Vectorizing finite element methods

Jocelyne Erhel, A. Lichnewsky, F. Thomasset

INRIA
CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N° 383

VECTORIZING FINITE ELEMENT METHODS

Jocelyne ERHEL
Alain LICHNEWSKY
François THOMASSET

Mars 1985

VECTORIZING FINITE ELEMENTS METHODS

Jocelyne ERHEL
INRIA, Rocquencourt, France

Alain LICHNEWSKY
Université Paris Sud, Orsay, France
and INRIA, Rocquencourt, France

François THOMASSET
INRIA, Rocquencourt, France

RESUME

Dans ce rapport, nous décrivons les techniques développées pour mettre en oeuvre des algorithmes d'algèbre linéaire sur des calculateurs vectoriels tels que CRAY-1S, Fujitsu VP200, Hitachi S810, ... Les algorithmes étudiés traitent des matrices creuses issues de méthodes d'éléments finis. En particulier, nous nous intéressons ici à l'algorithme du gradient conjugué préconditionné. Nous introduisons une renumérotation des inconnues pour supprimer les dépendances de données et augmenter le parallélisme. Le cas particulier des structures "régulières" de matrices correspond aux méthodes de différences finies. Dans le cas général, il faut exhiber des vecteurs assez longs pour compenser le coût des indirections.

Afin de systématiser la recherche de rangements parallèles de nos matrices et compléter les techniques heuristiques que nous venons de mentionner, nous introduisons une formulation en termes d'optimisation. Pour la résolution de ces problèmes nous mettons en oeuvre une méthode de Monté-Carlo basée sur le principe de la "trempe" en mécanique statistique.

ABSTRACT

This paper deals with some techniques we have developped to implement - on vector processors such as CRAY-1S, Fujitsu VP200, Hitachi S810, ... - linear algebra algorithms for sparse matrices associated with finite element methods ; the particular algorithm we consider here is preconditionned conjugate gradient. We introduce a renumbering of unknowns in order to remove data dependencies and to increase parallelism. We consider first the case of "regular" matrix structures, encountered in the context of finite difference methods. In the general case of finite element methods, we must find vectors long enough to mask the overhead inherent to the indirect indexation.

In order to deal systematically with the search of parallel matrix structures and to complete heuristic techniques mentionned above, we introduce a formulation in terms of optimisation. We solve this problem by a simulated annealing method, derived by analogy with statistical physics of random systems.

VECTORIZING FINITE ELEMENTS METHODS

Jocelyne ERHEL
INRIA,Rocquencourt,France

Alain LICHNEWSKY
Universite Paris Sud,Orsay,France,
and INRIA,Rocquencourt,France

François THOMASSET
INRIA,Rocquencourt,France

Paper presented to the International Seminar on Computer Networking and Performance Evaluation, September 18-20, 1985, Tokyo, Japan.

ABSTRACT

This paper deals with some techniques we have developped to implement - on vector processors such as CRAY-1S, FUJITSU/VP200, HITACHI/S810,... - linear algebra algorithms for sparse matrices associated with finite element methods; the particular algorithm we consider here is preconditionned conjugate gradient. First of all we consider the case of "regular" matrix structures which we encounter in the context of finite difference methods, and we introduce a renumbering of unknowns in order to remove data dependencies. Then we extend this renumbering to the general case of finite element methods. Finally we present an optimization procedure based on Monte-Carlo techniques to improve the renumbering.

0. INTRODUCTION

Preconditionned conjugate gradient methods are very efficient algorithms for linear systems of equations, whose matrix is symmetric and positive definite (see /MeGo 1983/ and the bibliography therein); it involves iterated solution of linear systems, with a fixed matrix M, the so-called "preconditionner". Although solving a system with matrix M is much simpler than the original problem, it is generally lots or recurrences that inhibit parallelisation; this can be overcome by applying proper renumberings of the unknowns. Unfortunately we generally observe that, the more implicit is the preconditionner, the faster the convergence of conjugate gradient iterates.

The efficiency of the Preconditionned Conjugate techniques and its adaptation to vector processes has been the result of many studies, including Meijerink and Van der Vorst /MVV 1977,1981/; Schreiber and Wei-Pei Tang /SWT1983/; Rogrigue and Wolitzer /RoWo1982/; Concus, Golub and Meurant /CGM1982/; Johnson,Micchelli and Paul /JMP1982/;Meurant /Meu1985/.

In the case of finite difference methods a family of renumberings is easily found by applying coloring considerations; similar structures can be obtained within the context of finite element methods using an algorithm derived from Cuthill-McKee's reordering technique for reducing the bandwidth of a finite element matrix /CMK1969/.

We are then faced with two sorts of difficulties: one is related with the structure of sparse matrices in finite element methods, which may be dealed with by calls to SCATTER/GATHER routines. The other difficulty concerns the quality of the renumberings: finding an "optimal" or even a "good" renumbering appears to be a hard problem of a combinatorial nature. We propose in the last section to apply random perturbations to the ordering of the unknowns, in a Monte-Carlo style, so as to decrease some criterium on the diagonals of the matrix.

1. THE PRECONDITIONNED CONJUGATE GRADIENT

Suppose we need to solve the linear system:

(1) $A \cdot x = b$,

where x and b are vectors, A is an order-n symmetric, positive definite matrix. Let M be another symmetric positive definite matrix (the "preconditionner"). Then the solution x is found as the limit of the following iterative process (r, z, p, s are auxiliary vectors; alpha, beta, newbeta are scalars):

```
(STEP-0)(Initialization):
      r <- b - A.x
      let z be the solution of : M.z = r
      p <- z
      beta <- z.r,   s <- A.p
      alpha <- beta / p.s

(STEP-1):
      x <- x + alpha * p
      r <- r - alpha * s
(STEP-2):
      let z be the solution of : M.z = r
(STEP-3):
      newbeta <- z.r
      p <- z + (newbeta/beta) * p
      beta <- newbeta
(SETP-4):
      s <- A.p
      alpha <- beta / p.s
(STEP-5):
      IF NOT CONVERGED THEN GOTO STEP-1.
```

The nicest property of this algorithm is its potentiallly fast convergence; this depends on spectral properties of matrices A and M: in short the closer the matrix M is from the inverse of A, the faster the convergence.

So we have some flexibility in the choice of M: the two

extreme choices are: i/ M=A, in which case each iteration is equivalent to the original problem; ii/ M=Identity: we have the classical conjugate gradient of Hestenes and Stiefel /HeSt1952/, which suffers from slow convergence.

We have chosen the technique introduced by Meijerink and Van der Vorst /MVV1977/ known as "Incomplete Cholesky factorization", which is described in the following section.

2. CHOICE OF THE PRECONDITIONNER: INCOMPLETE CHOLESKY FACTORIZATION

First let us recall that Cholesky factorization of matrix A consists in computing a lower triangular matrix N such that:

$$A = N \cdot N^t \qquad (2)$$

Unfortunately the sparsity pattern of A is not conserved in matrix N: N is much more dense than A ("fill-in" phenomenon), leading to excessive storage requirements. (Typically in finite difference methods, A would be a 5- or 7-diagonal matrix, while a whole bandwidth would be filled in N).

Thus instead of (2), we write:

$$A = L \cdot L^t + R, \qquad (3)$$

where the residual matrix R and the lower matrix L are such that the structure of L is simpler than with (2). Then we shall take as a preconditionner :

$$M = L \cdot L^t .$$

In order to describe the structural constraints on L, it is useful to associate to a symmetric matrix A its labelled undirected graph: $G(A)=(V(A),E(A))$. The vertex set V(A) consists of n elements (one vertex per line of A), and the edge set E(A) describes the sparsity of A in the following way:

$$\{i , j\} \in E(A) \quad \text{iff} \quad A(i,j) \neq 0 \qquad (4)$$

Let there be given a priori the graph G(M), which we shall note $G(IC)=(V(A),E(IC))$. Then the set of constraints on L and R are given by:

a/ the only nonzeroes in L are those compatible with G(IC), that is: $L(i,j)=0$ if $\{i,j\} \notin 0$

b/ impose $R(i,j)=0$ if $\{i,j\} \in E(IC)$.

This construction is possible for M-matrices or diagonally dominant matrices; for more general positive definite systems, the shifting method of /Mant1978/ can be used.

Now the graph G(IC) can be quite arbitrary, but G(IC) will be usually much smaller than the perfect elimination graph G(M), so as to limit the amount of fill-in. A typical choice,

which we have generally followed in our experiments, is: G(IC) = G(A). The choice of graphs with more edges improves the convergence rate of the conjugate gradient algorithm (see for instance /MVV1981/).

Now one iteration conjugate gradient method with preconditionning consists of:

* dot products;
* linear combinations of vectors;
* (matrix X vector) products;
* solution of linear system with matrix M.

Clearly the most difficult part to perform in parallel is the last one, which involves linear recurrences (unless M were diagonal, which would yield poor convergence rate) . Thus the problem we have is to find an ordering for the unknowns of the system, which would allow an efficient vectorisation without destroying the convergence properties.

3.THE FINITE DIFFERENCE CASE

Suppose we have the following structure for A:

$$\begin{bmatrix}
A1 & C2 & & & & & & \\
B2 & A2 & C3 & & & & & \\
 & B3 & A3 & C4 & & & & \\
 & & B4 & A4 & C5 & & & \\
 & & & B5 & A5 & C6 & & \\
 & & & & B6 & A6 & C7 & \\
 & & & & & B7 & A7 & C8 \\
 & & & & & & B8 & A8
\end{bmatrix}$$

with Ai a tridiagonal sub-matrix, Bi a diagonal sub-matrix and Ci=Bi. Such is the case for instance in the discretisation of Laplace's operator on a rectangular domain by the 5-point finite difference scheme. Here the unknowns have received the "natural" ordering, i.e. one line of the mesh after another. (One line of the mesh corresponds to one submatrix Ai).

Now if we re-order the unknowns: first all the unknowns corresponding to mesh-points on ODD lines, then those on EVEN lines, the structure of the matrix becomes:

$$\begin{bmatrix}
A1 & & & & C2 & & & \\
 & A3 & & & B3 & C4 & & \\
 & & A5 & & & B5 & C6 & \\
 & & & A7 & & & B7 & C8 \\
B2 & C3 & & & A2 & & & \\
 & B4 & C5 & & & A4 & & \\
 & & B6 & C7 & & & A6 & \\
 & & & B8 & & & & A8
\end{bmatrix}$$

Note that all the tridiagonal blocks Ai have the same size. The structure of the incomplete factor L is similarly decoupled:

$$\begin{bmatrix} L1 & & & & & & & \\ & L3 & & & & & & \\ & & L5 & & & & & \\ & & & L7 & & & & \\ U2 & V3 & & & L2 & & & \\ & U4 & V5 & & & L4 & & \\ & & U6 & V7 & & & L6 & \\ & & & U8 & & & & L8 \end{bmatrix}$$

(Li=bidiagonal lower triangular sub-matrices, Ui,Vi=diagonal sub-matrices). Then for instance the solution of a system L.y=z is parallelized in an obvious fashion: (the vectors y and z being partitionned accordingly to the above pattern)

```
FOR ALL ODD i
    solve: Li . yi = zi
FOR ALL EVEN i
    solve: Li . yi = zi - Ui.y(i-1) - V(i+1).y(i+1)
```

Note that because all sub-matrices have equal size, we can vectorize "across blocks", i.e. put the loop on i inside the solver for Li.yi = zi or zi = Ui.y(i-1)...

GRID 2D				GRID 3D			
I×J	CRAY-1S	VP200	S810	I×J×K	CRAY-1S	VP200	S810
40×40	33.80	38.55	30.73	20×20×20	43.2	82.0	37.5
50×50	36.07	40.18	31.13	30×30×30	45.9	107.0	38.4
60×60	35.09	41.96	31.03	10×50×50	47.5	80.7	38.8
120×120	38.76	51.79	31.23	RESULTS ARE GIVEN IN MFLOPS/S			
200×200	40.75	58.69	31.01				

Table 1

4. THE FINITE ELEMENT CASE

When using a Finite Element method on a general mesh, we get an irregular sparsity pattern of the matrix A . To obtain a structure similar to the previous one (i.e. the Finite Difference case), we use a renumbering technique, which exhibits paths in the graph G(A).

The Cuthill-McKee algorithm /CMK 1969/ (breadth-first numbering algorithm), can be performed in the following manner, leading to fronts and thus to a block-tridiagonal matrix :

```
            Initially, all nodes are unmarked

/step 0/    Pick up a set of nodes F0 (initial front)
            Mark the nodes in F0

/step 1/    Set k := k+1
            Fk = set of unmarked nodes adjacent to nodes in Fk-1
            Mark the nodes in Fk (kth front)

/step 2/    If some nodes remain unmarked then go to step 1
```

Now, the idea is to renumber the nodes within a front in order to give regular structures to submatrices Ai.

Consider for simplicity the case of triangular elements of degree 1 (P1 in 2-D domains).

If the fronts were paths, renumbering inside them would lead to tridiagonal submatrices Ai. Unfortunately, with general Finite Element meshes, this is not always true : therefore, we remove the spurious nodes from the fronts in order to obtain pathlike fronts. These nodes are numbered after all the nodes in the fronts.

Then, a zebra-like numbering of the fronts (themselves renumbered), yields a structure of the matrix similar to that of the Finite Difference case, with blocks marked E, F and G corresponding to the spurious nodes. The structure of the matrix A is given below :

$$\begin{bmatrix}
A1 & & & & C2 & & & & F1 \\
 & A3 & & & B3 & C4 & & & F2 \\
 & & A5 & & & B5 & C6 & & F3 \\
 & & & A7 & & & B7 & C8 & F4 \\
B2 & C3 & & & A2 & & & & F5 \\
 & B4 & C5 & & & A4 & & & F6 \\
 & & B6 & C7 & & & A6 & & F7 \\
 & & & B8 & & & & A8 & F8 \\
E1 & E2 & E3 & E4 & E5 & E6 & E7 & E8 & G
\end{bmatrix}$$

where $Ci = tBi$ $(i=2,8)$
$Fi = tEi$ $(i=1,8)$

The diagonal blocks Ai are themselves tridiagonal, and this corresponds to 2-diagonal blocks in the factorized triangular matrix L. The solution of the system L.x = y could be parallelized in the same fashion as in the regular case. Now, because the submatrices Ai have not the same size, it is not possible to vectorize accross the blocks. Therefore, we consider all odd-numbered (resp. even-numbered) blocks as one tridiagonal block X1 (resp. X2), and we are lead to solve first-order linear

recurrences, that can be vectorized by cyclic reduction for instance.

The blocks Bi and Ci, coupling the nodes of odd and even fronts, are sparse banded submatrices, with bandwith less than the maximum front size. We treat the submatrices Bi (resp. Ci) as one block Y (resp. Z), which is stored according to a sparse-diagonal-oriented scheme (storing only non-zero coefficients in diagonal order and using pointers). It is still possible to vectorize the product by the block Y or the block Z, but with the use of indirect adressing (GATHER/SCATTER operations). The performances are therefore degraded, compared to the regular case with filled diagonals and constant-increment vectors.

It should be noted that the product of the matrix A by a vector needs also to compute the product of Y and Z by a subvector.

In the next section, we introduce heuritiscs to improve the structure of the submatrix Y and to optimize the vectorization.

The blocks Ei and Fi containing the spurious nodes are sparse, with no regular pattern : we did not try to vectorize the operations on them (to treat them as dense blocks is slower by experience). The overhead due to those blocks remains acceptable, because, by experience, the number of spurious nodes is much smaller than the total nomber of unknowns (there is about a factor of 50). In the new structure of A, the blocks corresponding to the spurious nodes are Si (i=1,2) and Ti (i=1,2) (coupling spurious nodes with fronts) and S (coupling spurious nodes between them).

The structure of the matrix finally used is then :

$$\begin{bmatrix} X1 & Z & T1 \\ Y & X2 & T2 \\ S1 & S2 & S \end{bmatrix}$$

Where Ai (i=1,2) are tridiagonal
Y is "sparse-diagonal" - Z = tY
Si (i=1,2) and S are sparse (by rows or columns)

Remark :

In the case of elements of higher degree, we can use multicoloring as in /SWT 1983/ to obtain a similar pattern.

Results :

Two meshes have been tested :
- RING : a ring shaped domain with 649 unknowns.
- BINACA : a discretization of an industrial interest, with 1702 unknowns.

We have measured execution time and speed of convergence for both Dirichlet and Neumann boundary conditions. We have compared the

present method with renumbering and restructured matrix (FRONT), with a sparse method without renumbering (SPARSE). The algorithm considered in both cases is Preconditionned Conjugate Gradient by Incomplete Cholesky. We give times in seconds on the CRAY-1S.

DOMAIN	B.C.	SPARSE			FRONT		
		# ITER	CPU(Δ)	MFLOPS/S	# ITER	CPU(Δ)	MFLOPS/S
RING(649)	D	14	0.29	1.5	21	0.07	5.8
	N	34	0.88	1.5	54	0.25	5.9
BINACA(1702)	D	30	1.92	1.5	55	0.64	5.5
	N	57	3.86	1.5	88	1.13	5.6

Table 2

We have also tried a slighty different method with another storage scheme for submatrix Y and therefore another algorithm to compute the product of Y and tY by a subvector.

We give in the table below the obtained results, both on CRAY-1S and on FUJITSU VP200. It should be noted that the program was not tuned for VP200 ; in particular, the solution of first order linear recurrences was not vectorized. (1)

DOMAIN	CRAY-1S		VP200	
	CPU(Δ)	MFLOPS/S	CPU(Δ)	MFLOPS/S
RING(649)	0.059	6.9	0.051	8.1
BINACA(1702)	0.41	7.7	0.31	10.2

Table 3

5. THE RATIONALE FOR AN OPTIMIZATION PHASE.

Our motivations to improve the above given ordering by an optimisation technique are twofold. First, we feel that this may ultimately lead to a systematic treatment of the subject, and that we will be able to use adequate sets of constraints to impose our structural requirements on the matrices. Our second reason is that we want to be able to construct similar methods in the 3 Dimensional case, and that our current "front" heuristics will then have to be iterated, giving first surfaces and then lines. In this process, many "spurious" points are created departing from the appropriate matrix structure. The purpose of the optimization phase is then to regain control over these irregularities.

The optimization problem can then be stated in the following way:

---

(1) Up to date results will be presented at the conference

Given a matrix structure and an initial numbering of the unknowns U ,which satisfies a set of constraints U$\in$ Uad. (1)

Find a numbering U which satifies the same constraints U$\in$ Uad, while minimizing a given objective functional F(U).

The functionals F that we have currently investigated are the following:

a/ Number of diagonals in the matrix.

b/ Weigthted number of diagonals, where the weight depends on the diagonal location.

c/ $- \left(\sum_{i=-n,..,n} (\text{number of elements on diagonal}(i))^2 \right)$

d/ $- \sqrt{\left(\sum_{i=-n,..,n} (\text{number of elements on diagonal}(i))^2 \right)}$

All these functional express that we want to minimize the number of points that do not enter the by-diagonals organization of the matrix. This is quite obvious for the last two criteria also, if one keeps in mind that the total number of coefficients is invariant through all renumberings :

$$\sum_{i=-n,\ldots,n} \text{number of elements on diagonal}(i) = \text{constant}.$$

The last one seems to express our requirement in a less direct way that the two first ones, however, our first experiments have shown its practical value. The two "obvious" first ones have the drawback that the optimization method has little or no way to sense transformations that do not result in the creation or disparition of a diagonal. The third tends to favour diagonals with a large number of elements, which is certainly one of our qualitative objectives.

This choice leaves us with a hard problem. We have not worked out the details concerning the actual complexity of our constrained problems, but there are available results pertaining to related problems :

- there seems to be no efficient way to test for bandwidth 3. Which means that, even if one of the permutations of the matrix was tri-diagonal, U may be too hard to construct for practical matrix sizes /Tarj1976/.

- The minimum bandwith problem is known to be NP-complete.

6. A MONTE-CARLO / "SIMULATED ANNEALING" MINIMIZATION ALGORITHM.

---

(1) This constraints express the fact that the renumbered matrix has the structure given by figure 4.

It now seems quite natural to try to combine a heuristic construction method, which we had as a starting point, to a heuristic minimization technique of Monte-Carlo type. This approach, which is fully justified in statistical physics, has shown to be of large practical value for several hard problems /VeKi1983/. It then has mostly a heuristic justification, and we have, so far, made several experiments on simple configurations of our problem. We will now describe the method as well as typical results on our simple problems. Further work is being done in order to apply the method to full scale problems, to combine with the former heuristics, and to determine the most appropriate functionals. (2)

Our minimization algorithm exploits the following analogies with statistical physics:

atomic positions	configuration given by admissible permutations U
internal energy	objective functional F(.)
cool into low energy state	approximately minimize the functional F(.)
absolute temperature	parameter T in following algorithm

It proceeds along the following lines:

ANNEALING ALGORITHM:

Starting at step n with configuration U(n), we select an arbitrary random transposition of indices in the matrix which transforms U(n) into an admissible configuration U.

We take U(n+1) = U (3)
if F(U) = F(U(n))
or else with probability exp((F(U(n)) - F(U))/T).
Otherwise U(n+1) is set equal to U(n) (4).

This is repeated letting T go slowly to 0 so as to make the configurations corresponding to low values of the objective functional most probable.

We shall now describe the typical behaviour of this algorithm on two very simple cases. Although we plan to combine this algorithm with adaptations of Cuthil-McKee ordering in the long run, this preliminary study tests the intrinsic behaviour of the optimization process. In these two experiments, we started with the same arbitrary permutation of a 100 by 100 matrix, derived from a finite difference 5 point star on a square mesh. The selected criterion was functional (c).

---

(2) Up to date results covering these topics will be presented at the Conference.
(3) i.e. "we accept the new configuration"
(4) i.e. "we reject the new candidate".

Initial state

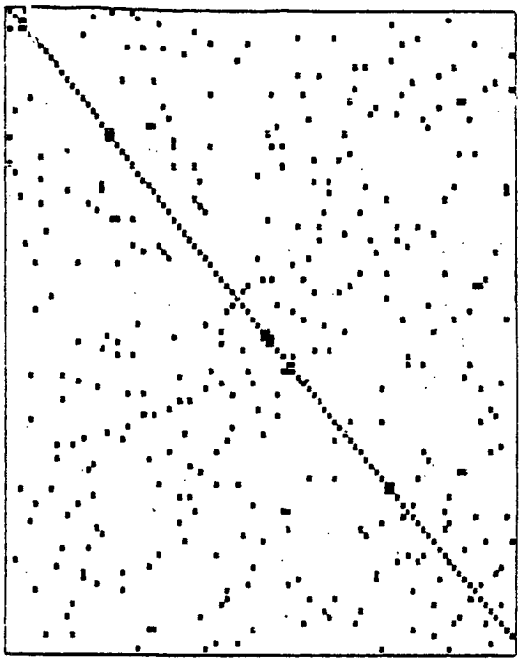
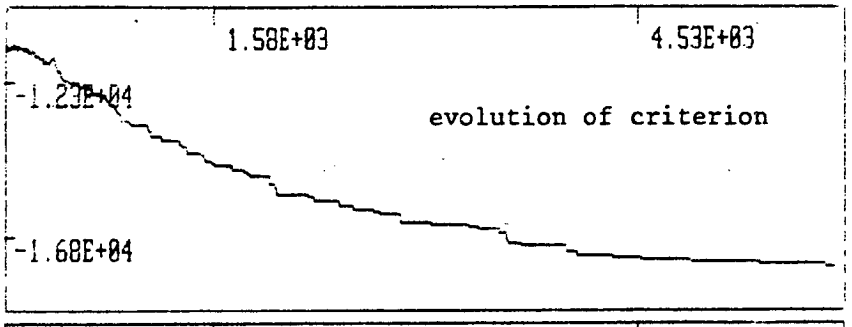


Figure 1



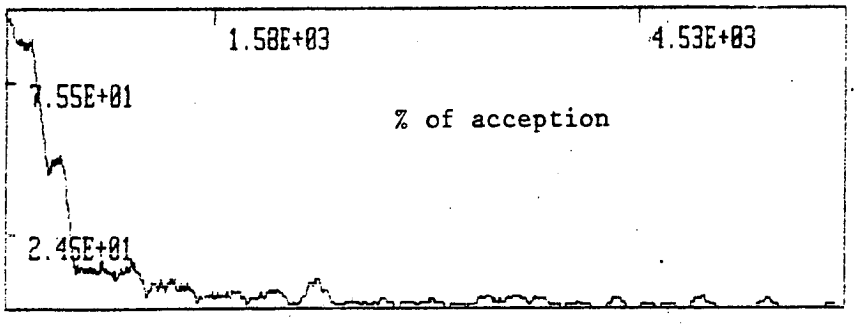
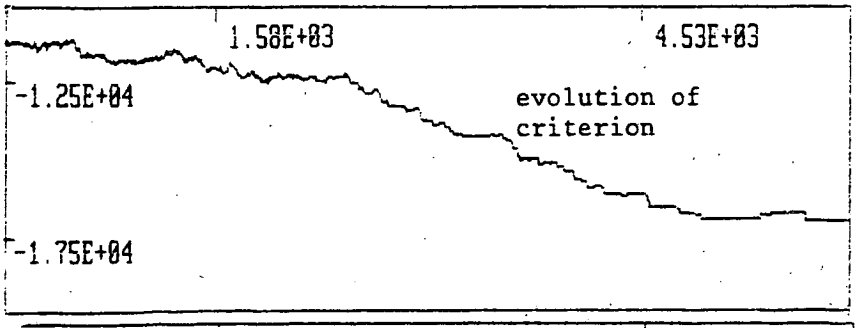


Figure 2



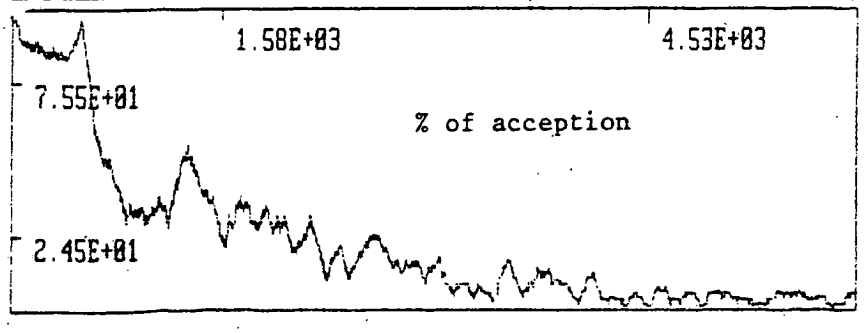


Figure 3

The minimization in these two examples were conducted as follows:

Ex1: (Figure 2):	200 iterations with T=100	
	200 iterations with T= 25	
	8500 iterations with T= 10.	
Ex2: (Figure 3):	500 iterations with T=100	
	12000 iterations with T= 25	

Two aspects of the results seem interesting. First, the algorithm succeeds in reducing the optimization criterion, even if we remain quite far from the absolute minimum which was expected due to the nature of the problem. The other aspect is that the matrices show quickly a lot of structure, especially as two strongly populated diagonals are created. This structure was not apparent in the initial state (Figure 1) .

REFERENCES

/Axel 1976/
AXELSSON (O.)
"Solution of linear systems of equations: iterative methods",in Sparse Matrix Techniques, V.A. Barker Ed., Springer.

/CGM1982/
CONCUS (P.), GOLUB (G.H.), MEURANT (G.) /CGM 1982/
"Block preconditionning for the conjugate method", Report LBL-14856, Stanford University, Stanford CA.

/CMK1969/
CUTHILL (E.),MAC KEE (J.)
"Reducing the bandwidth of sparse symmetric matrices", Proc. ACM National Conference, 1969.

/Erhe1983/
ERHEL (J.)
"Parallelisation d'un algorithme de gradient conjugue preconditionne",
Rapport de Recherche INRIA n0. 189, Fevrier 1983.

/HeSt1952/
HESTENES (M.R.),STIEFEL (E.)
"Methods of conjugate gradient for solving linear systems", NBS J. Res. 49 (1952), pp 409-436.

/JoMP1982/
JOHNSON (O.G.), C.A. MICHELLI & G. PAUL /JMP 1982/
"Polynomial preconditionning for conjugate gradient calculations", IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1982

/Lich1983/
LICHNEWSKY (A.)

"Some vector and parallel implementations for
preconditionned conjugate gradient algorithms",
NATO Advanced Research Workshop on High-Speed Computations,
20-22 June 1983, Julich, F.R.G.

/Mant1978/
MANTEUFFEL (T.A.)
"The shifted incomplete Cholesky factorization",
SAND/78/8226, Sandia Lab., Albuquerque, NM.

/MVV 1977/
MEIJERINK (J.A.), VAN DER VORST (H.A.)
"An iterative solution method for linear systems
of which the coefficient matrix is a symmetric M-matrix",
Math. Comp., Vol. 31, pp 148-162.

/MVV 1981/
MEIJERINK (J.A.), VAN DER VORST (H.A.)
"Guidelines for the usage of incomplete decomposition
in solving sets of linear equations
as they occur in practical problems",
J. Comp. Ph., Vol. 44, pp 134-155.

/MeGo1983/
MEURANT (G.),GOLUB (G.H.)
"Resolution numerique des grands systemes lineaires",
Eyrolles, Paris, June 1983.

/Meur1985/
MEURANT (G.)
"The block preconditionned conjugate gradient method on
vector computers",to appear in BIT.

/Reid1971/
REID (J.K.)
"On the method of conjugate gradients for the solution of
large sparse systems of equations",
pp 231-254 in "Large sparse sets of linear equations",
ed. J.K. Reid, Academic Press, 1971.

/Rowo1982/
RODRIGUE (G.), WOLITZER (D.)
"Preconditionning by incomplete block cyclic reduction",
Research Report UCID-19502, LLNL, Livermore, CA.

/SWT 1983/
SCHREIBER (R.), WEI-PEI-TANG
"Vectorizing the conjugate gradient method",
to appear.

/Tarj1976/
TARJAN (R.E.)
"Graph Theory and Gaussian Elimination",
in Sparse Matrix Computations, Bunch & Rose eds.,

Academic Press, New York.

/VDV 1983/
VAN DER VORST (H.A.)
"On the vectorization of simple ICCG methods",
First Int. Coll. on Vector & Parallel Computation,
AFCET-GAMNI-ISINA, Paris.

/VeKi1983/
VECCHI (M.P.), KIRKPATRICK (S.),
"Global wiring by simulated annealing",
IEEE Trans CAD, Vol CAD-2, Oct. 1983, pp 215-222

Imprimé en France

par

l'Institut National de Recherche en Informatique et en Automatique