# An Adaptive sparse unsymmetric linear system solver

Miloud Sadkane, Roger B. Sidje

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tel:(1)39 63 55 11

Rapports de Recherche

1992

25ème anniversaire

N° 1788

***Programme 6***
*Calcul Scientifique, Modélisation et*
*Logiciels numériques*

# AN ADAPTIVE SPARSE UNSYMMETRIC LINEAR SYSTEM SOLVER

**Miloud SADKANE**
**Roger B. SIDJE**

**Novembre 1992**

*RR_1788*

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTEMES ALEATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CEDEX FRANCE
Tél. : 99 84 71 00 - Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

# An Adaptive Sparse Unsymmetric Linear System Solver

Miloud Sadkane and Roger B. Sidje
IRISA-INRIA
Campus de Beaulieu
35042 Rennes Cedex
France

Publication interne $n°679$ - Octobre 1992 - 28 pages
Programme 6

**Abstract**

Projection methods are the most widely used methods for computing a few of the extreme eigenvalues of large sparse matrices. Many of those algorithms have been turned out to solve linear systems of equations. In this paper we present an unsymmetric linear system solver based on projection techniques. We call this method adaptive because unlike usual Krylov subspace approximations which when receiving in entry an initial vector, they construct in a steady way a basis for the polynomial subspace where the approximated solution is sought, our approach follows up the construction process in order to enrich the basis with informations on the direction of the solution. Moreover it allows ad hoc actions to handle breakdowns internally. The convergence of the method is established when the symmetric part of the coefficient matrix is positive definite. Finally we illustrate the usefulness of the method using a number of representative PDE problems run on a CRAY-2.

**Key words.** Krylov subspaces, iterative refinement, Arnoldi's method, GMRES, preconditioning.

# Un solveur adaptatif pour la résolution de systèmes linéaires creux non symétriques de grande taille

**Résumé**

Nous étudions une méthode itérative pour la résolution de systèmes linéaires non symétriques creux de grande taille. A chaque pas, cette méthode minimise le résidu dans un sous-espace construit a partir des solutions approchées obtenues d'abord de manière similaire aux méthodes de sous-espaces et corrigées ensuite par un procéde du type raffinement itératif. Nous établissons la convergence de cette approche pour les matrices dont la partie symétrique est définie positive et comparons ses performances, sur un CRAY-2, à ceux de la méthode GMRES sur un ensemble varié de problèmes d'EDP.

**Mots clés.** Espaces de Krylov, raffinement iteratif, méthode d'Arnoldi, GMRES, préconditionnement.

# 1 Introduction

Consider the linear system

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

where $A$ is a real square non singular matrix of order $n$. There are several strategies for approximating the solution of (1) with respect to order, structure and properties of the coefficient matrix $A$. If the order $n$ of the system is reasonable i.e. if the dense matrix can fit in the main memory without performance overhead, one may certainly use direct methods for which efficient and reliable codes adapted to nowadays parallel and vector supercomputers are worldwide available on high quality software packages [3]. When $A$ is large and sparse, among a wide variety of iterative algorithms one can obtain a splitting $A = M - N$ and then apply a classical iterative method of the form

$$x_{i+1} = (M^{-1}N)x_i + M^{-1}f. \tag{2}$$

The recurrence scheme (2) leads to the solution of (1) if and only if the spectral radius of $M^{-1}N$ ($\rho(M^{-1}N)$) is less than one. In such case (2) is said to be consistent. Unfortunately a consistent and straightforward (i.e analytic) splitting is known for some special classes of matrices only.

Galerkin type methods, i.e. projection methods onto a subspace $K_i$ having an orthonormal basis $V_i = \{v_1, v_2, ..., v_i\}$ consist of looking for $x_i$ belonging to $K_i$ such that

$$P_i(Ax_i - f) = 0 \tag{3}$$

where $P_i$ is a projection onto $K_i$ either orthonormal ($P_i = V_i V_i^T$ with $V_i^T V_i = I$) or oblique. Usually $K_i$ is a Krylov subspace i.e. $K_i = \{v_1, Av_1, ..., A^{i-1}v_1\}$ and $V_i$ results from the well known Lanczos or Arnoldi process depending on whether or not the matrix is symmetric [8, 5]. Projection approximations are suitable for large sparse matrices and are now used in various matrix computation other than eigenvalue or linear system problems [9].

The Generalized Minimum Residual method (GMRES) [10] for solving the linear system (1) is one of the most versatile subspace projection algorithm for nonsymmetric matrices. It is derived from the Arnoldi process for constructing an $l_2$-orthogonal basis of Krylov subspaces and instead of solving the weak form (3), it minimizes the residual norm over the current Krylov subspace and presents the rare quality of avoiding any breakdown.

We propose two algorithms based on projection techniques : the first algorithm is derived in the same spirit as in Arnoldi's method in the sense that it builds gradually a small matrix $H$ using projection techniques and replaces the linear system (1) by the small one $Hy = z$. The second algorithm minimizes the residual on some subspace and therefore can be viewed as a GMRES like method. We discuss both theoretical and practical aspects of these two versions and show the connection between them. We present numerical experiments on some elliptic PDE problems run on a CRAY-2. We illustrate a major advantage of those approaches over GMRES on the convergence test when preconditioning is used.

In the sequel we denote by $S = \frac{A+A^T}{2}$ and by $N = \frac{A-A^T}{2}$ the symmetric part and the skew-symmetric part of $A$ respectively. We demote by $\bar{x}$ the exact solution of (1) and by $M$ an appropriated preconditioning matrix. $MGS$ stands for the Modified Gram-Schmidt procedure [5]. The integer $m$ fixes the maximum attainable size for the basis.

## 2 Algorithm SASLU

Choose an initial guess-vector $v_1$ and set $V_1 = [v_1]$
**for** $i = 1, 2, ...$
$W_i = AV_i$
$H_i = V_i^T W_i \qquad z_i = V_i^T f$
solve $H_i y_i = z_i$

compute $x_i = V_i y_i$ and $r_i = f - W_i y_i$
**if** convergence **stop**
$\hat{v}_{i+1} = M^{-1} r_i$
**if** $dim(V_i) < m$
$V_{i+1} = MGS[V_i, \hat{v}_{i+1}]$
**else**
$V_{i+1} = MGS[x_i, \hat{v}_{i+1}]$
**restart**
**endif**
**endfor**

The main idea behind the algorithm is that at each step an approximate solution is computed cheaply using the weak form (3), and this approximate solution is then corrected by one step of a splitting type method. Indeed, if $x_i = V_i y_i$ denotes an approximate solution obtained from the small system $H_i y_i = z_i$, then a natural idea for improving the accuracy of $x_i$ is to look for an $\tilde{x}_i$ satisfying $M\tilde{x}_i = (M - A)x_i + f$ (Jacobi, Gauss-Seidel, SOR, SSOR fall in this class of iterative scheme with special choices of $M$). The new approximate solution is then given by $\tilde{x}_i = x_i + M^{-1} r_i$ where $r_i = f - Ax_i$.

Another argument which motivates the algorithm relies upon the iterative refinement [7] which is based on the principle that giving $x_i$ approximating the true solution $\bar{x}$, if we denote by $\delta x_i$ the deviation vector between $\bar{x}$ and $x_i$, i.e $\bar{x} = x_i + \delta x_i$, we have

$$0 = f - A\bar{x} = f - A(x_i + \delta x_i) = r_i - A\delta x_i \Longrightarrow \delta x_i = A^{-1} r_i.$$

Therefore $x_{i+1} = x_i + \delta x_i = x_i + A^{-1} r_i$ is an improvement over $x_i$ provided that $\delta x_i$ is obtained accurately. This gives raise to the following algorithm known as iterative refinement scheme :

1. $x_1 = A^{-1} f$ ! in low accuracy
2. **for** $i = 1, 2, ...$
   2.1. $r_i = f - Ax_i$ ! in high accuracy
   2.2. $x_{i+1} = x_i + A^{-1} r_i$ ! in low accuracy
   **endfor**

By using $M^{-1}$ in place of $A^{-1}$ in statement 2.2 of this algorithm, we have considered $V_{i+1} = [V_i, M^{-1} r_i]$ as a relatively low costly improved basis ($x_i \in span(V_i)$). In [6], it is shown that if an iterative method is used in conjunction with iterative refinement, it is possible to prove under

certain conditions that after some steps of iterative refinement, the solution will be accurate to machine precision. Practically, for avoiding excessive growth in storage, we need to restart periodically. A natural choice of the new initial basis is $V_{i+1} = [x_i]$, but it is easy to show that the next resulting residual will remain the same as the previous one. Hence by choosing $V_{i+1} = [x_i, M^{-1}r_i]$ at restart, we gather those two steps into one and thus skip the stagnant step.

We finally note that solving the projected system $H_i y_i = z_i$ is equivalent to the Galerkin condition : $V_i$ and $r_i$ are orthogonal (i.e. $V_i^T r_i = 0$).

## 2.1 Programming Considerations

We stress on the fact that the algorithm evolves almost entirely by means of updatings. We mention also that the solution $x_i = V_i y_i$ is carried out only when convergence is reached or at restart since the residual is obtained independently.

### 2.1.1 Updating Details

A step, among many others, where an update is involved is the Modified Gram-Schmidt orthonormalization of $[V_i, M^{-1}r_i]$. Since $V_i$ was already orthonormalized, we just have to correct the last column. If we let $V_{i+1} = [V_i, v_{i+1}]$ after the orthonormalization, the following formulas hold.

$$W_{i+1} = [W_i, Av_{i+1}] \tag{4}$$

$$z_{i+1} = \begin{bmatrix} z_i \\ v_{i+1}^T f \end{bmatrix} \tag{5}$$

$$H_{i+1} = \begin{bmatrix} H_i & V_i^T A v_{i+1} \\ v_{i+1}^T A V_i & v_{i+1}^T A v_{i+1} \end{bmatrix} \tag{6}$$

The coefficient matrix is accessed only through matrix vector products, hence the algorithm is independent of the matrix data structure and it allows the user to design a high efficient matrix vector product with respect to the structure, the sparsity and the features of the system matrix.

At each iteration, the algorithm needs to solve the projected system $H_i y_i = z_i$. If the matrix $A$ was symmetric positive definite then $H_i$ would have the same property and the derived projected system would have been solved by means of Choleski updating [5]. For the general case where $A$ is unsymmetric, we perform an $LU$ updating. If $H_i = L_i U_i$ denotes the factorization at iteration $i$, we make use of it to obtain $H_{i+1} = L_{i+1} U_{i+1}$ with

$$L_{i+1} = \begin{bmatrix} L_i & 0 \\ l_i & \lambda_i \end{bmatrix}, \quad U_{i+1} = \begin{bmatrix} U_i & u_i \\ 0 & \mu_i \end{bmatrix} \tag{7}$$

and

$$\begin{cases} l_i &= v_{i+1}^T A V_i U_i^{-1} \\ u_i &= L_i^{-1} V_i^T A v_{i+1} \\ \lambda_i \mu_i &= v_{i+1}^T A v_{i+1} - l_i u_i \end{cases} . \tag{8}$$

As long as we assume that the system matrix $A$ is positive definite, i.e., the symmetric matrix $S$ is positive definite, the interaction matrix $H_i$ is also positive definite and therefore non singular. Observe that

$$\lambda_i \mu_i = v_{i+1}^T A P_i v_{i+1} \tag{9}$$

where

$$P_i \equiv I - V_i H_i^{-1} V_i^T A.$$

Normally $\lambda_i$ is set to 1 in the standard $LU$ factorization.

### 2.1.2 Complexity Analysis

Concerning the storage, let us remind that $m$ is the maximum size for the basis. We need extra space to keep the basis $V_i$ whose maximum order is $(n, m)$, the work array $W_i(n, m)$, the interaction matrix $H_i(m, m)$ and the projected right hand side vector $z_i$ whose maximum length is $m$. This gives the total increase of $m(2n + m + 1)$ which is of common order to projection methods. We will see further that a good choice of $m$ should not exceed $2\sqrt{n}$. Thus the storage requirement is at most of order $4n\sqrt{n}$.

Concerning the floating point operations, let $MATVEC$ denotes the complexity of evaluating the matrix $A$ times a vector, $MSOLVE$ the complexity of solving $M\hat{v}_{i+1} = r_i$ and $HSOLVE(i)$ the cost for solving the projected system at iteration $i$. We have the following complexity results.

| | | |
|---|---|---|
| $W_i = AV_i$ | by (4) | : $1\, MATVEC$ |
| $z_i = V_i^T f$ | by (5) | : $2n$ |
| $H_i = L_i U_i$ | by (8) | : $2i^2 + (2n-1)(2i-1)$ |
| $y_i = H_i^{-1} z_i$ | | : $HSOLVE(i)$ in $\mathcal{O}(i^2)$ |
| $x_i = V_i y_i$ | | : $2ni$ |
| $r_i = f - W_i y_i$ | | : $n + 2ni$ |
| $\hat{v}_{i+1} = M^{-1} r_i$ | | : $1\, MSOLVE$ |
| $V_{i+1} = MGS[V_i, \hat{v}_{i+1}]$ | | : $4ni + 3n$ . |

Therefore we infer that the overall complexity per iteration in SASLU is about

$$2i^2 + 2(6n-1)i + 4n + 1 + HSOLVE(i) + 1\, MATVEC + 1\, MSOLVE$$

## 2.2 Convergence analysis

Without loss of generality we can assume here that $M$ is the identity matrix $I$ (see remark 2.1). We assume also that $S$, the symmetric part of $A$, is positive definite. The following simple lemma is needed to analyze the convergence of SASLU.

**Lemma 2.1** *The matrix $P_i = I - V_i H_i^{-1} V_i^T A$ introduced in (9) satisfies :*

$$\begin{aligned} P_i^2 &= P_i \\ P_i V_i &= 0 \\ AP_i &= P_i^T AP_i \\ V_i^T AP_i &= 0 \\ P_i^T r_i &= r_i \\ P_i(I - V_i V_i^T) &= P_i \\ (I - V_i V_i^T)P_i &= I - V_i V_i^T. \end{aligned}$$

*$\mathbf{R}^n = A^{-1}[V_i^\perp] \oplus V_i$ and $P_i$ is the projection associated to this decomposition, i.e., $P_i$ is a projection onto $A^{-1}[V_i^\perp]$ along $V_i$ where $V_i^\perp$ denotes the orthogonal complement of $V_i$, i.e., $V_i^\perp = span(I - V_i V_i^T)$.*

**Proposition 2.1** *The iterates generated by SASLU are related by the recurrence scheme*

$$x_{i+1} = x_i + \alpha_i P_i r_i \tag{10}$$

*where*

$$\alpha_i \equiv \frac{r_i^T r_i}{r_i^T P_i^T A P_i r_i} = \frac{\|r_i\|_2^2}{\|P_i r_i\|_2^2} \tag{11}$$

**Proof.** From (7) we derive

$$L_{i+1}^{-1} = \begin{bmatrix} L_i^{-1} & 0 \\ -\lambda_i^{-1} l_i L_i^{-1} & \lambda_i^{-1} \end{bmatrix} \quad and \quad U_{i+1}^{-1} = \begin{bmatrix} U_i^{-1} & -\mu_i^{-1} U_i^{-1} u_i \\ 0 & \mu_i^{-1} \end{bmatrix}$$

thus using (5), (8) and (9) we obtain

$$\begin{aligned} y_{i+1} &= H_{i+1}^{-1} z_{i+1} = U_{i+1}^{-1} L_{i+1}^{-1} z_{i+1} \\ &= \begin{bmatrix} H_i^{-1} + (\lambda_i\mu_i)^{-1} H_i^{-1} V_i^T A v_{i+1} v_{i+1}^T A V_i H_i^{-1} & -(\lambda_i\mu_i)^{-1} H_i^{-1} V_i^T A v_{i+1} \\ -(\lambda_i\mu_i)^{-1} v_{i+1}^T A V_i H_i^{-1} & (\lambda_i\mu_i)^{-1} \end{bmatrix} \begin{bmatrix} z_i \\ v_{i+1}^T f \end{bmatrix} \\ &= \begin{bmatrix} y_i \\ 0 \end{bmatrix} + (\lambda_i\mu_i)^{-1} \begin{bmatrix} H_i^{-1} V_i^T A v_{i+1} v_{i+1}^T A V_i y_i - H_i^{-1} V_i^T A v_{i+1} v_{i+1}^T f \\ -v_{i+1}^T A V_i y_i + v_{i+1}^T f \end{bmatrix} \\ &= \begin{bmatrix} y_i \\ 0 \end{bmatrix} + (\lambda_i\mu_i)^{-1} v_{i+1}^T r_i \begin{bmatrix} -H_i^{-1} V_i^T A v_{i+1} \\ 1 \end{bmatrix} \end{aligned}$$

therefore

$$x_{i+1} = V_{i+1}y_{i+1} = [V_i \;\; v_{i+1}]y_{i+1} = x_i + (\lambda_i\mu_i)^{-1}v_{i+1}^T r_i[-V_i H_i^{-1}V_i^T A v_{i+1} + v_{i+1}]$$
$$= x_i + \frac{v_{i+1}^T r_i}{v_{i+1}^T P_i^T A P_i v_{i+1}} P_i v_{i+1}.$$

Assuming that $M = I$ we have $v_{i+1} = \frac{r_i}{\|r_i\|_2}$ and the announced result follows. □

**Remark 2.1** *The use of the preconditioner $M$ in SASLU leads to the following relation*

$$x_{i+1} = x_i + \beta_i P_i M^{-1} r_i$$

*where* $\beta_i \equiv \frac{r_i^T M^{-1} r_i}{r_i^T M^{-T} P_i^T A P_i M^{-1} r_i}$.

**Lemma 2.2** *If we denote by $P_{i|V_i^\perp}$ the restriction of $P_i$ to $V_i^\perp$ then we have*

$$1 \le \sigma \le \sqrt{1 + \|H_i^{-1}V_i^T A\|_2^2} \le \sqrt{1 + \left(\frac{\sigma_{max}(A)}{\sigma_{min}(H_i)}\right)^2} \tag{12}$$

*for any singular value $\sigma$ of $P_{i|V_i^\perp}$*

**Proof.** Consider the eigenvalue problem $P_{i|V_i^\perp}^T P_{i|V_i^\perp} w = \sigma^2 w$ with $w \in V_i^\perp$. Taking into account that $V_i^T w = 0$ we have

$$P_{i|V_i^\perp}^T P_{i|V_i^\perp} w = w - V_i H_i^{-1}V_i^T A w + A^T V_i H_i^{-T} H_i^{-1} V_i^T A w = \sigma^2 w$$

hence

$$A^T V_i H_i^{-T} H_i^{-1} V_i^T A w - V_i H_i^{-1} V_i^T A w = (\sigma^2 - 1)w$$

which gives

$$\sigma^2 - 1 = \frac{w^T A^T V_i (H_i H_i^T)^{-1} V_i^T A w}{w^T w} = \frac{\|H_i^{-1}V_i^T A w\|_2^2}{\|w\|_2^2}$$

therefore

$$0 \le \sigma^2 - 1 \le \|H_i^{-1}V_i^T A\|_2^2$$

and finally

$$1 \le \sigma \le \sqrt{1 + \|H_i^{-1}V_i^T A\|_2^2} \le \sqrt{1 + \|H_i^{-1}\|_2^2 \|A\|_2^2}.$$

□

**Proposition 2.2** *Let $\bar{x}$ be the true solution and $e_i = \bar{x} - x_i$ be the error at iteration $i$, if we assume that the restriction of $AP_iA$ on $A^{-1}[V_i^{\perp}]$ is positive definite, then there exists $0 < \omega_i < 1$ such that*

$$\|e_{i+1}\|_S^2 \leq (1-\omega_i)\|e_i\|_S^2 \tag{13}$$

*where $\|x\|_S$ denotes the $S$-norm of $x$ which is define as*

$$\|x\|_S = \sqrt{x^T S x}.$$

**Proof.** We first note that

$$\begin{aligned} Ae_i &= r_i \quad (r_i \in V_i^{\perp}) \\ e_{i+1} &= e_i - \alpha_i P_i r_i. \end{aligned}$$

Using (11) and lemma 2.1 we have

$$\begin{aligned} e_{i+1}^T S e_{i+1} &= e_i^T S e_i - 2\alpha_i e_i^T S P_i r_i + \alpha_i^2 r_i^T P_i^T S P_i r_i \\ &= e_i^T S e_i - \alpha_i e_i^T A P_i r_i \\ &= e_i^T S e_i \left(1 - \alpha_i \frac{e_i^T A P_i r_i}{r_i^T A^{-1} r_i}\right). \end{aligned}$$

with

$$\alpha_i = \frac{r_i^T r_i}{(P_i r_i)^T A (P_i r_i)} > 0$$

and

$$e_i^T A P_i r_i = e_i^T A P_i A e_i > 0$$

since $AP_iA$ is assumed to be positive definite on $A^{-1}[V_i^{\perp}]$ and $e_i \in A^{-1}[V_i^{\perp}]$. Therefore

$$\begin{aligned} \|e_{i+1}\|_S^2 &= \|e_i\|_S^2 \left(1 - \frac{r_i^T r_i}{(P_i r_i)^T A (P_i r_i)} \frac{r_i^T A^{-T} A P_i r_i}{r^T A^{-1} r_i}\right) \\ &\leq \|e_i\|_S^2 \left(1 - \frac{\frac{r_i^T A^{-T} A P_i r_i}{r_i^T r_i}}{\|A\|_2 \|A^{-1}\|_2 \sigma_{max}^2(P_{i|V_i^{\perp}})}\right). \end{aligned}$$

If we let $\rho_i = \min_{u \neq 0} \frac{u^T A^{-T} A P_i u}{u^T u}$ then $\rho_i > 0$ and using lemma (2.2)

$$\|e_{i+1}\|_S^2 \leq \|e_i\|_S^2 \left(1 - \frac{\rho_i}{\kappa_2(A)\left(1 + \frac{\|A\|_2^2}{\sigma_{min}^2(H_i)}\right)}\right)$$

□

**Remark 2.2** *We notice that if $A$ is symmetric, then for every $u \in A^{-1}[V_i^{\perp}]$*

$$u^T A P_i A u = u^T A^2 u - (Au)^T V_i H_i^{-1} V_i^T A^2 u = u^T A^2 u > 0$$

*which means that $AP_iA$ is positive definite.*

The relation exhibited in proposition 2.1 above is a way to compute the undergoing solution of SASLU. At first glance, it seems to require a large amount of calculations although we have in mind the re-use of data. This gives emergence to the question of deciding if we must use that recurrence formula or the more regular and straightforward forward-elimination/backward-substitution induced by $LU$ decomposition which in our case is already available.

**Proposition 2.3** *In order to compute the next iterate $x_{i+1}$, if $i \leq \sqrt{4n+3}-2$, it is less costly to proceed directly by forward-elimination/backward-substitution than to use the updating formula (10).*

**Proof.**

- On one hand suppose that we use the formula $x_{i+1} = V_{i+1} U_{i+1}^{-1} L_{i+1}^{-1} z_{i+1}$ to compute the next iterate. Since the complexity of a triangular system of order $p$ is $p^2$, we infer that the cost of computing $x_{i+1}$ is $2n(i+1) + 2(i+1)^2$.

- On the other hand suppose now that we use the update formula (10) :

$$\begin{array}{lcll} \alpha_i & = & \frac{r_i^T r_i}{r_i^T P_i^T A P_i r_i} & \rightarrow 2n \\ & & & \rightarrow \textit{available see (9)} \quad 2n+1 \\ P_i r_i & = & r_i - \|r_i\|_2 V_i U_i^{-1} u_i & \textit{see (8)} \qquad i^2 + 2ni + 2n \end{array}$$

  therefore the cost for computing $x_{i+1} = x_i + \alpha_i P_i r_i$ is now $i^2 + 2ni + 6n + 1$.

Consider the function $\varphi(i) = (2n(i+1) + 2(i+1)^2) - (6n + 1 + 2ni + i^2) = i^2 + 4i + 1 - 4n$ which is a second degree polynomial in $i$. We can check that in the range $\left[1, \sqrt{4n+3} - 2\right]$ the so defined $\varphi(i)$ is negative. □

If we take $\sqrt{4n+3} - 2 \sim 2\sqrt{n}$, the above proposition shows that depending on the size of the basis, we must switch from $LU$ to the update formula (10) at a certain iteration. In practice, the maximum size $m$ is less than $2\sqrt{n}$ and such a switch is not necessary. The value of $HSOLVE(i)$ is then $2i^2$.

We now discuss another algorithm for solving the linear system (1). Unless otherwise mentioned, we keep the same notations $m$, $MGS$, $S = \frac{A+A^T}{2}$, $N = \frac{A-A^T}{2}$ as before.

# 3 Algorithm SASMIN

Choose a norm one start-vector $v_1$ and set $V_1 = [v_1]$
**for** $i = 1, 2, \ldots$
  $W_i = AV_i$
  find $y_i \in \mathbf{R}^i$ such that $\|f - W_i y_i\|_2$ *is minimal.*
  compute $x_i = V_i y_i$ and $r_i = f - W_i y_i$
  **if** convergence **stop**
  $\hat{v}_{i+1} = M^{-1} r_i$
  **if** $dim(V_i) < m$
    $V_{i+1} = MGS[V_i, \hat{v}_{i+1}]$
  **else**
    $V_{i+1} = MGS[x_i, \hat{v}_{i+1}]$
    **restart**
  **endif**
**endfor**

The main difference between SASLU and SASMIN is that in SASLU we solve a projected system at each step while in SASMIN we minimize the residual norm in the projected subspace. Both algorithms increase the basis in the same way, however in SASMIN, orthogonality of $V_i$ is not primordial and MGS operates for scaling only. It may be partial.

## 3.1 Convergence

First note that after the orthonormalization step, the assignment $\hat{v}_{i+1} = M^{-1} r_i$ is equivalent to solve the system $M(\hat{v}_{i+1} - V_i y_i) = r_i$ and this can be viewed as a correction of the approximate solution $V_i y_i$ which would be obtained by one step of iterative refinement provided that the spectral radius of $M^{-1}(M - A)$ is less than one. This step is therefore crucial to the success of the method. The following proposition is easy to prove.

**Proposition 3.1** *If we start SASLU or SASMIN with* $v_1 = \frac{M^{-1} f}{\|M^{-1} f\|_2}$, *and if no restart is used, then the basis* $V_i$ *generates the Krylov subspace* $K_i = \{v_1, M^{-1} A v_1, \ldots, (M^{-1} A)^{i-1} v_1\}$.

The following theorem gives sufficient conditions under which SASMIN converges.

**Theorem 3.1** *Assume that the symmetric part* $S_{AM^{-1}}$ *of* $AM^{-1}$ *is positive definite and that we start SASMIN with* $v_1 = \frac{M^{-1} f}{\|M^{-1} f\|_2}$, *if no restart is used, then the residual* $r_i = f - W_i y_i$ *computed at step* $i$ *satisfies* $\|f - W_i y_i\|_2 \leq \left[1 - \frac{\lambda^2_{min}(S_{AM^{-1}})}{\lambda_{max}\left((AM^{-1})^T (AM^{-1})\right)}\right]^{i/2} \|f\|_2$.

**Proof.** At step $i$, the residual $r_i$ satisfies the minimization problem

$$\begin{aligned} \|f - W_i y_i\|_2 &= \min_{v \in V_i} \|f - Av\|_2 \\ &= \min_{p_{i-1} \in \mathcal{P}_{i-1}} \|f - Ap_{i-1}(M^{-1}A)v_1\|_2 \end{aligned}$$

where $\mathcal{P}_{i-1}$ stands for the set of polynomial of degree not exceeding $i-1$.
If, on the other hand we start with $v_1 = \frac{M^{-1}f}{\|M^{-1}f\|_2}$ we have

$$\begin{aligned} \|f - W_i y_i\|_2 &= \min_{\{p_{i-1} \in \mathcal{P}_{i-1}\}} \|f - Ap_{i-1}(M^{-1}A)M^{-1}f\|_2 \\ &= \min_{\{p_{i-1} \in \mathcal{P}_{i-1}\}} \|f - AM^{-1}p_{i-1}(AM^{-1})f\|_2 \\ &= \min_{\{q_i \in \mathcal{P}_i \ ; \ q_i(0)=1\}} \|q_i(AM^{-1})f\|_2 \\ &\leq \left[1 - \frac{\lambda^2_{min}(S_{AM^{-1}})}{\lambda_{max}\left((AM^{-1})^T(AM^{-1})\right)}\right]^{i/2} \|f\|_2. \end{aligned}$$

The last inequality is derived in the same way as in [4]. □

Even when restart is used, the residual norm decrease because the previous best estimate is incorporated in the new initial basis and the minimization still goes on.

### 3.2 Practical Details and Complexity Analysis

As in SASLU, we intensively use updatings. However the drawback here seems more heavier since the resolution of the small linear system is replaced by the linear least square (LLS) problem :

$$\text{find } y_i \in \mathbf{R}^i \text{ such that } \|f - W_i y_i\|_2 = \min_{y \in \mathbf{R}^i} \|f - W_i y\|_2.$$

Nevertheless it is a full rank LLS problem and if we consider the QR decomposition of $W_i$

$$W_i = Q_i R_i, \text{ with } Q_i \in \mathbf{R}^{n \times i} \text{ and } R_i \in \mathbf{R}^{i \times i},$$

the unique solution is

$$\begin{aligned} y_i &= (W_i^T W_i)^{-1} W_i^T f \\ &= R_i^{-1} Q_i^T f. \end{aligned} \tag{14}$$

It is easy to prove that $W_{i+1} = [W_i, \ Av_{i+1}] = Q_{i+1}R_{i+1}$ with

$$Q_{i+1} = [Q_i, \ q_{i+1}], \quad R_{i+1} = \begin{bmatrix} R_i & \pi_{i+1} \\ 0 & \sigma_{i+1} \end{bmatrix},$$

where

$$\begin{aligned} \pi_{i+1} &= Q_i^T A v_{i+1} \\ \sigma_{i+1} &= \|(I - Q_i Q_i^T) A v_{i+1}\|_2 \\ q_{i+1} &= \frac{(I - Q_i Q_i^T) A v_{i+1}}{\sigma_{i+1}} \end{aligned}$$

In a similar way as in proposition 2.1, we can derive the following result.

**Proposition 3.2** *The iterates generated by SASMIN are related by the recurrence scheme*

$$x_{i+1} = x_i + \alpha_i P_i M^{-1} r_i$$

*where*

$$\begin{aligned} P_i &\equiv I - V_i R_i^{-1} Q_i^T A \\ \alpha_i &\equiv \frac{r_i^T A M^{-1} r_i}{r_i^T M^{-T} P_i^T A^T A P_i M^{-1} r_i}. \end{aligned}$$

*$P_i$ is the projection onto $A^{-1}[Q_i^{\perp}]$ along $V_i$ and satisfies :*

$$\begin{aligned} P_i^2 &= P_i \\ P_i V_i &= 0 \\ P_i^T A^T A P_i &= A^T A P_i \\ (I - Q_i Q_i^T) A &= A P_i. \end{aligned}$$

Our implementation of SASMIN does not make use of the above recurrence relation. It is based on Householder transformations [5]. Let $H_1, ..., H_{i-1}$ be the available Householder elementary reflectors at completion of iteration $i-1$. Iteration $i$ needs the application of previous reflectors on the new appended column of $W_i = [W_{i-1},\ w_i]$ followed by the computation of the new reflector. If we let $\hat{w}^{(i)} = H_{i-1} \cdots H_1 w_i$ be the resulting column vector after premultiplying previous Householder transformations on the new appended column, then the new reflector is constructed in the following manner :

$$\begin{aligned} \sigma_i &= -\text{sign}(\hat{w}_i^{(i)}) \|\hat{w}^{(i)}(i:n)\|_2 \\ \tau_i &= (\sigma_i^2 - \sigma_i \hat{w}_i^{(i)})^{-1} \\ H_i &= I - \tau_i u_i u_i^T \\ u_i &= \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \hat{w}_i^{(i)} - \sigma_i \\ \hat{w}_{i+1}^{(i)} \\ \vdots \\ \hat{w}_n^{(i)} \end{bmatrix}. \end{aligned}$$

The premultiplication of the so defined reflector on $\hat{w}^{(i)}$ does not affect its first $(i-1)$ st components but zeroes the last $(n-i)$ th ones. Moreover the $(i-1)$ st components of $\hat{w}^{(i)}$ constitutes the $i$-th column of the triangular factor $R_i$ of the QR decomposition of $W_i$ with $\sigma_i$ being on the diagonal. On the other hand the orthogonal factor $Q_i$ is the first $i$-th columns of the product of the elementary reflectors $\hat{Q}_i = H_1 \cdots H_i$., i.e., $\hat{Q}_i = [Q_i, \_]$. However, neither the application of $H_i$ on $\hat{w}^{(i)}$, nor the computation of $Q_i$ is done. We just compute $\hat{w}^{(i)}$, $\sigma_i$ and $\tau_i$. The upper triangular part of $W_i$ contains $R_i$ while the lower part contains non null components of Householder vectors $u_i$'s which can be used to generate $Q_i$. Observe that the first non null component of $u_i$ may be saved, thus overwriting $\sigma_i$ the $i$-th diagonal element of $R_i$. Indeed, $R_i$ is needed to compute the approximate solution at convergence or at restart only and the $\sigma_i$'s can be recovered at the right time since $\hat{w}_i^{(i)} - \sigma_i = -(\sigma_i \tau_i)^{-1}$. Hence there can be only one auxiliary vector for keeping the $\tau_i$'s.

Another point of interest is the computation of the residual which is done without knowing explicitly the actual approximate solution. Recall that $Q_i$ is not available and that we can only apply $\hat{Q}_i = H_1 \cdots H_i$ or $\hat{Q}_i^T = H_i \cdots H_1$ on a vector, if we let $\hat{z}^{(i)} = \hat{Q}_i^T f$, then we make use of the equalities

$$Q_i = \hat{Q}_i \begin{bmatrix} I_i \\ 0 \end{bmatrix} \quad \text{and} \quad Q_i^T = [I_i, 0]\hat{Q}_i^T$$

to obtain

$$r_i = f - Q_i Q_i^T f = f - \hat{Q}_i \begin{bmatrix} \hat{z}_1^{(i)} \\ \vdots \\ \hat{z}_i^{(i)} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

From this analysis we can see that the storage is limited to the basis $V(n,m)$, the work array $W(n,m)$, and two auxiliary vectors $\tau(m)$ and $\hat{z}(n)$. Therefore the overall extra space here is $2mn + m + n$. We can also derive the following complexity costs per iteration.

| | |
|---|---|
| $W_i = [W_{i-1}, Av_i]$ | : 1 $MATVEC$ |
| $\hat{w}^{(i)} = H_{i-1} \cdots H_1 w_i$ | : $(4n+7)i - 2i^2 - 4n - 5$ |
| $H_i$ i.e. $\sigma_i$, $\tau_i$, $u_i$ | : $2n + 6 - 2i$ |
| $\hat{z}^{(i)} = \hat{Q}_i^T f = H_i \hat{z}^{(i-1)}$ | : $4n + 5 - 4i$ |
| $y_i = R_i^{-1} \hat{z}^{(i)}(1:i)$ | : $i^2$ |
| $x_i = V_i y_i$ | : $2ni$ |
| $r_i = f - H_1 \cdots H_i [\hat{z}^{(i)}(1:i), 0, ..., 0]^T$ | : $(4n+3)i - 2i^2 + n$ |

$$\hat{v}_{i+1} = M^{-1} r_i \qquad : 1\ MSOLVE$$
$$V_{i+1} = MGS[V_i, \hat{v}_{i+1}] \qquad : 4ni + 3n$$

Therefore we infer that the overall complexity per iteration in SASMIN is about

$$2n + 6 + (12n + 4)i - 3i^2 + 1\ MATVEC + 1\ MSOLVE$$

Let denote by $\text{cplx}_i$(SASLU) the complexity of SASLU at iteration $i$ whithout counting the cost for computing $x_i$ and by $\text{cplx}_i$(SASMIN) the one of SASMIN whithout counting the cost for computing $y_i$ and $x_i$. We have $\text{cplx}_i$(SASLU)$-\text{cplx}_i$(SASMIN) $= 8i^2 + (2n - 12)i + 2n - 3$ which is greater than zero. Hence we notice that although solving a least square problem in its own right is more expensive than solving a linear system by LU-factorisation, the implementation of SASMIN is less costly per iteration than SASLU. This is due to the fact that in SASLU the factors L and U are explicitly constructed.

# 4 Numerical Experiments

This section presents timing results and numerical behaviour of SASLU and SASMIN compared to GMRES. The test matrices come from the application of the seven point centered difference operator on the elliptic partial differential equation of the form

$$au_{xx} + bu_{yy} + cu_{zz} + du_x + eu_y + fu_z + gu = F$$

on the unit cube. The true solution is analytically known and is used to generate the right hand side of the system. $a, b, c, e, f, g$ are functions of $x, y, z$.

| | Problem | Solution |
|---|---|---|
| (a) | $u_{xx} + u_{yy} + u_{zz} +$ $10^3 e^{xyz}(u_x + u_y - u_z) = F$ | $u = x + y + z$ |
| (b) | $(e^{xyz}u_x)_x + (e^{-xyz}u_y)_y + (e^{-xyz}u_z)_z -$ $250(x + y + z)u_x - 250[(x + y + z)u]_x$ $-u/(1 + x + y + z) = F$ | $u = e^{xyz} sin(\pi x) sin(\pi y) sin(\pi z)$ |
| (c) | $u_{xx} + u_{yy} + u_{zz} -$ $10^3(1 + x^2)u_x + 10^2(u_y + u_z) = F$ | $u = e^{xyz} sin(\pi x) sin(\pi y) sin(\pi z)$ |
| (d) | $u_{xx} + u_{yy} + u_{zz} -$ $10^3[(1 - 2x)u_x + (1 - 2y)u_y + (1 - 2z)u_z] = F$ | $u = e^{xyz} sin(\pi x) sin(\pi y) sin(\pi z)$ |

Table 1. The test problems.

The number of interior grids in each direction have been set to 12 and the obtained matrices are all of order $n = 12^3 = 1728$ with $nz = 11232$ non zero elements. Some of their characteristics are outlined in the tables describing the results; for more details on these problems see [1] and

references therein.

All the methods are carried out on one processor of a CRAY-2 in dedicated mode. The GMRES code is the one in the official release version 2 of the Sparse Linear Algebra Package (SLAP) available via *netlib*. Two common preconditioners for sparse unsymmetric matrices are used : ILU0 and ILUTH. ILUTH(*tol*) means that the drop tolerance parameter is $tol * \|A\|_\infty$ and therefore elements appearing in LU decomposition whose magnitude are less than this drop tolerance are ignored. We denote by *nzilu* the fill-in created during the incomplete decomposition.

### 4.1 GMRES : the shortcoming of the convergence test

When dealing with left preconditioning in GMRES, we solve the system

$$\hat{A}\hat{x} = \hat{f} \tag{15}$$

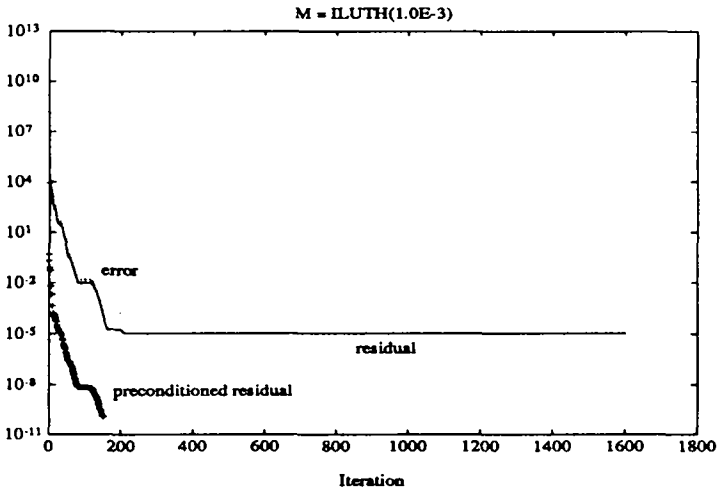
where $\hat{A} = M^{-1}A$ and $\hat{f} = M^{-1}f$.

By construction GMRES will compute without extra work the residual norm of (15) [10, 2] which is somewhat the preconditioned residual $\frac{\|M^{-1}r_i\|_2}{\|M^{-1}f\|_2} = \frac{\|M^{-1}(f-Ax_i)\|_2}{\|M^{-1}f\|_2}$ of the original system. Therefore a stopping test based upon this cheaply computed residual can be misleading. Indeed if $M$ is nearly singular the preconditioned residual may be small while the relative error $\frac{\|\bar{x}-x_i\|_2}{\|\bar{x}\|_2}$ is still large. Moreover, note that since one works with the modified system (15) the approximate iterate $x_i$ can also be worstly computed. Let us consider the two following convergence function checkers :

**logical function** cvge() ! convergence checker
  cvge = $\left(\frac{\|M^{-1}r_i\|_2}{\|M^{-1}f\|_2} \leq \epsilon\right)$
**end**
**logical function** mcvge() ! modified convergence checker
  mcvge = $\left(\frac{\|M^{-1}r_i\|_2}{\|M^{-1}f\|_2} \leq \epsilon\right)$
  if (mcvge) mcvge = $\left(\frac{\|f-Ax_i\|_2}{\|f\|_2} \leq \epsilon\right)$
**end**

The execution of **cvge** which corresponds to the execution of the first statement of **mcvge** does not require extra cost but the execution of the second statement of **mcvge** requires to obtain $x_i$ explicitly and to do one more matrix-vector multiplication. However **mcvge** is more reliable than **cvge** as it is illustrated in the following example.

| Problem (a)<br>*Re*($\lambda(A)$) *all positive, A not definite,* $\|A\|_\infty = 428.95$ | | | | | | |
|---|---|---|---|---|---|---|
| SOLVER | M | NMULT | $\frac{\|M^{-1}r_i\|_2}{\|M^{-1}f\|_2}$ | $\frac{\|\bar{x}-x_i\|_2}{\|\bar{x}\|_2}$ | starts | time |
| GMRES (cvge) | I | 927 | 1.0E-10 | 0.4E-09 | 12 | 6.97 |
| | ILU0 | 1601 | 0.1E-07 | 0.1E+09 | 21 | 31.12 |
| | ILUTH ($10^{-3}$) | 151 | 1.0E-10 | 0.3E-03 | 2 | 4.30 |
| | | | $\frac{\|r_i\|_2}{\|f\|_2}$ | | | |
| GMRES (mcvge) | I | 927 | 1.0E-10 | 0.4E-09 | 12 | - |
| | ILU0 | 1601 | 0.9E+08 | 0.1E+09 | 21 | - |
| | ILUTH ($10^{-3}$) | 3052 | 0.2E-03 | 0.1E-03 | 21 | - |
| SASLU | I | 1262 | 0.9E-10 | 0.1E-09 | 16 | 21.50 |
| | ILU0 | 1600 | 0.6E-03 | 0.1E-02 | 20 | 47.92 |
| | ILUTH ($10^{-3}$) | 396 | 0.9E-10 | 0.1E-09 | 5 | 15.21 |
| SASMIN | I | 1555 | 1.0E-10 | 0.3E-09 | 20 | 16.86 |
| | ILU0 | 1600 | 0.2E-01 | 0.8E-01 | 20 | 35.88 |
| | ILUTH ($10^{-3}$) | 712 | 0.9E-10 | 0.1E-09 | 9 | 21.95 |




Problem (a)
Misleading of preconditioned GMRES

GMRES succeeds when no preconditioner is used i.e. $M = I$, fails when $M = ILU0$ and converges very poorly when $M = ILUTH(10^{-3})$. However the use of the stopping checker **cvge** does not lead to this conclusion. The explanation is that both preconditioners are nearly singular. If we denote by $e = (1, ..., 1)^T$ then we have

$$\frac{\|M^{-1}Ae\|_2}{\|e\|_2} = \begin{cases} 1.81E+11 & \text{for } M = ILU0 \\ 8.75E+05 & \text{for } M = ILUTH(10^{-3}). \end{cases}$$

Therefore to prevent this shortcoming we base the convergence test on **mcvge** rather than on **cvge** and the speed of convegence is not really affected since this test is done only periodically. This drawback does not occur in SASLU and SASMIN since the original residual must be computed. We refer to the ratio $\gamma = \frac{\|M^{-1}Ae\|_2}{\|e\|_2}$ as an indicator of the quality of the preconditioner $M$. We will say that the preconditioning matrix $M$ is of good quality if this ratio is not far from 1.

## 4.2 Comparisons between SASLU, SASMIN and GMRES

For all tests, the convergence criterion is

$$\frac{\|f - Ax_i\|_2}{\|f\|_2} \leq \epsilon.$$

The prespecified $\epsilon$ is $10^{-10}$; $itmax$, the maximum number of outer iterations i.e. the number of restarts is set to 20 and $m$, the maximum size for the basis is set to 80. Entries in tables contain for each algorithm :

- the **type of preconditioner** denoted by "M"
- the **number of matrix-vector multiplications** denoted by "NMULT"
- the **relative residual** $\frac{\|f - Ax_i\|_2}{\|f\|_2}$ denoted by "residual"
- the **relative error** $\frac{\|\bar{x} - x_i\|_2}{\|\bar{x}\|_2}$ denoted by "error"
- the **number of restarts** denoted by "starts"
- the **execution time** in seconds denoted by "time"
- the **exit completion code** denoted by "code" which is "F" for failure or "S" for success :
  - F/LU means that failure was due to ILU decomposition which has failed or was nearly singular
  - the symbol $\oslash$ means that if the stopping test was based on the convergence function checker **cvge**, then the mentionned misleading of GMRES would have occurred (see problem (**d**)).

Globally we observe that SASLU and SASMIN converge or fail both together and when they fail GMRES also do. But the converse is not always true. We particularly observe that :

- positive definite property of $A$ ensure convergence and if it is the case, GMRES reaches the desire accuracy always the first when no preconditioner is used. But when a preconditioner is used SASMIN performs slightly better. SASLU is a little slow since it oscillates as one can see on the curves

- if $A$ is not definite but $Re(\lambda(A))$ are all positive (or all negative), then convergence can also be achieved. It can be slow when no preconditioning is used and is not always guaranteed
- if $A$ is not definite and the spectrum of $A$ lies on both sides of the complex plane, then all the methods need a good preconditioner.

| (n = 1728 and nz = 11232 for all problems) | | | |
|---|---|---|---|
| Problem | M | nzilu | $\gamma = \frac{\|M^{-1}Ae\|_2}{\|e\|_2}$ |
| (a) | ILU0 | | 1.81E+11 |
| | ILUTH($10^{-3}$) | 209620 | 8.75D+05 |
| (b) | ILU0 | | 1.20 |
| | ILUTH($10^{-3}$) | 25752 | 1.04 |
| (c) | ILU0 | | 28.50 |
| | ILUTH($10^{-3}$) | 43280 | 22.96 |
| (d) | ILU0 | | 2.58D+05 |
| | ILUTH($10^{-3}$) | 91393 | 1.642 |

Table 2. Fill-in and quality of preconditioners.

| Problem (b) $Re(\lambda(A))$ *all positive*, *A definite*, $\|A\|_\infty = 111.96$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| SOLVER | M | NMULT | residual | error | starts | time | code |
| SASLU | I | 384 | 0.86E-10 | 0.52E-10 | 5 | 6.47 | S |
| | ILU0 | 13 | 0.69E-10 | 0.67E-10 | 1 | 0.26 | S |
| | ILUTH ($10^{-3}$) | 13 | 0.29E-10 | 0.24E-10 | 1 | 0.26 | S |
| SASMIN | I | 593 | 0.97E-10 | 0.15E-09 | 8 | 6.38 | S |
| | ILU0 | 13 | 0.51E-10 | 0.51E-10 | 1 | 0.24 | S |
| | ILUTH ($10^{-3}$) | 12 | 0.78E-10 | 0.13E-09 | 1 | 0.23 | S |
| GMRES | I | 367 | 0.98E-10 | 0.18E-09 | 5 | 2.75 | S |
| | ILU0 | 14 | 0.57E-10 | 0.45E-10 | 1 | 0.25 | S |
| | ILUTH ($10^{-3}$) | 15 | 0.13E-10 | 0.12E-10 | 1 | 0.26 | S |

Problem **(b)** is such that the coefficient matrix is positive definite. The three methods converge. Moreover the preconditioners are of good quality and this explain the significant gain when they are used. With $M = I$ SASLU has a small number of matrix-vector multiplication than SASMIN but it has much more expensive complexity so its execution time is the highest.

| Problem (c) $Re(\lambda(A))$ *all positive, A not definite,* $\|A\|_\infty = 153.38$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| SOLVER | M | NMULT | residual | error | starts | time | code |
| SASLU | I | 631 | 0.85E-10 | 0.71E-10 | 8 | 10.76 | S |
| | ILU0 | 49 | 0.68E-10 | 0.54E-10 | 1 | 1.27 | S |
| | ILUTH ($10^{-3}$) | 45 | 0.63E-10 | 0.13E-09 | 1 | 1.18 | S |
| SASMIN | I | 896 | 0.94E-10 | 0.31E-09 | 12 | 9.70 | S |
| | ILU0 | 47 | 0.87E-10 | 0.15E-09 | 1 | 0.97 | S |
| | ILUTH ($10^{-3}$) | 44 | 0.96E-10 | 0.21E-09 | 1 | 0.95 | S |
| GMRES | I | 608 | 0.98E-10 | 0.38E-09 | 8 | 4.58 | S |
| | ILU0 | 54 | 0.95E-10 | 0.13E-09 | 1 | 0.94 | S |
| | ILUTH ($10^{-3}$) | 46 | 0.60E-10 | 0.12E-09 | 1 | 0.90 | S |

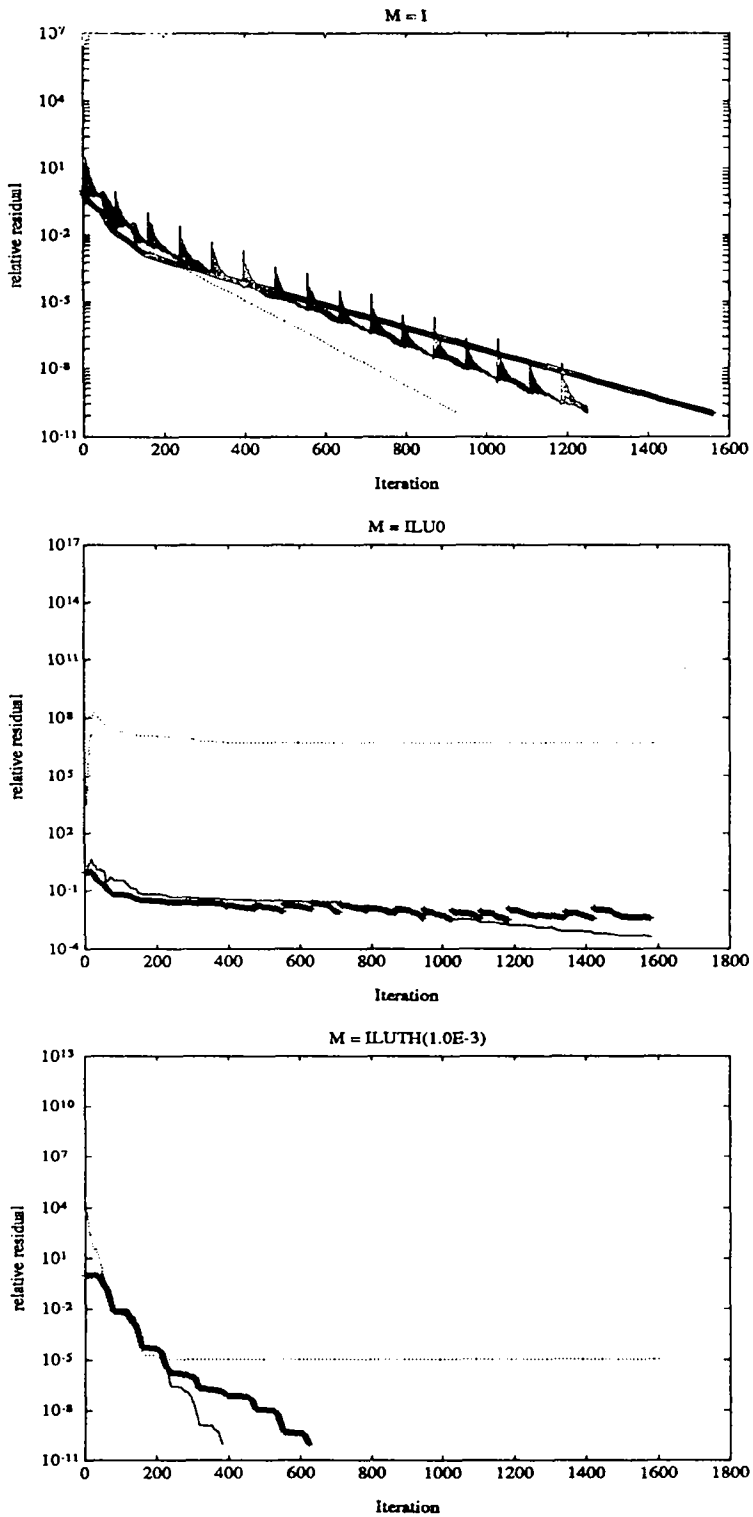
| Problem (d) $Re(\lambda(A))$ *all positive, A not definite,* $\|A\|_\infty = 165.76$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| SOLVER | M | NMULT | residual | error | starts | time | code |
| SASLU | I | 435 | 0.91E-10 | 0.14E-09 | 6 | 7.24 | S |
| | ILU0 | 239 | 0.74E-10 | 0.91E-10 | 3 | 7.15 | S |
| | ILUTH ($10^{-3}$) | 27 | 0.40E-10 | 0.49E-10 | 1 | 0.69 | S |
| SASMIN | I | 410 | 0.97E-10 | 0.33E-09 | 6 | 4.43 | S |
| | ILU0 | 310 | 0.97E-10 | 0.25E-09 | 4 | 6.92 | S |
| | ILUTH ($10^{-3}$) | 26 | 0.99E-10 | 0.14E-09 | 1 | 0.58 | S |
| GMRES | I | 433 | 0.99E-10 | 0.39E-09 | 6 | 3.25 | S |
| | ILU0 | 3086 | 0.72E-04 | 0.48E-04 | 21 | 41.45 | ⊘ |
| | ILUTH ($10^{-3}$) | 29 | 0.38E-10 | 0.49E-10 | 1 | 0.60 | S |

For problems **(c)** and **(d)** the coefficient matrices are not definite but their eigenvalues lie on the same side of the complex plane. SASLU and SASMIN converge even in problem **(d)** using the preconditioner ILU0 which is not of good quality and is nearly singular. GMRES would have end up with the convergence flag set on if the stopping checker was the function **cvge** although it is not the case.

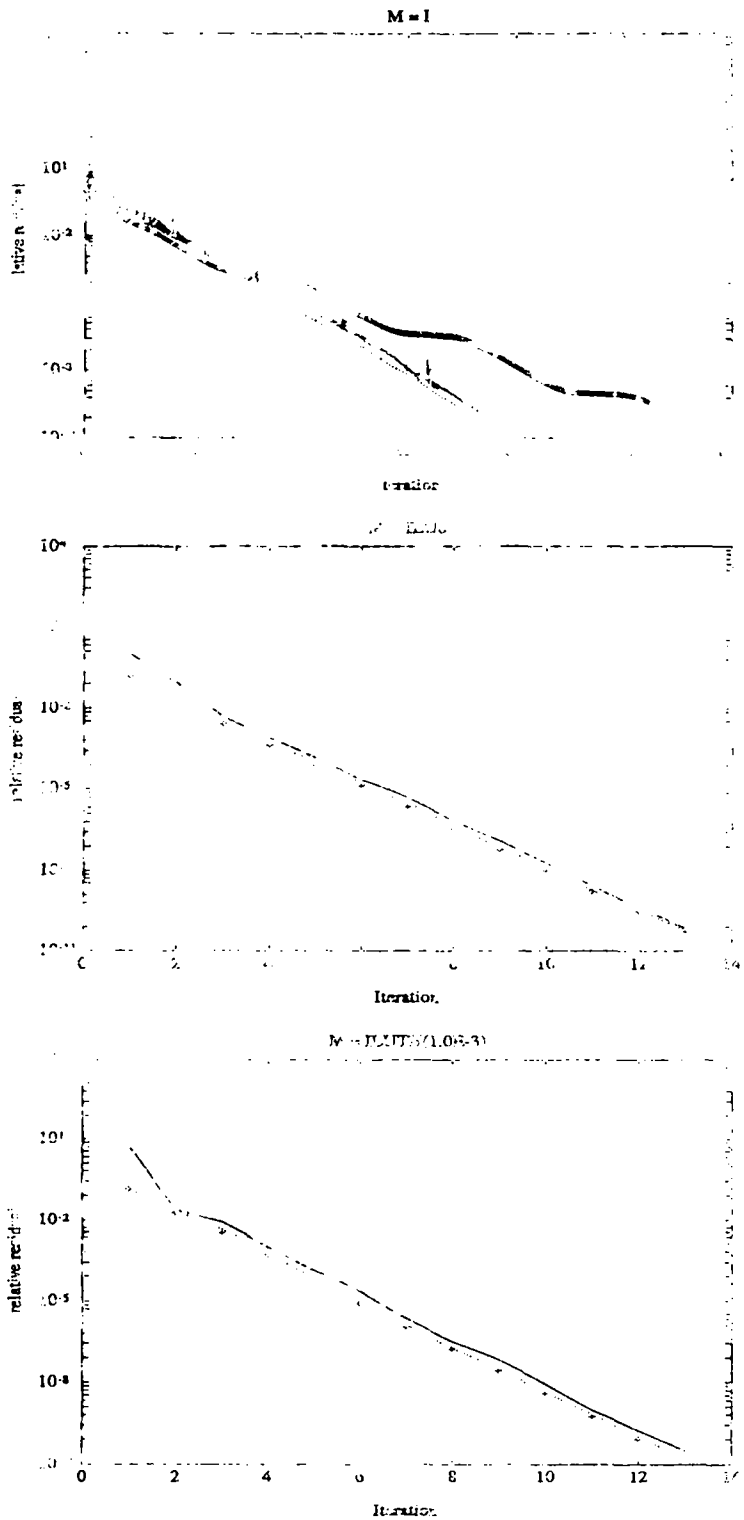
# 5 Conclusion

In this paper we have presented a new projection approach based on a gradually improved basis. We have proposed two variants of this approach, the first one is linear system oriented while the second one is least square oriented. Both variants increase the basis in the same idea : the new basis vector is the improved solution over the current subspace. Convergence and complexity have been studied in the case where the symmetric part of the matrix is positive definite. The two strategies are of practical value and implementation details can still be improved. In case

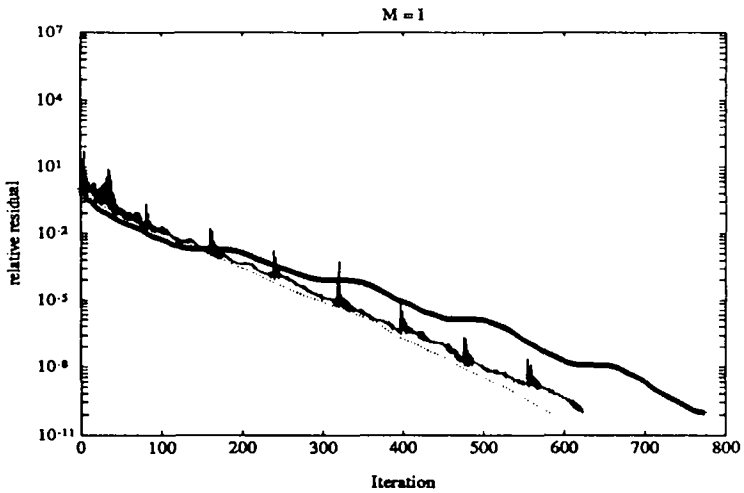
of multiple right hand sides for example, a block version can be designed and we have tested numerically that it has a significant acceleration on the global convergence with respect to the size of the blocks. Theoritically when the coefficient matrix is positive definite break-down cannot occur. However special care should be taken to handle it. The processes fail when the new basis vector can be generated by the previous ones i.e $v_{i+1} \in span(V_i)$. But since calculations do not exploit any mathematical property related to this new vector, another one can be incorporated in place of it in order to avoid to restart. The methods succeed in solving a variety of PDE problems, including some problems where the matrix is not positive definite. The original residual is always available and in case of nearly singular preconditioners, contrary to $GMRES$, the methods do not render unrealistic solutions.

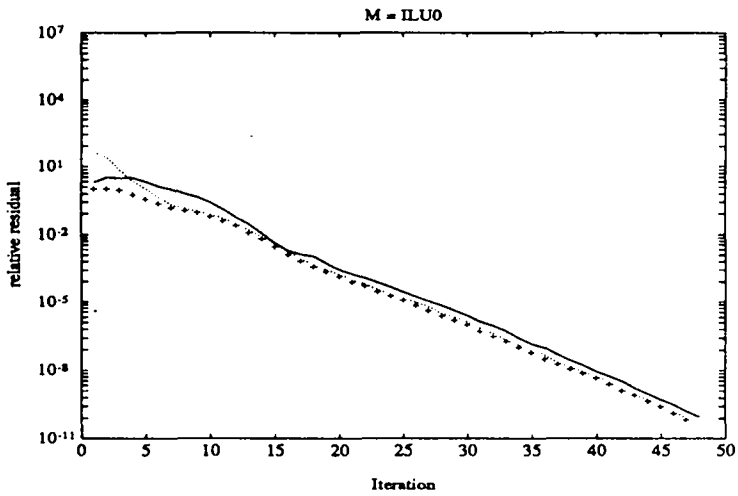


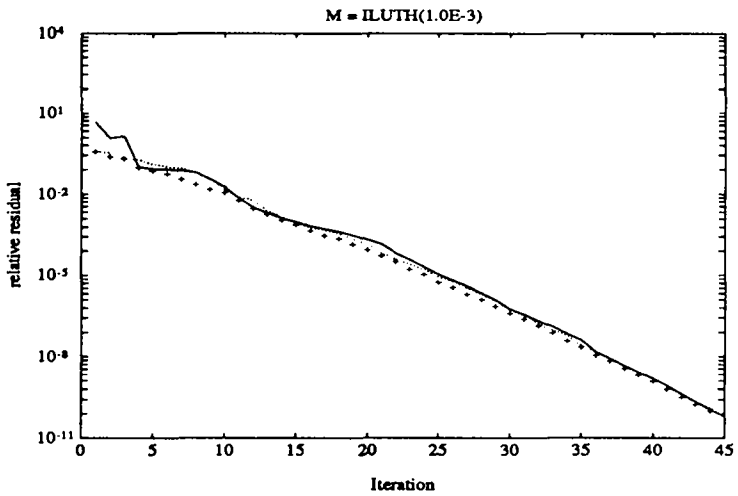

Problem (a)
Relative residual versus Iteration number
— : SASLU, + : SASMIN, ··· : GMRES



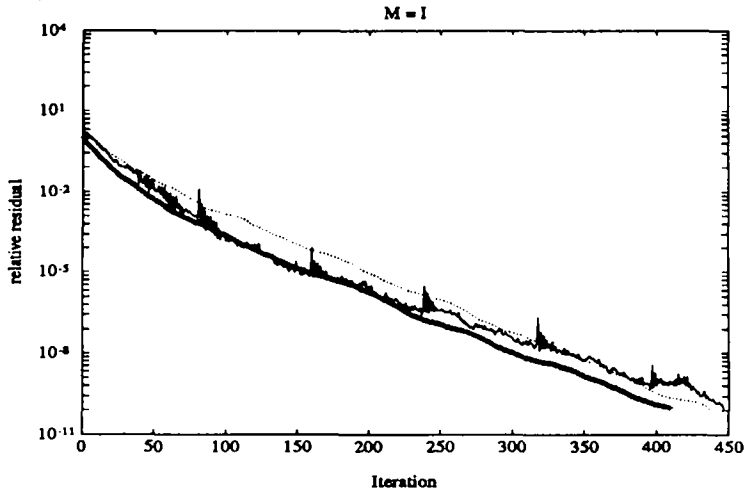

Problem (b)
Relative residual versus Iteration number
— : SASEO, + : SASMIN, ··· : GMRES

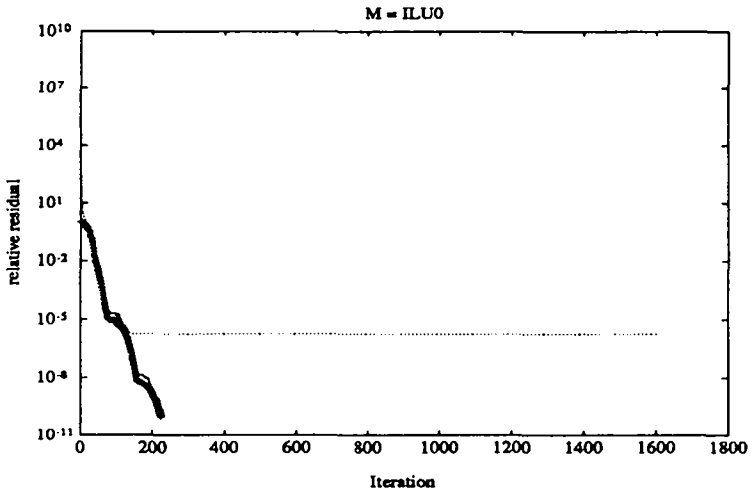


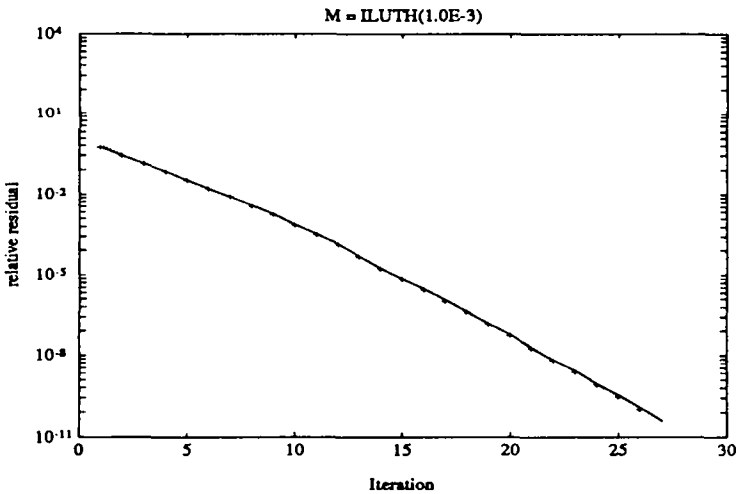




Problem (c)
Relative residual versus Iteration number
— : SASLU, + : SASMIN, ··· : GMRES




Problem (d)
Relative residual versus Iteration number
— : SASLU, + : SASMIN, ··· : GMRES

[1] C. B. Brakley, *Red Projection Methods for Linear Systems*, PhD thesis, CSRD-University of Illinois, May 1980.

[2] P. N. Brown, *A theoretical comparison of the Arnoldi and GMRES algorithms*, SIAM J. Sci. and Stat. Comp., 12 (1991), pp. 58–78.

[3] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Method for Sparse Matrices*, Clarendon Press, Oxford, 1989.

[4] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Variational iterative methods for nonsymmetric systems of linear equations.*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, second ed., 1989.

[6] M. Jankowski and H. Wozniakowski, *Iterative refinement implies numerical stability*, BIT., 17 (1977), pp. 303–311.

[7] J. M. Ortega, *Introduction to Parallel and Vector Solutions of Linear Systems*, Plenum Press, New York, 1989.

[8] B. N. Parlett, *The symmetric eigenvalue problem*, Prentice-Hall, Englewood Cliffs, N.J, 1980.

[9] Y. Saad, *Analysis of some krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal, 29 (1992), pp. 208–227.

[10] Y. Saad and M. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. and Stat. Comp., (1986), pp. 856–869.

FAUTES DE

WITH

ORIENTED

ISSN 0249-6399