AIAA JOURNAL
Vol. 34, No. 7, July 1996

Parallel Computing for Analysis of Variable Geometry Trusses

Mitsunori Miki*
Doshisha University, Kyoto 610-03, Japan
and
Takahiro Koita†
Nara Institute of Science and Technology, Nara 630-01, Japan

A fine-grained parallel algorithm is proposed to perform the integrated geometric/structural analysis of variable geometry trusses (VGTs) with complicated behaviors such as the interaction between the response of an actuator and a member force, and it is implemented to a multiple instruction stream/multiple data stream type parallel computer. One processor is assigned to one truss node, and each node moves synchronously or asynchronously so that its unbalanced nodal force vanishes. The effect of parallel processing is investigated, and the comparison of synchronous and asynchronous algorithms is performed from the viewpoints of processor efficiency, load balance, and interprocessor communications. The parallel integrated structural and geometric analyses of several VGTs are carried out to find the effectiveness of the proposed method.

Introduction

MASSIVELY parallel computers are attaining practical performance recently, and many methods for distributed parallel processing have been proposed in various analyses. Those are, in the solid mechanics fields, domain decomposition,[1-3] parallel generation and assembly of stiffness matrices,[4] parallel optimization,[5-7] parallel reliability analysis,[8] parallel solving of a nonlinear problem,[9] parallel simulation of multibody systems,[10] etc. However, parallel processing methods have not been studied for truss structures so far, because their analysis is simpler than that of continuous structures, and the demands on parallelization are not severe. On the other hand, the analysis of variable geometric trusses (VGTs) with complicated behaviors with respect to material and geometric nonlinearities and actuator response becomes very complicated as the coupling of geometric and structural behaviors occurs. In such cases, parallel computation becomes necessary to alleviate the complexity, not only to reduce the computation time.

VGTs are attracting attention in space engineering.[11] Such trusses have actuators in their members, and the whole geometry can be changed by varying the length of the active members. The accurate modeling is very important in the detail analysis of VGTs to evaluate the interaction between the member force and the displacement of an actuator. However, this interaction has not been considered so far, and the geometric and structural analyses are carried out independently. On the other hand, the conventional analysis method is not so flexible for the changes in the allocations of members and actuators, and it is difficult to perform many trial and error analyses in the conceptual design stage of VGTs.

Miki and Murotsu[12] have proposed a new object-oriented approach to this problem. The computation time, however, becomes very long in the method, although it can treat several nonlinearities very easily. The parallel processing is necessary also from this point of view.

This paper presents a distributed parallel processing method for the integrated geometric/structural analysis of a VGT that enables us to consider the interaction between the nonlinear responses of actuators and member forces of a VGT. A parallel algorithm is proposed, and several computational results obtained by a massively parallel computer, the discussions on the convergence of the calculation, the efficiencies of the parallel algorithms, and the detail on the parallel processing are presented. The problem is limited within static ones in this paper.

Presented as Paper 95-1307 at the AIAA/ASME/ASCE/AHS/ASC 36th Structures, Structural Dynamics, and Materials Conference, New Orleans, LA, April 10–12, 1995; received Sept. 19, 1995; revision received March 1, 1996; accepted for publication March 5, 1996. Copyright © 1996 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Professor, Department of Knowledge Engineering and Computer Science. Member AIAA.

†Graduate Student, Department of Information Science.

Parallel Distributed Problem-Solving Approach

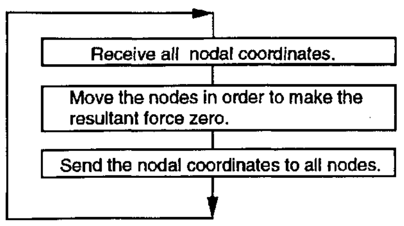
Structural and Geometric Analysis

Miki and Murotsu[12] introduced an object-oriented approach to the structural analysis of trusses and proposed a new method of treating various complicated phenomena such as material and geometric nonlinearities and failure as a result of global and/or local buckling. The object-oriented truss geometric/structural analysis is very similar to the parallel distributed problem-solving approach, which is getting much attention in the artificial intelligence field.[13] That is, the method is not a global method such as solving a global stiffness equation or solving an optimization problem where the potential energy of a whole system is minimized, but a local and distributed method where local constraints are satisfied. The local rule in the method is that the truss node moves so that its unbalanced nodal force vanishes. The geometry change as a result of active members also is obtained by using this rule. This type of distributed problem solving inherently can be carried out by parallel processing.

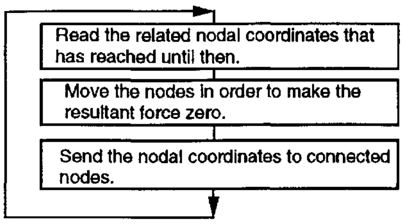
Parallel Distributed Algorithm

The previous method, that is, the object-oriented truss analysis, contains a collection of many objects, such as truss nodes, truss members, supports, materials, sectional shapes, forces, and coordinates, and these objects fulfill their functions distributedly. In parallel processing, assigning a processor to an object is not a good means from the standpoints of ensuring the parallel processing and of giving an appropriate granularity of a task in each processor. Each truss node can move in parallel with each other, but the truss node, truss member, and material cannot perform their tasks in parallel with each other. If a processor is assigned to an object, the function or knowledge in each object is relatively simple, and then the granularity of the tasks, which is defined as the ratio of the amount of computation to the amount of communication, becomes very low, and the computational efficiency also becomes very low. From these considerations, a processor is assigned to a truss node, and the processor fulfills the functions of other objects such as truss members, materials, and sectional shapes.

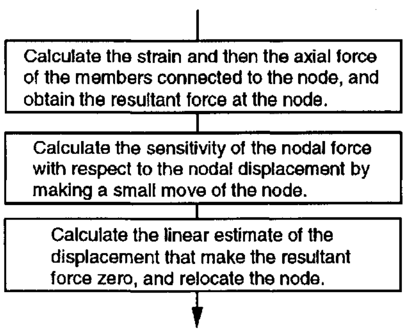
The local constraint condition in the integrated geometric/structural analysis is the balance of force at truss nodes, and the truss node moves to satisfy this constraint. The nodal stiffness matrix is evaluated every time the node moves, and this yields the capability of treating many nonlinearities, such as structural and geometric nonlinearities, and nonlinear behavior of the actuators in active members. The convergence condition is imposed on the residual unbalanced force at the truss nodes.


Receive all nodal coordinates.
Move the nodes in order to make the resultant force zero.
Send the nodal coordinates to all nodes.

a) Synchronous move of the nodes using all data


Read the related nodal coordinates that has reached until then.
Move the nodes in order to make the resultant force zero.
Send the nodal coordinates to connected nodes.

b) Asynchronous move of the nodes without waiting


Calculate the strain and then the axial force of the members connected to the node, and obtain the resultant force at the node.
Calculate the sensitivity of the nodal force with respect to the nodal displacement by making a small move of the node.
Calculate the linear estimate of the displacement that make the resultant force zero, and relocate the node.

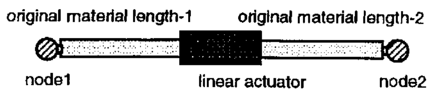
c) Detail of process "move the nodes to make the resultant force zero"

Fig. 1 Synchronous and asynchronous algorithms.

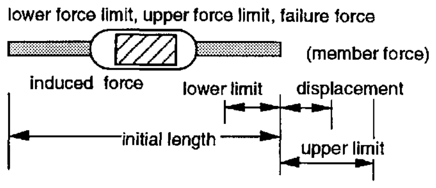
```
Object
  Truss
  TrussNode
  TrussMember
    TrussMemberWithActuator

  Actuatur
    LinearActuator
      DisplacementActuator
      ForceActuator
    RotaryActuator
      RotationActuator
      TorqueActuator
```

a) Class hierarchy for truss member and actuator



b) Modeling of truss member with actuator



c) Modeling of actuator

Fig. 2 Modeling of actuator and active member.

The parallel algorithms for the relaxation of the unbalanced nodal force are shown in Fig. 1, where Fig. 1a shows a synchronous algorithm, Fig. 1b shows an asynchronous one, and Fig. 1c shows the detail of the move of the truss node. In the synchronous algorithm, each truss node moves synchronously after receiving all of the coordinates data, and the iteration process is terminated when the maximum unbalanced nodal force becomes lower than the lower limit, whereas in the asynchronous algorithm, each truss node moves every time it receives the coordinates data from the truss nodes connected to it.

Nonlinear Response of Truss Members and Actuators

Truss members have nonlinear stress-strain relationships as a result of material nonlinearity and buckle under compression load. The actuators are classified into force actuators and displacement actuators depending on the actuator mechanisms. Each actuator has a nonlinear interaction with its axial force and does not act beyond its limit load and displacement. These complicated behaviors of actuators are modeled as shown in Fig. 2.

The behavior of a displacement-controlled actuator is as follows. A target displacement is given, and the actuator extends/contracts to coincide its displacement with the target, but if the axial load exceeds the limit value, the actuator stops acting, and if the axial load exceeds the failure load, the actuator loses load-sustaining capacity.

To incorporate these complicated behaviors into truss structural and geometric analysis, the time sequence of the application of an external force and the movement of actuators is very important since a different load path yields a different result in nonlinear systems. A global clock is used in this system, and the actuator responses are evaluated with small intervals of time. The nonelastic behavior and the buckling of a truss member also can be evaluated similarly. The structural analysis is carried out by these steps with the synchronous or asynchronous algorithm.

Results and Discussions

Computational Implement

The parallel computer used here is nCUBE2 with 64 processors. This computer is a multiple instruction stream, multiple data stream (MIMD) type parallel computer; each processor has an operating system and memory, and it has a processor network of an n-dimensional hypercube. The programming language is C with additional functions for interprocessor communication.

In the proposed method, each processor is assigned to each truss node, and the code in each processor is the same. One processor is assigned for controlling the global state of the other processors. Each truss node has to know the coordinates of the truss nodes connected with it, and the interprocessor communication is needed to exchange the coordinates. In the asynchronous algorithm, each truss node sends its own coordinates to the connected nodes every time it moves, and each processor moves as it receives more than one set of coordinates data. On the other hand, in the synchronous algorithm, all of the coordinates data are exchanged by using an efficient communication method that is called global casting.

The data exchange method by global casting is illustrated for the three-dimensional hypercube network in Fig. 3. For the n-dimensional network, n steps of communication are required to exchange all of the data. If each node sends its own coordinates to all processors by using broadcasting, the total amount of communication is increased, and deadlocks in the network occur.

Aside from the proposed synchronous and asynchronous algorithms, another algorithm, where each node synchronously sends its own coordinates to the connected nodes instead of all nodes, can be considered. Seemingly the total amount of communication is less than the proposed synchronous algorithm, but it eventually increases the amount of communication because the global cast is a very efficient means. On the other hand, another asynchronous algorithm where each node moves after receiving all of the coordinates of the connected nodes might be considered. However, this type of algorithm does not work without synchronization. From these considerations, the proposed two algorithms are considered to be fundamental.

Effect of Parallel Processing

First, the effect of parallel processing is investigated. The integrated geometric/structural analysis is performed by iterating the structural analysis. Therefore, the effect of parallel processing can be investigated by the structural analysis.

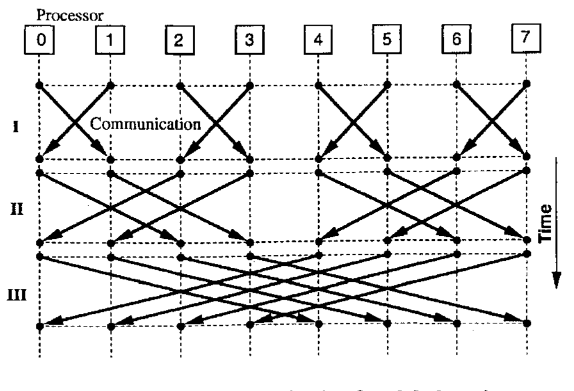


Fig. 3 Processor communication for global casting.

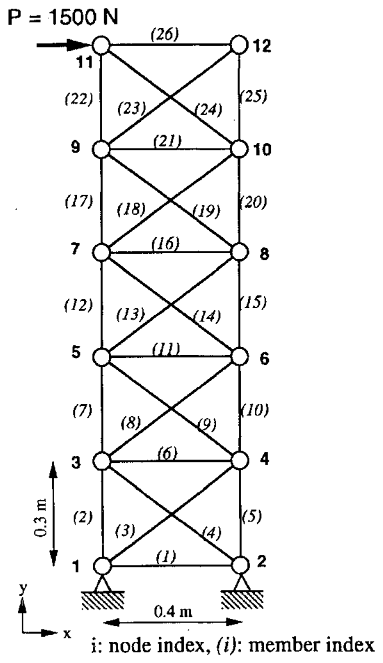


Fig. 4 Truss structure used for the computation.

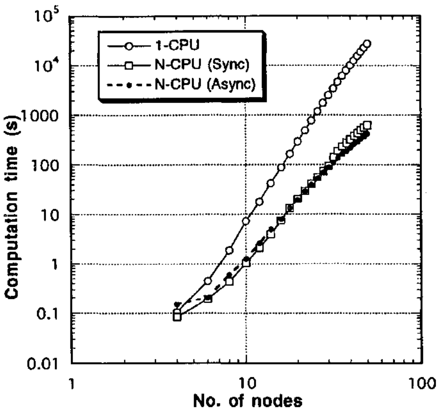


Fig. 5 Relation between the calculation time and the number of truss nodes.

The structural analyses of two-dimensional trusses with 1 stage (4 nodes) to 24 stages (50 nodes), such as shown in Fig. 4, are performed, and the computation time is shown in Fig. 5 as a function of the number of truss nodes. Each truss member is an aluminum circular bar with the radius of 10 mm, and the stress-strain relationship is linear with the modulus of 70 GPa. A horizontal load of 1500 N is applied at node 11. The calculation is terminated as the maximum unbalanced nodal force becomes less than 10 N. The buckling of a member is not considered for simplicity. In Fig. 5, the result obtained by one processor of nCUBE is designated by 1-CPU, and the result by using n processors for an n-node truss is designated by N-CPU. Note that this result is different from the usual one since the problems to be analyzed become more complex as the number of the truss nodes or processors increases, whereas the usual figure depicts time profiles for solving the same problems.

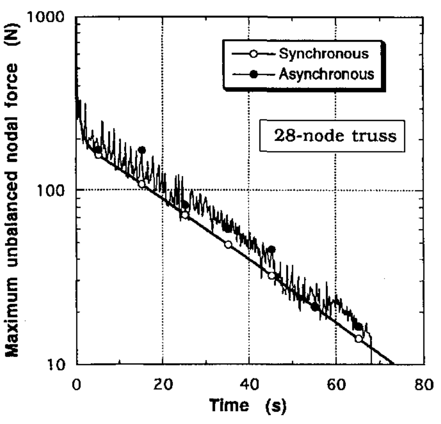


Fig. 6 Histories of the maximum unbalanced nodal force.

This result provides the following conclusions.

1) The relationship between the number of truss nodes and the computation time is approximately linear on a log-log scale; i.e., the computation time is a power of the number of truss nodes.

2) The parallel processing yields considerable speedup.

3) Comparing the synchronous and the asynchronous algorithms, the latter brings a short computation time, and this difference becomes greater as the number of nodes increases.

4) The slopes in these relationships are 4.94 for one processor, 3.85 for the synchronous algorithm, and 3.48 for the asynchronous one. The value of the slope for parallel processing is smaller than that in the sequential algorithm, and then the parallel processing becomes more efficient as the number of nodes increases.

5) The reduction of the computation time in the parallel processing is very important since the integrated geometric/structural analysis requires many iterations of the structural analysis.

Figure 6 shows the history of the maximum unbalanced nodal force during the iteration process. The maximum unbalanced nodal forces decrease rapidly at the beginning, irrespective of the synchronous and asynchronous algorithms, and they decrease linearly on a semilog scale, whereas the nodal force in the asynchronous algorithm shows some small increases and decreases. The reason for such small increases is the lack of the coincidence of the coordinates data along time. The convergence in the asynchronous algorithm is discussed later.

To investigate the efficiency in the parallel processing, an efficiency of parallel processing that is defined as the ratio of the computation time for one processor to the number of processors multiplied by the computation time in parallel processing is considered. The efficiency is shown in Fig. 7, where the number of processors includes one additional processor for the global control. The calculation method used for one processor is the same as that in the synchronous algorithm. The efficiencies increase gradually for the synchronous algorithm and rapidly for the asynchronous algorithm as the number of nodes increases.

At first, the efficiency of the synchronous algorithm is investigated. When the same algorithms are used for sequential and parallel processing, the number of iterations does not change, and therefore the efficiency cannot exceed 1.0. The low efficiency at the small number of truss nodes is a result of the processor communication overhead. As the number of truss nodes or processors increases, the problems themselves become more complex, and the computation time for solving them increases, as shown in Fig. 5. Therefore, the communication overhead is remarkable at the small number of nodes.

The efficiency of the synchronous algorithm increases with the number of truss nodes, but it shows sudden drops at the numbers of

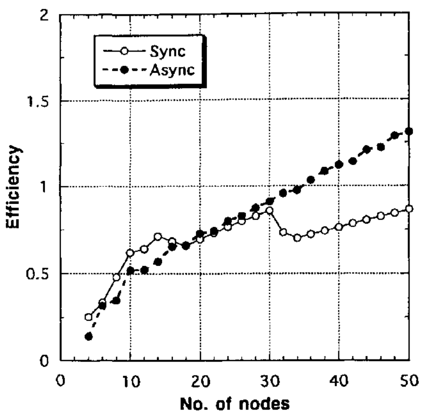


Fig. 7 Efficiency of parallel processing.

truss nodes 16 and 32. It is a result of the change of the dimensions of the hypercube processor network; that is, an additional communication step is required in changing the dimension of the network. Since the number of processors is the number of truss nodes plus one processor, these drops occur at the numbers of processors 17 and 33, and these numbers correspond to the change of the dimension of the network from four to five and from five to six. The efficiency of the synchronous algorithm is considered to be bounded by the global casting of the coordinates data.

On the other hand, the efficiency of the asynchronous algorithm increases as the number of truss nodes increases because this algorithm does not need the global casting of coordinates data to all nodes.

The reason that the efficiency of the asynchronous algorithm exceeds 1.0 is the difference in the algorithms between the parallel and sequential computations. The sequential or one-processor computation uses the same algorithm as the synchronous one where all of the truss nodes move to new positions based on the previous coordinates, whereas in the asynchronous algorithm the coordinates of some of the neighboring nodes are updated. It provides some speedup, and therefore the efficiency of the asynchronous algorithm exceeds 1.0. In either case, the efficiency of the asynchronous one is very high compared with the synchronous one in the range of large structures.

Load Balance Between Processors

In the range of small structures, there is no large difference between the synchronous and asynchronous algorithms. It is a result of small overhead in the global casting, but if the load balance between processors is not proper, much difference occurs between the synchronous and asynchronous algorithms. To find the difference, we analyzed a 50-node truss, which is a 20-node truss with 30 additional members made of a material having zero modulus that are connected to one node; i.e., the computational load is much higher at that node. Figure 8 shows the comparison between the unbalanced 50-node truss case and the balanced 20-node truss case. For the synchronous algorithm, there is much difference in the computation time between the balanced and unbalanced processor load conditions. However, there is little difference in the asynchronous algorithm. It is found from this result that the processor load balance is very important for the synchronous algorithm, but the asynchronous algorithm is efficient even for improper load balance.

Communication Details in Asynchronous Algorithm

To find the detail about the updating of the coordinates of truss nodes in the asynchronous algorithm, the number of received data at a node sent from neighboring nodes is investigated. Each processor has a communication buffer, and the received coordinates data are buffered during the calculation of node moving. After a truss node or processor reads all of the buffered coordinates, some of the coordinates of the neighboring nodes are updated. Then the number of the updated coordinates data is more than one and is not constant.

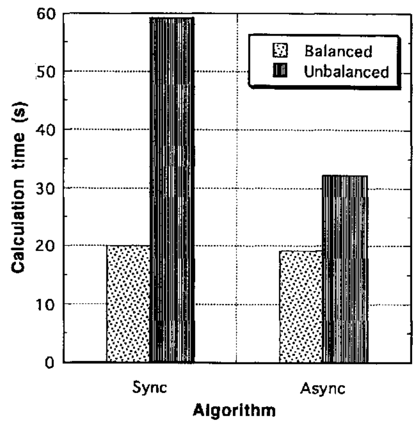


Fig. 8 Comparison between the synchronous and asynchronous algorithms under unbalanced processor load balance.

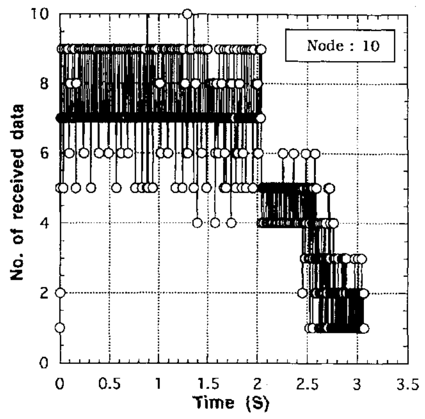


Fig. 9 Number of received data at node 10 during the iteration for the asynchronous algorithim.

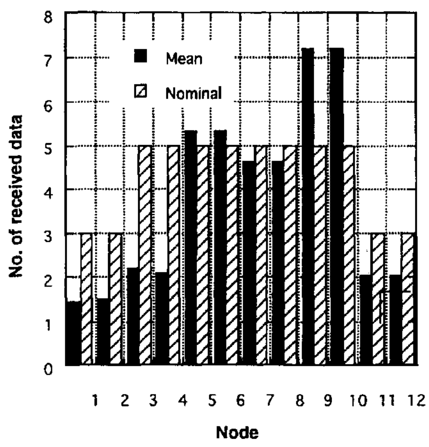


Fig. 10 Comparisons between the nominal and the mean values of the number of received data.

Figure 9 shows the time history of the number of received coordinates data at node 10 of the 12-node truss shown in Fig. 4. It is clear that the number of the received data is not constant, and it decreases when the computation approaches its convergence.

The mean values of the received data for their stationary regions are shown in Fig. 10 where they are compared with the nominal values, which are the numbers of the nodes connected to it. The number of the received data becomes larger than the nominal value when some of the neighboring nodes have relatively small granularity of task, and vice versa. Nodes 11 and 12 move very frequently with each other, whereas nodes 9 and 10 receive their coordinates data very frequently.

These results show that the frequencies of the move at truss nodes are different from each other, and the coordinate data used for the move do not have simultaneity along time. This causes small increase and decrease in the unbalanced nodal force shown in Fig. 6. It is a disadvantage for the convergence in the asynchronous algorithm, whereas the communication overhead in the synchronization causes much disadvantage in the synchronous algorithm.

Convergence of Computation

Using the synchronous and asynchronous algorithms proposed here, the iterative computations for various truss structures and loading conditions were converged. The sufficient condition for the convergence for the Jacobi method is represented as follows[14]:

$$\max_i \left\{ \sum_{j<i} |k_{ij}/k_{ii}| + \sum_{j>i} |k_{ij}/k_{ii}| \right\} < 1 \tag{1}$$

where k_{ij} is an element of the global stiffness matrix.

For truss structures, a simple consideration yields that Eq. (1) does not hold, and the left-hand side of the equation is equal to 1. That is, the sufficient condition for the convergence is not satisfied in the Jacobi method for truss structures. However, Eq. (1) is somewhat too strict, because the computations on many truss structures were converged.

The sufficient condition for the Gauss–Seidel method is looser than the Jacobi method. When a global stiffness matrix is positive definite, the Gauss–Seidel method can be proved to converge.[15] The stiffness matrix of a structure is always positive definite, and consequently the Gauss–Seidel method converges for trusses.

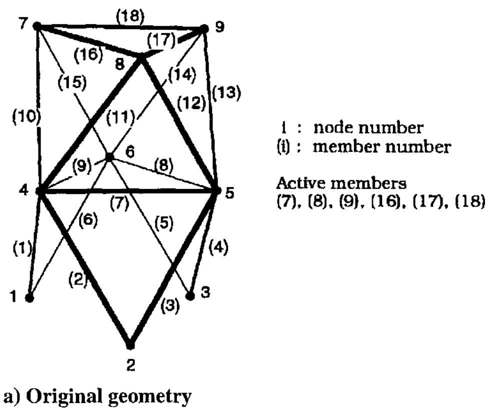
The sufficient condition of the convergence of computation for the asynchronous algorithm also is not clear, but the algorithm is considered to exist between the Jacobi and Gauss–Seidel methods, and therefore the convergence of computation in the asynchronous algorithm can be considered to be assured. From the viewpoint of energy, since any local relaxation of the unbalanced nodal force always reduces the strain energy of the whole structure, any relaxation method including the asynchronous algorithm converges. The small increase and decrease in the unbalanced nodal force shown in Fig. 6 is a local phenomenon, and it is found that it is only a result of the transfer of the unbalanced force from one node to another.

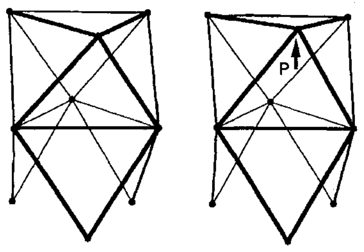
Structural and Geometric Analyses of VGTs

The proposed method is applied to the structural and geometric analysis of several VGTs. The computation is performed at proper time intervals so that the actuators displace synchronously, and the stop or failure of the actuators is evaluated.

Figure 11 shows a nine-node octahedron-type truss with six active members. As stated before, nine processors are assigned to nine nodes. The actuators embedded in the members are of displacement type, and their limit load is 5 kN. The initial geometry is shown in Fig. 11a, where the length of each member is 200 mm. Figure 11b shows the geometries with the target actuator displacement of 25 mm, and Fig. 11c shows the geometries with the target displacement of 50 mm, without loading and with the loading of 7 kN at node 8, as shown. In the loaded case, the actuator in member 7 stops at the displacement of 7.5 mm because the actuator load or the member force exceeds the limit load of the actuator. Comparing the shapes in the unloaded and loaded cases, the length of member 7 is shorter in the loaded cases. Such change in the member force is a result of the change in the geometry.

From these results, the integration of geometric and structural analyses can be carried out using the proposed method, and the interaction between the response of the actuators and the axial force of the member can be easily analyzed. The integration realizes the geometric/structural analysis of indeterminate trusses with active members easily, as shown in Fig. 12. Figure 12a shows the passive deformation and Fig. 12b shows the deformation with the extensions and contractions by 5 cm at members 17, 20, 22, and 25. In these cases, the number of processors used is 12. VGTs generally are statically determinate structures, but this type of active trusses provides a function of changing a load path as well as a function of changing geometries.





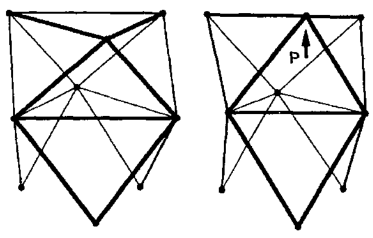


Fig. 11 Integrated structural and geometric analyses of a nine-node octahedron-type VGT.

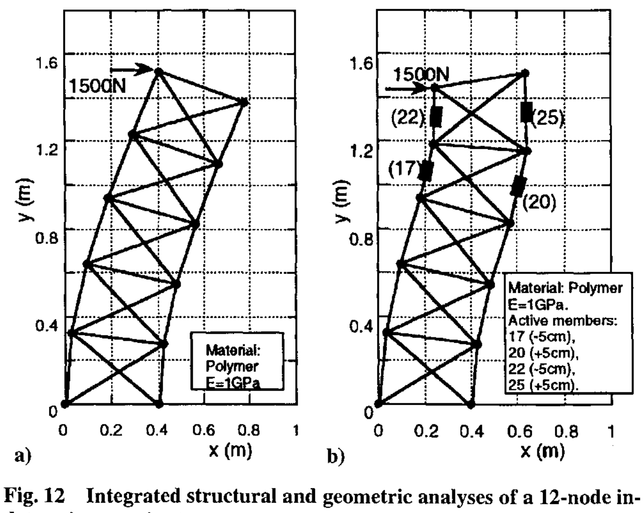


Fig. 12 Integrated structural and geometric analyses of a 12-node indeterminate active truss.

Thirty processors are used for the integrated analysis of a 30-node three-dimensional octahedron-type VGT shown in Fig. 13. The active members, in this case, are all located at one side, which is represented as thick lines, and then their extensions cause bending. The figure shows unloaded and loaded cases with and without actuator extensions, where a horizontal load of 7 kN is applied at one of the top nodes. The calculation time for the loaded case with the extension of 50 mm was about 3 h, whereas it requires more than two days using one processor. It is clear that parallel processing is inevitable for the integrated analysis.

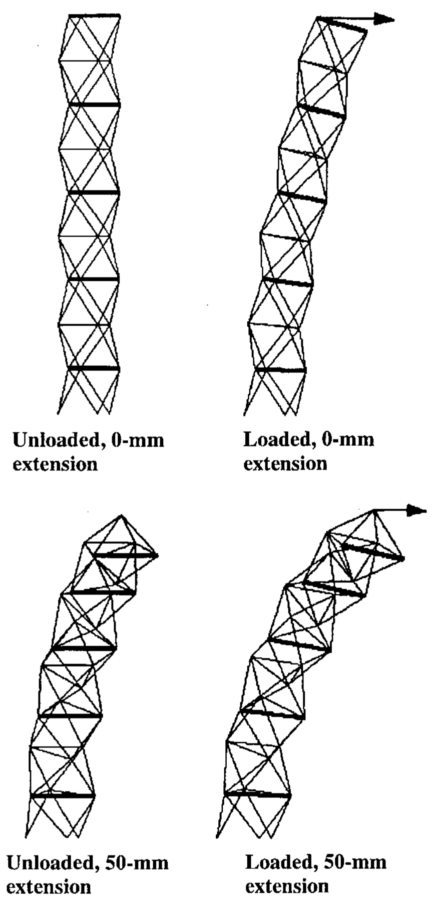


Fig. 13 Integrated structural and geometric analyses of a 30-node octahedron-type VGT.

Conclusions

To perform the geometric and structural analyses of VGTs simultaneously and to analyze the coupling behavior between actuator responses and member forces, new fine-grained parallel algorithms have been proposed and implemented to an MIMD-type parallel computer, nCUBE2. One processor is assigned to one truss node, and the integrated geometric/structural analysis is carried out by using a local rule and message passing with each other node. The proposed synchronous and asynchronous parallel algorithms are found to be effective, whereas the asynchronous one shows a better efficiency, although the simultaneity of the coordinate data sent from the neighboring nodes is not assured in the asynchronous algorithm. Further, the asynchronous algorithm is efficient even in the case where a considerable unbalance of processor load exists. The proposed method is capable of treating the nonlinear interactions between actuator response and member force, and thus the integration of the geometric and structural analyses is performed. The method is also effective for indeterminate VGTs. The computation time of the integrated analysis was about 3 h for a 30-node variable geometry truss using 30 processors. It is found that this type of local-rule-based approach is capable of treating any nonlinear behaviors, and parallel processing is inevitable.

References

[1]Bennighof, J. K., and Wu, J. Y., "Formulation of the Independent Subdomain Response Method for Parallel Transient Computation," *Proceedings of the AIAA 31st Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1990, pp. 673–681.

[2]Calalo, R., Nour-Omid, B., and Wright, M., "An Implementation of the Multifrontal Solution Algorithm on MIMD Computers," *Proceedings of the AIAA 33rd Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1992, pp. 839–847.

[3]Bennighof, J. K., and Kim, C. K., "An Adaptive Multi-Level Substructuring Method for Efficient Modeling of Complex Structures," *Proceedings of the AIAA 33rd Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1992, pp. 1631–1639.

[4]Baddourah, M. A., Storaasli, O. O., Carmona, E. A., and Nguyen, D. T., "A Parallel Algorithm for Generation and Assembly of Finite Element Stiffness and Mass Matrices," *Proceedings of the AIAA 32nd Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1991, pp. 1547–1553.

[5]Shieh, R. C., and Wilson, G. V., "A Massively Parallel Nonlinear Optimization Code Capability and Its Application in Structural Design," *Proceedings of the AIAA 32nd Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1991, pp. 636–643.

[6]Rhodes, G. S., and Sues, R. H., "Portable Parallel Stochastic Optimization for the Design of Aeropropulsion Components," *Proceedings of the AIAA 35th Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1994, pp. 882–891.

[7]Johanson, R., Papalambros, P., Mack, N., and Kikuchi, N., "A Parallel Algorithm for Topology Optimization," *Proceedings of the AIAA 35th Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1994, pp. 2235–2239.

[8]Sues, R. H., Chen, H. C., Twisdale, L. A., Chamis, C. C., and Murthy, P. L. N., "Programming Probabilistic Structural Analysis for Parallel Processing Computer," *Proceedings of the AIAA 32nd Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1991, pp. 1243–1253.

[9]Watson, B. C., and Kamat, M. P., "A Study of Equation Solvers for Linear and Non-Linear Finite Element Analysis on Parallel Processing Computers," *Proceedings of the AIAA 33rd Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1992, pp. 862–869.

[10]Chiou, J. C., Park, K. C., and Farhat, C., "A Natural Partitioning Scheme for Parallel Simulation of Multibody Systems," *Proceedings of the AIAA 32nd Structures, Structural Dynamics, and Materials Conference*, AIAA, Washington, DC, 1991, pp. 2590–2602.

[11]Miura, K., and Furuya, H., "Adaptive Structure Concept for Future Space Applications," *AIAA Journal*, Vol. 26, No. 8, 1988, pp. 995–1002.

[12]Miki, M., and Murotsu, Y., "Object-Oriented Approach to Modeling and Analysis of Truss Structures," *AIAA Journal*, Vol. 32, No. 2, 1994, pp. 348–354.

[13]Bond, A. H., and Gasser, L., "Readings in Distributed Artificial Intelligence," Morgan Kaufmann, San Mateo, CA, 1988.

[14]Ralston, A., *A First Course in Numerical Analysis*, McGraw–Hill, Kogakusha, Tokyo, Japan, 1965, p. 430.

[15]Moldovan, D. I., "Parallel Processing, from Applications to Systems," Morgan Kaufmann, San Mateo, CA, 1993, p. 75.