International Journal of Cast Metals Research

ISSN: 1364-0461 (Print) 1743-1336 (Online) Journal homepage: http://www.tandfonline.com/loi/ycmr20

Development of pre-/post-processors for 3D numerical simulation using cut cell method

S-Y Park, C-H Lim, J-H Nam, H-Y Shin & J-K Choi

To cite this article: S-Y Park, C-H Lim, J-H Nam, H-Y Shin & J-K Choi (2011) Development of pre-/post-processors for 3D numerical simulation using cut cell method, International Journal of Cast Metals Research, 24:3-4, 257-262, DOI: 10.1179/136404611X13001922708911

To link to this article: http://dx.doi.org/10.1179/136404611X13001922708911

Published online: 12 Nov 2013.

Development of pre-/post-processors for 3D numerical simulation using cut cell method

S.-Y. Park[1], C.-H. Lim*[2], J.-H. Nam[2], H.-Y. Shin[1] and J.-K. Choi[2]

Today, in the casting engineering field, various heat flow simulations are being applied to manufacturing processes. The most general flow simulation techniques have been improved and developed according to the characteristics where they are applied in the casting industry. In this study, the authors have developed a pre-/post-processing system to use a cut cell method, which is one of the finite volume methods among the numerical analysis methods mentioned above. In this paper, the authors will briefly introduce a pre-processor that requires only the user's simple input information to automatically generate meshes and performs simulation and post-processing, which maps the simulation result data into an original computer aided design file directly. This paper will also examine a few cases of actual application of these technologies to casting products.

Keywords: Simulation, Finite volume method, Pre-processing, Post-processing, Mesh generation

This paper is part of a special issue of papers selected from MCSP8, the 8th Pacific Rim Conference on Modeling of Casting and Solidification Processes

Introduction

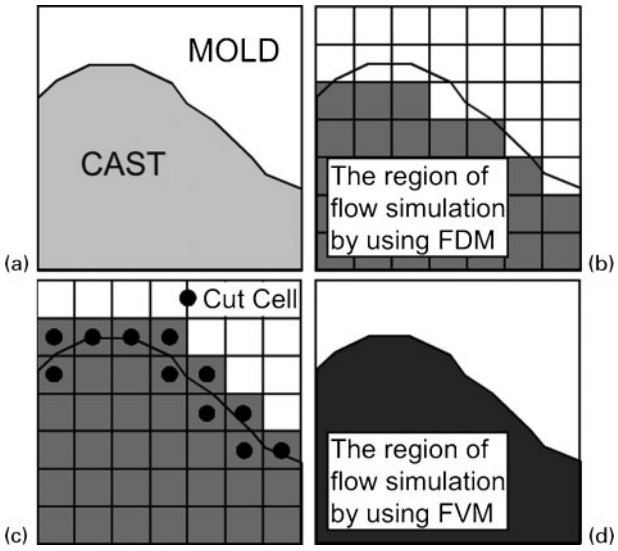
For a long time, a lot of studies on heat flow, solidification and heat treatment have been performed using the finite difference method (FDM) and thermal stress simulation using the finite element method (FEM).[1,2] These study results have also been applied to various casting processes and products. Pre-processors that generate the meshes required for simulation using FDM and FEM have been developed and made in order to perform simulations easily in the casting industry. The FDM and FEM have their own advantages and disadvantages. The FDM can generate meshes fast and easily but cannot display an original product shape due to Cartesian grids. For this reason, the meshes must be generated to have a large number of meshes for accurate simulation, which causes difficulties in carrying out an accurate simulation of products with complicated shapes. Contrary to FDM, FEM can analyse products with complicated shapes using a small number of mesh data but requires a lot of time to generate meshes exactly. Sometimes, FEM is so difficult to use that professional skills are needed for mesh generation. To solve these problems, the authors have been studying the cut cell method (CCM), one of the finite volume methods (FVMs).[3,4] More accurate simulation results can be obtained using CCM because of using surface shape information like area, volume from voxel and a Cartesian mesh factor handled by FDM. In detail, the original voxels are used for the inner elements of a product, and the shape information on voxels cut by the surface of the product is obtained for boundaries where the product surface meets voxels. Using such product surface shape information, problems such as slow down due to momentum loss caused by stair step shape of curves can be solved. This method makes it possible to consider accurate flow patterns. Above all, it requires a user friendly mesh generator (pre-processor) to use this method. The authors have developed a pre-processor that automatically generates meshes in an easy way. The general process of computation using this mesh generator is presented in Fig. 1. When the cast and mould models are input into the system, as in Fig. 1*a*, Cartesian grids are generated, as shown in Fig. 1*b*. Then, the voxels that meet with the surface of the product (which is called as cut cells) are detected, as shown in Fig. 1*c*. In addition, the volume and areas of cast at cut cells are computed as shown in Fig. 1*d*.

Simulation is performed using information on the computed volume and area of cut cells. In addition, in carrying out the computation above, a new surface of the products can be composed with the information on cut cells. Based on such new surface information, the simulation results can be displayed to look as similar as possible to the original model shape. In other words, this method overcomes the disadvantage of the existing methods that cannot display flow and solidification simulation results in a way to look similar to the original product shape because they use voxel shape. A faster and more realistic post-processing technology has been developed using the new algorithms. Such post-processing technology is named 'actual shape mapping' (ASM). The ASM can be applied to not only the results of FVM but also the results of FDM. How to compute the information required for the FVM simulation mentioned above and how to handle the ASM will be briefly explained. Cases of application of this technology to actual products will also be examined in the next chapter.

[1]Korea Advanced Institute of Science and Technology, Daejeon, Korea
[2]Korea Institute of Industrial Technology, Incheon, Korea

*Corresponding author, email chlim@kitech.re.kr

Received 3 June 2010; accepted 27 September 2010
DOI 10.1179/136404611X13001922708911



a model input; *b* voxel generation; *c* find cut cells; *d* region of simulation

1 Process of FVM mesh generation

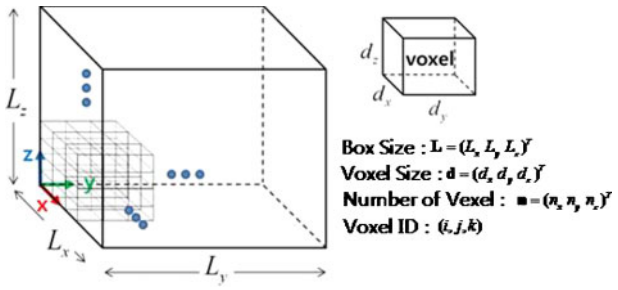
Pre-processing

A new method to construct a volume and to clip volumetric cells is proposed. First, three-dimensional (3D) model stereolithography (STL) files are used in the pre-processor system, and they should be complete two-dimensional (2D) manifolds. Totally separated multiple STLs are allowed, but if there are models intersecting each other, they need to be unified before applying the proposed method. In addition, 3D model files are not needed to have topological information.

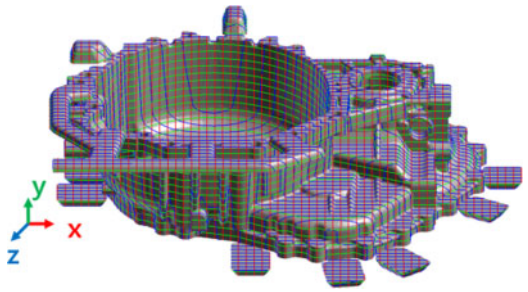
Next, uniform grids containing the input model need to be generated. Figure 2 shows the volume structure.

The final results of the pre-processing are volume and area values taken by input STL at each cell. The pre-processing step is composed of two sequential steps: voxel type classification and clipping. In case of dealing with uniform grids, it is known that the scan conversion algorithm is one of the most rapid methods to classify the type of each voxel in the computer graphics area. Especially, 3D scan conversion[5] is often used if the input model is a piecewise linear surface, such as a triangular. Doing in this work, the authors can construct volume and clipped surface geometries of cells.

For various purposes, the authors create 2D section curves between the input surfaces and the axis aligned planes along the x, y and z directions, which are passing the faces of each cell. The intervals of section planes are the same as the size of a voxel. When doing this work, it



2 Generate uniform grid



3 Example of section curves corresponding to *x*, *y* and *z* directions

is time consuming to query intersecting triangles at each section plane, so the authors use a triangle based computation, which finds intersecting planes at each triangle. This approach reduces the computation time dramatically as compared to the plane based search, and the results of computation are stored at each plane.

Since this study only deals with a complete 2D manifold model composed of triangles, the intersecting curves (in fact, polygons) are always closed. There are several separated closed loops per one section plane, and the edges of the polygons along the counterclockwise direction information are saved.

Classification of voxels

As explained above, the volume cells (voxels) need to be classified, and all the voxels are partitioned into the following three sets in the problem:

(i) cut cell: the voxel intersecting with solid boundary
(ii) inner cell: the voxel encompassed by the input solid
(iii) outer cell: the voxel outside the solid.

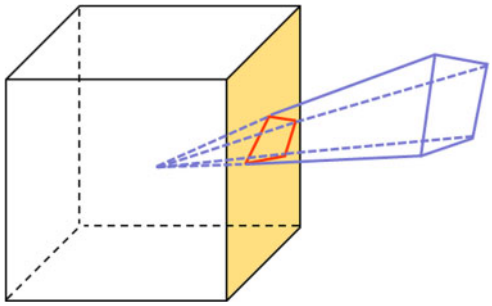
Solid means the region from the STL models that is computed using FVM. Since the volume of the in-cell is always $d_x d_y d_z$ in Fig. 2 and the value of the outer cell is always 0, the exact volume values of the cut cells are calculated by computing the polyhedral shape of the cell. After finding the cut cells, the flood–fill algorithm can be applied to classify inner/outer cells, which are fast and exact when the boundary is closed.

Two-dimensional scan conversion

To classify all the cut cells without missing one, the section curves computed at the previous step were used. Since the curves are computed as 2D, this study deals with a 2D scan converting algorithm, which takes less computation time than the 3D one. To find all the cells passed by section curves as in Fig. 3, the well known Bresenham's algorithm is used.[6] While tracing curves, the index of the edge to the passing cells is saved for the later clipping step. Since the section curves were generated with the same step size of voxels, two neighbour cells always share the same part of the curve. To remove duplicated cells, only one size is stored between the two cells, and the memory of the system is saved.

Additional works

There is a problematic case when applying 2D scanning rather than 3D scan conversion or other methods. When a cell contains a sharp vertex, as shown in Fig. 4, this cell may not be found as a cut cell when the cell stores the edges only at the three-side faces.



4 Sharp vertex: cell including sharp vertex may not be found only with section curves

This problem can be easily solved by additionally querying cells containing the vertices in the input model (note that the cells containing vertices of 3D triangles do not match the vertices of the section curves).

Finally, as described above, the remaining cells are classified as in/out cell groups since the outer cells are not needed. The flood–fill algorithm is to search same type cells by propagating neighbours from a seed cell until reaching the boundary.

For the automatic work, first, all the remaining cells are set as in cell type. Then, since it was assumed that the volume encompasses the model entirely, the corner cells of the volume should be always out cells, and the out cells are extracted by propagating from an arbitrary corner cell.

Clipping cut cells

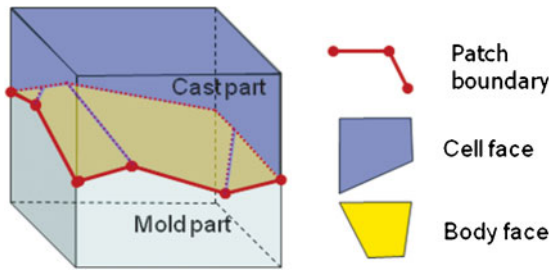
To calculate accurate volume values at each cell, it is important to compute the exact shape of the cut cell. In addition, they need to be saved efficiently in order to compose the rapid post-processing step. This process is called clipping step, and this approach is described in the following sections.

Definitions of cut cell geometries

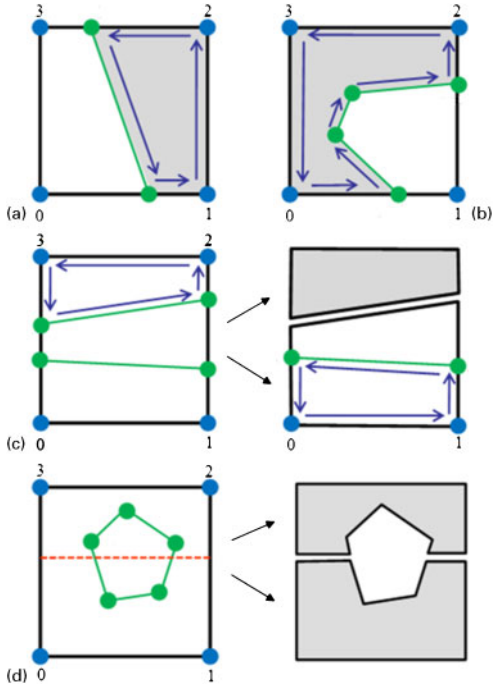
Since voxel and triangles are all of convex shape, the cut face of a cell by one triangle forms a convex polygon, and more than one among six boundary faces are intersecting with the section curves in a cut cell. The process to clip triangles intersecting with a cell cut cell is called as 3D clipping. Before describing the detailed way of clipping, several terms are defined about the cell geometries, as in Fig. 5.

Definition 1

Let the new cut face be created by intersecting triangles between voxel and solid boundary in a cut cell body face (Bf). This face always forms a convex shape, and there exist h body faces in a cell if h triangles intersect with this.



5 Geometries of cut cell



a **type 1: one edge crosses face;** ***b*** **type 2: several connected chains pass face;** ***c*** **type 3: face is separated by edges;** ***d*** **type 4: polygon is included in face**

6 Two-dimensional clipping by intersection types

Definition 2

When a section curve is passing by one of the boundary faces of a cut cell, the face is separated as more than two polygons. Two-dimensional clipping is used to compute these cut polygons. The face obtained from 2D clipping is defined as cell face (Cf).

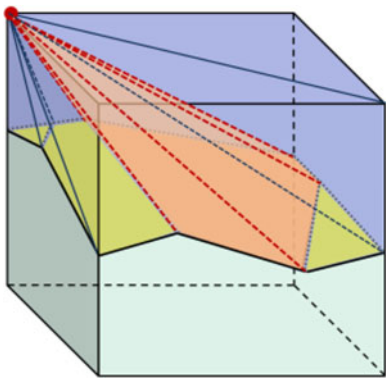
Definition 3

Patch boundary is the set of all the intersecting edges between body faces and cell faces in a cut cell. Usually, it is composed of one closed loop, but it may contain several closed loops.

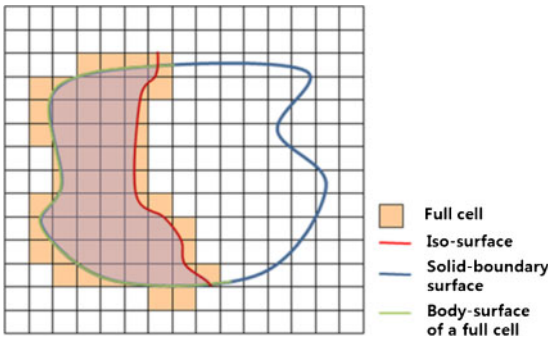
Clipping cell faces

Cell faces can be obtained by 2D polygon rectangle clipping, which is well known in the computer graphics area. Especially, the Weiler–Atherton algorithm[2] is basically used, which can deal with the arbitrary shaped polygon.

However, the clipping types are classified by the more rapid computation, as shown in Fig. 6*a–c*. Additionally, the degenerate case, such as a hole shown in Fig. 6*d*, is



7 Volume computation by partitioning into pyramids



8 Surface extraction

converted to two separated simple polygons since all the faces are stored as polygonal chains for fast computing.

For the face that did not pass by section curves in the cut cell, it is stored if it is inside the solid and discarded in the other case.

Clipping body faces

When this study stored section curves at the previous step, it also saved the information from which triangle to use at this stage. To obtain the area clipped by a cut cell, first, all the edges coming from the same triangle in a patch boundary is found. Then, the same Weiler–Atherton algorithm is applied to obtain the result of polygon–triangle clipping. The orientation of body faces can be aligned with the normal vectors of the original input triangles, and these faces are stored for the computation of volume and the post-processing step.

Final grid generation

After calculating the geometries of all the cut cells, the topological information is checked using the stored faces at each cell. Finally, the grid information is generated, such as the volumes of cells and the areas of faces required in the FVM simulation. The volume of a cut cell can be computed by partitioning the polyhedron into pyramids, and then the summation of signed volume values becomes the volume of the polyhedron, as shown in Fig. 7. The sign of a cone is (+) when the top point is in the opposite direction with the normal vector of the base plane and (−) in the opposite case.

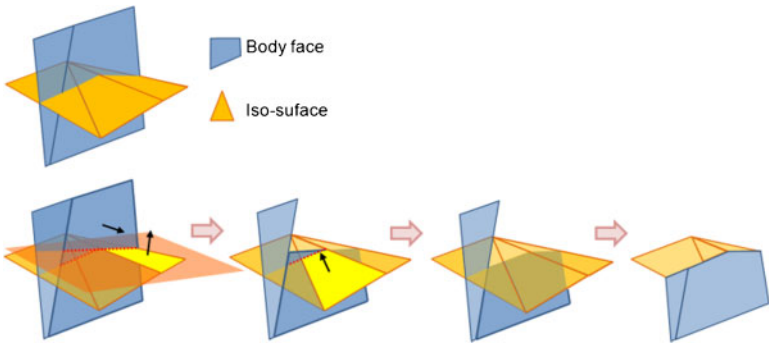
Post-processing

The post-processing step is to check the result of the flow simulation by displaying the process of filling with time. This is composed of three substeps, and the first step is to extract the iso-surfaces (free surface of flow) passing a certain scalar value in the scalar field of filling values. In this case, there is a well known way to extract the iso-surface in the volume graphics area, i.e. the marching cubes algorithm,[7] and this is also used in this work. Since the marching cubes algorithm requires a node based scalar field, the values at volume nodes are precomputed by taking the average from eight neighbour cells.

The second step is to find the part of solid boundary surfaces where the filling fluid meets. This becomes trivial work using the body surfaces computed and stored with each cell at the pre-processing. The fully charged cells can be easily found by checking the filling value given to the cell. Figure 8 shows two different types of surfaces.

After extracting two kinds of surfaces, finally, the instance when the iso-surfaces and the body faces meet in one cell needs to be clipped, and it is the most time consuming work in the post-processing. The computation time for post-processing is more important in order for user convenience.

There are iso-surfaces and body surfaces in the intersection, i.e. the cut cells among the cells found by the iso-surfacing step are the candidates of clipping. At each candidate cell, the polygon–triangle intersection test is first applied to check whether this cell requires clipping. If so, to clip a body face, one representative triangle is chosen among the iso-surfaces, which has the longest intersecting line with it.



9 Clipping process at post-processing stage

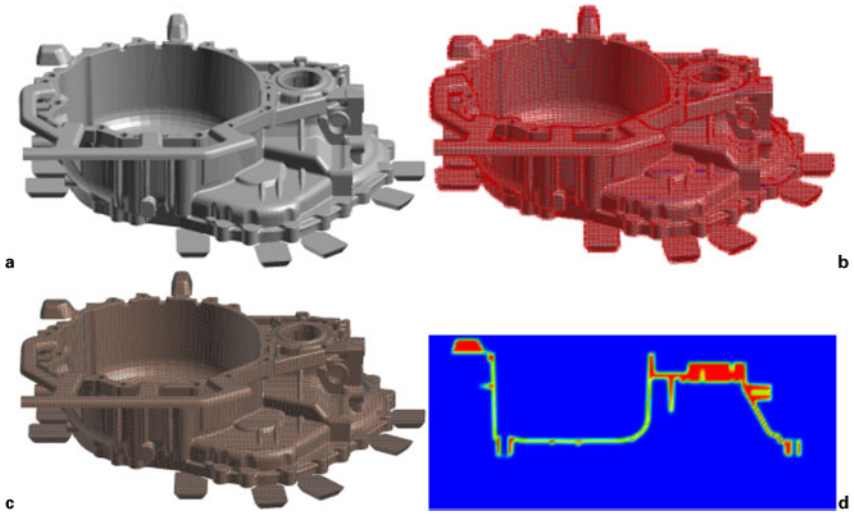
coordinate : -16.969627 -16.980819 -16.967062

Number of Voxel information : 202 107 75

Size of Voxel : 16.801680 16.812725 16.799122 Number of Cast Voxel : 50912

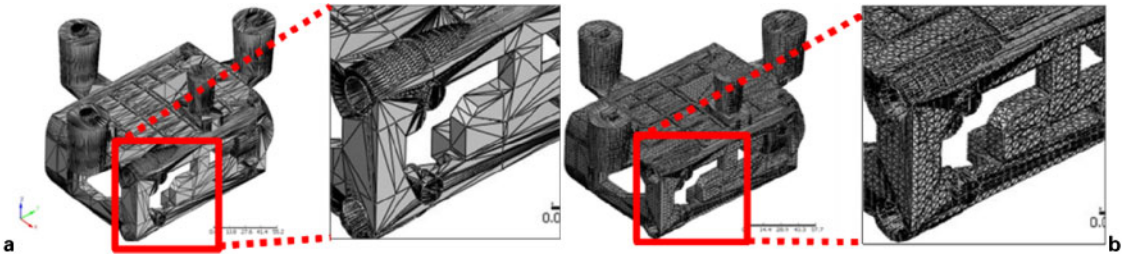
Voxel Number			STL info		Area Info					Volume Info
118	81	14	15	0.000000	0.000000	0.914520	0.919323	0.614168	0.000000	0.297258
102	97	14	15	0.373288	0.209466	0.059961	0.560650	0.766239	0.977083	0.455222
118	82	14	15	0.568088	0.614168	0.001435	0.455250	0.455445	1.000000	0.541703
117	82	14	15	0.441764	0.999997	0.133634	0.568088	0.554884	0.999999	0.524252
102	98	14	15	1.000000	0.765239	0.827197	0.693701	1.000000	1.000000	0.929149
81	81	23	1	0.319047	0.955134	0.467792	0.555281	0.000000	0.000000	0.451854
80	81	18	15	0.037492	0.999995	0.351400	0.489650	0.000000	0.183794	0.222647
80	81	17	15	0.075189	0.999998	0.544022	0.972881	0.058853	0.351400	0.532225
79	81	17	15	0.000000	0.132618	0.012226	0.075189	0.000000	0.002403	0.004922
139	18	36	14	0.000000	1.000000	0.357810	0.000000	0.043230	0.541168	0.134594
139	18	35	14	0.000000	1.000000	0.214222	0.000000	0.000000	0.357810	0.078543

10 Several information for FVM simulation



a input STL; *b* voxel generation; *c* result of clipping; *d* one of volume graph

11 Result of FVM mesh generation through pre-processing



a under 140 total voxels; *b* under 129 500 total voxels

12 Reconstructed surface triangle shape by different total voxel numbers

The intersection test can be computed rapidly by way of the method proposed by Möller.[8] Then, the positive part of the plane equation is clipped including the triangle. If the resulting polygon still intersects with the other iso-surfaces, the face is clipped in the same way until it does not intersect with any iso-surface. After applying this way to all the body surfaces, the iso-surfaces can be clipped with the same way. Figure 9 shows the process of clipping at the post-processing step. To do this work, the intersection test time is saved by letting the cell store its faces.

Application results

As shown in Fig. 10, the information on volume and areas necessary for FVM simulation can be obtained with the method suggested above.

As in Fig. 11*a*, the product model file is input. If multiple model files are input, union is carried out. Then, the voxels are generated as in Fig. 11*b*.

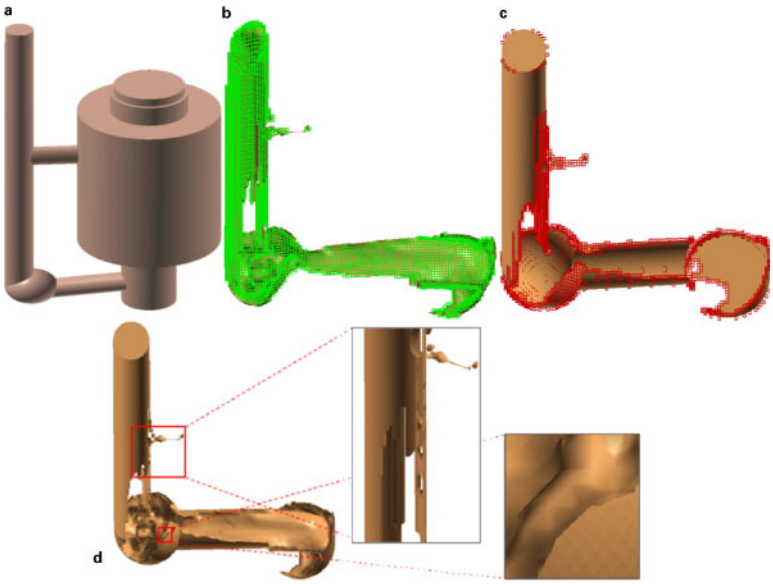
The boundaries between the product surface and the voxels are clipped in order to obtain information that is required for the analysis, as shown in Fig. 11*c*. Figure 11*d* shows the reconstructed configuration through TechPlot using the obtained volume information. As shown in Fig. 11*d*, it can be confirmed that volume information is accurately computed.

Figure 12 shows the reconstructed surface after clipping by the given voxels. For the same model, Fig. 12 shows the results of clipping when there are 140 and 129 500 voxels respectively. As clearly shown in the figures, as there are more voxels, the surface is reconstructed with more dense triangles.

The time that the pre-processor needs depends on the complexity of the model configuration, the number of voxels and the hardware capacity. Table 1 shows the calculation time required for mesh generation needed for the CCM for the previous model in Fig. 12 using a computer equipped with 2·93 i7 CPU 940, 6·0 GB RAM and NVDIA Getforce 9800 graphic board. It can be confirmed that it takes more time than mesh generation with FDM simulation, but it does not take so long that a user feels inconvenient because automatic computation

Table 1 Calculation time for mesh generation and related information under several total voxel numbers

		No. of voxel			
		140	129 500	3 496 500	16 187 500
Time/ms	Union	4503			
	Mesh generation	860	8937	131 922	540 563
No. of triangle element		53 165	262 634	891 599	1 775 069



a input STL; *b* iso-surface extraction; *c* body face extraction; *d* some result through post-processing

13 Display for realistic flow ASM process

is carried out after a user inputs the initial values. If the stored information on the product surface reconstructed after clipping by the pre-processor is utilised, a more realistic flow can be displayed after post-processing of the flow simulation results, as seen in Fig. 13. If the product STL file is input and flow simulation on gravity casting is performed as in Fig. 13*a*, then flow simulation results based on voxels can be obtained. As shown in Fig. 13*b*, free surface of flow is detected from the results of flow simulation, and the iso-surface can be extracted. As in Fig. 13*c*, the body surface that is intersecting between the iso-surface and the product STL surface can be extracted. If the boundaries between these two surfaces are clipped and displayed, results similar to the actual flow can be obtained in Fig. 13*d*.

Like the pre-processing procedure, the computation time necessary for post-processing depends on the complexity of the model shape, the number of voxels and the hardware capacity. The construction time for ASM post-processing computation seen in Fig. 13*d* is about under 20 ms using the same computer system above. Therefore, flow simulation results can be handled directly without inconvenience by delay. In addition, this ASM technology can be used without limits to handle FDM simulation results if the results of product surface reconstruction can be computed by the pre-processor.

Conclusions

In this study, we have developed a user friendly pre-processing system to use the CCM, one of the FVMs. The ASM technology has also been developed, which is a post-processing method where the actual flow and the original shape of a product can be displayed. The authors are studying the method to make these technologies useful in the industry field now. The authors believe that these technologies will maximise productivity in the casting manufacturing field. The authors plan to continue studying multiblock grid generation and FEM mesh interlocking technology, etc., to enable engineers to easily perform various simulations.

References

1. S. Xun, H. Y. Hwang, *et al.*: 'Numerical simulation of mold filling processes of castings by using of predictor-two step corrector-VOF', *J. KFS*, 2002, **6**, (22), 35–39.
2. B. H. Choi, S. Y. Kwak, *et al.*: 'The optimum design of casting process through prediction and control of thermal deformation', *J. KFS*, 2005, **5**, (25), 31–37.
3. P. G. Tucker and Z. Pan: 'A Cartesian cut cell method for incompressible viscous flow', *Appl. Math. Modell.*, 2000, **24**, 591–606.
4. Y. S. Choi, *et al.*: 'Study cut-cell method for casting process simulation', Proc. 7th Pacific Rim Int. Conf. on 'Modeling of casting and solidification processes', Dalian, China, August 2007, Dalian University of Technology, 371–376.
5. A. Kaufman: 'Efficient algorithms for scan-converting 3D polygons', *Comput. Graph.*, 1998, **12**, (2), 213–219.
6. J. D. Foley, A. van Dam, S. K. Feiner and H. F. Hughes: 'Computer graphics: principles and practice'; 1996, Reading, MA, Addison-Wesley.
7. W. E. Lorensen and H. E. Clien: 'Marching cubes: a high resolution 3D surface construction algorithm', Proc. Int. Conf. on 'Computer graphics and interactive technique', Anaheim, CA, USA, July 1987, ACM Press, 163–169.
8. T. Möller: 'A fast triangle-triangle intersection test', *J. Graph. Tools*, 1997, **2**, (2), 25–30.