \right) XY\phi_{k-j-1}.$$

Proof. We will prove the theorem by induction :
For $k = 0$, since $\phi_0 = 1$ and $\psi_0 = Y$, it is true.
Assuming it is true at step k, then

$$\begin{aligned}
\phi_{k+1} &= \phi_k - \alpha_k X \psi_k \\
&= P_k(XY) - \alpha_k XYQ_k(XY) \\
&= P_{k+1}(XY). \\
\psi_{k+1} &= Y\phi_{k+1} + \beta_{k+1}\psi_k \\
&= YP_{k+1}(XY) + \beta_{k+1}YQ_k(XY) \\
&= Y(P_{k+1}(XY) + \beta_{k+1}Q_k(XY)) \\
&= YQ_{k+1}(XY).
\end{aligned}$$

Now we prove the recurrence formula for ϕ_k :

$$\begin{aligned}
\phi_{k+1} &= \phi_k - \alpha_k X(Y\phi_k + \beta_k\psi_{k-1}) \\
&= \phi_k - \alpha_k(XY\phi_k + \beta_k X\,(Y\phi_{k-1} + \beta_{k-1}\psi_{k-2})) \\
&= \phi_k - \alpha_k(XY\phi_k + \beta_k XY\phi_{k-1} + \beta_k\beta_{k-1}X\psi_{k-2}) \\
&= \dots \\
&= \phi_k - \alpha_k(XY\phi_k + \beta_k XY\phi_{k-1} + \dots + \beta_k \dots \beta_1 XY\phi_0).
\end{aligned}$$

□

This simple polynomial formulation leads to a new definition of the solution space $\mathcal{K}_{m,k}(A, s_0, r_0)$ as a direct sum of two Krylov spaces.

Theorem 3.3 *For $k \geq 0$, $Span(r_0, \dots, r_k) = \mathcal{K}_k(P^*AP, r_0)$ and*

$$\mathcal{K}_{m,k}(A, s_0, r_0) = \mathcal{K}_m(A, s_0) \overset{\perp_A}{\oplus} \mathcal{K}_k(P^*AP, r_0).$$

Proof. From theorem 3.2, it is easy to see that the polynomial P_k is of degree k, so that polynomials $(P_i)_{i=0\dots k}$ are independent and build a basis of $\mathbb{R}_k[X]$. Therefore

$$Span(r_0, \dots, r_k) = \mathcal{K}_k(AP, r_0).$$

Now, it is easy to show that $P^*AP = AP$ so that by induction $(P^*AP)^i r_0 = (AP)^i r_0$ and

$$Span(r_0, \dots, r_k) = \mathcal{K}_k(P^*AP, r_0).$$

The characterization of $\mathcal{K}_{m,k}$ follows immediately. □

We can now use classical results of conjugate gradient theory : our algorithm AugCG computes the minimum over $\mathcal{K}_{m,k}$ which is lower than the minimum over $\mathcal{K}_k(P^*AP, r_0)$.

Corollary 3.1 *Let κ_1 be the condition number of P^*AP ; the error at iteration k in algorithm AugCG satisfies*

$$\|e_k\|_A \leq 2\|e_0\|_A \left(\frac{\sqrt{\kappa_1}-1}{\sqrt{\kappa_1}+1}\right)^k .$$

This result means that the asymptotic rate of convergence of AugCG is better than for the classical CG because the condition number κ_1 of P^*AP is smaller than the condition number κ_0 of A.

4 Practical considerations

We first consider operations count. The initial guess is the most time-consuming overhead. Both computations of x_0 and p_0 are BLAS2 type operations (projection onto a subspace of size m). On the other hand, each iteration adds merely one dot-product and one vector update (BLAS1 type operations) which are very cheap. We emphasize that AugCG does not induce new matrix-vector products. Hence the global overhead introduced should be easily balanced by the reduction in the number of iterations. Moreover, all operations are of type BLAS2 or BLAS1 so that they easily vectorize or parallelize.

Let us examine now the memory requirements. Here too, the initial guess computation is the most expensive. It requires to store the m vectors W and the other m vectors AW to avoid re-calculation which would be too expensive. On the contrary, the iterations need no more than two extra-vectors (w_m and Aw_m). Hence it seems reasonable to use secondary storage for W and AW if the main memory is not large enough.

If we do not rely on secondary storage, the question of the choice of m and s_0 arises. This will be analyzed in numerical experiments.

Now, we must also analyze the effect of rounding errors. Quite often in Lanczos-type methods, rounding errors lead to a loss of orthogonality which implies poor convergence or divergence. This situation is analyzed for modified Lanczos method in [17] and [25]. In Augmented CG, we enforce the initial orthogonality conditions (4) and two orthogonality conditions during the iterations. As far as (4) is concerned we follow the scheme proposed in [25] for the initial residual r_0 which merely implements a Modified-Gram-Schmidt process. We extend this scheme to the initial descent direction p_0. This ensures numerical stability for both r_0 and p_0. Details are given in algorithm 4 below. During iterations, the new descent direction p_{k+1} is enforced to be A-orthogonal to the last vector in the previous system w_m and to the previous descent direction p_k. Here too, we apply a Modified Gram-Schmidt process, by first A-orthogonalizing against w_m then A-orthogonalizing against p_k. We finally get the practical implementation described in algorithm 4 below.

ALGORITHM 4: Stable-AugCG

* iterative solution of $Ax = b$;

* vectors w_j and Aw_j stored in secondary storage $j = 1, \ldots, m-1$;
* vectors w_m and Aw_m stored in main memory ;
* quantities (w_j, Aw_j) stored in a vector of lenght m ;
* Initialisation ;
choose x_{-1} ;
$x_0 = x_{-1}$;
$r_0 = b - Ax_0$;
for $j = 1$ **to** m **do**
    $x_0 = x_0 + (r_0, w_j)/(w_j, Aw_j)w_j$;
    $r_0 = r_0 - (r_0, w_j)/(w_j, Aw_j)Aw_j$;
endfor ;
$z_0 = r_0$;
for $j = 1$ **to** m **do**
    $z_0 = z_0 - (z_0, Aw_j)/(w_j, Aw_j)w_j$;
endfor ;
$p_0 = z_0$;
* Iteration ;
for $k = 0, 1, \ldots$ **until** convergence **do**
    $\alpha_k = (r_k, z_k)/(p_k, Ap_k)$;
    $x_{k+1} = x_k + \alpha_k p_k$;
    $r_{k+1} = r_k - \alpha_k Ap_k$;
    $\mu_{k+1} = (r_{k+1}, Aw_m)/(w_m, Aw_m)$;
    $z_{k+1} = r_{k+1} - \mu_{k+1} w_m$;
    $\beta_{k+1} = (r_{k+1}, z_{k+1})/(r_k, z_k)$;
    $p_{k+1} = z_{k+1} + \beta_{k+1} p_k$;
endfor

This algorithm is mathematically equivalent to the previous formulation (algorithm 2) so that all theoretical results still apply. But this implementation appears to be numerically stable, contrary to the previous one, as numerical experiments show.

5 Numerical Experiments

We report here results on a few test problems We present some numerical experiments which study the efficiency of our InitCG and AugCG methods. We report here results on a few test problems but we could draw the same conclusions from many other experiments. We implemented the method in Matlab, so we give only convergence results, since CPU times would not give realistic comparisons. We solve a first system $Ay = c$ then a second system $Ax = b$ for different matrices A and right-hand sides c and b.

Since we study only the convergence behaviour, we select diagonal matrices for which we can easily choose the eigenvalues. This is quite usual as noted in [25, 24]. We also run experiments with the matrix of the Laplacian discretized by a five-point finite difference

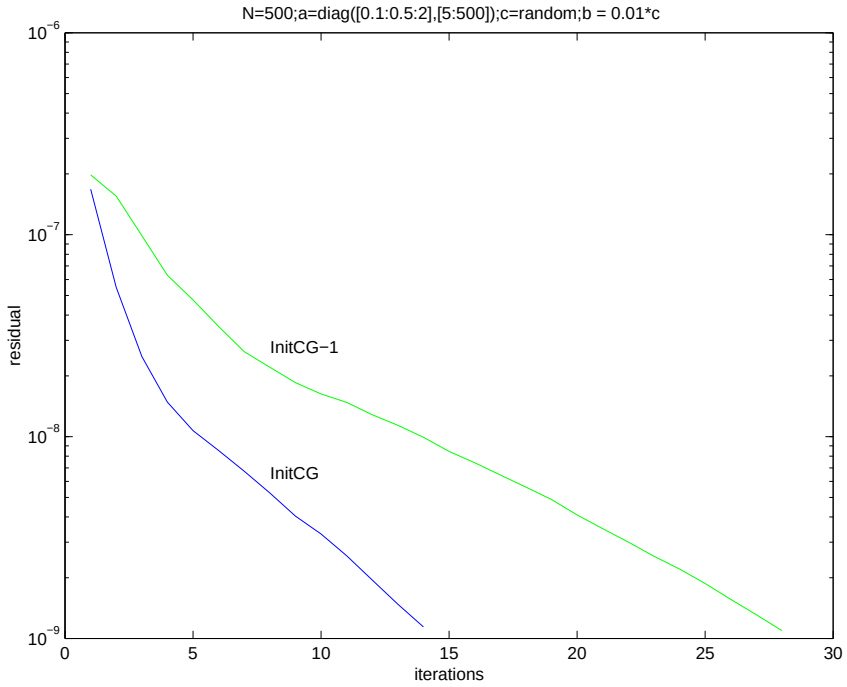

Figure 1: Results for example 1.

scheme over a square grid, because this matrix is very often used. For the right-hand side c, we choose a random vector. For the right-hand side b, we choose either a vector close to c by defining $b = c + u$ where u is a random vector with $\|u\|_\infty$ small or $b = \alpha c$ or a vector quite different (a random vector). The guess x_{-1} is taken as y_m, the solution obtained after m iterations in the first system. For these various systems, we also study the effect of the size m, that is to say the number of vectors $(w_0, w_1, \ldots, w_m)$ we keep from the first system. The different examples we report here are described in Table 1. D(random) means a diagonal matrix with random entries whereas $D([0.1 : 0.5 : 1.6], [5 : 500])$ means a diagonal matrix with entries $0.1, 0.6, 1.1, 1.6, 5, 6, 7, \ldots, 500$.

We implemented five different versions of InitCG and AugCG in order to study numerical stability. They are described in Table 2 which includes also classical CG for comparison. Unstable P^*r_{-1} and Pr_0 refers to the straightforward implementation for computing the initial residual r_0 and the initial descent direction p_0 as written in algorithm 2. On the contrary, stable P^*r_{-1} and Pr_0 refers to the implementation given in algorithm 4. Similarly, first $Pr_{k+1} + \beta_{k+1}p_k$ and modified $Pr_{k+1} + \beta_{k+1}p_k$ refer respectively to the implementation of iterative correction in algorithms 2 and 4. Finally, total $Pr_{k+1} + \beta_{k+1}p_k$ means an implementation using not only w_m but also all the vectors w_j as used in [9] for example.

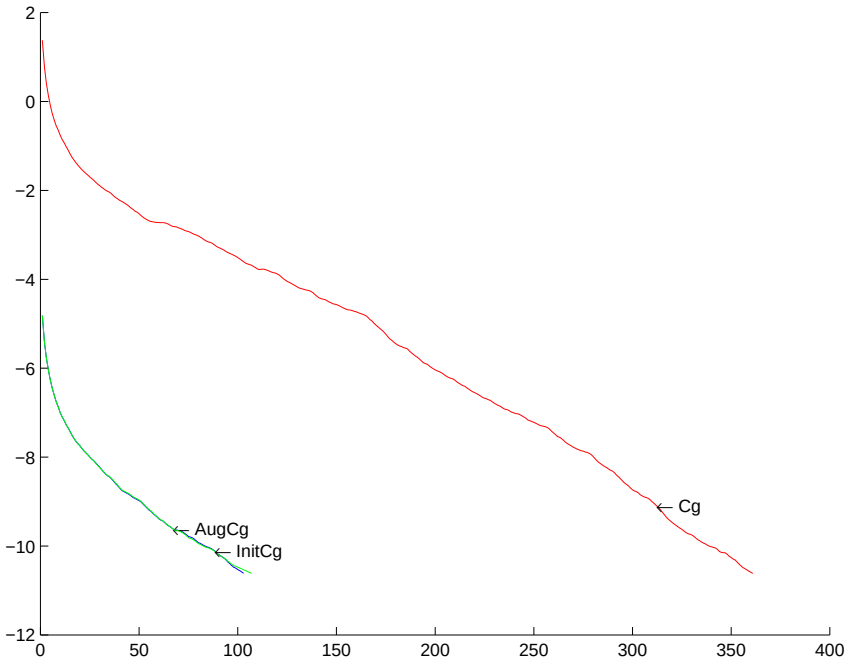

Figure 2: Results for example 2.

Table 1: Description of examples tested

example	N	matrix A	vector c	vector b	m
1	500	$D([0.1:0.5:1.6],[5:500])$	random	$0.01 \times c$	40
2	5000	D(random)	random	$b = c + u, \|u\|_\infty = 1.e-6$	50
3	5000	D(random)	random	random	50
4	900	Laplacian	random	random	80
5	5000	D(random)	random	$b = c + u, \|u\|_\infty = 1.e-6$	variable

All figures plot the norms of the residuals versus the number of iterations (which is here equal to the number of matrix-vector products).

We first study the numerical stability of the initial vectors computation. Figure 1 compares the convergence of InitCG-1 and InitCG for example 1. Here the gain due to the initial guess is very high because b and c are very close. As can be seen from Figure 1, there is a slight difference in the initialisation between InitCG and InitCG-1, which increases because

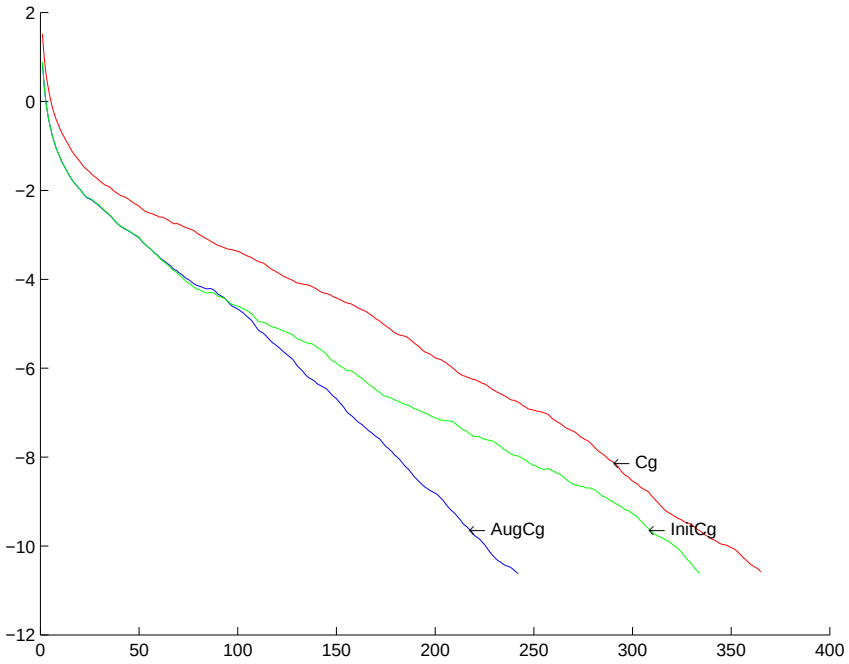

Figure 3: Results for example 3.

Table 2: Different InitCG and AugCG versions

version	initial residual	iterative correction
CG	r_{-1}	none
InitCG-1	unstable P^*r_{-1} and Pr_0	none
InitCG	stable P^*r_{-1} and Pr_0	none
AugCG-1	stable P^*r_{-1} and Pr_0	first $Pr_{k+1} + \beta_{k+1}p_k$
AugCG	stable P^*r_{-1} and Pr_0	modified $Pr_{k+1} + \beta_{k+1}p_k$
AugCG-total	stable P^*r_{-1} and Pr_0	total $Pr_{k+1} + \beta_{k+1}p_k$

of numerical instability in InitCG-1, as expected from section 4. Hence InitCG will be the reference for further comparisons.

We then study the numerical stability of the iterative correction. In all our experiments, we do not observe any difference between AugCG-1,AugCG and AugCG-all. Therefore we conclude that the three schemes are quite robust in many cases and we keep AugCG as our final algorithm.

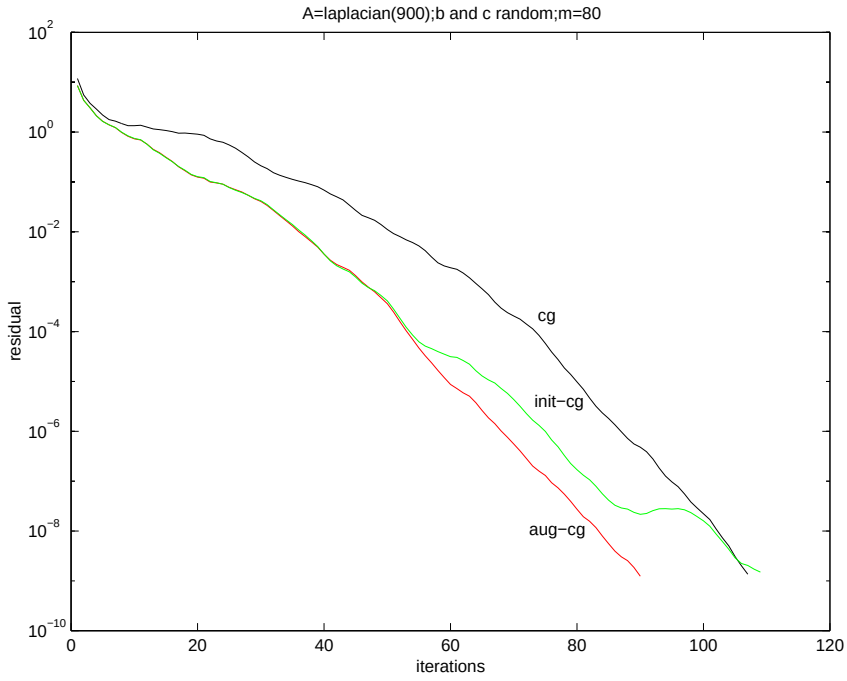

Figure 4: Results for example 4.

Now we compare InitCG with CG. The main difference comes from the initial choice. As long as $k \leq m/2$, there is some benefit in InitCG which is then similar to the block-CG(s_0, r_0) algorithm, as proved in Theorem 2.1. So InitCG is efficient when the second right-hand side b is close enough to the first one c. This is cleary illustrated on Figure 2 which compares InitCG, AugCG and CG for example 2.

Figures 3 and 4 show the convergence curves for examples 3 and 4. The behavior of AugCG and InitCG are similar during the first iterations, say about for $k \leq m/2$. However, the asymptotic behavior of InitCG and CG are quite similar. On the other hand, the asymptotic behavior of AugCG is much better, as expected from Corollary 3.1. When the right-hand sides b and c are not close, AugCG provides an impressive acceleration compared to InitCG and CG.

We finally study the impact of the Krylov subspace size m on the efficiency of AugCG. Clearly, m must be large enough in order to capture the smallest eigenvalues of A which mostly affect the convergence of CG [24, 22]. Nevertheless, memory constraints limit the size m: indeed, the initial guess computation requires the storage of m full vectors of length N. If secondary storage is used, the gain in iterations must balance the overhead due to I/O. Also the CPU cost increases with m for the initial guess computation. Moreover, the size $m+k$ of

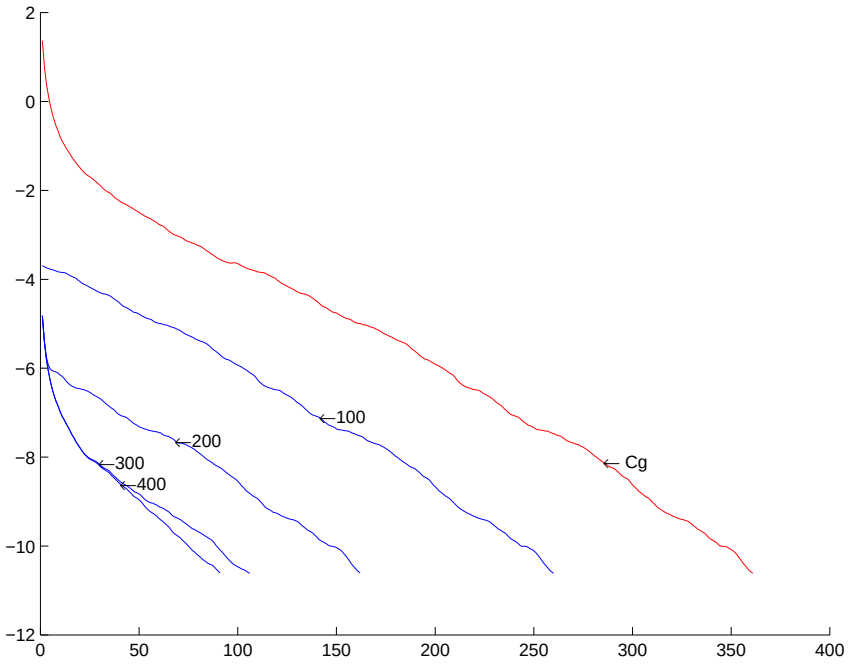

Figure 5: Results for example 5.

the solution space $K_{m,k}$ must stay smaller than N. Otherwise, loss of orthogonality occurs since $(W_m, p_0, p_1, \ldots, p_k)$ is no longer a basis and the method fails. Figure 5 shows the influence of m for example 5. As expected, the number of iterations decreases significantly when m increases. But the optimal choice of m yielding a minimal CPU time is still an open question.

6 Conclusion

This papers considers the problem of solving $Ax = b$ once another system $Ay = c$ has already been solved, where A is symmetric positive definite. The Conjugate Gradient applied to $Ay = c$ generates a Krylov subspace $K_m(A, s_0)$ which is used in the second system. The method InitCG computes an initial guess x_0, residual r_0 and an initial descent direction p_0 using a stable formulation of the A-orthogonal projections P and P^* onto respectively $\mathcal{K}_m(A, s_0)^{\perp_A}$ and $\mathcal{K}_m(A, s_0)^{\perp}$. The method AugCG not only starts with x_0 and p_0 but also modifies the descent direction p_{k+1} using a cheap and stable formulation. This paper shows that during the first $m/2$ iterations, InitCG is equivalent to a block-CG method started with the 2-block (s_0, r_0). It also shows that convergence of AugCG is governed by the condition number of P^*AP, so that AugCG converges asymptotically faster than CG. Numerical experiments demonstrate the efficiency of the method. In particular, we did not observe relevant loss of orthogonality for our stable variant.

Though this method is described here for only two systems, it can be easily extended to more, by involving all previous Krylov subspaces in the current system. However, the main drawback of this approach is the rapidly increasing memory requirement. But secondary storage is here possible for large systems because it would be used only in the initialisation part of AugCG. We are planning further experiments on large and realistic scientific problems in order to measure performances.

References

[1] J. Baglama, D. Calvetti, G. Golub, and L. Reichel, *Adaptively preconditioned GMRES algorithms*, tech. rep., Kent State University, 1996.

[2] A. Bruaset, *A survey of preconditioned iterative methods*, Pitman Research Notes in Mathematics Series, Longman Scientific and Technical, 1995.

[3] T. Chan and M. Ng, *Galerkin projection methods for solving multiple linear systems*, Tech. Rep. 96-31, UCLA, Sept. 1996.

[4] T. F. Chan and W. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM Journal on Scientific Computing, 18 (1997).

[5] A. Chapman and Y. Saad, *Deflated and augmented Krylov subspace techniques*, Supercomputer Institute Research report UMSI 95/181, University of Minnesota, 1995.

[6] R. Choquet, *Etude de la methode de Newton-GMRES - Application aux equations de Navier-Stokes compressibles*, thèse de doctorat, Université de Rennes 1, Dec. 1995.

[7] J. Erhel and K. Burrage, *On the performance of various adaptive preconditioned GMRES strategies*, Research report 1081, INRIA, Feb. 1997.

[8] J. Erhel, K. Burrage, and B. Pohl, *Restarted GMRES preconditioned by deflation*, Journal of Computational and Applied Mathematics, 69 (1996), pp. 303–318.

[9] C. Farhat, L. Crivelli, and F.-X. Roux, *Extending substructure based iterative solvers to multiple load and repeated analyses*, Computer methods in applied mechanics and engineering, 117 (1994), pp. 195–209.

[10] P. F. Fisher, *Projection techniques for iterative solution of Ax=b with successive right-hand sides.*, Tech. Rep. 93-90, ICASE, Dec. 1993.

[11] G. H. Golub and U. von Matt, *Generalized cross-validation for large scale problems*, Tech. Rep. TR-96-28, ETH, Zürich, Sept. 1996.

[12] P. Joly, *Résolution de systèmes linéaires avec plusieurs seconds membres par la méthode du gradient conjugué*, Tech. Rep. R-91012, Université Pierre et Marie Curie, Paris, Mar. 1991.

[13] W. D. Joubert and T. A. Manteuffel, *Iterative Methods for Nonsymmetric Linear Systems*, Academic Press, 1990, ch. 10, pp. 149–171.

[14] R. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. matrix Anal. App., 16 (1995), pp. 1154–1171.

[15] D. P. O'Leary, *The Block Conjugate Gradient Algorithm and Related Methods*, Linear Algebra and its Applications, 29 (1980), pp. 293–322.

[16] M. Papadrakakis and S. Smerou, *A new implementation of the Lanczos method in linear problems*, International Journal for Numerical Methods in Engineering, 29 (1990), pp. 141–159.

[17] B. Parlett, *A new look at the lanczos algorithm for solving symmetric systems of linear equations*, Linear algebra and its applications, 29 (1980), pp. 323–346.

[18] C. Rey, *Développement d'algorithmes parallèles de résolution en calcul non-linéaire de structures hétérogènes : Application au cas de la butée acier-élastomère.*, thèse de doctorat, Ecole normale supérieure de Cachan., 1994.

[19] Y. Saad, *On the lanczos method for solving symmetric linear systems with several right-hand sides*, Mathematics of computation, 178 (1987), pp. 651–662.

[20] ——, *Iterative methods for sparse linear systems*, PWS Publishing Company, 1996.

[21] R. Sidje and A. Williams, *Fast generalised cross validation*, in International Linear Algebra Year:Linear Algebra in Optimization, CERFACS, Albi-Toulouse, France, 1996.

[22] G. Sleijpen and A. van der Sluis, *Further results on the convergence behavior of Conjugate-Gradients and Ritz values*, Linear algebra and its applications, 246 (1996), pp. 233–278.

[23] C. SMITH, A. PETERSON, AND R. MITTRA, *A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields*, IEEE Transactions on Antennas and Propagation, 37 (1989), pp. 1490–1493.

[24] A. VAN DER SLUIS AND H. VAN DER VORST, *the rate of convergence of conjugate gradients*, Numerische Mathematik, 48 (1986), pp. 543–560.

[25] H. VAN DER VORST, *An iterative method for solving f(A)x=b using Krylov subspace information obtained for the symmetric positive definite matrix A*, Journal of Computational and Applied Mathematics, 18 (1987), pp. 249–263.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399