DEPARTMENT OF APPLIED MATHEMATICS

INSTITUTE OF GEONICS Ostrava

PCG – 1 iterative solver

R. Blaheta and R. Kohut

Rep.9501, June 1995

Institute of Geonics, Academy of Sciences of the Czech Republic
Studentská 1768, 708 00 Ostrava – Poruba, Czech Republic
phone: +42-69-691 86 51, fax: +42-69-691 94 52
e–mail: blaheta@ugn.cas.cz, kohut@ugn.cas.cz.

1 Introduction

We shall describe the progam PCG–1 for the preconditioned conjugate gradient method which was designed for the solution of linear systems arising from finite element analysis of 3D elasticity and elasto–plasticity problems. This program is a part of the IGN–GEM32 finite element software.

The presented code PCG–1 emphasizes simplicity, clearity and portability and can be easily modified for testing another preconditioners etc.. On the other hand the performance of this code can be substantially speeded up if specific features of the computer architecture, operating system and compiler are exploited, cf. [3].

Presented code enables the use of unpreconditioned or diagonally preconditioned conjugate gradient method but primarily it is designed for implementing the conjugate gradient method with preconditioning by incomplete factorization, SSOR or DD–IF. The DD–IF preconditioner is more efficient than SSOR but it can be unstable in some cases. From this reason the PCG–1 code enables an authomatic switch of the incomplete factorization preconditioners (primarily DD–IF is switched to SSOR if the DD–IF preconditioner is not stable).

2 Programming

PCG–1 iterative solver is programmed in FORTRAN 77 with emphasis on simplicity and portability. Historically, two versions of the program were developed for the use on

- personal computer (PC) under Microsoft FORTRAN 5.0 compiler,
- workstation (WS) IBM RISC 6000 under XL FORTRAN compiler.

The PC version works with the program rows which are not marked or marked by E, the WS version works with the program rows which are not marked or marked by D.

Note that the differences in the two versions are not due to differences in the compilers but due to the development of the program.

3 Data files

The PCG–1 program works with the following data files:

FV.G32	– FORMATTED ASCI file which contains the basic information about the solved problem, especially grid parameters NX, NY, NZ and the number of degrees of freedom ND, see next Section.
FKBC.G32	– BINARY file for storage of the stiffness matrix.
FRBC.G32	– BINARY file containing the right hand side (load) vector.
FU.G32	– BINARY file for the vector of the unknows (displacement).
PCG.REP	– FORMATTED ASCI file containing report on the convergence of iterations .
FKBC.REP	– FORMATTED ASCI version of the file FKBC.G32
FRBC.REP	– FORMATTED ASCI version of the file FRBC.G32

4 Matrix storage

IGN–GEM32 finite element software exploits regular structured grids which, simply speaking, are the grids arising as a result of deformation of some regular rectangular grid (the pattern grid).

The nodes of our grids can be therefore easily indexed by the triples (I,J,K),

$$1 \le I \ \le NX, \quad 1 \le J \le NY, \quad 1 \le K \ \le \ NZ$$

or enumerated by 1D numbers

$$INDN = I \ + (J-1) * NX + (K \ -1) * NX * NY.$$

For 3D elasticity problems, we have additionally 3 degrees of freedom (DOF) in each node. These DOF correspond to the displacements in the directions X, Y, Z . The overall number of degrees of freedom is

$$ND = 3 * NX * NY * NZ$$

and the degrees of freedom at the node with index INDN can be indexed by the numbers

$$
\begin{aligned}
INDX &= 3*INDN-2,\\
INDY &= 3*INDN-1,\\
INDZ &= 3*INDN.
\end{aligned}
$$

The discretization is given by the decomposition of the domain into eight-node bricks and consequently into tetrahedral finite elements. Hence, the stiffness matrix (each brick is divided into 6 tetrahedra) has all nonzero entries within a 27–node (81 – DOF) regular stencil. Supposing the symmetry of the matrix, we can additionally store only the upper triangular part of the matrix. It can be done row-by-row by using regular 42 element stencil for storage of the nonzero matrix entries.

The row numbers are given by the order of the records, column numbers have not to be stored due to the regular stencil. More exactly, the column numbers for the regular stencil and for the row with index I will be the following:

If **I = INDX** for some DOF,
then the column numbers used in the 42 element stencil are subsequently:

$$\mathrm{J} = \mathrm{I},\ \mathrm{I}+1,\ \ldots,\ \mathrm{I}+5$$

and further

$$\mathrm{J} = \mathrm{K},\ \mathrm{K}+1,\ \ldots,\ \mathrm{K}+8 \quad \text{for } \mathrm{K} = \mathrm{I}+\mathrm{K1},\ \mathrm{I}+\mathrm{K2},\ \mathrm{I}+\mathrm{K3},\ \mathrm{I}+\mathrm{K4},$$

where

$$K1 = 3*NX-3, \quad K2 = 3*NX*NY-3*NX-3,$$

$$K3 = 3*NX*NY-3, \quad K4 = 3*NX*NY+3*NX-3.$$

If **I = INDY** for some DOF,
then the last entry in the stencil is not used and for the remaining positions, we have,

$$J = I, I \ +1, ..., I \ +4$$

and further

$$J = K, K+1, ..., K \ +8$$

for $\quad K \ = I-1+K1, \quad I-1+K2, \quad I-1+K3, \quad I \ -1+K4.$

If **I = INZ** for some DOF,
then the last two entries in the stencil are not used and

$$J = I, I \ +1, ..., I \ +3$$

and further

$$J = K, K \ +1, ..., K \ +8$$

for $\quad K = I-2+K1, \quad I-2+K2, \quad I-2+K3, \quad I-2+K4.$

In the FKBC.G32 file, one record involves three stencils for the rows which correspond to three degrees of freedom in one node. This enlarging of the record improves the speed of I/0 operations.

The use of the regular stencil enables us not to store the column numbers, which halves the storage demands (both column numbers and matrix entries require the same amount of storage – 4B). Because the decrease of the storage is very important (the matrix have to be transferred from the external memory within each iteration), we can also consider more economical storage for some special cases as e.g. rectangular grids, uniform division of the bricks to tetrahedral finite elements etc. cf. [3]. But these possibilities are not implemented in this PCG–1 code.

The use of the matrix storage scheme can be also seen from the subroutines for conversion of FKBC.G32 and FRBC.G32 BINARY files to ASCI files and especially from the subroutine MXV for matrix by vector multiplication.

5 Conversion programs

Program BASYS converts the BINARY files FKBC.G32, FRBC.G32 and FU.G32 into FORMATTED ASCI files FKBC.REP, FRBC.REP. and FU.REP. Program ABSYS converts FORMATTED ASCI files into BINARY files. These programs are documented by their listings. Note that the program BASYS can be also used for checking the storage efficiency , see the fille NULFKBC.REP.

ASCI files can be used for viewing, debuging and transfering, but we must not forget that eg. FKBC.REP is almost 6 time longer then FKBC.G32.

6 MXV subroutine

The subroutine MXV performs the operations

$$v \longleftarrow A\,u$$

$$s \;=< u, v >$$

where u, v are vectors and A is the matrix stored in the file FKBC.G32. The scalar $s \;=< u, v \;>$ is the inner product of u and v , $s \;= u^T v$.

7 PROGRAM ITER

Program ITER creates an environment which is necessary for calling PCG subroutine.

Note that running of this program requires proper selection of the parameter IDIM according to the size of problems which will be solved and according to the available computer memory. We can solve the problems for which

$$3 * NX * NY * NZ + 3 * NX * NY + 3 * NX + 5 \leq IDIM$$

where NX, NY, NZ are the grid parameters.

The program ITER have to be linked with the subroutines PCG, MXV, PCINIT, PCSOLV, RTIME and for the WS–version with the subroutine GMOPEN.

8 Subroutine PCG

Subroutine PCG performs the preconditioned conjugate gradient method for solving linear system

$$Au = f$$

with matrix A stored in the file FKBC.G32 and vector f stored in the file FRBC.G32. The solution vector u is stored in the file FU.G32.

The subroutine PCG first call the subroutine PCINIT described later which creates required preconditioner C.

Then the PCG subroutine works with the arrays u, r, w, v according to the following computational scheme:

I. Preliminary phase (IT = 0)

$r \longleftarrow f$	read f into r,
$u \longleftarrow$ init	set initial approximation which can be $u = 0$ or $C^{-1}r$ or readed from FU.G32,
$w \longleftarrow Au$	matrix by vector multiplication is performed by MXV subroutine,
$r \longleftarrow r - w$	the initial residual is stored in r,
$v \longleftarrow C^{-1}r$	the initial search direction is stored in w. $C^{-1}r$ is performed by the PCSOLV subroutine.

$srr0 = < r, r >$

$srw0 = < r, C^{-1}r >$

$sff = < f, f >$ are three computed inner products.

II. Repeated computational phase (IT=1,2, ...)

$w \longleftarrow Av,$
$s =< Av, v >=< w, v >,$
$a = \ srw0/s,$
$u \longleftarrow u + a * v$ new iteration,
$r \longleftarrow r - a * w$ new residual,
$w \longleftarrow C^{-1}r,$
$srr =< r, r >,$ $srw =< r, C^{-1}r >,$
$b = \ srw/srw0,$
$srw0 = srw \ ,$
$v \longleftarrow w + b * v$ new search direction.

III. Control phase performed after each iteration

Here norms of residual $\sqrt{srr}$, $\sqrt{srw}$ and the reduction factors are reported, the vector u is written into FU.G32 and the iterations are stoped if

$$\frac{srr}{sff} = \frac{< r, r >}{< f, f >} \leq (TOL)^2$$

The behaviour of iterative process can be seen from the protocol in the file PCG.REP.

9 PCINIT subroutine

We are interested in solving the system

$Au = f$

with matrix $A = D + L + L^T$ where D is the diagonal and L is the strictly lower triangular part of A.

For this system, we shall consider the following choices of the preconditioners C:

$IPC = 0 \Longrightarrow C = I \ ,$
$IPC = 1 \Longrightarrow C = X,$

$IPC = 2 \Longrightarrow C = (X + L)X^{-1}(X + L^T)$,
$IPC = 3 \Longrightarrow C = (X + \tilde{L})X^{-1}(X + \tilde{L}^T)$

Here IPC is the number of the preconditioner, X is a diagonal matrix which depends on the choice of preconditioner, $\tilde{L}$ is the matrix which can be obtained from $L = (l_{ij})$ by assigning zero to all entries in the positions (i, j) which correspond to DOF representing displacemens in different coordinate directions. The diagonal matrix X is computed by the subroutine PCINIT. More exactly

- if IPC $= 1$, we take the *diagonal preconditioning* with $X = D$.
- if IPC $= 2$, we take the *SSOR preconditioning* with $X = \xi D$ with a parameter ξ. For some theory concerning parameter ξ see e.g.[1] Here, we shall use the value $\xi = 2/3$
- if IPC $= 3$, we take the DD–IF preconditioning, which will be described more exactly in the following, see also [2].

For DD–IF preconditioning, the preconditioner C taken as the MIC(O)* modified incomplete factorization of the matrix $\tilde{A}$ which arises form A when we put zero all entries of A in the positions (i, j) where i, j correspond to nodal displacements in different directions.

The matrix X is computed from the criterion of equal rowsums,

$$\text{rowsum} \quad (C) = \text{rowsum} \quad (\tilde{A} + \tilde{D})$$

where $\tilde{D}$ is some suitable additional diagonal perturbation. If $X = \operatorname{diag}(x_1, ..., x_n)$ then the rowsum criterion gives the formula

$$x_i = a_{ii} + \tilde{d}_i + \sum_{j=1}^{i-1} a_{ij} \frac{w_j}{x_j},$$

where $\qquad w_j = -\sum_{k=j+1}^{n} a_{jk}.$ We take, cf. [2],

$$\tilde{d}_i = \begin{cases} p_1 a_{ii} & \text{if} \quad a_{ii} \geq 1.9\, w_i \\ p_2 a_{ii} & \text{if} \quad a_{ii} < 1.9\, w_i \end{cases}$$

with

$$
\begin{aligned}
p_1 &= \min \left\{ 0.01, \frac{1}{2}\pi^2 \left(\frac{1}{3}ND\right)^{-2/3} \right\}, \\
p_2 &= \sqrt{p_1}\,.
\end{aligned}
$$

In [2], it is discussed when this incomplete factorization is *stable*, i.e. when we obtain all x_i positive.

If the stability is lost then we can perform some modifications of the described algorithm or switch to another (stable) preconditioner. In our code we use switch to the SSOR preconditioning. If the number of preconditioning is set to be IPC=11 then the program PCG–1 first tries to create DD–IF preconditioner and if this incomplete factorization is not stable then it simply take the SSOR preconditioner.

10 SUBROUTINE PCSOLV

The subroutine PCSOLV performs solution of the system

$Cw = r$

with the chosen preconditioner C. Moreover, it computes the inner products $srr =< r, r >, \quad srw =< r, w >$.

For IPC = 0, we have

$$C = I \quad \text{so that} \quad w = r.$$

For IPC = 1, we have

$$C = \text{diag}(x) \quad \text{so that} \quad w_i = r_i/x_i \quad \text{for} \quad i = 1, ..., ND.$$

For IPC = 2 or 3 we have the factorization

$$C = (X + L)X^{-1}(X + L^T)$$

or

$$C = (X + \tilde{L})X^{-1}(X + \tilde{L}^T)$$

where both upper triangular matrices L^T and $\tilde{L}^T$ can be read from the file FKBC.G32.

The solution of the system $Cw = r$ is then performed in two steps.

First, we perform the forward substitution

$$w \longleftarrow r \quad \text{and} \quad w \longleftarrow (X + L)^{-1}w \quad \text{or} \quad (X + \tilde{L})^{-1}w$$

and then the backward substitution

$$w \longleftarrow (I \ + X^{-1}L^T)^{-1} \quad \text{or} \quad (I \ + X^{-1}\tilde{L}^T)^{-1}w.$$

The use of DD–IF preconditioner with $\tilde{L}$ differs from the use of SSOR preconditioner with L only in fact that some entries from FKBC.G32 are skipped. This is implemented by exploiting the mask constant KD.

11 TEST PROBLEMS

For testing and debugging we can use the following two problems.

Problem A: We solve the system of linear algebraic equations which arises from the finite element solution of the linear elasticity problem

- in the domain, which is the parallelepiped

  $$\Omega =< XMIN, XMAX > \times < YMIN, YMAX > \times < ZMIN, ZMAX >$$

- with isotropic elastic material described by its elastic modulus E [MPa], Poison's constant P, and the specific weight GAMMA [g cm^{-3}] .

- which is loaded by the weight of the material and the pressure PZ on the top side of the domain.

Figure 1: Rectangular and inclined rectangular grids

- with boundary conditions given by zero normal displacement component and zero tangential stress components on the boundary of the domain Ω with exception of the top side where the pressure PZ is supposed.

For the discretization, we

- divide the domain into eight–node bricks by a uniform rectangular grid or an inclined regular grid with grid parameters NX, NY, NZ, see Figure 1.
- divide each eight–node brick into six tetrahedra by the same manner, see Figure 2.

Note that for the problem description, we can use the special preprocessor ING32A or more general preprocessor TGRID [4]. For creation of the linear system, we can use the program FEMSYS [4].

Note also that for weightless material (GAMMA=0) we have analytical solution of the described problem which can be used to check the results. The analytical solution gives the displacement

Figure 2: A division of eight–node brick into six tetrahedra

$$
\begin{aligned}
u_x &= u_y = 0 & (1)\\
u_z &= k(Z - ZMAX) & (2)
\end{aligned}
$$

where $k = -PZ * (1 + P) * (1 - 2 * P) * E^{-1} * (1 - P)^{-1}$.

For debugging, we can use two special cases of the described problem with the following input data

XMIN = YMIN = ZMIN = 0 ,
XMAX = YMAX = ZMAX = 100,
E = 800, P = .25, GAMMA = 0 ,
PZ = 10,
NX = NY = NZ = 5.

These cases, which will be denoted by Ar and Ai, differ in the use of rectangular and inclined rectangular grids.

<u>Problem B:</u> We solve the system of linear algebraic equations which arises from the finite element solution of the linear elasticity problem

- in the domain

$$\Omega =< 0,1 > \times < 0,1 > \times < 0,1 >$$

- with isotropic elastic material given by the modulus E = 3.6 and Poisson's constant P = 0.2
- with displacement boundary condition on the whole boundary $\partial\Omega$
- and with a volume force with density Fx, Fy, Fz.

  The boundary conditions and the volume force correspond to the following exact solution of the problem:

$$\begin{aligned} u_x &= u_y = xyz(1-x)(1-y)(1-z), && (3) \\ u_z &= \sin(\pi x)\sin(\pi y)\sin(\pi z). && (4) \end{aligned}$$

For the discretization, we use the uniform rectangular grid and the same division into tetrahedra as in the previous problem.

For description of this problem, we can use another special preprocessor ING32B, and for checking the computed results, we can use TTG32B program.

For debuging we use the special case *Br* with rectangular grid with grid parameters NX=NY=NZ=5.

12 Concluding remarks

The presented code can be speed up by merging some loops, use move compact storage of the stiffnes matrix and increase the length of the record in the matrix storage file. The speed up of approximately 4 times for DD–IF is obtained by the code by Vít Lednický, see [3] .

Acknowledgement:

The work was supported by grants 201/94/1861 and 105/95/1273 of the Grant Agency of the Czech Republic.

References

[1] O. Axelsson, Linear system solvers, Cambridge Univ. Press, 1994.

[2] R. Blaheta: Displacement decomposition – incomplete factorization preconditioning techniques for linear elasticity problems, Numer. Lin. Alg. Appl. 1(1994), 107-126.

[3] Vít Lednický: Efficient implementation of the preconditioned conjugate gradient method, master thesis, TU Ostrava 1995, in Czech.

[4] R. Blaheta and R. Kohut: Benchmarks and programs for testing iterative solvers for 3D linear elasticity problems, Rep. 9502, Dept. Appl. Math., Institute of Geonics, Ostrava 1995.

Appendix A Solution of test problems 5x5x5

NAME= Problem Ar 5x5x5
DATE= 30.03.95

NO PRECONDITIONING

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.69660E+09	.00000E+00	.00000E+00	.69660E+09	.31809E+05
1	.31167E+09	.44742E+00	.44742E+00	.31167E+09	.14232E+05
5	.56278E+08	.65187E+00	.60460E+00	.56278E+08	.25698E+04
10	.28888E+08	.87514E+00	.72740E+00	.28888E+08	.13191E+04
15	.10189E+08	.81186E+00	.75453E+00	.10189E+08	.46526E+03
20	.22142E+07	.73691E+00	.75009E+00	.22142E+07	.10111E+03
25	.58322E+06	.76581E+00	.75321E+00	.58322E+06	.26631E+02
30	.13212E+06	.74306E+00	.75151E+00	.13212E+06	.60328E+01
35	.30550E+05	.74612E+00	.75073E+00	.30550E+05	.13950E+01
40	.14485E+05	.86135E+00	.76375E+00	.14485E+05	.66142E+00
45	.25721E+04	.70774E+00	.75731E+00	.25721E+04	.11745E+00
50	.23106E+03	.61757E+00	.74202E+00	.23106E+03	.10551E-01
55	.17990E+02	.60015E+00	.72784E+00	.17990E+02	.82146E-03
60	.28545E+01	.69199E+00	.72478E+00	.28545E+01	.13034E-03
65	.26870E+00	.62338E+00	.71643E+00	.26870E+00	.12270E-04
66	.14589E+00	.54295E+00	.71343E+00	.14589E+00	.66619E-05

COMPUTING TIME= 6.450

DIAGONAL SCALING

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.15829E+05	.00000E+00	.00000E+00	.73089E+02	.72281E+00
1	.22490E+05	.14208E+01	.14208E+01	.97761E+02	.10270E+01
5	.41784E+04	.65653E+00	.76614E+00	.18023E+02	.19080E+00
10	.16118E+03	.52151E+00	.63210E+00	.81958E+00	.73598E-02
15	.33286E+02	.72944E+00	.66301E+00	.14669E+00	.15199E-02
20	.27139E+01	.60571E+00	.64820E+00	.12704E-01	.12392E-03
25	.13519E+00	.54887E+00	.62699E+00	.62158E-03	.61730E-05

COMPUTING TIME= 2.340

SSOR PRECONDITIONER, X=.67*D

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.39807E+05	.00000E+00	.00000E+00	.25665E+03	.18177E+01
1	.84984E+04	.21349E+00	.21349E+00	.82928E+02	.38806E+00
5	.18037E+03	.38169E+00	.33981E+00	.11158E+01	.82362E-02
10	.99426E+00	.35340E+00	.34654E+00	.49436E-02	.45400E-04
12	.98481E-01	.31472E+00	.34102E+00	.59551E-03	.44969E-05

COMPUTING TIME= 3.000

DD-IF PRECONDITIONING

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.13348E+05	.00000E+00	.00000E+00	.13046E+03	.60952E+00
1	.84021E+04	.62945E+00	.62945E+00	.88432E+02	.38366E+00
5	.41551E+03	.47157E+00	.49961E+00	.28327E+01	.18973E-01
10	.21021E+01	.34739E+00	.41660E+00	.10912E-01	.95988E-04
13	.86782E-01	.34561E+00	.39903E+00	.46413E-03	.39627E-05

COMPUTING TIME= 3.040

NAME= Problem As 5x5x5

DATE= 30.03.95

NO PRECONDITIONING

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.88337E+09	.00000E+00	.00000E+00	.88337E+09	.38701E+05
1	.44662E+09	.50559E+00	.50559E+00	.44662E+09	.19567E+05
5	.72240E+08	.63418E+00	.60608E+00	.72240E+08	.31649E+04
10	.39654E+08	.88696E+00	.73319E+00	.39654E+08	.17373E+04
15	.16200E+08	.83607E+00	.76599E+00	.16200E+08	.70973E+03
20	.64732E+07	.83238E+00	.78208E+00	.64732E+07	.28360E+03
25	.27690E+07	.84380E+00	.79405E+00	.27690E+07	.12131E+03
30	.67814E+06	.75474E+00	.78736E+00	.67814E+06	.29710E+02
35	.38125E+06	.89121E+00	.80142E+00	.38125E+06	.16703E+02
40	.14747E+06	.82699E+00	.80457E+00	.14747E+06	.64608E+01
45	.46269E+05	.79308E+00	.80329E+00	.46269E+05	.20271E+01
50	.11096E+05	.75158E+00	.79796E+00	.11096E+05	.48612E+00
55	.36935E+04	.80252E+00	.79837E+00	.36935E+04	.16182E+00
60	.11851E+04	.79665E+00	.79823E+00	.11851E+04	.51922E-01

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
65	.21631E+03	.71164E+00	.79121E+00	.21631E+03	.94769E-02
70	.40616E+02	.71568E+00	.78556E+00	.40616E+02	.17794E-02
75	.76881E+01	.71685E+00	.78078E+00	.76881E+01	.33682E-03
80	.12524E+01	.69564E+00	.77517E+00	.12524E+01	.54868E-04
85	.20829E+00	.69853E+00	.77043E+00	.20829E+00	.91253E-05

COMPUTING TIME= 8.120

DIAGONAL SCALING

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.16908E+05	.00000E+00	.00000E+00	.74626E+02	.74075E+00
1	.25005E+05	.14789E+01	.14789E+01	.11568E+03	.10955E+01
5	.87482E+04	.76908E+00	.87653E+00	.37422E+02	.38327E+00
10	.18042E+04	.72924E+00	.79950E+00	.81123E+01	.79043E-01
15	.38828E+03	.73548E+00	.77757E+00	.16735E+01	.17011E-01
20	.97694E+02	.75883E+00	.77284E+00	.41093E+00	.42800E-02
25	.75069E+01	.59858E+00	.73434E+00	.32658E-01	.32889E-03
30	.18657E+01	.75696E+00	.73806E+00	.79615E-02	.81738E-04
35	.20831E+00	.64502E+00	.72399E+00	.93743E-03	.91263E-05

COMPUTING TIME= 3.370

SSOR PRECONDITIONER, X=.67*D

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.41581E+05	.00000E+00	.00000E+00	.24678E+03	.18217E+01
1	.10687E+05	.25702E+00	.25702E+00	.87032E+02	.46822E+00
5	.65815E+03	.49815E+00	.43640E+00	.35896E+01	.28834E-01
10	.66360E+01	.39876E+00	.41716E+00	.33104E-01	.29073E-03
14	.18376E+00	.40793E+00	.41450E+00	.88520E-03	.80506E-05

COMPUTING TIME= 4.230

DD-IF PRECONDITIONING

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.16091E+05	.00000E+00	.00000E+00	.14316E+03	.70495E+00
1	.86988E+04	.54061E+00	.54061E+00	.10995E+03	.38110E+00
5	.64718E+03	.52226E+00	.52588E+00	.37309E+01	.28354E-01
10	.37492E+01	.35693E+00	.43325E+00	.20366E-01	.16425E-03
13	.15306E+00	.34433E+00	.41088E+00	.73079E-03	.67058E-05

COMPUTING TIME= 3.020

NAME= Problem B 5x5x5

DATE= 30.03.95

NO PRECONDITIONING

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.20361E+01	.00000E+00	.00000E+00	.20361E+01	.78672E+00
1	.36738E+00	.18044E+00	.18044E+00	.36738E+00	.14195E+00
5	.18072E-01	.47095E+00	.38872E+00	.18072E-01	.69828E-02
10	.24884E-03	.42441E+00	.40618E+00	.24884E-03	.96151E-04
12	.20565E-04	.28747E+00	.38344E+00	.20565E-04	.79460E-05

COMPUTING TIME= 1.020

DIAGONAL SCALING

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.17408E+01	.00000E+00	.00000E+00	.93051E+00	.67264E+00
1	.35913E+00	.20630E+00	.20630E+00	.19197E+00	.13877E+00
5	.18281E-01	.47499E+00	.40202E+00	.97718E-02	.70638E-02
10	.46135E-04	.30228E+00	.34860E+00	.24660E-04	.17826E-04
11	.13921E-04	.30175E+00	.34406E+00	.74411E-05	.53790E-05

COMPUTING TIME= .930

SSOR PRECONDITIONER, X=.67*D

IT :	RES :	CONV.F. :	A.CONV.F. :	PC.RES :	RES/RHS
0	.18507E+01	.00000E+00	.00000E+00	.20204E+01	.71510E+00
1	.24205E+00	.13079E+00	.13079E+00	.19371E+00	.93526E-01
5	.34038E-03	.19365E+00	.17903E+00	.21452E-03	.13152E-03
7	.12108E-04	.18860E+00	.18171E+00	.66827E-05	.46783E-05

COMPUTING TIME= 1.820

```
DD-IF PRECONDITIONING
 IT :    RES    :    CONV.F.  :  A.CONV.F.  :   PC.RES  :     RES/RHS

    0  .90975E+00  .00000E+00   .00000E+00   .60508E+00  .35152E+00
    1  .15260E+00  .16774E+00   .16774E+00   .99234E-01  .58964E-01
    5  .16067E-03  .18013E+00   .17758E+00   .88045E-04  .62082E-04
    7  .95265E-05  .24350E+00   .19434E+00   .49100E-05  .36810E-05
COMPUTING TIME=     1.710
```

Appendix B MXV subroutine

```
c
      SUBROUTINE MXV (ND,NX,NY,NZ,U, V,S)
C
C
C     R. BLAHETA, 1/1994,11/1994
C
C     MULTIPLICATION: MATRIX (FILE FKBC.G32) TIME VECTOR U
C
C     PARAMETERS: ND   ... NUMBER OF EQUATIONS
C                 NX   ... PARAMETER OF GRID
C                 NY   ... PARAMETER OF GRID
C                 NZ   ... PARAMETER OF GRID
C                 U    ... =U(N+3*NX*NY+3*NX+5) INPUT VECTOR.
C                 V    ... =V(N+3*NX*NY+3*NX+5) OUTPUT VECTOR
C                 S    ... INNER PRODUCT <U,V>
C
C     DATA FILES : FKBC.G32 --> MXV
C
      INTEGER*2  NX,NY,NZ,L,K
      INTEGER*4  ND,IZ,J(42),JI(42),ID
      REAL*4     U(ND+3*NX*NY+3*NX+5),V(ND+3*NX*NY+3*NX+5)
      REAL*4     A(126)
      REAL*8     S,UI,VI
```

```
C
d     external gmopen
d     integer gmopen
d     integer io

      S=0.
C
C
C      INITIALIZATION
C

       DO 10 IZ=ND+1,ND+3*NX*NY+3*NX+5
10       U(IZ)=0.

       DO 20 IZ=1,ND+3*NX*NY+3*NX+5
20       V(IZ)=0.

C
C       INITIAL STENCIL
C

        DO 30 K= 1,6
30        JI(K) = -3+K
        DO 40 K= 7,15
40        JI(K) = 3*NX-12+K
        DO 50 K= 16,24
50        JI(K) = 3*NX*NY-3*NX-21+K
        DO 60 K= 25,33
60        JI(K) = 3*NX*NY-30+K
        DO 70 K= 34,42
70        JI(K) = 3*NX*NY+3*NX-39+K


e     OPEN(2,FILE= 'FKBC.G32',ACCESS='DIRECT',RECL=504)
d     io = gmopen(2,'FKBC',0)

C
```

```
C     loop over nodes
C

      DO 200 IZ=3,ND,3

          READ(2,REC=IZ/3) A

          DO 190 L=1,3

            ID=(L-1)*42

            IF(L.EQ.1) THEN
              DO 110 K=1,42
                J(K)=JI(K)+IZ
110           CONTINUE
            ELSE
              DO 111 K=1,41
                J(K)=J(K+1)
111           CONTINUE
              J(42)=J(42)+1
            ENDIF

            VI=V(J(1))
            DO 120 K=1,42
              VI=VI+A(K+ID)*U(J(K))
120         CONTINUE
            V(J(1))=VI

            UI=U(J(1))
            DO 130 K=2,42
              V(J(K))=V(J(K))+A(K+ID)*UI
130         CONTINUE


            S=S+UI*VI

190       CONTINUE
```

```
200     CONTINUE

      CLOSE (2)

      RETURN
      END
```

Appendix C PCINIT subroutine

```
c
      SUBROUTINE PCINIT ( ND, NX, NY, NZ, X, IPC)

C
C     SET UP OF PRECONDITIONER - DIAGONAL X
C
C     R.BLAHETA 10/94 VER.2.0
C
C     ND            NUMBER OF EQUATIONS
C     NX, NY, NY    GRID AND MATRIX STORAGE PARAMETERS
C     X             DIAGONAL MATRIX FOR THE PRECONDITIONER
C
C     IPC  = 0      NO PRECONDITIONING
C          = 1      DIAGONAL PRECOND.
C          = 2      SSOR PRECOND.
C          = 3      DD-IF PRECOND.
C          = 11     AUTOMATIC CHOICE : dd-if --> ssor
C
C     SSOR-see e.g. R.Blaheta, Incomplete factorization
C          preconditioning techniques for linear elasticity
C          problems, ZAMM 71(1991), T638-640
C     DD-IF -see R.Blaheta, Displacement decomposition-
C          incomplete factorization preconditioning techniques
C          for linear elasticity problems, NLAA 1(1994),
```

```
C          107-128


           INTEGER*4   ND, IZ, II, NZER, J(42), JI(42)
           INTEGER*2   IPC, NX, NY, NZ, IO, K,L,ID
           REAL*4      X(ND+3*NX*NY+3*NX+5)
           REAL*4      A(126), P1,P2,WI,XI,DI
d          EXTERNAL    GMOPEN
d          integer*2   GMOPEN

C
C     DD-IF PRECONDITIONING
C
      IF((IPC.EQ.3).OR.(IPC.EQ.11)) THEN

        P1=((4*ATAN(1.))**2)*((0.333*ND)**(-2./3.))/2.
        IF(P1.GT.0.01) P1=0.01
     P2=SQRT(P1)

E    OPEN(2,FILE= 'FKBC.G32',ACCESS='DIRECT',RECL=504)
D    io=gmopen (2,'fkbc',0)

C
C       INITIAL STENCIL
C

        DO 10 K= 1,6
10        JI(K) = -3+K
        DO 20 K= 7,15
20        JI(K) = 3*NX-12+K
        DO 30 K= 16,24
30        JI(K) = 3*NX*NY-3*NX-21+K
        DO 40 K= 25,33
40        JI(K) = 3*NX*NY-30+K
        DO 50 K= 34,42
50        JI(K) = 3*NX*NY+3*NX-39+K
```

```
C
C    computation of X
C

     NZER=0
        DO 60 II=1,ND
60        X(II)=0.

     DO 200  IZ=3,ND,3

          READ (2,REC=IZ/3) A
       DO 190 L=1,3

            ID=(L-1)*42

            IF(L.EQ.1) THEN
              DO 110 K=1,42
                J(K)=JI(K)+IZ
110           CONTINUE
            ELSE
              DO 111 K=1,41
                J(K)=J(K+1)
111           CONTINUE
              J(42)=J(42)+1
            ENDIF


         WI=0.
            DO 120  K=4,42,3
120        WI=WI-A(K+ID)

         IF(A(1+ID).GE.(1.9*WI))THEN
           DI=P1*A(1+ID)
         ELSE
           DI=P2*A(1+ID)
         ENDIF
```

```
          XI = A(1+ID)+X(IZ-3+L)+DI

          IF((XI.LE.0).AND.(IPC.EQ.11)) THEN
           WRITE(98, '(/,'' NONSTABLE DD-IF'',/)')
           IPC=2
           GO TO 300
          ENDIF
             IF((XI.LE.0).AND.(IPC.EQ.3)) THEN
              XI=A(1+ID)
              NZER=NZER+1
          ENDIF

          X(IZ-3+L)=XI
          WI=WI/XI

          DO 130  K=4,42,3
130         X(J(K))=X(J(K))+A(K+ID)*WI

190         CONTINUE
200       CONTINUE

          CLOSE(2)
          IF(IPC.EQ.11) IPC=3

        ENDIF

300     CONTINUE

C
C       DIAG.SCALING OR SSOR : X=DAIG(A)
C
      IF ((IPC.EQ.1).OR.(IPC.EQ.2)) THEN
E     OPEN (2, FILE='FKBC.G32', ACCESS='DIRECT', RECL=504)
d       io=gmopen(2,'fkbc',0)

        DO 310 IZ= 3, ND, 3
```

```
         READ(2,rec=iz/3) A
         X(IZ-2)=A(1)
         X(IZ-1)=A(43)
         X(IZ)  =A(85)
310     CONTINUE

        CLOSE(2)
      ENDIF

C
C     FOR SSOR :   X=(1/K)*DIAG(A), K=1.5
C
      IF (IPC.EQ.2) THEN
        DO 320 IZ=1,ND
320       X(IZ)=0.67*X(IZ)
      ENDIF

      IF(IPC.EQ.0)
     *  WRITE(98,'(/, '' NO PRECONDITIONING'',/)')
      IF(IPC.EQ.1)
     *  WRITE(98,'(/, '' DIAGONAL SCALING'',/)')
      IF(IPC.EQ.2)
     *  WRITE(98,'(/, '' SSOR PRECONDITIONER, X=.67*D '',/)')
      IF(IPC.EQ.3)
     *  WRITE(98,'(/, '' DD-IF PRECONDITIONING'')')
      IF(IPC.EQ.3)
     *  WRITE(98,'(/, '' NUMBER OF XI<=0 IS '',I7,/)') NZER


      RETURN
      END
```

Appendix D PCSOLV subroutine

```
c
      SUBROUTINE PCSOLV (IPC,ND,NX,NY,NZ,X,R,W,SRR,SRW)

C
C     W=INV(C)*R, SRR=<R,R>, SRW=<R,W>
C
C     R. BLAHETA  1/94, 10/94 - VER. 2.0, 11/94 VER. 2.1
C
C     ND            NUMBER OF EQUATIONS
C     NX, NY, NY    GRID AND MATRIX STORAGE PARAMETERS
C     X             DIAGONAL MATRIX FOR THE PRECONDITIONER
C
C     IPC  = 0      NO PRECONDITIONING
C          = 1      DIAGONAL PRECOND.
C          = 2      SSOR PRECOND.
C          = 3      DD-IF PRECOND.
C
           INTEGER*4   ND, II, J(42),I4,JI(42)
           INTEGER*2   IPC, NX, NY, NZ, K, L, IO,ID,KD
           REAL*4      X(ND+3*NX*NY+3*NX+5)
           REAL*4      R(ND+3*NX*NY+3*NX+5)
           REAL*4      W(ND+3*NX*NY+3*NX+5)
           REAL*4      A(126),REI
           REAL*8      SRR, SRW, RR, W1
D          EXTERNAL    GMOPEN
D          INTEGER*2   GMOPEN

C
C     NO PRECONDITIONING
C
      IF (IPC.EQ.0) THEN

          SRR=0.
```

```
         DO 1 I4=1,ND
            REI=R(I4)
            SRR=SRR+REI**2
            W(I4)=REI
1        CONTINUE
         SRW=SRR

      ENDIF


C
C     DIAGONAL SCALING
C
      IF(IPC.EQ.1) THEN

         SRR=0.
         SRW=0.
         DO 5 II=1, ND
           RR=R(II)
           W(II)=RR/X(II)
           SRR=SRR+RR*RR
           SRW=SRW+RR*W(II)
5       CONTINUE
      ENDIF

C
C     SSOR and DD-IF PRECONDITIONING
C
      IF((IPC.EQ.2).OR.(IPC.EQ.3)) THEN

C
C       INITIAL STENCIL
C

        DO 10 K= 1,6
10        JI(K) = -3+K
        DO 20 K= 7,15
```

```
20          JI(K) = 3*NX-12+K
        DO 30 K= 16,24
30          JI(K) = 3*NX*NY-3*NX-21+K
        DO 40 K= 25,33
40          JI(K) = 3*NX*NY-30+K
        DO 50 K= 34,42
50          JI(K) = 3*NX*NY+3*NX-39+K

C
C       W = R
C
        DO 60  II= 1,ND
60          W(II) = R(II)

C       MASK CONSTANT KD=1 for SSOR, KD=3 for DD-IF

        IF(IPC.EQ.2) KD=1
        IF(IPC.EQ.3) KD=3

E        OPEN(2,FILE= 'FKBC.G32', ACCESS= 'DIRECT',RECL=504)
D        io=gmopen(2,'fkbc',0)

C         FORWARD SUBSTITUTION

          DO 190  II=3,ND,3
            READ(2,rec=II/3) A

            DO 180  L=1,3

              IF(L.EQ.1) THEN
                DO 110 K=1,42
                  J(K)=JI(K)+II
110             CONTINUE
              ELSE
                DO 111 K=1,41
                  J(K)=J(K+1)
111             CONTINUE
```

```
              J(42)=J(42)+1
            ENDIF

            ID=(L-1)*42

            W1 =W(J(1))/X(II-3+L)
            W(J(1))=W1

            DO 120  K=KD+1,42,KD
120           W(J(K))=W(J(K))-A(K+ID)*W1

180       CONTINUE
190     CONTINUE

        DO 195 II=ND+1,ND+3*NX*NY+3*NX+5
195       W(II)=0.

C
C       BACKWARD SUBSTITUTION
C
        DO 290 II=ND,3,-3
          READ(2,rec=II/3) A

           DO  280  L=3,1,-1
             ID=(L-1)*42

             J(41)=JI(41)+II+L-2
             J(42)=J(42)+II+L-1
             DO 200 K=1,43-L
               J(K)=JI(K+L-1)+II
200           CONTINUE

             W1=0
             DO 210  K=KD+1,42,KD
210            W1= W1+A(K+ID)*W(J(K))

             W(J(1))= W(J(1)) - W1/X(II-3+L)
```

```
280          CONTINUE
290       CONTINUE

          CLOSE(2)

          SRR=0.
          SRW=0.
          DO 300 II=1,ND
            SRR=SRR+R(II)*R(II)
            SRW=SRW+R(II)*W(II)
300       CONTINUE

      ENDIF

      RETURN
      END
```