1989-04-01

Automatic Meshing of Free-Form Deformation Solids

Ssuta S. Hsu
Brigham Young University - Provo

AUTOMATIC MESHING OF FREE-FORM DEFORMATION SOLIDS

SSUTA S. HSU

AUTOMATIC MESHING OF FREE-FORM DEFORMATION SOLIDS

A Dissertation

Presented to the

Department of Civil Engineering

Brigham Young University

In Partial Fulfillment

of the Requirement for the Degree

Doctor of Philosophy

by

Ssuta S. Hsu

April 1989

This dissertation, by Ssuta S. Hsu, is accepted in its present form by the Department of Civil Engineering of Brigham Young University as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Michael B. Stephenson, Committee Chairman

Thomas W. Sederberg, Committee Member

Steven E. Benzley, Committee Member

20 Mar 89
Date

LaVare Merritt, Department Chairman

ACKNOWLEDGEMENTS

The author wants to give thanks and appreciation to Dr. Michael B. Stephenson for the idea of meshing, to Dr. Thomas W. Sederberg for the technical advise in modeling. Deeply thanks and appreciation gives to Dr. Steven E. Benzley for his time and patience in expert advise given throughout the development of this research. Above all, appreciation and love go to the author's loving and patient wife, Eva for all the time and effort she has put into supporting this long stay in the school.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

1.0 INTRODUCTION

Computer-aided techniques are established as an effective way to approach modern engineering design and analysis. One aspect of Mechanical Computer-Aided Engineering(MCAE) consists of geometric modeling, mesh generation, finite element analysis, and result post-processing. Geometric modelers provide the capability to define the geometric model. Mesh generators create the finite element grid of the geometric model to be used for analysis. The finite element analysis software generates a solution to the boundary conditions applied to the geometric model. Post-processing provides the capability to display the analysis results using high performance graphics.

Mesh generation is the bridge between the geometry and finite element analysis procedure. Although the geometric and finite element analysis procedures are well developed, the process of discretizing a geometry into discrete numerical models for analysis is generally time consuming and tedious. There have been many programs written to generate finite element grids, but their applications are limited. Current research is directed toward automatic mesh generation. Such a procedure can produce a valid finite element mesh for an object with limited user input.

Solid free-form deformation geometric modeling, developed by Sederberg and Parry[29], is a very powerful tool in MCAE. It has the ability to create a geometric model by

sculpting surfaces with flexibility akin to clay in a sculptor's hand. In addition, any predefined grid can be modified by the free-form deformation process to create any shape defined by the user. Free-form deformation involves mapping by using a trivariate tensor product Bernstein polynomial[29]. Figure 1.1 is an example of a model created by solid free-form deformation.

Often, the free-form deformation process distorts the initial elements so that they are unacceptable for finite element analysis. To correct this difficulty, a new method of mesh generation needs to be developed to create acceptable finite elements for the model.

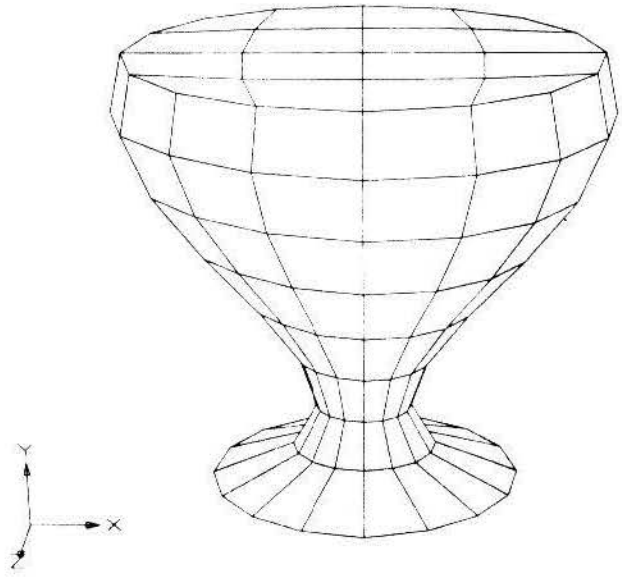


Figure 1.1 Free-form deformation grid model

The objective of this work is to develop a three-dimensional solid finite element mesh generator that will

automatically create acceptable hexahedral finite elements in conjunction with free-form deformation solid modeling. This automatic free-form deformation mesh generator combines the creation of the geometric model and automatic generation of hexahedral finite elements as shown in figure 1.2.

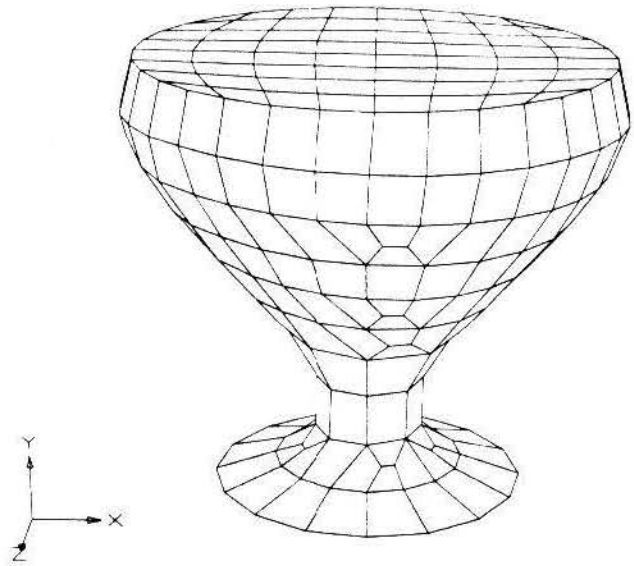


Figure 1.2 Automatic Free-form deformation model

Section two of this paper provides the background information on both the free-form deformation solid modeling and the automatic mesh generator. A description of the development of a new automatic mesh generator based on free-form deformation solid modeling is given in section three. Section four presents some examples which have been meshed by this new method and explains the user interaction required. Section five gives the conclusions and discussion.

2.0 BACKGROUND INFORMATION

Meshes were generated manually until in the early 1960s when computer assistance began to simplify the task of discretizing more complex objects. For more than twenty years, researchers have been developing methods to reduce the time and cost necessary to construct finite element models by computer assistance.

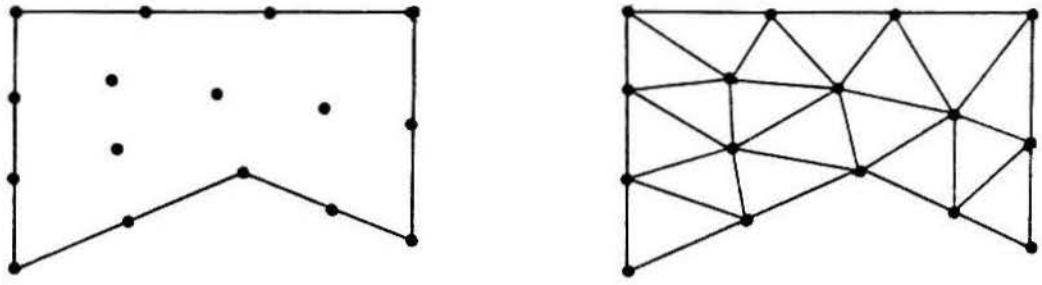
Buell and Bush[1] reviewed mesh generation procedures prior to 1972. Those methods, though rudimentary, have become the foundation of some current meshers. From 1972 to 1980 mesh generation techniques emphasized constructing irregular grids to better model curved boundaries. Thacker[2] has briefly reviewed the different techniques.

Although mesh generation is a very broad and diverse area of research, its general approach can be divided into the following groups:

2.1 NODE GENERATION APPROACH

In this procedure, the interior nodes are generated first. They are then connected to form triangular or quadrilateral elements. This is a very basic method which has been used by many researchers, such as: Nguyen[3], Cavendish, Field and Frey[4], Cendes, Shenton and Shahnasser[5], Sadek[6] and Lo[7]. Such an approach is useful for automatic generation because it is very easy to generate triangular, tetrahedral and quadrilateral elements; however, it is unable to adequately generate the hexahedron elements.

Figure 2.1 shows the basic idea of the node generation approach.



(a) (b)

Figure 2.1 Node generation method: (a) all nodes are generated, then(b) connected to form elements.

2.2 MAPPING APPROACH

The mapping approach begins by developing the mesh in a unit square in a reference coordinate parametric space. It is then mapped to the final coordinate frame by blending functions.

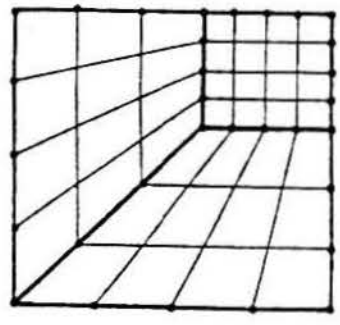


Figure 2.2 Mapping approach

For effective use of this procedure, an arbitrary object must first subdivided into four-sided regions, and then each region is meshed separately. Zienkiewicz and Phillips[8], Cohen[9], Perucchio, Ingraffea and Abel[10], Cook[11] and Gordon and Hall[12] have developed this approach. Figure 2.2 shows a model generate by this mapping technique.

2.3 GRID-BASED APPROACH

This method begins with an infinite grid superimposed on the object. The grid cells that fall outside the object are removed, then the grid cells that intersect the object boundary are adjusted or trimmed to fit the object. Thacker, Gonzalez and Putland[32] and Heighway and Biddlecombe[33] use this approach to create finite element models. Yerry and Shephard[22-24] use a modified quadtree and octree method to generate grids and to develop the final mesh. Figure 2.3 shows the a model generate by grid-based approach.

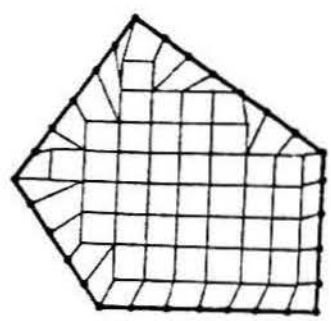


Figure 2.3 Grid-based approach

2.4 DECOMPOSITION APPROACH

The decomposition approach is a technique that continuously subdivides the model, until certain element sizes or limits are reached. Bykat[14], Yeung and Hsu[15], Sluiter and Hansen[16], Sluiter[17], Talbert[18], Woo and Thomasma[20], Moscardini, Lewis, and Cross[21] have used this approach to subdivide a surface to create quadrilateral elements or triangular elements, and subdivided a solid to create tetrahedral elements. An example is given in figure 2.4.

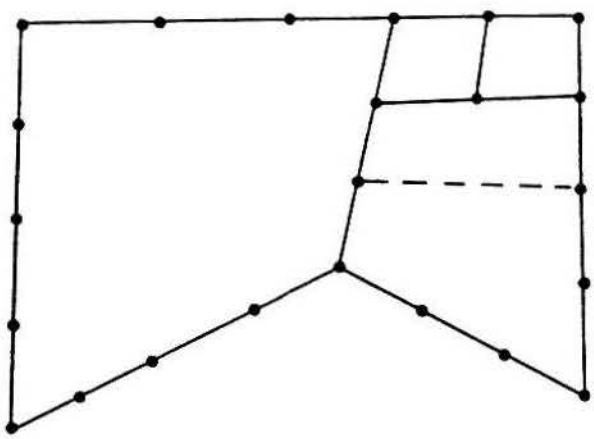


Figure 2.4 Decomposition approach

2.5 DRAG METHOD

For this procedure the basic elements are created in one dimension then "drug" into two dimensions. Similarly, simple elements in two dimensions can be "drug" to generate a three dimensional mesh as discussed by Park and Washam[25]

and Pissanetzky[26]. However this method is useful for symmetric models only, because the size of elements are different inside the solid. Figure 2.5 shows a model created by the drag method.

K Ho-Le[34] gives a more detailed classification of most of the method.

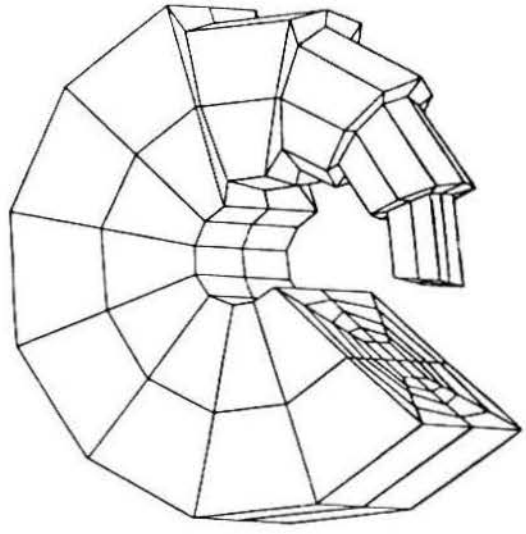


Figure 2.5 Model created by the drag method

2.6 AUTOMATIC MESHING OF THREE DIMENSIONAL SOLIDS

The methods described above generally have the capability to generate either quadratic, or triangular two dimensional elements. Drag method and coordinate blending functions have the capability to generate three dimensional hexahedral elements, however, they are limited to uniform and regular shaped geometries. The other methods can generate only tetrahedral elements.

Although the tetrahedral meshes are not as accurate

numerically in analysis[13], they are ideal for automatic decomposition of complex polyhedra[20]. This is based on the fact that every polyhedron can be divided into an arbitrary number of tetrahedra. Other elements can be formed by combining[19,21] or subdividing[16] tetrahedra, however, the final elements may have very irregular shapes.

A possible subdivision of a tetrahedron into four hexahedral is shown in figure 2.6. The aspect ratio of the elements is still suitable for finite element analysis.

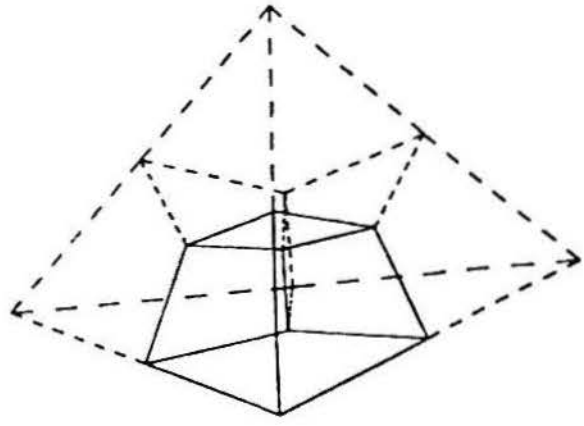


Figure 2.6 Four hexahedra from a tetrahedron

The subdivision method[16], divides the six surfaces to get the surface elements first, then continues the subdivision for the volume to get the final solid elements. This method is very good for generating tetrahedral elements, however problem arise when creating three dimensional hexahedral elements in the volume, i.e. an impossible subdivision case may develop in the final volume.

Figure 2.7 shows this impossible case.

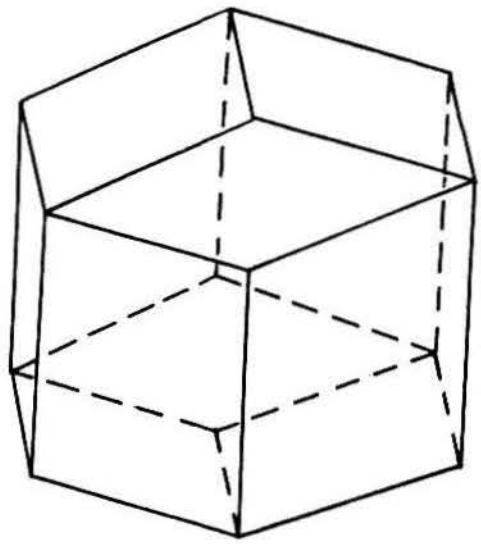


Figure 2.7 Impossible hexahedron elements

Such difficulties in creating hexahedral elements with an automatic procedure is the reason tetrahedral elements are most often used.

Pissanetzky[26] and Park and Washam[25] have successfully implemented the drag method to create hexahedral elements. Now quadralateral elements are first created on a boundary surface then the elements are swept through the region to form solid hexahedral elements. This procedure is a good for regular shapes, however it is poor for irregular shapes. Furthermore, the element size may be different from point to point. Automatic meshing of irregular solid models with hexahedron elements remains a research challenge.

2.7 FREE-FORM DEFORMATION

The contents of this section assumes that the reader is familiar with the Bézier curves and surfaces patches[31].

Free-form deformation is defined in terms of a tensor product trivariate rational Bernstein polynomial[28-31] which takes the following form.

$$P(s,t,u) = \frac{\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} w_{ijk}\, B_i^l(s)\, B_j^m(t)\, B_k^n(u)\, P_{ijk}}{\sum_{i=0}^{l}\sum_{j=0}^{m}\sum_{k=0}^{n} w_{ijk}\, B_i^l(s)\, B_j^m(t)\, B_k^n(u)}$$

where

$$B_i^l(s) = \frac{l!}{i!(l-i)!}(1-s)^{l-i}s^i, \qquad i = 0,1,\ldots,l.$$

$$B_j^m(t) = \frac{m!}{j!(m-j)!}(1-t)^{m-j}t^j, \qquad j = 0,1,\ldots,m.$$

$$B_k^n(u) = \frac{n!}{k!(n-k)!}(1-u)^{n-k}u^k, \qquad k = 0,1,\ldots,n.$$

$$s = \frac{X - Xmin}{Xmin - Xmax}, \quad t = \frac{Y - Ymin}{Xmin - Xmax}, \quad u = \frac{Z - Zmin}{Xmin - Xmax}$$

$B_i^l(s)$, $B_j^m(t)$, and $B_k^n(u)$ are called Bézier blending functions, and s,t,u are the local coordinates of a point with respect to the deformation region. P_{ijk} is the x,y,z coordinates of the displacement control points i,j,k in a solid that form an (m+1) x (n+1) x (l+1) array of points. W_{ijk} is the weight of each control point. M,n,l are the total numbers of the control points to be used in each i,j,k direction. Any node that falls inside the control points boundary box $0 \le s \le 1$, $0 \le t \le 1$, $0 \le u \le 1$ will be calculated according to the Bernstein polynomial and moved to a new

location. Many more important properties of Bézier curves can be found in the article written by Bohm, Farin and Kahmann[31].

The following example explains the basic procedure in creating the free-form deformation model. First consider a simple cuboid flexible plastic mesh model in the space which is to be deformed. Figure 2.8 shows this simple flexible plastic model.

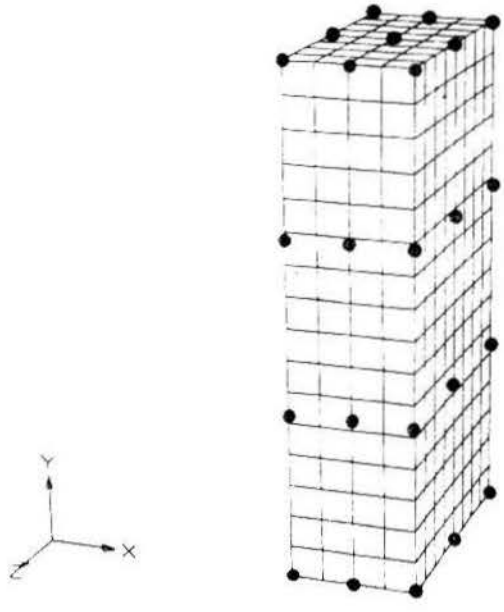


Figure 2.8 Original plastic model with control points

Next create free-form deformation control points (the black dots as shown in the figure 2.8) for the model. The model is now ready for deformation. The next step is to move the control points to new locations as shown in figure 2.9. The flexible plastic model inside the control points has been deformed to the shape as shown in figure 2.10.

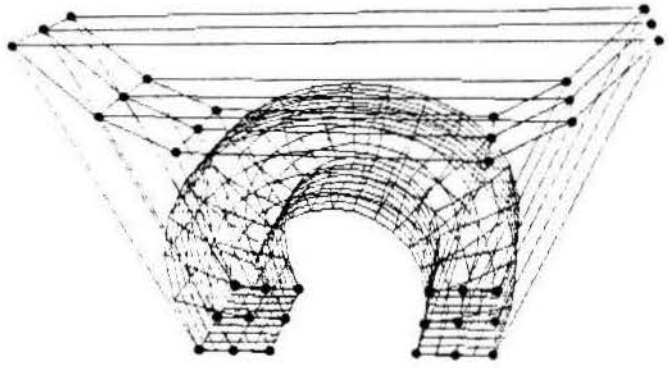


Figure 2.9 Control points in deformed position

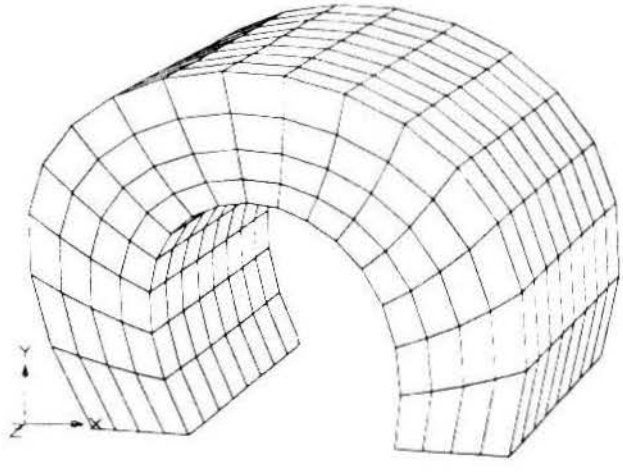


Figure 2.10 Deformed plastic model

2.8 MESH SMOOTHING

Once a solid model has been completely divided into hexahedral elements, the mesh is ready for a finite element

analysis. However, to insure optimum finite element results there is an element shape requirement[27]. A good element should have an aspect ratio, i.e. max-side/min-side, that does not exceed roughly 7 for good displacement results and roughly 3 for good stress results. Generally an element that has an internal angle of less than 20 degrees or greater than 130 degrees is considered inadequate.

Smoothing is used to improve the shapes of the elements in a mesh by moving the interior nodes according to some rule. Effective smoothing provides the following capabilities:

(1) Nearly equal areas or size of adjacent elements,

(2) Gradually changing sizes of elements,

(3) Gradually changing shapes of elements,

(4) Nearly right angle corners in each element.

The Laplacian method is the most commonly used mesh smoothing technique. It changes the coordinates of each mesh node, in turn, to the average of the coordinates of all the immediate neighbors of that node. As shown in figure 2.11, the new position V.

$$V = 1/n \sum_{i=1}^{n} V_i$$

where $V_i = (x_i, y_i)$, i = 1 to n are the directly connected neighbors of this node.

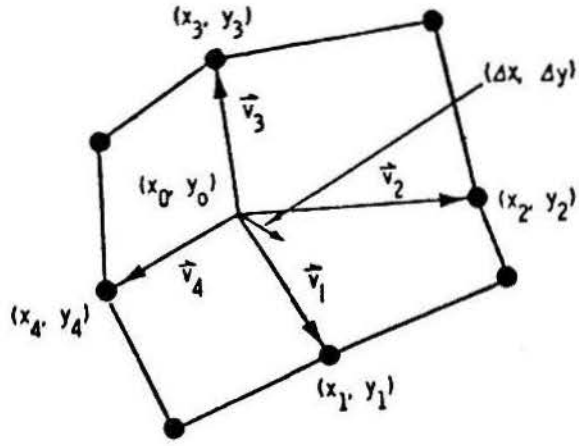


Figure 2.11 Laplacian smoothing

In order to get as close to the same element size as possible, the Laplacian method needs to be modified by using the area of the surrounding element to pull the node.

$$V = \frac{\Sigma\ A_i\ C_i}{\Sigma\ A_i}$$

where A_i is the area of the surrounding i-th elements and C_i is the vector from the corner node to the centroid of that element. The centroid of a element is defined by the average of the coordinates of four corner nodes:

$$C = (1/4 \sum_{i=1}^{4} x_i,\ 1/4 \sum_{i=1}^{4} y_i)$$

where (x_i, y_i), i = 1,2,3,4, are the positions of the corners

of the element. If we set all these A_i's to 1.0, the resulting smoothing technique is similar to Laplacian smoothing. Thus, the method is call "area-weighted Laplacian" or "centroid area pull." Figure 2.12 shows the centroid area pull smoothing method. This centroid area pull method gives better equal size results than the Laplacian method[27].

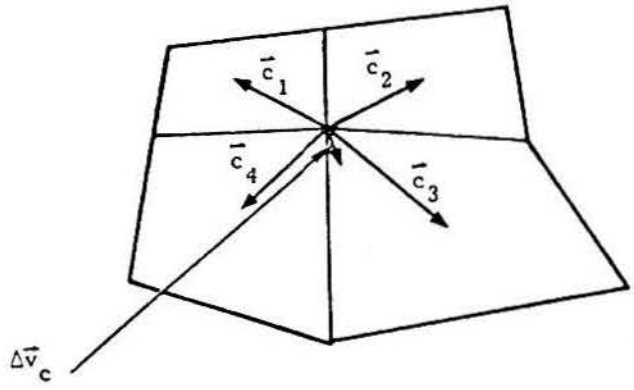


Figure 2.12 Centroid area pull smoothing

3.0 AUTOMATIC MESHING WITH HEXAHEDRON ELEMENTS

Generating an acceptable three dimensional finite element mesh of irregular shapes using hexahedral elements, with only a minimum of human interaction, is the goal of this work. This section describes the method developed by this work to generate such a mesh. The procedure couples both free-form deformation solid modeling and mesh smoothing as described in section 2.0.

The complete automatic meshing procedure consists of several steps. The first step is to create an initial cuboid primitive by eight points in the space. This primitive should consist of six surfaces and twelve lines as shown in figure 3.1. The next step applies free-form deformation to the initial cuboid such that the desired final geometrical shape is created. This step often severely distorts the mesh such that it is not acceptable for finite element calculations as shown in figure 3.2.

To establish an acceptable mesh on the deformed shape, a solid subdivision and mesh transition step is invoked. This step creates hexahedral elements from the solid model. An important aspect of this step is to insure that element continuity is completed. The final step invokes a smoothing process to bring all elements into an acceptable shape.

3.1 INITIAL CUBOID PRIMITIVE

At the beginning of the procedure, a cuboid primitive and a specific element size is defined. This cuboid

primitive is defined in the space by eight corner points. The user can choose any size and orientation of the cuboid primitive he desires, however a cuboid primitive relatively close to the final shape is easier for adjusting the control points of free-form deformation. After the eight points and element size have been chosen the model is ready for geometric deformation. A simple surface mesh is created in figure 3.1 for a view of the primitive model.

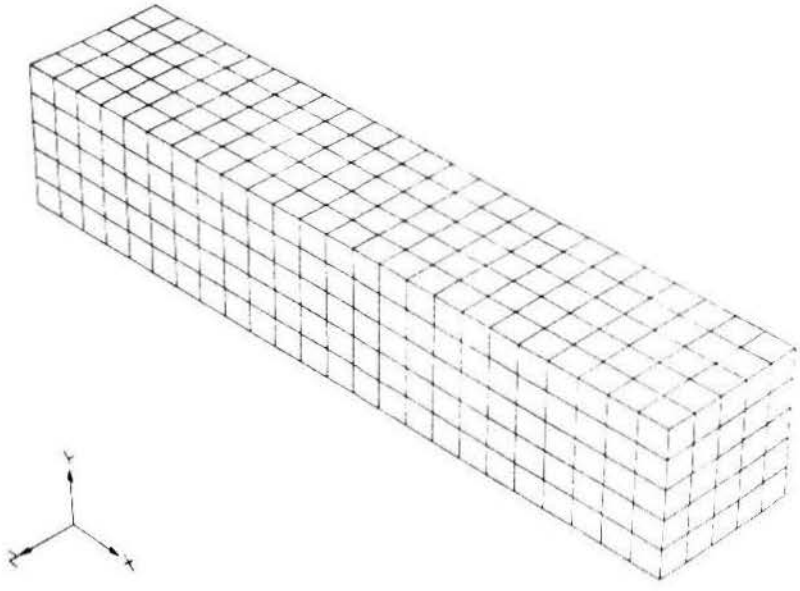


Figure 3.1 Cuboid primitive

3.2 FREE-FORM DEFORMATION

By moving the control points to new locations, as described in section 2 the model is deformed to the shape desired. However, the elements in this deformed model may have been distorted to very high aspect ratios. Figure 3.2

gives such a situation in the mesh which is deformed from the cuboid primitive created in figure 3.1.

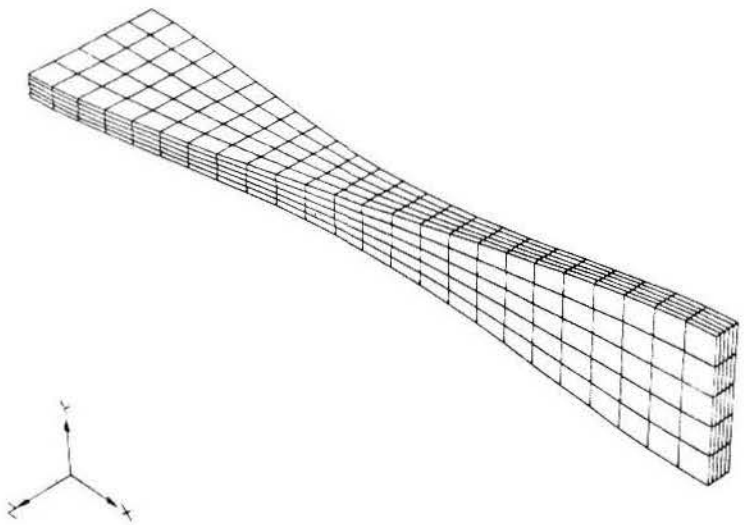


Figure 3.2 A deformed model with high aspect ratio elements

3.3 SOLID SUBDIVISION

The appropriate sized elements are generated by first locating the nodes on each of the boundary edges of the solid to set the initial nodal locations. These locations are chosen such that the element size will be near the size initially set by the user. This is followed by a subdivision process on the solid. This proces is accomplished as follows:

1. Initially dividing the existing solid into two solids.

2. Selecting the number of nodes on the lines that define the joining boundary of these two solids. The number of nodes along this boundary is chosen such that the element

size will be near the size initially set by the user.

3. Force the condition that an even number of nodes exist on the lines that define the joining boundary of the two solids.

4. At this point, all edges of each solid have predefined nodes. Each solid is then checked to see if a) futher subdivision is required, b) a transition sitiuation exists, or c) the meshing is complete.

5. If subdivision is required, return to step 1.

6. If transition is required, proceed to the algorithm defined in Section 3.4.

7. When all subdivisions and transitions are completed, all element have eight nodesm and the mesh is ready for smoothing as described in Section 3.5.

A complete description of the above algorithm is outlined in Reference 36.

3.4 TRANSITION METHOD

If, during the processing by the above defined subdivision method, the number of nodes on one of the edges at one parameter axis equals two, and one or more of the remaining edges is greater than two, a transition region must be created. This means it is necessary to add nodes on the surfaces and inside the volume to create more new elements. In making these transitions, the three cases that may develop are explained below.

1) Transition case 1 - one of the four lines in a

parallel direction contains more than two nodes and the remaining three lines have only two nodes. This region will be processed by adding two nodes on each surface adjacent to the line containing the four nodes. These new nodes are labeled with an A in figure 3.3. Two more nodes, labeled with a B in figure 3.3, are created inside the cuboid to create five hexahedral elements.

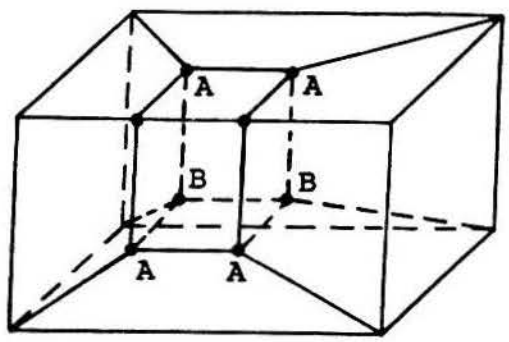


Figure 3.3 Transition case 1

2) Transition case 2 - two adjacent lines in a parallel direction contain more than two nodes and another pair of the lines have only two nodes. Adding two nodes (shown labeled with an A in figure 3.4) on both of the surfaces, creates four hexahedral elements to accommodate this transition.

3) Transition case 3 - three lines in a parallel direction contain more than two nodes and a single line contains two nodes. There is no way to directly accomplish the transition for this case, therefore an indirect method

must be used. This indirect method adds a cube inside the cuboid to create seven elements, as shown in figure 3.5. These new elements can be processed by the previous two methods to produce the eight node hexahedral elements.

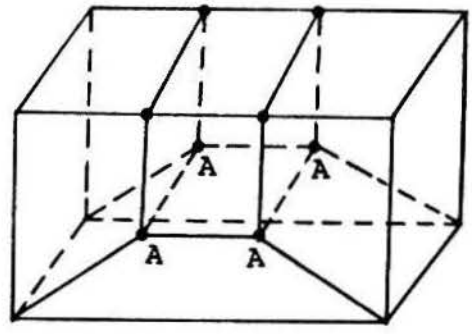


Figure 3.4 Transition case 2

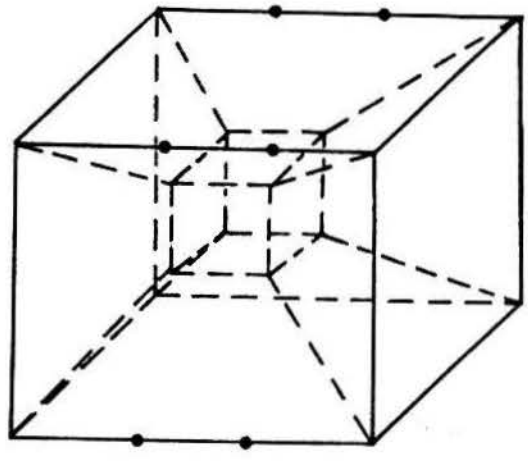


Figure 3.5 Transition case 3

3.5 SMOOTHING PROCEDURE

As described in the previous section, a smoothing procedure is required to provide proper elements for the

finite element analysis. After extensive testing on the samples, the centroid area pull method gave satisfactory results in element size. After running through all the steps of the process, the final result is shown in figure 3.6.

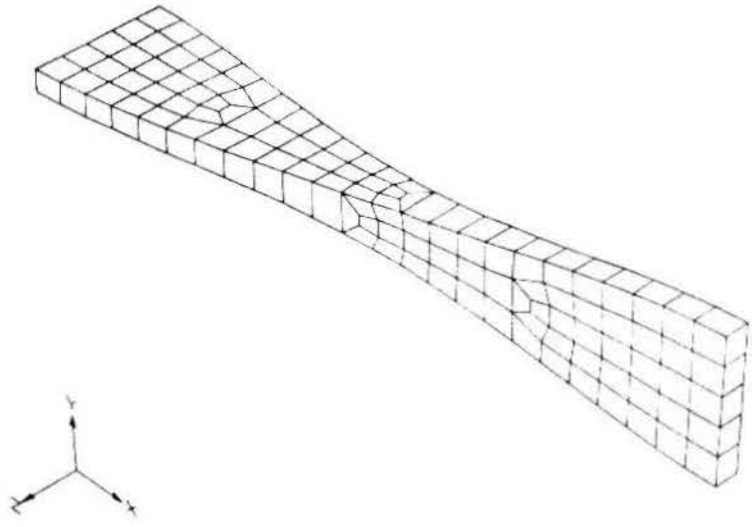


Figure 3.6 Final result of finite element mesh

4.0 EXAMPLES OF AUTOMATIC MESHING

This section presents some examples and provides some explanation for the basic technique for creating the free-form finite element models.

4.1 Transition head

The first example gives a step by step explanation of the capability of this procedure by modeling a part that transforms from a rectangular surface to a circular surface. The initial step defines the cuboid by specifying the information for the eight vertex points and the required size for the elements. This original cuboid with user's defined element size is shown in figure 4.1.

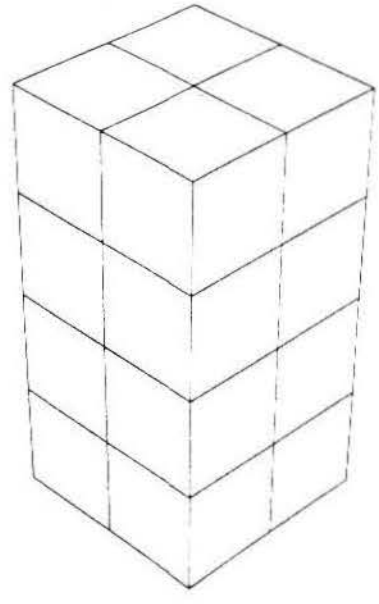
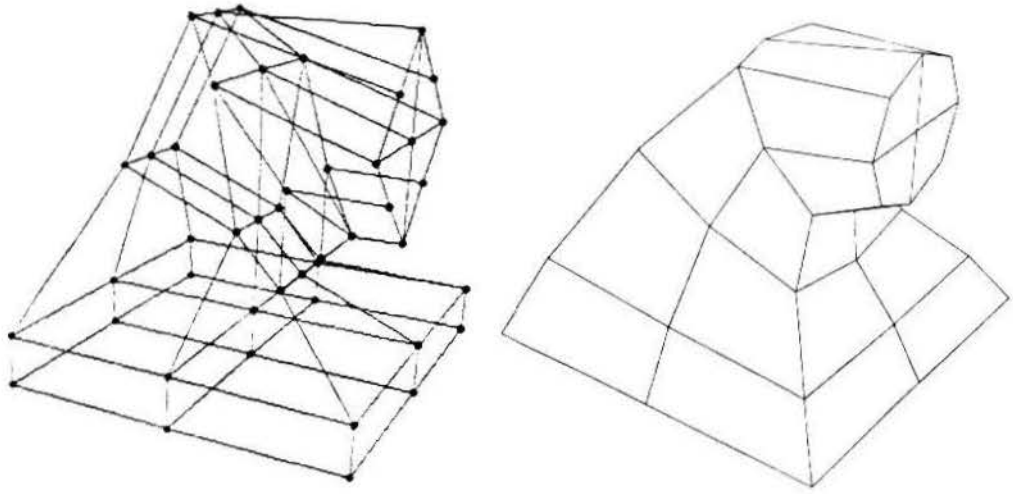


Figure 4.1 Original cuboid mesh

Next the control points for the free-form deformation are input. The location of control points are then adjusted so that all the nodes inside the bounding box form the shape

of the transition region. Figure 4.2 shows the control point net as well as the first deformation of this example.



(a) (b)

Figure 4.2 (a) Control point net and (b) deformation of the transition head

The next step is to use the mesh generation procedure to create the hexahedral elements using the solid subdivision and transition methods. Figure 4.3 shows the results of the finite element mesh generated by this procedure.

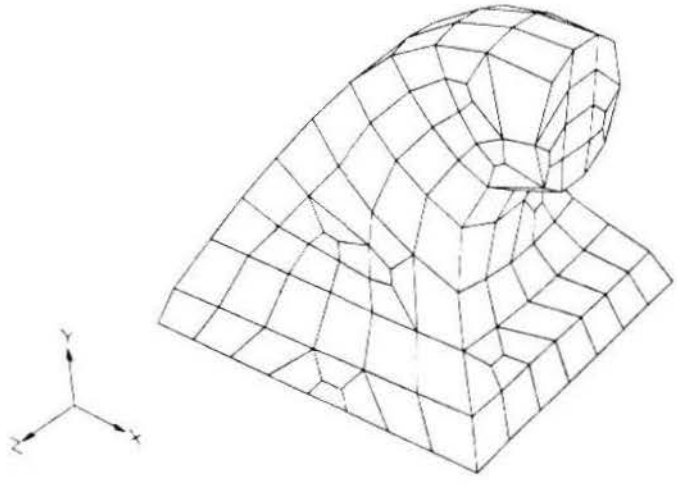


Figure 4.3 Uniform mesh of transition head

After the finite element mesh is generated, the smoothing procedure is applied to readjust the shape of the elements. Figure 4.4 shows the results after the centroid area pull smoothing method has been used.

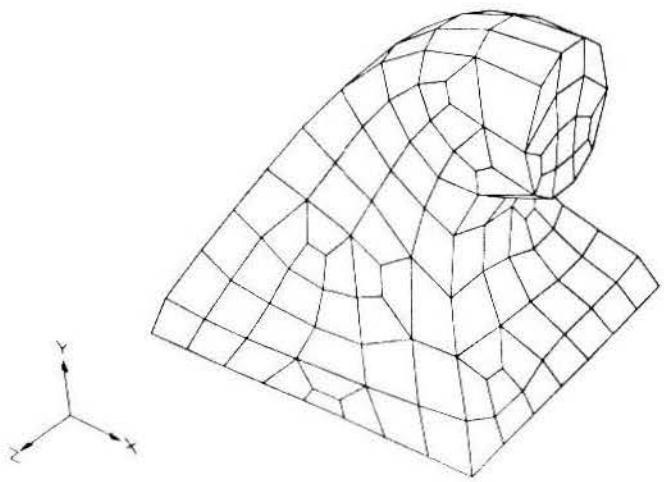


Figure 4.4 Smoothed mesh of transition head

4.2 Hour glass model

This example demonstrates the capability to create uniformly sized elements in a model with large dimensional variations. As before, the initial model begins from the regular cuboid similar to that shown in figure 4.1. Figure 4.5 shows the hour glass model after free-form deformation. The sizes of the elements at the middle of the model are now significantly different from those elements on both ends. These middle elements have very high aspect ratios which are obviously not well formed for finite element analysis.

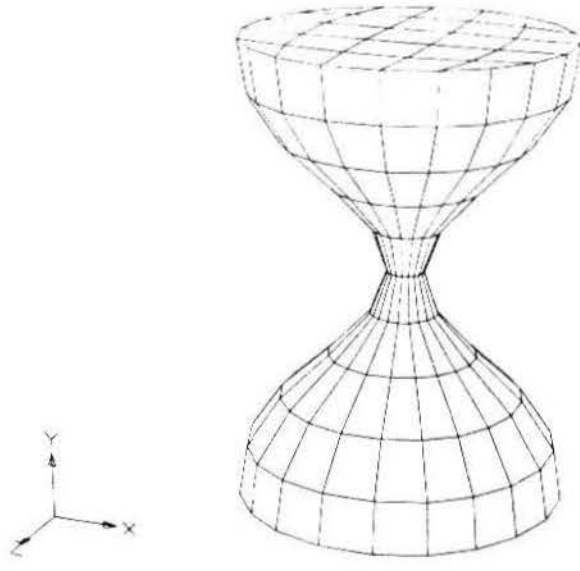


Figure 4.5 Uneven element size of hour glass model

Figure 4.6 shows the results of the model after the subdivision and transition is applied. The elements at the middle of the model are becomming uniform in size and have better aspect ratios. Finally the smoothed result is shown in figure 4.7.

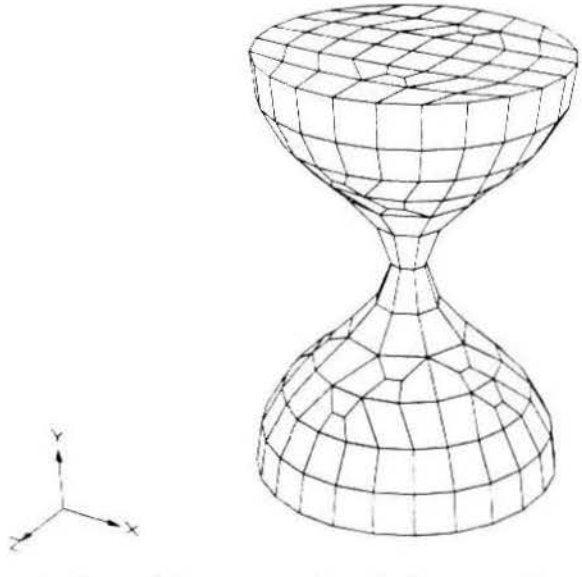


Figure 4.6 Uniform mesh of hour glass model

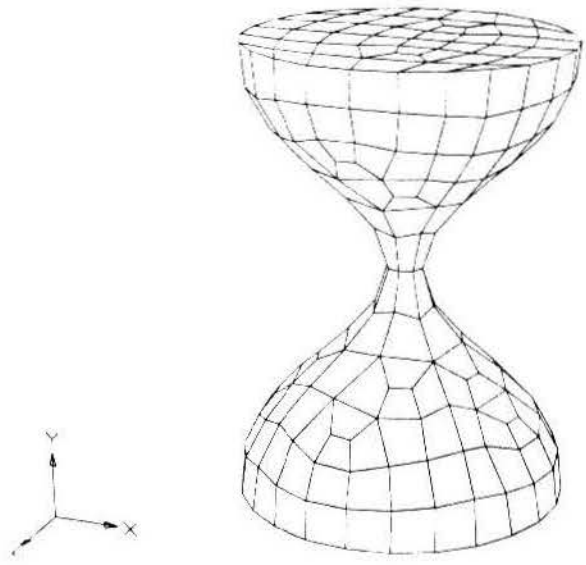


Figure 4.7 Smoothed elements of hour glass model

4.3 Hammer head

A significant capability of the free-form deformation procedure is the ability to generate arbitrary shapes. The hammer head model created here uses 2x2x9 degrees of free-form deformation to define the shape. Figure 4.8 shows the original mesh of the model.

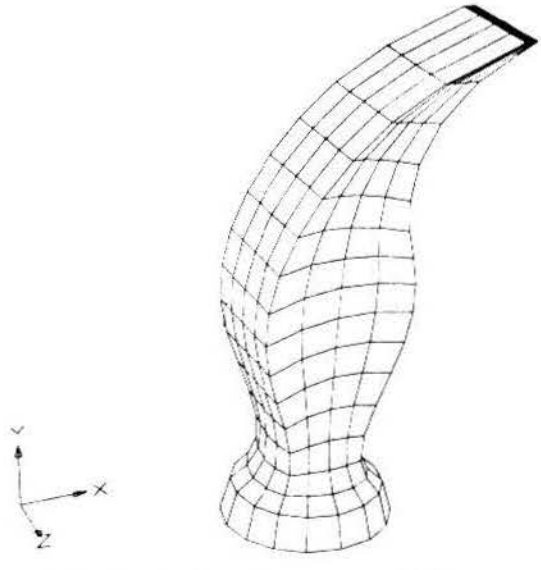


Figure 4.8 Original mesh of hammer model

Note that the model is not symmetric, therefore it is tedious for a traditional geometry modeler to create a model like this one. Figure 4.9 shows a mesh of the object.

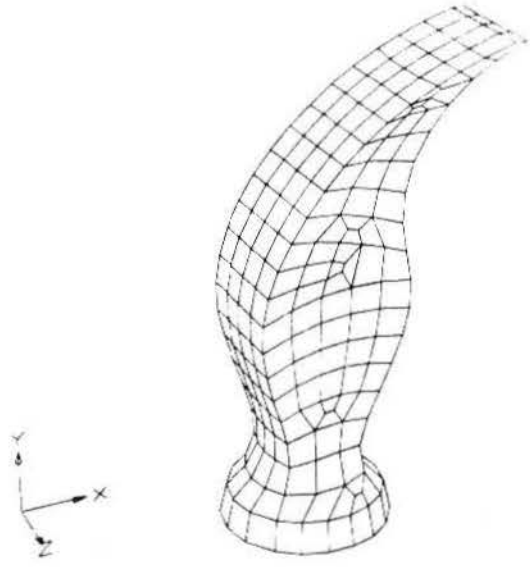


Figure 4.9 Mesh result of hammer head

5.0 CONCLUSIONS

The free-form deformation method is a very powerful tool to use in creating solid models of complex objects. Hexahedral (i.e. brick) elements are generally the element of choice for performing three dimensional finite element analysis. The work presented here couples the ability to create complex solid models using the free-form deformation method and the generation of acceptable finite elements for analysis purposes. The method is automated because a user is required to provide very little information to the algorithm that defines the procedure.

The conclusions that can be drawn from this work are:

1. The free-form deformation solid generation procedure can create solid geometric models appropriat for hexahedral finite element modeling.
2. A subdivision method that automatically generates hexahedral elements is appropriate to initially create elements within the deformed solid.
3. A transition scheme is necessary to allow the elements to effectively model sever changes in geometry.
4. Centroid area pull smoothing is adequate to properly reform the elements into acceptable shapes for finite element analysis.
5. Proper use of the method developed here will allow very complex finite element models of solid objects to be generated with a minimum of input from a user.

BIBLIOGRAPHY

1. Buell, W. R. and Bush, B. A. "Mesh generation - a survey," Transactions of the ASME Journal of Engineering for Industry, pp. 332-338, February 1973.

2. Thacker, W. C. "A Brief Review of Techniques for Generating Irregular Computational Grids," International Journal for Numerical Methods in Engineering, Vol. 15, pp. 1335-1341, 1980.

3. Nguyen, V. P. "Automatic Mesh Generation with Tetrahedron Elements," International Journal for Numerical Methods in Engineering, Vol. 18, pp. 273-280, 1982.

4. Cavendish, James C., Field, David A. and Frey, William H. "An Approach to Automatic Three-Dimensional Finite Element Mesh Generation," International Journal for Numerical Methods in Engineering, Vol. 21, pp. 329-347, 1985.

5. Cendes, Z. J., Shenton, D. and Shahnasser, H. "Three-Dimensional Finite Element Mesh Generation Using Delaunay Tesselation," IEEE Trans. Mag. Vol MAG-21 No. 6, pp. 2535-2538, Nov. 1985.

6. Sadek, E. "A Scheme for The Automatic Generation of Triangular Finite Elements," International Journal for Numerical Methods in Engineering, Vol. 15, pp. 1813-1822, 1980.

7. Lo, S. H. "A New Mesh Generation Scheme for Arbitrary Planar Domains" International Journal for Numerical Methods in Engineering, Vol. 21, pp. 1403-1426, 1985.

8. Zienkiewicz, O. C. and Phillips, D. V. "An Automatic Mesh Generation Scheme for Plane and Curved Surfaces by 'Isoparametric' Coordinates," International Journal for Numerical Methods in Engineering, Vol. 3, pp. 519-528, 1971.

9. Cohen, H. D. "A Method for The Automatic Generation of Triangular Elements on a Surface," International Journal for Numerical Methods in Engineering, Vol. 15, pp. 470-476, 1980.

10. Perucchio, Renato and Ingraffea, Anthony R. and Abel, John F. "Interactive Computer Graphic Preprocessing for Three-Dimensional Finite Element Analysis," International Journal for Numerical Methods in Engineering, Vol. 18, pp. 907-926, 1982.

11. Cook, W. A. "Body Oriented(natural) Coordinates for Generating Three-Dimensional Meshes," International Journal for Numerical Methods in Engineering, Vol. 8, pp. 27-43, 1974.

12. Gordon, W. J., and Hall, C. A. "Construction of Curvilinear Co-Ordinate System and Applications to Mesh Generation," International Journal for Numerical Methods in Engineering, Vol. 7, pp. 461-477, 1973.

13. Wördenweber, B. "Finite element mesh generation," Computer-Aided Design, Vol. 16, pp. 285-291, Sept. 1984.

14. Bykat, A. "Design of a recursive, shape controlling mesh generator," International Journal for Numerical Methods in Engineering, Vol. 19, pp.1375-1390, 1983.

15. Yeung, S. F. and Hsu, M. B. "A Mesh Generation Method Based on Set Theory," Computer & Structure Vol. 3, pp. 1063-1077, 1973.

16. Sluiter, M. L. C. and Hansen, D. L. "A General Purpose Automatic Mesh Generator For Shell and Solid Finite Elements," Computer in Engineering, Proceeding of the 2nd International Computer Engineering Conference held in San Diego, California, August 15-19, 1982.

17. Sluiter, M. L. C, " Automatic meshing with quadrilateral elements," Lecture notes from a presentation at Brigham Young University, pp. 1-28, 1982.

18. Talbert, J. A. "Development of A Two-Dimensional Finite Element Mesh Generator with Automatic Mesh Refinement and Bézier Curve Boundary Definition," Thesis 1987.

19. Heighway, E. A. "A Mesh Generator for Automatically Subdividing Irregular Polygons Into Quadrilaterals" IEEE Trans. Mag. Vol MAG-19, No. 6, pp. 2535-2538, Nov 1983.

20. Woo, T. C. and Thomasma, T. "An Algorithm for Generating Solid Elements in Object with Holes," Computers and Structures, Vol. 18, pp. 33-342, 1984.

21. Moscardini, A. O., Lewis, B. A. and Cross, M. "AGOTHOM-Automatic Generation of Triangular and Higher Order Meshes," International Journal for Numerical Methods in Engineering, Vol. 19, pp. 1331-1353, 1983.

22. Yerry, M. A., and Shephard, M. S. "A Modified Quadtree Approach to Finite Element Mesh Generation," IEEE Computer Graphics and Applications, pp. 39-46, 1983.

23. Yerry, M. A., and Shephard, M. S. "Automatic Three Dimensional Mesh Generation by Modified-Octree Technique," International Journal for Numerical Methods in Engineering, Vol. 20, pp. 1965-1990, 1984.

24. Yerry, M. A., and Shephard, M. S. "Automatic Mesh Generation for Three-Dimensional Solids," Computers and Structures, Vol. 20, pp. 31-39, 1985.

25. Park, S. and Washam, C. J. "Drag Method as A Finite Element A Mesh Generation Scheme," Computer and Structures, Vol. 10, pp. 343-346, 1979.

26. Pissanetzky, Sergio, "Kubik: An Automatic Three-Dimensional Finite Element Mesh Generator," International Journal for Numerical Methods in Engineering, Vol. 17, pp.255-269 1981.

27. Jones, R. E. "Qmesh: A Self-organizing Mesh Generation Program," SLA-73-1088, Applied Mathematics divisions 2642, Sandia Labs, pp. 1-60, 1974.

28. Sederberg, T. W. "Computer Aided Geometric Design," CE694R Class Notes, Fall Semester 1986.

29. Sederberg, T. W. and Parry, S. R. "Free-Form Deformation of Solid Geometric Models," SIGGRAPH Conference Proceedings, August 1986.

30. Parry, S. R. "Free-Form Deformations in a Constructive Solid Geometry Modeling System," Dissertation 1986.

31. Bohm, W., Farin, G. and Kahmann, J. "A survey of curve and surface methods in CAGD," Computer Aided Geometric Design 1,1. pp. 1-60, July 1984.

32. Heighway, E. A. and Biddlecombe, C. S. "Two Dimensional Automatic Triangular Mesh Generation for Finite Element Electromagnetic Package PE2D," IEEE Trans. on Magnetic, MAG-8, No. 1, pp.594-598, March 1982.

33. Thacker, W. C., Gonzalez, A. and Putland, G. E. "A method for automating the construction of irregular computational grids for storm surge forecast models," Journal of computational Physics pp. 371-387, Vol. 37, 1980.

34. K Ho-Le "Finite element mesh generation methods: a review and classification" Computer-Aided Design pp. 17-38, Jan. 1989.

35. Piegl, L. "A Generalization of the Bernstein-Bézier

Method," Computer Aid Design, Vol. 16, pp. 209-215, July 1984.

36. Hsu, Ssuta "Automatic meshing of free-form Deformation Solids," Ph.D. Dissertation, Brigham Young University, Provo, Utah April 1988.

37. Kimura, Fumihiko, "Geomap-III: Designing Solids with Free-Form Surfaces," IEEE Computer Graphics and Applications, pp. 58-71, June 1984.

APPENDICES

APPENDIX A

RESULTS ANALYSIS AND FUTURE DEVELOPMENT

RESULTS ANALYSIS

Because this mesh generator is supported by free-form deformation (whenever a subdivision line is created it needs to be deformed first to obtain the final element size), the processing costs are high.

Figure A.1 shows the smoothed result of the hammer head model. It can be seen that the model near the bottom of the hammer is squeezed in. This is because the smoothing processor relocated the nodes to an unsymetric location on the surface. It can be solved by increasing the number of elements in the model to gradually change the surface in that area, or by creating two models separatly at that critical area and by using the G2 continuity technique[28] to connect them into a continuous model.

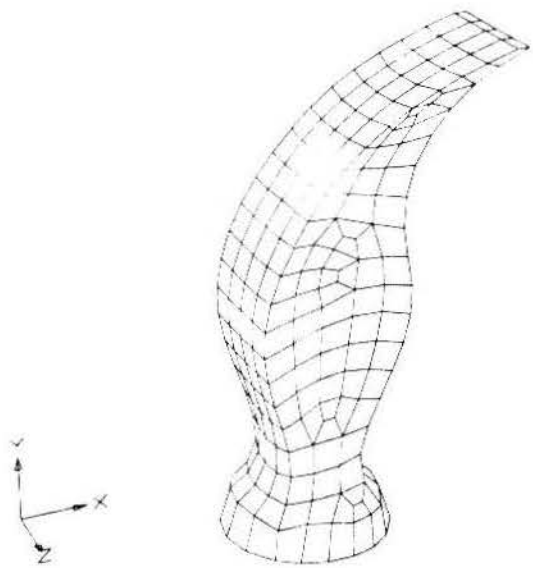


Figure A.1 Hammer model after smoothing

FUTURE DEVELOPMENT

The free-form deformation technique used in this research is based on the Bézier method which is a global control type of deformation. Therefore, it is easy to create the global smooth models, but models with irregular shapes requiring many local deformations (and therefore many control points) are difficult to do. future refinements can be made to enhance its capabilities.

1. The development of free-form deformation with the B-spline base[35] will better handle irregular shape.

2. The improvement of the subdivision processor to access many local deformations would also increase performance on handle irregular shapes.

3. Developing the variable element densities in certain locations may meet the requirements for the change of different local stresses or stains and produce better results in finite element analysis. This development can also be used to solve some uneven shapes.

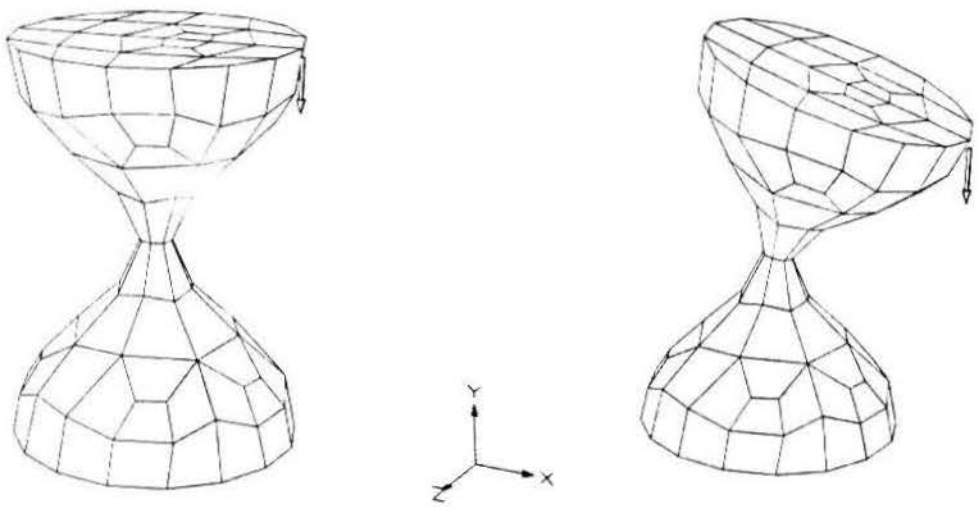
CONNECTION WITH MOVIESTAR

MOVIESTAR, which was developed by Civil Engineering Department, is a very powerful pre/post processor which is connected with some very famous finite element software such as: ABAQUS, NASTRAN and MARC. A connection with MOVIESTAR would be a final destination for this automatic free-form deformation mesher.

At this time only a CAEDOS neutral file is created for

MOVIESTAR to do the pre-processing with the finite element software. Connecting directly with the MOVIESTAR database can make the best usage of the model created by this free-form deformation mesher. The meshed models can be run through the OPTIMIZER in MOVIESTAR to concatenate with other models into a continuous consolidated model for best simulation of the real object.

An ABAQUS finite element analysis result is shown in figure A.2, it can be proven that this automatic mesh generator is working properly in modeling and meshing.



(a) (b)

Figure A.2 A finite element analysis result (a) before testing, and (b) after testing.

APPENDIX B

COMPLETE DISCUSSION OF ALGORITHM

PREPARE ANALYSIS

Before running the program two files need to be prepared: a cuboid data file, and a free-form deformation control points file. All the input files need to be prepared with the same name but with different extensions. The cuboid data file is created with the extension *.sol; it contains the user defined length of the elements and the coordinates of the eight corner points. The free-form deformation file needs to be created with extension *.ffd; it contains the relative movments of all the control points and their weights.

The mesh program will begin by reading the cuboid data file and free-form deformation file. A geometry model will then be created by the defined control points. At this time a surface deformation file with the extension *.def is created for MOVIE.BYU to view the picture. The nodes on all the edges of the geometry will then be readjusted to have uniform length all around the edges. This nodal information will then be mapped back to the original cuboid as a reference for the subdivision process.

AUTOMATIC MESHING

After all the new boundary nodal information is defined and stored, the automatic meshing is ready to process for hexahedral elements. The steps of the entire algorithm are described as below:

1. A flag check is performed to check the number of

nodes on the edges of the solid to decide which processing step to chose next. For each of i,j,k direction, if the number of nodes on the edges of a direction are all greater than two, which means this solid can be subdivided, then the subdivision is performed. If the number of nodes on all the edges are all equal two then the subdivision process is done, and hexahedral element is created. If the number of the nodes on one of the edges is equal two and at least one of the rest of the edge has the number of nodes greater than two then there is a transition required to break the solid down into several hexahedral elements.

2. In cases where a subdivision is required, the subdivision priority is set to j-direction, then the k-direction, and last the i-direction. The subdivision starts with the surface containing the edge which has the least number of nodes, by defining a sudivision line between two opposite edges on the surface to be the best line for division. While looking for the best subdivision line, a search for the existing old line between those two edges needs to be performed first. If there is an old line existing then use it, otherwise find the shortest distance to be the best line and store it for next line check. The purpose of checking the existing old lines is to ensure continuity between two adjacent elements.

3. This subdivision line then passes through the free-form deformation to determine the real length in the

final state, and new nodes are then added to this line to create the length for the elements. A check for en even number of nodes on the surface is required to creating hexahedral elements. If an odd number appears on the surface then one more node need to be added to that subdivision line to make it even.

4. After one surface is divided then the node on that edge will be used for the adjacent surface. The subdivision process will continue until three surfaces have been processed. The last surface will be divided by using the nodes passed in through both opposite edges, and without checking the best sudivision line.

5. After four surfaces in a parametric direction have been divided, two solids are then created and pushed into a stack for further processing.

6. In case a transition is required, then it is performed according to the transition rules explained in section three and stored in the stack for further processing.

7. While performing a transition procedure a check for old lines is also necessary to ensure continuity with adjacent elements.

8. After a subdivision or transition is performed the movment will then return to step 1 for the next appropriate processing. All the processes will then repeat until the nodes on each edge are equal to two, then a linear

hexahedral element is created.

9. After all the elements are created, smoothing will then be perfomed to readjust the nodes so they will have better aspect ratios for the elements.

10. Three files which will be created at the end of the program under the same name but with different extensions (*.geo, *.mov, *.nav) for further analysis.

APPENDIX C

CONTINUITY OF ELEMENTS

To insure that adjacent elements have common nodes, the order of subdivision and the transition process is critical. Figure C.1 shows the problem of such a discontinuity between two adjacent elements. This possibility can be avoided by checking the subdivision order.

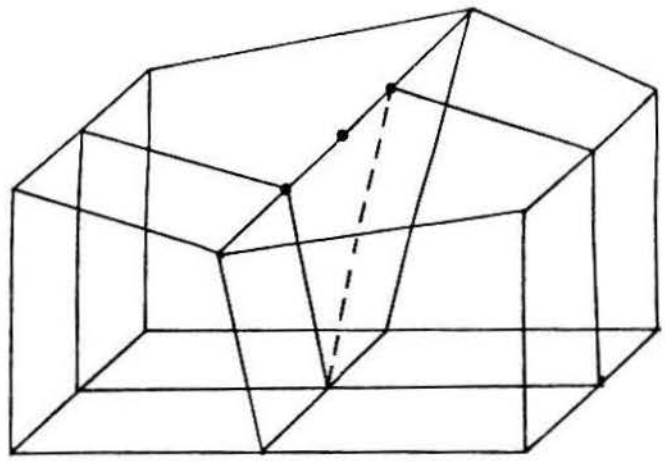


Figure C.1 Discontinuous surface elements

1. Checking old lines: While the new lines and nodes are created, the old information needs to be stored in a reference area. Before a new line is going to be created first check this new line with the old reference lines to see if this new line has already been created. Since a surface can only be used by just two solids, the line on that common surface can only be used twice. Therefore, if this line was an existing line, then use the old line data for this new subdivision line and delete that old line data from the reference area. If this new line was not an old

line then create it as a brand new line and store its information for the next reference line check.

2. Checking priorities: The order of subdivision priority must be set. If the direction of all the subdivisions are available at the same time then a processing order must be established. In this algorithm the j direction had been set to the first priority, k direction set to the second priority, and i direction set to the last priority. So after the meshing was done and ready to plot, the element at the bottom layer will be plotted first, and plotted from left to right as defined in the inverse of that mesh priority order.

For the case of transition 1 and 2 an old reference lines bucket needs to be set up for checking. For the case of priority in transition, an area need to be set up too, however it is a little different from the subdivision situation. The transition case 3 had been set to the first priority so that whenever a complicated case occurs it can process this case into the simple transition casees for further processing.

The transition case 2 had been set to the second priority and several conditions need to be met to qualify this case. Figure C.2 shows the example of the transition priority. The k direction transition seems to have the higher priority, however if this k direction transition had been processed first, as shows on the dashed line, then it

will cause discontinuity of the elements in the i direction on the top surface. Therefore, the i direction transition must be processed first.

Finally the transition case 1 has been set for last priority, because there is only one line to be processed.

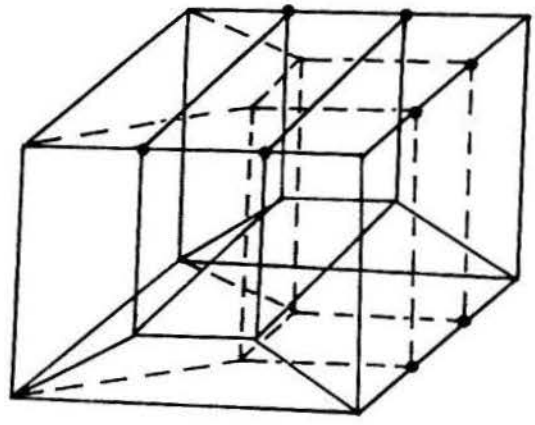


Figure C.2 Transition priorities

APPENDIX D

BÉZIER THEORY

(As presented by Professor T. S. Sederberg)

BÉZIER CURVES

Bézier curves are named after their inventor, Dr. Pierre Bézier. Bézier was an engineer with the Renault car company and set out in the early 1960's to develop a curve formulation which would lend itself to shape design[28].

Bézier curves can be understood by thinking in terms of the center of mass of a set of point masses. For example, consider the four masses m_0, m_1, m_2, m_3 located at points P_0, P_1, P_2, P_3 (Figure D.1).

The center of mass of these four point masses is given by the equation

$$P = \frac{m_0P_0 + m_1P_1 + m_2P_2 + m_3P_3}{m_0 + m_1 + m_2 + m_3}$$

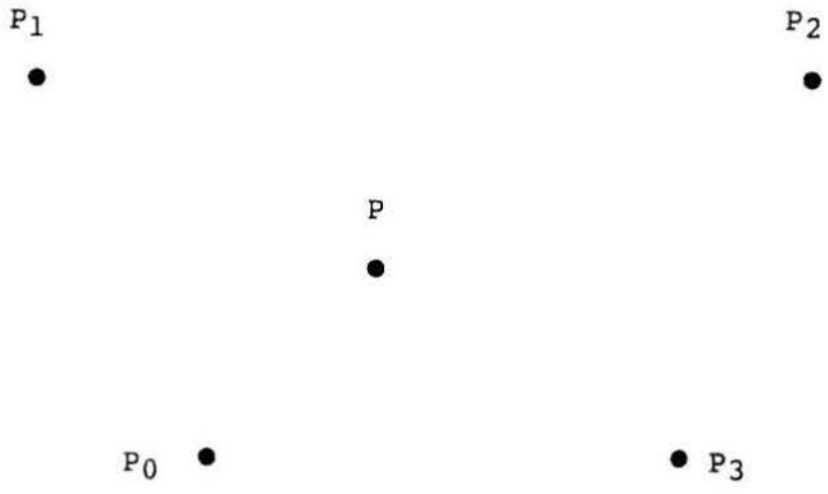


Figure D.1 Center of mass P, in four points masses.

Now, instead of having a fixed constant value of the masses at each point, each mass varies as a function of some

parameter t. Specifically let:

$$m_0 = (1-t)^3, \quad m_1 = 3t(1-t)^2, \quad m_2 = 3t^2(1-t), \quad m_3 = t^3.$$

Then the value of these masses as a function of t is shown in figure D.2.

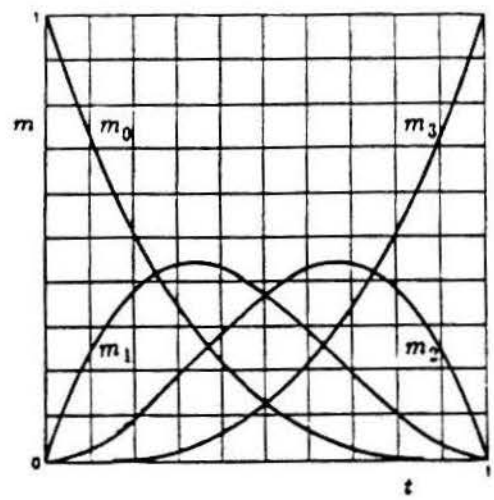


Figure D.2 Cubic Bézier blending functions

As t varies between 0 and 1, assume the masses have different weights, then their center of mass changes continuously. And a curve is swept out by the center of masses. This curve is called cubic Bézier curve, because the mass equations are cubic polynomials in t. For any value of t, $m_0 + m_1 + m_2 + m_3 \equiv 1$ the equation of this Bézier curve can be written as

$$P = m_0P_0 + m_1P_1 + m_2P_2 + m_3P_3.$$

Note that when t=0, $m_0=1$ and $m_1=m_2=m_3=0$ The curve is forced to pass through P_0. Also, when t=1, $m_3=1$ and $m_0=m_1=m_2=0$, the curve passes through point P_3. Furthermore, the curve is tangent to P_0-P_1 and P_3-P_2. These properties make Bézier

curve an intuitively meaningful way to describe free-form shapes.

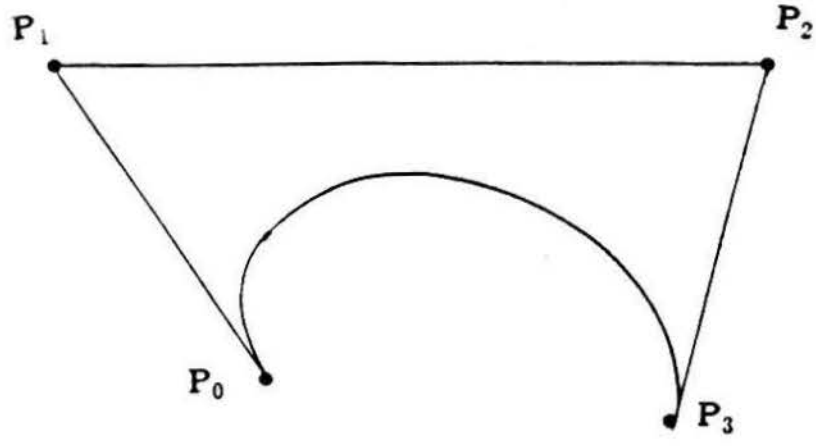


Figure D.3 Cubic Bézier curve

Some examples of cubic Bézier curves which illustrate these properties are given in figure D.4.

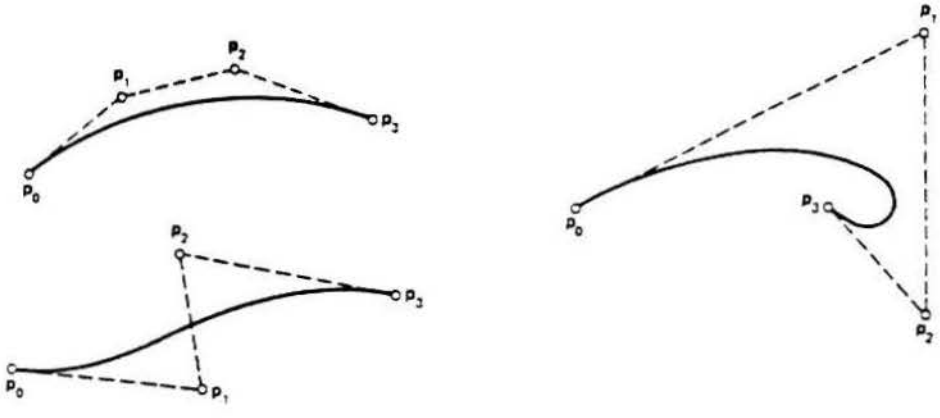


Figure D.4 Example of Bézier curves.

These variable masses m_i are normally called blending

functions and their locations P_i are known as control points or Bézier points. These blending functions, in the case of Bézier curves, are known as Bernstein polynomials.

Bézier curves of any degree can be defined. A Bézier curve of degree n will have n+1 control points whose blending functions are given as $B_i^n(t)$

$$B_i^n(t) = \frac{n!}{i!(n-i)!}(1-t)^{n-i}t^i, \qquad i = 0,1,\ldots,n.$$

For example, let n=3 then $m_0=B_0^3=(1-t)^3$, $m_1=B_1^3=3t(1-t)^3$, $m_2=B_2^3=3t^2(1-t)^3$ and $m_3=B_3^3=t^3$. $B_i^n(t)$ is also referred to as the ith Bernstein polynomial of degree n. Many important proteries of Bézier curves can be found in the article by Bohm, Farin and Kahmann[31].

SUBDIVISION METHOD

The most important and fundamental algorithm for dealing with Bézier curves is the subdivision algorithm. This was devised in 1950 by F. de Casteljau and is sometimes referred to as the de Casteljau algorithm. It is also know as the geometric construction algorithm.

As a Bézier curve is defined for parameter values between 0 and 1. It is often useful to be able to subdivide this curve into two new Bézier curves, one of them in the range $0 \le t \le \alpha$, and the other in the range $\alpha \le t \le 1$. These two curves are considered to be together and equivalent to the single original curve from which they were derived. The control points for these two new curves are found using a

simple geometric rule. We illustrate with a cubic Bézier curve in figure B.5.

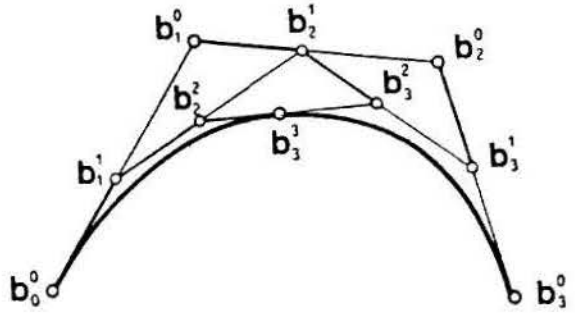


Figure D.5 Subdividing a cubic Bézier curve.

Let the original four control points be denoted b_0^0, b_1^0, b_2^0, b_3^0. The control polygon consists of three lines connected between four control points. ã is the parameter value at which we wish to split the curve, then more auxiliary points have been created between the original control points. The auxiliary points are located at the distance ã of the previous control points on the control lines. The recurrence relation of the control points can be expressed by the formula:

$$b_i^k(\alpha) = (1-\alpha)b_{i-1}^{k-1} + \alpha b_i^{k-1}$$

Now, two new Bézier curves are defined from these auxiliary points. For the left portion where $0 \le t \le \alpha$, there exist control points b_0^0, b_1^1, b_2^2, and b_3^3. The right portion where $\alpha \le t \le 1$ has control points b_3^3, b_3^2, b_3^1, and b_3^0. These two

Béziers can be repeated subdivided until small enough, then the Bézier curve points are defined.

The Bézier curves which have been defined above are known as integeral Bézier curves. Since it is essential for the blending function to sum to one (otherwise the curve will change with the coordinate system) the blending function must be normalized bu dividing through their total value.

$$X(t) = \frac{\sum_{i=0}^{n} w_i B_i^n(t) P_i}{\sum_{i=0}^{n} w_i B_i^n(t)}$$

The curve is called a rational Bézier curve, because the blending functions are rational polynomials. The important reason for rational Bézier curves is that they can define a curves without using higher order control points. Figure D.6 shows the change of weights in the control points to influence the shape of the curve.

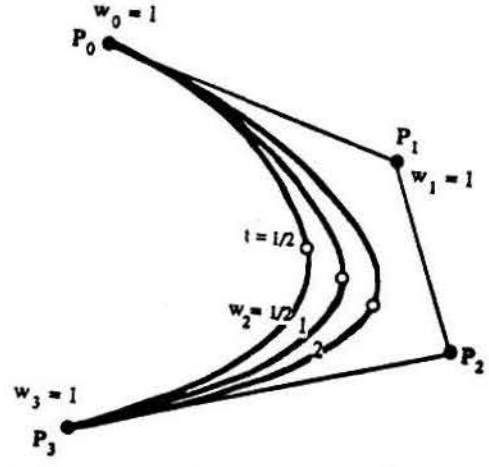
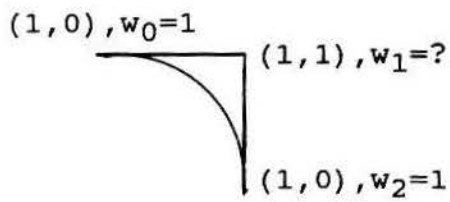


Figure D.6 Change the curve shape by the weights

The following example shows the way to create a circle by two method using the Bézier curve. To create a quarter circle from a line by degree two control points, the weight control method[28] needs to be used. In the following figure let a rational quadratic degree two Bézier curve was defined as:



$$x(t) = \frac{w_0x_0(1-t)^2 + 2w_1x_1t(1-t) + w_2x_2t^2}{w_0(1-t)^2 + 2w_1t(1-t) + w_2t^2}$$

$$y(t) = \frac{w_0y_0(1-t)^2 + 2w_1y_1t(1-t) + w_2y_2t^2}{w_0(1-t)^2 + 2w_1t(1-t) + w_2t^2}$$

In order to satisfied a circular equation, $x(t)^2 + y(t)^2 = 1$, we substitute x and y into the equation to solve for the weight at w_1, and then the weight for the middle point $w_1=\sqrt{2}/2$. Then this value will be used as a control point weight at that middle point. After the free-form deformation this line will become a quarter circle as defined.

Additionally, there is a fixed control points weight method can be used. By increasing the degree of the Bézier curve and adjusting the location of some control points, one can get the same quarter circle. Let a fourth degree Bézier curve have the same weight all around the control points,

$w_0=w_1=w_2=w_3=w_4=1$ and set $P_0=(0,1)$, $P_1=(0.4,1)$, $P_2=(0.8,0.8)$, $P_3=(1,0.4)$, $P_4=(1,0)$ the resulting curve will be exactly the same as the second degree curve has different weights on each control point.

BÉZIER SURFACE

After the Bézier curve was defined, this curve can be expand into a surface. Points on a Bézier surface are given by an extension of the general equation for points on a Bézier curve.

$$P(s,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij} B_i^m(s) B_j^n(t) \qquad s,t \in [0,1]$$

Where P_{ij} are control points of the Bézier surface that form an (m+1) x (n+1) rectangular array of points. By using the binomial representation of the cubic tensor product, the Bézier surface patch takes the form:

$$P(s,t) = [s^3 \; 3s^2(1-s) \; 3s(1-s)^2 \; (1-s)^3] \; P \begin{bmatrix} t^3 \\ 3t^2(1-t) \\ 3t(1-t)^2 \\ (1-t)^3 \end{bmatrix}$$

where $$P = \begin{bmatrix} P_{33} & P_{23} & P_{13} & P_{03} \\ P_{32} & P_{22} & P_{12} & P_{02} \\ P_{31} & P_{21} & P_{11} & P_{01} \\ P_{30} & P_{20} & P_{10} & P_{00} \end{bmatrix}$$

The matrix P contains the position of the control points for Bézier surface. In the Bézier surface only the four corner points where s=0, s=1, t=0, t=1 are lie on the patch. Figure D.7 shows the bicubic Bézier surface.

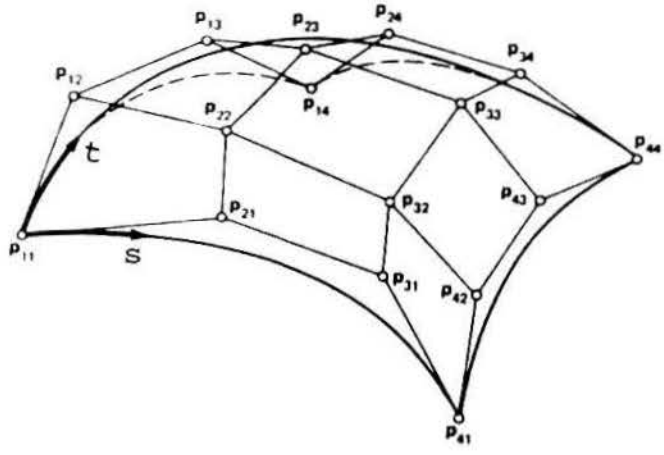


Figure D.7 Bicubic Bézier surface

APPENDIX E

SOURCE CODE OF FREE-FORM DEFORMATION MESH GENERATOR

```
CNAME GTSOLI.FOR
c*********************************************************************
      subroutine gtsoli(fname)
c*********************************************************************
C
C  PROGRAM: GTSOLI.FOR
C
C  PURPOSE:  get the required data for volume and prepared them for
c            volume meshing.
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: READIN THE GEOMETRY FILE AND DEFORMATION FILE
C       THEN DO THE LINE DEFORMATION AND STORE THE DATA FOR MESHING
c
c*********************************************************************
c
c  called:
c       readso.for -- read in the 8 corner point in a volume
c       orimsh.for -- store the premesh deformed data
c       frform.for -- readin the free formed deformation control points
c       linnod.for -- find the nodes in each line
c       defor1.for -- do the deformation for the lines
c       mshlin.for -- mesh the boundary lines first
c       mpback.for -- mapping the deformed nodes back to original hexahedron
c       drstor.for -- store the nodes for draw picture
c       prepar.for -- store the nodes information for volume meshing
c
c*********************************************************************
c
c  called by: threed.for -- the user's interface
c
c*********************************************************************
      include 'para.inc'
      dimension xg(8),yg(8),zg(8),ck(8),nnum(12),dist(12)
      dimension x(maxsid,12),y(maxsic,12),z(maxsid,12),w(maxsid,12)
      dimension xo(maxsid,12),yo(maxsid,12),zo(maxsid,12)
      character fname*(*)
c
c  store deformation information
c
      call frform(fname,ierr)
c
c  read in the prime volume 8 points
c
      call readso(fname,nx,ny,nz,unit,xg,yg,zg,ierr)
      if(ierr .ne. 0)then
         write(*,*)' None exist file name'
         return
      end if
c
c check the dimension
c
      maxd=max0(nx,ny,nz)+1
      if(maxd .gt. maxsid)then
        write(*,*) ' The <maxsid> need to set to',maxd
        stop
      end if
c
c find the nodes for twelve lines
c
      call linnod(nx,ny,nz,xg,yg,zg,x,y,z,maxsid,nnum)
c
c store the node for plot the original mesh only, can be deleted
c
      call nodsur(nx,ny,nz,x,y,z,maxsid)
      call nodsto(fname,nx,ny,nz)
c
c get the deform shape for the prime volume lines
c
      call defor1(xo,yo,zo,x,y,z,nnum)
c
c mesh the twelve boundary line nodes
c
      call mshlin(nx,ny,nz,x,y,z,nnum,unit,dist)
c
c map the deformed nodes back to the normal cube
```

```
c
        call mpback(nx,ny,nz,xg,yg,zg,dist,xo,yo,zo,x,y,z,w,nnum)
c
c store data for drafting purpose, no need at this time
c do not open it while maxdeg<30 or nodsto is in used
c       call drstor(x,y,z,nnum,unit)
c
c reset the storage address
c
        call prepar(x,y,z,w,nnum,unit)
        return
        end

CNAME VMESH.FOR
C-----------------------------------------------------------------------
        subroutine vmesh(cname)
C-----------------------------------------------------------------------
C
C  PROGRAM: VMESH.FOR
C
C  PURPOSE:  CHOOSE THE REQUIRE MESH PROCESSER TO DO MESHING
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C-----------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION: check the number of nodes in each line, if the
c       number equal two then that will be the final state for that element
c       or the element need todo the transition on it, otherwise continue
c       the subdivision algorithm.
C
C-----------------------------------------------------------------------
C
c  called:
c       push.for    -- stack one step in the loop
c       pop.for     -- pop up one step in the loop
c       vstore.for  -- store nodes for plot
c       xmesh.for   -- mesh in x-direction
c       ymesh.for   -- mesh in y-direction
c       zmesh.for   -- mesh in z-direction
c       lnflag.for  -- set flag to choose the required transition method
c       transi.for  -- set the required transition method
c       ffdefo.for  -- process the free formed deformation
c       wrtmov.for  -- write out the movie and navgraph files for futher
c                      processing.
c-----------------------------------------------------------------------
c
c  called by: threed.for -- user's interface
c
c-----------------------------------------------------------------------
        character cname*(*)
c
        write(*,*)' Automatic mesh generating now ...'
c
        iter = 0
        loop = 1
c
        call push(loop,iter)
c
c...start the looping method
c
10      continue
        call pop(loop,iter)
c
c  check the number of nodes in all the lines, then do the proper method
c
        call lnflag(loop,iflagx,iflagy,iflagz)
        if(iflagx.eq.0 .and. iflagy.eq.0 .and. iflagz.eq.0)then
           call vstore(loop)
c
c   check the nodes in z,y,and x direction and mesh it
c
        else if(iflagy.eq.6)then
           call ymesh(loop,iter)
        else if(iflagz.eq.6)then
           call zmesh(loop,iter)
        else if(iflagx.eq.6)then
           call xmesh(loop,iter)
        else
```

```
c
c   do the transition for all the possible ways (for 0 <iflag < 8)
c
            call transi(loop,iter,iflagx,iflagy,iflagz)
         end if
         if(iter.gt.0)go to 10
c
         call surbnd
         call smooth
         call ffdefo
         call wrtmov(cname)
         call wrtnav(cname)
c
         return
         end

CNAME XMESH.FOR
c*********************************************************************
         subroutine xmesh(loop,iter)
c*********************************************************************
C
C  PROGRAM: XMESH.FOR
C
C  PURPOSE:  do subdivision in x direction
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: QCT.8, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: CHECK THE BEST NODE FOR SUBDIVISION AND ADD THE
C        NEW NODES TO THE SUBDIVISION LINES
C
c*********************************************************************
c
c  called:
c        vgtnod.for -- choose the required node to split with
c        vnewnd.for -- find the new node
c        vnewln.for -- find the new line
c        push.for   -- stack for the new loop
c
c---------------------------------------------------------------------
c
c  called by: vmesh.for -- volume mesh controler
c
c---------------------------------------------------------------------
         include 'para.inc'
         dimension coor(4,maxnod),node(maxsid,12,maxloop)
         common/auto/coor,node,last,unit
         logical found
c
c store extra data for remesh check
c
         do 20 j = 1,12
           do 20 i= 1,node(1,j,loop)+1
              node(i,j,maxloop-1) = node(i,j,loop)
20       continue
         klast = last
c
c  find the split line ending node for surface 1-3, then connect the
c  third and fourth nodes to create the split line for surface 4,
c
         mini = 10000
         do 10 i = 1,3
            if(node(1,i,loop).lt.mini)then
               mini = node(1,i,loop)
               line = i
            end if
10       continue
         if(line.eq.1)then
            line1 = 1
            num1 = 2
            call vgtnod(loop,1,line1,line2,num1,num2)
            call vgtnod(loop,1,line2,line4,num2,num4)
            call vgtnod(loop,2,line1,line3,num1,num3)
         else if(line.eq.2)then
            line2 = 2
            num2 = 2
            call vgtnod(loop,1,line2,line4,num2,num4)
            call vgtnod(loop,2,line2,line1,num2,num1)
```

```
            call vgtnod(loop,2,line1,line3,num1,num3)
         else
            line3 = 3
            num3 = 2
            call vgtnod(loop,1,line3,line1,num3,num1)
            call vgtnod(loop,1,line1,line2,num1,num2)
            call vgtnod(loop,1,line2,line4,num2,num4)
         end if
c
c  find the nodes for each new split lines.
c
         call vnewnd(line1,line2,num1,num2,loop)
         call vnewnd(line1,line3,num1,num3,loop)
         call vnewnd(line2,line4,num2,num4,loop)
         call vnewnd(line3,line4,num3,num4,loop)
c
c split the old boundary line into two new lines
c
         call vnewln(line1,num1,loop)
         call vnewln(line2,num2,loop)
         call vnewln(line3,num3,loop)
         call vnewln(line4,num4,loop)
c
c check for the impossible transition case
c
         call push(loop,iter)
         loop = loop+1
         call push(loop,iter)
c
         return
         end

CNAME YMESH.FOR
c*****************************************************************
         subroutine ymesh(loop,iter)
C*****************************************************************
C
C  PROGRAM: YMESH.FOR
C
C  PURPOSE:  do subdivision in y direction
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION:  FIND THE BEST SUBDIVISION LOCATION AND ADD THE
C       NOCES TO THOSE SUBDIVISION LINES
c
c*****************************************************************
c
c  called:
c       vgtnod.for -- choose the required node to split with
c       vnewnd.for -- find the new node
c       vnewln.for -- find the new line
c       push.for   -- stack for the new loop
c
c*****************************************************************
c
c  called by: vmesh.for -- volume mesh controler
c
c*****************************************************************
         include 'para.inc'
         dimension coor(4,maxnod),node(maxsid,12,maxloop)
         common/auto/coor,node,last,unit
         logical found
c
c store extra data for remesh check
c
         do 20 j = 1,12
           do 20 i= 1,node(1,j,loop)+1
              node(i,j,maxloop-1) = node(i,j,loop)
20       continue
         klast = last
c
c  find the split line ending node for surface 5-7, then connect the
c  third and fourth nodes to create the split line for surface 8.
c
         mini = 1000
         do 10 i = 5,7
```

```
            if(node(1,i,loop).lt.mini)then
               mini = node(1,i,loop)
               line = i
            end if
10       continue
         if(line.eq.5)then
            line1 = 5
            num1 = 2
            call vgtnod(loop,1,line1,line2,num1,num2)
            call vgtnod(loop,1,line2,line4,num2,num4)
            call vgtnod(loop,2,line1,line3,num1,num3)
         else if(line.eq.6)then
            line2 = 6
            num2 = 2
            call vgtnod(loop,1,line2,line4,num2,num4)
            call vgtnod(loop,2,line2,line1,num2,num1)
            call vgtnod(loop,2,line1,line3,num1,num3)
         else
            line3 = 7
            num3 = 2
            call vgtnod(loop,1,line3,line1,num3,num1)
            call vgtnod(loop,1,line1,line2,num1,num2)
            call vgtnod(loop,1,line2,line4,num2,num4)
         end if
c
c  find the nodes for each new split lines.
c
         call vnewnd(line1,line2,num1,num2,loop)
         call vnewnd(line1,line3,num1,num3,loop)
         call vnewnd(line2,line4,num2,num4,loop)
         call vnewnd(line3,line4,num3,num4,loop)
c
c split the old boundary line into two new lines
c
         call vnewln(line1,num1,loop)
         call vnewln(line2,num2,loop)
         call vnewln(line3,num3,loop)
         call vnewln(line4,num4,loop)
c
         call push(loop,iter)
         loop = loop+1
         call push(loop,iter)
c
         return
         end
```

CNAME ZMESH.FOR

```
c*************************************************************************
         subroutine zmesh(loop,iter)
c*************************************************************************
C
C  PROGRAM: ZMESH.FOR
C
C  PURPOSE: do subdivision in z direction
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*************************************************************************
C
C  ALGORITHM EXPLANATION:  FIND THE BEST SUBDIVISION LOCATION AND ADD THE
C       NOCES TO THOSE SUBDIVISION LINES
C
c*************************************************************************
c
c  called:
c       vgtnod.for -- choose the required node to split with
c       vnewnd.for -- find the new node
c       vnewln.for -- find the new line
c       push.for   -- stack for the new loop
c
c*************************************************************************
c
c  called by: vmesh.for -- volume mesh controler
c
c*************************************************************************
         include 'para.inc'
         dimension coor(4,maxnod),node(maxsid,12,maxloop)
         common/auto/coor,node,last,unit
         logical found
```

```
c
c store extra data for remesh check
c
        do 20 j = 1,12
          do 20 i= 1,node(1,j,loop)+1
             node(i,j,maxloop-1) = node(i,j,loop)
20      continue
        klast = last
c
c  find the split line ending node for surface 9-11, then connect the
c  third and fourth nodes to create the split line for surface 12.
c
        mini = 10000
        do 10 i = 9,11
           if(node(1,i,loop).lt.mini)then
              mini = node(1,i,loop)
              line = i
           end if
10      continue
        if(line.eq.9)then
           line1 = 9
           num1 = 2
           call vgtnod(loop,1,line1,line2,num1,num2)
           call vgtnod(loop,1,line2,line4,num2,num4)
           call vgtnod(loop,2,line1,line3,num1,num3)
        else if(line.eq.10)then
           line2 = 10
           num2 = 2
           call vgtnod(loop,1,line2,line4,num2,num4)
           call vgtnod(loop,2,line2,line1,num2,num1)
           call vgtnod(loop,2,line1,line3,num1,num3)
        else
           line3 = 11
           num3 = 2
           call vgtnod(loop,1,line3,line1,num3,num1)
           call vgtnod(loop,1,line1,line2,num1,num2)
           call vgtnod(loop,1,line2,line4,num2,num4)
        end if
c
c  find the nodes for each new split lines.
c
        call vnewnd(line1,line2,num1,num2,loop)
        call vnewnd(line1,line3,num1,num3,loop)
        call vnewnd(line2,line4,num2,num4,loop)
        call vnewnd(line3,line4,num3,num4,loop)
c
c split the old boundary line into two new lines
c
        call vnewln(line1,num1,loop)
        call vnewln(line2,num2,loop)
        call vnewln(line3,num3,loop)
        call vnewln(line4,num4,loop)
c
        call push(loop,iter)
        loop = loop+1
        call push(loop,iter)
c
        return
        end
CNAME CHOLIN.FOR
c-----------------------------------------------------------------------
        subroutine cholin(nod1,nod2,oline,nold)
c-----------------------------------------------------------------------
C
C  PROGRAM: CHOLIN.FOR
C
C  PURPOSE: CHECK TO FIND THE POSSIBLE OLD LINE
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C-----------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION:
c
c---------------------------------------------------------------------
c
c called by: vnext1.for -- find the next possible line
c
c---------------------------------------------------------------------
```

```
      include 'para.inc'
      dimension ick(maxsid),itemp(20)
      logical oline
      common/oldln/nnum(20,maxsid+6),lold(maxsid+6),nall
c
      oline=.false.
c
c...check for the discrete line case or partial line case
c
      ntemp = nod1
      icnt = 0
10    continue
        do 40 i = 1,nall
c...start normal checking
          if(ntemp.eq.nnum(1,i))then
            icnt = icnt + 1
            ick(icnt) = i
            ntemp = nnum(lold(i),i)
c...regular coincident case and final condition of small lines case
            if(ntemp.eq.nod2)then
               oline = .true.
               go to 60
            end if
c...partial line case
            if(lold(i).gt.2)then
               do 20 k = 2,lold(i)
               if(nod2.eq.nnum(k,i))then
                  oline = .true.
                  nall = nall + 1
                  mc = 1
                  nnum(1,nall) = nod1
                  do 21 m = 1,icnt-1
                   do 21 n = 2,lold(ick(m))
                     mc = mc + 1
                     nnum(mc,nall) = nnum(n,ick(m))
21                continue
                  do 30 j = 2,k
                     mc = mc + 1
                     nnum(mc,nall) = nnum(j,i)
30                continue
                  lold(nall) = mc
                  nold = nall
c...rearrenge the original line
                  do 31 j = 0,lold(i)-k
                     nnum(j+1,i) = nnum(j+k,i)
31                continue
                  lold(i) = lold(i)-k+1
                  return
               end if
20             continue
            end if
c...many small line segments case
            go to 10
          end if
40        continue
60    continue
c
      return
      end
CNAME CKOLIN.FOR
c-----------------------------------------------------------------------
      subroutine ckolin(nod1,nod2,oline,nold)
C-----------------------------------------------------------------------
C
C  PROGRAM: CKOLIN.FOR
C
C  PURPOSE:  CHECK NODE IN ALL THE OLD LINE
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C-----------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION: CHECK THROUGH THE ALL OLD LINES TO SEE IF THE
C       GIVEN LINE WAS AN EXIST OLD LINE
c
c-----------------------------------------------------------------------
c
c called by: fnewnd.for -- find the new node
c
```

```
c*********************************************************************
      include 'para.inc'
      dimension ick(maxsid),itemp(20)
      logical oline
      common/oldln/nnum(20,maxsid*8),lold(maxsid*8),nall
c
      oline=.false.
c
c...check for the discrete line case or partial line case
c
      ntemp = nod1
      icnt = 0
10    continue
        do 40 i = 1,nall
c...check for the inverse special case
          if(nod1.eq.nnum(lold(i),i) .and. nod2.eq.nnum(1,i))then
            do 50 km = 1,lold(i)
              itemp(km) = nnum(km,i)
50          continue
            do 51 km = 1,lold(i)
              nnum(km,i) = itemp(lold(i)-km+1)
51          continue
          end if
c...start normal checking
          if(ntemp.eq.nnum(1,i))then
            icnt = icnt + 1
            ick(icnt) = i
            ntemp = nnum(lold(i),i)
c...regular coincident case and final condition of small lines case
            if(ntemp.eq.nod2)then
               oline = .true.
               go to 60
            end if
c...partial line case
            if(lold(i).gt.2)then
               do 20 k = 2,lold(i)
               if(nod2.eq.nnum(k,i))then
                  oline = .true.
                  nall = nall + 1
                  mc = 1
                  nnum(1,nall) = nod1
                  do 21 m = 1,icnt-1
                   do 21 n = 2,lold(ick(m))
                     mc = mc + 1
                     nnum(mc,nall) = nnum(n,ick(m))
21                continue
                  do 30 j = 2,k
                     mc = mc + 1
                     nnum(mc,nall) = nnum(j,i)
30                continue
                  lold(nall) = mc
                  nold = nall
c...restore the original line
                  do 31 j = 0,lold(i)-k
                     nnum(j+1,i) = nnum(j+k,i)
31                continue
                  lold(i) = lold(i)-k+1
                  return
                end if
20              continue
              end if
c...many small line segments case
            go to 10
          end if
40        continue
60      continue
        if(oline)then
          if(icnt.gt.1)then
c...store the the lines found into one line
            nall = nall + 1
            nnum(1,nall) = nod1
            jc = 1
            do 90 i = 1,icnt
             do 90 j = 2,lold(ick(i))
               jc = jc + 1
               nnum(jc,nall) = nnum(j,ick(i))
90          continue
            lold(nall) = jc
c...delete those small lines
            do 71 ik = icnt,1,-1
```

```
                do 70 i = ick(ik),nall-1
                  lold(i) = lold(i+1)
                  do 80 jj = 1,lold(i)
                    nnum(jj,i) = nnum(jj,i+1)
80                continue
70              continue
                nall = nall - 1
71            continue
              nold = nall
            else
              nold = ick(icnt)
            end if
          end if
c
          return
          end
CNAME CKTRAN.FOR
c*****************************************************************************
          subroutine cktran(temp,kt)
C*****************************************************************************
C
C  PROGRAM: CKTRAN.FOR
C
C  PURPOSE: CHECK THE EXIST TRANSITION NODES
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION: check, if the node already exist then use it
c  for the new surface and delete it from the check list (to save time
c  and space), otherwise create a new node.
c
c*****************************************************************************
c
c called by:
C         tran1.for -- transition method 1
C         tran2.for -- transition method 2
c
c*****************************************************************************
          include 'para.inc'
          dimension coor(4,maxnod),node(maxsid,12,maxloop),temp(4)
          common/auto/coor,node,last,unit
          common/tran/ntran(1000),kall
          logical found
c
          found = .false.
          do 10 i = 1,kall
            if(abs(temp(1)-coor(1,ntran(i))).le.0.005 .and.
     $         abs(temp(2)-coor(2,ntran(i))).le.0.005 .and.
     $         abs(temp(3)-coor(3,ntran(i))).le.0.005) then
              kt = ntran(i)
              nod = i
              found = .true.
              go to 20
            end if
10        continue
20        continue
          if(found)then
            do 30 j = nod,kall-1
              ntran(j) = ntran(j+1)
30          continue
            kall = kall - 1
          else
            kall = kall + 1
            ntran(kall) = kt
            last = kt
          end if
c
          return
          end

CNAME DEFOR1.FOR
c*****************************************************************************
          subroutine defor1(xo,yo,zo,x,y,z,nodnum)
c*****************************************************************************
C
C  PROGRAM: DEFOR1.FOR
C
```

```
C  PURPOSE: DO THE DEFORMATION FOR BOUNDARY LINE
C
C  WRITTEN BY:  SSUTA HSU        CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                  CHECKED BY:            DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: deform the boundary lines first, then store
c       xo,yo,zo as reference points to map the real nodes back to original
c       volume for auto meshing
c
c*********************************************************************
c
c  called: deform.for -- free formed deformation
c
c*********************************************************************
c
c  called by: gtsoli.for -- get volume data
c
c*********************************************************************
        include 'para.inc'
        dimension x(maxsid,12),y(maxsid,12),z(maxsid,12),nodnum(12)
        dimension xo(maxsid,12),yo(maxsid,12),zo(maxsid,12)
c
c deformed after meshing the cube surfaces
c
        do 10 j=1,12
        do 10 i=1,nodnum(j)
           xi=x(i,j)
           yi=y(i,j)
           zi=z(i,j)
           call deform(xi,yi,zi,xd,yd,zd)
           x(i,j)=xd
           y(i,j)=yd
           z(i,j)=zd
           xo(i,j)=xd
           yo(i,j)=yd
           zo(i,j)=zd
10      continue
c
        return
        end
CNAME DRSTOR.FOR
c*********************************************************************
        subroutine drstor(xx,yy,zz,nodnum,unit)
c*********************************************************************
C
C  PROGRAM: DRSTOR.FOR
C
C  PURPOSE:  store the nodal information for plot
C
C  WRITTEN BY:  SSUTA HSU        CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                  CHECKED BY:            DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: STORE DATA INTO GRAPHIC BUCKET
c
c*********************************************************************
c
c  called: deform.for -- do the free formed deformation
c
c*********************************************************************
c
c  called by:
c       gtsoli.for -- get the original volume data
c       vmesh.for  -- volume mesh controler
c
c*********************************************************************
        include 'para.inc'
        dimension xx(maxsid,12),yy(maxsid,12),zz(maxsid,12),nodnum(12)
        dimension x(0:maxdeg,maxid),y(0:maxdeg,maxid),
     $          z(0:maxdeg,maxid),ideg(maxid),isurf(4,6)
        common/data/x,y,z,ideg
        data isurf/1,6,2,5, 1,10,3,9, 10,8,12,6, 2,12,4,11,
     $           9,7,11,5, 3,8,4,7/
c
c from the boundary lines segments to get surface boundary nodes
c
        do 10 k = 1,6
```

```
            num=-1
            do 20 i=1,4
              line=isurf(i,k)
              ln=nodnum(line)-1
              if(i.eq.1 .or. i.eq.2)then
                do 30 j=1,ln
                   num=num+1
                   xi=xx(j,line)
                   yi=yy(j,line)
                   zi=zz(j,line)
                   call deform(xi,yi,zi,xd,yd,zd)
                   x(num,k)=xd
                   y(num,k)=yd
                   z(num,k)=zd
30              continue
              else
                do 40 j=ln+1,2,-1
                   num=num+1
                   xi=xx(j,line)
                   yi=yy(j,line)
                   zi=zz(j,line)
                   call deform(xi,yi,zi,xd,yd,zd)
                   x(num,k)=xd
                   y(num,k)=yd
                   z(num,k)=zd
40              continue
              end if
20          continue
            ideg(k) = num
10        continue
c
          return
          end
```

```
CNAME EVENUM.FOR
c-----------------------------------------------------------------------
          subroutine evenum(dist,nodnum,iflag)
C-----------------------------------------------------------------------
C
C  PROGRAM: EVENUM.FOR
C
C  PURPOSE: set the surface close loop to be an even number
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C-----------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION: CHECK THE CLOSE LOOP OF A SURFACE IF THE TOTAL
C       NUMBER WAS NOT AN EVEN NUMBER THEN ADD 1 TO MAKE IT EVEN
C
c-----------------------------------------------------------------------
c
c  called by: mshlin.for -- mesh the boundary lines first
c
c-----------------------------------------------------------------------
          dimension nodnum(12),isurf(4,6),line(12),iflag(12),dist(12)
          logical found
          data line /1,2,3,4,5,6,7,8,9,10,11,12/
          data isurf /1,6,2,5, 1,10,3,9, 10,8,12,6, 2,12,4,11,
     $               5,9,7,11, 3,8,4,7/
c
c  increase the largest line segment by one to change the opposit
c  odd number to even number.
c
          do 10 i=1,6
            do 20 j = 1,2
              if(mod(nodnum(isurf(j,i)),2) .ne.
     $                  mod(nodnum(isurf(j+2,i)),2))then
                if(line(isurf(j,i)).gt.0 .and.
     $                  line(isurf(j+2,i)).gt.0)then
                  if(nodnum(isurf(j,i)).gt.nodnum(isurf(j+2,i)))then
                    ic = isurf(j,i)
                  else
                    ic = isurf(j+2,i)
                  end if
                else if(line(isurf(j,i)).gt.0)then
                  ic = isurf(j,i)
                else if(line(isurf(j+2,i)).gt.0)then
                  ic = isurf(j+2,i)
```

```
                end if
                nodnum(ic) = nodnum(ic)+1
                iflag(ic) = 1
              end if
              line(isurf(j,i)) = -isurf(j,i)
              line(isurf(j+2,i)) = -isurf(j+2,i)
20          continue
10        continue
c
          return
          end

CNAME FNEWND.FOR
c-------------------------------------------------------------------------
          subroutine fnewnd(line1,line2,num1,num2,left,loop,nseg,
     $                      oline,nold)
C-------------------------------------------------------------------------
C
C  PROGRAM: FNEWND.FOR
C
C  PURPOSE: FIND THE NEW NODES
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C-------------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION: FIND THE OLD NODES, OR CREATE NEW NODES
c
c-------------------------------------------------------------------------
c
c called: vnewds.for --  find the distance for the deformed line
c         ckolin.for --  check the existence of the old line
c
c-------------------------------------------------------------------------
c
c called by: vnewnd.for -- find the new node
c
c-------------------------------------------------------------------------
          include 'para.inc'
          dimension coor(4,maxnod),node(maxsid,12,maxloop),xo(maxsid),
     $              yo(maxsid),zo(maxsid)
          real new,newsum
          logical oline
          common/auto/coor,node,last,unit
          common/oldln/nnum(20,maxsid-6),lold(maxsid-6),nall
c
          oline=.false.
          num = node(1,left,loop)
          nod1 = node(num1+1,line1,loop)
          nod2 = node(num2+1,line2,loop)
c if this line was an already exist old line then return require parameter
c for copy its line data
          do 30 i = 1,nall
            if(nod2.eq.nnum(lold(i),i) .and. nod1.eq.nnum(1,i))then
              oline = .true.
              nold = i
              return
            end if
30        continue
c...check for the discontinue line case or partial line case
          call ckolin(nod1,nod2,oline,nold)
          if(oline) return
c
c find the distance for the deformed line
c
          call vnewds(nod1,nod2,num,xo,yo,zo,dist)
c
c find the require segments (even number in a close polygon) for
c the deformated line
c
          nseg = nint(dist/unit) + 1
          if(nseg.lt.2) nseg = 2
          isum = num1 + num2 + num + nseg
          if(mod(isum,2).ne.0)then
             nseg = nseg + 1
          end if
c
c map this line back to the reference hexahedron
c
```

```
      new=dist/(nseg-1)
      old=1.0e30
      k = 0
      newsum=new
      i = 0
      oldsum=0.0
      ratio =0.0
10    continue
        if(newsum.le.oldsum+0.001)then
          diff = newsum - (oldsum-old)
          if(diff.gt.new)diff=new
          ratio = ratio + diff/old/(num-1)
          k = k+1
          coor(1,last+k)=coor(1,nod1) + (coor(1,nod2)-coor(1,nod1))
     $                    *ratio
          coor(2,last+k)=coor(2,nod1) + (coor(2,nod2)-coor(2,nod1))
     $                    *ratio
          coor(3,last+k)=coor(3,nod1) + (coor(3,nod2)-coor(3,nod1))
     $                    *ratio
          coor(4,last+k)=ratio
          newsum = new*(k+1)
        else
          diff = oldsum - new*k
          if(diff.gt.old)diff=old
          ratio = ratio + diff/old/(num-1)
          i = i+1
          old = sqrt((xo(i+1)-xo(i))**2 + (yo(i+1) -
     $                yo(i))**2 + (zo(i+1)-zo(i))**2)
          oldsum = oldsum + old
        end if
        if(k.ge.nseg-2) go to 20
        go to 10
20    continue
c
      return
      end
CNAME GTSIDE.FOR
c*********************************************************************
      subroutine gtside(ibeg,next,left,nrit)
C*********************************************************************
C
C  PROGRAM: GTSIDE.FOR
C
C  PURPOSE: FIND THE OTHER OPPOSITE LINES OF A GIVEN SURFACE
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: from the given opposite lines to find the othe
c      opposite lines of that surface
c
c*********************************************************************
c
c  called by:
c      vgtnod.for -- find the best node for split line
c      vnewnd.for -- find the new distance
c
c*********************************************************************
c for x direction
      if(ibeg.eq.1 .and. next.eq.2)then
        left = 5
        nrit = 6
      else if(ibeg.eq.1 .and. next.eq.3)then
        left = 9
        nrit = 10
      else if(ibeg.eq.2 .and. next.eq.4)then
        left = 11
        nrit = 12
      else if(ibeg.eq.3 .and. next.eq.4)then
        left = 7
        nrit = 8
c  for y direction
      else if(ibeg.eq.5 .and. next.eq.6)then
        left = 1
        nrit = 2
      else if(ibeg.eq.5 .and. next.eq.7)then
        left = 9
        nrit = 11
```

```
        else if(ibeg.eq.6 .and. next.eq.8)then
           left = 10
           nrit = 12
        else if(ibeg.eq.7 .and. next.eq.8)then
           left = 3
           nrit = 4
c  for z direction
        else if(ibeg.eq.9 .and. next.eq.10)then
           left = 1
           nrit = 3
        else if(ibeg.eq.9 .and. next.eq.11)then
           left = 5
           nrit = 7
        else if(ibeg.eq.11 .and. next.eq.12)then
           left = 2
           nrit = 4
        else if(ibeg.eq.10 .and. next.eq.12)then
           left = 6
           nrit = 8
        end if
c
        return
        end
CNAME LFLAG.FOR
c*******************************************************************************
        subroutine lflag(loop,line1,line2,line3,line4,iflag)
C*******************************************************************************
C
C  PROGRAM: LFLAG.FOR
C
C  PURPOSE: SET FLAG FOR THE LINES
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*******************************************************************************
C
C  ALGORITHM EXPLANATION: check the number of nodes on each line at x,y,
c       or z direction and set flag for them
c
c*******************************************************************************
c
c  called by: lnflag.for -- get flag number for lines
c
c*******************************************************************************
        include 'para.inc'
        dimension coor(4,maxnod),node(maxsid,12,maxloop)
        common/auto/coor,node,last,unit
c
        num1 = node(1,line1,loop)
        num2 = node(1,line2,loop)
        num3 = node(1,line3,loop)
        num4 = node(1,line4,loop)
        if(num1.eq.2 .and. num2.eq.2 .and.
     $          num3.eq.2 .and. num4.eq.2)then
           iflag = 0
c
c for transition case 1.
        else if(num1.eq.4 .and. num2.eq.2 .and. num3.eq.2
     $          .and. num4.eq.2)then
           iflag = 11
        else if(num2.eq.4 .and. num3.eq.2 .and. num4.eq.2
     $          .and. num1.eq.2)then
           iflag = 12
        else if(num3.eq.4 .and. num4.eq.2 .and. num1.eq.2
     $          .and. num2.eq.2)then
           iflag = 13
        else if(num4.eq.4 .and. num1.eq.2 .and. num2.eq.2
     $          .and. num3.eq.2)then
           iflag = 14
c
c for transition case 2.
        else if((num1.eq.2 .and. num3.eq.2 .and.
     $          mod(num2,2).eq.0 .and. mod(num4,2).eq.0) .or.
     $          (num2.eq.2 .and. num4.eq.2 .and.
     $          mod(num1,2).eq.0 .and. mod(num3,2).eq.0))then
           iflag = 21
        else if((num1.eq.2 .and. num2.eq.2 .and.
     $          mod(num3,2).eq.0 .and. mod(num4,2).eq.0) .or.
     $          (num3.eq.2 .and. num4.eq.2 .and.
```

```
     $          mod(num1,2).eq.0 .and. mod(num2,2).eq.0))then
           iflag = 22
c
c for transition case 3.
c       else if(num1.eq.3 .and. num2.eq.3 .and. ((num3.eq.2 .and.
c     $    mod(num4,2).eq.0).or.(mod(num3,2).eq.0 .and. num4.eq.2)))then
c          iflag = 31
c       else if(num3.eq.3 .and. num4.eq.3 .and. ((num1.eq.2 .and.
c     $    mod(num2,2).eq.0).or.(mod(num1,2).eq.0 .and. num2.eq.2)))then
c          iflag = 32
c       else if(num1.eq.3 .and. num3.eq.3 .and. ((num2.eq.2 .and.
c     $    mod(num4,2).eq.0).or.(mod(num2,2).eq.0 .and. num4.eq.2)))then
c          iflag = 33
c       else if(num2.eq.3 .and. num4.eq.3 .and. ((num1.eq.2 .and.
c     $    mod(num3,2).eq.0).or.(mod(num1,2).eq.0 .and. num3.eq.2)))then
c          iflag = 34
c
c for transition case 4
c       else if((num1.eq.3 .and. mod(num4,2).eq.0 .and. mod(num3,2).eq.0
c     $          .and. mod(num2,2).eq.0) .or.
c     $          (num1.eq.3 .and. mod(num2,2).eq.0 .and. mod(num3,2).eq.0
c     $          .and. mod(num4,2).eq.0) .or.
c     $          (num1.eq.3 .and. mod(num3,2).eq.0 .and. mod(num4,2).eq.0
c     $          .and. mod(num2,2).eq.0)) then
c          iflag = 41
c       else if((num2.eq.3 .and. mod(num3,2).eq.0 .and. mod(num4,2).eq.0
c     $          .and. mod(num1,2).eq.0) .or.
c     $          (num2.eq.3 .and. mod(num1,2).eq.0 .and. mod(num4,2).eq.0
c     $          .and. mod(num3,2).eq.0) .or.
c     $          (num2.eq.3 .and. mod(num4,2).eq.0 .and. mod(num3,2).eq.0
c     $          .and. mod(num1,2).eq.0)) then
c          iflag = 42
c       else if((num3.eq.3 .and. mod(num2,2).eq.0 .and. mod(num1,2).eq.0
c     $          .and. mod(num4,2).eq.0) .or.
c     $          (num3.eq.3 .and. mod(num1,2).eq.0 .and. mod(num2,2).eq.0
c     $          .and. mod(num4,2).eq.0) .or.
c     $          (num3.eq.3 .and. mod(num4,2).eq.0 .and. mod(num1,2).eq.0
c     $          .and. mod(num2,2).eq.0)) then
c          iflag = 43
c       else if((num4.eq.3 .and. mod(num1,2).eq 0 .and. mod(num2,2).eq.0
c     $          .and. mod(num3,2).eq.0) .or.
c     $          (num4.eq.3 .and. mod(num2,2).eq 0 .and. mod(num1,2).eq.0
c     $          .and. mod(num3,2).eq.0) .or.
c     $          (num4.eq.3 .and. mod(num3,2).eq.0 .and. mod(num2,2).eq.0
c     $          .and. mod(num1,2).eq.0)) then
c          iflag = 44
c
c for regular subdivision
       else if(num1.ne.2 .and. num3.ne.2 .and.
     $         num2.ne.2 .and. num4.ne.2)then
          iflag = 6
c
c unknown situation --- transition case 5
c
       else
          iflag = 5
       end if
c
       return
       end

CNAME LINDIS.FOR
c*****************************************************************
       subroutine lindis(num,j,x,y,z,maxd,dist)
C*****************************************************************
C
C  PROGRAM: LINDIS.FOR
C
C  PURPOSE: find the distance for 12 boundary lines
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: USE LINEAR METHOD TO FIND THE DISTANCE
c
c*****************************************************************
c
c  called by: mshlin.for -- mesh the boundary lines first
```

```
c
c*****************************************************************************
      dimension x(maxd,12),y(maxd,12),z(maxd,12)
c
      sum=0.0
      do 10 i=1,num
         sum = sum+sqrt((x(i+1,j)-x(i,j))**2 +
     $          (y(i+1,j)-y(i,j))**2 + (z(i+1,j)-z(i,j))**2)
10    continue
      dist = sum
c
      return
      end
CNAME LINNOD.FOR
c*****************************************************************************
      subroutine linnod(nx,ny,nz,xg,yg,zg,x,y,z,maxd,nodnum)
C*****************************************************************************
C
C  PROGRAM: LINNOD.FOR
C
C  PURPOSE: find the nodal data for twelve lines
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*****************************************************************************
c
c  called by: gtsoli.for -- prepare the information for meshing
c
c*****************************************************************************
      dimension x(maxd,12),y(maxd,12),z(maxd,12),nodnum(12)
      dimension xg(8),yg(8),zg(8),ick(24)
      data ick/1,4,5,8,2,3,6,7, 1,2,5,6,4,3,8,7, 1,2,4,3,5,6,8,7/
c
c store  nodes on x direction line 1,2,3,4
c
      do 20 m=1,4
        do 10 i=1,nx+1
          ratio=float(i-1)/float(nx)
          x(i,m)=xg(ick(m))+(xg(ick(m+4))-xg(ick(m)))*ratio
          y(i,m)=yg(ick(m))+(yg(ick(m+4))-yg(ick(m)))*ratio
          z(i,m)=zg(ick(m))+(zg(ick(m+4))-zg(ick(m)))*ratio
10      continue
        nodnum(m)=nx+1
20    continue
c
c store  nodes on y direction line 5,6,7,8
c
      do 40 m=9,12
        do 30 i=1,ny+1
          ratio=float(i-1)/float(ny)
          x(i,m-4)=xg(ick(m))+(xg(ick(m+4))-xg(ick(m)))*ratio
          y(i,m-4)=yg(ick(m))+(yg(ick(m+4))-yg(ick(m)))*ratio
          z(i,m-4)=zg(ick(m))+(zg(ick(m+4))-zg(ick(m)))*ratio
30      continue
        nodnum(m-4)=ny+1
40    continue
c
c store nodes on z direction line 9,10,11,12
c
      do 60 m=17,20
        do 50 i=1,nz+1
          ratio=float(i-1)/float(nz)
          x(i,m-8)=xg(ick(m))+(xg(ick(m+4))-xg(ick(m)))*ratio
          y(i,m-8)=yg(ick(m))+(yg(ick(m+4))-yg(ick(m)))*ratio
          z(i,m-8)=zg(ick(m))+(zg(ick(m+4))-zg(ick(m)))*ratio
50      continue
        nodnum(m-8)=nz+1
60    continue
c
      return
      end
CNAME LNFLAG.FOR
c*****************************************************************************
      subroutine lnflag(loop,iflagx,iflagy,iflagz)
```

```
C*****************************************************************
C
C  PROGRAM: LNFLAG.FOR
C
C  PURPOSE: SET NODAL FLAG FOR X,Y,Z DIRECTION
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: check the number of nodes on each line to
c       determind the require flag number for proper processing.
c
c*****************************************************************
c
c  called : lflag.for -- set flag for the require direction
c
c*****************************************************************
c
c  called by: vmesh.for -- volume mesh controler
c
c*****************************************************************
c
c set flag for x direction
c
        call lflag(loop,1,2,3,4,iflagx)
c
c set flag for y direction
c
        call lflag(loop,5,6,7,8,iflagy)
c
c set flag for z direction
c
        call lflag(loop,9,10,11,12,iflagz)
c
        return
        end

CNAME MPBACK.FOR
c*****************************************************************
        subroutine mpback(nx,ny,nz,xg,yg,zg,dis,xo,yo,zo,x,y,z,w
     $                   ,nodnum)
C*****************************************************************
C
C  PROGRAM: MPBACK.FOR
C
C  PURPOSE: mapped the nodes back to original volume for meshing
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: FIND THE PROPER LOCATION OF EACH DEFORMED NODE
C       IN IT'S ORIGINAL CUBE FOR FURTHER PROCESSING
c
c*****************************************************************
c
c  called by: gtsoli.for -- prepare the volume information
c
c*****************************************************************
        include 'para.inc'
        real new,newsum
        dimension x(maxsid,12),y(maxsid,12),z(maxsid,12),w(maxsid,12)
        dimension xo(maxsid,12),yo(maxsid,12),zo(maxsid,12),
     $            nodnum(12),dist(12)
        dimension xg(8),yg(8),zg(8),ick(24)
        data ick/1,4,5,8,1,2,5,6,1,2,4,3, 2,3,6,7,4,3,8,7,5,6,8,7/
c
        do 20 j=1,12
          if(j.le.4)then
             nm=nx
          else if(j.gt.8)then
             nm=nz
          else
             nm=ny
          end if
            new=dist(j)/(nodnum(j)-1)
            old=1.0e30
```

```
            k=0
            newsum=0.0
            i=0
            oldsum=0.0
             ratio =0.0
10          continue
            if(newsum.le.oldsum+0.001)then
               diff = newsum - (oldsum-old)
               if(diff.gt.new)diff=new
               ratio = ratio + diff/old/nm
               if(ratio.gt.1.0)ratio=1.0
               k = k+1
               x(k,j)=xg(ick(j))+(xg(ick(j+12))-xg(ick(j)))*ratio
               y(k,j)=yg(ick(j))+(yg(ick(j+12))-yg(ick(j)))*ratio
               z(k,j)=zg(ick(j))+(zg(ick(j+12))-zg(ick(j)))*ratio
               w(k,j) = ratio
               newsum = new*k
            else
               diff = oldsum - new*(k-1)
               if(diff.gt.old)diff=old
               ratio = ratio + diff/old/nm
               i = i+1
               old = sqrt((xo(i+1,j)-xo(i,j))**2 + (yo(i+1,j) -
     $                     yo(i,j))**2 + (zo(i+1,j)-zo(i,j))**2)
               oldsum = oldsum + old
            end if
            if(k.ge.nodnum(j)) go to 20
            go to 10
20        continue
c
          return
          end

CNAME MSHLIN.FOR
c*****************************************************************************
          subroutine mshlin(nx,ny,nz,x,y,z,nodnum,unit,dist)
c*****************************************************************************
C
C  PROGRAM: MSHLIN.FOR
C
C  PURPOSE:  MESH THE TWELVE BOUNDARY LINE
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION: FIND THE NODES ON BOUNDARY LINE BY THE REQUIRE
C       ELEMENT SIZE
c
c*****************************************************************************
c
c  called:
c       lindis.for -- find the total distance of a deformed line
c       evenum.for -- set the total node number to be even number
c       renode.for -- if the number is not even then reset the number
c
c*****************************************************************************
c
c  called by: gtsoli.for -- prepare the information for meshing
c
c*****************************************************************************
          include 'para.inc'
          dimension x(maxsid,12),y(maxsid,12),z(maxsid,12),nodnum(12)
          dimension xb(maxsid),yb(maxsid),zb(maxsid),dist(12),iflag(12)
c
          do 20 j=1,12
            sum=0.0
            nnod=0
            gdis=0.0
            if(j.le.4)then
                  num=nx
            else if(j.gt.4 .and. j.le.8)then
                  num=ny
            else if(j.gt.8)then
                  num=nz
            end if
c
c find the boundary distances for each line first
c
```

```
          call lindis(num,j,x,y,z,maxsid,dis)
c
c reset the segment of each line after deformated
c
          nseg = nint(dis/unit)
          if(nseg.eq.0) nseg=1
          fdis=dis/nseg
          do 10 i=1,num+1
            if(i.gt.1)then
              delt = sqrt((x(i,j)-x(i-1,j))**2 + (y(i,j)-
     $                     y(i-1,j))**2 + (z(i,j)-z(i-1,j))**2)
              sum = sum+delt
            else
              delt = 1.e10
            end if
c
c if the final length between two node (delt) is greater than deformed length
c then increase the deformed node number otherwise increase the final node
c number.
11          if(gdis.le.sum+.001)then
              nnod=nnod+1
              if(nnod.gt.maxsid)then
                write(*,*)'ERROR : increase maxsid to',nnod
                return
              end if
              diff=sum-gdis
              xb(nnod) = x(i,j) - (x(i,j)-x(i-1,j))/delt*diff
              yb(nnod) = y(i,j) - (y(i,j)-y(i-1,j))/delt*diff
              zb(nnod) = z(i,j) - (z(i,j)-z(i-1,j))/delt*diff
              gdis=gdis+fdis
              go to 11
            end if
10        continue
          nodnum(j)=nnod
c
c  restore the line node data
          do 30 i=1,nnod
             x(i,j)=xb(i)
             y(i,j)=yb(i)
             z(i,j)=zb(i)
30        continue
          dist(j)=dis
20      continue
c
c  set the surface close loop is even number to perform hexahedron mesh
c
        call evenum(dist,nodnum,iflag)
        call renode(nx,ny,nz,x,y,z,nodnum,dist,iflag)
c
        return
        end

CNAME NODLIN.FOR
c*********************************************************************
        subroutine nodlin(xg,yg,zg,nx,ny,nz,x,y,z)
c*********************************************************************
C
C  PROGRAM: NODLIN.FOR
C
C  PURPOSE: store the nodes data of twelve lines
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*********************************************************************
c
c  called:
c
c*********************************************************************
c
c  called by: orimsh.for -- find the original mesh
c
c*********************************************************************
        dimension x(nx+1,ny+1,nz+1),y(nx+1,ny+1,nz+1),z(nx+1,ny+1,nz+1)
        dimension xg(8),yg(8),zg(8),ick(8)
        data ick/1,2,4,3,5,6,8,7/
```

```
c
c store the corner point first
c
        m = 0
        do 10 k = 1,nz+1,nz
          do 10 j = 1,ny+1,ny
            do 10 i = 1,nx+1,nx
              m = m+1
              x(i,j,k) = xg(ick(m))
              y(i,j,k) = yg(ick(m))
              z(i,j,k) = zg(ick(m))
10      continue
c
c store nodes into x-direction boundary lines
c
        do 20 k=1,nz+1,nz
          do 20 j=1,ny+1,ny
            do 20 i=2,nx
               ratio=float(i-1)/float(nx)
               x(i,j,k) = x(1,j,k) + (x(nx+1,j,k) - x(1,j,k))*ratio
               y(i,j,k) = y(1,j,k) + (y(nx+1,j,k) - y(1,j,k))*ratio
               z(i,j,k) = z(1,j,k) + (z(nx+1,j,k) - z(1,j,k))*ratio
20      continue
c
c store nodes into y-direction boundary lines
c
        do 30 k=1,nz+1,nz
          do 30 i=1,nx+1,nx
            do 30 j=2,ny
               ratio=float(j-1)/float(ny)
               x(i,j,k) = x(i,1,k) + (x(i,ny+1,k) - x(i,1,k))*ratio
               y(i,j,k) = y(i,1,k) + (y(i,ny+1,k) - y(i,1,k))*ratio
               z(i,j,k) = z(i,1,k) + (z(i,ny+1,k) - z(i,1,k))*ratio
30      continue
c
c store nodes into z-direction boundary lines
c
        do 40 i=1,nx+1,nx
          do 40 j=1,ny+1,ny
            do 40 k=2,nz
               ratio=float(k-1)/float(nz)
               x(i,j,k) = x(i,j,1) + (x(i,j,nz+1) - x(i,j,1))*ratio
               y(i,j,k) = y(i,j,1) + (y(i,j,nz+1) - y(i,j,1))*ratio
               z(i,j,k) = z(i,j,1) + (z(i,j,nz-1) - z(i,j,1))*ratio
40      continue
c
        return
        end
CNAME NODSTO.FOR
c*****************************************************************************
        subroutine nodsto(cname,nx,ny,nz)
c*****************************************************************************
C
C  PROGRAM: NODSTO.FOR
C
C  PURPOSE: store all the nodes for movie file display
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*****************************************************************************
c
c  called:
c
c*****************************************************************************
c
c  called by: orimsh.for -- find the original mesh
c
c*****************************************************************************
        include 'para.inc'
        dimension x(0:maxdeg,maxid),y(0:maxdeg,maxid),
     $            z(0:maxdeg,maxid),ideg(maxid)
        common/data/x,y,z,ideg
        dimension coor(4,maxnod),node(maxsid,12,maxloop),jp(4*maxid)
        common/auto/coor,node,last,unit
        character cname*(*)
```

```
c
      imov = 13
      open(imov,file=cname//'.def',status='new')
c
      np = 1
      ic = 0
      npt = 0
c
c store mesh for the front,back face
c
      ic = 0
      m = (nx+1)*(ny+1)
      do 10 k = 0,m,m
        do 10 j = 1,ny
          do 10 i = 1,nx
             jp(ic+1) = k+(nx+1)*(j-1)+i
             jp(ic+2) = k+(nx+1)*(j-1)+i+1
             jp(ic+3) = k+(nx+1)*j+i+1
             jp(ic+4) = -(k+(nx+1)*j+i)
             ic = ic + 4
10    continue
c
c store mesh for the left,right face
c
      n = (ny+1)*(nz+1)
      do 20 k = 2*m,(2*m+n),n
        do 20 j = 1,ny
          do 20 i = 1,nz
             jp(ic+1) = k+(nz+1)*(j-1)+i
             jp(ic+2) = k+(nz+1)*(j-1)+i+1
             jp(ic+3) = k+(nz+1)*j+i+1
             jp(ic+4) = -(k+(nz+1)*j+i)
             ic = ic + 4
20    continue
c
c store mesh for the top,bottom face
c
      l = (nx+1)*(nz+1)
      do 30 k = 2*(m+n),(2*(m+n)+l),l
        do 30 j = 1,nz
          do 30 i = 1,nx
             jp(ic+1) = k+(nx+1)*(j-1)+i
             jp(ic+2) = k+(nx+1)*(j-1)+i+1
             jp(ic+3) = k+(nx+1)*j+i+1
             jp(ic+4) = -(k+(nx+1)*j+i)
             ic = ic + 4
30    continue
      nedge = ic
      nj = last
      npt = ic/4
c
c store mesh for plot
c
      do 40 id = 1,npt
         do 50 mc = 1,4
           if(mc.eq.4)then
              x(mc-1,id) = coor(1,-jp((id-1)*4+mc))
              y(mc-1,id) = coor(2,-jp((id-1)*4+mc))
              z(mc-1,id) = coor(3,-jp((id-1)*4+mc))
           else
              x(mc-1,id) = coor(1,jp((id-1)*4+mc))
              y(mc-1,id) = coor(2,jp((id-1)*4+mc))
              z(mc-1,id) = coor(3,jp((id-1)*4+mc))
           end if
50       continue
         ideg(id) = 3
40    continue
c
c deform the mesh for first view of the shape
c
      do 60 i = 1,last
         call deform(coor(1,i),coor(2,i),coor(3,i),xo,yo,zo)
         coor(1,i) = xo
         coor(2,i) = yo
         coor(3,i) = zo
60    continue
c
c write movie polygon file
c
      write(imov,150) np,nj,npt,nedge
```

```
        write(imov,150) np,npt
        write(imov,180) ((coor(i,j),i=1,3),j=1,nj)
        write(imov,150) (jp(i),i=1,nedge)
        close(imov)
c
150     format(10i8)
180     format(1p6e12.5)
c
        return
        end
CNAME NODSUR.FOR
c*****************************************************************************
        subroutine nodsur(nx,ny,nz,x,y,z,maxd)
c*****************************************************************************
C
C  PROGRAM: NODSUR.FOR
C
C  PURPOSE: store the nodes data of six surfaces
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*****************************************************************************
c
c  called:
c
c*****************************************************************************
c
c  called by: orimsh.for -- find the original mesh
c
c*****************************************************************************
        include 'para.inc'
        dimension x(maxd,12),y(maxd,12),z(maxd 12)
        dimension coor(4,maxnod),node(maxsid,1  maxloop)
        common/auto/coor,node,last,unit
c
        nod = 0
c
c store nodes on two z direction surfaces
c
        do 10 k = 5,7,2
        do 10 j = 1,ny+1
          do 10 i = 1,nx+1
            nod = nod+1
            ratio = float(i-1)/float(nx)
            coor(1,nod) = x(j,k)+(x(j,k+1)-x(j,k))*ratio
            coor(2,nod) = y(j,k)+(y(j,k+1)-y(j,k))*ratio
            coor(3,nod) = z(j,k)+(z(j,k+1)-z(j,k))*ratio
10      continue
c
c store  nodes on two x direction surfaces
c
        do 20 k = 5,6
          do 20 j = 1,ny+1
            do 20 i = 1,nz+1
              nod = nod+1
              ratio = float(i-1)/float(nz)
              coor(1,nod) = x(j,k)+(x(j,k+2)-x(j,k))*ratio
              coor(2,nod) = y(j,k)+(y(j,k+2)-y(j,k))*ratio
              coor(3,nod) = z(j,k)+(z(j,k+2)-z(j,k))*ratio
20      continue
c
c store  nodes on two y direction surfaces
c
        do 30 k = 9,11,2
          do 30 j = 1,nz+1
            do 30 i = 1,nx+1
              nod = nod+1
              ratio = float(i-1)/float(nx)
              coor(1,nod) = x(j,k)+(x(j,k+1)-x(j,k))*ratio
              coor(2,nod) = y(j,k)+(y(j,k+1)-y(j,k))*ratio
              coor(3,nod) = z(j,k)+(z(j,k+1)-z(j,k))*ratio
30      continue
        last = nod
c
        return
```

```
      end
CNAME ORIMSH.FOR
c*********************************************************************
      subroutine orimsh(xg,yg,zg,nx,ny,nz)
c*********************************************************************
C
C  PROGRAM: ORIMSH.FOR
C
C  PURPOSE: store the nodes data for original mesh
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: store the original mesh data for check the
c      difference with automatic meshing
c
c*********************************************************************
c
c  called:
c      nodlin.for -- get the nodes for lines
c      nodsur.for -- get the nodes for surface
c      nodsto.for -- store all the nodes for graphic bucket
c
c*********************************************************************
c
c  called by: gtsoli.for -- prepare the information for meshing
c
c*********************************************************************
      dimension x(20,20,20),y(20,20,20),z(20,20,20)
      dimension xg(8),yg(8),zg(8),ick(8)
c
c store the line nodes
c
      call nodlin(xg,yg,zg,nx,ny,nz,x,y,z)
c
c store the surface nodes
c
      call nodsur(nx,ny,nz,x,y,z)
c
c store into the mesh for original mesh plot
c
      call nodsto(nx,ny,nz,x,y,z)
c
      return
      end
CNAME PREPAR.FOR
c*********************************************************************
      subroutine prepar(x,y,z,w,nodnum,unitt)
c*********************************************************************
C
C  PROGRAM: PREPAR.FOR
C
C  PURPOSE:  store the nodal data for volume meshing
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*********************************************************************
c
c  called by: gtsoli.for -- prepare the information for meshing
c
c*********************************************************************
      include 'para.inc'
      dimension x(maxsid,12),y(maxsid,12),z(maxsid,12),
     $          w(maxsid,12),nodnum(12)
      dimension coor(4,maxnod),node(maxsid,12,maxloop)
      common/auto/coor,node,last,unit
c
c store the coordinate and weight for each nodes
c
      k = 0
      do 10 j = 1,12
         if(j.le.4)then
            ibeg = 1
```

```
            iend = nodnum(j)
         else
            ibeg = 2
            iend = nodnum(j)-1
         end if
c
c  get the line nodal information for the first volume
c
         node(1,j,1) = nodnum(j)
         do 20 i = ibeg,iend
            k = k+1
            coor(1,k)=x(i,j)
            coor(2,k)=y(i,j)
            coor(3,k)=z(i,j)
            coor(4,k)=w(i,j)
            node(i+1,j,1) = k
20       continue
c
c  find the first and last node for rest of the lines
c
         if(j.gt.4)then
           do 30 i = 1,nodnum(j),nodnum(j)-1
             do 40 m = 1,4
             do 40 n = 1,nodnum(m),nodnum(m)-1
               delx = abs(x(i,j)-x(n,m))
               dely = abs(y(i,j)-y(n,m))
               delz = abs(z(i,j)-z(n,m))
               if(delx.lt.0.001 .and. dely.lt.0.001 .and.
     $                 delz.lt.0.001)then
                 node(i+1,j,1) = noce(n+1,m,1)
               end if
40           continue
30         continue
         end if
10     continue
       last = k
       unit = unitt
c
c store extra data for smoothing purpose
c
       do 50 j = 1,12
         do 60 i= 1,node(1,j,1)+1
           node(i,j,maxloop) = node(i,j,1)
60       continue
50     continue
c
       return
       end

CNAME READSO.FOR
c*********************************************************************
       subroutine readso(fname,nx,ny,nz,unit,xg,yg,zg,ierr)
c*********************************************************************
C
C  PROGRAM: READSO.FOR
C
C  PURPOSE:  read in the volume data by counter clockwise order
C
C  WRITTEN BY:  SSUTA HSU      CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                CHECKED BY:          DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*********************************************************************
c
c called by: gtsoli.for -- prepare volume information
c
c*********************************************************************
       dimension xg(8),yg(8),zg(8)
       character fname*(*)
c
c  read in the prime volume 8 points
c
       open(11,file=fname//'.sol',status='old',err=10)
       read(11,*,end=99)unit
       read(11,*)(xg(i),yg(i),zg(i),i=1,8)
c
c find the increments on each direction
```

```
c
      nx = nint(sqrt((xg(1)-xg(2))**2 + (yg(1)-yg(2))**2 +
     $          (zg(1)-zg(2))**2)/unit)
      if(nx.le.1) nx=1
      ny = nint(sqrt((xg(1)-xg(4))**2 + (yg(1)-yg(4))**2 +
     $          (zg(1)-zg(4))**2)/unit)
      if(ny.le.1) ny=1
      nz = nint(sqrt((xg(1)-xg(5))**2 + (yg(1)-yg(5))**2 +
     $          (zg(1)-zg(5))**2)/unit)
      if(nz.le.1) nz=1
      go to 99
c
10    ierr=1
99    close(11)
      return
      end
CNAME RENODE.FOR
c*****************************************************************************
      subroutine renode(nx,ny,nz,x,y,z,nodnum,dist,iflag)
C*****************************************************************************
C
C  PROGRAM: RENODE.FOR
C
C  PURPOSE:  re-mesh line for newly even number
C
C  WRITTEN BY:  SSUTA HSU        CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                  CHECKED BY:          DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION: IF THE CLOSE LOOP WAS AN ADD ONE TO MAKE IT EVEN
C       LOOP THEN RE-ARRANGE THE NODAL LOCATION
c
c*****************************************************************************
c
c  CALLED BY:MSHLIN.FOR
C
c*****************************************************************************
      include 'para.inc'
      dimension x(maxsid,12),y(maxsid,12),z(maxsid,12)
      dimension xb(maxsid),yb(maxsid),zb(maxsid),
     $          nodnum(12),dist(12),iflag(12)
c
      do 20 j=1,12
      if(iflag(j).eq.1)then
        sum=0.0
        nnod=0
        gdis=0.0
        fdis=dist(j)/(nodnum(j)-1)
        do 10 i=1,nodnum(j)-1
          if(i.gt.1)then
            delt=dist(j)/(nodnum(j)-2)
            sum=delt*(i-1)
          else
            delt = 1.e10
          end if
11        if(gdis.le.sum+.001)then
            nnod=nnod+1
            diff=sum-gdis
            xb(nnod) = x(i,j) - (x(i,j)-x(i-1,j))/delt*diff
            yb(nnod) = y(i,j) - (y(i,j)-y(i-1,j))/delt*diff
            zb(nnod) = z(i,j) - (z(i,j)-z(i-1,j))/delt*diff
            gdis=gdis+fdis
            go to 11
          end if
10      continue
        nodnum(j)=nnod
c
c  restore the line node data
          do 30 i=1,nnod
            x(i,j)=xb(i)
            y(i,j)=yb(i)
            z(i,j)=zb(i)
30        continue
        end if
20      continue
c
        return
        end
CNAME SMAREA.FOR
```

```
c*****************************************************************
      subroutine smarea(id,coox,cooy,cooz,area)
C*****************************************************************
C
C  PROGRAM: SMAREA.FOR
C
C  PURPOSE: FIND THE AREA FOR THE ELEMENT
C
C  WRITTEN BY:  SSUTA HSU        CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                  CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: FIND THE AREA BY TRIANGULAR METHOD
c
c*****************************************************************
c  called by:
c       smooth.for -- mesh smoother
c
c*****************************************************************
      include 'para.inc'
      common/auto/coor,node,last,unit
      common/fin/ncon(0:10,maxid)
      dimension node(maxsid,12,maxloop),coor(4,maxnod),cood(3,20)
c
c...find the center vector
      coox = (coor(1,ncon(0,id)) + coor(1,ncon(1,id)) +
     $         coor(1,ncon(2,id)) + coor(1,ncon(3,id)))/4.0
      cooy = (coor(2,ncon(0,id)) + coor(2,ncon(1,id)) +
     $         coor(2,ncon(2,id)) + coor(2,ncon(3,id)))/4.0
      cooz = (coor(3,ncon(0,id)) + coor(3,ncon(1,id)) +
     $         coor(3,ncon(2,id)) + coor(3,ncon(3,id)))/4.0
c...find the area for that surface
      a = sqrt((coor(1,ncon(0,id))-coor(1,ncon(1,id)))**2 +
     $         (coor(2,ncon(0,id))-coor(2,ncon(1,id)))**2 +
     $         (coor(3,ncon(0,id))-coor(3,ncon(1,id)))**2)
      b = sqrt((coor(1,ncon(0,id))-coor(1,ncon(3,id)))**2 +
     $         (coor(2,ncon(0,id))-coor(2,ncon(3,id)))**2 +
     $         (coor(3,ncon(0,id))-coor(3,ncon(3,id)))**2)
      c = sqrt((coor(1,ncon(3,id))-coor(1,ncon(1,id)))**2 +
     $         (coor(2,ncon(3,id))-coor(2,ncon(1,id)))**2 +
     $         (coor(3,ncon(3,id))-coor(3,ncon(1,id)))**2)
      d = sqrt((coor(1,ncon(2,id))-coor(1,ncon(1,id)))**2 +
     $         (coor(2,ncon(2,id))-coor(2,ncon(1,id)))**2 +
     $         (coor(3,ncon(2,id))-coor(3,ncon(1,id)))**2)
      e = sqrt((coor(1,ncon(3,id))-coor(1,ncon(2,id)))**2 +
     $         (coor(2,ncon(3,id))-coor(2,ncon(2,id)))**2 +
     $         (coor(3,ncon(3,id))-coor(3,ncon(2,id)))**2)
      s1 = (a + b + c)/2.0
      s2 = (c + d + e)/2.0
      area1 = sqrt(s1*(s1-a)*(s1-b)*(s1-c))
      area2 = sqrt(s2*(s2-d)*(s2-e)*(s2-c))
      area = area1 + area2
c
      return
      end
CNAME SMOOTH.FOR
c*****************************************************************
      subroutine smooth
C*****************************************************************
C
C  PROGRAM: SMOOTH.FOR
C
C  PURPOSE: SMOOTHING THE ELEMENTS FOR BETTER VIEW
C
C  WRITTEN BY:  SSUTA HSU        CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                  CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: optimize the node coordinate for best hexahedron
c       element by area pull method( modifed Laplacian method)
c
c*****************************************************************
c  called:
c       smarea.for -- find the area for the element
c
c*****************************************************************
c  called by:
c       vmesh.for -- volume mesh controler
```

```
c
c*************************************************************************
      include 'para.inc'
      dimension x(0:maxdeg,maxid),y(0:maxdeg,maxid),
     $          z(0:maxdeg,maxid),ideg(maxid)
      common/data/x,y,z,ideg
      dimension node(maxsid,12,maxloop),coor(4,maxnod),isurf(maxid/4)
      common/auto/coor,node,last,unit
      common/fin/ncon(0:10,maxid)
      common/oldln/nnum(20,maxsid*6),lold(maxsid*6),nall
      common/bsur/isurf,icont
      logical found
c
      write(*,*)' Smoothing now please wait...'
      do 9 itr = 1,2
       do 10 nod = nnum(1,1),last
          sumx = 0.0
          sumy = 0.0
          sumz = 0.0
          suma = 0.0
          found = .false.
c
c check to see if the node was in boundary(and transition) surface or not
c
          do 20 i = 1,icont
            do 30 j = 0,3
              if(nod.eq.ncon(j,isurf(i)))then
                found = .true.
                go to 50
              end if
30          continue
20        continue
c
c if the node was in boundary surfaces, but not in boundary lines then do
c the surface smoothing, otherwise do the solid smoothing
c
50        continue
          if(found)then
            do 60 j = 1,12
            do 60 i = 1,node(1,j,maxloop)
               if(nod.eq.node(i+1,j,maxloop))go to 10
60          continue
            do 70 id = 1,maxid
              if(ncon(0,id).le.0)go to 120
c...check if the element is on the boundary surface or not
              if(nod.eq.ncon(0,id) .or. nod.eq.ncon(1,id) .or.
     $           nod.eq.ncon(2,id) .or. nod.eq.ncon(3,id))then
                do 90 im = 1,icont
                  if(id .eq. isurf(im))then
                     call smarea(id,coox,cooy,cooz,area)
                     sumx = sumx + area*coox
                     sumy = sumy + area*cooy
                     sumz = sumz + area*cooz
                     suma = suma + area
                  end if
90              continue
               end if
70          continue
          else
c.... do the volume smoothing
            do 100 id = 1,maxid
              if(ncon(0,id).le.0)go to 120
              do 110 nc = 0,3
                if(nod.eq.ncon(nc,id))then
                  call smarea(id,coox,cooy,cooz,area)
                  sumx = sumx + area*coox
                  sumy = sumy + area*cooy
                  sumz = sumz + area*cooz
                  suma = suma + area
                end if
110           continue
100         continue
        end if
120     continue
        if(suma.le.0)go to 10
        coor(1,nod) = sumx/suma
        coor(2,nod) = sumy/suma
        coor(3,nod) = sumz/suma
10    continue
9     continue
```

```
c
c store the coordinate into common block for plot
c
        iele = 1
        do while (ideg(iele).ne.0)
           do 130 i = 0,ideg(iele)
              x(i,iele)=coor(1,ncon(i,iele))
              y(i,iele)=coor(2,ncon(i,iele))
              z(i,iele)=coor(3,ncon(i,iele))
130        continue
           iele = iele+1
        end do
c
        return
        end
CNAME SURBND.FOR
c*********************************************************************
        subroutine surbnd
c*********************************************************************
        include 'para.inc'
        common/auto/coor,node,last,unit
        common/fin/ncon(0:10,maxid)
        common/oldln/nnum(20,maxsid*6),lold(maxsid*6),nall
        common/tran/ntran(1000),kall
        common/bsur/isurf,icont
        dimension coor(4,maxnod),node(maxsid,12,maxloop),isurf(maxid/4)
        logical lck(0:3),coin(0:3)
c
c check to see if the node was in boundary(and transition) surface or not
c
        icont = 0
        do 10 id = 1,maxid
          if(ncon(0,id).le.0)go to 100
          do 20 i = 0,3
            lck(i) = .false.
20        continue
            do 30 k = 0,3
c...check boundary line nodes
              do 32 i = 1,12
              do 32 j = 2,node(1,i,maxloop)+1
                if(ncon(k,id).eq.node(j,i,maxloop))then
                  lck(k) = .true.
                  go to 31
                end if
32            continue
c...check boundary surface nodes
              do 40 i = 1,nall
              do 40 j = 1,lold(i)
                if(ncon(k,id).eq.nnum(j,i))then
                  lck(k) = .true.
                  go to 31
                end if
40            continue
c...check boundary transition nodes
              do 50 i = 1,kall
                if(ncon(k,id).eq.ntran(i))then
                   lck(k) = .true.
                   go to 31
                end if
50            continue
31            continue
              if(.not.lck(k))go to 10
30          continue
c
c if the all the nodes were on boundary surfaces then store that surface id
c but check for the coincident surface which is a internal surface between
c two boundary shell
c
            if(lck(0) .and. lck(1) .and. lck(2) .and. lck(3))then
               do 60 i = 1,icont
                 do 80 im = 0,3
                   coin(im) = .false.
                   do 90 in = 0,3
                     if(ncon(im,id).eq.ncon(in,isurf(i)))then
                       coin(im) = .true.
                     end if
90                 continue
                   if(.not.coin(im))go to 60
80               continue
                 if(coin(0) .and. coin(1) .and. coin(2) .and. coin(3))then
```

```
                    do 70 k = i,icont
                      isurf(k) = isurf(k+1)
70                  continue
                    icont = icont -1
                    go to 10
                  end if
80              continue
                icont = icont + 1
                isurf(icont) = id
              end if
10        continue
c
100       return
          end

CNAME TRAN1.FOR
c*****************************************************************************
      subroutine tran1(loop,iter,jt,iflag)
C*****************************************************************************
C
C  PROGRAM: TRAN1.FOR
C
C  PURPOSE: DO TRANSITION BY #1 METHOD
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:              DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:              DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION: do transition on only one line
c
c*****************************************************************************
c
c  called:
c       cktran  -- check the nodes on the line to see if the node have
c                  been used or not
c       tran11  -- do the transition on line 1,5,9.
c       tran12  -- do the transition on line 2,6,10.
c       tran13  -- do the transition on line 3,7,11.
c       tran14  -- do the transition on line 4,8,12.
c       push    -- store the data into new loop
c
c*****************************************************************************
c
c  called by: transi.for -- transition flag checker
c
c*****************************************************************************
      include 'para.inc'
      dimension coor(4,maxnod),node(maxsid,12,maxloop),kt(6)
      common/auto/coor,node,last,unit
c
      if(iflag.eq.11)then
         jn = jt + 1
         jm = jt + 2
         jo = jt + 3
      else if(iflag.eq.12)then
         jn = jt + 2
         jm = jt - 1
         jo = jt + 1
      else if(iflag.eq.13)then
         jn = jt + 1
         jm = jt - 2
         jo = jt - 1
      else if(iflag.eq.14)then
         jn = jt - 2
         jm = jt - 1
         jo = jt - 3
      end if
      kt(1) = last + 1
      njt = node(1,jt,loop)
      do 10 i = 1,4
         coor(i,kt(1)) =((coor(i,node(2,jn,loop)) +
     $                    coor(i,node(3,jn,loop)))/2.0 +
     $                    coor(i,node(3,jt,loop)))/2.0
10    continue
      call cktran(coor(1,kt(1)),kt(1))
      kt(2) = last + 1
      do 20 i = 1,4
         coor(i,kt(2)) =((coor(i,node(2,jn,loop)) +
     $                    coor(i,node(3,jn,loop)))/2.0 +
```

```
     $                    coor(i,node(njt,jt,loop)))/2.0
 20    continue
       call cktran(coor(1,kt(2)),kt(2))
       kt(3) = last + 1
       do 30 i = 1,4
          coor(i,kt(3)) =((coor(i,node(2,jm,loop)) +
     $                    coor(i,node(3,jm,loop)))/2.0 +
     $                    coor(i,node(3,jt,loop)))/2.0
 30    continue
       call cktran(coor(1,kt(3)),kt(3))
       kt(4) = last + 1
       do 40 i = 1,4
          coor(i,kt(4)) =((coor(i,node(2,jm,loop)) +
     $                    coor(i,node(3,jm,loop)))/2.0 +
     $                    coor(i,node(njt,jt,loop)))/2.0
 40    continue
       call cktran(coor(1,kt(4)),kt(4))
       kt(5) = last + 1
       last = kt(5)
       do 50 i = 1,4
          coor(i,kt(5)) =((coor(i,node(2,jo,loop)) +
     $                    coor(i,node(3,jo,loop)))/2.0 +
     $                    coor(i,node(3,jt,loop)))/2.0
 50    continue
       kt(6) = last + 1
       last = kt(6)
       do 60 i = 1,4
          coor(i,kt(6)) =((coor(i,node(2,jo,loop)) +
     $                    coor(i,node(3,jo,loop)))/2.0 +
     $                    coor(i,node(njt,jt,loop)))/2.0
 60    continue
       if(iflag.eq.11)then
          call tran11(loop,jt,jn,jm,jo,kt)
       else if(iflag.eq.12)then
          call tran12(loop,jt,jn,jm,jo,kt)
       else if(iflag.eq.13)then
          call tran13(loop,jt,jn,jm,jo,kt)
       else if(iflag.eq.14)then
          call tran14(loop,jt,jn,jm,jo,kt)
       end if
       call push(loop,iter)
       do 70 i = 1,4
          loop = loop+1
          call push(loop,iter)
 70    continue
c
       return
       end

CNAME TRAN11.FOR
C*********************************************************************
       subroutine tran11(loop,jt,jn,jm,jo,kt)
C*********************************************************************
C
C  PROGRAM: TRAN11.FOR
C
C  PURPOSE: do the transition by # 11 method
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: transition on line 1,5,9
c
c*********************************************************************
c
c  called:  gtside -- given the two opposite lines on a surface then
c                     get the other two lines.
c
c*********************************************************************
c
c  called by: tran1.for -- do transition on a line
c
c*********************************************************************
       include 'para.inc'
       dimension coor(4,maxnod),node(maxsid,12,maxloop),kt(6)
       common/auto/coor,node,last,unit

c...store them into 5 new loop
```

```
      loop2 = loop + 1
      loop3 = loop + 2
      loop4 = loop + 3
      loop5 = loop + 4
c
c store the loop 5, the left side
c
      do 10 j = 1,12
         node(1,j,loop5) = 2
10    continue
c store line 1
      node(2,jt,loop5) = node(2,jt,loop)
      node(3,jt,loop5) = node(3,jt,loop)
c store line 2
      node(2,jn,loop5) = node(2,jn,loop)
      node(3,jn,loop5) = kt(1)
c store line 3
      node(2,jm,loop5) = node(2,jm,loop)
      node(3,jm,loop5) = kt(3)
c store line 4
      node(2,jo,loop5) = node(2,jo,loop)
      node(3,jo,loop5) = kt(5)
c store line 5 & 6
      call gtside(jt,jn,left,nrit)
      do 11 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
11    continue
      node(2,nrit,loop5) = node(3,jt,loop)
      node(3,nrit,loop5) = kt(1)
c store line 7 & 8
      call gtside(jm,jo,left,nrit)
      do 12 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
12    continue
      node(2,nrit,loop5) = kt(3)
      node(3,nrit,loop5) = kt(5)
c store line 9 & 10
      call gtside(jt,jm,left,nrit)
      do 13 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
13    continue
      node(2,nrit,loop5) = node(3,jt,loop)
      node(3,nrit,loop5) = kt(3)
c store line 11 & 12
      call gtside(jn,jo,left,nrit)
      do 14 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
14    continue
      node(2,nrit,loop5) = kt(1)
      node(3,nrit,loop5) = kt(5)
c
c store the loop 4, the back side
c
      do 20 j = 1,12
         node(1,j,loop4) = 2
20    continue
c store line 1
      node(2,jt,loop4) = kt(3)
      node(3,jt,loop4) = kt(4)
c store line 2
      node(2,jn,loop4) = kt(5)
      node(3,jn,loop4) = kt(6)
c store line 3
      do 21 i = 1,node(1,jm,loop)+1
         node(i,jm,loop4) = node(i,jm,loop)
21    continue
c store line 4
      do 22 i = 1,node(1,jo,loop)+1
         node(i,jo,loop4) = node(i,jo,loop)
22    continue
c store line 5 & 6
      call gtside(jt,jn,left,nrit)
      node(2,left,loop4) = kt(3)
      node(3,left,loop4) = kt(5)
      node(2,nrit,loop4) = kt(4)
      node(3,nrit,loop4) = kt(6)
c store line 7 & 8
      call gtside(jm,jo,left,nrit)
      do 23 i = 1,node(1,left,loop)+1
         node(i,left,loop4) = node(i,left,loop)
```

```
23      continue
        do 24 i = 1,node(1,nrit,loop)+1
           node(i,nrit,loop4) = node(i,nrit,loop)
24      continue
c store line 9 & 10
        call gtside(jt,jm,left,nrit)
        node(2,left,loop4) = kt(3)
        node(3,left,loop4) = node(node(1,left,loop)+1,left,loop)
        node(2,nrit,loop4) = kt(4)
        node(3,nrit,loop4) = node(node(1,nrit,loop)+1,nrit,loop)
c store line 11 & 12
        call gtside(jn,jo,left,nrit)
        node(2,left,loop4) = kt(5)
        node(3,left,loop4) = node(node(1,left,loop)+1,left,loop)
        node(2,nrit,loop4) = kt(6)
        node(3,nrit,loop4) = node(node(1,nrit,loop)+1,nrit,loop)
c
c store the loop 3, the top side
c
        do 30 j = 1,12
           node(1,j,loop3) = 2
30      continue
c store line 1
        node(2,jt,loop3) = kt(1)
        node(3,jt,loop3) = kt(2)
c store line 2
        do 31 i = 1,node(1,jn,loop)+1
           node(i,jn,loop3) = node(i,jn,loop)
31      continue
c store line 3
        node(2,jm,loop3) = kt(5)
        node(3,jm,loop3) = kt(6)
c store line 4
        do 32 i = 1,node(1,jo,loop)+1
           node(i,jo,loop3) = node(i,jo,loop)
32      continue
c store line 5 & 6
        call gtside(jt,jn,left,nrit)
        node(2,left,loop3) = kt(1)
        node(3,left,loop3) = node(node(1,left,loop)+1,left,loop)
        node(2,nrit,loop3) = kt(2)
        node(3,nrit,loop3) = node(node(1,nrit,loop)+1,nrit,loop)
c store line 7 & 8
        call gtside(jm,jo,left,nrit)
        node(2,left,loop3) = kt(5)
        node(3,left,loop3) = node(node(1,left,loop)+1,left,loop)
        node(2,nrit,loop3) = kt(6)
        node(3,nrit,loop3) = node(node(1,nrit,loop)+1,nrit,loop)
c store line 9 & 10
        call gtside(jt,jm,left,nrit)
        node(2,left,loop3) = kt(1)
        node(3,left,loop3) = kt(5)
        node(2,nrit,loop3) = kt(2)
        node(3,nrit,loop3) = kt(6)
c store line 11 & 12
        call gtside(jn,jo,left,nrit)
        do 33 i = 1,node(1,left,loop)+1
           node(i,left,loop3) = node(i,left,loop)
33      continue
        do 34 i = 1,node(1,nrit,loop)+1
           node(i,nrit,loop3) = node(i,nrit,loop)
34      continue
c
c store the loop 2, the middle bottom side
c
        do 40 j = 2,12
          node(1,j,loop2) = 2
40      continue
c store line 1
        num = node(1,jt,loop)
        node(1,jt,loop2) = num-2
        do 50 j = 1,num-2
           node(j+1,jt,loop2) = node(j+2,jt,loop)
50      continue
c store line 2
        node(2,jn,loop2) = kt(1)
        node(3,jn,loop2) = kt(2)
c store line 3
        node(2,jm,loop2) = kt(3)
        node(3,jm,loop2) = kt(4)
```

```
c store line 4
      node(2,jo,loop2) = kt(5)
      node(3,jo,loop2) = kt(8)
c store line 5 & 6
      call gtside(jt,jn,left,nrit)
      node(2,left,loop2) = node(3,jt,loop)
      node(3,left,loop2) = kt(1)
      node(2,nrit,loop2) = node(num,jt,loop)
      node(3,nrit,loop2) = kt(2)
c store line 7 & 8
      call gtside(jm,jo,left,nrit)
      node(2,left,loop2) = kt(3)
      node(3,left,loop2) = kt(5)
      node(2,nrit,loop2) = kt(4)
      node(3,nrit,loop2) = kt(8)
c store line 9 & 10
      call gtside(jt,jm,left,nrit)
      node(2,left,loop2) = node(3,jt,loop)
      node(3,left,loop2) = kt(3)
      node(2,nrit,loop2) = node(num,jt,loop)
      node(3,nrit,loop2) = kt(4)
c store line 11 & 12
      call gtside(jn,jo,left,nrit)
      node(2,left,loop2) = kt(1)
      node(3,left,loop2) = kt(5)
      node(2,nrit,loop2) = kt(2)
      node(3,nrit,loop2) = kt(8)
c
c store the loop 1, the right side
c
c store line 1
      node(1,jt,loop) = 2
      node(2,jt,loop) = node(num,jt,loop)
      node(3,jt,loop) = node(num+1,jt,loop)
c store line 2, skip 2nd node
      node(1,jn,loop) = 2
      node(2,jn,loop) = kt(2)
c store line 3, skip 2nd node
      node(1,jm,loop) = 2
      node(2,jm,loop) = kt(4)
c store line 4, skip 2nd node
      node(1,jo,loop) = 2
      node(2,jo,loop) = kt(8)
c store line 5,skip 8
      call gtside(jt,jn,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = node(num,jt,loop)
      node(3,left,loop) = kt(2)
c store line 7 ,skip 8
      call gtside(jm,jo,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(4)
      node(3,left,loop) = kt(8)
c store line 9, skip 10
      call gtside(jt,jm,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = node(num,jt,loop)
      node(3,left,loop) = kt(4)
c store line 11, skip 12
      call gtside(jn,jo,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(2)
      node(3,left,loop) = kt(8)
c
      return
      end

CNAME TRAN12.FOR
C*********************************************************************
      subroutine tran12(loop,jt,jn,jm,jo,kt)
C*********************************************************************
C
C  PROGRAM: TRAN12.FOR
C
C  PURPOSE:  do the transition by # 12 method
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*********************************************************************
```

```
C
C  ALGORITHM EXPLANATION: transition on line 2,6,10
c
c*****************************************************************
c
c  called:  gtside -- given the two opposite lines on a surface then
c                     get the other two lines.
c
c*****************************************************************
c
c called by: tran1.for -- #1 trasition controler
c
c*****************************************************************
      include 'para.inc'
      dimension coor(4,maxnod),node(maxsid,12,maxloop),kt(6)
      common/auto/coor,node,last,unit

c...store them into 5 new loop
      loop2 = loop + 1
      loop3 = loop + 2
      loop4 = loop + 3
      loop5 = loop + 4
c
c store the loop 5, the left side
c
      do 10 j = 1,12
         node(1,j,loop5) = 2
10    continue
c store line 1
      node(2,jm,loop5) = node(2,jm,loop)
      node(3,jm,loop5) = kt(3)
c store line 2
      node(2,jt,loop5) = node(2,jt,loop)
      node(3,jt,loop5) = node(3,jt,loop)
c store line 3
      node(2,jo,loop5) = node(2,jo,loop)
      node(3,jo,loop5) = kt(5)
c store line 4
      node(2,jn,loop5) = node(2,jn,loop)
      node(3,jn,loop5) = kt(1)
c store line 5 & 6
      call gtside(jm,jt,left,nrit)
      do 11 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
11    continue
      node(2,nrit,loop5) = kt(3)
      node(3,nrit,loop5) = node(3,jt,loop)
c store line 7 & 8
      call gtside(jo,jn,left,nrit)
      do 12 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
12    continue
      node(2,nrit,loop5) = kt(5)
      node(3,nrit,loop5) = kt(1)
c store line 9 & 10
      call gtside(jm,jo,left,nrit)
      do 13 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
13    continue
      node(2,nrit,loop5) = kt(3)
      node(3,nrit,loop5) = kt(5)
c store line 11 & 12
      call gtside(jt,jn,left,nrit)
      do 14 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
14    continue
      node(2,nrit,loop5) = node(3,jt,loop)
      node(3,nrit,loop5) = kt(1)
c
c store the loop 4, the back side
c
      do 20 j = 1,12
         node(1,j,loop4) = 2
20    continue
c store line 1
      node(2,jm,loop4) = kt(5)
      node(3,jm,loop4) = kt(6)
c store line 2
      node(2,jt,loop4) = kt(1)
      node(3,jt,loop4) = kt(2)
```

```
c store line 3
      do 21 i = 1,node(1,jo,loop)+1
         node(i,jo,loop4) = node(i,jo,loop)
21    continue
c store line 4
      do 22 i = 1,node(1,jn,loop)+1
         node(i,jn,loop4) = node(i,jn,loop)
22    continue
c store line 5 & 6
      call gtside(jm,jt,left,nrit)
      node(2,left,loop4) = kt(5)
      node(3,left,loop4) = kt(1)
      node(2,nrit,loop4) = kt(6)
      node(3,nrit,loop4) = kt(2)
c store line 7 & 8
      call gtside(jo,jn,left,nrit)
      do 23 i = 1,node(1,left,loop)+1
         node(i,left,loop4) = node(i,left,loop)
23    continue
      do 24 i = 1,node(1,nrit,loop)+1
         node(i,nrit,loop4) = node(i,nrit,loop)
24    continue
c store line 9 & 10
      call gtside(jm,jo,left,nrit)
      node(2,left,loop4) = kt(5)
      node(3,left,loop4) = node(node(1,left,loop)+1,left,loop)
      node(2,nrit,loop4) = kt(6)
      node(3,nrit,loop4) = node(node(1,nrit,loop)+1,nrit,loop)
c store line 11 & 12
      call gtside(jt,jn,left,nrit)
      node(2,left,loop4) = kt(1)
      node(3,left,loop4) = node(node(1,left,loop)+1,left,loop)
      node(2,nrit,loop4) = kt(2)
      node(3,nrit,loop4) = node(node(1,nrit,loop)+1,nrit,loop)
c
c store the loop 3, the bottom side
c
      do 30 j = 1,12
         node(1,j,loop3) = 2
30    continue
c store line 1
      do 31 i = 1,node(1,jm,loop)+1
         node(i,jm,loop3) = node(i,jm,loop)
31    continue
c store line 2
      node(2,jt,loop3) = kt(3)
      node(3,jt,loop3) = kt(4)
c store line 3
      do 32 i = 1,node(1,jo,loop)+1
         node(i,jo,loop3) = node(i,jo,loop)
32    continue
c store line 4
      node(2,jn,loop3) = kt(5)
      node(3,jn,loop3) = kt(6)
c store line 5 & 6
      call gtside(jm,jt,left,nrit)
      node(2,left,loop3) = node(2,left,loop)
      node(3,left,loop3) = kt(3)
      node(2,nrit,loop3) = node(2,nrit,loop)
      node(3,nrit,loop3) = kt(4)
c store line 7 & 8
      call gtside(jo,jn,left,nrit)
      node(2,left,loop3) = node(2,left,loop)
      node(3,left,loop3) = kt(5)
      node(2,nrit,loop3) = node(2,nrit,loop)
      node(3,nrit,loop3) = kt(6)
c store line 9 & 10
      call gtside(jm,jo,left,nrit)
      do 33 i = 1,node(1,left,loop)+1
         node(i,left,loop3) = node(i,left,loop)
33    continue
      do 34 i = 1,node(1,nrit,loop)+1
         node(i,nrit,loop3) = node(i,nrit,loop)
34    continue
c store line 11 & 12
      call gtside(jt,jn,left,nrit)
      node(2,left,loop3) = kt(3)
      node(3,left,loop3) = kt(5)
      node(2,nrit,loop3) = kt(4)
      node(3,nrit,loop3) = kt(6)
```

```
c
c store the loop 2, the middle top side
c
      do 40 j = 1,12
        node(1,j,loop2) = 2
40    continue
c store line 1
      node(2,jm,loop2) = kt(3)
      node(3,jm,loop2) = kt(4)
c store line 2
      num = node(1,jt,loop)
      node(1,jt,loop2) = num-2
      do 50 j = 1,num-2
         node(j+1,jt,loop2) = node(j+2,jt,loop)
50    continue
c store line 3
      node(2,jo,loop2) = kt(5)
      node(3,jo,loop2) = kt(6)
c store line 4
      node(2,jn,loop2) = kt(1)
      node(3,jn,loop2) = kt(2)
c store line 5 & 6
      call gtside(jm,jt,left,nrit)
      node(2,left,loop2) = kt(3)
      node(3,left,loop2) = node(3,jt,loop)
      node(2,nrit,loop2) = kt(4)
      node(3,nrit,loop2) = node(num,jt,loop)
c store line 7 & 8
      call gtside(jo,jn,left,nrit)
      node(2,left,loop2) = kt(5)
      node(3,left,loop2) = kt(1)
      node(2,nrit,loop2) = kt(6)
      node(3,nrit,loop2) = kt(2)
c store line 9 & 10
      call gtside(jm,jo,left,nrit)
      node(2,left,loop2) = kt(3)
      node(3,left,loop2) = kt(5)
      node(2,nrit,loop2) = kt(4)
      node(3,nrit,loop2) = kt(6)
c store line 11 & 12
      call gtside(jt,jn,left,nrit)
      node(2,left,loop2) = node(3,jt,loop)
      node(3,left,loop2) = kt(1)
      node(2,nrit,loop2) = node(num,jt,loop)
      node(3,nrit,loop2) = kt(2)
c
c store the loop 1, the right side
c
c store line 1
      node(1,jm,loop) = 2
      node(2,jm,loop) = kt(4)
c store line 2
      node(1,jt,loop) = 2
      node(2,jt,loop) = node(num,jt,loop)
      node(3,jt,loop) = node(num+1,jt,loop)
c store line 3
      node(1,jo,loop) = 2
      node(2,jo,loop) = kt(6)
c  store line 4
      node(1,jn,loop) = 2
      node(2,jn,loop) = kt(2)
c store line 5,skip 6
      call gtside(jm,jt,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(4)
      node(3,left,loop) = node(num,jt,loop)
c store line 7 ,skip 8
      call gtside(jo,jn,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(6)
      node(3,left,loop) = kt(2)
c store line 9, skip 10
      call gtside(jm,jo,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(4)
      node(3,left,loop) = kt(6)
c store line 11, skip 12
      call gtside(jt,jn,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = node(num,jt,loop)
```

```
      node(3,left,loop) = kt(2)
c
      return
      end

CNAME TRAN13.FOR
C-----------------------------------------------------------------------
      subroutine tran13(loop,jt,jn,jm,jo,kt)
C-----------------------------------------------------------------------
C
C  PROGRAM: TRAN13.FOR
C
C  PURPOSE: do the transition by # 13 method
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C-----------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION: transition on line 3,7,11
c
c-----------------------------------------------------------------------
c
c  called:  gtside -- given the two opposite lines on a surface then
c                     get the other two lines.
c
c-----------------------------------------------------------------------
c
c  called by: tran1.for -- #1 transition controler
c
c-----------------------------------------------------------------------
      include 'para.inc'
      dimension coor(4,maxnod),node(maxsid,12,maxloop),kt(8)
      common/auto/coor,node,last,unit

c...store them into 5 new loop
      loop2 = loop + 1
      loop3 = loop + 2
      loop4 = loop + 3
      loop5 = loop + 4
c
c store the loop 5, the left side
c
      do 10 j = 1,12
         node(1,j,loop5) = 2
10    continue
c store line 1
      node(2,jm,loop5) = node(2,jm,loop)
      node(3,jm,loop5) = kt(3)
c store line 2
      node(2,jo,loop5) = node(2,jo,loop)
      node(3,jo,loop5) = kt(5)
c store line 3
      node(2,jt,loop5) = node(2,jt,loop)
      node(3,jt,loop5) = node(3,jt,loop)
c store line 4
      node(2,jn,loop5) = node(2,jn,loop)
      node(3,jn,loop5) = kt(1)
c store line 5 & 8
      call gtside(jm,jo,left,nrit)
      do 11 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
11    continue
      node(2,nrit,loop5) = kt(3)
      node(3,nrit,loop5) = kt(5)
c store line 7 & 8
      call gtside(jt,jn,left,nrit)
      do 12 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
12    continue
      node(2,nrit,loop5) = node(3,jt,loop)
      node(3,nrit,loop5) = kt(1)
c store line 9 & 10
      call gtside(jm,jt,left,nrit)
      do 13 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
13    continue
      node(2,nrit,loop5) = kt(3)
      node(3,nrit,loop5) = node(3,jt,loop)
c store line 11 & 12
```

```
      call gtside(jo,jn,left,nrit)
      do 14 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
14    continue
      node(2,nrit,loop5) = kt(5)
      node(3,nrit,loop5) = kt(1)
c
c store the loop 4, the front side
c
      do 20 j = 1,12
         node(1,j,loop4) = 2
20    continue
c store line 1,2
      do 21 i = 1,node(1,jm,loop)+1
         node(i,jm,loop4) = node(i,jm,loop)
21    continue
      do 22 i = 1,node(1,jo,loop)+1
         node(i,jo,loop4) = node(i,jo,loop)
22    continue
c store line 3
      node(2,jt,loop4) = kt(3)
      node(3,jt,loop4) = kt(4)
c store line 4
      node(2,jn,loop4) = kt(5)
      node(3,jn,loop4) = kt(6)
c store line 5 & 6
      call gtside(jm,jo,left,nrit)
      do 23 i = 1,node(1,left,loop)+1
         node(i,left,loop4) = node(i,left,loop)
23    continue
      do 24 i = 1,node(1,nrit,loop)+1
        node(i,nrit,loop4) = node(i,nrit,loop)
24    continue
c store line 7 & 8
      call gtside(jt,jn,left,nrit)
      node(2,left,loop4) = kt(3)
      node(3,left,loop4) = kt(5)
      node(2,nrit,loop4) = kt(4)
      node(3,nrit,loop4) = kt(6)
c store line 9 & 10
      call gtside(jm,jt,left,nrit)
      node(2,left,loop4) = node(2,left,loop)
      node(3,left,loop4) = kt(3)
      node(2,nrit,loop4) = node(2,nrit,loop)
      node(3,nrit,loop4) = kt(4)
c store line 11 & 12
      call gtside(jo,jn,left,nrit)
      node(2,left,loop4) = node(2,left,loop)
      node(3,left,loop4) = kt(5)
      node(2,nrit,loop4) = node(2,nrit,loop)
      node(3,nrit,loop4) = kt(6)
c
c store the loop 3, the top side
c
      do 30 j = 1,12
         node(1,j,loop3) = 2
30    continue
c store line 1
      node(2,jm,loop3) = kt(5)
      node(3,jm,loop3) = kt(6)
c store line 2
      do 31 i = 1,node(1,jo,loop)+1
        node(i,jo,loop3) = node(i,jo,loop)
31    continue
c store line 3
      node(2,jt,loop3) = kt(1)
      node(3,jt,loop3) = kt(2)
c store line 4
      do 32 i = 1,node(1,jn,loop)+1
        node(i,jn,loop3) = node(i,jn,loop)
32    continue
c store line 5 & 6
      call gtside(jm,jo,left,nrit)
      node(2,left,loop3) = kt(5)
      node(3,left,loop3) = node(node(1,left,loop)+1,left,loop)
      node(2,nrit,loop3) = kt(6)
      node(3,nrit,loop3) = node(node(1,nrit,loop)+1,nrit,loop)
c store line 7 & 8
      call gtside(jt,jn,left,nrit)
      node(2,left,loop3) = kt(1)
```

```
      node(3,left,loop3) = node(node(1,left,loop)+1,left,loop)
      node(2,nrit,loop3) = kt(2)
      node(3,nrit,loop3) = node(node(1,nrit,loop)+1,nrit,loop)
c store line 9 & 10
      call gtside(jm,jt,left,nrit)
      node(2,left,loop3) = kt(5)
      node(3,left,loop3) = kt(1)
      node(2,nrit,loop3) = kt(8)
      node(3,nrit,loop3) = kt(2)
c store line 11 & 12
      call gtside(jo,jn,left,nrit)
      do 33 i = 1,node(1,left,loop)+1
        node(i,left,loop3) = node(i,left,loop)
33    continue
      do 34 i = 1,node(1,nrit,loop)+1
        node(i,nrit,loop3) = node(i,nrit,loop)
34    continue
c
c store the loop 2, the middle bottom side
c
      do 40 j = 1,12
        node(1,j,loop2) = 2
40    continue
c store line 1
      node(2,jm,loop2) = kt(3)
      node(3,jm,loop2) = kt(4)
c store line 2
      node(2,jo,loop2) = kt(5)
      node(3,jo,loop2) = kt(8)
c store line 3
      num = node(1,jt,loop)
      node(1,jt,loop2) = num - 2
      do 50 j = 1,num - 2
         node(j+1,jt,loop2) = node(j+2,jt,loop)
50    continue
c store line 4
      node(2,jn,loop2) = kt(1)
      node(3,jn,loop2) = kt(2)
c store line 5 & 6
      call gtside(jm,jo,left,nrit)
      node(2,left,loop2) = kt(3)
      node(3,left,loop2) = kt(5)
      node(2,nrit,loop2) = kt(4)
      node(3,nrit,loop2) = kt(8)
c store line 7 & 8
      call gtside(jt,jn,left,nrit)
      node(2,left,loop2) = node(3,jt,loop)
      node(3,left,loop2) = kt(1)
      node(2,nrit,loop2) = node(num,jt,loop)
      node(3,nrit,loop2) = kt(2)
c store line 9 & 10
      call gtside(jm,jt,left,nrit)
      node(2,left,loop2) = kt(3)
      node(3,left,loop2) = node(3,jt,loop)
      node(2,nrit,loop2) = kt(4)
      node(3,nrit,loop2) = node(num,jt,loop)
c store line 11 & 12
      call gtside(jo,jn,left,nrit)
      node(2,left,loop2) = kt(5)
      node(3,left,loop2) = kt(1)
      node(2,nrit,loop2) = kt(8)
      node(3,nrit,loop2) = kt(2)
c
c store the loop 1, the right side
c
c store line 1
      node(1,jm,loop) = 2
      node(2,jm,loop) = kt(4)
c store line 2
      node(1,jo,loop) = 2
      node(2,jo,loop) = kt(8)
c store line 3
      node(1,jt,loop) = 2
      node(2,jt,loop) = node(num,jt,loop)
      node(3,jt,loop) = node(num+1,jt,loop)
c store line 4
      node(1,jn,loop) = 2
      node(2,jn,loop) = kt(2)
c store line 5 ,skip 6
      call gtside(jm,jo,left,nrit)
```

```
        node(1,left,loop) = 2
        node(2,left,loop) = kt(4)
        node(3,left,loop) = kt(6)
c store line 7,skip 8
        call gtside(jt,jn,left,nrit)
        node(1,left,loop) = 2
        node(2,left,loop) = node(num,jt,loop)
        node(3,left,loop) = kt(2)
c store line 9, skip 10
        call gtside(jm,jt,left,nrit)
        node(1,left,loop) = 2
        node(2,left,loop) = kt(4)
        node(3,left,loop) = node(num,jt,loop)
c store line 11, skip 12
        call gtside(jo,jn,left,nrit)
        node(1,left,loop) = 2
        node(2,left,loop) = kt(6)
        node(3,left,loop) = kt(2)
c
        return
        end

CNAME TRAN14.FOR
C*************************************************************************
        subroutine tran14 (loop,jt,jn,jm,jo,kt)
C*************************************************************************
C
C  PROGRAM: TRAN14.FOR
C
C  PURPOSE:  do the transition by # 14 method
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*************************************************************************
C
C  ALGORITHM EXPLANATION: transition on line 4 8,12
c
c*************************************************************************
c
c  called:  gtside -- given the two opposite li es on a surface then
c                     get the other two lines.
c
c*************************************************************************
c
c  called by: tran1.for -- # 1 transition con roler
c
c*************************************************************************
        include 'para.inc'
        dimension coor(4,maxnod),node(maxsid,12,maxloop),kt(6)
        common/auto/coor,node,last,unit

c...store them into 5 new loop
        loop2 = loop + 1
        loop3 = loop + 2
        loop4 = loop + 3
        loop5 = loop + 4
c
c store the loop 5, the left side
c
        do 10 j = 1,12
           node(1,j,loop5) = 2
10      continue
c store line 1
        node(2,jo,loop5) = node(2,jo,loop)
        node(3,jo,loop5) = kt(5)
c store line 2
        node(2,jn,loop5) = node(2,jn,loop)
        node(3,jn,loop5) = kt(1)
c store line 3
        node(2,jm,loop5) = node(2,jm,loop)
        node(3,jm,loop5) = kt(3)
c store line 4
        node(2,jt,loop5) = node(2,jt,loop)
        node(3,jt,loop5) = node(3,jt,loop)
c store line 5 & 6
        call gtside(jo,jn,left,nrit)
        do 11 i = 1,node(1,left,loop)+1
           node(i,left,loop5) = node(i,left,loop)
11      continue
```

```
      node(2,nrit,loop5) = kt(5)
      node(3,nrit,loop5) = kt(1)
c store line 7 & 8
      call gtside(jm,jt,left,nrit)
      do 12 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
12    continue
      node(2,nrit,loop5) = kt(3)
      node(3,nrit,loop5) = node(3,jt,loop)
c store line 9 & 10
      call gtside(jo,jm,left,nrit)
      do 13 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
13    continue
      node(2,nrit,loop5) = kt(5)
      node(3,nrit,loop5) = kt(3)
c store line 11 & 12
      call gtside(jn,jt,left,nrit)
      do 14 i = 1,node(1,left,loop)+1
         node(i,left,loop5) = node(i,left,loop)
14    continue
      node(2,nrit,loop5) = kt(1)
      node(3,nrit,loop5) = node(3,jt,loop)
c
c store the loop 4, the front side
c
      do 20 j = 1,12
         node(1,j,loop4) = 2
20    continue
c store line 1
      do 21 i = 1,node(1,jo,loop)+1
         node(i,jo,loop4) = node(i,jo,loop)
21    continue
c store line 2
      do 22 i = 1,node(1,jn,loop)+1
         node(i,jn,loop4) = node(i,jn,loop)
22    continue
c store line 3
      node(2,jm,loop4) = kt(5)
      node(3,jm,loop4) = kt(8)
c store line 4
      node(2,jt,loop4) = kt(1)
      node(3,jt,loop4) = kt(2)
c store line 5 & 6
      call gtside(jo,jn,left,nrit)
      do 23 i = 1,node(1,left,loop)+1
         node(i,left,loop4) = node(i,left,loop)
23    continue
      do 24 i = 1,node(1,nrit,loop)+1
         node(i,nrit,loop4) = node(i,nrit,loop)
24    continue
c store line 7 & 8
      call gtside(jm,jt,left,nrit)
      node(2,left,loop4) = kt(5)
      node(3,left,loop4) = kt(1)
      node(2,nrit,loop4) = kt(8)
      node(3,nrit,loop4) = kt(2)
c store line 9 & 10
      call gtside(jo,jm,left,nrit)
      node(2,left,loop4) = node(2,left,loop)
      node(3,left,loop4) = kt(5)
      node(2,nrit,loop4) = node(2,nrit,loop)
      node(3,nrit,loop4) = kt(8)
c store line 11 & 12
      call gtside(jn,jt,left,nrit)
      node(2,left,loop4) = node(2,left,loop)
      node(3,left,loop4) = kt(1)
      node(2,nrit,loop4) = node(2,nrit,loop)
      node(3,nrit,loop4) = kt(2)
c
c store the loop 3, the bottom side
c
      do 30 j = 1,12
         node(1,j,loop3) = 2
30    continue
c store line 1
      do 31 i = 1,node(1,jo,loop)+1
         node(i,jo,loop3) = node(i,jo,loop)
31    continue
c store line 2
```

```
      node(2,jn,loop3) = kt(5)
      node(3,jn,loop3) = kt(6)
c store line 3
      do 32 i = 1,node(1,jm,loop)+1
         node(i,jm,loop3) = node(i,jm,loop)
32    continue
c store line 4
      node(2,jt,loop3) = kt(3)
      node(3,jt,loop3) = kt(4)
c store line 5 & 6
      call gtside(jo,jn,left,nrit)
      node(2,left,loop3) = node(2,left,loop)
      node(3,left,loop3) = kt(5)
      node(2,nrit,loop3) = node(2,nrit,loop)
      node(3,nrit,loop3) = kt(6)
c store line 7 & 8
      call gtside(jm,jt,left,nrit)
      node(2,left,loop3) = node(2,left,loop)
      node(3,left,loop3) = kt(3)
      node(2,nrit,loop3) = node(2,nrit,loop)
      node(3,nrit,loop3) = kt(4)
c store line 9 & 10
      call gtside(jo,jm,left,nrit)
      do 33 i = 1,node(1,left,loop)+1
         node(i,left,loop3) = node(i,left,loop)
33    continue
      do 34 i = 1,node(1,nrit,loop)+1
         node(i,nrit,loop3) = node(i,nrit,loop)
34    continue
c store line 11 & 12
      call gtside(jn,jt,left,nrit)
      node(2,left,loop3) = kt(5)
      node(3,left,loop3) = kt(3)
      node(2,nrit,loop3) = kt(6)
      node(3,nrit,loop3) = kt(4)
c
c store the loop 2, the middle top side
c
      do 40 j = 1,12
        node(1,j,loop2) = 2
40    continue
c store line 1
      node(2,jo,loop2) = kt(5)
      node(3,jo,loop2) = kt(6)
c store line 2
      node(2,jn,loop2) = kt(1)
      node(3,jn,loop2) = kt(2)
c store line 3
      node(2,jm,loop2) = kt(3)
      node(3,jm,loop2) = kt(4)
c store line 4
      num = node(1,jt,loop)
      node(1,jt,loop2) = num-2
      do 50 j = 1,num-2
         node(j+1,jt,loop2) = node(j+2,jt,loop)
50    continue
c store line 5 & 6
      call gtside(jo,jn,left,nrit)
      node(2,left,loop2) = kt(5)
      node(3,left,loop2) = kt(1)
      node(2,nrit,loop2) = kt(6)
      node(3,nrit,loop2) = kt(2)
c store line 7 & 8
      call gtside(jm,jt,left,nrit)
      node(2,left,loop2) = kt(3)
      node(3,left,loop2) = node(3,jt,loop)
      node(2,nrit,loop2) = kt(4)
      node(3,nrit,loop2) = node(num,jt,loop)
c store line 9 & 10
      call gtside(jo,jm,left,nrit)
      node(2,left,loop2) = kt(5)
      node(3,left,loop2) = kt(3)
      node(2,nrit,loop2) = kt(6)
      node(3,nrit,loop2) = kt(4)
c store line 11 & 12
      call gtside(jn,jt,left,nrit)
      node(2,left,loop2) = kt(1)
      node(3,left,loop2) = node(3,jt,loop)
      node(2,nrit,loop2) = kt(2)
      node(3,nrit,loop2) = node(num,jt,loop)
```

```
c
c store the loop 1, the right side
c
c store line 1
      node(1,jo,loop) = 2
      node(2,jo,loop) = kt(8)
c store line 2
      node(1,jn,loop) = 2
      node(2,jn,loop) = kt(2)
c store line 3
      node(1,jm,loop) = 2
      node(2,jm,loop) = kt(4)
c store line 4
      node(1,jt,loop) = 2
      node(2,jt,loop) = node(num,jt,loop)
      node(3,jt,loop) = node(num+1,jt,loop)
c store line 5,skip 6
      call gtside(jo,jn,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(8)
      node(3,left,loop) = kt(2)
c store line 7 ,skip 8
      call gtside(jm,jt,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(4)
      node(3,left,loop) = node(num,jt,loop)
c store line 9, skip 10
      call gtside(jo,jm,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(8)
      node(3,left,loop) = kt(4)
c store line 11, skip 12
      call gtside(jn,jt,left,nrit)
      node(1,left,loop) = 2
      node(2,left,loop) = kt(2)
      node(3,left,loop) = node(num,jt,loop)
c
      return
      end

CNAME TRAN2.FOR
c*******************************************************************
      subroutine tran2(loop,iter,ibeg)
C*******************************************************************
C
C  PROGRAM: TRAN2.FOR
C
C  PURPOSE: DO TRANSITION BY #2 METHOD
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*******************************************************************
C
C  ALGORITHM EXPLANATION: do the transition on two opposit line of
c       a surface in a volume
c
c*******************************************************************
c
c  called:
c       cktran  -- check the nodes on the line to see if the node have
c                  been used or not
c       tran21  -- do the transition on line 1-3,1-2,...etc.
c       tran22  -- do the transition on line 2-4,3-4,...etc.
c       push    -- store the data into new loop
c
c*******************************************************************
c
c  called by: transi.for
c
c*******************************************************************
      include 'para.inc'
      dimension coor(4,maxnod),node(maxsid,12,maxloop),isur(4,12)
      dimension kt(4)
      common/auto/coor,node,last,unit
      data isur/9,7,11,5,  9,3,10,1,  5,2, 6,1,  5,11,7,9,
     $           1,6, 2,5,  1,10,3,9, 10,8,12,6, 11,4,12,2,
     $           7,4, 8,3,  6,12,8,10, 3,8, 4,7,  2,12,4,11/
c
      iend = ibeg+8
```

```
      num1 = node(1,isur(1,ibeg),loop)
      num2 = node(1,isur(3,ibeg),loop)
      if(num1.ge.4 .and. num2.eq.2) then
         j1 = 1
         j2 = 3
      else if(num2.ge.4 .and. num1.eq.2) then
         j1 = 3
         j2 = 1
      end if
      kt(1) = last+1
      do 20 i = 1,4
         coor(i,kt(1)) =((coor(i,node(2,isur(j2,ibeg),loop)) +
     $                   coor(i,node(3,isur(j2,ibeg),loop)))/2.0 +
     $                   coor(i,node(3,isur(j1,ibeg),loop)))/2.0
20    continue
      call cktran(coor(1,kt(1)),kt(1))
      kt(2) = last+1
      numj1 = node(1,isur(j1,ibeg),loop)
      do 30 i = 1,4
         coor(i,kt(2)) =((coor(i,node(2,isur(j2,ibeg),loop)) +
     $                   coor(i,node(3,isur(j2,ibeg),loop)))/2.0 +
     $                   coor(i,node(numj1,isur(j1,ibeg),loop)))/2.0
30    continue
      call cktran(coor(1,kt(2)),kt(2))
      kt(3) = last+1
      do 40 i = 1,4
         coor(i,kt(3)) =((coor(i,node(2,isur(j2,iend),loop)) +
     $                   coor(i,node(3,isur(j2,iend),loop)))/2.0 +
     $                   coor(i,node(3,isur(j1,iend),loop)))/2.0
40    continue
      call cktran(coor(1,kt(3)),kt(3))
      kt(4) = last+1
      numj2 = node(1,isur(j1,iend),loop)
      do 50 i = 1,4
         coor(i,kt(4)) =((coor(i,node(2,isur j2,iend),loop)) +
     $                   coor(i,node(3,isur(j2,i nd),loop)))/2.0 +
     $                   coor(i,node(numj2,isur(. 1,iend),loop)))/2.0
50    continue
      call cktran(coor(1,kt(4)),kt(4))
      if(num1.ge.4 .and. num2.eq.2) then
         call tran21(loop,ibeg,iend,kt)
      else if(num2.ge.4 .and. num1.eq.2) then
         call tran22(loop,ibeg,iend,kt)
      end if
      call push(loop,iter)
      do 60 i = 1,3
         loop = loop+1
         call push(loop,iter)
60    continue
c
      return
      end

CNAME TRAN21.FOR
c-----------------------------------------------------------------------
      subroutine tran21(loop,ibeg,iend,kt)
C-----------------------------------------------------------------------
C
C  PROGRAM: TRAN21.FOR
C
C  PURPOSE: do the bottom transition for # 2
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C-----------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION:
c
c-----------------------------------------------------------------------
c
c  called:  gtside -- given the two opposite lines on a surface then
c                     get the other two lines.
c
c-----------------------------------------------------------------------
c
c  called by: tran2.for
c
c-----------------------------------------------------------------------
      include 'para.inc'
```

```
      dimension coor(4,maxnod),node(maxsid,12,maxloop),isur(4,12)
      dimension kt(4)
      logical oline
      common/auto/coor,node,last,unit
      common/oldln/nnum(20,maxsid*6),lold(maxsid*6),nall
      data isur/9,7,11,5,  9,3,10,1,  5,2, 6,1,  5,11,7,9,
     $           1,6, 2,5,  1,10,3,9, 10,8,12,6, 11,4,12,2,
     $           7,4, 8,3,  6,12,8,10, 3,8, 4,7,  2,12,4,11/
c
c...store them into 4 new loop
c
      loop4 = loop + 1
      loop3 = loop + 2
      loop2 = loop + 3
      call gtside(isur(1,ibeg),isur(1,iend),n9,n10)
      call gtside(isur(3,ibeg),isur(3,iend),n11,n12)
c
c for transition on top line 2,4
c
c store the loop 4, at the top
      nn = 0
      nm = -1
      do 10 i = ibeg,iend,(iend-ibeg)
        do 20 j = 1,4
          node(1,isur(j,i),loop4) = 2
20      continue
c store line 2,4,1,3
        do 30 j = 1,2
          node(j+1,isur(3,i),loop4) = node(j+1,isur(3,i),loop)
          nn = nn + 1
          node(j+1,isur(1,i),loop4) = kt(nn)
30      continue
c store line 5 & 7
          nm = nm + 2
          node(2,isur(4,i),loop4) = kt(nm)
          node(3,isur(4,i),loop4) = node(2,isur(3,i),loop)
c store line 6 & 8
          node(2,isur(2,i),loop4) = kt(1-nm)
          nc = node(1,isur(2,i),loop)+1
          node(3,isur(2,i),loop4) = node(nc,isur(2,i),loop)
10    continue
c
c store line 11,12
c
      do 40 i = n11,n12,(n12-n11)
        do 50 j = 1,node(1,i,loop)+1
          node(j,i,loop4) = node(j,i,loop)
50      continue
40    continue
c
c store line 9,10
c
      nm = 0
      line1 = isur(1,ibeg)
      line2 = isur(1,iend)
      left = n11
      do 60 j = n9,n10,(n10-n9)
        k = last
        nm = nm+1
        oline = .false.
        call fnewnd(line1,line2,nm,nm,left,loop4,nnod,oline,nold)
        if(oline)then
          nnod = lold(nold)
          node(1,j,loop4) = nnod
          do 70 i=1,nnod
            node(i+1,j,loop4) = nnum(i,nold)
70        continue
          last = k
        else
          if(j.eq.n9 .and. nnod.ge.node(1,n11,loop))then
             nnod = node(1,n11,loop)
          else if(j.eq.n10 .and. nnod.ge.node(1,n12,loop))then
             nnod = node(1,n12,loop)
          end if
          node(1,j,loop4) = nnod
          node(2,j,loop4) = node(nm+1,line1,loop4)
          do 80 i = 2,nnod-1
            node(i+1,j,loop4) = k+i-1
80        continue
          node(nnod+1,j,loop4) = node(nm+1,line2,loop4)
```

```
            last = k + nnod - 2
          end if
          left = n12
60      continue
c
c store the loop 2, at the left side
c
        do 110 i = ibeg,iend,(iend-ibeg)
c store line 1,3
          node(1,isur(1,i),loop2) = 2
          node(2,isur(1,i),loop2) = node(2,isur(1,i),loop)
          node(3,isur(1,i),loop2) = node(3,isur(1,i),loop)
c store line 2,4
          node(1,isur(3,i),loop2) = 2
          node(2,isur(3,i),loop2) = node(3,isur(4,i),loop4)
          node(3,isur(3,i),loop2) = node(2,isur(4,i),loop4)
c store line 5,7
          node(1,isur(4,i),loop2) = node(1,isur(4,i),loop)
          do 100 j = 1,node(1,isur(4,i),loop)
            node(j+1,isur(4,i),loop2) = node(j+1,isur(4,i),loop)
100       continue
c store line 6 & 8
          node(1,isur(2,i),loop2) = 2
          node(2,isur(2,i),loop2) = node(3,isur(1,i),loop)
          node(3,isur(2,i),loop2) = node(2,isur(1,i),loop4)
110     continue
c
c store line 9 & 11
c
        do 120 i = n9,n11,(n11-n9)
          do 130 j = 1,node(1,i,loop)+1
            node(j,i,loop2) = node(j,i,loop)
130       continue
120     continue
c
c store line 12
c
        do 131 i = 1,node(1,n9,loop4)+1
          node(i,n12,loop2) = node(i,n9,loop4)
131     continue
c
c store line 10
c
        k = last
        line1 = isur(1,ibeg)
        line2 = isur(1,iend)
        oline = .false.
        call fnewnd(line1,line2,2,2,left,loop,nnod,oline,nold)
c
c store split line into right line of loop two
c
        if(oline)then
          nnod = lold(nold)
          node(1,n10,loop2) = nnod
          do 132 i=1,nnod
            node(i+1,n10,loop2) = nnum(i,nold)
132       continue
          last = k
c
c if the old line was found then delete that line in nnum to save space
c
            do 133 i = nold,nall-1
              lold(i) = lold(i+1)
              do 134 jj = 1,lold(i)
                nnum(jj,i) = nnum(jj,i+1)
134           continue
133         continue
            nall = nall - 1
        else
          if(nnod.ge.node(1,n9,loop)) nnod = node(1,n9,loop)
          node(1,n10,loop2) = nnod
          node(2,n10,loop2) = node(3,line1,loop)
          do 135 i = 2,nnod-1
             node(i+1,n10,loop2) = k+i-1
135       continue
          node(nnod+1,n10,loop2) = node(3,line2,loop)
          last = k + nnod - 2
c
c store this new line data for old line reference check
c
```

```
              nall = nall + 1
              do 136 i = 1, nnod
                nnum(i,nall) = node(i+1,n10,loop2)
136           continue
              lold(nall) = nnod
          end if
c
c store the loop 3, the bottom side
c
          do 170 i = ibeg,iend,(iend-ibeg)
c store line 2,4
            do 140 j = 1,3
              node(j,isur(3,i),loop3) = node(j,isur(1,i),loop4)
140         continue
c store line 1,3
            nct = node(1,isur(1,i),loop)-2
            node(1,isur(1,i),loop3) = nct
            do 150 j = 1,nct
              node(j+1,isur(1,i),loop3) = node(j+2,isur(1,i),loop)
150         continue
c store line 5,7
            node(1,isur(4,i),loop3) = 2
            node(2,isur(4,i),loop3) = node(3,isur(1,i),loop)
            node(3,isur(4,i),loop3) = node(2,isur(1,i),loop4)
c store line 6,8
            node(1,isur(2,i),loop3) = 2
            node(2,isur(2,i),loop3) = node(nct+2,isur(1,i),loop)
            node(3,isur(2,i),loop3) = node(3,isur(1,i),loop4)
170       continue
c
c store line 9,11,12
c
            do 180 j = 1,node(1,n10,loop2)+1
              node(j,n9,loop3) = node(j,n10,loop2)
180         continue
            do 190 j = 1,node(1,n9,loop4)+1
              node(j,n11,loop3) = node(j,n9,loop4)
190         continue
          do 191 i = 1,node(1,n10,loop4)+1
            node(i,n12,loop3) = node(i,n10,loop4)
191       continue
c
c store line 10
c
          k = last
          num1 = node(1,line1,loop)-1
          num2 = node(1,line2,loop)-1
          oline = .false.
          left = n10
          call fnewnd(line1,line2,num1,num2,left,loop,nnod,oline,nold)
          k = last
          if(oline)then
            nnod = lold(nold)
            node(1,n10,loop3) = nnod
            do 192 i=1,nnod
              node(i+1,n10,loop3) = nnum(i,nold)
192         continue
            last = k
c
c if the old line was found then delete that line to save space
c
              do 195 i = nold,nall-1
                lold(i) = lold(i+1)
                do 196 jj = 1,lold(i)
                  nnum(jj,i) = nnum(jj,i+1)
196             continue
195           continue
              nall = nall - 1
          else
            if(nnod.ge.node(1,n10,loop)) nnod = node(1,n10,loop)
            node(1,n10,loop3) = nnod
            node(2,n10,loop3) = node(num1+1,line1,loop)
            do 193 i = 2,nnod-1
               node(i+1,n10,loop3) = k+i-1
193         continue
            node(nnod+1,n10,loop3) = node(num2+1,line2,loop)
            last = k + nnod - 2
c
c store this new line data for old line reference check
c
```

```
            nall = nall + 1
            do 194 i = 1, nnod
              nnum(i,nall) = node(i+1,n10,loop3)
194         continue
            lold(nall) = nnod
          end if
c
c store the loop 1, the right side
c
          do 230 i = ibeg,iend,(iend-ibeg)
c store line 2,4
            do 200 j = 1,3
              node(j,isur(3,i),loop) = node(j,isur(2,i),loop4)
200         continue
c store line 1,3
            node(1,isur(1,i),loop) = 2
            node(2,isur(1,i),loop) = node(2+nct,isur(1,i),loop)
            node(3,isur(1,i),loop) = node(3+nct,isur(1,i),loop)
c store line 5,7
            do 220 j = 1,3
              node(j,isur(4,i),loop) = node(j,isur(2,i),loop3)
220         continue
c
c skips line 6,8
c
230       continue
c
c store line 9,11
c
          do 240 i = 1,node(1,n10,loop3)+1
            node(i,n9,loop) = node(i,n10,loop3)
240       continue
          do 241 i = 1,node(1,n12,loop3)+1
            node(i,n11,loop) = node(i,n12,loop3)
241       continue
c
c skips line n10,n12
c
          return
          end

CNAME TRAN22.FOR
c*****************************************************************
          subroutine tran22(loop,ibeg,iend,kt)
C*****************************************************************
C
C  PROGRAM: TRAN22.FOR
C
C  PURPOSE: do the top transition for # 2
C
C  WRITTEN BY:  SSUTA HSU        CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                  CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: do transition for line 2,4, or 6,8, or 10,12
c
c*****************************************************************
c
c  called:  gtside -- given the two opposite lines on a surface then
c                     get the other two lines.
c
c*****************************************************************
c
c  called by: tran2.for
c
c*****************************************************************
          include 'para.inc'
          dimension coor(4,maxnod),node(maxsid,12,maxloop),isur(4,12)
          dimension kt(4)
          logical oline
          common/auto/coor,node,last,unit
          common/oldln/nnum(20,maxsid-6),lold(maxsid-6),nall
          data isur/9,7,11,5,  9,3,10,1,  5,2, 6,1,  5,11,7,9,
     $              1,6, 2,5,  1,10,3,9, 10,8,12,6, 11,4,12,2,
     $              7,4, 8,3,  6,12,8,10, 3,8, 4,7,  2,12,4,11/
c
c...store them into 4 new loop
c
          loop3 = loop + 1
```

```
        loop4 = loop + 2
        loop2 = loop + 3
        call gtside(isur(1,ibeg),isur(1,iend),n9,n10)
        call gtside(isur(3,ibeg),isur(3,iend),n11,n12)
c
c store the loop 4, at the bottom side
c
        nn = 0
        nm = -1
        do 10 i = ibeg,iend,(iend-ibeg)
          do 20 j = 1,4
            node(1,isur(j,i),loop4) = 2
20        continue
c store line 1 & 3
          do 30 j = 1,2
            node(j+1,isur(1,i),loop4) = node(j+1,isur(1,i),loop)
c store line 2 & 4
            nn = nn + 1
            node(j+1,isur(3,i),loop4) = kt(nn)
30        continue
c store line 5 & 7
          nm = nm + 2
          node(2,isur(4,i),loop4) = node(2,isur(1,i),loop)
          node(3,isur(4,i),loop4) = kt(nm)
c store line 6 & 8
          node(2,isur(2,i),loop4) = node(2,isur(2,i),loop)
          node(3,isur(2,i),loop4) = kt(1+nm)
10      continue
c store line 9 & 10
c
c find the lines on both sides
c
        do 40 i = n9,n10,(n10-n9)
          nod = node(1,i,loop)
          do 50 j = 1,nod+1
            node(j,i,loop4) = node(j,i,loop)
50        continue
40      continue
c
c store line 11,12
c
        nm = 0
        line1 = isur(3,ibeg)
        line2 = isur(3,iend)
        left = n9
        do 60 j = n11,n12,(n12-n11)
          k = last
          nm = nm+1
          oline = .false.
          call fnewnd(line1,line2,nm,nm,left,loop4,nnod,oline,nold)
          if(oline)then
            nnod = lold(nold)
            node(1,j,loop4) = nnod
            do 70 i=1,nnod
              node(i+1,j,loop4) = nnum(i,nold)
70          continue
            last = k
          else
            if(j.eq.n11 .and. nnod.ge.node(1,n9,loop))then
              nnod = node(1,n9,loop)
            else if(j.eq.n12 .and. nnod.ge.node(1,n10,loop))then
              nnod = node(1,n10,loop)
            end if
            node(1,j,loop4) = nnod
            node(2,j,loop4) = node(nm+1,line1,loop4)
            do 73 i = 2,nnod-1
              node(i+1,j,loop4) = k+i-1
73          continue
            node(nnod+1,j,loop4) = node(nm+1,line2,loop4)
            last = k + nnod - 2
          end if
          left = n10
60      continue
c
c store the loop 2, at the left side
c
        do 110 i = ibeg,iend,(iend-ibeg)
c store line 1,3
          do 80 j = 1,3
            node(j,isur(1,i),loop2) = node(j,isur(4,i),loop4)
```

```
80          continue
c store line 2 & 4
            node(1,isur(3,i),loop2) = 2
            do 90 j = 1,2
              node(j+1,isur(3,i),loop2) = node(j+1,isur(3,i),loop)
90          continue
c store line 5 & 7
            node(1,isur(4,i),loop2) = node(1,isur(4,i),loop)
            do 100 j = 1,node(1,isur(4,i),loop)
              node(j+1,isur(4,i),loop2) = node(j+1,isur(4,i),loop)
100         continue
c store line 6 & 8
            node(1,isur(2,i),loop2) = 2
            node(2,isur(2,i),loop2) = node(2,isur(3,i),loop4)
            node(3,isur(2,i),loop2) = node(3,isur(3,i),loop)
110       continue
c store line 9 & 11
c
          do 120 i = n9,n11,(n11-n9)
            nod = node(1,i,loop)
            do 130 j = 1,nod+1
              node(j,i,loop2) = node(j,i,loop)
130         continue
120       continue
c
c store line 10
c
          do 140 i = 1,node(1,n11,loop4)+1
            node(i,n10,loop2) = node(i,n11,loop4)
140       continue
c
c store line 12
c
          k = last
          line1 = isur(3,ibeg)
          line2 = isur(3,iend)
          left = n11
          oline = .false.
          call fnewnd(line1,line2,2,2,left,loop,nnod,oline,nold)
          if(oline)then
            nnod = lold(nold)
            node(1,n12,loop2) = nnod
            do 150 i=1,nnod
              node(i+1,n12,loop2) = nnum(i,nold)
150         continue
            last = k
c
c if the old line was found then delete that line to save space
c
              do 151 i = nold,nall-1
                lold(i) = lold(i+1)
                do 152 jj = 1,lold(i)
                  nnum(jj,i) = nnum(jj,i+1)
152             continue
151           continue
              nall = nall - 1
          else
            if(nnod.ge.node(1,n11,loop)) nnod = node(1,n11,loop)
            node(1,n12,loop2) = nnod
            node(2,n12,loop2) = node(3,line1,loop)
            do 160 i = 2,nnod-1
               node(i+1,n12,loop2) = k+i-1
160         continue
            node(nnod+1,n12,loop2) = node(3,line2,loop)
            last = k + nnod - 2
c
c store this new line data for old line reference check
c
              nall = nall + 1
              do 161 i = 1, nnod
                nnum(i,nall) = node(i+1,n12,loop2)
161           continue
              lold(nall) = nnod
          end if
c
c store the loop 3, the top side
c
          do 170 i = ibeg,iend,(iend-ibeg)
c store line 1,3
              node(1,isur(1,i),loop3) = 2
```

```
            node(2,isur(1,i),loop3) = node(2,isur(3,i),loop4)
            node(3,isur(1,i),loop3) = node(3,isur(3,i),loop4)
c store line 2,4
            nct = node(1,isur(3,i),loop)-2
            node(1,isur(3,i),loop3) = nct
            do 180 j = 1,nct
              node(j+1,isur(3,i),loop3) = node(j+2,isur(3,i),loop)
180         continue
c store line 5,7
            node(1,isur(4,i),loop3) = 2
            node(2,isur(4,i),loop3) = node(2,isur(3,i),loop4)
            node(3,isur(4,i),loop3) = node(3,isur(3,i),loop)
c store line 6,8
            node(1,isur(2,i),loop3) = 2
            node(2,isur(2,i),loop3) = node(3,isur(3,i),loop4)
            node(3,isur(2,i),loop3) = node(nct+2,isur(3,i),loop)
170       continue
c
c store line 9,10,11
c
            do 190 j = 1,node(1,n11,loop4)+1
              node(j,n9,loop3) = node(j,n11,loop4)
190         continue
            do 191 j = 1,node(1,n12,loop4)+1
              node(j,n10,loop3) = node(j,n12,loop4)
191         continue
          do 192 i = 1,node(1,n12,loop2)+1
            node(i,n11,loop3) = node(i,n12,loop2)
192       continue
c
c store line 12
c
          k = last
          num1 = node(1,line1,loop)-1
          num2 = node(1,line2,loop)-1
          left = n12
          oline = .false.
          call fnewnd(line1,line2,num1,num2,left loop,nnod,oline,nold)
          k = last
          if(oline)then
            nnod = lold(nold)
            node(1,n12,loop3) = nnod
            do 194 i=1,nnod
              node(i+1,n12,loop3) = nnum(i,nold)
194         continue
            last = k
c
c if the old line was found then delete that line to save space
c
              do 195 i = nold,nall-1
                lold(i) = lold(i+1)
                do 198 jj = 1,lold(i)
                  nnum(jj,i) = nnum(jj,i+1)
198             continue
195           continue
              nall = nall - 1
          else
            if(nnod.ge.node(1,n12,loop)) nnod = node(1,n12,loop)
            node(1,n12,loop3) = nnod
            node(2,n12,loop3) = node(num1+1,line1,loop)
            do 197 i = 2,nnod-1
               node(i+1,n12,loop3) = k+i-1
197         continue
            node(nnod+1,n12,loop3) = node(num2+1,line2,loop)
            last = k + nnod - 2
c
c store this new line data for old line reference check
c
              nall = nall + 1
              do 198 i = 1, nnod
                nnum(i,nall) = node(i+1,n12,loop3)
198           continue
              lold(nall) = nnod
          end if
c
c store the loop 1, the right side
c
          do 230 i = ibeg,iend,(iend-ibeg)
c store line 1,3
            node(1,isur(1,i),loop) = 2
```

```
            node(2,isur(1,i),loop) = node(3,isur(2,i),loop4)
            node(3,isur(1,i),loop) = node(2,isur(2,i),loop4)
c store line 2,4
            node(1,isur(3,i),loop) = 2
            node(2,isur(3,i),loop) = node(nct+2,isur(3,i),loop)
            node(3,isur(3,i),loop) = node(nct+3,isur(3,i),loop)
c store line 5,7
            do 220 j = 1,3
              node(j,isur(4,i),loop) = node(j,isur(2,i),loop3)
220         continue
c
c skips line 6,8
c
230       continue
c
c store line 9,11
c
          do 240 i = 1,node(1,n10,loop3)+1
            node(i,n9,loop) = node(i,n10,loop3)
240       continue
          do 241 i = 1,node(1,n12,loop3)+1
            node(i,n11,loop) = node(i,n12,loop3)
241       continue
c
c skips line 10,12
c
          return
          end

CNAME TRAN5.FOR
c*****************************************************************
        subroutine tran5(loop,iter)
C*****************************************************************
C
C  PROGRAM: TRAN5.FOR
C
C  PURPOSE: DO TRANSITION BY #5 METHOD
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: If model was too complex to subdivided by
c  either tran1 or tran2 method, then add a cube inside the old solid
c
c*****************************************************************
c
c  called:
c       tran51  -- add a new hexahedron in the center of old hexahedeon
c       push    -- store the data into new loop
c
c*****************************************************************
c
c  called by:  transi.for -- transition flag checker
c
c*****************************************************************
        include 'para.inc'
        dimension coor(4,maxnod),node(maxsid,12,maxloop),cent(4)
        common/auto/coor,node,last,unit
c
        k = last
c
c find the 8 point of center cube
c
        num = node(1,4,loop)
        do 10 i = 1,4
           diff = coor(i,node(num+1,4,loop)) - coor(i,node(2,1,loop))
           coor(i,k+1) = coor(i,node(2,1,loop)) + diff/3.0
           coor(i,k+8) = coor(i,node(2,1,loop)) + diff*2.0/3.0
10      continue
        num = node(1,1,loop)
        do 20 i = 1,4
           diff = coor(i,node(2,4,loop)) - coor(i,node(num+1,1,loop))
           coor(i,k+2) = coor(i,node(num+1,1,loop)) + diff/3.0
           coor(i,k+7) = coor(i,node(num+1,1,loop)) + diff*2.0/3.0
20      continue
        num = node(1,3,loop)
        do 30 i = 1,4
           diff = coor(i,node(num+1,3,loop)) - coor(i,node(2,2,loop))
```

```
            coor(i,k+3) = coor(i,node(2,2,loop)) + diff/3.0
            coor(i,k+6) = coor(i,node(2,2,loop)) + diff*2.0/3.0
30      continue
        num = node(1,2,loop)
        do 40 i = 1,4
            diff = coor(i,node(2,3,loop)) - coor(i,node(num+1,2,loop))
            coor(i,k+4) = coor(i,node(num+1,2,loop)) + diff/3.0
            coor(i,k+5) = coor(i,node(num+1,2,loop)) + diff*2.0/3.0
40      continue
        call tran51(loop)
        last = k+8
        call push(loop,iter)
        do 50 i = 1,6
            loop = loop+1
            call push(loop,iter)
50      continue
c
        return
        end

CNAME TRAN51.FOR
C*****************************************************************************
        subroutine tran51(loop)
C*****************************************************************************
C
C  PROGRAM: TRAN51.FOR
C
C  PURPOSE: DO TRANSITION BY #51 METHOD
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION: add a cube to the center of the complex
c       transition solid
c
c*****************************************************************************
c
c  called by: tran5.for -- special number of no es controler
c
c*****************************************************************************
        include 'para.inc'
        dimension coor(4,maxnod),node(maxsid,12,maxloop),isur(4,6)
        common/auto/coor,node,last,unit
        data isur/1,6,2,5, 1,10,3,9, 10,8,12,6, 2,12,4,11,
     $             5,11,7,9, 3,8,4,7/
c...store them into 5 new loop
        k = last
        loop2 = loop + 1
        loop3 = loop + 2
        loop4 = loop + 3
        loop5 = loop + 4
        loop6 = loop + 5
        loop7 = loop + 6
c initialize the node number
        do 20 j = 1,6
          do 10 i = 1,12
           node(1,i,loop+j) = 2
10        continue
20      continue
c
c store one surface for each loop first
c
        do 50 m = 1,6
          do 40 j = 1,4
            do 30 i = 1,node(1,isur(j,m),loop)+1
              node(i,isur(j,m),loop+m) = node(i,isur(j,m),loop)
30          continue
40        continue
50      continue
c store the loop 2, the front side
c store line 3,4,7,8
        inc = -1
        do 60 i = 3,4
          inc = inc + 2
          node(2,i,loop2) = k+inc
          node(3,i,loop2) = k+inc+1
60      continue
        inc = 0
```

```
      do 70 i = 7,8
        inc = inc + 1
        node(2,i,loop2) = k+inc
        node(3,i,loop2) = k+inc+2
70    continue
c store line 9,10,11,12
      inc = 0
      do 80 i = 9,12
        inc = inc + 1
        node(2,i,loop2) = node(2,i,loop)
        node(3,i,loop2) = k+inc
80    continue
c store the loop 3, the bottom side
c store line 2,4,11,12
      inc = -3
      do 90 i = 2,4,2
        inc = inc + 4
        node(2,i,loop3) = k+inc
        node(3,i,loop3) = k+inc+1
90    continue
      inc = 0
      do 100 i = 11,12
        inc = inc + 1
        node(2,i,loop3) = k+inc
        node(3,i,loop3) = k+inc+4
100    continue
c store line 5,6,7,8
      inc = 0
      do 110 i = 5,6
        inc = inc + 1
        node(2,i,loop3) = node(2,i,loop)
        node(3,i,loop3) = k+inc
110    continue
      inc = 4
      do 120 i = 7,8
        inc = inc + 1
        node(2,i,loop3) = node(2,i,loop)
        node(3,i,loop3) = k+inc
120    continue
c
c store the loop 4, the right side
c store line 9,11,5,7
      inc = 0
      do 130 i = 9,11,2
        inc = inc + 2
        node(2,i,loop4) = k+inc
        node(3,i,loop4) = k+inc+4
130   continue
      inc = -2
      do 140 i = 5,7,2
        inc = inc + 4
        node(2,i,loop4) = k+inc
        node(3,i,loop4) = k+inc+2
140   continue
c store line 1,2,3,4
      inc = 0
      do 150 i = 1,4
        num = node(1,i,loop)
        inc = inc + 2
        node(2,i,loop4) = k+inc
        node(3,i,loop4) = node(num+1,i,loop)
150   continue
c
c store the loop 5, the top side
c store line 1,3,9,10
      inc = -1
      do 160 i = 1,3,2
        inc = inc + 4
        node(2,i,loop5) = k+inc
        node(3,i,loop5) = k+inc+1
160    continue
      inc = 2
      do 170 i = 9,10
        inc = inc + 1
        node(2,i,loop5) = k+inc
        node(3,i,loop5) = k+inc+4
170    continue
c store line 5,6,7,8
      inc = 2
      do 180 i = 5,6
```

```
          num = node(1,i,loop)
          inc = inc + 1
          node(2,i,loop5) = k+inc
          node(3,i,loop5) = node(num+1,i,loop)
180      continue
        inc = 6
        do 190 i = 7,8
          num = node(1,i,loop)
          inc = inc + 1
          node(2,i,loop5) = k+inc
          node(3,i,loop5) = node(num+1,i,loop)
190      continue
c
c store the loop 6, the left side
c store line 10,12,6,8
        inc = -1
        do 200 i = 10,12,2
          inc = inc + 2
          node(2,i,loop6) = k+inc
          node(3,i,loop6) = k+inc+4
200      continue
        inc = -3
        do 210 i = 6,8,2
          inc = inc + 4
          node(2,i,loop6) = k+inc
          node(3,i,loop6) = k+inc+2
210      continue
c store line 1,2,3,4
        inc = -1
        do 220 i = 1,4
          inc = inc + 2
          node(2,i,loop6) = node(2,i,loop)
          node(3,i,loop6) = k+inc
220      continue
c
c store the loop 7, the back side
c store line 1,2,5,6
        inc = 3
        do 240 i = 1,2
          inc = inc + 2
          node(2,i,loop7) = k+inc
          node(3,i,loop7) = k+inc+1
240      continue
        inc = 4
        do 250 i = 5,6
          inc = inc + 1
          node(2,i,loop7) = k+inc
          node(3,i,loop7) = k+inc+2
250     continue
c store line 9,10,11,12
        inc = 4
        do 260 i = 9,12
          num = node(1,i,loop)
          inc = inc + 1
          node(2,i,loop7) = k+inc
          node(3,i,loop7) = node(num+1,i,loop)
260     continue
c
c store loop1, the center cube
c store line 1,2,3,4
        inc = -1
        do 270 i = 1,4
          node(1,i,loop) = 2
          inc = inc + 2
          node(2,i,loop) = k+inc
          node(3,i,loop) = k+inc+1
270     continue
c store line 5,6,7,8
        inc = 0
        do 280 i = 5,6
          node(1,i,loop) = 2
          inc = inc + 1
          node(2,i,loop) = k+inc
          node(3,i,loop) = k+inc+2
280     continue
        inc = 4
        do 290 i = 7,8
          node(1,i,loop) = 2
          inc = inc + 1
          node(2,i,loop) = k+inc
```

```
          node(3,i,loop) = k+inc+2
290     continue
c store line 9,10,11,12
        inc = 0
        do 300 i = 9,12
          node(1,i,loop) = 2
          inc = inc + 1
          node(2,i,loop) = k+inc
          node(3,i,loop) = k+inc+4
300     continue
c
        return
        end

CNAME TRANSI.FOR
c*******************************************************************************
        subroutine transi(loop,iter,iflagx,iflagy,iflagz)
C*******************************************************************************
C
C  PROGRAM: TRANSI.FOR
C
C  PURPOSE: CHOOSE THE PROPER TRANSITION
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*******************************************************************************
C
C  ALGORITHM EXPLANATION: check the transition flag and assign the proper
C       transition
c
c*******************************************************************************
c
c  called:
c       tran1.for -- transition method 1
c       tran2.for -- transition method 2
c       tran5.for -- transition method 5
c
c*******************************************************************************
c
c  called by:
c       vmesh.for -- volume meshing controler
c
c*******************************************************************************
c
c do #5 transition --- add cube in the center
c
        if(iflagx.eq.5 .or. iflagy.eq.5 .or. iflagz.eq.5)then
           call tran5(loop,iter)
c
c do z-direction transition
c
        else if(iflagz.eq.21 .and. iflagx.ne.21)then
           call tran2(loop,iter,2)
        else if(iflagz.eq.22 .and. iflagy.ne.21)then
           call tran2(loop,iter,1)
        else if(iflagz.eq.11 .and. iflagx.ne.21 .and. iflagy.ne.21)then
           call tran1(loop,iter,9,iflagz)
        else if(iflagz.eq.12 .and. iflagx.ne.21 .and. iflagy.ne.21)then
           call tran1(loop,iter,10,iflagz)
        else if(iflagz.eq.13 .and. iflagx.ne.21 .and. iflagy.ne.21)then
           call tran1(loop,iter,11,iflagz)
        else if(iflagz.eq.14 .and. iflagx.ne.21 .and. iflagy.ne.21)then
           call tran1(loop,iter,12,iflagz)
c
c do y-direction transition
c
        else if(iflagy.eq.21 .and. iflagx.ne.22)then
           call tran2(loop,iter,3)
        else if(iflagy.eq.22 .and. iflagz.ne.21)then
           call tran2(loop,iter,4)
        else if(iflagy.eq.11 .and. iflagx.ne.22 .and. iflagz.ne.21)then
           call tran1(loop,iter,5,iflagy)
        else if(iflagy.eq.12 .and. iflagx.ne.22 .and. iflagz.ne.21)then
           call tran1(loop,iter,6,iflagy)
        else if(iflagy.eq.13 .and. iflagx.ne.22 .and. iflagz.ne.21)then
           call tran1(loop,iter,7,iflagy)
        else if(iflagy.eq.14 .and. iflagx.ne.22 .and. iflagz.ne.21)then
           call tran1(loop,iter,8,iflagy)
c
```

```
c do x-direction transition
c
        else if(iflagx.eq.21 .and. iflagy.ne.22)then
           call tran2(loop,iter,5)
        else if(iflagx.eq.22 .and. iflagz.ne.22)then
           call tran2(loop,iter,6)
        else if(iflagx.eq.11 .and. iflagy.ne.22 .and. iflagz.ne.22)then
           call tran1(loop,iter,1,iflagx)
        else if(iflagx.eq.12 .and. iflagy.ne.22 .and. iflagz.ne.22)then
           call tran1(loop,iter,2,iflagx)
        else if(iflagx.eq.13 .and. iflagy.ne.22 .and. iflagz.ne.22)then
           call tran1(loop,iter,3,iflagx)
        else if(iflagx.eq.14 .and. iflagy.ne.22 .and. iflagz.ne.22)then
           call tran1(loop,iter,4,iflagx)
        end if
c
        return
        end

CNAME VGTNOD.FOR
c*****************************************************************
        subroutine vgtnod(loop,next,line1,line2,num1,num2)
C*****************************************************************
C
C  PROGRAM: VGTNOD.FOR
C
C  PURPOSE: FIND THE BEST SPLIT NODES BETWEEN TWO LINES
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 8, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: CHECK THROUGH THE OLD LINE TO FIND THE BEST SPLIT
C       NODES, IF NOT FOUND THEN CREATE A BEST LINE
c
c*****************************************************************
c
c  called:
c       vnext1.for -- find the next best node
c       gtside.for -- get other pair of lines in a surface
c       vnewds.for -- get new distance
c
c*****************************************************************
c
c  called by:
c        xmesh.for -- x-direction mesher
c        ymesh.for -- y-direction mesher
c        zmesh.for -- z-direction mesher
c
c*****************************************************************
        include 'para.inc'
        dimension coor(4,maxnod),node(maxsid,12,maxloop),xo(maxsid),
     $            yo(maxsid),zo(maxsid),iorder(2,12)
        common/auto/coor,node,last,unit
        logical found
        data iorder/2,3, 4,1, 1,4, 3,2,   6,7, 8,5, 5,8, 7,6,
     $             10,11, 12,9, 9,12, 11,10/
c
c check for the existing line, if not found then make a new one
c
        line2 = iorder(next,line1)
        call vnext1(loop,line1,line2,num1,num2,found)
        if(found) return
c
c  find the best split line
c
        nnod = node(1,line2,loop)
        nod1 = node(num1+1,line1,loop)
        dmin = 1.e10
        dist = 1.e9
        k = 0
C       k = 1
        do while (dist.lt.dmin .and. k.lt.nnod)
           dmin = dist
           k = k+2
C          k = k+1
           if(k.lt.nnod)then
             nod2 = node(k+1,line2,loop)
             dist = sqrt((coor(1,nod1)-coor(1,nod2))**2+
```

```
     $                    (coor(2,nod1)-coor(2,nod2))**2+
     $                    (coor(3,nod1)-coor(3,nod2))**2)
c
c  get the deformed distance -- better result
c           call vnewds(nod1,nod2,num,xo,yo,zo,dist)
          end if
        end do
        num2 = k-2
C       num2 = k-1
c
        return
        end

CNAME VNEWDS.FOR
c-----------------------------------------------------------------------
        subroutine vnewds(nod1,nod2,num,xo,yo,zo,dist)
C-----------------------------------------------------------------------
C
C  PROGRAM: VNEWDS.FOR
C
C  PURPOSE: find the distance of a deformed lines
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C-----------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION:
c
c-----------------------------------------------------------------------
c
c  called: deform.for -- deformed the nodes
c
c-----------------------------------------------------------------------
c
c  called by:
c       vgtnod.for -- find the best node for split line
c       fnewnd.for -- find the new nodes
c
c-----------------------------------------------------------------------
        include 'para.inc'
        dimension coor(4,maxnod),node(maxsid,12,maxloop),xo(maxsid),
     $            yo(maxsid),zo(maxsid)
        common/auto/coor,node,last,unit
c
c assume the number of nodes require same as the left line to start with
c
        sum=0.0
        do 10 j = 1,num
          xi = coor(1,nod1) + (coor(1,nod2)-coor(1,nod1))*(j-1)/(num-1)
          yi = coor(2,nod1) + (coor(2,nod2)-coor(2,nod1))*(j-1)/(num-1)
          zi = coor(3,nod1) + (coor(3,nod2)-coor(3,nod1))*(j-1)/(num-1)
          call deform(xi,yi,zi,xo(j),yo(j),zo(j))
          if(j.gt.1)then
            sum = sum + sqrt((xo(j)-xo(j-1))**2 + (yo(j)-yo(j-1))**2 +
     $           (zo(j)-zo(j-1))**2)
          end if
10      continue
        dist = sum
c
        return
        end

CNAME VNEWLN.FOR
c-----------------------------------------------------------------------
        subroutine vnewln(line,num,loop)
C-----------------------------------------------------------------------
C
C  PROGRAM: VNEWLN.FOR
C
C  PURPOSE: store the old line into two new loop lines at subdivision node
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C-----------------------------------------------------------------------
C
C  ALGORITHM EXPLANATION:
c
c-----------------------------------------------------------------------
```

```
c
c  called by:
c         xmesh.for -- x-direction mesher
c         ymesh.for -- y-direction mesher
c         zmesh.for -- z-direction mesher
c
c*****************************************************************
         include 'para.inc'
         dimension coord(4,maxnod),node(maxsid,12,maxloop),mtemp(maxsid)
         common/auto/coord,node,last,unit
c
         do 10 i = 1,node(1,line,loop)
            mtemp(i) = node(i+1,line,loop)
10       continue
c
c...store data into new loop one
c
         node(1,line,loop) = node(1,line,loop)-num+1
         do 20 i = 1,node(1,line,loop)
            node(i+1,line,loop) = mtemp(i+num-1)
20       continue
c
c...store data into new loop two
c
         loop2 = loop + 1
         node(1,line,loop2) = num
         do 30 i = 1,num
            node(i+1,line,loop2) = mtemp(i)
30       continue
c
         return
         end

CNAME VNEWND.FOR
c*****************************************************************
         subroutine vnewnd(line1,line2,nnod1,nnod2,loop)
C*****************************************************************
C
C  PROGRAM: VNEWND.FOR
C
C  PURPOSE:  find the new nodes for the split line
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION: CHECK THE EXISTENCE OF THE OLD LINE IF NOT EXIST
C       THEN ADD NEW NODE TO THAT LINE, OTHERWISE COPY THE OLD LINE NODES
c
c*****************************************************************
c
c  called:
c       gtside.for -- get other pair of lines in a surface
c       fnewnd.for -- find the new nodes
c
c*****************************************************************
c
c  called by:
c         xmesh.for -- x-direction mesher
c         ymesh.for -- y-direction mesher
c         zmesh.for -- z-direction mesher
c
c*****************************************************************
         include 'para.inc'
         dimension coord(4,maxnod),node(maxsid,12,maxloop)
         logical oline
         common/auto/coord,node,last,unit
         common/oldln/nnum(20,maxsid*6),lold(maxsid*6),nall
c
         if(line2.lt.line1)then
            mbeg = line2
            next = line1
            num1 = nnod2
            num2 = nnod1
         else
            mbeg = line1
            next = line2
            num1 = nnod1
            num2 = nnod2
```

```
        end if
c
c find the lines on both sides
c
        call gtside(mbeg,next,left,nrit)
c
c  find the require nodes in the new line and adjust the next line num
c  to fit them into even number.
c
        call fnewnd(mbeg,next,num1,num2,left,loop,nnod,oline,nold)
c
c...store old left line into new left line loop two
c
        k=last
        numrit = node(1,nrit,loop)
        numlft = node(1,left,loop)
        loop2 = loop + 1
        do 30 i = 1,numlft+1
           node(i,left,loop2) = node(i,left,loop)
30      continue
c
c store split line into right line of loop two
c
        if(oline)then
          nnod = lold(nold)
          node(1,nrit,loop2) = nnod
          do 31 i=1,nnod
            node(i+1,nrit,loop2) = nnum(i,nold)
31        continue
          last = k
        else
          node(1,nrit,loop2) = nnod
          node(2,nrit,loop2) = node(num1+1,mbeg,loop)
          do 40 i = 2,nnod-1
             node(i+1,nrit,loop2) = k+i-1
40        continue
          node(nnod-1,nrit,loop2) = node(num2+1,next,loop)
          last = k + nnod - 2
        end if
c
c...store split line into left line of loop one
c*  no need to store the old right line(it doesn't change loop number)
c
        do 50 i = 1,nnod+1
           node(i,left,loop) = node(i,nrit,loop2)
50      continue
c
c if the old line was found then delete that line to save space
c
        if(oline)then
           do 80 i = nold,nall-1
             lold(i) = lold(i+1)
             do 90 j = 1,lold(i)
               nnum(j,i) = nnum(j,i+1)
90           continue
80         continue
           nall = nall - 1
c
c if this was a new line then store for old line reference check
c
c       if(line1.ne.1 .and. line1.ne.5 .and. line1.ne.9 .and.
c     $    line2.ne.1 .and. line2.ne.5 .and. line2.ne.9)then
        else
          nall = nall + 1
          do 51 i = 1, nnod
            nnum(i,nall) = node(i+1,left,loop)
51        continue
          lold(nall) = nnod
        end if
c
c...plot the new spliting line **** remove it after done
c
c       x1 = coord(1,node(2,nrit,loop2))
c       y1 = coord(2,node(2,nrit,loop2))
c       z1 = coord(3,node(2,nrit,loop2))
c       do 60 i = 2,nnod
c          x2 = coord(1,node(i+1,nrit,loop2))
c          y2 = coord(2,node(i+1,nrit,loop2))
c          z2 = coord(3,node(i+1,nrit,loop2))
c          call pipe(x1,y1,z1,x2,y2,z2)
```

```
c           call circ(x2,y2,z2,0.5,4)
c           x1 = x2
c           y1 = y2
c           z1 = z2
c60      continue
c
         return
         end

CNAME VNEXT1.FOR
c*****************************************************************************
         subroutine vnext1(loop,line1,line2,num1,num2,found)
C*****************************************************************************
C
C  PROGRAM: VNEXT1.FOR
C
C  PURPOSE: get the previous defined line to be the starting line for next
c        volume subdivision process
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*****************************************************************************
c
c  called : cholin.for -- check all the possible old lines
c
c*****************************************************************************
c
c  called by: vgtnod.for -- find the best node for split line
c
c*****************************************************************************
         include 'para.inc'
         dimension coor(4,maxnod),node(maxsid,1 ,maxloop)
         common/auto/coor,node,last,unit
         common/oldln/nnum(20,maxsid-6),lold(ma  id-6),nall
         logical found
c
         found = .false.
         nod1 = node(num1+1,line1,loop)
         do 30 n = 3,node(1,line2,loop)
             nod2 = node(n,line2,loop)
             j = 1
             do while (j.le.nall .and. .not.found)
                if((nod1.eq.nnum(1,j).and.nod2.eq.nnum(lold(j),j)).or.
     $            (nod2.eq.nnum(1,j).and.nod1.eq.nnum(lold(j),j)))then
                  num2 = n - 1
                  found = .true.
                  return
                end if
                j = j + 1
c check for partial line case
              call cholin(nod1,nod2,found,nold)
                 if(found)then
                    num2 = n - 1
                    return
                 end if
             end do
30       continue
c
c if still not found then recheck all the nodes one by one, however this will
c reset the node1(ie.num1)
c
         do 20 i = 3,node(1,line1,loop)
            nod1 = node(i,line1,loop)
            do 10 n = 3,node(1,line2,loop)
              nod2 = node(n,line2,loop)
              j = 1
              do while (j.le.nall .and. .not.found)
                if((nod1.eq.nnum(1,j).and.nod2.eq.nnum(lold(j),j)).or.
     $            (nod2.eq.nnum(1,j).and.nod1.eq.nnum(lold(j),j)))then
                  num1 = i - 1
                  num2 = n - 1
                  found = .true.
                  return
                end if
                j = j + 1
```

```
            end do
            call cholin(nod1,nod2,found,nold)
               if(found)then
                  num1 = i - 1
                  num2 = n - 1
                  return
               end if
10          continue
20       continue
         return
         end
CNAME VSTORE.FOR
c*****************************************************************************
         subroutine vstore(loop)
C*****************************************************************************
C
C  PROGRAM: VSTORE.FOR
C
C  PURPOSE: STORE THE NODAL NUMBER FOR PLOT
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION: store the node number for draw polygone by
C       counter clockwise order
c
c*****************************************************************************
c
c  called by: vmesh.for -- volume mesh controler
c
c*****************************************************************************
         include 'para.inc'
         dimension x(0:maxdeg,maxid),y(0:maxdeg maxid),
     $             z(0:maxdeg,maxid),ideg(maxid ,isurf(4,6)
         common/data/x,y,z,ideg
         dimension coor(4,maxnod),node(maxsid,1: maxloop)
         common/auto/coor,node,last,unit
         common/fin/ncon(0:10,maxid)
         data isurf/1,6,2,5, 9,3,10,1, 10,8,12,6  2,12,4,11,
     $              5,11,7,9, 7,4,8,3/
         save id
c
         if(id.gt.maxid)then
           write(*,*)'ERROR -- Increase <maxid> to larger than ',id
           return
         end if
         do 10 k = 1,6
            num = -1
            id=id+1
            do 20 l = 1,2
            do 20 i = 1,node(1,isurf(l,k),loop)-1
               num=num+1
               ncon(num,id) = node(i+1,isurf(l,k),loop)
20          continue
            do 30 l = 3,4
            do 30 i = node(1,isurf(l,k),loop),2,-1
               num=num+1
               ncon(num,id) = node(i+1,isurf(l,k),loop)
30          continue
            ideg(id) = num
c
c store the coordinate into common block for drawing
c
            do 40 i = 0,ideg(id)
               x(i,id)=coor(1,ncon(i,id))
               y(i,id)=coor(2,ncon(i,id))
               z(i,id)=coor(3,ncon(i,id))
40          continue
10       continue
c
         return
         end

CNAME WRTMOV.FOR
c*****************************************************************************
         subroutine wrtmov(cname)
c*****************************************************************************
C
```

```
C  PROGRAM: WRTMOV.FOR
C
C  PURPOSE: WRITE THE MOVIE SURFACE GEOMETRY FILE
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*************************************************************************
c
c  called by: vmesh.for -- volume mesh controler
c
c*************************************************************************
      include 'para.inc'
      common/auto/coor,node,last,unit
      common/fin/ncon(0:10,maxid)
      common/bsur/isurf,icont
      dimension coor(4,maxnod),node(maxsid,12,maxloop),jp(4*maxid)
     $          ,isurf(maxid/4)
      character cname*(*)
c
      write(*,*)' Write movie geometry surface file...'
      imov = 14
      open(imov,file=cname//'.geo',status='new')
c
c check to see if the node was in boundary surface or not
c
      np = 1
      ic = 0
      do 10 id = 1,icont
        if(ncon(0,id).gt.0)then
          do 20 k = 0,3
            ic = ic+1
            jp(ic) = ncon(k,isurf(id))
            if(mod(ic,4).eq.0)jp(ic) = -jp(ic)
20        continue
        end if
10    continue
      nedge = ic
      nj = last
      npt = icont
c
c write movie polygon file
c
      write(imov,150) np,nj,npt,nedge
      write(imov,150) np,npt
      write(imov,180) ((coor(i,j),i=1,3),j=1,nj)
      write(imov,150) (jp(i),i=1,nedge)
c
150   format(10i8)
180   format(1p6e12.5)
c
      return
      end
CNAME WRTNAV.FOR
c*************************************************************************
      subroutine wrtnav(cname)
C*************************************************************************
C
C  PROGRAM: WRTNAV.FOR
C
C  PURPOSE:  write out the movie and navgraph solid geometry files
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*************************************************************************
c
c  called by:
c       vmesh.for -- volume meshing controler
c
c*************************************************************************
      include 'para.inc'
```

```
      dimension jp(4*maxid),invt(4)
      dimension coor(4,maxnod),node(maxsid,12,maxloop)
      character cname*(*)
      common/auto/coor,node,last,unit
      common/fin/ncon(0:10,maxid)
      data invt/0,3,2,1/
c
      write(*,*)' Write MOVIESTAR solid CADOS file'
      imov = 15
      inav = 16
      open(imov,file=cname//'.mov',status='new')
      open(inav,file=cname//'.nav',status='new')
c
      np = 1
      nj = last
      do 10 id = 1,maxid
         if(ncon(0,id).le.0)then
            npt = id-1
            go to 20
         end if
10    continue
20    continue
      ic = 0
      do 30 id = 1,npt
        if(mod(id,6) .eq. 1)then
          do 40 j = 0,3
            ic = ic+1
            jp(ic) = ncon(j,id)
40        continue
        else if(mod(id,6) .eq. 0)then
          do 50 j = 0,3
            ic = ic + 1
            jp(ic) = ncon(invt(j+1),id)
50        continue
        end if
30    continue
      nedge = ic
      npt = npt/6
c
c write movie solid element file
c
      write(imov,150) -np,nj,npt,nedge
      write(imov,150) np,npt
      write(imov,160) ((coor(i,j),i=1,3),j=1,nj)
      write(imov,150) (jp(i),i=1,nedge)
c
c write navgraph neutral file
c
      ntrnum = 0
      nmpnum = 1
      ngpnum = 0
      do 60 j = 1,nj
         write(inav,180)j,ntrnum,(coor(i,j),i=1,3),const
60    continue
      ibeg=1
      do 70 j = 1,npt
         iend = ibeg+7
         write(inav,190)j,nmpnum,ngpnum,(jp(i),i=ibeg,iend)
         ibeg = iend+1
70    continue
c
150   format(10i8)
160   format(1p6e12.5)
180   format('NODE,',2(I5,','),3(G14.7,','),G14.7,';')
190   format('ELEM,',I5,',LSO,',I5,',',I5,8(',',I5),';')
c
      return
      end
```

```
                ****************************************
                *   The following files are for the    *
                *   use of free-form deformation       *
                ****************************************

CNAME ALLDEF.FOR
c*********************************************************************
        subroutine alldef
C*********************************************************************
C
C  PROGRAM: ALLDEF.FOR
C
C  PURPOSE: DEFORMATION PROCESSOR
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: call 'deform' to perform the deformation
c
c*********************************************************************
c
c  called: deform.for -- deform the nodes
c
c*********************************************************************
c
c  called by: vmesh.for -- volume mesher
c
c*********************************************************************
        include 'para.inc'
        dimension x(0:maxdeg,maxid),y(0:maxdeg,maxid),
     $            z(0:maxdeg,maxid),ideg(maxid)
        common/data/x,y,z,ideg
c
c deformed all the data
c
        write(*,*)' Freeform deformation running, please wait...'
        id = 1
        do while (ideg(id).gt.0)
           do 20 i = 0,ideg(id)
             xi = x(i,id)
             yi = y(i,id)
             zi = z(i,id)
             call deform(xi,yi,zi,xo,yo,zo)
             x(i,id) = xo
             y(i,id) = yo
             z(i,id) = zo
20         continue
           id = id +1
        end do
c
        return
        end
CNAME FFDEFO.FOR
c*********************************************************************
        subroutine ffdefo
C*********************************************************************
C
C  PROGRAM: FFDEFO.FOR
C
C  PURPOSE: DEFORMATION PROCESSOR
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: call 'deform' to perform the deformation
c
c*********************************************************************
c
c  called: deform.for -- deform the nodes
c
c*********************************************************************
c
c  called by: vmesh.for -- volume mesher
c
c*********************************************************************
        include 'para.inc'
```

```
      dimension xx(0:maxdeg,maxid),yy(0:maxdeg,maxid),
     $          zz(0:maxdeg,maxid),ideg(maxid)
      common/data/xx,yy,zz,ideg
      dimension coor(4,maxnod),node(maxsid,12,maxloop)
      common/auto/coor,node,last,unit
      common/fin/ncon(0:10,maxid)
c
c deformed after meshing the cube surfaces
c
c     write(*,*)' Freeform deformation running, please wait...'
      do 10 j = 1,last
         xi = coor(1,j)
         yi = coor(2,j)
         zi = coor(3,j)
         call deform(xi,yi,zi,xo,yo,zo)
         coor(1,j) = xo
         coor(2,j) = yo
         coor(3,j) = zo
10    continue
c
c store the coordinate into common block for drawing
c
      id = 1
      do while (ideg(id).ne.0)
         do 20 i = 0,ideg(id)
            xx(i,id)=coor(1,ncon(i,id))
            yy(i,id)=coor(2,ncon(i,id))
            zz(i,id)=coor(3,ncon(i,id))
20       continue
         id = id+1
      end do
c
      return
      end
CNAME FFEDIT.FOR
C*****************************************************************
      subroutine ffedit
c*****************************************************************
      common/defor/xmin,ymin,zmin,xmax,ymax,zmax,l,m,n
      common/delta/dxyz(400)
c
      call editff(dxyz,l,m,n)
c
      return
      end

C*****************************************************************
      subroutine editff(dxyz,l,m,n)
c*****************************************************************
      dimension dxyz(4,0:l,0:m,0:n)
c
      write(*,100)l,m,n
      write(*,*)' Enter control point location number (I,J,K)'
      read(*,*)i,j,k
      write(*,101)dxyz(1,i,j,k),dxyz(2,i,j,k),dxyz(3,i,j,k),dxyz(4,i,j,k)
      write(*,*)' Enter new value (X,Y,Z,W)'
      read(*,*)dxyz(1,i,j,k),dxyz(2,i,j,k),dxyz(3,i,j,k),dxyz(4,i,j,k)
c
100   format(1x,'Control point dimension I =',i3,'J =',i3,'K =',i3)
101   format(1x,'Current control point value is ( ',4(f8.2,1x),')')
c
      return
      end
CNAME FRFORM.FOR
c*****************************************************************
      subroutine frform(cname,ierr)
C*****************************************************************
C
C  PROGRAM: FRFORM.FOR
C
C  PURPOSE: store data for free form deformation.
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*****************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*****************************************************************
```

```
c
c  called: ffread.for -- readin the deformation data
c
c*********************************************************************
c
c  called by: gtsoli.for -- solid mesh procesor
c
c*********************************************************************
        common/defor/xmin,ymin,zmin,xmax,ymax,zmax,l,m,n
        common/delta/dxyz
        dimension dxyz(400)
        character cname*(*)
        ierr = 0
c
c read in the deformation control points data
c
        open(2,file= cname//'.ffd',status='old',err=10)
          read (2,*)xmin,ymin,zmin,xmax,ymax,zmax,l,m,n
          kall = 4*(l+1)*(m+1)*(n+1)
          if(kall .gt. 400)then
             write(*,*)'...Error in FRFORM increase dxyz() to',kall
             stop
          end if
          call ffread(l,m,n,dxyz)
        go to 11
10      ierr=1
11      close(2)
c
c perform free form deformation to all the nodes for further processing
c
c       call ffdefo
c
        return
        end
CNAME FFREAD.FOR
c*********************************************************************
        subroutine ffread(l,m,n,d)
C*********************************************************************
C
C  PROGRAM: FRFORM.FOR
C
C  PURPOSE: read in data for free form deformation.
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*********************************************************************
c
c  called: none
c
c*********************************************************************
c
c  called by: frform.for -- deformation processor
c
c*********************************************************************
        dimension d(4,0:l,0:m,0:n)
c
        do 10 k = 0,n
        do 10 j = 0,m
           read (2,*)(d(1,i,j,k),d(2,i,j,k),d(3,i,j,k),
     $                d(4,i,j,k),i=0,l)
10      continue
c
        return
        end

CNAME NODEFO.FOR
c*********************************************************************
        subroutine nodefo
C*********************************************************************
C
C  PROGRAM: FRFORM.FOR
C
C  PURPOSE: deleted data of free-form deformation.
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
```

```
C  MODIFIED BY:                CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*****************************************************************************
c
c  called: none
c
c*****************************************************************************
c
c  called by: main.for -- user's interface
c
c*****************************************************************************
        common/defor/xmin,ymin,zmin,xmax,ymax,zmax,l,m,n
        xmin = 0.0
        xmax = 0.0
        ymin = 0.0
        ymax = 0.0
c
c deform all the nodes for further processing
c
        call ffdefo
c
        return
        end

CNAME PDEF.FOR
C*****************************************************************************
        subroutine pdef
C*****************************************************************************
C
C  PROGRAM: PDEF.FOR
C
C  PURPOSE: plot the free form deformation control points
c
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*****************************************************************************
c
c  called: none
c
c*****************************************************************************
c
c  called by: deform.for -- deformation processor
c
c*****************************************************************************
        common/defor/xmin,ymin,zmin,xmax,ymax,zmax,l,m,n
        common/delta/dxyz
        dimension dxyz(400)
c
        call pdefom(l,m,n,dxyz,xmin,ymin,zmin,xmax,ymax,zmax)
c
        return
        end
CNAME PDEFOM.FOR
c*****************************************************************************
        subroutine pdefom(l,m,n,d,xmin,ymin,zmin,xmax,ymax,zmax)
C*****************************************************************************
C
C  PROGRAM: PDEFOM.FOR
C
C  PURPOSE: calculate the free form deformation control points convex hull
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
c*****************************************************************************
c
```

```
c  called: circ.for -- draw control point
c          pipe.for -- draw convex hull
c
c*********************************************************************
c
c  called by: main.for -- user's interface
c
c*********************************************************************
        common/window/wxmin,wymin,wzmin,wxmax,wymax,wzmax
        common/vport/vxmin,vymin,vzmin,vxmax,vymax,vzmax
        dimension d(4,0:l,0:m,0:n),xc(0:200),yc(0:200),zc(0:200)
c
c for contral point circle
c
        r=5.0*(wxmax-wxmin)/(vxmax-vxmin)
        xdiff=(xmax-xmin)
        ydiff=(ymax-ymin)
        zdiff=(zmax-zmin)
        if(xdiff.ne.0.0 .and. ydiff.ne.0 .and. zdiff.ne.0)then
          do 30 k=0,n
           do 30 j=0,m
              do 30 i=0,l
                ijk = i + j*(l+1) + k*(m+1)*(l+1)
                xc(ijk) = (xmin + i*(xdiff)/l + d(1,i,j,k))
                yc(ijk) = (ymin + j*(ydiff)/m + d(2,i,j,k))
                zc(ijk) = (zmin + k*(zdiff)/n + d(3,i,j,k))
                call circ(xc(ijk),yc(ijk),zc(ijk),r,5)
                if(i.gt.0)call pipe(xc(ijk-1),yc(ijk-1),zc(ijk-1),
     $                                xc(ijk),yc(ijk),zc(ijk))
30         continue
          do 40 k=0,n
           do 40 j=1,m
             do 40 i=0,l
                ijk = i + j*(l+1) + k*(m+1)*(l+1)
                lst = i + (j-1)*(l+1) + k*(m+1)*(l+1)
                call pipe(xc(lst),yc(lst),zc(lst),
     $                    xc(ijk),yc(ijk),zc(ijk))
40         continue
          do 50 k=1,n
            do 50 j=0,m
              do 50 i=0,l
                ijk = i + j*(l+1) + k*(m+1)*(l+1)
                lst = i + j*(l+1) + (k-1)*(m+1)*(l+1)
                call pipe(xc(lst),yc(lst),zc(lst),
     $                    xc(ijk),yc(ijk),zc(ijk))
50         continue
        end if
c
        return
        end

c*************************************************************
        subroutine deform(x,y,z,xdu,ydu,zdu)
c*************************************************************
c
c  this subroutine calculate the free form deformation.
c
c*************************************************************
C*********************************************************************
C
C  PROGRAM: FRFORM.FOR
C
C  PURPOSE: calculate the free form deformation.
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*********************************************************************
C
C  ALGORITHM EXPLANATION: perform free-form deformation by Casteljau
c       algorithm
c
c*********************************************************************
c
c  called: none
c
c*********************************************************************
c
c  called by: deform.for -- deformation processor
c
```

```
c*******************************************************************
      common/defor/xmin,ymin,zmin,xmax,ymax,zmax,l,m,n
      common/delta/dxyz
      dimension dxyz(400)
c
      call exdefm(x,y,z,xdu,ydu,zdu,xmin,ymin,zmin,
     $            xmax,ymax,zmax,l,m,n,dxyz)
c
      return
      end
CNAME EXDEFM.FOR
c*******************************************************************
      subroutine exdefm(x,y,z,xdu,ydu,zdu,xmin,ymin,zmin,
     $            xmax,ymax,zmax,l,m,n,d)
C*******************************************************************
C
C  PROGRAM: FRFORM.FOR
C
C  PURPOSE: calculate the free form deformation.
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:          DATE:
C
C*******************************************************************
C
C  ALGORITHM EXPLANATION: perform free-form deformation by Casteljau
c       algorithm
c
c*******************************************************************
c
c  called: none
c
c*******************************************************************
c
c  called by: deform.for -- deformation processor
c
c*******************************************************************
      dimension d(4,0:l,0:m,0:n)
      dimension xd(0:10),yd(0:10),zd(0:10),wd 0:10)
      dimension xds(0:10),yds(0:10),zds(0:10) wds(0:10)
      dimension xdt(0:10),ydt(0:10),zdt(0:10) wdt(0:10)
c
      xdiff=(xmax-xmin)
      ydiff=(ymax-ymin)
      zdiff=(zmax-zmin)
c
c if there is no defomation boundary then return the same values
c
      if(xdiff.eq.0.0 .or. ydiff.eq.0.0 .or. zdiff.eq.0.0)then
         xdu=x
         ydu=y
         zdu=z
      else
         s=(x-xmin)/xdiff
         t=(y-ymin)/ydiff
         u=(z-zmin)/zdiff
c
c if the point in the defomation boundary then deform it
c
      if(s.ge.-0.005 .and.s.le.1.005 .and. t.ge.-0.005.and.t.le.1.005
     $      .and. u.ge.-0.005 .and. u.le.1.005)then
         do 9 k=0,n
           do 10 j=0,m
              do 20 i=0,l
                wd(i) = d(4,i,j,k)
                xd(i) = (xmin + i*(xdiff)/l + d(1,i,j,k))*d(4,i,j,k)
                yd(i) = (ymin + j*(ydiff)/m + d(2,i,j,k))*d(4,i,j,k)
                zd(i) = (zmin + k*(zdiff)/n + d(3,i,j,k))*d(4,i,j,k)
20            continue
              wds(j)=point(wd,l,s)
              xds(j)=point(xd,l,s)
              yds(j)=point(yd,l,s)
              zds(j)=point(zd,l,s)
10         continue
              wdt(k)=point(wds,m,t)
              xdt(k)=point(xds,m,t)
              ydt(k)=point(yds,m,t)
              zdt(k)=point(zds,m,t)
9        continue
              wdu=point(wdt,n,u)
```

```
                xdu=point(xdt,n,u)
                ydu=point(ydt,n,u)
                zdu=point(zdt,n,u)
                xdu=xdu/wdu
                ydu=ydu/wdu
                zdu=zdu/wdu
c
c if the point out of the defomation boundary then no deformation
c
           else
                xdu=x
                ydu=y
                zdu=z
           end if
        end if
c
        return
        end

CNAME CIRC.FOR
c*******************************************************************************
        subroutine circ(xc,yc,zc,r,n)
C*******************************************************************************
C
C  PROGRAM: CIRC.FOR
C
C  PURPOSE: DRAW CIRCLE ON THE GIVEN POINT
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
C*******************************************************************************
C
C  ALGORITHM EXPLANATION:
C
C*******************************************************************************
c parameter:
c       xc      x-coordinate center of the circle
c       yc      y-coordinate center of the circle
c       r       radious of the circle
c       n       segments needs to form a circle
c*******************************************************************************
c
c  called: pipe.for -- graphics line ploter
c
c*******************************************************************************
c
c  called by: pdefor.for -- plot control point for free-form deformation
c
c*******************************************************************************
        dimension x(50),y(50)
        real r
c
        pi=acos(-1.0)
        if(n.le.3) n=3
        a=2.0*pi/n
        do 10 i=1,n
           x(i)=xc + r*cos(a*i)
           y(i)=yc + r*sin(a*i)
10      continue
c
        do 20 j=1,n-1
           call pipe (x(j),y(j),zc,x(j+1),y(j+1),zc)
20      continue
           call pipe (x(n),y(n),zc,x(1),y(1),zc)
c
        return
        end
CNAME PUSH.FOR
c*******************************************************************************
        subroutine push(loop,ipt)
C*******************************************************************************
C
C  PROGRAM: PUSH.FOR
C
C  PURPOSE: PUT THE DATA INTO STACK
C
C  WRITTEN BY:  SSUTA HSU       CHECKED BY:            DATE: OCT. 6, 1988
C  MODIFIED BY:                 CHECKED BY:            DATE:
C
```

```
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
C*****************************************************************************
C
C  CALLED: NONE
C
C*****************************************************************************
C
C  CALLED BY: MOST OF THE ROUTINE
C
C*****************************************************************************
        include 'para.inc'
        common/stack/iarray(maxloop)
c
           ipt = ipt + 1
           iarray(ipt)=loop
c
        return
        end

CNAME POP.FOR
c*****************************************************************************
        subroutine pop(loop,ipt)
C*****************************************************************************
C
C  PROGRAM: PUSH.FOR
C
C  PURPOSE: POP THE DATA OUT OF STACK
C
C  WRITTEN BY:  SSUTA HSU      CHECKED BY:          DATE: OCT. 6, 1988
C  MODIFIED BY:                CHECKED BY:          DATE:
C
C*****************************************************************************
C
C  ALGORITHM EXPLANATION:
c
C*****************************************************************************
C
C  CALLED: NONE
C
C*****************************************************************************
C
C  CALLED BY: MOST OF THE ROUTINE
C
C*****************************************************************************
        include 'para.inc'
        common/stack/iarray(maxloop)
c
           loop=iarray(ipt)
           ipt = ipt - 1
c
        return
        end
```

```
c
c maximum internal node number on each surface for 2-D case
c
      parameter (intnod=1000)
c
c maximum allowable node number on each line
c
      parameter (maxsid=100)
c
c maximum degree of a curve or line segments
c
c      parameter (maxdeg=20) for 2-D case
      parameter (maxdeg=4)
c
c maximum number of polygons = maxele*6
c
      parameter (maxid=12000)
c
c maximun number of elements
c
      parameter (maxele=2000)
c
c maximum number of nodes in any mesh
c
      parameter (maxnod=100000)
c
c  maximum number of stack necessary for the loops
c
      parameter (maxloop=20)
c
c set pi for processing 2-D case
c
      parameter (pi=3.141593)
      parameter (twopi=6.283185)
      parameter (halfpi=1.570795)
c
c set zero for error measurement
c
      parameter (zero=0.0001)
```

AUTOMATIC MESHING OF FREE-FORM DEFORMATION SOLIDS

Ssuta S. Hsu

Department of Civil Engineering

Ph.D. Degree, April 1989

ABSTRACT

Development of computer models and subsequent finite element analysis, are important aspects of modern engineering design. In this process, the geometry creation and finite element analysis software are well developed; however, the process of discretizing a geometry into a proper finite element model is time consuming and tedious.

The work presented here uses the free-form deformation method to create smooth solid models, and invokes a solid subdivision and transition method to generate the hexahedron finite elements. The combination of these two techniques provides an automatic mesh generator that is easy to use, creates acceptable hexahedron elements for finite element analysis, and can model basically any complex shape.

COMMITTEE APPROVAL:

Michael B. Stephenson, Committee Chairman

Thomas W. Sederberg, Committee Member

Steven E. Benzley, Committee Member

LaVare Merritt, Department Chairman