Hybrid Mesh Generation for Viscous Flow Simulation

Yuanli Wang[1] and Samuel Murgie[2]

[1] ALGOR, Inc., 150 Beta Dr., Pittsburgh, PA USA, 15238
ywang@algor.com
[2] ALGOR, Inc., 150 Beta Dr., Pittsburgh, PA USA, 15238
smurgie@algor.com

Abstract. This paper presents a robust and automated approach to generate unstructured hybrid grids comprised of prismatic and tetrahedral elements for viscous flow computations. The hybrid mesh generation starts from a triangulated surface mesh. The prismatic elements are extruded based on the weak solutions of the Eikonal equation to generate anisotropic elements at boundaries, and finally the isotropic tetrahedral grids are generated to fill the rest of the domain. The presented hybrid meshing algorithm was validated using a ball valve model under both steady and unsteady conditions.

1 Introduction

A great challenge for viscous flow simulations is to gain anisotropic elements at the vicinity of the boundary area, especially for complex geometric solid surfaces. When such domains are discretized, it is important that the grid fits the boundaries well, and no conflict occurs during boundary mesh generation processes. These issues become even more difficult to achieve with strongly curved boundaries. The first issue relates to surface discretization or surface mesh generation, which is not discussed in this paper.

To effectively avoid the warping of normal directions during boundary meshing, it is critical to generate good-quality, high-aspect-ratio cells in the vicinity of boundaries for wall-dominated phenomena. For this purpose, a new approach based on a hybrid meshing methodology was proposed, in which normal directions are calculated using a weak solution of the Eikonal equation at each solid surface node of the boundaries to be propagated [1], [5], and [6].

The purpose of this paper is to validate the quality and effectiveness of the proposed new hybrid mesh generation strategy using a ball valve model. This paper is arranged as follows: Section 2 describes the problem to be solved. Section 3 gives a brief introduction of the methodology used in this work.

Section 4 illustrates the mesh results. Validation of the flow using the presented method is demonstrated in Section 5. Finally, Section 6 summarizes the presentation.

2 Problem Description

In this paper, the steady and unsteady fluid flows in a three-dimensional (3D) model of a ball valve are computed. Figure 1 shows the geometric image of a pipe section including a ball valve. The valve is in the half open position. The goal of the fluid flow analysis is to determine the velocity and the pressure of the fluid as it exits the section.

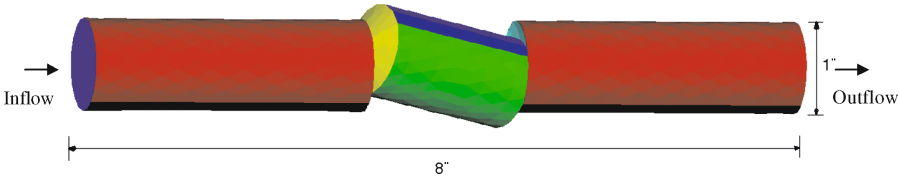

Fig. 1. Diagram of the pipe section

Topologically, the illustrated geometry is very simple and is equivalent to a cylinder. Geometrically, this configuration has both convex and concave shapes in its body. The most difficult consideration in this test case is how to correctly calculate normal directions at the four singularity points as shown in Figure 2. In this case, there are four surfaces that pass through and share the singularity point. The traditional normal calculation method uses the geometric information surrounding the point to be propagated, i.e., the weighted average normal vectors of neighboring surfaces. The definition of the normal vector may become ambiguous at special cases when normal direction is performed in this way. For example, in Figure 2, the definition of normal vectors for Surface 2 and Surface 4 are almost in opposite directions. It is difficult to determine a compromise normal vector at this point using the average normal vectors of neighboring surfaces.

Instead of using the traditional normal calculation, this paper uses a new way to calculate normal vectors to prevent the above-mentioned problem. The propagation strategy is described in Section 3, and the resulting meshes at singularity points are presented in Section 4.

To analyze the flow pattern within this system, geometric sizes are set as illustrated in Figure 1. The total length of the pipe section is 8.0 in and the diameter at intake areas and valves is 1.0 in. Several physical characteristics of the fluid are also pre-defined, including the flow rate at the inlet area as 0.5 in/s and the fluid as water.

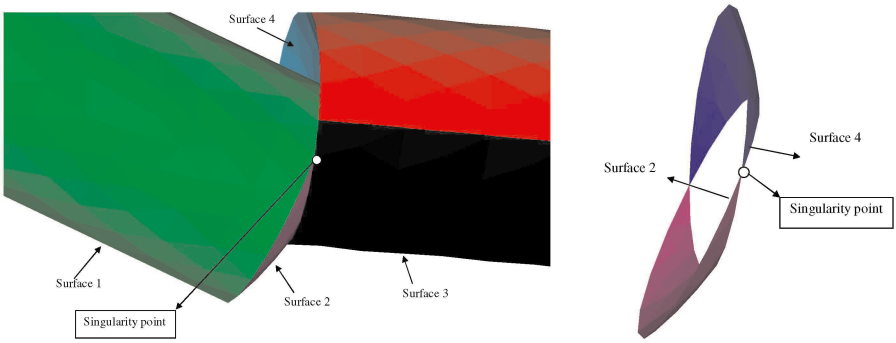

Fig. 2. Singularity point

3 Methodology

A primary challenge that exists in dealing with viscous flow concerns boundary mesh generation. Many boundary mesh generation methods have been proposed, and two important problems still exist. The first problem relates to the potential self-intersection of the front as it grows from the original surface. *Local* self-intersection may occur during propagation when the offset distance is greater than the local curvature radius in concave regions. *Global* self-intersection, on the other hand, arises when the distance between two distinct points on the curve or surface reaches a local minimum. The second difficulty, which is a more recent issue in offset construction, is the establishment of a common connectivity between the original and offset surfaces.

Two major types of methods in dealing with the computation of offset curves and surfaces are proposed: *direct offset methods (DOM)*, which propagate curves or surfaces directly based on a geometric construction; and *indirect offset methods (IOM)*, which cast the curve or surface offset problem into a set of partial differential equations (PDE), in which, geometric information is implicitly represented.

The advantage of DOM is that the entire or partial original parameterization information can be preserved (which is a rather attractive merit to boundary meshing), but the self-intersection problem cannot be avoided, and extra care is required for removing self-intersections. The ability to effectively eliminate these self-intersections is an important criterion in the applicability of such methods in the context of an automated procedure. One representative attempt to eliminate self-intersections is the *Advancing Front Method* developed by Pirzadeh [9] based on a grid-marching strategy. The solution is to simply stop the advancement of the front before self-intersections occur. Based on a similar marching idea, Sullivan [10] presented a self-intersection

detection and removal algorithm in 2D. The 3D algorithm developed by Guibault [8] eliminates self-intersections by first detecting tangled-loops and then re-locating the points located within this area. In the algorithm described by Glimm et al [11], a hybrid algorithm is applied to resolve self-intersections by either re-triangulating triangles after removing unphysical surfaces, or reconstructing the interface within each rectangular grid block in which crossing is detected. Other types of techniques to eliminate self-intersections use the properties of curves and surfaces, i.e., control points, derivatives, curvature, etc. Blomgren [12], Tiller and Hanson [13], and Coquillar [14] approached the problem by offsetting the control polygon for NURBS curves. Nachman [15] extended this idea to propagate surfaces by offsetting control points. Piegl and Tiller [16] sampled offset curves and surfaces based on bounds on the second derivatives to avoid self-intersections. In the method developed by Sun et al. [17], control points are repositioned to reduce local curvature in areas where local self-intersections may occur, while the rest of the control points remain unchanged. Farouki [18] described an algorithm which first decomposes the original surfaces into parametric patches, and then uses Hermite interpolation to construct the offset surfaces.

The representative work of IDM is the *level set method* developed by Sethian and Osher [19, 20, 21, 22] which models front propagation problems as a hyperbolic differential equation. In this method, self-intersection problems can be avoided, but at the cost of completely losing the connectivity information stored in the original geometric front. In general, to restore a similar connectivity between the original and offset fronts is not a trivial task.

The present work proposes to use a boundary mesh generation method discussed in [1], [5] and [6], which combines the advantages of the two types of methods to build a new offset construction method that maintains parametric connectivity between the original and offset surfaces, and still avoids self-intersections through the use of a weak solution to the shortest distance problem. Figure 3 shows the outline of the proposed mesh generation process. The details for each step are explained below.

Surface Mesh Generation

In the present hybrid meshing strategy, the surfaces are discretized into triangles [23, 24]. Figure 4 shows the initial surface mesh used in this test. There are no restrictions for the shape of surface mesh elements. The proposed boundary mesh process can accept any type of elements, e.g., triangular, quad, etc.

Boundary Mesh Generation

There are four essential steps in the boundary meshing process: (1) calculate the ϕ value for each grid node using the fast sweeping algorithm; (2) calculate normal directions for the front grid nodes; (3) propagate front points along

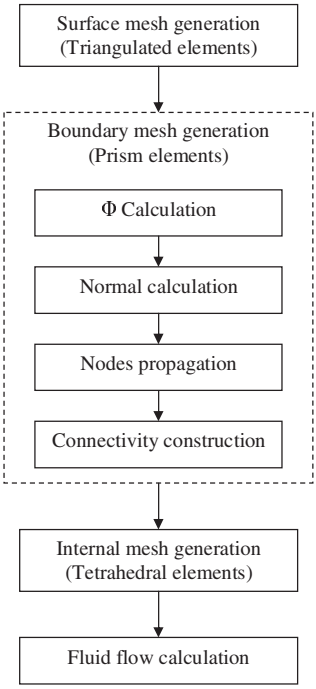

Fig. 3. Outline of mesh generation process

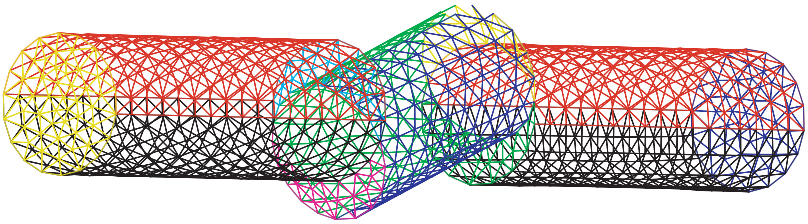

Fig. 4. Surface mesh

their local normal directions according to the given distance; (4) construct blocks or the relative topological connectivity around the boundary area.

1. ϕ calculation
   The present work proposes using the *Offset Distance Equation*, which is a variation of the Eikonal equation, to model the surface offset problem. This equation is given below:

$$\begin{cases} \nabla\phi \cdot \nabla\phi = 1 \\ \phi = 0 \qquad if\ \mathbf{P} \in \Gamma \end{cases} \tag{1}$$

   where ϕ is the minimum Euclidean distance from an arbitrary point (P) in the computational domain to the front (Γ) to be propagated. Equation 1

expresses the condition for the shortest Euclidean distance from any space position to the boundary. To enforce the boundary condition, $\phi = 0$ when $\mathbf{P} \in \Gamma$, exact values are first assigned to boundary nodes. Then, Gauss-Seidel iterations with alternating sweeping orderings are used to update the ϕs at the rest of the grid nodes. The details of this fast sweeping algorithm can be found in [2].

2. Normal calculation
   At each grid node, the normal vector is represented by the equation

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \tag{2}$$

   where $\mathbf{n}$ is the normal direction and ϕ is the weak solution of the Offset Distance Equation. The term $\nabla\phi/|\nabla\phi|$ can be viewed as a unit propagation speed in the normal direction: positive for an outward propagation, and negative for an inward propagation.
3. Nodes propagation
   The propagation equation at each node is:

$$\mathbf{x}_t = F\mathbf{n} \tag{3}$$

   Setting the propagation speed F at each node to 1.0, the new propagated points are obtained by iteratively solving Equation 3 using the fourth-order Runge-Kutta method [3].
4. Connectivity construction
   All the propagated points are sequentially connected according to their original connectivities. The final propagation surface forms the input surface mesh for the volume mesh generation process. Figures 5 and 6 show the overall boundary mesh interface and the final boundary mesh of this ball valve model.

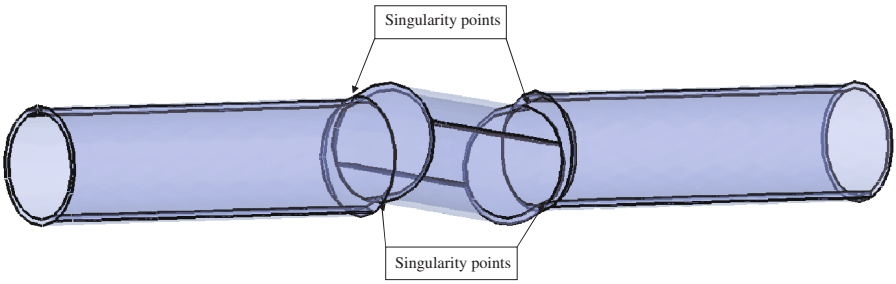

Fig. 5. Boundary mesh interface

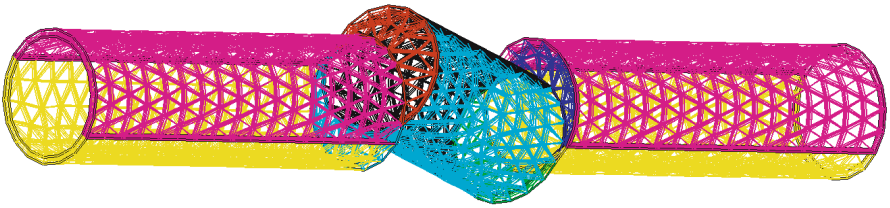

Fig. 6. Boundary mesh

Internal Mesh Generation

After the boundary meshing process, a new surface mesh is generated, which possesses exactly the same topological definition (triangle connectivity) as the original solid surface. This new surface mesh becomes the input surface mesh for tetra mesh generation. The entire computational domain defined by this surface mesh is tessellated by isotropic tetrahedra [7, 4]. Figures 7 and 8 illustrate the final mesh generated for this model, and Figures 9 shows the final mesh at the middle section.

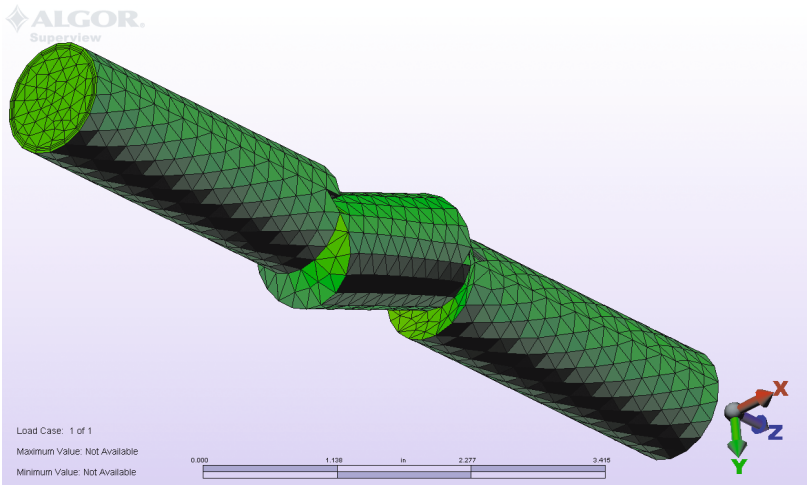

Fig. 7. Final mesh (outside view)

4 Mesh validation

This section will only evaluate the quality of the resulting mesh for prismatic elements.

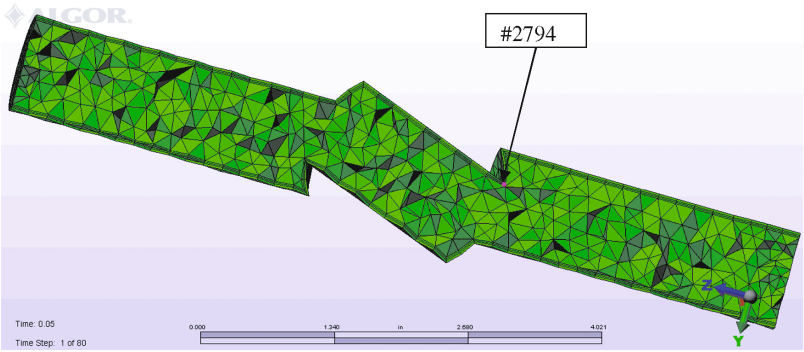

Fig. 8. Final mesh (internal view)

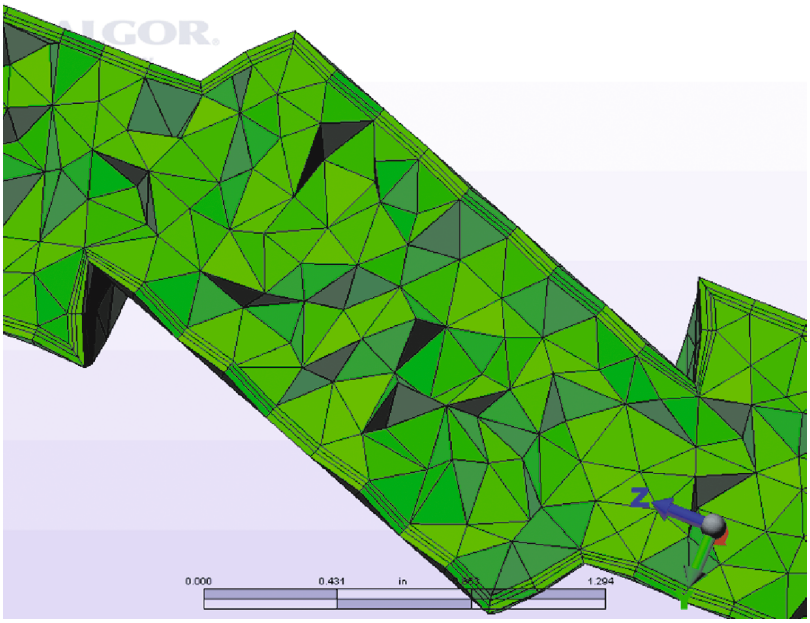

Fig. 9. Final mesh (in the middle section)

Volume Distribution

For validation, the original boundary is decomposed into 12 surfaces, and three layers are generated after propagation. Table 1 illustrates the size of the minimum and maximum volume for prismatic elements at each layer. From this table, we can see that the volume of the prisms is reduced when marching proceeds toward the internal direction.

Figure 10 illustrates partial volume distribution of the prismatic elements. Because the normal directions at the surface mesh are defined toward the

Table 1. Prismatic element volume range

Layer 1		Layer 2		Layer 3	
Min. Size	Max. Size	Min. Size	Max. Size	Min. Size	Max. Size
4.776e-5	4.516e-4	2.374e-5	2.851e-4	1.504e-5	2.701e-4

outward direction of the domain, all the volume values are represented in negative values. In this figure, blue color represents maximum volume, and red represents minimum when absolute values are used. This figure shows that the quality of the surface mesh can greatly affect the quality of the boundary mesh. As a more uniform distribution is generated for the original surface mesh, the quality of the prismatic elements improves. Table 2 shows the overall volume range generated for this model.

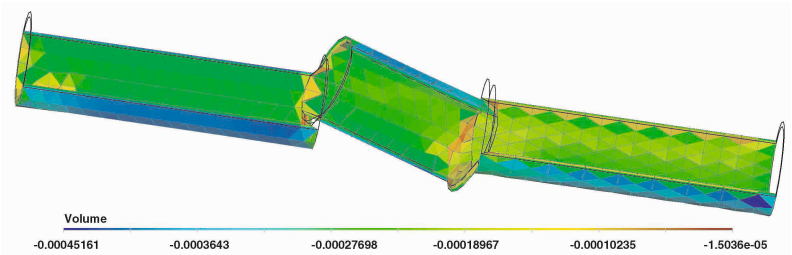

Fig. 10. Volume distribution of prismatic elements

Table 2. Overall volume range

Type	Min. Size	Max. Size
Prismatic element	1.503e-5	4.516e-4
Tetrahedral element	2.904e-5	1.802e-3

Aspect Ratio

The definition of aspect ratio for a prism element is:

$$aspect\ ratio = \frac{averaged\ beam\ length}{averaged\ circumcircle\ radius} \tag{4}$$

Table 3 illustrates the histogram of aspect ratio for prismatic elements. From the table we can see that the averaged aspect ratio is 0.012632. The reason of this boundary mesh has such a small aspect ratio value is that the initial surface mesh is very coarse. Refining the surface mesh size or increasing the offset distance will help to increase the aspect ratio.

Table 3. Histogram of aspect ratio

Range	Number of prisms	%
0.003443 - 0.007660	2241	42.395006
0.007660 - 0.011877	1010	19.107075
0.011877 - 0.016093	562	10.631858
0.016093 - 0.020310	423	8.002270
0.020310 - 0.024527	362	6.848278
0.024527 - 0.028743	293	5.542944
0.028743 - 0.032960	183	3.461975
0.032960 - 0.037177	68	1.286417
0.037177 - 0.041393	64	1.210745
0.041393 - 0.045610	80	1.513432
total	**5286**	**100**
Averaged aspect ratio = 0.012623		

Table 4. Histogram of normalized equiangular skewness

Value of normalized equiangular skewness	Number of prisms	%	Cell quality
0.000000 - 0.250000	710	13.431706	Excellent
0.250000 - 0.500000	3827	72.398789	Good
0.500000 - 0.750000	391	7.396897	Fair
0.750000 - 0.900000	332	6.280742	Poor
0.900000 - 1.000000	26	0.491865	Bad
total	**5286**	**100**	
Averaged normalized equiangular skewness = 0.16386			

Normalized Equiangular Skewness

Thedefinitionofnormalizedequiangularskewness(NES)andcellquality can be found in [25]. According to [25], 0 indicates the ideal element, 1 indicates the bad element, and the acceptable NES range in 3D is 0-0.4. Table 4 illustrates the histogram of NES. From this table, we can see that the averaged normalized equiangular skewness is 0.16386 which falls in the acceptable range.

Sharp Corner Behavior

Theoretically, this boundary mesh generation method allows the propagation to proceed to any distance without losing original connectivity at sharp corners. However, the cost is that some of the elements may degrade down to zero-volume elements under certain circumstances.

For example, Figure 11 shows the propagation behavior at one sharp corner in two dimensions. The boundary line is propagated toward its inner direction

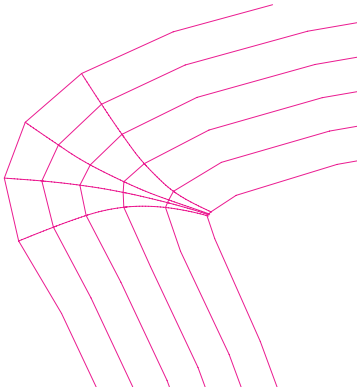

Fig. 11. Propagation behavior at a sharp corner

(from left to right), the propagation paths are skewed together after a certain distance. The post-redistribution algorithm is required to avoid zero-volume elements during the boundary mesh generation process.

Singularity Point Propagation Behavior

Figure 12 shows the propagation behavior at singularity points. The upper figure illustrates the enlarged view of the boundary mesh interface at the part

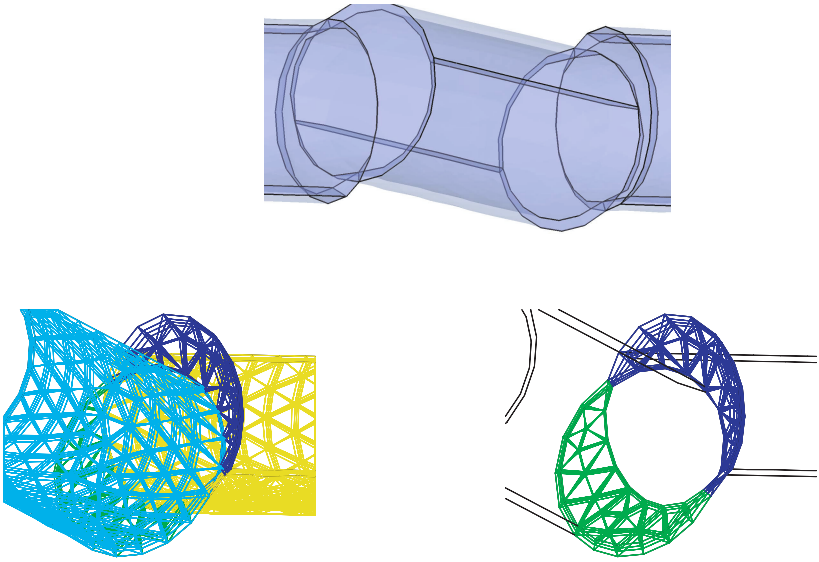

Fig. 12. Mesh at singularity points

with singularity points. The two bottom figures show the mesh at one singularity point. From these figures, we can see that there is only one propagation path at the singularity point, which is shared by the four surfaces' meshes. The reason we can get unique normal directions at the singularity points is because the ϕ value at any point in the computational domain is a unique value. This means the normal vector, $\frac{\nabla\phi}{|\nabla\phi|}$, is a unique value at any point of the computational domain. Thus, in this case, the proposed method can avoid the problem caused by the traditional normal calculation method.

5 Flow Analysis

The flow calculation is performed by ALGOR Professional Multiphysics, which is finite element analysis (FEA) software developed by ALGOR, Inc. Fluid enters the pipe, which is connected to a ball valve in the half-open position. Since the Reynolds number is very low - the viscous force is dominant - a viscous layer on model boundaries is expected in the flow. Two analyses (steady and unsteady flow) are performed using this mesh. Their velocity and pressure analysis results are shown below, respectively. In the illustrated results, dark blue color represents low value, and red represents high value.

Stead Flow

The flow at the inlet is set as a uniform velocity in the direction of the z-axis. The expected velocity solution in the domain is shown in Figures 13 and 14. As predicted, a vortex flow pattern develops in the downstream area. Figure 15 illustrates an enlarged view of Figure 14. Figure 16 is the internal pressure distribution which is coincident with the anticipated result.

Unsteady Flow

In this case, the unsteady solver is applied. Figures 17 and 18 show the velocity result plotted by magnitude and vector, respectively. From these figures, we can see that the final fluid velocity distribution demonstrates a similar behavior as the result calculated by the steady solver when a sufficient physical time is predefined. Figure 19 is an enlarged view of Figure 18 which illustrates the vortex pattern developed in the downstream area. Figure 20 shows the velocity variation at node #2794, which is a singularity point. Its position is illustrated in Figure 8. All the other nodes demonstrate the same convergent behavior. After a certain iteration time, the variation of velocity at any computational node converges to a steady value. Figure 21 illustrates the internal pressure distribution of this model.

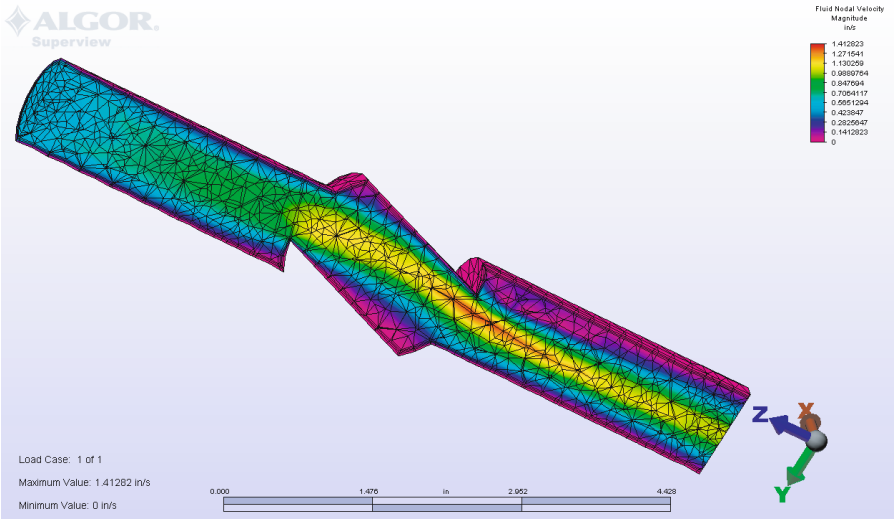

Fig. 13. Velocity distribution (plot by magnitude)

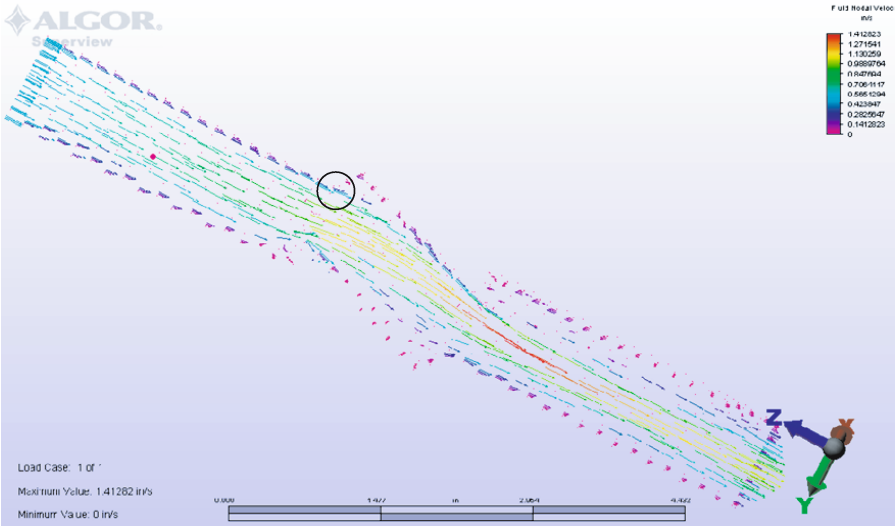

Fig. 14. Velocity distribution (plot by vector)

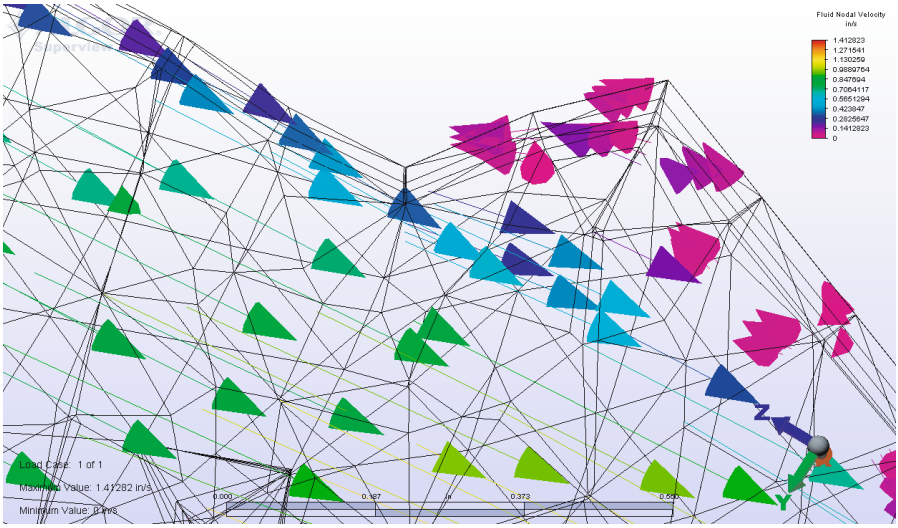

Fig. 15. Vortex pattern (enlarged view)

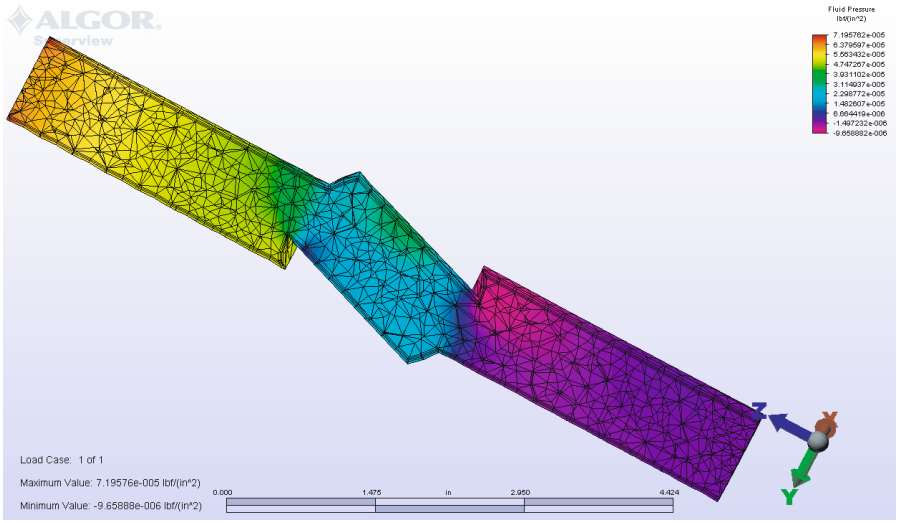

Fig. 16. Pressure distribution

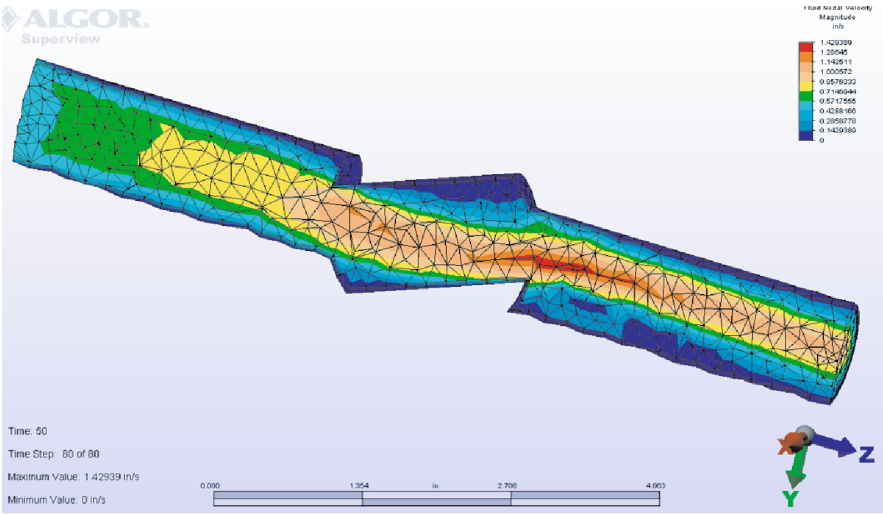

Fig. 17. Velocity distribution (plot by magnitude)

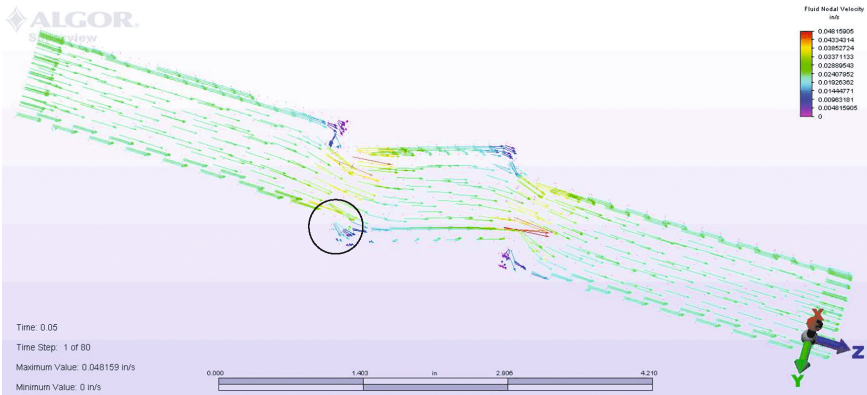

Fig. 18. Velocity distribution (plot by vector)

6 Conclusion

An automated hybrid grid generation method has been developed. The boundary anisotropic elements are extruded along the normal directions which are represented by the weak solution of the Eikonlal equation. The method was applied to the ball valve model for both steady and unsteady fluid flow analysis using ALGOR Professional Multiphysics software. The preliminary results show that the proposed method practically generated well-qualified grid distribution for viscous flow.

Further research and development work needs to be done in the following areas: 1) A higher-order numerical solution of the Eikonal equation is to be

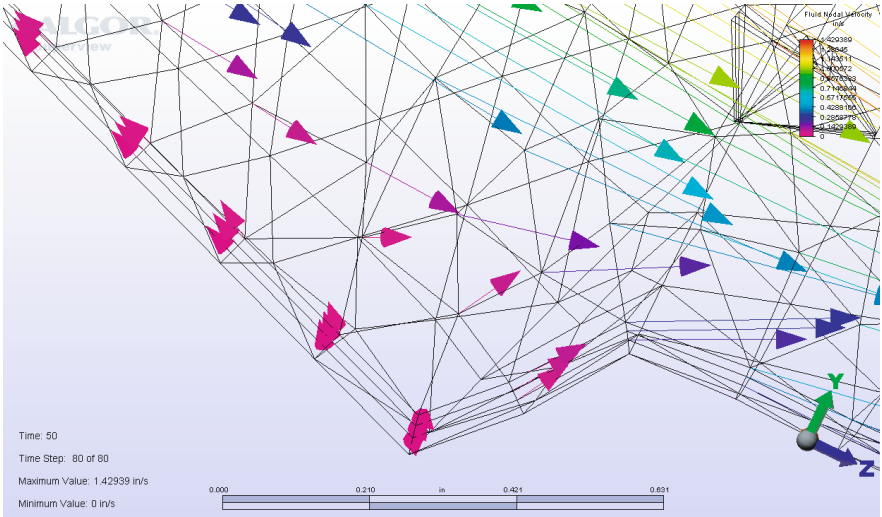

Fig. 19. Vortex pattern (enlarged view)

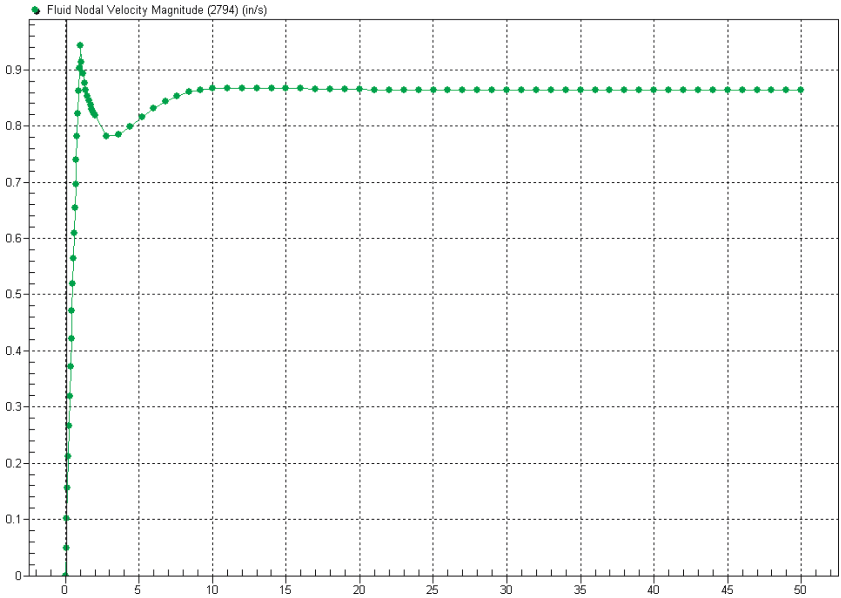

Fig. 20. Velocity variation at node #2794

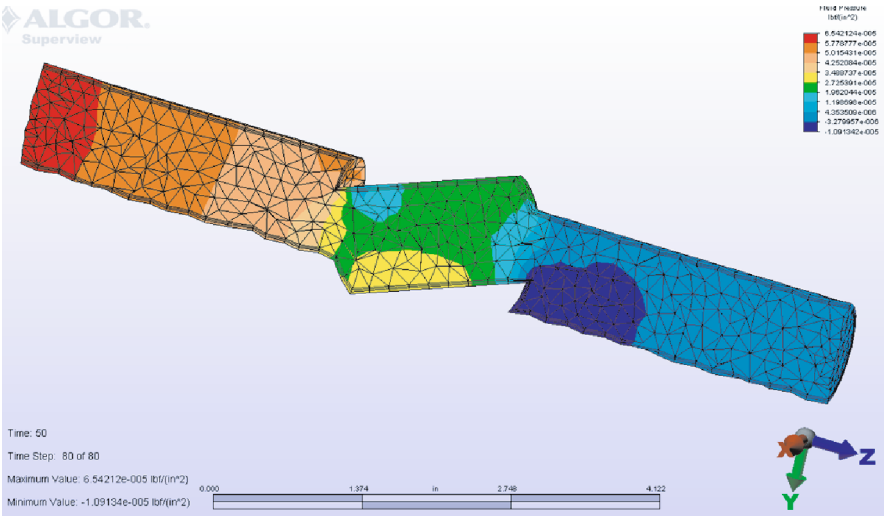

Fig. 21. Pressure distribution

explored. The benefit of doing this is to get more accurate normal calculation which will obviously strengthen control ability during the propagation process; 2) In this work, the propagation stops after three propagations in order to avoid zero-volume elements. Thus, a mesh optimization procedure is to be added as a complementary work of the presented method.

Acknowledgements

Many thanks to Zhongman Ding from ALGOR, Inc. for his discussions, and Minli Chi for his help and suggestions in writing this paper.

References

1. Y. L. Wang, F. Guibault, R. Camarero, and K. -F Tchon, "A parametrization transporting surface offset construction method based on the eikonal equation," in *17th AIAA CFD Conf.*, June 2005.
2. H. K. Zhao, "A fast sweeping method for eikonal equations," *Mathematics of Computation*, vol. 74, no. 250, pp. 603–627, 2005.
3. J. Hoffman, *Numerical Methods for Engineers and Scientists.* McGraw-Hill Inc., 1992.
4. H. Borouchaki, Hecht, F., E. Saltel and P. L. George, "Reasonably Efficient Delaunay Based Mesh Generator in 3 Dimensions," in *Proceedings 4th International Meshing Roundtable*, Sandia National Laboratories, pp. 3–14, 1995
5. Y. L. Wang, F. Guibault, and R. Camarero, "Automatic near-body domain decomposition using the eikonal equation," in *Proceedings of the 14th International Meshing Roundtable*, Sandia National Laboratory, Sept. 2005.

6. Y. L. Wang, F. Guibault, and R. Camarero (2006) *Int. J. Numer. Meth. Engng*, Submitted.
7. P. L. George, F. Hecht, and E. Saltel, "Automatic Mesh Generator with Specified Boundary," in *Computer Methods in Applied Mechanics and Engineering*, North-Holland, Vol 92, pp. 269–288, 1991
8. F. Guibault, *Un mailleur hybride structuré/non structuré en trois dimensions.* PhD thesis, École Polytechnique de Montréal, 1998.
9. S. Pirzadeh, "Unstructured viscous grid generation by advancing-layers method," in *AIAA 11th Applied Aerodynamics Conference*, no. AIAA-93-3453, (Monterey, CA), pp. 420–434, Aug. August 9–11, 1993.
10. J. Suillivan and J. Zhang, "Adaptive mesh generation using a normal offsetting technique," *Finite Elements in Analysis and Design*, vol. 25, pp. 275–295, 1997.
11. J. Glimm, J. Grove, X. Li, and D. Tan, "Robust computational algorithms for dynamic interface tracking in three dimensions," *SIAM J. Sci. Comp.*, vol. 21, no. 6, pp. 2240–2256, 2000.
12. R. Blomgren, "B-spline curves, boeing document, class notes, b-7150-bb-wp-2811d-4412," 1981.
13. W. Tiller and E. Hanson, "Offsets of two-dimensional profiles," *IEEE Computer Graphics and Applications*, vol. 4, no. 9, pp. 36–46, 1984.
14. S. Coquillart, "Computing offsets of b-spline curves," *Comp.-Aided Design*, vol. 19, no. 6, pp. 305–309, 1987.
15. A. Kulczycka and L. Nachman, "Qualitative and quantitative comparisons of b-spline offset surface approximation methods," *Comp.-Aided Design*, vol. 34, pp. 19–26, Jan. 2002.
16. L. A. Piegl and W. Tiller, "Computing offsets of nurbs curves and surfaces," *Comp.-Aided Design*, vol. 31, pp. 147–156, Feb. 1999.
17. Y. F. Sun, A. Y. C. M. Nee, and K. S. Lee, "Modifying free-formed nurbs curves and surfaces for offsetting without local self-intersection," *Comp.-Aided Design*, vol. 36, no. 12, pp. 1161–1169, 2004.
18. R. T. Farouki, "The approximation of non-degenerate offset surfaces," *Comp.-Aided Geom. Design*, vol. 3, no. 1, pp. 15–43, 1986.
19. S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *J. Comp. Phys.*, vol. 79, pp. 12–49, 1988.
20. J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 4, pp. 1591–1595, 1996.
21. J. A. Sethian, *Level set methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge University Press, 1999.
22. J. A. Sethian, "Fast marching methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.
23. T. D. Blacker and M. B. Stephenson, "Paving: A new approach to Automated Quardrilateral Mesh Generation," *SIAM Journal on Numerical Analysis*, vol. 32, pp. 811–847, 1991.
24. R. J. Cass and S. E. Benzley and R. J. Meyers and T. D. Blacker, "Generalized 3-D Paving: An Automated Quadrilateral Surface Mesh Generation Algorithm," *SIAM Journal on Numerical Analysis*, vol. 39, no. 9, pp. 1475–1489, 1998.
25. "TGrid 3.5 User's Manual," *Fluent Inc.*, April, 2003.