3rd International Symposium on Impact Engineering '98, 7-9 December 1998, Singapore

A PARALLEL ALGORITHM FOR THE DEFORMATION AND INTERACTION OF STRUCTURES MODELED WITH LAGRANGE MESHES IN AUTODYN-3D

M. S. Cowler, T. Wilson
Century Dynamics Inc. 2333 San Ramon Valley Blvd. San Ramon, CA 94583, USA
(E-mail: msc@autodyn.com)

O. La'adan
Institute of Computer Science, Hebrew University of Jerusalem,
Givat Ram, Jerusalem, 91904, Israel (E-mail: orenl@cs.huji.ac.il)

Abstract

A parallel algorithm for treating the deformation of structures modeled with Lagrange meshes and their impact and interaction with each other has been implemented in the AUTODYN-3D program. The algorithm, which uses separate parallel decompositions for the structural deformation calculation and the contact detection / interaction computation, allows efficient processing on both Massively Parallel Processors (MPP) and Scalable Computing Clusters (SCC), including clusters of PC's. This allows many simulations including impacts, explosions and fluid-structure interactions to take advantage of the high-speed computing capabilities of modern parallel processing systems.

In this paper we describe the novel parallel algorithm and its dynamic load-balancing scheme for contact detection. We also analyze the performance of impact calculations using this algorithm in terms of timing and scalability on two computer systems:

- A four-workstation cluster using an Ethernet network (10 Mb/s)
- A Linux cluster utilizing 16 PCs with a fast Ethernet network (100 Mb/s).

Keywords: parallel processing, Lagrange, impact, AUTODYN.

1. Introduction

AUTODYN-3D is a general-purpose non-linear solid, fluid and gas dynamics program that is routinely used to calculate a wide variety of impact problems [1]. These calculations can be highly computationally intensive if large numbers of elements are employed and extensive surface areas exist, that require testing for impact. To allow such calculations to be performed in more practical time frames, a new parallel domain decomposition and contact detection algorithm have been implemented in AUTODYN-3D for Lagrange grids. It was developed for message-passing MIMD parallel architectures, such as Massively Parallel Processors (MPP) and Scalable Computing Clusters (SCC).

The approach that has been adopted is straightforward domain decomposition whereby each Lagrange grid is statically partitioned into smaller grids called sub-domains. These sub-domains are distributed amongst the processors of the parallel machine so as to minimize inter-processor communications and balance the computational load. Contacts occur when a surface element of one grid interpenetrates another surface element, necessitating a global search of Cartesian space to find possible contacts. Since

contacts can occur between sub-domains owned by different processors, a second, dynamic domain decomposition of Cartesian space is used for the contact calculations.

2. The Serial Contact Algorithm

The Lagrange computation used for each time step in AUTODYN-3D is described in the AUTODYN Theory Manual [2]. At the end of this computation, the tentative new positions of surface nodes have been computed, but there is a possibility that these surfaces may have interpenetrated. A contact algorithm has been implemented to test for, and deal with such cases.

In principal, any two surface elements anywhere in the simulation can come in contact with each other during some timestep, even those that belong to the same object (self-interaction). Checking for all such contacts requires a global search in Cartesian space, and in practice can take 50% of the overall CPU time. For efficiency, the contact nodes and faces are spatially sorted to speed this computation and avoid unnecessary tests of distant elements. Thus, the algorithm can be considered in two parts. Firstly, a calculation is performed to identify neighboring nodes/faces that require to be checked for interaction. Secondly, a detailed interaction calculation is performed for all these identified nodes/faces.

Determining which nodes/faces require to be checked for interactions is achieved by a bucket-sort. A grid of virtual work units is defined in Cartesian space. Each work unit is a cube, with sides twice the smallest face dimension of all interacting faces. In tests, this cube size was found to not only yield the most efficient computing times (due to the fine sort), but also to generate sufficient work units to allow efficient load-balancing for the parallelization (described later). These work units are virtual because storage for a particular work unit is only allocated when it is determined that the work unit contains nodes/faces that are to be tested for interaction.

The bucket-sort loops over all the surface nodes and faces of a problem, and constructs a linked list of the actual work units required for a particular analysis. The sort is performed in two passes, in which all the nodes are sorted, and then the faces are sorted.

First, each node is added to the work unit, which contains it. A hash table is used to achieve fast access time to the virtual work units, effectively speeding the entire sort. If the work unit does not exist, it is created at that stage.

Next, looping over all surface faces of the problem, each face is added to all work units, which contain nodes that might interact with the face. This is determined by checking each node of the face to see if it is contained within, or is in close proximity to, a work unit's domain. At this stage, only work units that already contain surface nodes are considered. The proximity test is based on the size of the contact detection zone used for the interaction logic and the amount of "slack" allowed to enable the calculations described here to be performed less frequently than every cycle.

Finally, the node and face tables built for each work unit in the linked list are examined to determine the total number of node/face interactions that will be required to be computed for the work unit (this number is used to facilitate load-balancing in the parallel version). In general, this will equal the total number of nodes in the work unit times the total number of faces. However, this can be reduced if, for example, self-interaction of a subgrid with itself is not permitted, or two subgrids have been specified not to interact with each other. If the total number of interactions required to be computed for a work unit is found to be zero, then the work unit is removed from the linked list.

At the end of this procedure a compact group of work units has been generated, each containing a list of surface nodes and faces that require testing for interaction. Each node has been uniquely assigned to a particular work unit. Faces have been assigned to multiple work units, as required. These lists may be valid for a number of computational cycles, depending on the proximity test used to determine potential

node-face interactions and on changes in surface definitions (if Lagrange zones are eroded or removed, surfaces need to be redefined).

The detailed interaction calculation that is performed between the nodes and faces in each work unit list is very robust in that every impact is detected and dealt with correctly regardless of the deformation and relative movement of bodies or changes in surface definitions. The method is based on the work published by Thoma and Vinckier [3].

The key to the robustness of this method is the construction of a contact detection zone around each surface face that provides complete "padding" to the face. The contact detection zone for a face is shown in figure 1. It consists of two quadrilateral zones (one on each side of the face), four semi-cylindrical zones along the edges of the face, and spherical zone segments that close the areas around the four corners of the face. If the face is not co-planar, it is split into four triangular faces around its mid-point and these are treated in the same way. In AUTODYN, the radius of the contact detection zone is referred to as the "gap" size. Since contact detection zones of adjacent surface faces always overlap, the algorithm guarantees that no holes exist in the contact surface.

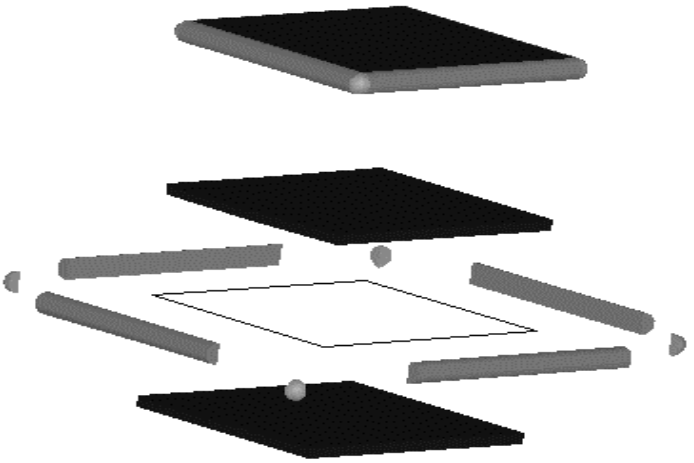

Figure 1: Contact detection zone (top) with exploded view (bottom)

Nodal forces due to contact are added to the internal forces obtained for each element. These forces are computed using a penalty method whereby if any node enters the contact detection zone of a face, it is repelled by a force proportional the depth of penetration of the node into the zone. The gap size is usually set between 10% and 50% of the smallest face dimension.

An additional time step constraint is required for interaction calculations, which satisfies the condition that no node is able to travel more than 20% of the gap distance in one time step.

The example in figure 2 illustrates the use of the contact detection algorithm. It shows the crushing of an octagonal steel girder impacting a rigid wall at 20 m/s. The girder was modeled with 4100 shell elements and in this case it was possible to accurately simulate the shell thickness by setting the gap size to half this dimension. This problem is a severe test for any contact logic and despite the gross buckling, no erroneous surface penetrations were observed.

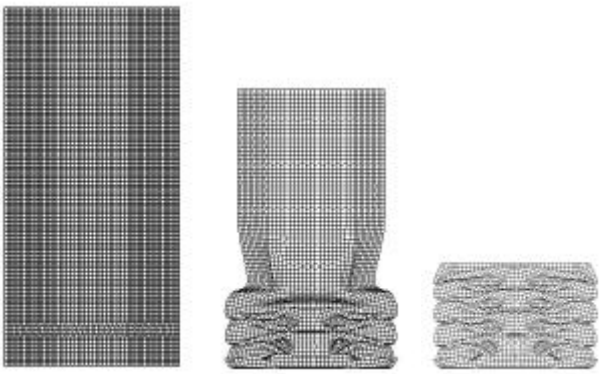

Figure 2: Crushing of octagonal steel girder

3. The Parallel Lagrange Calculation

Parallelizing the Lagrange calculation is a fairly straightforward task. Each Lagrange grid is defined in a three-dimensional index space (I,J,K). Each grid element interacts only with the neighboring elements in this index space. For parallel calculations, each Lagrange grid is divided along index planes in the I, J and K directions to form smaller grids called sub-domains. A load-balancing algorithm is employed to distribute these sub-domains evenly amongst the available processors, with each sub-domain being processed as if it were a Lagrange grid in serial processing (this allows most of the source code for serial processing to be used without modification). When the nodal forces have been calculated for the sub-domains, inter-processor communications are required to exchange forces on the boundaries of sub-domains that are owned by different processors. The load-balancing algorithm attempts to minimize the amount of data exchanged in this way.

It is important to note that because the grid structure does not normally change during the simulations (except Lagrange zones that may be eroded and therefore removed), a static decomposition of the entire index space is sufficient to achieve good performance.

4. The Parallel Contact Algorithm

The contact detection algorithm was developed to allow straightforward parallelization. The approach used is very similar to that used to parallelize the processing of Lagrange subgrids. The work units generated to contain the nodes and faces to be tested for impact are treated in much the same way as sub-domains. Thus, the domain decomposition used for the contact detection algorithm is different from that used for Lagrange calculations.

A second load-balancing algorithm has been implemented that efficiently distributes the work units among the available processors, assuming each processor has either the same speed or a pre-determined relative speed provided by the user. Since different domain decompositions are used for the Lagrange calculation and the interaction computation, this algorithm attempts to minimize the inter-processor communications required between these two decompositions.

Although a static decomposition is used for the Lagrange calculation, a dynamic decomposition is used for the interaction computation. Consequently, load balancing of the newly formed work units is performed for each cycle on which a sort is carried out. This allows interaction calculations to remain well load-balanced even when surfaces are reconfigured as the simulation progresses, or during the erosion (removal) of Lagrange zones.

As our results will show, the contact algorithm generates sufficient work units during the sort phase to allow efficient load balancing for parallel processing. Furthermore, the scheme uses simpler communication patterns than those that rely on recursive coordinate bisection (RCB) to assign equal amounts of nodes to all processors, and adapts well to heterogeneous systems where processors may have different CPU speeds and workloads that may vary with time.

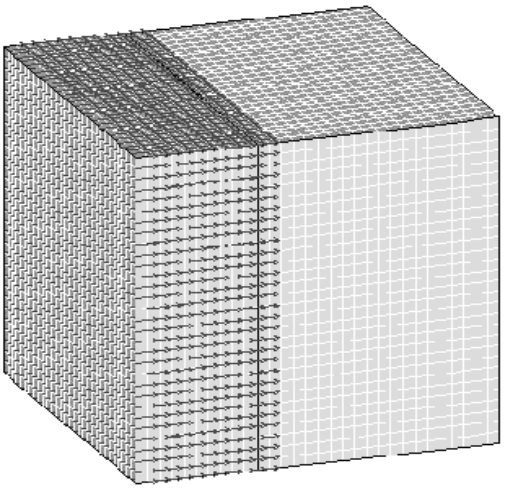

Figure 3: Benchmark calculation

5. Benchmark Calculations

Two benchmark calculations have been performed to test the efficiency and scalability of the parallel processing for typical impact situations. Both calculations simulated a 30cm x 30cm x 10cm steel projectile plate impacting a 30cm x 30cm x 20cm aluminum target plate (figure 3). The calculations used 27,000 elements. The first benchmark assumed that coincident projectile and target nodes were joined together forming a continuous, non-slip interface. This meant that no contact detection processing was necessary and the calculation tested only the efficiency of the Lagrange calculations using domain decomposition. The second benchmark assumed free-slip contact surfaces between the projectile and target and therefore tested the efficiency of both the domain decomposition and the contact detection process.

Tests were performed on an Ethernet network (10 Mb/s) of four DEC Alpha machines using the PVM message passing protocol. Calculation times using 1-4 machines are given in table 1 for both benchmarks.

Table 1: Average cycle times for benchmark #1 and benchmark #2 (Ethernet)

No. of machines	Benchmark # 1			Benchmark # 2		
	Sec/Cycle	Speed-up	Efficiency	Sec/Cycle	Speed-up	Efficiency
1	57.0	1	100%	59.5	1	100%
2	29.5	1.93	97%	31.2	1.91	97%
3	20.1	2.84	95%	21.4	2.78	93%
4	16.0	3.56	89%	17.4	3.42	86%

Similar benchmarks were performed using a 16 PC (Pentium II 300MHz) cluster running Linux on a fast Ethernet network (100 Mb/s) with PVM. For this system, a larger grid representing a 40cm x 40cm x 10cm steel projectile plate impacting a 40cm x 40cm x 30cm aluminum target plate was used. Both calculations used 64,000 elements. Calculation times on this system are given in table 2 for these benchmarks.

Table 2: Average cycle times for benchmark #1 and benchmark #2 (Fast Ethernet)

No. of machines	Benchmark # 1			Benchmark # 2		
	Sec/Cycle	Speed-up	Efficiency	Sec/Cycle	Speed-up	Efficiency
1	90.9	1	100%	94.0	1	100%
2	45.0	2.02	100%	47.8	1.97	98%
4	22.5	4.04	100%	24.4	3.85	96%
8	11.4	7.97	99%	12.3	7.64	96%
16	5.9	15.4	96%	6.4	14.69	92%

Figure 4 shows the performance of the two benchmarks and the speed-up obtained, using the PC cluster. The dashed line denotes perfect speed-up. In both parallel and serial calculations, most of the computation time is spent either in Lagrange calculations or in contact detection. As the results show, both portions of the code perform well, and a good parallel efficiency is achieved. In spite of the slow network that incurs relatively large latency, a near-optimal speed-up is obtained, with the increase in the number of processors.

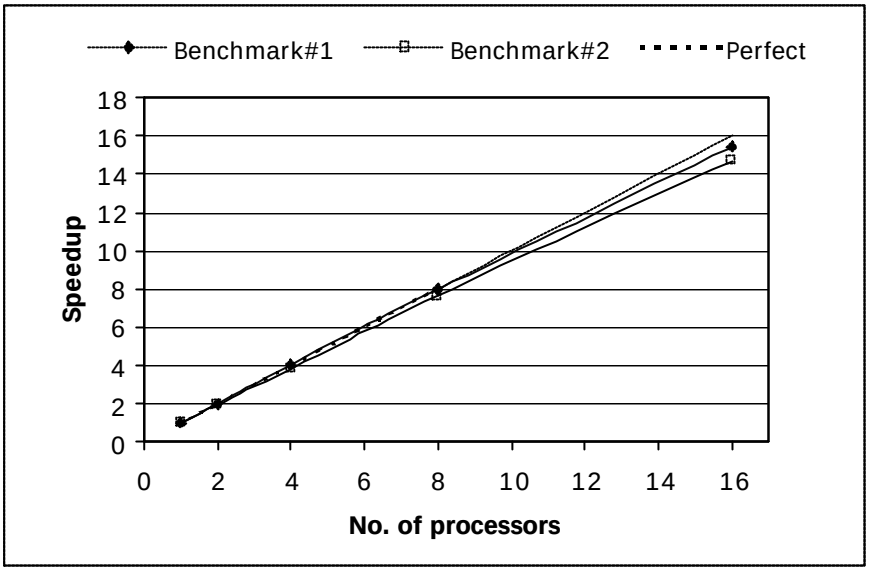

Figure 4: Performance of both benchmarks on a Linux based PC cluster, with fast Ethernet.

It is interesting to examine the behavior of the parallel interaction algorithm. The benchmark used a total of 64,000 elements. Of these, 12,000 (20%) were surface elements. About 3200 work units were allocated during the sort, with each work unit holding 4-10 nodes and 12-30 faces. The fine sort eliminated unnecessary impact tests, and allowed large flexibility for the dynamic load-balancing algorithm to distribute the workload evenly among the processors. Indeed, the speedup for the second benchmark is hardly impaired even for 16 processors (with relatively fine problem granularity).

5. Conclusions

An efficient and scalable parallel algorithm for computing the deformation and interaction of structures using Lagrange grids has been developed. The algorithm uses a static decomposition for processing the Lagrange grid and a dynamic decomposition for computing contact surfaces.

Calculations using the new algorithm have been shown to be more than 90% efficient on low-speed Ethernet networks (10 Mb/s) and close to 100% efficient on a fast Ethernet (100 Mb/s) network. Benchmark calculations show almost perfect scalability for up to 16 processors (the maximum we have tested so far).

Testing is currently underway on PC clusters running Windows NT and on the scalable computing cluster at HUJI utilizing over 80 PCs on a high-speed Myrinet (1200 Mb/s) network.

References

[1] N. K. Birnbaum, M. S. Cowler, M. Itoh, M. Katayama, H. Obata, AUTODYN – *An interactive non-linear dynamic analysis program for microcomputers through supercomputers*, 9th International conference on Structural Mechanics in Reactor Technology (SmiRT), Lausanne, August 1987.

[2] *AUTODYN Theory Manual*, Century Dynamics, 1997.

[3] K. Thoma, D. Vinckier, *Numerical simulation of a high velocity impact on fiber reinforced materials*, IMPACT IV, SMiRT post-conference seminar, Berlin, Germany, August 23-24 1993.