Annual Report of
ADVENTURE Project
ADV-99-1 (1999)

AUTOMATIC GENERATION OF TETRAHEDRAL MESH WITH TEN-MILLION DOFS

Shinobu YOSHIMURA*, Yoshikazu KATAI, and Hiroshi AKIBA****

* Graduate School of Frontier Sciences, University of Tokyo
e-mail:yoshi@q.t.u-tokyo.ac.jp

**Allied Engineering Corporation
e-mail:katai@alde.co.jp, akiba@alde.co.jp

Key words: Mesh Generation, Tetrahedral Elements, Fuzzy Knowledge Processing, Computational Geometry, Parallel Processing

1. INTRODUCTION

This report describes a program for parallel automatic generation of mesh with ten-million DOFs, which is one of pre-processing modules of ADVENTURE system. Parallel processing is the key technique to perform this kind of large-scale mesh generation because of required high-speed-processing and large memory usage. In this pre-processing system, a user specifies geometric shapes, nodal-density-control information, boundary conditions etc. with the help of a GUI on a front-end PC with Windows, as shown in Figure.1. Then, using the specified information, ten million nodes and tetrahedral elements are generated in the back-end parallel-processing environment (i.e. Alpha cluster). The appearance of meshes is concealed from users, because the scale of mesh is very large. Namely, the geometric model is regarded as the operating object and a user specifies data such as node-density-control information or boundary conditions onto it. The information is passed to the parallel processing environment, and meshes are generated by using the bucketing, Delaunay or other methods. The front-end and back-end machines communicate with each other through TCP/IP sockets. Users can select the parallel-processing environment according to their situation.

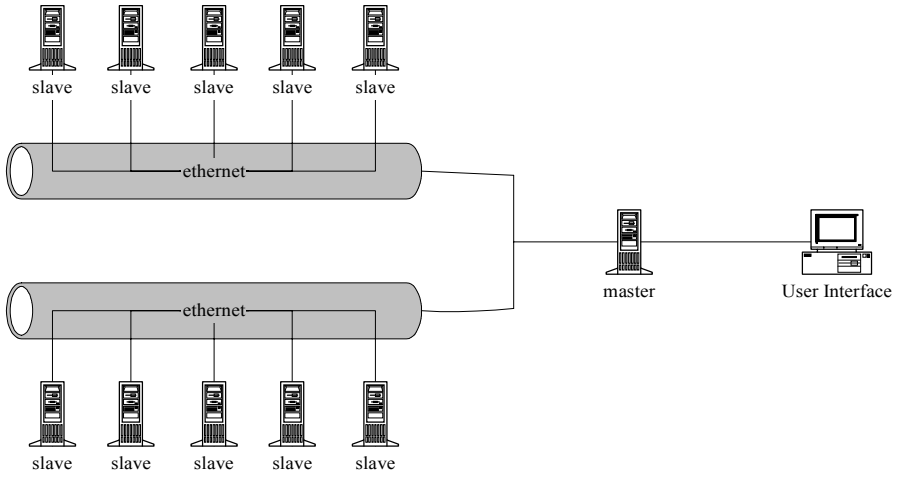


Figure 1 System configuration for parallel generation of tetrahedral mesh

2. METHOD

2.1 Analysis Flow

2.1.1 Definition of an analysis model

The GUI part of the pre-process is created by referring a general-purpose solid modeler, on a PC with Windows. On this GUI, a user composes an analysis model by specifying geometric shapes of the model, node-density-control information, boundary conditions, physical quantities etc., then the information is passed to the parallel processing environment connected by a network. Because the scale of mesh is quite large, the appearance of the meshes is concealed from users. For example, it is difficult to specify information such as boundary conditions directly onto each node or element, in case of ten-million-scale DOFs. Therefore, the geometric model is regarded as an operating object instead of mesh; a user specifies nodal-density-control information, boundary conditions, physical quantities etc. onto it. The specified nodal-density pattern can be confirmed by displaying its contour map or equi-surface. Also the parallel-processing environment, which is an actual part of the mesh generation, can be chosen by users according to their situation (ex. their analysis scale). The mesh generator treats all boundaries as planes in order to decrease CPU time for node generation on surfaces, IN/OUT check decision etc., as shown in Figure 2. Furthermore, standard data formats such as IGES can be treated and input data for the mesh generator are created by converting the geometry data including free-form surfaces into polygon patch data.

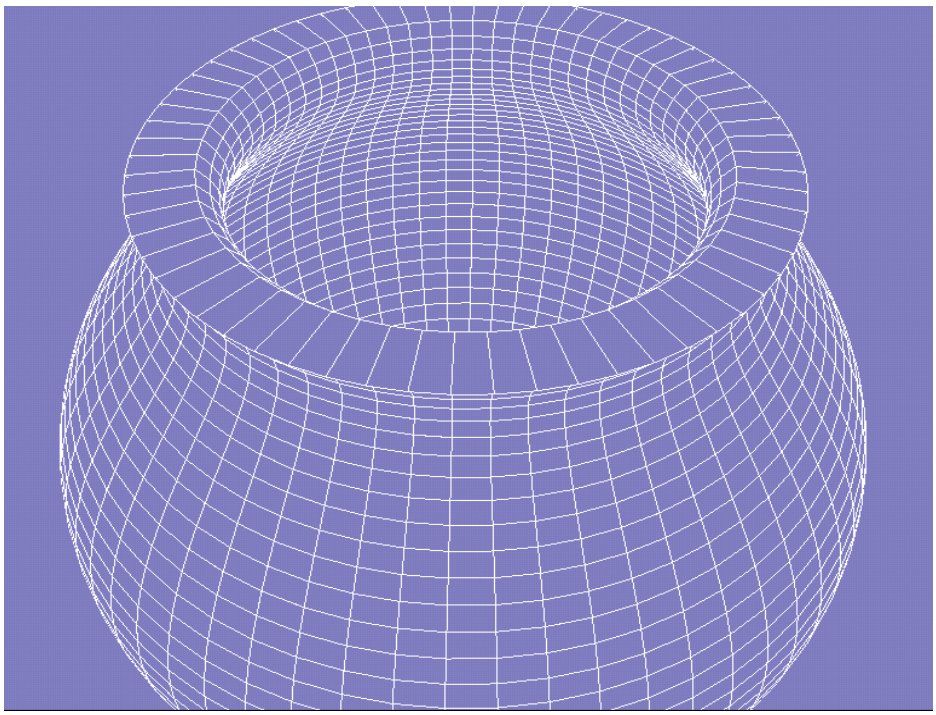


Figure 2 Example of polygon patch

2.1.2 Nodal density control

A user specifies the nodal density by setting several nodal-density functions (i.e. local nodal-density-control information), which are prepared by the system or defined by the user, to typical points of the model. Since these nodal-density functions are independently specified and their effective regions overlap each other, it is ambiguous which density function should be applied. In this system, therefore, the density pattern is determined over the whole analysis region by taking the sum of sets for the overlapping density functions and adopting the maximum value of the density. The processes are shown in Figures 3 and 4. Also the total number of the nodes can be estimated with the obtained density pattern. If the estimated value differs far from the total node number specified by a user or it exceeds the upper limit applicable to the computer environment, the number of nodes can be adjusted by multiplying some correcting coefficient to the whole node density function.

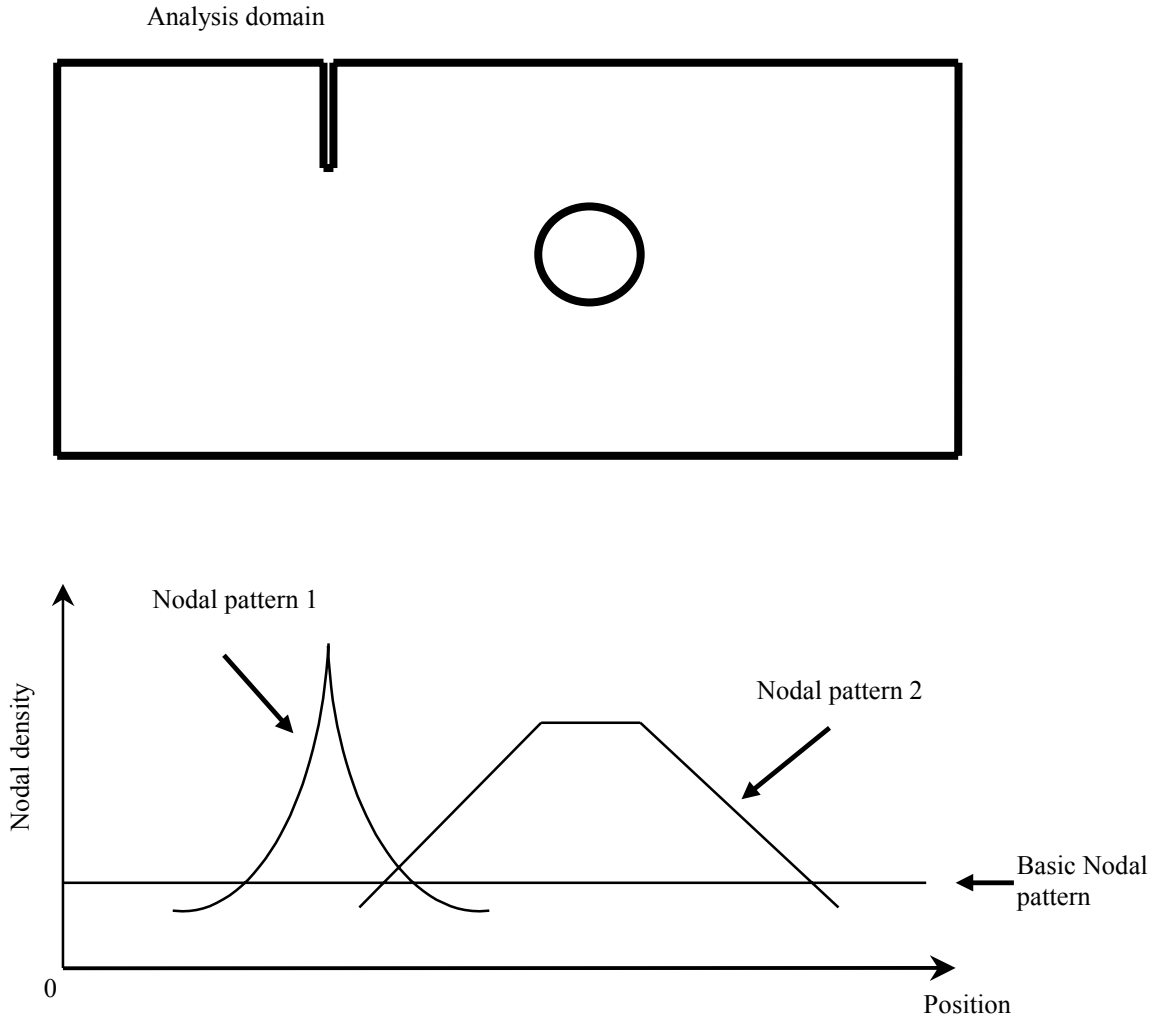


Figure 3 Superposition of nodal patterns based on fuzzy theory (1)

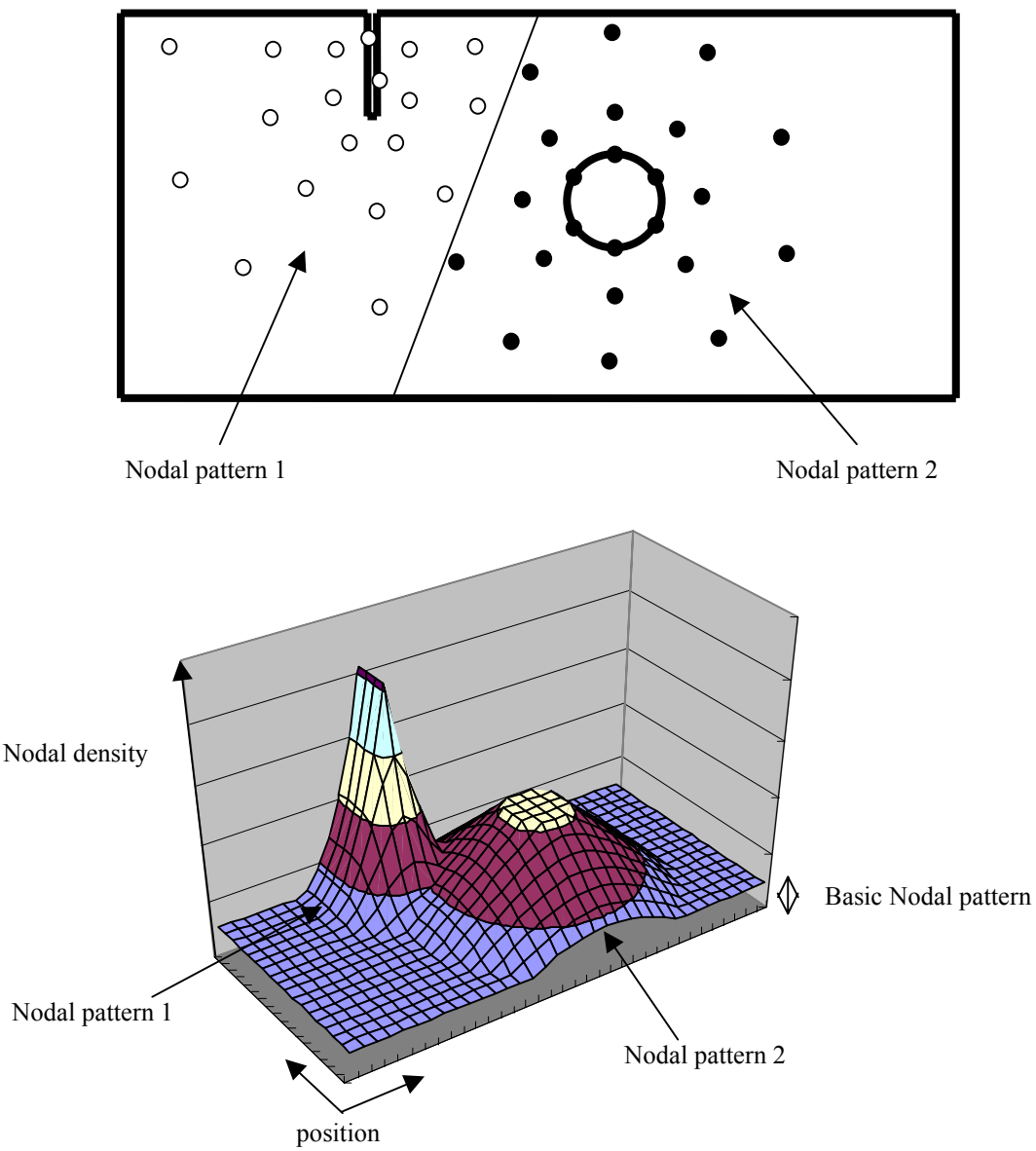


Figure 4 Superposition of nodal patterns based on fuzzy theory (2)

2.1.3 Representation of nodal-density-pattern information

When generating nodes, nodal density is frequently calculated at node-candidate points. If the set of the nodal density functions specified by the user is used at each calculation and also the nodal density field is complicated, this procedure must consume much CPU time. To avoid this, this system adopts an approximation for the nodal density by introducing orthogonal grids, as shown in Figure 5. Namely, the density is calculated in advance at the lattice points of the three-dimensional orthogonal grids with using the specified nodal density functions. Then the density can be obtained at any points by linearly interpolating the values at

eight surrounding vertex points of the rectangular parallelepiped which includes the calculating point. By using this grid method, the density can be obtained in a definite time no matter how complicated nodal-density pattern is specified.

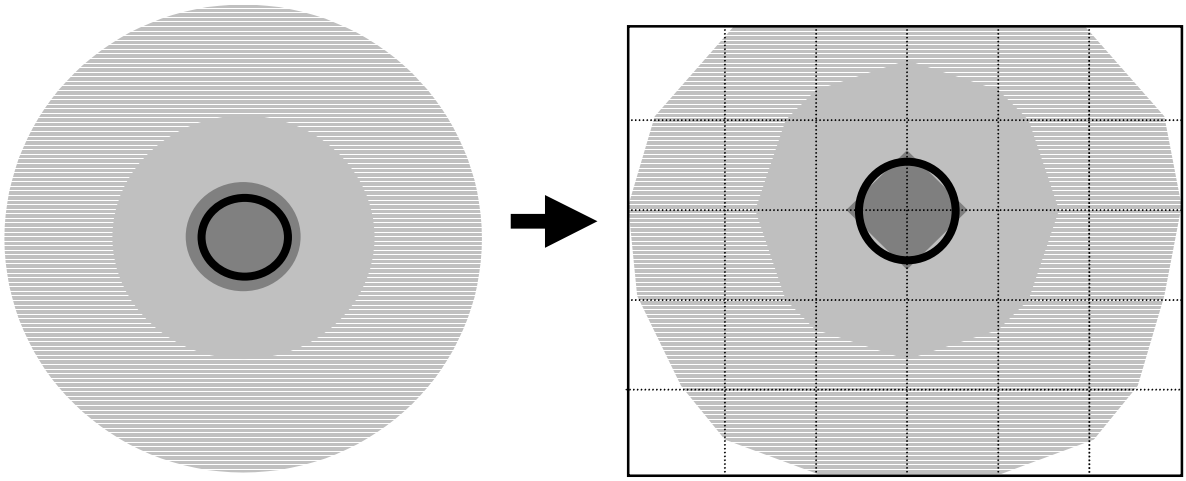


Figure 5 Approximation of node density distribution using an orthogonal grid

2.1.4 Bucketing method

The node generation process is a time-consuming part in mesh generation as well as element generation process. In this process a whole region is decomposed at first into smaller domains called bucket. Then a lot of candidate points are generated from the maximum density within this domain. Suppose the domain is fixed and the density varies rapidly in it, many candidate points will be generated even in a sparse region. This causes much CPU time. Therefore, as shown in Figure 6, the domain is recursively decomposed by the octree method so that the number of candidate points in each bucket becomes almost constant. For each bucket obtained in this way, candidate points are generated with about 1/5 of the node separation calculated by the maximum density in the bucket, as shown in Figure 7. To adopt a candidate as an actual node, the following conditions are imposed.

a) The point is inside the analysis domain.
b) No other registered points are included in a sphere the radius of which is the node separation from that point.

If these two conditions are satisfied, the point is registered as a node. In the search process b), the search area can be restricted to the buckets within the nodal separation. This makes CPU time for node generation proportional to the total number of the nodes. [Yoshimura et al., 1995]

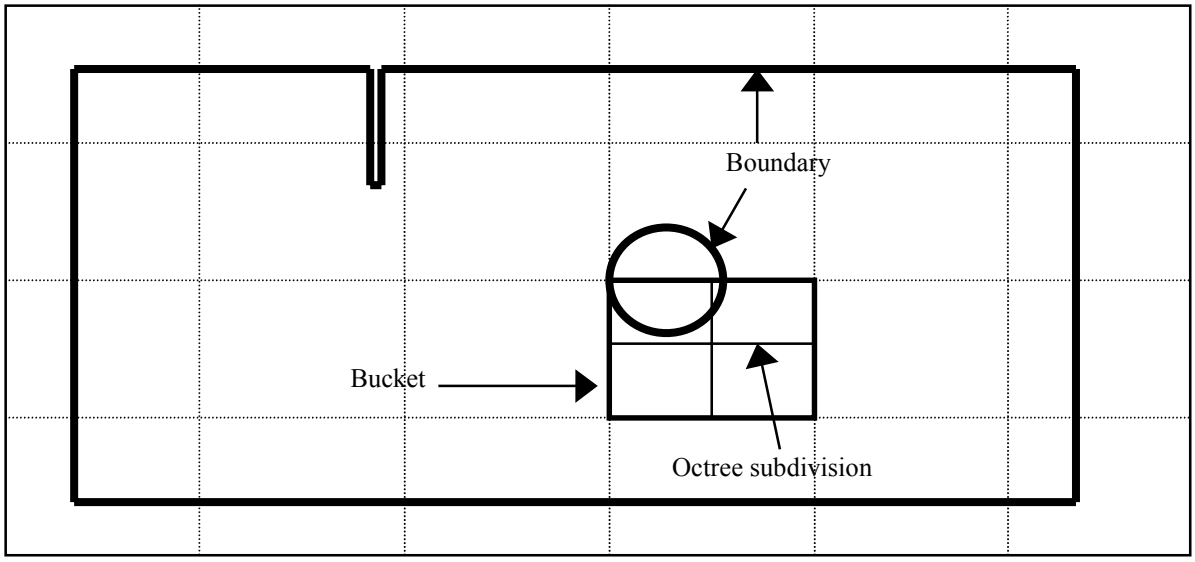


Figure 6 Example of bucket decomposition

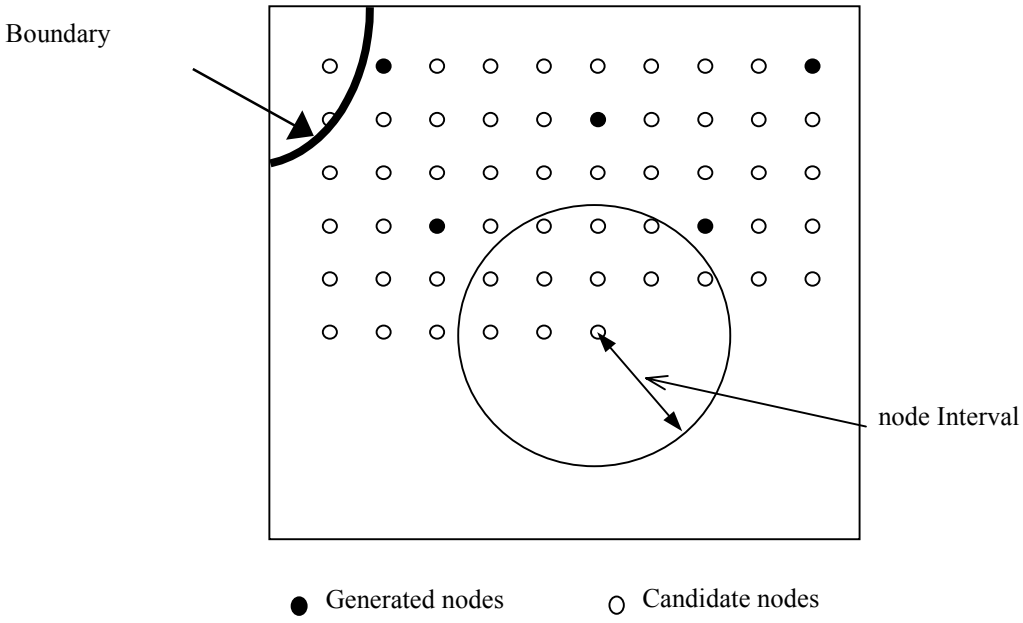


Figure 7 Node generation in one of buckets

2.1.5 Node generation in each parallel processing domain

A rectangular parallelepiped, which is initially prepared on bucket decomposition (i.e. a domain not re-decomposed by the octree method), is regarded as a parallel processing domain. As shown in Figure 8, a child PE receives a parallel processing domain which is not processed yet from the parent PE and starts the process at each time when it finishes previous one, so that the load can be dynamically distributed. As shown in Figure 9, the nodes are generated in advance on the boundary between each parallel processing domain and shared by the neighboring domain. More clearly, the nodes are generated in the following order: on vertices, on edges, on faces and inside of the domain. By generating the nodes on the boundary surfaces in advance, nodes inside the domain can be generated independently. Because of sharing the nodes, the neighboring domains have to communicate; the amount of the communication fairly depends on how to distribute the precessing domain to each child PE. The parent PE manages to distribute the domains to child PEs so that each domain is connected and boundaries with other domains remain as small as possible.

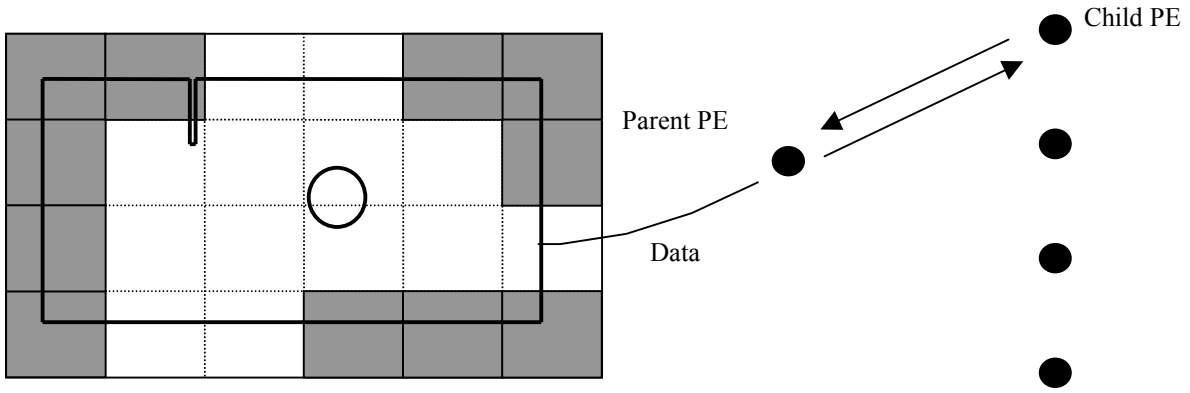


Figure 8 Dynamic load distribution

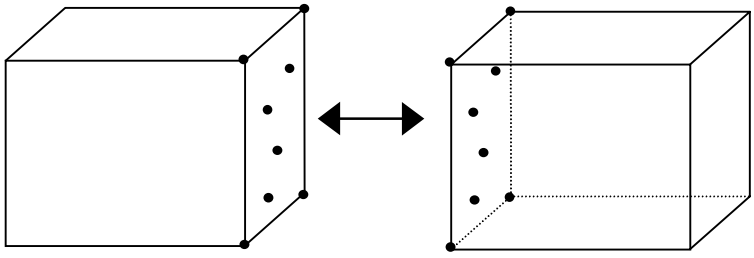


Figure 9 Node generation on a boundary surface between parallel processing domains

2.1.6 Parallel Delaunay method

As shown in Figure 10, Delaunay tetrahedron decomposition is performed for each parallel processing domain with using nodes generated in parallel. A boundary-constraint-Delaunay method (see [J. Conraud, 1995]) is used in order to make the same triangles shared on the

boundary surface between neighboring domains. Here the boundaries are expressed only by planes. If they have the same nodal pattern, the same triangles appear on the boundary surface. Namely, when each domain is decomposed into Delaunay tetrahedrons, the triangles on boundary surfaces satisfy the Delaunay conditions on the plane. After checking inconsistencies due to degeneration etc. the meshes in neighboring domains are connected to make whole meshes. Among several Delaunay-tetrahedral-decomposition methods such as the successive attach method or the decomposed rule method, the successive attach method by Watson's algorithm [D. F. Watson, 1981] is used in this system. It is noted that an element-generation time can also be approximately proportional to the number of the nodes in the domain, if the node-attaching order is adequately designed. That is, the node is attached to each bucket at the first time when a tetrahedron which includes the node in the circumscribed circle, is found. The search region can be approximately restricted to the inside of the bucket and the CPU time is proportional to the number of nodes in the domain.

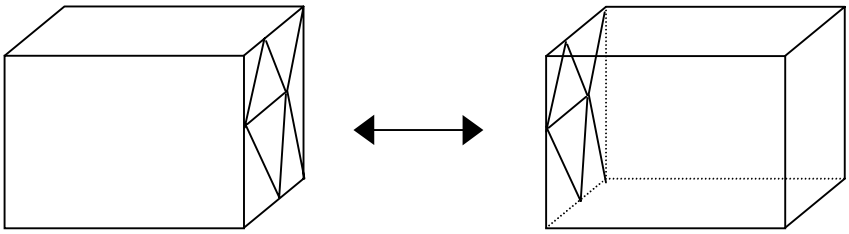


Figure 10 Tetrahedron decomposition in each parallel processing domain

2.1.7 Smoothing operation

The generated elements are checked with their aspect ratio, the dihedral angle etc. and the found distortion is remedied using a smoothing method such as Laplacian smoothing technique. Furthermore, since the method described above generates nodes on the boundaries of the parallel processing domain with top priority, nodes may not be generated on the boundaries of the analysis domains. Therefore the nodal pattern should be corrected so that the nodes stay on the analysis boundaries.

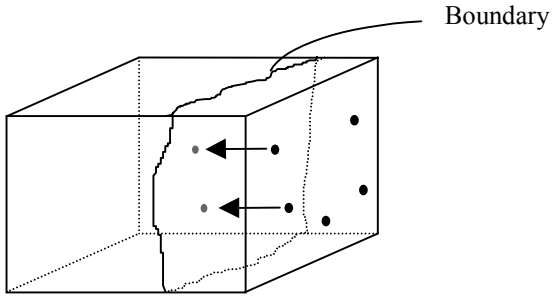


Figure 11 smoothing operation

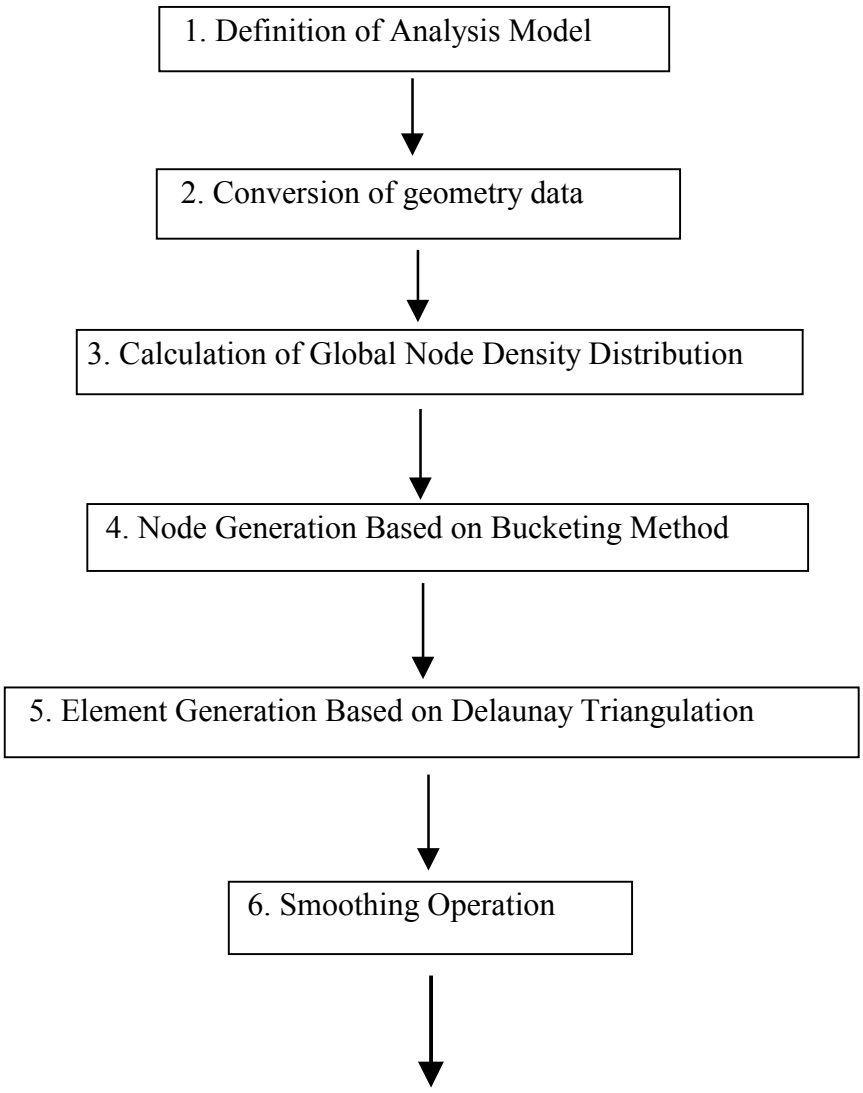


Figure 12 Flow of process

2.1.8 Flow of process

1. Define a whole analysis model. Define a geometric model using one of commercial geometric modelers. Then Attach node density patterns and boundary conditions to the geometry model.
2. Convert geometry data including free-form surfaces into polygon patch data.
3. Calculate global node density distribution using fuzzy knowledge processing.
4. Generate nodes inside the analysis domain, on its boundary surface, edges and vertices using the bucketing method.
5. Generate tetrahedral elements using the Delaunay triangulation technique.
6. Remedy distortion of elements and mismatch elements using the Laplacian smoothing technique and others.

2.2 Implementation

As for the GUI part, Windows NT Ver 4.0, DESIGNBASE Ver 5.2 and C++ (VC++ Ver 4.2) are used as OS, a generic CAD library, and a developing language, respectively. On the parallel processing environment, an Alpha 533 MHz (with 2MB cache, 256MB RAM and 2GB HD) cluster, Linux, C++ (g++ ver 2.7.2) and MPI are used as OS, a developing language and a parallel processing library, respectively.

3. RESULTS AND DISCUSSIONS

Figures 13 and 14 show examples of tetrahedral meshes. They are generated using one PE. Figures 15 and 16 show histograms of aspect ratio and dihedral angle. Almost all aspect ratios are lower than 4.0 and even the worst value is around 10.0. These are rather good results. The dihedral angles tend to concentrate to 90, however this can be avoided by changing the method generating candidate points on the node generation process. Also, as shown in Figures 17 and 18, the CPU time is almost proportional to the number of nodes both on the node and element generation processes.

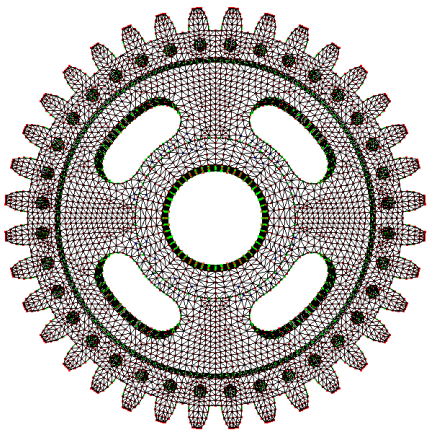


Figure 13 Sample1 (Gear : 20000 Nodes)

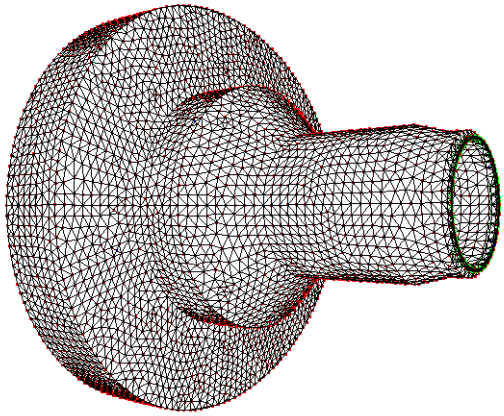


Figure 14 Sample2 (Nozzle : 15000 Nodes)

	Maximum	Minimum
Aspect ratio	10.751	1.0169
Dihedral angle	178.92	71.318

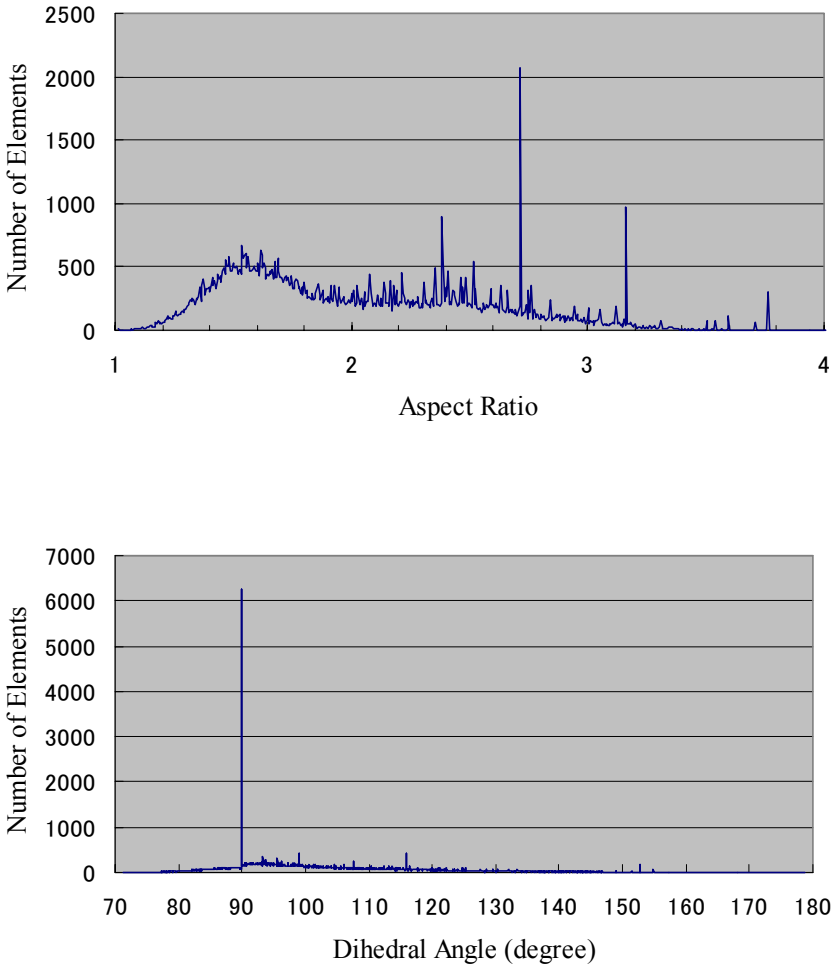
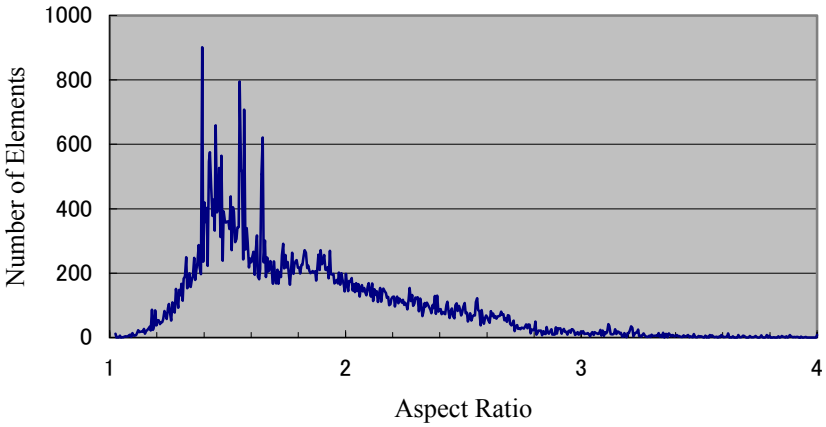


Figure 15 Aspect ratio and Dihedral angle (Sample1:Gear)

	Maximum	Minimum
Aspect ratio	10.704	1.02382
Dihedral angle	178.916	71.7479



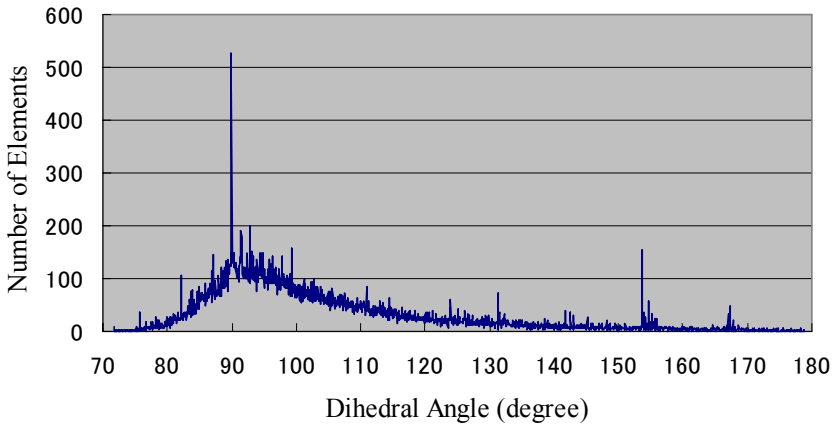


Figure 16 Aspect ratio and Dihedral angle (Sample2:Nozzle)

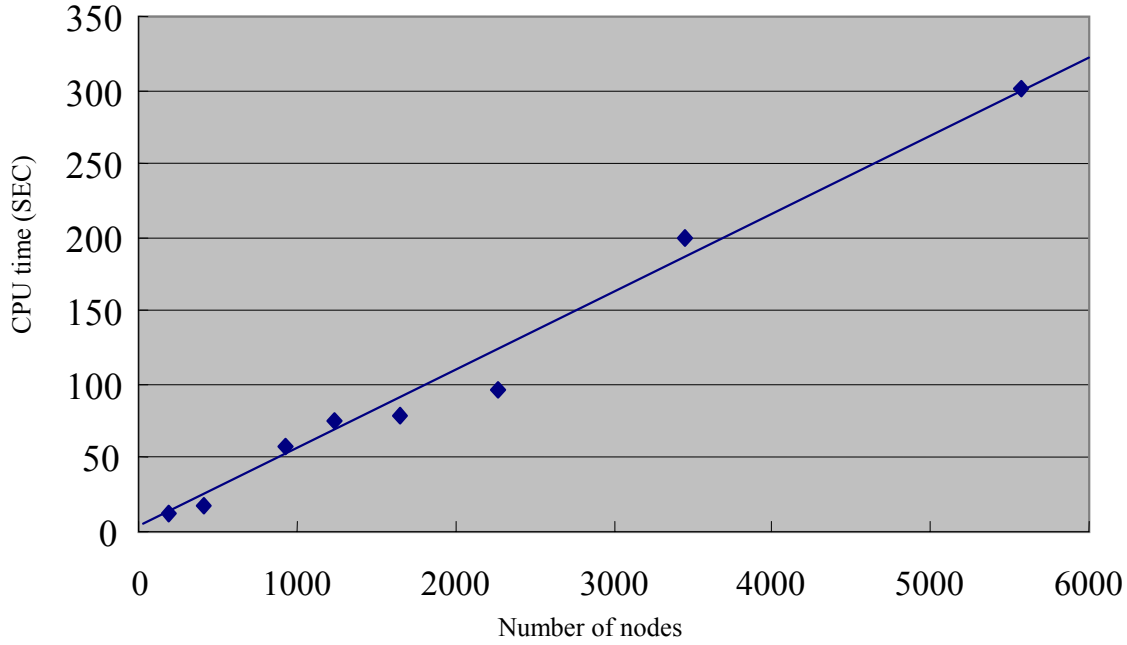


Figure 17 CPU time for node generation (DEC Alpha 533MHz)

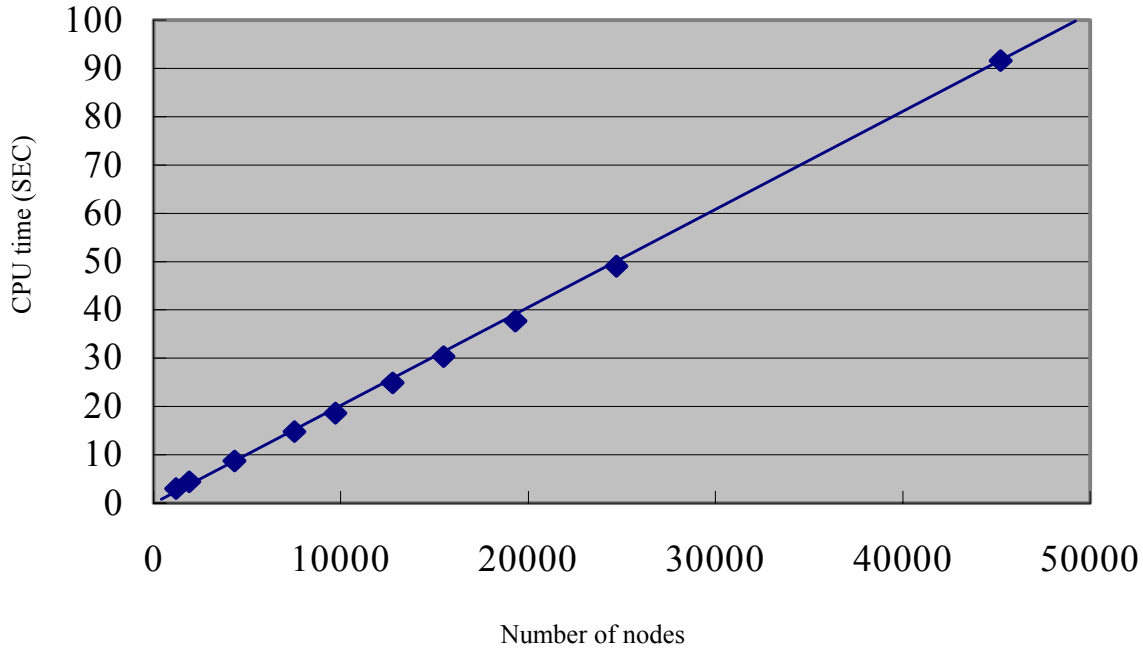


Figure 18 CPU time for element generation (DEC Alpha 533MHz)

4. CONCLUSION

This report described a system for automatic parallel generation of tetrahedral mesh with ten-million DOFs. Fuzzy knowledge processing is adopted to well control a node density distribution, while the bucketing method and Delaunay triangulation are used to generate nodes and elements, respectively. These processes are parallelized. GUI is constracted on a front-end PC with Windows, and a PC cluster with Linux is adoped to deal with all the parallel processes. Furthermore, meshes with one-million-scale degrees of freedom is generated on 1 CPU with using a part of the system, and satisfactory results are obtained with respect to the quality of the meshes and the CPU time.

REFERENCES

Yagawa, G., Kawai, H., Yoshimura, S. (1993), Parallel CAE System For Large-Scale 3-D Finite Element Analyses, Proc.12 SMiRT, Stuttgart, Germany, B, pp183-194

Yoshimura, S., Lee, J.S., Yagawa, G. (1995), Automated Analysis of Stress Intensity Factor for 3-D Cracks, Proc.5th JSME/ASME Joint International Conference on Nuclear Engineering, Kyoto, Japan, 1, pp.357-362

Yagawa, G., Yoshimura, S., Nakao, K. (1995), Automatic Mesh Generation of Complex Geometries Based on Fuzzy Knowledge Processing and Computational Geometry, Integrated Computer-Aided Engineering, 2, pp.265-280

Watson, D.F. (1981), Computing the n-Dimensional Delaunay Tessellation with Application to Voronoi Polytopes, The Computer Journal, 24, pp.167-172

Conraud, J. (1995), Lazy constrained tetrahedralization, Proc.4th International Meshing Roundtable, Sandia National Laboratories, pp.15-26