LABORATORY FOR
CONCURRENT COMPUTING SYSTEMS

COMPUTER SYSTEMS ENGINEERING
School of Electrical Engineering
Swinburne Institute of Technology
John Street, Hawthorn 3122, Victoria, Australia.

Parallelisation of the SDEM Distinct Element Stress Analysis Code

Technical Report 31-021

G.K. Egan
S.K. Tang
M.A. Coulthard

Computer Systems Engineering
School of Electrical Engineering
Swinburne Institute of Technology
John Street
Hawthorn 3122
Australia.

CSIRO
Division of Geomechanics
Mount Waverley 3149

Version 1.0 Original Document 10/12/90

Key Words: *parallel processing, seismic modelling, distinct element method*

Abstract:

The SDEM code models systems of interacting blocks of rock using the distinct element (DE) method. The DE method represents these systems as discontinuums with each block acting under Newton's Laws of motion. The data structures associated with the DE method are comprised largely of linked lists make the task of obtaining performance gains through vectorisation difficult. As the systems however are comprised of thousands of blocks there is however the potential of performing block interaction calculations in parallel.

This paper details the analysis and refinement steps used in implementing a parallel version of the SDEM.

PARALLELISATION OF SDEM DISTINCT ELEMENT STRESS ANALYSIS CODE

G.K. Egan, S.K. Tang and M.A. Coulthard

1. Introduction

Computational stress analysis is now widely used in geomechanics for back analysis of observed rock mass behaviour around surface and underground excavations and as a tool for excavation design in mining and civil engineering. The distinct element (DE) method, which represents a rock mass as a discontinuum, has been shown to be more realistic than finite element (FE) or boundary element (BE) (continuum) methods for modelling systems such as subsiding strata over underground coal mine excavations. However, whereas even 3D FE and BE analyses can now be performed readily on engineering workstations or the more powerful personal computers, the DE method generally requires orders of magnitude more computer processing time for analyses of comparable complexity. This has so far prevented the DE method from being applied widely in excavation design in industry.

The SDEM code models systems of interacting blocks of rock using the distinct element (DE) method. The DE method represents these systems as discontinuums with each block acting under Newton's Laws of motion. The data structures associated with the DE method are comprised largely of linked lists make the task of obtaining performance gains through vectorisation difficult. As the systems however are comprised of thousands of blocks there is however the potential of performing block interaction calculations in parallel.

This paper details the analysis and refinement steps used in implementing a parallel version of the SDEM.

2. The Distinct Element Method

The DE method of stress analysis was introduced in [1] to deal with problems in rock mechanics which could not be treated adequately by the conventional continuum methods. The earliest DE programs (e.g. program RBM in [2]) assumed that the blocks were rigid, so that all deformations within the system took place at the block interfaces. A second program described in [2], SDEM, allowed modelling of three simple modes of deformation of each block - two compressive and one shear mode. The DE programs which are most widely used at present are UDEC [3] and 3DEC [4]; the blocks in each of these may be modelled as fully deformable via internal finite difference zoning.

The main factor working against the adoption of these programs for routine engineering design of excavations in highly jointed rock is the very large computer execution times which are required for analyses involving substantial numbers of distinct elements.

In this paper we describe our experiences in parallelising SDEM a representative DE stress analysis code for the analysis of 2 dimensional systems of interacting, simply deformable polygonal DEs [2][5].

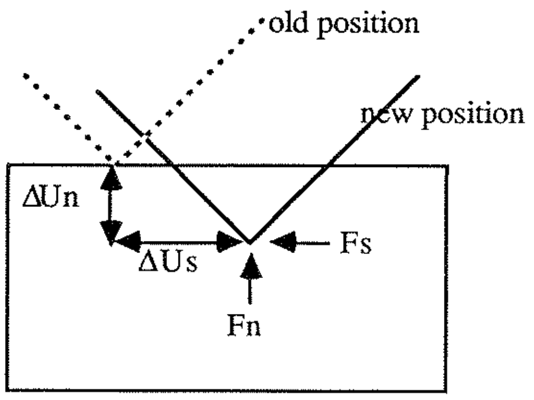
G.K. Egan is Professor of Computer Systems Engineering and Director of the Laboratory for Concurrent Computing Systems at the Swinburne Institute of Technology, John Street, Hawthorn 3122, ACSNet: gke@stan.xx.swin.oz.au.

S.K. Tang is a research scientist in the Laboratory for Concurrent Computing Systems at the Swinburne Institute of Technology, John Street, Hawthorn 3122, ACSNet: skt@stan.xx.swin.oz.au.

Dr. M.A. Coulthard is Principal Research Scientist at the CSIRO Division of Geomechanics, P.O. Box 54, Mt. Waverley 3149, ACSNet: mac@dogmelb.dog.oz.au.

2.1 Theoretical basis

Most DE programs are based on force-displacement relations describing block interactions and Newton's second law of motion for the response of each block to the unbalanced forces and moments acting on it.



$$\Delta Fn = \Delta Un \,.\, Kn \quad (1)$$

$$\Delta Fs = \Delta Us \,.\, Ks \quad (2)$$

where Kn = normal stiffness
ΔUn = incremental normal displacement

Ks = shear stiffness
ΔUs = incremental shear displacement

Figure 1. Block Interactions

The normal forces developed at a point of contact between blocks are calculated from the notional overlap of those blocks and the specified normal stiffness of the inter-block joints. Tensile normal forces are usually not permitted, i.e. there is no restraint placed upon opening of a contact between blocks.

Shear interactions are load-path dependent, so incremental shear forces are calculated from the increments in shear displacement, in terms of the shear stiffness of the joints. The maximum shear force is usually limited by a Mohr-Coulomb or similar strength criterion.

The motion of each block under the action of gravity, external loadings and the forces arising from contact with other blocks is determined from Newton's second law. A damping mechanism is also included in the model to account for dissipation of vibrational energy in the system.

The equations of motion may be integrated with respect to time using a central difference scheme to yield velocities and then integrated again to yield displacements. The velocity dependent damping terms have been omitted here for simplicity, but the same form of equations hold even when damping is included.

$$u'_i(t+\Delta t/2) = u'_i(t - \Delta t/2\,) + (\textstyle\sum F_i(t)/m + g_i)\,.\,\Delta t \quad (3)$$

$$u_i(t+\Delta t) = u_i(t) + u'_i\,(t+\Delta t/2)\,.\,\Delta t \quad (4)$$

where i = 1,2 correspond to x and y directions respectively;

u_i are the components of displacement of the block centroid;
F_i are the components of non-gravitational forces acting on the block;
g_i are the components of gravitational acceleration;
m is the mass of the particular block.

The equation of rotation for each block can be integrated similarly. Note that, in the integrated equations, block velocities and displacements are expressed explicitly in term of values at a previous time and so may each be calculated independently.

The calculation cycle proceeds, with the calculated displacements being used to update the geometry of the system, and thence to determine new block interaction forces. These, in turn, are used in the next stage of the explicit integration of Newton's equations.

This explicit time integration scheme is only conditionally stable. Physically, the time step must be small enough that information cannot pass between neighbouring blocks in one step, thus justifying the independence of the integrated equations of motion.

3. SDEM

SDEMH's main computational cycle (Figure 2) consists of computing the new position of blocks given the current forces acting on them (MOTION), and then from these positions, determine the stresses (STRESS) and new forces induced by blocks on their neighbours (FORD); these steps are repeated until the system stabilises. Contact lists, or lists of the other blocks a block is touching, are maintained to exploit locality; these data structures are the principal source of difficulties for vectorising compilers. If the displacement of any block is greater than some threshold then the contact lists of all blocks are updated (UPDAT); this deals with sudden events/collapses in the system occurring in a particular time step. The final annotated sources for CYCLE and its associated subroutines are listed in Appendix A.

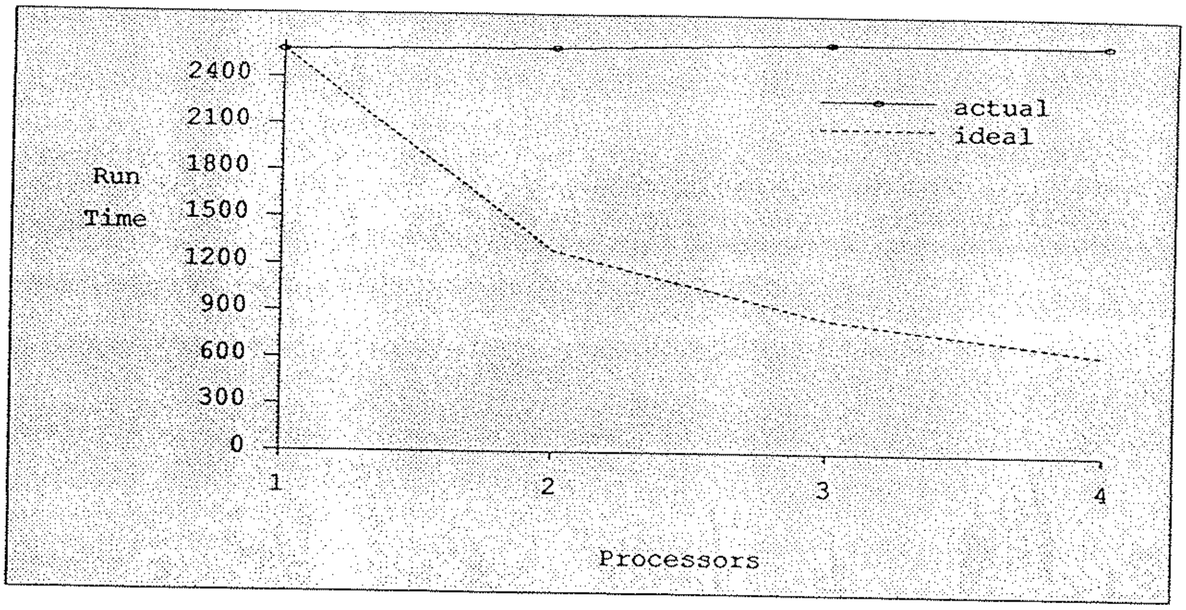
```
while not stable do

   if update required then
       call updat

   do each block
       call motion  {   compute motion
                        if current block velocity > udmax then
                            udmax = current block velocity
                        do each block corner
                            if corner then
                                call rebox
                    }
   do each block
       call stress

   do each block
       do each contact
           call ford
```

Figure 2. SDEM computational flow (sequential implementation)

4. Results

4.1 Automatic annotation

The Encore Multimax epf compiler was used to automatically annotate and compile the original FORTRAN source. The runtimes and speedup obtained are shown in Figure 3.



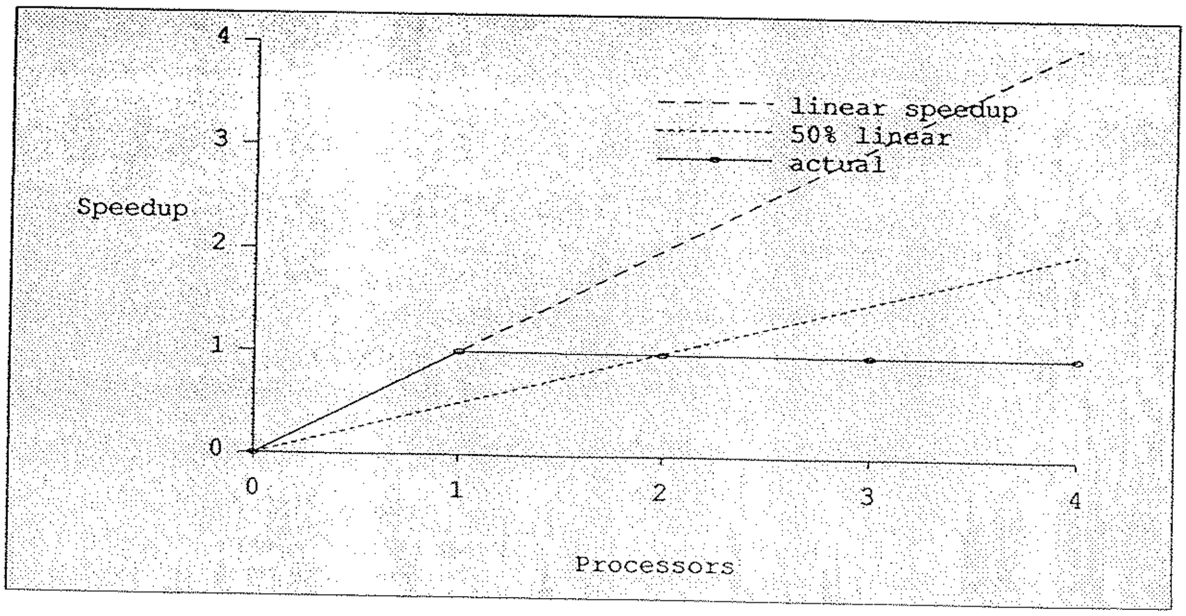


Figure 3. Speedup with automatic annotation (105 DE system for 20000 time steps)

It can be seen that not only was no speedup obtained, but there was actually some slow down. In the case of SDEM all major processes are called as subroutines e.g. computation of block motion. Inspection of the analysis listings (.lst) produced by epf showed that any statement involving a subroutine call was deemed to be not concurrent i.e. epf does not perform inter-procedural analysis. This is reasonable if one assumes that FORTRAN routines may be linked as precompiled modules and that they may involve hidden manipulation of structures through COMMON.

4.2 Explicit annotation

With one exception it was decided to reject the expansion of subroutines inline as a solution, as this would result in a cumbersome difficult to maintain source. The approach taken then was to explicitly annotate CYCLE and its associated subroutines in order of their contribution to the run time as identified by the gprof UNIX utility (Table 1). The original structure of cycle with its subroutine calls was generally preserved in the refinement process i.e. loops containing subroutine calls were forced to be parallel where it was determined that no data dependencies existed between loop cycles. Where dependencies did exist, they were removed as described in the following sections.

	%
FORD	55.8
MOTION	34.2
CYCLE	3.8
STRESS	3.6
FORBOU	2.3
UPDAT	0.1

Table 1. Contribution to run time of SDEM subroutines

4.2.1 The Motion and Stress Subroutines

Initial analysis suggested that subroutine MOTION would be most ammenable to parallelisation. It was also apparent that the STRESS subroutine could be incorporated into the into the MOTION subroutine. As can be seen from Table 1, MOTION contributes 34.2% of the processing time. This subroutine:

- computes each blocks new position and velocity;
- determines the velocity of the most rapidly moving block;
- recomputes the "bounding box" of a block if any corner moves.

The two inter-loop dependencies in the subroutine MOTION were:

- tracking maximum displacement for possible global consistency update
- tracking block corner displacement for possible recomputation of bounding box

The steps taken to resolve these dependencies were to:

(i) fuse the loops containing the MOTION and STRESS subroutines and inline the body of the stress computation in the MOTION loop;

(ii) retain all block velocities and transfer the calculation of maximum block velocity (UDMAX) to the body of CYCLE for determination of maximum block velocity by simple sequential search;

(iii) retain changes position of block corners and move the call to REBOX to CYCLE.

There are a number of alternatives to the method chosen in (ii) above including:

- 'locking' the UDMAX variable for comparision with the velocity of the current block and conditional update;
- using a simple 'flag' which would be set if any block reached a velocity requiring a call to UPDATE.

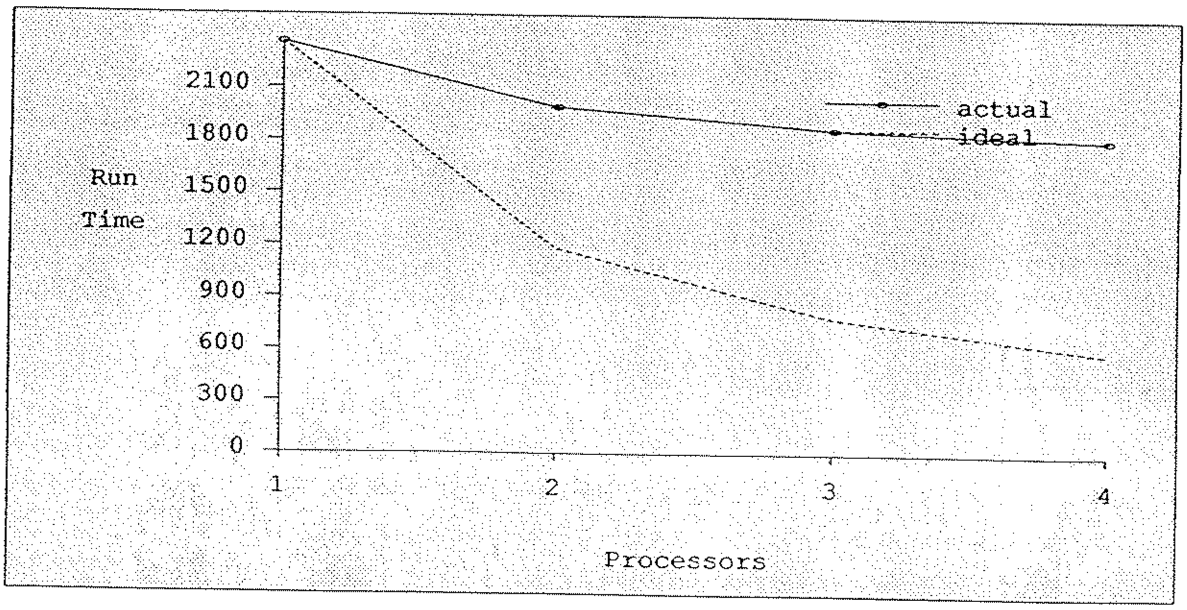
The first alternative has the potential for conflict on the lock and was not implemented. The second alternative does not preserve the algorithm although it would eliminate the sequential search for maximum block velocities.

```
doall blocks
    call motion {
                    compute motion
                    record change to block corner positions
                    record block velocities
                }

search for maximum block velocity
if maximum block velocity exceeded then
    set update required

doall blocks
    do each corner
     if corner moved then
        call rebox
```

sequential

Figure 5. Annotation of code involving MOTION

The resulting runtimes and speedup for this refinement are shown in Figure 6.



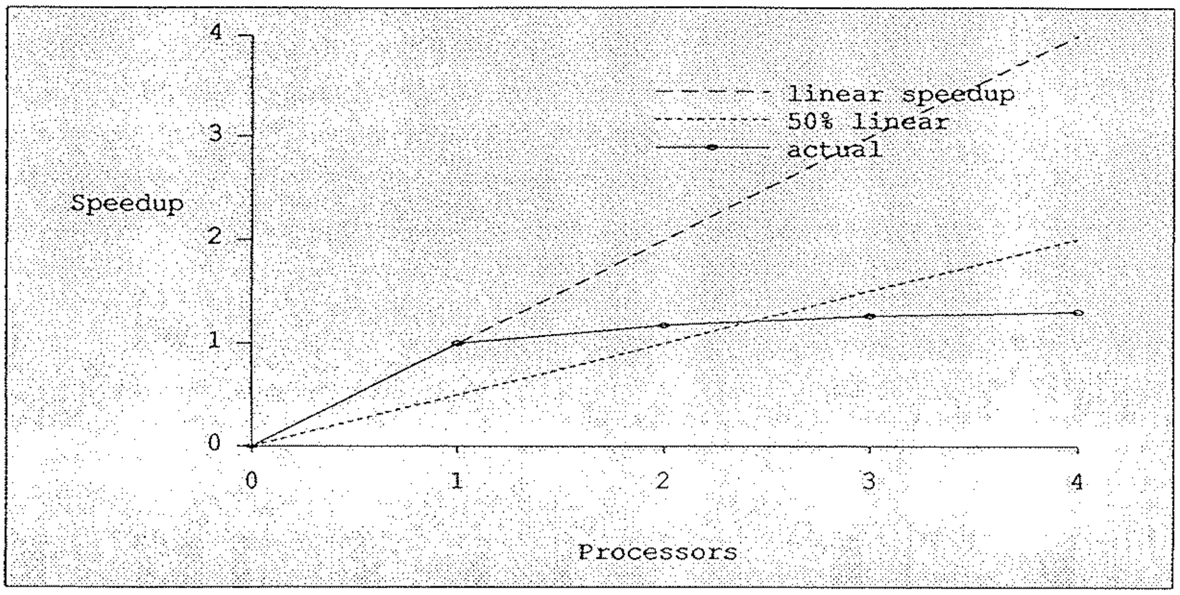


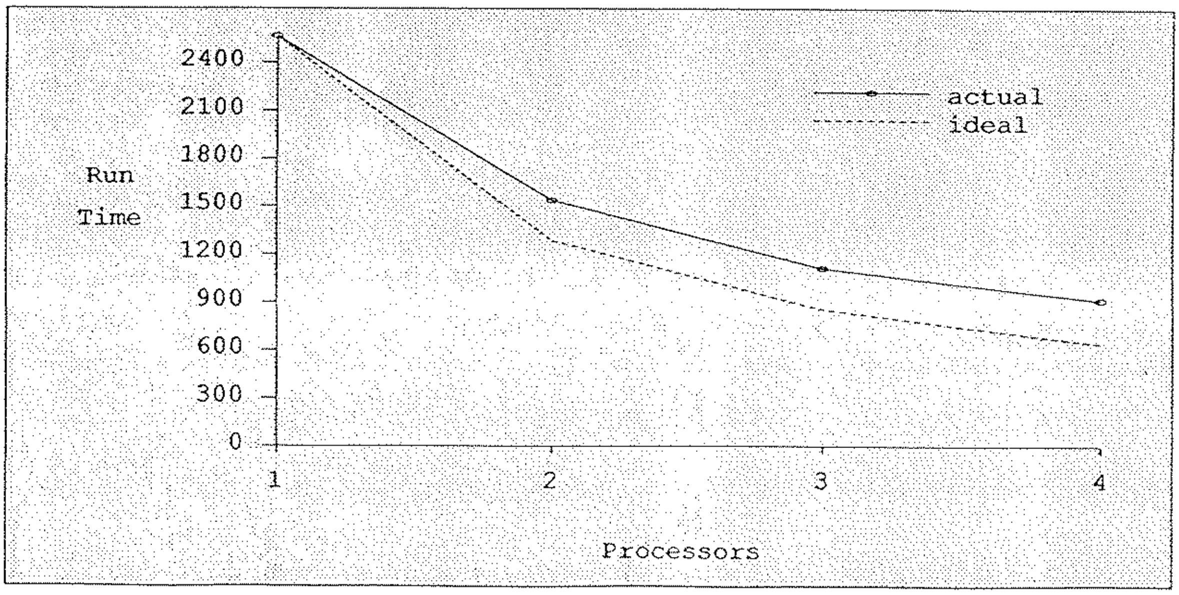
Figure 6. Speedup for MOTION refinement

4.2.2 Subroutine FORD

In subroutine FORD the forces on each block in the system are recomputed. The computation is done one block at a time for all blocks contacting the block of interest accumulating the total forces acting on the block. There is the potential for two processes to attempt to simulataneously update the same accumulation of forces. To resolve this the records of the two blocks involved in the contact are 'locked' during the accumulation step as shown in Figure 8.

```
doall blocks
   call ford {
              compute forces
              lock block records
                  accumulate inter block forces
              unlock block records
          }
```

Figure 8. Annotation of FORD



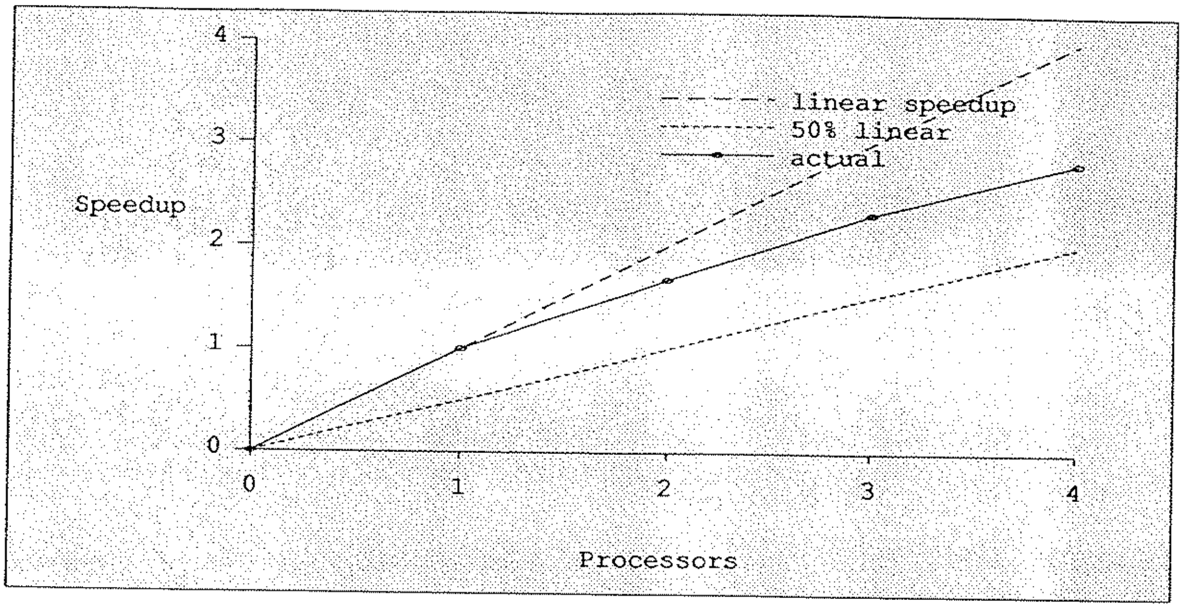


Figure 9. Speedup for FORD, MOTION and STRESS refinements

4.2.3 Subroutine UPDAT

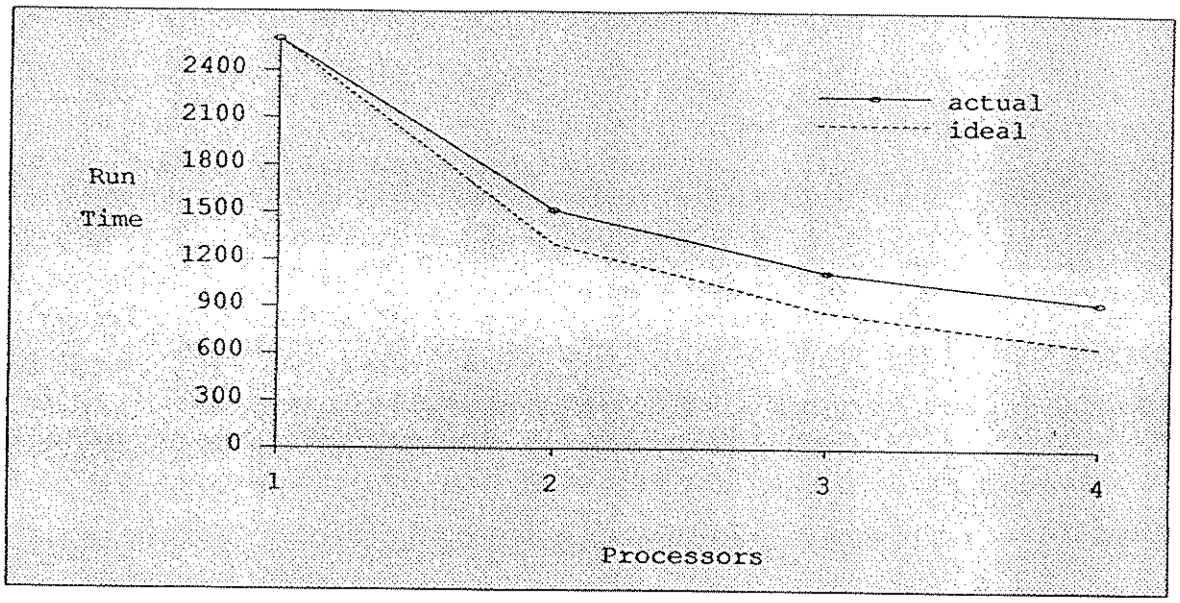
Subroutine UPDAT is called infrequently for the 105 DE system however for systems where there is major dislocation or collapse this routine may be called frequently. In UPDAT the lists of contacting blocks are updated resulting in release of list entries were blocks are no longer contacting and addition of new list entries were new contacts are formed.

There is the potential for conflict on the 'pointers' to the pool of free list entries. This is resolved by 'locking' the free list pointer (NEMPT) while releasing contact entries or acquiring new contact entries.

The runtimes and speedup for the UPDAT refinement and previous refinements is shown in Figure 11. In this case a call to update was forced on every CYCLE iteration to confirm that UPDAT had been succesfully parallelised.

```
call updat {
        doall blocks
            do each edge
                do each surrounding box in j direction
                    do each surrounding box in j direction
                        if block is contacting then
                            lock NEMPT
                            get new entry
                            unlock NEMPT
                            store new contact data

        for each block
            release old contact entries
}
```

Figure 10. Annotation of UPDAT



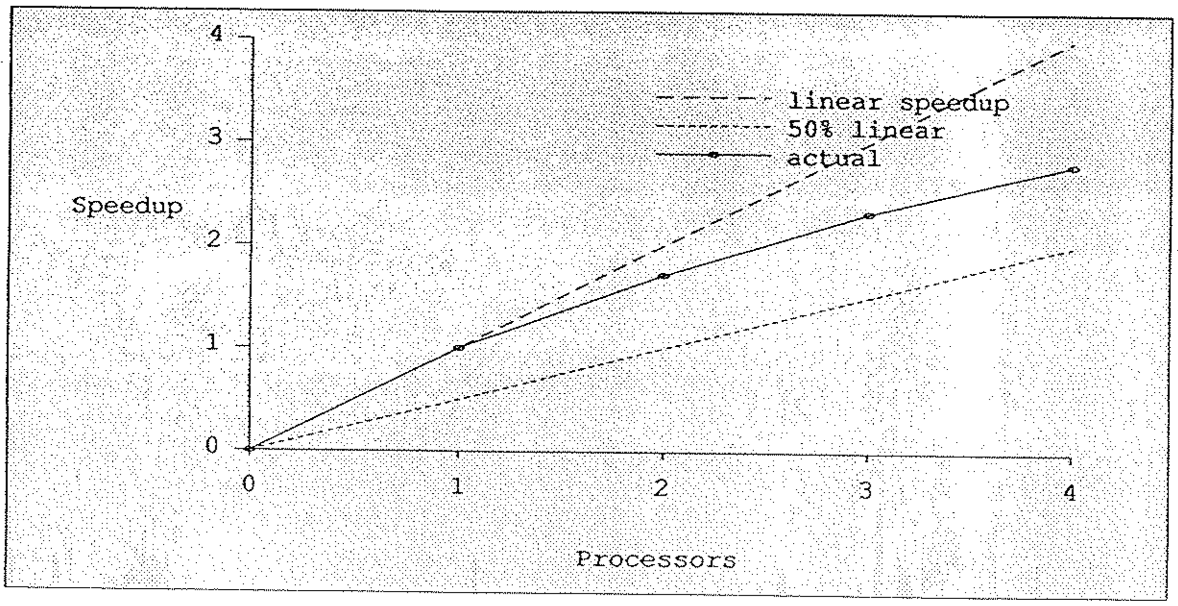
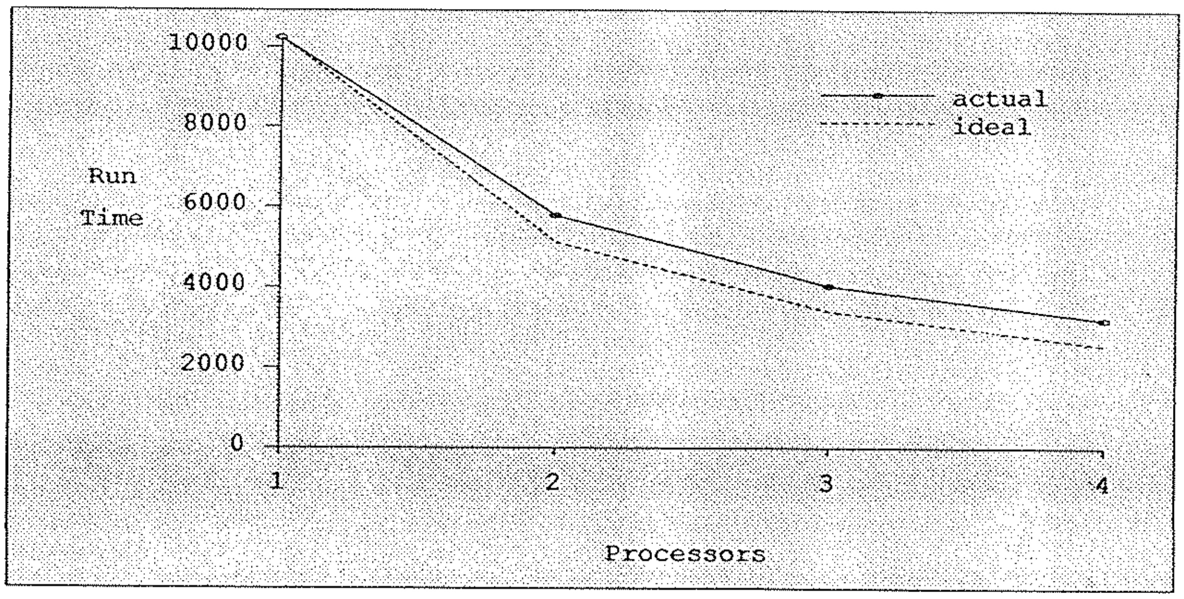


Figure 11. Speedup for UPDAT, FORD, MOTION and STRESS refinements



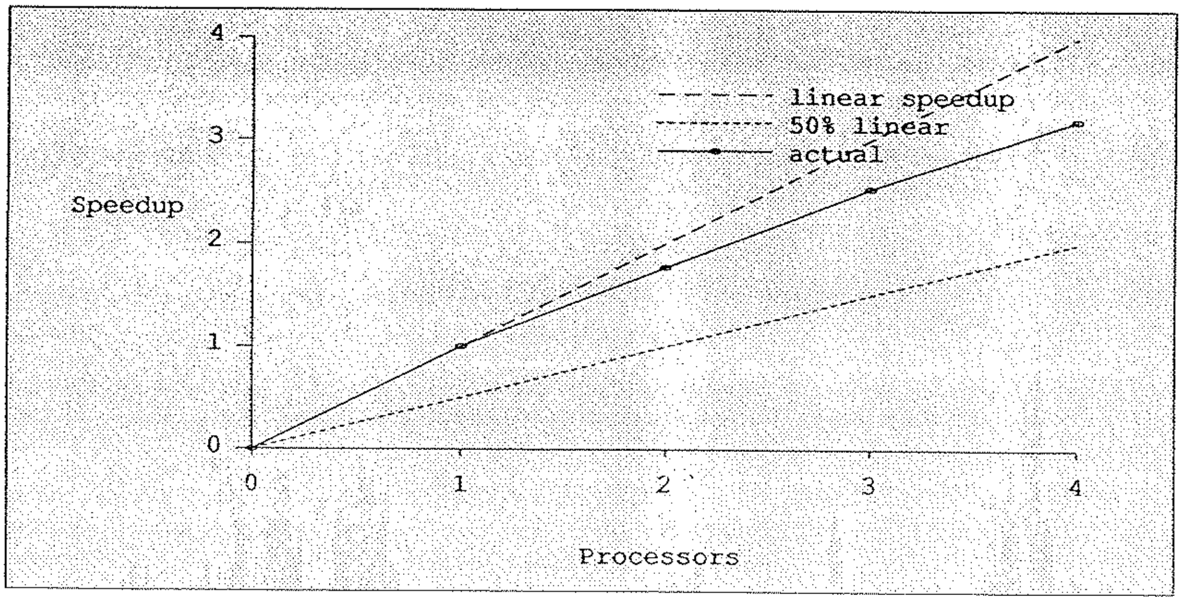


Figure 12. Speedup for with all refinements for a 3000 block system over 200 iterations

5. Conclusions

The study has shown that it is possible to obtain good speedup on current multiprocessors. These processors have relatively poor scientific performance but it is likely that next generation processors will be greatly improved in this respect.

Current FORTRAN annotators provide some promise of implicit parallel programming without resorting to explicit annotation although there appears to be some scope for improvement in the heuristics used by the epf annotator.

Acknowledgements

The authors thank the members of the Laboratory for Concurrent Computing Systems, at the Swinburne Institute of Technology, for their contributions to the work presented in this paper.

References

1. Cundall, P.A., A computer model for simulating progressive large-scale movements in blocky rock systems. Proc. ISRM Symp. on Rock Fracture, Nancy, vol. 1, paper II-8. 1971.

2. Cundall, P.A. et al., Computer modeling of jointed rock masses. Technical Report N-78-4, U.S. Army Engineer Waterways Experiment Station, Vicksburg, Miss., 1978.

3. Itasca. UDEC - Universal distinct element code, version ICG1.6; User's manual. Itasca Consulting Group, Inc., Minneapolis, 1990.

4. Itasca. 3DEC - 3-D distinct element code, version 1.2; User's manual. Itasca Consulting Group, Inc., Minneapolis, 1990.

5. Lemos, J.V., A hybrid distinct element - boundary element computational model for the half-plane. M.S. Thesis, Dept. of Civil and Mineral Engng., University of Minnesota, Minneapolis, 1983.

6. Choi, S.K. and M.A. Coulthard, Mechanics of jointed rock masses using the distinct element method. Int. Conf. on Mechanics of Jointed and Faulted Rock, Vienna, 1990, (in press).

7. Taylor, L.M., BLOCKS: A block motion code for geomechanics studies, Sandia National Laboratories, Albuquerque, Report SAND-82-2373, 1983.

APPENDIX A: Parallel FORTRAN code for Cycle and its subroutines

```
C
C     EPF/MULTIMAX        6.12.00 (A1_BASE)   o3,r3,d3 24 Sep 1990 11:40:33
      SUBROUTINE CYCLE
C
C------------------- DRIVER FOR ITERATIONS ---
C
      COMMON A(25000)
      DIMENSION IA(1)
      EQUIVALENCE (A,IA)
      PARAMETER(MX=200,MXNC=MX*4)
      COMMON/GEOM/XB(MXNC),YB(MXNC),NODDEF(MXNC),KFLG(MX),NODES(MXNC)
     .           ,XD(MXNC),YD(MXNC)
      LOGICAL DELETED(MX)
      EQUIVALENCE(DELETED(1),KFLG(1))
      PARAMETER(LBLOCK=162)
C     LBLOCK IS THE LENGTH OF CBLOCK
      COMMON /CBLOCK/ JNK(20),NBLOKM,NBOXES,M1,M2,M3,M4,M5,M6,M7,
     1                NBLOKS,NCYC,MCYCLE,NEMPT,RFLAG,UFLAG,EFLAG,TFRAC,
     2                TDEL,IBOXES,JBOXES,XL,XU,YL,YU,BSIZE,SFACT,XSIZE,
     3                YSIZE,UDMAX,UMOST,STIFN,STIFS,FRIC,BETA,BDT,ALPHB,
     4                CON1,CON2,ALPHA,NVARB,NBR,IBR,NPR,LAME1,LAME2,G2,
     5                G,CON1B,CON2B,GRAVX,GRAVY,LOC1,LOC2,NUPDAT,YIELD,
     6                LOCK,NDEL,NDEMAX,NBDEL(60),IPCNT,NPOIN,LZ,DEBUG
     7                ,NAPF,NBOUC,NBOUN,M9,M10,M11,M12,M13,TRIANGLE
     8                ,M8,NFRIC,FRIC0,ICOUP,SFX,SFY,ICORE,NEQ,INP,IOU
     9                ,TOLL
      COMMON/ADC/AXDC(105),AYDC(105)
      COMMON/RB/RBFLAG(105,4)
      LOGICAL RFLAG,UFLAG,EFLAG,LOCK,LZ,DEBUG,RBFLAG(105,4)
      INTEGER TRIANGLE
      REAL LAME1,LAME2
      COMMON /FOLLOW/ JFOLL,JCYC,JBLK(4),JCOR(4),JVAR(4),DXY(4)
      DIMENSION LAB(2)
      CHARACTER*2 LAB
C
C
      PARAMETER(NX=80,NBLK=50,NX2=NX*2,NX2B=NX2*NBLK,NX2P=NX2+NBLK)
      PARAMETER (LFREE=NX2*NX2/4 + NX2 + 3 + NX2*NBLK + 4*NX2P)
      INTEGER II1, FRIC1
C
C     NX = MAXIMUM NUMBER OF BOUNDARY ELEMENTS/NODES
C
C#     DIMENSION
C#   .JW(5),AW0(5),DW0(5),NW5(5),NWR5(5,8),WR5(5,8),AK(NX2)
      DATA LAB /'DX', 'DY'/
      EVENT II5(20)
      PARALLEL
      INTEGER I
C
      PRIVATE I
      DOALL (I=1:NBLOKS)
      DELETED(I)=.FALSE.
    5 CONTINUE
      END DOALL
      END PARALLEL
C
      PARALLEL
      INTEGER I,J
      PRIVATE I,J
      DOALL (I=1:NBLOKS)
      DO 2 J=1,4
    2 RBFLAG(I,J)=.FALSE.
      END DOALL
        END PARALLEL
C
      DO 6 I=1,NDEL
      DELETED(NBDEL(I))=.TRUE.
    6 CONTINUE

C
      DO 100 NCYCLE=1,NCYC
      MCYCLE=MCYCLE+1
C-------------------- UPDATE IF NECESSARY ---
C       IF((.NOT.UFLAG).AND.MCYCLE.EQ.IUPDAT) CALL UPDAT
      IF(UFLAG) CALL UPDAT
      IF(NCYCLE.EQ.1) CALL INPUT2
      NAK= NEQ
C-------------------- SCAN ALL BLOCKS ---
      UDMAX=0.0
      PARALLEL
      INTEGER NBR, JFIX, NC, LIB
      PRIVATE NBR, JFIX, NC, LIB,S22C,S12C
      DOALL (NBR=1:NBLOKS)
C
C-------------------- CHECK IF BLOCK DELETED ---
C
      IF (.NOT.DELETED(NBR)) THEN
      LIB=IA(NBR)
      JFIX = IA(LIB)
      NC = IA(LIB+1)
      CALL MOTION(A(LIB),JFIX,NC,NBR)
C      CALL STRESS(A(LIB))
C------------------- STRESS ROTATION CORRECTION TERMS ---
        S22C=(A(LIB+17)+A(LIB+18))*A(LIB+6)
        S12C=(A(LIB+16)-A(LIB+19))*A(LIB+6)
C------------------- NEW STRESSES ------------------------
      A(LIB+16)=A(LIB+16)+(A(LIB+12)*LAME1+A(LIB+15)*LAME2-S22C)*TDEL
        A(LIB+17)=A(LIB+17)+(A(LIB+13)*G2+S12C)*TDEL
        A(LIB+18)=A(LIB+18)+(A(LIB+14)*G2+S12C)*TDEL
        A(LIB+19)=A(LIB+19)+(A(LIB+15)*LAME1+A(LIB+12)*LAME2+S22C)*TDEL
C
      END IF
   20 CONTINUE
      END DOALL
      END PARALLEL
C
C
      DO 22 NBR=1,NBLOKS
       DO 24 NPR=1,NC
   24    IF (RBFLAG(NB,NPR)) CALL REBOX
   22 CONTINUE
C
      DO 28 NB=1,NBLOKS
   28     UDMAX=AMAX1(UDMAX,ABS(AXDC(NB)),ABS(AYDC(NB)))
C-------------------- EXIT IF NOTHING MOVED ---
      IF(UDMAX.EQ.0.0.AND.MCYCLE.GT.1) GOTO 110
```

```
C------------------- UPDATE CONTACTS ? ---
      UMOST=UMOST+UDMAX*TDEL
      IF (UMOST .GE. 1.0) UFLAG = .TRUE.
C------------------- SCAN ALL CONTACTS ---
      PARALLEL
      INTEGER NB,IB,IBE,I6,J6,JBC,IBC,IBC1,IBE1,II
      PRIVATE NB, IB,IBE,I6,J6,JBC,IBC,IBC1,IBE1,FRIC,II
     1        FRICF,FXC,FYC,XCC,YCC,XCE,YCE,DS,DN,XD,YD,DUS,DUN,DFN,DFS

C
      DOALL (NB=1:NBLOKS)
C
C------------------- CHECK IF BLOCK DELETED ---
C
      IF (.NOT.DELETED(NB)) THEN
      IBE=IA(NB)
      J6=M5+NB-1
   40 I6=IA(J6)
      IF (I6 .NE. 0) THEN
      JBC=IA(I6+3)
      IBC=IA(JBC)
      IF(NFRIC.EQ.0) GOTO 401
      II= M8
      DO 400 I= 1, NFRIC
      IF(IA(II).NE.NB) GOTO 400
      IF(IA(II+1).NE.IA(I6+1)) GOTO 400
      FRIC= A(II+2)
 400  II= II+3
 401  CONTINUE
      IBC1 = IA(IBC)
      IBE1 = IA(IBE)
      CALL FORD(A(I6),A(IBC),A(IBE),IBC1,IBE1,NB,JBC)
      FRIC= FRIC0
      J6=I6+4
      GOTO 40
      END IF
      END IF
   50 CONTINUE
      END DOALL
      BARRIER
      END PARALLEL
C
C------------------- END CYCLE LOOP ---
C
      CALL FORBOU(AK,NAK)
C
      IF (JFOLL .NE. 0) THEN
      IF ((MCYCLE / JCYC) * JCYC .EQ. MCYCLE) THEN
      PARALLEL
      INTEGER J
C
      PRIVATE J
      DOALL (J=1:JFOLL)
      DXY(J) = A(IA(JBLK(J))+NVARB+3*IA(IA(JBLK(J))+1)+2*JCOR(J)+JVAR(J)
     X  -3)
 9990 CONTINUE
      END DOALL
      END PARALLEL
      J = MAX (JFOLL, 0) + 1
```

```
C
      WRITE(IOU,9998) MCYCLE,(LAB(JVAR(J)),JBLK(J),JCOR(J),DXY(J),
     1                              J=1,JFOLL)
      END IF
      END IF
 9998 FORMAT(I5,5X,1P,4(A2,'(',I3,',',I1,')=',E10.3,5X))
 9995 CONTINUE
C
  100 CONTINUE
C
  110 CONTINUE
      IF(ICOUP.EQ.1.AND.ICORE.EQ.0) CLOSE(11)
C
      RETURN
C
      END


C     EPF/MULTIMAX        6.12.00 (A1_BASE)   o3,r3,d3 24 Sep 1990 11:41:26
      SUBROUTINE MOTION(B,JFIX,NC,NB)
C
C------------------- LAW OF MOTION FOR A SINGLE BLOCK ---
C
      PARAMETER(LBLOCK=162)
C     LBLOCK IS THE LENGTH OF CBLOCK
      COMMON /CBLOCK/ JNK(20),NBLOKM,NBOXES,M1,M2,M3,M4,M5,M6,M7,
     1                NBLOKS,NCYC,MCYCLE,NEMPT,RFLAG,UFLAG,EFLAG,TFRAC,
     2                TDEL,IBOXES,JBOXES,XL,XU,YL,YU,BSIZE,SFACT,XSIZE,
     3                YSIZE,UDMAX,UMOST,STIFN,STIFS,FRIC,BETA,BDT,ALPHB,
     4                CON1,CON2,ALPHA,NVARB,NBR,IBR,NPR,LAME1,LAME2,G2,
     5                G,CON1B,CON2B,GRAVX,GRAVY,LOC1,LOC2,NUPDAT,YIELD,
     6                LOCK,NDEL,NDEMAX,NBDEL(60),IPCNT,NPOIN,LZ,DEBUG
     7                ,NAPF,NBOUC,NBOUN,M9,M10,M11,M12,M13,TRIANGLE
     8                ,M8,NFRIC,FRIC0,ICOUP,SFX,SFY,ICORE,NEQ,INP,IOU
     9                ,TOLL
      COMMON/ADC/AXDC(105), AYDC(105)
      COMMON/RB/RBFLAG(105,4)
      LOGICAL RFLAG,UFLAG,EFLAG,LOCK,LZ,DEBUG,RBFLAG(105,4)
      INTEGER TRIANGLE,NB
      REAL LAME1,LAME2
      DIMENSION B(1)
C     EQUIVALENCE (FIX,JFIX),(ANC,NC)
C     PARALLEL
C     PRIVATE B, XMIN, YMIN, YMAX, AREA, K1, K2, NP, X, Y, XW, YW, AM,
C    1          EM11, EM12, EM22, IC, IK, NPR, IC1, XARM, YARM, AXDC,
C    2          AYDC, IBX, IBY
C
C
C     FIX=B(1)
C     ANC = B(2)
C     JFIX = FIX
C------------------- IS THIS BLOCK FIXED ? ---
      IF(JFIX.NE.0) RETURN
C------------------- NO ---
C------------------- VELOCITIES FROM ACCELERATIONS ---
      B(5)=(B(5)*CON1+(B(10)/B(8)+GRAVX)*TDEL)*CON2
      B(6)=(B(6)*CON1+(B(11)/B(8)+GRAVY)*TDEL)*CON2
      B(7)=(B(7)*CON1+(B(12)/B(9))*TDEL)*CON2
      B(10)=0.0
      B(11)=0.0
      B(12)=0.0
C
C     ANC=B(2)
C
C-------------------- DETERMINE AREA & WIDTH ---
      XMIN=XSIZE
      XMAX=0.0
      YMIN=YSIZE
      YMAX=0.0
      AREA=0.0
      K1=NVARB+1
      DO 50 NP=1,NC
      K2=K1+3
      IF(NP.EQ.NC) K2=NVARB+1
      X=B(K1)
      Y=B(K1+1)
      XMIN=AMIN1(XMIN,X)
      XMAX=AMAX1(XMAX,X)
      YMIN=AMIN1(YMIN,Y)
      YMAX=AMAX1(YMAX,Y)
      AREA=AREA+(B(K2)-X)*(B(K2+1)+Y)
   50 K1=K2
      AREA=AREA*2.0
      XW=XMAX-XMIN
      YW=YMAX-YMIN
      AM=TDEL/B(8)
C-------------------- EFFECTIVE MASSES ---
      EM11=AM/(XW*XW)
      EM12=AM/(YW*YW)
      EM21=EM11
      EM22=EM12
C-------------------- NEW STRAIN RATES ---
      B(13)=(B(13)*CON1B+(4.0*B(21)-B(17)*AREA)*EM11)*CON2B
      B(14)=(B(14)*CON1B+(4.0*B(22)-B(18)*AREA)*EM12)*CON2B
      B(15)=(B(15)*CON1B+(4.0*B(23)-B(19)*AREA)*EM21)*CON2B
      B(16)=(B(16)*CON1B+(4.0*B(24)-B(20)*AREA)*EM22)*CON2B
C-------------------- UPDATE BLOCK CORNERS ---
      IC=NVARB+1
      IK= IC+3*NC
      DO 150 NPR=1,NC
      IC1=IC+1
      XARM=B(IC)-B(3)
      YARM=B(IC1)-B(4)
      AXDC(NB)=B(5)+B(13)*XARM+(B(14)-B(7))*YARM
      AYDC(NB)=B(6)+B(16)*YARM+(B(15)+B(7))*XARM
      IBX=B(IC)
      IBY=B(IC1)
      B(IC)=B(IC)+AXDC(NB)*TDEL
      B(IC1)=B(IC1)+AYDC(NB)*TDEL
      IF(IBX.NE.IFIX(B(IC)).OR.IBY.NE.IFIX(B(IC1))) THEN
       RBFLAG(NB,NPR)=.TRUE.
      END IF
      B(IK)= B(IK)+AXDC(NB)*TDEL
      B(IK+1)= B(IK+1)+AYDC(NB)*TDEL
      IK= IK+2
  150 IC=IC+3
C-------------------- RIGID BODY DISPLACEMENTS ---
      B(3)=B(3)+B(5)*TDEL
      B(4)=B(4)+B(6)*TDEL
C-----------------------------------------------------------
      B(21)=0.0
      B(22)=0.0
      B(23)=0.0
      B(24)=0.0
C     END PARALLEL
      RETURN
C
      END


C     EPF/MULTIMAX       6.12.00 (A1_BASE)   o3,r3,d3 24 Sep 1990 11:42:05
      SUBROUTINE UPDAT
C
C------------------- UPDATE ALL CONTACTS ---
C
      COMMON A(25000)
      DIMENSION IA(1)
      EQUIVALENCE (A,IA)
      PARAMETER(LBLOCK=162)
C     LBLOCK IS THE LENGTH OF CBLOCK
      COMMON /CBLOCK/ JNK(20),NBLOKM,NBOXES,M1,M2,M3,M4,M5,M6,M7,
     1                NBLOKS,NCYC,MCYCLE,NEMPT,RFLAG,UFLAG,EFLAG,TFRAC,
     2                TDEL,IBOXES,JBOXES,XL,XU,YL,YU,BSIZE,SFACT,XSIZE,
     3                YSIZE,UDMAX,UMOST,STIFN,STIFS,FRIC,BETA,BDT,ALPHB,
     4                CON1,CON2,ALPHA,NVARB,NBR,IBR,NPR,LAME1,LAME2,G2,
     5                G,CON1B,CON2B,GRAVX,GRAVY,LOC1,LOC2,NUPDAT,YIELD,
     6                LOCK,NDEL,NDEMAX,NBDEL(60),IPCNT,NPOIN,LZ,DEBUG
     7                ,NAPF,NBOUC,NBOUN,M9,M10,M11,M12,M13,TRIANGLE
     8                ,M8,NFRIC,FRIC0,ICOUP,SFX,SFY,ICORE,NEQ,INP,IOU
     9                ,TOLL
      LOGICAL RFLAG,UFLAG,EFLAG,LOCK,LZ,DEBUG
      INTEGER TRIANGLE
      REAL LAME1,LAME2
C      LOGICAL FIRST
      logical LOK
      DATA TOL /1.0/, TOLB/0.1/
      DEBUG=.FALSE.
C
C------------------- SCAN EACH BLOCK ---
C
      LOK=.false.
C
      PARALLEL
      INTEGER NB,KKK,I2,NC,IEND2,IEND1,NP,NXL,NXU,NYL,NYU,JBOX,NBOX
      INTEGER   IBOX,I4,I2C,IC,J6,I6,ICR,ICL,JJ
C      REAL COSA,SINA,X1,X2,Y1,Y2,XDIF,YDIF,Z,XX1,XX2,YY1,YY2,YT,XT,YTR,YTL
      LOGICAL FIRST
      PRIVATE NB,KKK,FIRST,I2,NC,IEND1,IEND2,NP,X1,X2,Y1,Y2,XDIF,YDIF,Z
      PRIVATE   XX1,XX2,YY1,YY2,NXL,NXU,NYL,NYU,JBOX,NBOX,IBOX,I4,COSA
        PRIVATE SINA,I2C,IC,YT,XT,J6,I6,ICR,YTR,ICL,YTL,JJ
      DOALL (NB=1:NBLOKS)
C      DO 500 NB=1,NBLOKS
C
C------------------- CHECK IF BLOCK DELETED ---
C
      IF (NDEL .NE. 0) THEN
      DO 21 KKK=1,NDEL
      IF(NB.EQ.NBDEL(KKK)) GO TO 405
   21 CONTINUE
      END IF
   22 CONTINUE
      I2=IA(NB)
      NC=IA(I2+1)
C-------------------- SCAN EACH EDGE OF BLOCK ---
      IEND2=I2+NVARB
      DO 400 NP=1,NC
      FIRST=.TRUE.
      IEND1=IEND2
      IEND2=IEND1+3
      IF(NP.EQ.NC) IEND2=I2+NVARB
C-------------------- DETERMINE BOXES TO BE SEARCHED ---
      X1=A(IEND1)
      X2=A(IEND2)
      Y1=A(IEND1+1)
      Y2=A(IEND2+1)
      XDIF=X2-X1
      YDIF=Y2-Y1
      Z=SQRT(XDIF*XDIF+YDIF*YDIF)
      A(IEND1+2)=Z
      XX1=AMIN1(X1,X2)
      XX2=AMAX1(X1,X2)
      YY1=AMIN1(Y1,Y2)
      YY2=AMAX1(Y1,Y2)
      NXL=IFIX((XX1-TOL)/BSIZE)+1
      NXU=MIN0(IFIX((XX2+TOL)/BSIZE)+1,IBOXES)
      NYL=IFIX((YY1-TOL)/BSIZE)+1
      NYU=MIN0(IFIX((YY2+TOL)/BSIZE)+1,JBOXES)
C-------------------- SEARCH EACH CANDIDATE BOX ---
      DO 350 JBOX=NYL,NYU
      NBOX=(JBOX-1)*IBOXES+NXL-1
      DO 300 IBOX=NXL,NXU
      NBOX=NBOX+1
      I4=IA(M3+NBOX-1)
C-------------------- END OF BOX LIST ? ---
  210 CONTINUE
      IF (I4 .NE. 0) THEN
C
C-------------------- CHECK IF BLOCK DELETED ---
C
      IF (NDEL .NE. 0) THEN
      DO 31 KKK=1,NDEL
      IF(IA(I4+1).EQ.NBDEL(KKK)) GO TO 215
   31 CONTINUE
      END IF
   32 CONTINUE
C-------------------- NO, IS THIS BLOCK NB ? ---
      IF(IA(I4+1).NE.NB) GOTO 220
C-------------------- YES ---
C-------------------- GET NEXT ENTRY IN THIS BOX ---
  215 I4=IA(I4+2)
      GOTO 210
C-------------------- FIRST TIME THROUGH ? ---
  220 IF (FIRST) THEN
C-------------------- YES, COMPUTE COS, SIN  FOR THIS EDGE ---
      FIRST=.FALSE.
      COSA=XDIF/Z
      SINA=YDIF/Z
      END IF
C-------------------- COMPUTE CORNER COORDINATES RELATIVE TO EDGE ---
  230 I2C=IA(I4+1)
      IF(DEBUG)WRITE(*,*)MCYCLE,NB,NBOX,I4,I2C,IA(I2C),IA(I4)
      I2C=IA(I2C)
      IC=I2C+IA(I4)*3+NVARB-3
      IF(DEBUG)WRITE(*,*)'IC',IC
      YT=(A(IC+1)-Y1)*COSA
     .  -(A(IC)   -X1)*SINA
      IF(DEBUG)WRITE(*,*)' YT',YT
      IF(YT.GT.1.0) GOTO 215
      IF(YT.LE.-3.0) GOTO 215
      XT=(A(IC)   -X1)*COSA
     .  +(A(IC+1)-Y1)*SINA
      IF(DEBUG) WRITE(*,*)' XT',XT
      IF(XT.GT.Z+TOLB) GOTO 215
      IF(XT.LT.-TOLB) GOTO 215
C-------------------- CONTACT LIST FOR BLOCK NB ---
      J6=M5+NB-1
      I6=IA(J6)
C-------------------- END OF LIST ? ---
  235 CONTINUE
      IF(I6.EQ.0) GOTO 250
C-------------------- NO. SAME CORNER AND EDGE ? ---
      IF (IA(I4) .NE. IA(I6+2) .OR. IA(I4+1) .NE. IA(I6+3) .OR. NP .NE.
     X    IA(I6+1)) THEN
C-------------------- NO. GET NEXT ENTRY IN CONTACT LIST ---
      IF(DEBUG) WRITE(*,*)' GET NEXT ENTRY'
C      WRITE(*,*)' GET NEXT ENTRY'
      J6=I6+4
      I6=IA(J6)
      GOTO 235
      END IF
C-------------------- CONTACT ALREADY STORED ---
  240 IF (YT .LE. -2.0) WRITE (IOU, 3000)
C-------------------- UPDATE CONTACT DATA ---
  245 A(I6+9)=SINA
      A(I6+10)=COSA
      A(I6+11)=A(IC)
      A(I6+12)=A(IC+1)
      IA(I6)=1
      IF(DEBUG) WRITE(*,*)' UPDATE CONTACT DATA'
C       WRITE(*,*)' UPDATE CONTACT DATA'
      GOTO 215
C-------------------- NEW CONTACT ? ---
  250 CONTINUE
      IF(DEBUG) WRITE(*,*)' GO TO 250'
C       WRITE(*,*)' GO TO 250'
      IF(YT.GT.0.1) GOTO 215
C-------------------- YES ---
      IF(DEBUG) WRITE(*,*)' NEW CONTACT'
C      WRITE(*,*)' NEW CONTACT'
      ICR=IC-3
      IF(IA(I4).EQ.1) ICR=IC+3*(IA(I2C+1)-1)
      YTR=(A(ICR+1)-Y1)*COSA
     .   -(A(ICR)   -X1)*SINA
      IF(DEBUG) WRITE(*,*)' YTR',YTR
      IF(YTR.LE.-2.0) GOTO 215
      ICL=IC+3
      IF(IA(I4).EQ.IA(I2C+1)) ICL=I2C+NVARB
      YTL=(A(ICL+1)-Y1)*COSA
     .   -(A(ICL)   -X1)*SINA
      IF(DEBUG) WRITE(*,*)' YTL',YTL
      IF(YTL.LE.-2.0) GOTO 215
      IF(XT.LT.2.0.AND.YTR.GT.2.0) GOTO 215
      IF(XT.GT.Z-2.0.AND.YTL.GT.2.0) GOTO 215
      IF(YTR.GT.2.0.AND.YTL.GT.2.0) GOTO 215
      JJ=IA(M5+IA(I4+1)-1)
 2600 IF (JJ .NE. 0) THEN
      IF (IA(JJ+3) .EQ. NB) THEN
      IF(ABS(A(IC)-A(JJ+11)).LE.1.0.AND.
     1   ABS(A(IC+1)-A(JJ+12)).LE.1.0) GOTO 215
      END IF
 2601 JJ=IA(JJ+4)
      GOTO 2600
      END IF
 2602 CONTINUE
C------------------- ANY SPACE AVAILABLE IN EMPTY LIST ? ---
```

call spinlock(lok)

```
      I6=NEMPT
      IF (NEMPT + 14 .GE. M8) THEN
      WRITE(IOU,3001)
      WRITE(IOU,3002) M8,NEMPT
      CALL FINISH
      END IF
C------------------- NEW CONTACT ---
  260 A(I6+5)=0.0
      A(I6+6)=0.0
      A(I6+7)=0.0
      A(I6+8)=0.0
      IA(I6+1)=NP
      IA(I6+2)=IA(I4)
      IA(I6+3)=IA(I4+1)
        NEMPT=IA(I6+4)
      IA(J6)=I6
      IA(I6+4)=0
```

call spinunlock(lok)

```
      GOTO 245
      END IF
C------------------- END OF BOX SCAN ---
  300 CONTINUE
  350 CONTINUE
C------------------- END OF EDGE SCAN ---
  400 CONTINUE
  405 CONTINUE
  490 CONTINUE
      IF(DEBUG)DEBUG=  .FALSE.
  500 CONTINUE
      END DOALL
 3000 FORMAT(30X,' DRIFT CORRECTION REQUIRED')
 3001 FORMAT(30X,' NO MORE MEMORY FOR CONTACT LIST')
 3002 FORMAT(30X,'    - NEED TO INCREASE M8 FROM',I7,/
     1       30X,'                    TO AT LEAST',I7,' + 14')
```

```
      END PARALLEL
C-------------------- END OF BLOCK SCAN ---
C
C-------------------- SCAN CONTACT LIST FOR PRESERVE FLAGS ---
      DO 499 NBB=1,NBLOKS
      J66 =M5+NBB-1
      I66=IA(J66)
C-------------------- END OF LIST ? ---
  410 IF (I66 .NE. 0) THEN
C-------------------- NO ---
C-------------------- IS THE PRESERVE FLAG SET ? ---
      IF(IA(I66).EQ.0) GOTO 420
C-------------------- YES ---
      IF(DEBUG) WRITE(*,*)' PRESERVE FLAG SET'
C      WRITE(*,*)' PRESERVE FLAG SET'
      IA(I66)=0
  415 J66=I66+4
      GOTO 430
C-------------------- NO ---
  420 IF(LOCK) GOTO 415
      IA(J66)=IA(I66+4)
      IA(I66+4)=NEMPT
      NEMPT=I66
      IF(DEBUG) WRITE(*,*)' PRESERVE FLAG NOT SET'
C      WRITE(*,*)' PRESERVE FLAG NOT SET'
C-------------------- GET NEXT CONTACT ---
  430 I66=IA(J66)
      GOTO 410
      END IF
  499 CONTINUE
C
      NUPDAT=NUPDAT+1
      UFLAG=.FALSE.
      UMOST=0.0
      RETURN
C
C
      END

C     EPF/MULTIMAX        6.12.00 (A1_BASE)   o3,r3,d3 24 Sep 1990 11:40:33
      SUBROUTINE FORD(C,BC,BE,IBC1,IBE1,NB,NBC)
C
C-------------------- FORCE DISPLACEMENT LAW FOR SINGLE CONTACT ---
C
      PARAMETER(LBLOCK=162)
C     LBLOCK IS THE LENGTH OF CBLOCK
      COMMON /CBLOCK/ JNK(20),NBLOKM,NBOXES,M1,M2,M3,M4,M5,M6,M7,
     1                NBLOKS,NCYC,MCYCLE,NEMPT,RFLAG,UFLAG,EFLAG,TFRAC,
     2                TDEL,IBOXES,JBOXES,XL,XU,YL,YU,BSIZE,SFACT,XSIZE,
     3                YSIZE,UDMAX,UMOST,STIFN,STIFS,FRIC,BETA,BDT,ALPHB,
     4                CON1,CON2,ALPHA,NVARB,NBR,IBR,NPR,LAME1,LAME2,G2,
     5                G,CON1B,CON2B,GRAVX,GRAVY,LOC1,LOC2,NUPDAT,YIELD,
     6                LOCK,NDEL,NDEMAX,NBDEL(60),IPCNT,NPOIN,LZ,DEBUG
     7                ,NAPF,NBOUC,NBOUN,M9,M10,M11,M12,M13,TRIANGLE
     8                ,M8,NFRIC,FRIC0,ICOUP,SFX,SFY,ICORE,NEQ,INP,IOU
     9                ,TOLL
      COMMON /LOK/lok_nb(105)
C      LOCK lok_nb(105)
      LOGICAL RFLAG,UFLAG,EFLAG,LOCK,LZ,DEBUG
      INTEGER TRIANGLE
      REAL LAME1,LAME2
        logical lok_nb(105)
      DIMENSION C(1),BC(1),BE(1)
C
C
C      EQUIVALENCE (BC1,IBC1), (BE1,IBE1)
C
C-------------------- RELATIVE X & Y VELOCITIES ACROSS CONTACT ---
      XCC=C(12)-BC(3)
      YCC=C(13)-BC(4)
      XCE=C(12)-BE(3)
      YCE=C(13)-BE(4)
C     BC1= BC(1)
C     BE1= BE(1)
      IF (IBC1 .GE. 4 .OR. IBE1 .GE. 4) THEN
      DS= 0.0
      DN= 0.0
      ELSE
 90   CONTINUE
      XD=BC(13)*XCC+(BC(14)-BC(7))*YCC+BC(5)
     .  -BE(13)*XCE-(BE(14)-BE(7))*YCE-BE(5)
      YD=BC(16)*YCC+(BC(15)+BC(7))*XCC+BC(6)
     .  -BE(16)*YCE-(BE(15)+BE(7))*XCE-BE(6)
C-------------------- SHEAR & NORMAL DISPLACEMENT INCREMENTS ---
      DUS=(XD*C(11)+YD*C(10))*TDEL
      DUN=(YD*C(11)-XD*C(10))*TDEL
C-------------------- NORMAL FORCE ---
      DFN=-DUN*STIFN
C      WAIT LOCK (lok_nb(nb)) (lok_nb(nbc)))
C      WAIT LOCK (lok_nb(1))
      C(8)=C(8)+DFN
      IF(LOCK) GOTO 10
C-------------------- TEST FOR TENSION ---
      IF(C(8).GE.0.0) GOTO 15
      C(8)=0.0
      C(9)=0.0
      DN=0.0
      DS=0.0
      GOTO 60
C-------------------- LOCKED JOINT ---
   10 FRICF=ABS(FRIC*C(8))
      GOTO 20
C-------------------- SHEAR FORCE ---
   15 FRICF=FRIC*C(8)
   20 DFS=DUS*STIFS
      C(9)=C(9)+DFS
      IF (ABS (C(9)) .GT. FRICF) THEN
      C(9)=SIGN(FRICF,C(9))
      DS=0.0
      ELSE
C-------------------- DASHPOT FORCES ---
   40 DS=BDT*DFS
      END IF
   50 DN=BDT*DFN
      END IF
C-------------------- GLOBAL CONTACT FORCES ---
   60 FYC=(C(9)+DS)*C(10)-(C(8)+DN)*C(11)
      FXC=(C(9)+DS)*C(11)+(C(8)+DN)*C(10)
      FC12=FYC*XCC-FXC*YCC
      FC21=FXC*XCC
      FC22=FXC*YCC
      FC23=FYC*XCC
      FC24=FYC*YCC
      FE12=FYC*XCE-FXC*YCE
      FE21=FXC*XCE
      FE22=FXC*YCE
      FE23=FYC*XCE
      FE24=FYC*YCE
C------------------- ADD CONTRIBUTION TO BLOCK FORCES ---

  2     call spinlock(lok_nb(nb))
        if (cspinlock(lok_nb(nbc))) goto 1
        call spinunlock(lok_nb(nb))
        goto 2
  1     continue

      BC(10)=BC(10)-FXC
      BC(11)=BC(11)-FYC
      BC(12)=BC(12)-FC12
C------------------- APPLIED STRESS IN BLOCKS ---
      BC(21)=BC(21)-FC21
      BC(22)=BC(22)-FC22
      BC(23)=BC(23)-FC23
      BC(24)=BC(24)-FC24
C------------------- ADD CONTRIBUTION TO BLOCK FORCES ---
      BE(10)=BE(10)+FXC
      BE(11)=BE(11)+FYC
      BE(12)=BE(12)+FE12
C------------------- APPLIED STRESS IN BLOCKS ---
      BE(21)=BE(21)+FE21
      BE(22)=BE(22)+FE22
      BE(23)=BE(23)+FE23
      BE(24)=BE(24)+FE24

      call spinunlock(lok_nb(nbc))
      call spinunlock(lok_nb(nb))
C
      RETURN
C
      END
```