A Parallel Finite Element Procedure for Contact-Impact Problems

Jason Har *

충돌해석을 위한 병렬유한요소 알고리즘

하재선

Key Words : Parallel processing(병렬처리), Domain Decomposition(영역분할), Contact-impact problems(충돌문제), Explicit time integration(외연시간적분), Finite element(유한요소), Bucket-sorting algorithm(버켓소팅 알고리즘)

Abstract

This paper presents a newly implemented parallel finite element procedure for contact-impact problems. Three sub-algorithms are includes in the proposed parallel contact-impact procedure, such as a parallel Belytschko-Lin-Tsay (BLT) shell element generation, a parallel explicit time integration scheme, and a parallel contact search algorithm based on the master slave slide-line algorithm. The underlying focus of the algorithms is on its effectiveness and efficiency for inclusion in future finite element systems on parallel computers. Throughout this research, a prototype code, named GT-PARADYN, is developed on the IBM SP2, a distributed-memory computer. Some numerical examples are provided to demonstrate the timing results of the procedure, discussing the accuracy and efficiency of the code.

1. Introduction

Parallel processing using a multiple instruction multiple data (MIMD) parallel computer is a promising approach to the solution of engineering problems, which usually require considerable computer time on past pipelined vector supercomputers in simulating their structural behaviors. This research presents a parallel processing procedure to the solution of contact-impact response of shell structures, based on displacement-finite element methods, resulting in considerable reduction of computation time. The foundation of the procedure is placed on a perfect domain decomposition strategy and an inter-processor communication minimization strategy. The emphasis of the algorithm is placed on an element-wise block domain decomposition strategy for the parallel BLT shell element generation, a node-wise cyclic domain decomposition strategy for calculating contact force and a parallel explicit time integration scheme. The procedure proposed in this paper is implemented on the IBM-SP2 of The Maui High Performance Computing Center by using MPI. Based on the previous work [1] by the author, this research has been carried out as the extension of that [1] to the contact-impact problems. To show the accuracy and efficiency of the algorithms, pipe whip examples are demonstrated by GT-PARADYN.

* 대한항공, 항공기술연구원
E-mail : mephdjh@krpost.net
TEL : (042)868-6272 FAX : (042)868-6128

2. Belytschko Shell Element Formulation

The BLT shell element [2] has numerous features, including a bilinear four-node quadrilateral shell element with one-point quadrature classified as U 1 element [3] by Hughes, a co-rotational velocity-strain formulation, and an efficient hourglass control eliminating zero-energy modes. The hourglass mode that can destroy the solution due to the inherent singularity of the element might be treated by the algorithm presented by Flanagan and Belytschko [4]. The mass matrix might be also transformed to the lumped mass matrix. This paper took the method proposed by Key [5] and Hughes [6]. The Jaumann stress rate is employed in constitutive relationships to resolve the need for an objective rate of Cauchy stress. Since the material experiences combined kinematic and isotropic hardening beyond the yield

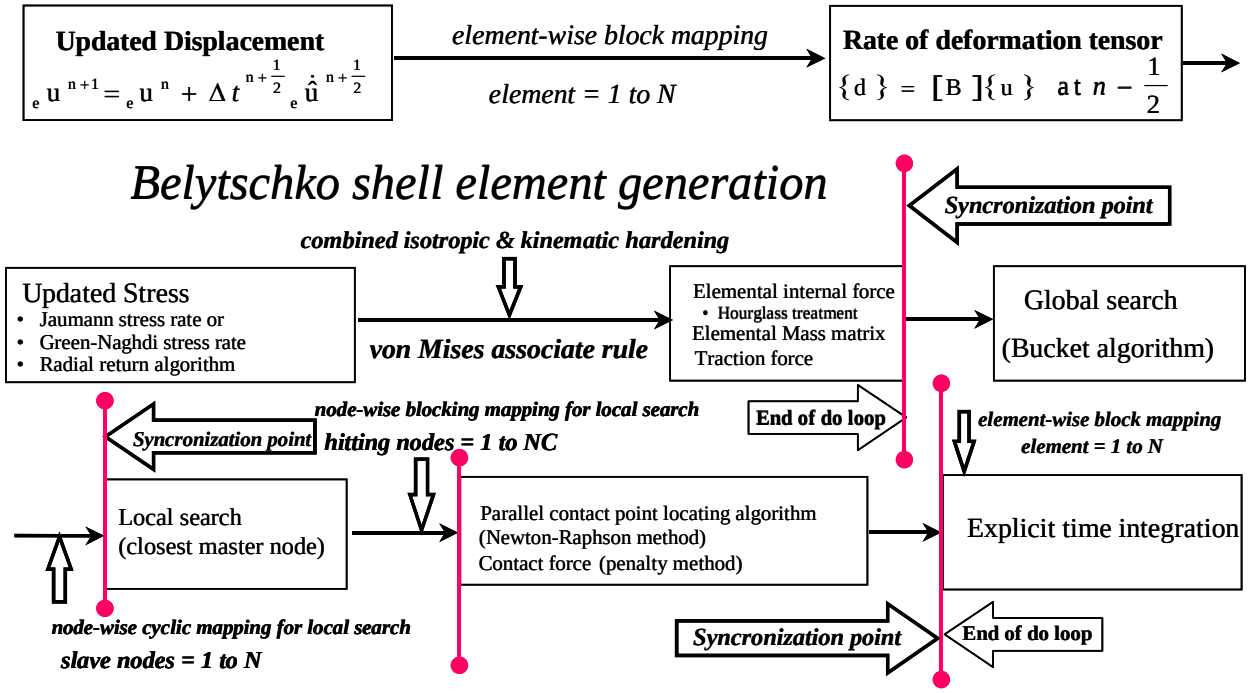

Figure 1 Procedure for parallel contact-impact problems

surface, the computational plasticity procedure related to the radial return method should be applied in this work

3. Parallel Contact-Impact Algorithm

As a contact algorithm, the so-called master slave slide-line algorithm [7] applied in GT-PARADYN includes two issues, e.g., contact search and contact force calculation. The goal of the contact search is the location of the contact point of each hitting node on the master surface (target surface). The search takes place in three stages. As sometimes termed global search, the first stage is to search for a target node (master node) which is closest to a hitting node - in this paper, a bucket sort algorithm can be applied [8]. Benson and et al. [8] exploited an efficient search algorithm, namely a bucket sort algorithm which had been used in the computer science community. The second and third stages are referred to as local search. The second stage is to search for a target segment (master segment) which contains the target node and is closest to the hitting node. Finally the third is to calculate the closest point of the target segment to the hitting node in terms of the isoparametric coordinates of the target segment. According to Hallquist [7], the contact point might be located as the point on the master element that contains the master node closest to the slave node. The contact normal force vector may be calculated by the stiffness factor, based on the penalty method [7]. An equal and opposite force over the master element may be determined. In what follows, we describe the strategy of the parallel contact-impact algorithm the schematic diagram of which is seen in Figure 1.

This work employs the maladroit method as a domain decomposition method. The whole domain can be divided into several sub-domains based on the element numbers which can be sorted in ascending order. A processor whose rank is zero is, hereafter, called “primary processor”. Other processors are referred to as ordinary processors. The primary processor is responsible to acquire information on input data containing the total number of elements and nodes, time incremental size, degrees of freedom per node, boundary conditions, integration points, material property, section property, given load data, element connectivity, coordinates of nodes, and so on. And then, the primary

Table 1 Three-dimensional bucket sorting algorithm

```
ntbxy = ntbx*ntby;
do i=i,ntbz;
        do j=i,nbkz(i);
icell=(i-1)*ntbxy+(nby(ibz(j,i))-1)*ntbx+nbx(ibz(j,i));
ic(icell)=ic(icell)+i
ibxyz(ic(icell),icell)=ibz(j,i);
        enddo ;
ncell(icell)=ic(icell) ;
enddo;
do i=i,nnode;
nbxyz(i)=(nbz(i)-1)*ntbxy+(nby(i)-1)*ntbx+nbx(i) ;
enddo
```

processor is to broadcast input data to ordinary processors. The nodal information should be made on element level, since GT-PARADYN conducts the BLT shell element generation on element level in parallel [1]. The algorithm for the domain decomposition was illustrated in Har [1]. All processors including the primary processor are take part in the three-dimensional sorting scheme at the same time. The three-dimensional bucket sort scheme is so simple and does not take long time. Thus after conducting the sorting scheme respectively and independently, every processor retains the same information, for example, the bucket number of a node and the number of nodes inside a bucket. In Table 1, '*ntbx*', '*ntby*', and '*ntbz*' represent the total numbers of the buckets in the '*x*', '*y*', and '*z*'-direction respectively. '*nbx(i)*', '*nby(i)*', and '*nbz(i)*' represent the bucket number of the node '*i*' in the '*x*', '*y*', and '*z*'-direction respectively, which means one-dimensional bucket sorting. '*nbkx(i)*', '*nbky(i)*', and '*nbkz(i)*' represent the numbers of nodes inside the '*i*'-th bucket in the '*x*', '*y*', and '*z*'-direction respectively. '*ibx(j,i)*', '*iby(j,i)*', and '*ibz(j,i)*' represent the '*j*'-th node number of the 'i'-th bucket in the '*x*', '*y*', and '*z*'-direction respectively. '*icell*' represents the bucket number in the three-dimensional space. '*ibxyz(j,i)*' represents the '*j*'-th node inside the '*i*'-th bucket in the three dimensional space. Thus '*ncell(i)*' represents the number of nodes in the '*i*'-th bucket in the three-dimensional space. Finally, '*nbxyz(i)*' indicates the bucket number of the node '*i*' in the three-dimensional space as seen Table 1.

4. Domain Decomposition on Node Level for Parallel Contact Point Searching

For contact problems, load balanced decomposition can not be achieved by any static domain decomposition techniques unless contacting area is found in advance, because of the fact that some part of the body is in contact, but the other part is not and contact area varies continuously. Thus all processors are responsible to search for potential hitting nodes and master elements that are in contact. In Table 2, let '*myid*' denote the rank identification number of each processor, and let '*nsa*' denote the first node of the slave body, and '*nsb*' indicate the last node of slave body. In order to achieve a symmetric contact implementation, the slave and master body must be swapped every time increment. Thus in the next time increment, '*nsa*' should be replaced by '*nma*' representing the first node of the master body, and 'nsb' should be replaced by '*nmb*' representing the last node of the master body. For all processors to search for potential hitting nodes and elements, the node-wise cyclic domain decomposition is applied so that load balancing between processors can be obtained. A processor with a slave node is about to determine a master node having a minimum distance between the slave node and the master node on the ground that two nodes should share the same bucket. Then each processor stores information on how many slave nodes are concerned, and what slave nodes are potentially about to contact, and what master segments corresponding to hitting nodes are potentially in contact into separate variables with respect to assigned slave nodes. Ordinary processors are next to send their information to the primary processor. Each ordinary processor has different number of nodes and master elements which are potentially in contact. The primary processor plays a role of summing these data and distributing work load. After the primary processor determines information on how many slave nodes and what slave nodes and master elements are potentially in contact over a whole domain, and then it is responsible to make a node-wise block domain decomposition and distribute work balanced load to ordinary processors.

Table 2 Algorithm of searching for potential contacting nodes and elements

```
ks=0;
kr=0;
do i=myid+nsa,nsb,np ;
node-wise domain decomposition
icell=nbxyz(i);
do j=1, ncell(icell);
if (ibxyz(j,icell).gt.nsb) then
dis(j)=sqrt((x(1,i)-x(1,ibxyz(j,icell)))**2+(x(2,I)-
x(2,ibxyz(j,icell)))**2+(x(3,i)-
x(3,ibxyz(j,icell)))**2)
kr=kr+1 ;
else ;
dis(j)=0.0 ;
endif ;
enddo;
if (kr.eq.0) goto 100 ;
dmin=dis(1);
do j=1,ncell(icell)
if (dmin.ge.dis(j).and.dis(j).ne.0.0) then;
dmin=dis(j) ;
  mel=j ; endif;
  enddo;
  knode=ibxyz(mel,icell) ;
  ks=ks+1 ;
  nosl(ks)=i ;
  noma(ks)=knode ;
  nomp(ks)=mel
100 continue;
  enddo   ! i - loop
```

5. Parallel Contact-Point Locating Scheme

The primary processor is ready to make node-wise block domain decomposition for getting ideal load balance. As seen in Table 3, the primary processor distributes work load to each ordinary processor. '*kont*' represents the total number of potential hitting nodes in the slave body. This number can be regarded to as a total load which is about to be divided into the number of available processors including primary processor. Then '*kave*' is the average resulting from dividing '*kont*' by '*np*'.

6. Pipe Whip Problems

Pipe whip problems, which have been investigated by a number of researchers, are considered as good examples to test a contact-impact algorithm. The Nuclear Power Plant design regulations require pipes to be designed against pipe whip. The high-pressure fluid inside the pipe can cause a pipe to whip and impact another pipe. Several examples are presented here. The first one is taken to evaluate the accuracy of engineering results using GT-PARADYN, and the second one to measure the parallel performance of the code.

The two cylinders we have share the same material properties, along with the same length, same thickness and same radius. The Young's modulus is 200 GPa, and Poisson's ratio 0.28, the mass density 7,860 kg/cubic meter, and the yielding modulus is 250 MPa, the hardening modulus is zero. The total length of a cylinder is 2 m, thickness 3 mm, radius 8 cm, the initial gap between pipes 0.01cm. The axes of two cylinders are perpendicular to each other. The finite element model of each pipe is composed of 512 four node quadrilateral shell elements (16 elements around the circumference and 32 elements along the axis). The time step size is taken as 10^{-6} second and the response time is taken up to 2.5 milliseconds, which means time integration loops are carried out 2,500 times. The displacement of each pipe was investigated at 6 points of the pipe. At first, we wanted to see the accuracy of the code rather than parallel performance. The upper

Table 3 Contact point locating algorithm

do i=1, kave ! i = potential hitting node number
All Processors Conduct Local Search
Contact Point Determination
Check out If Penetration Occurs
Contcat Force Calculation
enddo

cylinder is flying with a velocity, -2 m/sec in the 'z'-direction, while the lower pipe is fixed at both ends. The obtained results imply that the two pipes begin to rebound at about 2 milliseconds. For 2.5 milliseconds of the response analysis on one SP2 wide node, GT-PARADYN required about 800 seconds. Malone [10], for the same example, obtained the response up to 1 millisecond. Malone's result showed the displacement of the contact point on the lower cylinder was 0.8 mm, while that of the upper cylinder 1.0 mm. His results implied that 0.2 mm gap between the upper and lower cylinders remained constant up to 1.0 millisecond. By the way, GT-PARADYN showed that the displacement of the contact point on the lower cylinder was 0.86 mm, which was almost in agreement with that by Malone. Malone predicted the displacement of the center point of the upper cylinder at 1 millisecond was about 1.5 mm, while GT-PARADYN showed that it was 2.54 mm. Figure 2 shows the deformed configuration of both pipes which are contacting at 40 milliseconds each other with an initial speed, 20 m/sec, rather than 2 m/sec in order to see a visual exaggeration behavior of the shell structures.

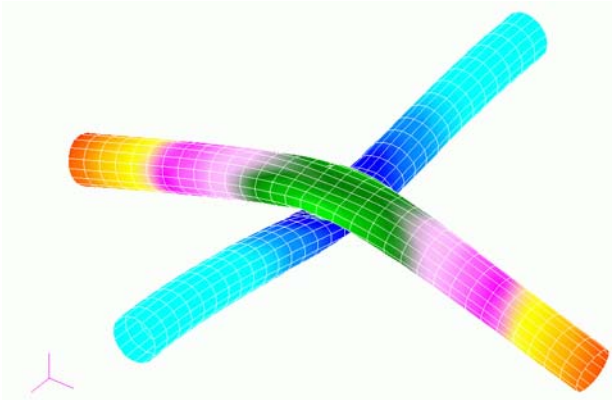

Figure 2 Deformed configuration at 40 milliseconds

And Figure 3 shows the deformed configuration at 60 milliseconds for both pipes contacting without fixed boundary conditions.

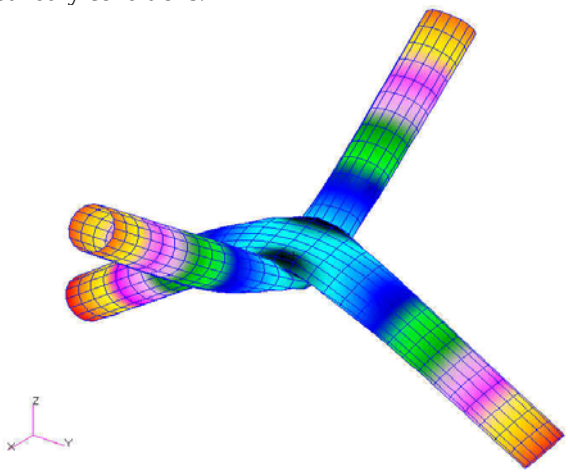

Figure 3 Deformed configuration at 60 milliseconds

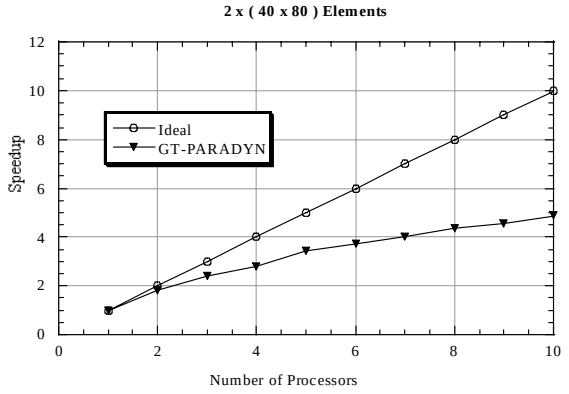

Figure 4 Speed-up diagram for pipe whip problem

Next model explores a scalable parallel performance of the code on the IBM SP2. The number of elements has increased and the pipe sizes are much bigger than those of the previous models. Each pipe consists of 3,200 four node quadrilateral shell elements (40 elements around the circumference and 80 elements along the axis). Both pipes are flying with the same speed, 20 m/sec, at right angles with each other, and toward each other. The total model has 6,400 elements and 6,480 finite element nodes. The time step size is taken as 10^{-6} millisecond and the response duration time is taken up to 1.7 milliseconds, which means time integration loops are carried out 1,700 times. Figure 3 shows the speed-up diagram obtained in this work. Up to 10 processors was applied to see the parallel performance for pipe whip problems. While 11,008 seconds on one processor was consumed as CPU time, 2,256 seconds on 10 processors were needed. The speedup came out with 4.88, and the parallel efficiency became 48.8 %.

7. Conclusion

The contact-impact algorithm that this research exploits consists of three parts, such as the three dimensional bucket sort algorithm, the contact point searching algorithm which includes the contact point location algorithm, and the contact force calculation algorithm. The three dimensional bucket sort algorithm has severe data dependency. The bucket sort algorithm occupies 20 % of the total CPU time of contact-impact treatment when only one processor is employed.

After all, the present parallel contact algorithm aims to reduce 80 % of the total CPU time of contact-impact treatment as the number of processors increases. The contact search algorithm contains lots of branches so that many synchronization points must be inserted. However as long as the number of nodes, which are potentially in contact, increases, the CPU time might be reduced, because these potential nodes must be checked for the penetration possibility. We recommend that more efficient contact algorithms be expected to appear in the light of the results of this work.

Acknowledgment

Computations were performed on the IBM SP2 at the Maui High Performance Computing Center and Georgia Tech. The support of the Maui High Performance computing Center, the Office of Information Technology at Georgia Tech, and KISTI in Taejon are gratefully acknowledged

References

(1) Har, J., 2002, A Scalable Parallel Algorithm for the Nonlinear Transient Dynamic Response Analysis of Shell Structures, Proceedings of the KSME 2002 Spring Annual Meeting,, pp. 476-481.

(2) Belytschko, T., Lin, J. I., and Tsay, C. S., 1984, "Explicit Algorithms for the Nonlinear Dynamics of Shells, *Computer Methods in Mechanics & Engineering*, Vol. 42, pp. 225-251.

(3) Hughes, T.J.R., Cohen, M., Haroun, M., 1978, Reduced and selective integration techniques in finite element analysis of plates, *Nuclear Engineering & Design*, Vol. 46, pp. 203-222.

(4) Flanagan, D. P. and Belytschko, T., 1981, "A Uniform Strain Hexahedron and Quadrilateral with Orthogonal Hourglass Control", *International Journal for Numerical Method In Engineering*, Vol. 17, pp.679-706.

(5) Key, S.W. and Beisinger, Z.E., 1971, "The transient dynamic analysis of thin shells by the finite element method", *Third Conference of Matrix Methods in Structural Mechanics*, pp. 479-518, Wright-Patterson A.F.B., Dayton, Ohio.

(6) Hughes, T.J.R., Liu, W. K. and Levit, I., 1981, "Nonlinear dynamic finite element analysis of shells", *Nonlinear Finite Element Analysis in Structural Mechanics*, Springer, Berlin, pp.151-168.

(7) Hallquist, J. O., Goudreau, G. L., Benson, D. J., 1985, Sliding interfaces with contact-impact in large-scale Lagrangian computations, *Computer Methods in Mechanics & Engineering*, Vol. 51, pp. 107-137.

(8) Benson, D. J., Hallquist, J. O., 1990, A single surface contact algorithm for the post-buckling analysis of shell structures, *Computer Methods in Mechanics & Engineering*, Vol. 78, pp. 141-163.

(9) Key, S. W., 1974, "A Finite Element Procedure for Large Deformation Dynamic Response of Axisymmetric Solids, *Computer Methods in Mechanics & Engineering*, Vol. 4, pp.195-218.

(10) Malone, J. G. and Johnson, N. L., 1994, "A Parallel Finite Element Contact/Impact Algorithm for Non-Linear Explicit Transient Analysis: Part I-The Search Algorithm and Contact Mechanics", *International Journal for Numerical Method in Engineering*, Vol. 37, pp. 559-590.