Laboratory for Concurrent Computing Systems
Technical Report 31-042

Version 1.0 April 1993

Submitted to the ASE 1993 Conference

Parallelisation of the SDEM Distint Element Stress Analysis Code

Professor G. K. Egan
Swinburne University of Technology
Australia

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 which make the task of obtaining performance gains through vectorisation difficult. The systems, however, are comprised of thousands of blocks and there is 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. Experience has shown that the gains due to automatic annotation of the original FORTRAN source were limited but that with a modest amount of effort critical data dependencies may be resolved. Satisfactory gains may then be obtained with additional manual annotation of the source. The paper details the procedures used and presents results for Cray Research multiprocessors. The results show that although gains due to vectorisation are limited, the gains due to the parallel implementation are quite satisfactory.

LABORATORY FOR CONCURRENT COMPUTING SYSTEMS
COMPUTER SYSTEMS ENGINEERING
School of Electrical Engineering
Swinburne University of Technology
John Street, Hawthorn 3122, Victoria, Australia.

Parallelisation of the SDEM Distinct Element Stress Analysis Code

G.K. Egan
Laboratory for Concurrent Computing Systems, Swinburne University of Technology, John Street, Hawthorn 3122, Australia

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 which make the task of obtaining performance gains through vectorisation difficult. The systems, however, are comprised of thousands of blocks and there is 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. Experience has shown that the gains due to automatic annotation of the original FORTRAN source were limited but that with a modest amount of effort critical data dependencies may be resolved. Satisfactory gains may then be obtained with additional manual annotation of the source. The paper details the procedures used and presents results for Cray Research multiprocessors. The results show that although gains due to vectorisation are limited, the gains due to the parallel implementation are quite satisfactory.

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 work stations 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 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 making the task of obtaining performance gains through vectorisation difficult. As the systems are comprised of thousands of blocks there is however the potential of performing block interaction calculations in parallel.

The paper describes the parallelisation of SDEM, a representative DE stress analysis code for the analysis of two dimensional systems of interacting, simply deformable polygonal DEs [2][5].

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 programs based on the DE method for routine engineering design of excavations in highly jointed rock is the very large computer execution time which is required for analyses involving substantial numbers of distinct elements.

Theoretical basis

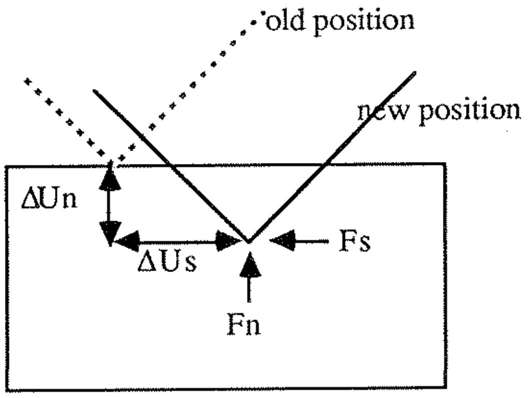
Most DE programs are based on force-displacement relations describing block interactions and Newton's second law of motion for the response of each block to the unbalanced forces and moments acting on it.

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 are 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.



$$\Delta Fn = \Delta Un \,.\, Kn \qquad (1)$$
$$\Delta Fs = \Delta Us \,.\, Ks \qquad (2)$$

where Kn = normal stiffness
ΔUn = incremental normal displacement

Ks = shear stiffness
ΔUs = incremental shear displacement

Figure 1. Block Interactions

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

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

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.

Block velocities and displacements are expressed explicitly in terms of values at the previous time step and so each may be calculated independently.

The calculated displacements are used to update the geometry of the system and to determine new block interaction forces. These, in turn, are used in the next time step.

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

SDEM

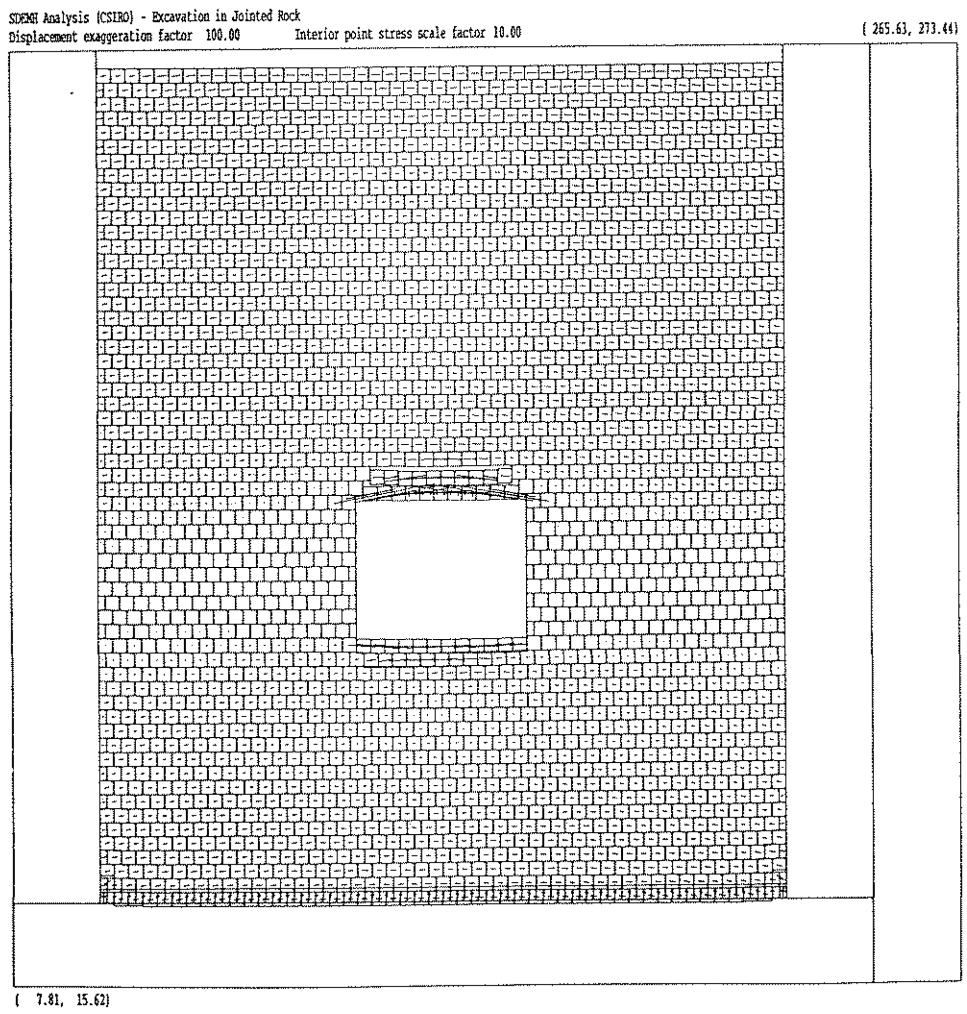
SDEM's main computational cycle is contained within the CYCLE subroutine (Figure 2): the new positions of blocks are computed using the current forces acting on them (MOTION); from these positions, the stresses (STRESS) and new forces induced by blocks on their neighbours (FORD) are determined; these 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.

```
while not stable do
    if update required then
        call updat
    do each block
        call motion{compute motion
                if current block velocity > max velocity then
                    max velocity = current block velocity
                do each block corner
                if corner outside box then
                    call rebox
                }
    do each block
        call stress
    do each block
        do each contact
            call ford
```

Figure 2. SDEM's main computational flow

For this study a system comprised of some 3000 blocks arranged in a "brick wall" is used. After a number of initial "settlement" cycles, approximately 200 blocks are removed and another large number of iterations, or sufficient iterations for all motion to stop, are performed.

Two systems were used for the study. They were the "entry level" Cray Research YMP-EL and the top of the line Cray Research C90. These systems share a common architecture allowing codes to be developed on the less expensive YMP-EL.


SDEMH Analysis (CSIRO) - Excavation in Jointed Rock
Displacement exaggeration factor 100.00
Interior point stress scale factor 10.00
(265.63, 273.44)
(7.81, 15.62)

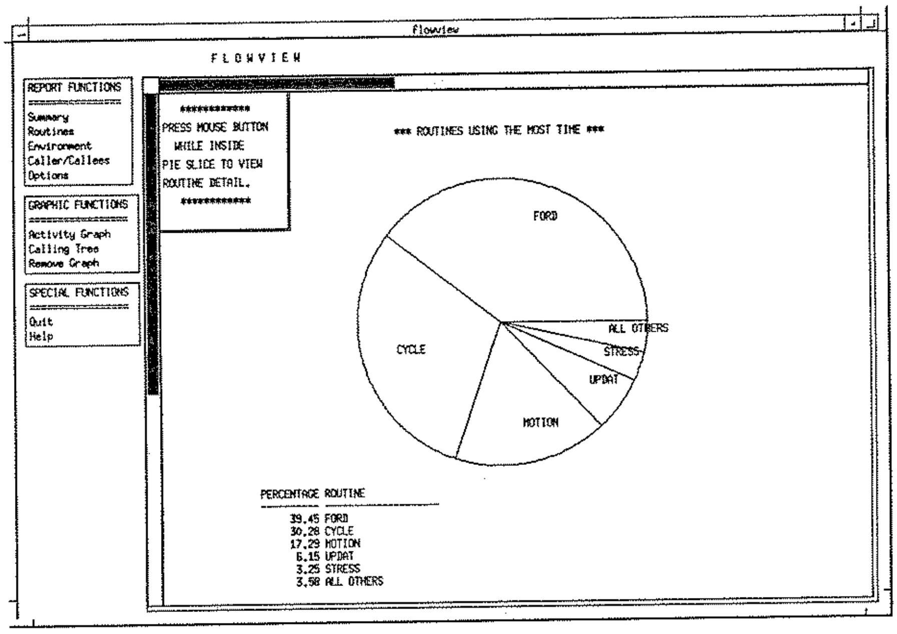
(3000 blocks 2000+50000 iterations)

Figure 3. Partially settled system of blocks showing stress vectors

VECTORISATION

The FORTRAN tool suite [8] on the Cray Research systems is extensive and is a marked advance on those available generally only a few years ago. The tool suite runs under X Windows.

The flowview tool identifies key subroutines and subroutines which are candidates for inlining; inlining avoids the significant overhead associated with subroutine calls and function invocations. flowview takes as its input profile files generated at run time by flowtrace. Figures 4 and 5 show representative windows from flowview.

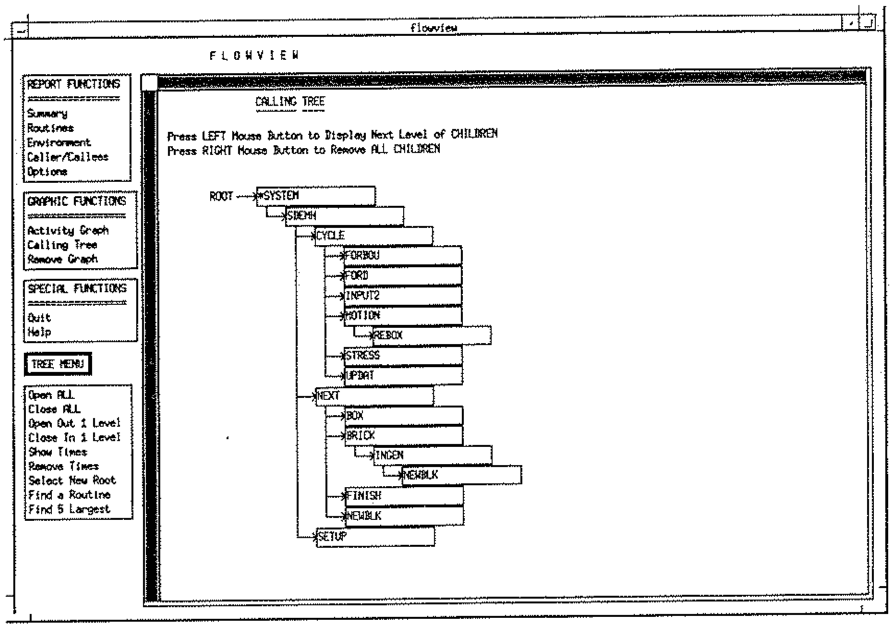


```
************
PRESS MOUSE BUTTON
  WHILE INSIDE
PIE SLICE TO VIEW
ROUTINE DETAIL.
************
```

```
*** ROUTINES USING THE MOST TIME ***
```

```
PERCENTAGE ROUTINE
    39.45 FORD
    30.28 CYCLE
    17.29 MOTION
     6.15 UPDAT
     3.25 STRESS
     3.58 ALL OTHERS
```

Figure 4. Key SDEM subroutines identified by flowview



CALLING TREE

```
Press LEFT Mouse Button to Display Next Level of CHILDREN
Press RIGHT Mouse Button to Remove ALL CHILDREN
```

Figure 5. SDEM subroutine call tree

The FORTRAN preprocessor (fpp) was used to inline subroutines identified by flowview. To inline these subroutines required:

- inserting COMMON statements used by inlined routines but not present in the enclosing subroutines;
- making the names used in the COMMON statements consistent across subroutine boundaries;
- replacing DATA statements with PARAMETER declarations or assignments.

Results for vectorisation

The times for non-vectorised and vectorised runs are presented in Table 1; it can be seen that there is no improvement. The conditional branching in the inner loops is too complicated for the optimisers; however, like many older codes it may be possible to restructure the loop bodies to reduce their complexity. This will be the subject of further work.

Time (Sec.)	System	Wall Clock
unvectorised cf77 -Zc -Wf"-I inlinesubs"	2567.887	2667.735
vectorised cf77 -Zv -Wf"-I inlinesubs"	2588.648	2659.164

(3000 blocks 4000 iterations Cray YMP-EL)
Table 1. Times for non-vectorised and vectorised runs

PARALLELISATION

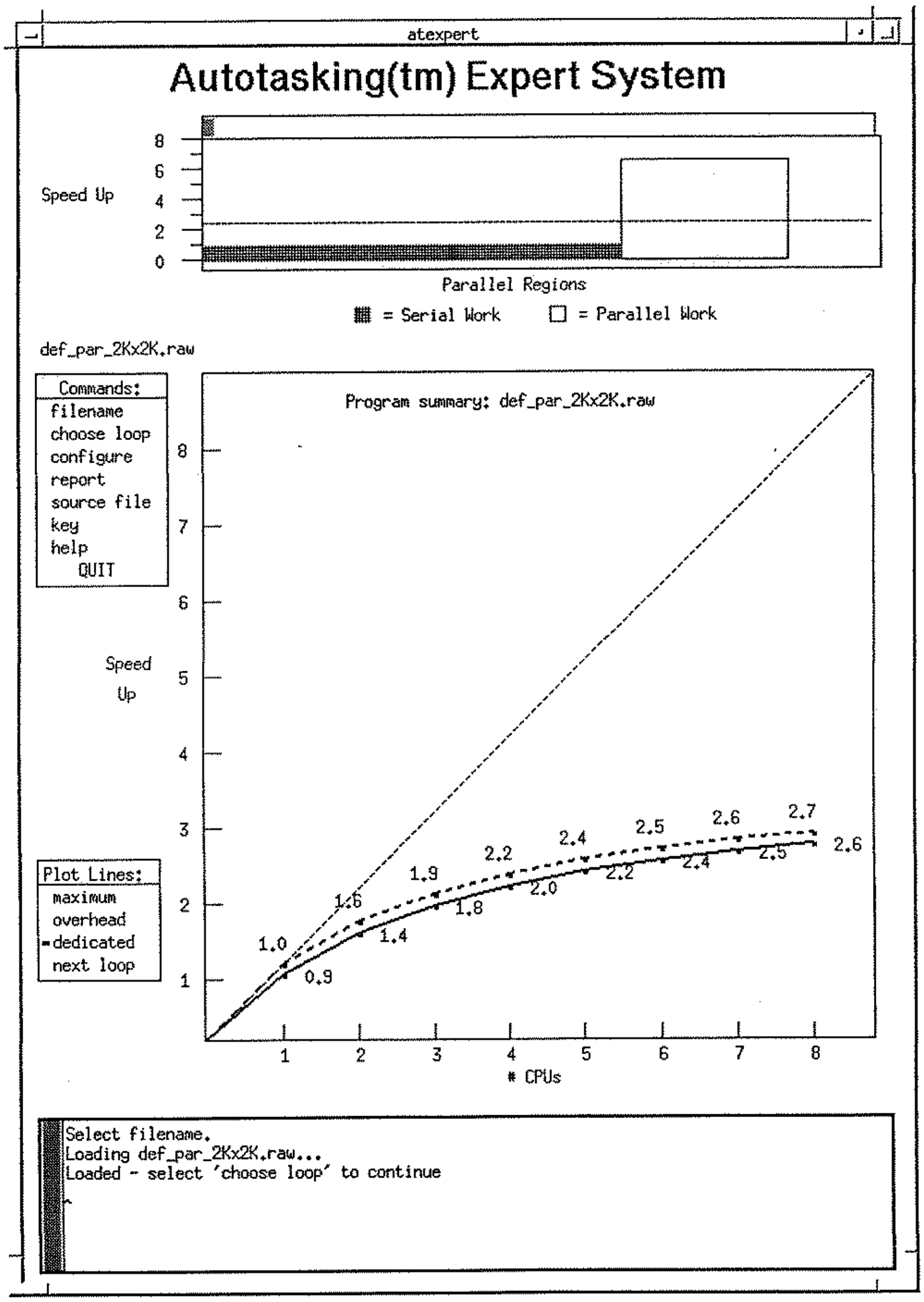
Automatic annotation

Subroutine CYCLE and the inlined subroutines it enclosed was analysed using fpp. fpp failed to discover significant parallel regions as it was unable to resolve the apparently complicated indexing used in SDEM's data structures and associated data dependencies spanning loop iterations.

Results for automatic parallelisation

The atexpert measurement tool was used to examine individual DO loops for predicted speedup. The tool accurately predicts performance for dedicated systems. The tool provides parallelism profiles and allows routines associated with parallel or sequential regions to be examined and analysed interactively.

Figure 6 shows the predicted speedup for the automatically annotated program.



```
Select filename.
Loading def_par_2Kx2K.raw...
Loaded - select 'choose loop' to continue
```

(3000 blocks 4000 iterations)
Figure 6. Predicted speedup of the automatically parallelised SDEM

Explicit annotation

The atscope annotator provides an X-windows based interactive tool for annotating FORTRAN DO loops (Figure 8). In most cases atscope can identify which variables are private to the loop and which are shared. The user is then invited to identify the scope of the remaining variables.

It was discovered that it was essential to inline subroutines prior to this analysis as the shared/private annotations are only checked prior to inlining by fpp. The shared/private status of variables after inlining appears to be assumed as shared.

The inlined code of the MOTION subroutine computes each block's new position and velocity due to the forces acting on it. It also tracks the velocity of the most rapidly moving block and calls the REBOX subroutine if a block's corner moves outside its bounding box. REBOX is called infrequently. Potentially there may be more than one processor updating the maximum velocity variable and the bounding box size and position at any one time.

The FORD subroutine computes the accumulated forces acting on a block due to all other contacting blocks. Potentially there may be more than one processor updating the variables in which the forces are being accumulated.

Major dislocation or collapse in the simulated system is indicated by the maximum block velocity tracked within MOTION; subroutine UPDAT is called to update the lists of contacting blocks resulting in release of list entries where blocks are no longer contacting and addition of new list entries where new contacts are formed. The variable "pointing" to the next empty list item may be potentially updated by more than one processor simultaneously.

Analysis of CYCLE, drawing in part on initial work described in [9], reveals that its major loop bodies may be made parallel by:

- tracking maximum block velocity in MOTION with a variable local to the loop chunk and then updating the global maximum velocity variable at the end of each loop chunk;
- recording whether a block needs to have its bounding box recomputed then recompute the new bounding boxes after the new position of all blocks has been determined by MOTION;
- locking the contact data structures for both interacting blocks in FORD;
- locking the free list pointer variable in UPDAT while creating new and discarding old contact records;

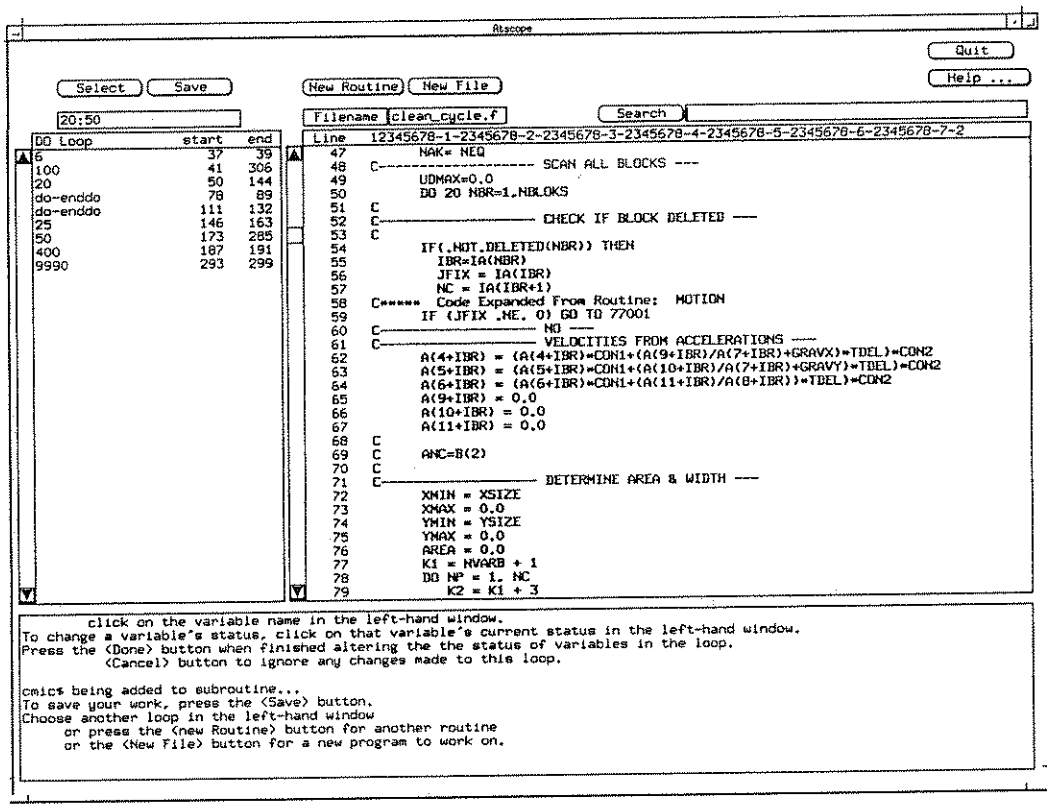
Locking means that no processor may execute a section of code, or change a data structure, guarded by a lock until it obtains the lock. Processors contend for a lock until they are successful in obtaining it. They then execute the section of code or change the data structure associated with the lock releasing the lock on completion. Some care is required when it is necessary to lock two block records as it is possible for deadlock to occur. For example block A requires Block B but another processor has Block B already and requires Block A to consider another interaction. In this example both processors should release the block they have and try again; if both try again immediately then the problem may re-occur however! The solution adopted here is to immediately release the first lock if the second lock cannot be obtained.

```
while not stable do

    if update required then
        do parallel all blocks
            do each edge
                do each surrounding box in both directions
                    if block is contacting then
                        guard free list variable
                            get new entry
                        release guard
                        store new contact data
        do parallel all blocks
            release old contact entries

    do parallel all blocks
        compute motion
        if current block velocity > local velocity then
            local velocity = current block velocity
        do each block corner
            if corner outside box then
                dorebox(block no)=true
        guard
            global max velocity = max( global velocity,local velocity)
        end guard
    do parallel all blocks
        if dorebox then
            rebox
    do parallel all blocks
        compute stress
    do parallel all blocks
        do each contact
            lock both_block records
                compute force between block and contact
            unlock
```

Figure 7. Manual annotation scheme for CYCLE

The additional locks (or guards) protecting the critical regions in Figure 7 were added manually.

The default annotation form of *do all* was replaced by the *do parallel* to permit the tracking of maximum block velocity as a reduction operation [7]. The optional *numchunks* directive specifies how many near equal size sub ranges the loop range is broken into for allocation as tasks to processors. The default number of chunks is equal to the loop range incurring parallel task start up costs for each loop cycle. For this study the number of chunks was chosen to be 32 or four times the maximum number of processors. If the number of chunks is set equal to the number of processors then one or more processors will get less work than the others particularly when the loop body is not executed due to a block or blocks being deleted. If the number of chunks is set too high then more loop start up cost is incurred, this being approximately constant for each loop chunk started.



20:50

DO Loop	start	end
6	37	39
100	41	306
20	50	144
do-enddo	78	89
do-enddo	111	132
25	146	163
50	173	285
400	187	191
9990	293	299

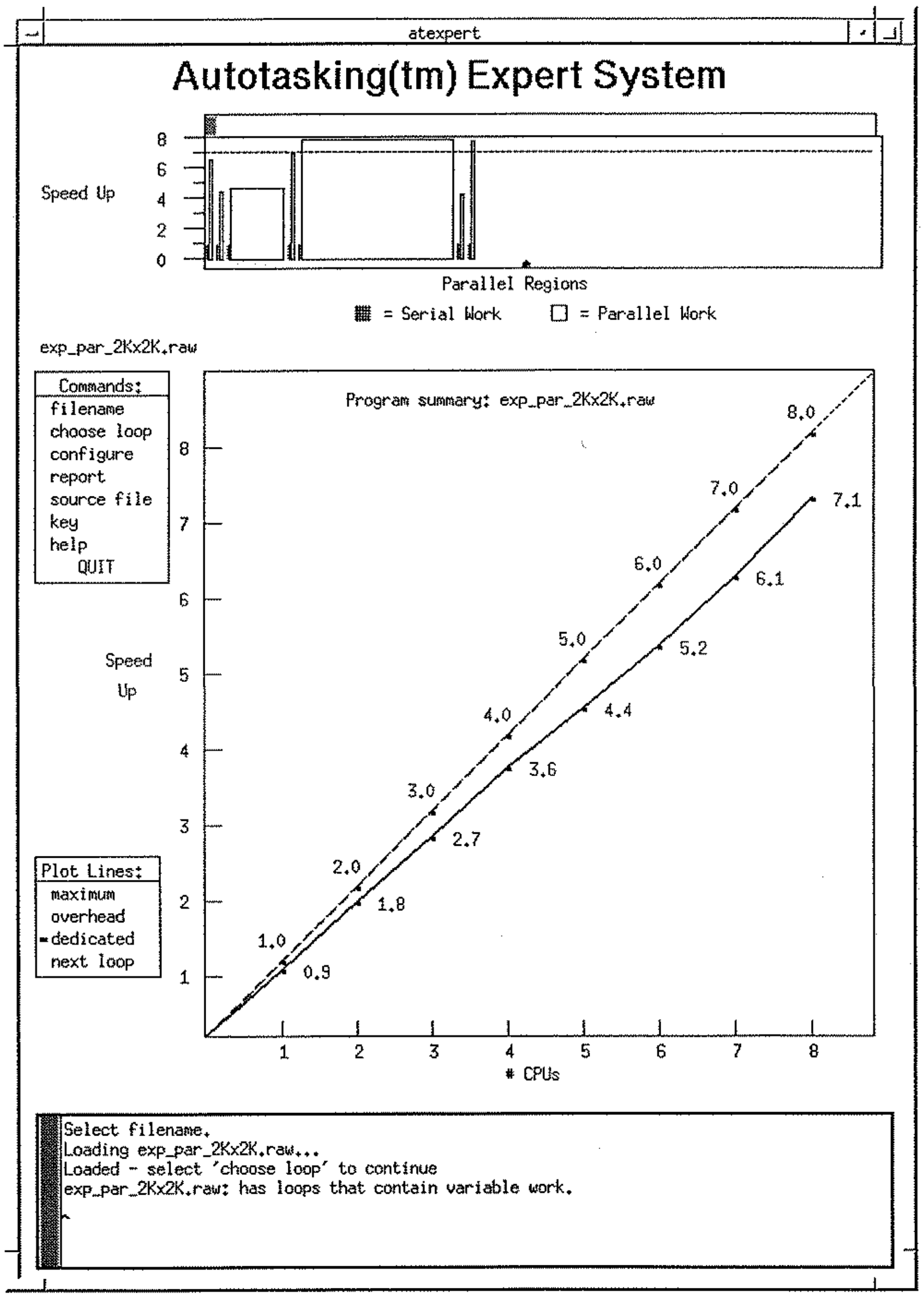
```
Line   12345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
             NAK= NEQ
       C------------------ SCAN ALL BLOCKS ---
             UDMAX=0.0
             DO 20 NBR=1,NBLOKS
       C
       C------------------ CHECK IF BLOCK DELETED ---
       C
             IF(.NOT.DELETED(NBR)) THEN
               IBR=IA(NBR)
               JFIX = IA(IBR)
               NC = IA(IBR+1)
       C*****  Code Expanded From Routine:  MOTION
             IF (JFIX .NE. 0) GO TO 77001
       C------------------ NO ---
       C------------------ VELOCITIES FROM ACCELERATIONS ---
             A(4+IBR) = (A(4+IBR)*CON1+(A(9+IBR)/A(7+IBR)+GRAVX)*TDEL)*CON2
             A(5+IBR) = (A(5+IBR)*CON1+(A(10+IBR)/A(7+IBR)+GRAVY)*TDEL)*CON2
             A(6+IBR) = (A(6+IBR)*CON1+(A(11+IBR)/A(8+IBR))*TDEL)*CON2
             A(9+IBR) = 0.0
             A(10+IBR) = 0.0
             A(11+IBR) = 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 NP = 1, NC
                K2 = K1 + 3
```

```
        click on the variable name in the left-hand window.
To change a variable's status, click on that variable's current status in the left-hand window.
Press the <Done> button when finished altering the the status of variables in the loop.
        <Cancel> button to ignore any changes made to this loop.

cmic$ being added to subroutine...
To save your work, press the <Save> button.
Choose another loop in the left-hand window
     or press the <new Routine> button for another routine
     or the <New File> button for a new program to work on.
```

Figure 8. Annotation window of atscope

Results for explicit annotation

The predicted speedup (Figure 10) for explicit annotation is very satisfactory being significantly better than in previous studies of the SDEM code[9][10]. The predicted and actual speedup achieved for a 3000 block system over 11 iterations for a Cray Research C90 is given in Table 2; constraints on account time prevented a larger number of iterations. The system time for this problem on a Cray Research YMP-EL is 27.945 Sec. for the sequential version and 34.054 Sec. for the parallel version giving some insight into the parallelisation overheads.



```
Select filename.
Loading exp_par_2Kx2K.raw...
Loaded - select 'choose loop' to continue
exp_par_2Kx2K.raw: has loops that contain variable work.
```

(3000 blocks 4000 iterations)
Figure 9. Predicted speedup of manually annotated SDEM using atexpert

Figure 10 shows the general scheme used by SDEM to number blocks in a system. Blocks 57-59 are fixed and contain the movable blocks 1-56. Blocks 32,33,38 and 39 have been removed.

With the loop partitioning scheme used processor-1 may be responsible for the cluster of blocks 1-21, processor-2 for 22-42 and processor-3 for 43-58. With this workload allocation several things may occur:

- having contended for block 57 simultaneously all processors may reach the end of their first layer of blocks and contend for block 58;
- processor-2 appears to have less work than processor-1 but may be busier when blocks fall into the excavation.

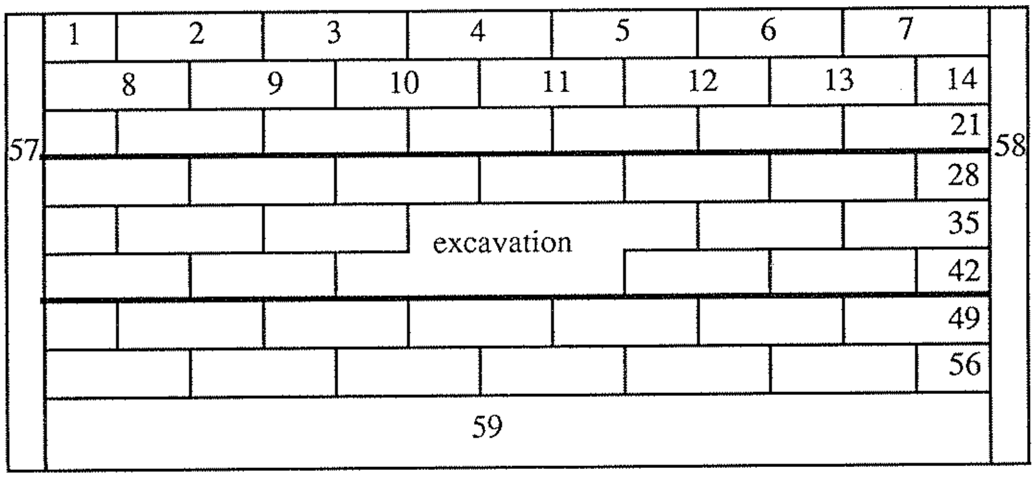


Figure 10. SDEM Block numbering scheme

Table 2 shows the speedup predicted by atexpert, actual speedup relative to the time for the unannotated sequential version of the program, wall clock time, processor time and the number of lock contentions for three different schemes:

i) default loop partitioning with no record locking;
ii) default loop partitioning scheme described above with record locking;
iii) random mapping of processors to blocks.

Case (i) is simply to determine the underlying speedup with no contention on locks. This case may be used to estimate the degradation in performance due to interference by other tasks on the non-dedicated system.

For case (ii) and two processors there has been major contention with run away processor time as processors repeatedly attempt to acquire locks. This situation is highly undesirable and usually unpredictable. The use of large blocks to contain the moving blocks significantly increases the probability of contention. Most internal blocks will have approximately four contacting blocks. The boundary blocks may have hundreds of contacting blocks.

Processors	Speedup ref. Seq.	Wall (Sec.)	System (Sec.)	Lock Conflicts	Speedup $n*(T_1/T_n)$
Sequential program with no annotation					
1	1	4.3	4.3	▨	▨
atexpert predicted on C90					
1	*0.8*	*5.6*	-	▨	▨
2	*1.5*	*3.0*	-	▨	▨
4	*2.8*	*1.6*	-	▨	▨
8	*5.6*	*0.8*	-	▨	▨
Manually annotated version with no locks					
1	1.0	4.3	4.3	0	1.0
2	1.4	3.0	4.3	0	2.0
4	1.8	2.5	4.3	0	3.9
8	2.0	2.1	4.4	0	7.8
Default mapping of blocks in clusters					
1	0.8	5.3	5.2	2	1.0
2	0.1	**29.9**	**57.9**	**7194730**	0.2
4	1.6	2.8	5.4	6729	3.9
8	1.8	2.4	5.5	4551	7.6
Random mapping of blocks using lookup table					
1	0.8	5.4	5.2	1	1.0
2	1.1	3.9	5.4	15586	1.9
4	1.4	3.0	5.4	7278	3.9
8	1.6	2.7	6.1	31159	6.8

(3000 blocks 11 iterations C90)
Table 2. Times for manually annotated SDEM

In case (iii) the sequential block access pattern was circumvented by constructing a look up table whereby the next block to be processed was selected at random for any given loop variable value. The amount of contention is lower but still significant. This is due to the still significant probability of processors requiring the same large fixed boundary block simultaneously. The speedup is lower than the underlying speedup of case (i) probably because of memory accessing patterns.

The system was carrying a significant timeshare load violating the atexpert assumption of a dedicated system. Atexpert also appears not to model locks within loop bodies although it does deal with loop work load imbalance caused by conditional execution within loops. It may be assumed that the speedup for case (i) would be close to the atexpert prediction if the system was dedicated to this application therefore the degradation of case (ii) and (iii) may be compared with case (i) . As the total machine time does not grow quickly from one to eight processors it may be assumed that the load balance is reasonable. The upper bound or ideal speedup can be computed using $n*(T_1/T_n)$ where n is the number of processors and T_1 and T_n are the total machine time for 1 and n processors respectively. This however assumes the code is perfectly parallel.

CONCLUSIONS

The analysis and initial parallelisation of SDEM was made difficult by the complicated data structures. The major data structure is contained in a single integer/real vector with several consecutive elements of the vector constituting a record describing a block. Some of the elements point in turn to other records of contact blocks and all fields are referred to numerically rather than symbolically. It is not surprising that the annotators and vectorisers have some difficulty extracting performance gains.

The test cases typically used by SDEM consist of many layers of blocks contained by large fixed blocks at either end spanning all layers (Figure 3). The systematic numbering of blocks leads to significant contention for exclusive access to the containing fixed blocks. A random mapping of blocks to processors reduced the probability of contention and the subsequent generally unpredictable run away of processor time. This occurred as processors actively attempted to obtain the block records they need by repeatedly reading the lock variable. SDEM is currently being modified to subdivide large fixed blocks into blocks of nearly average size for the system being considered. It is not possible to subdivide blocks internal to the system which are not fixed. Therefore a stratum of larger blocks overlayed by or overlaying a stratum of smaller blocks may continue to cause some difficulty.

Block systems may be very active in some regions as collapses occur and the clustering effect while perhaps minimising lock contention may increase the prospect of work imbalance of processors responsible for low versus high activity regions. Clustering is likely however to reduce communication overhead when supporting this class of problem on collections of high-performance work stations; further work is being done in this area. The strategy of a random mapping of blocks to processors, however, appears to offer some advantages with shared memory multiprocessors.

Experience has shown that the gains due to automatic annotation of the original FORTRAN source were limited but that with a modest amount of effort, critical data dependencies may be resolved using additional manual annotation. The Cray Research FORTRAN tool set significantly eased the task.

ACKNOWLEDGMENTS

The author thanks Dr. M.A. Coulthard of the CSIRO Division of Geomechanics for his support of this research and the provision of the SDEM code studied. The author also thanks the members of the Laboratory for Concurrent Computing Systems, at the Swinburne University 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', *Proceedings ISRM Symposium on Rock Fracture*, Nancy, Vol. 1, Paper II-8. 1971.

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

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

4. Itasca '3DEC - 3-D distinct element code', Version 1.2; User's Manual, Itasca Consulting Group, Incorporated, Minneapolis, 1990.

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

6. Choi, S.K. and M.A. Coulthard 'Mechanics of jointed rock masses using the distinct element method' *International Conference on Mechanics of Jointed and Faulted Rock*, Vienna, 1990.

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

8. Cray Research, 'CF77 Compiling System Volume 4: Parallel Processing Guide', Cray Research, Incorporated, SG-3074 5.0, 1991.

9. Egan G.K. and Coulthard, M.A, 'Parallel Processing for the Distinct Element Method of Stress Analysis', *3rd Australian Supercomputing Conference*, Melbourne, December, 1990.

10. Tang, S.K., Egan, G.K. and Coulthard, M.A., 'Parallelisation of a Distinct Element Computational Stress Analysis Program', American Society of Civil Engineers Technical Council on Computer Practices, *Eighth National Conference Computing in Civil Engineering and Geographic Information Systems Symposium*, June, 1992.