BACKGROUND OVERLAY GRID SIZE FUNCTIONS

Jin Zhu, Ted Blacker, Rich Smith

500 Davis Street, Evanston, IL, U.S.A. jz@fluent.com

ABSTRACT

This paper describes a new technology of mesh size control using a background overlay grid size function. A background overlay grid is generated first according to the defined size functions and then is used as the base grid for determining the mesh size at each point during the meshing process. The definitions, classifications, implementations and control algorithms of three types of size functions including a fixed size function, a curvature size function and a proximity size function are presented in detail. Meshing results with controlled mesh sizes are given, and considerations for further improvement are listed.

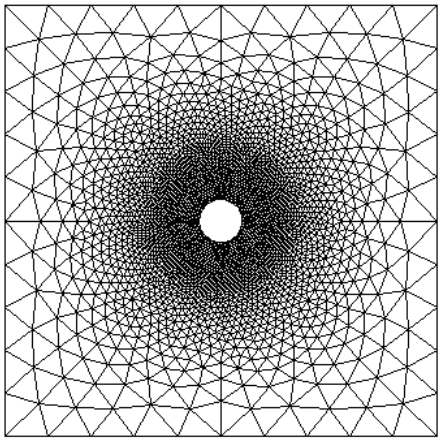
Keywords: mesh generation, size control, background grid, size functions.

1. INTRODUCTION

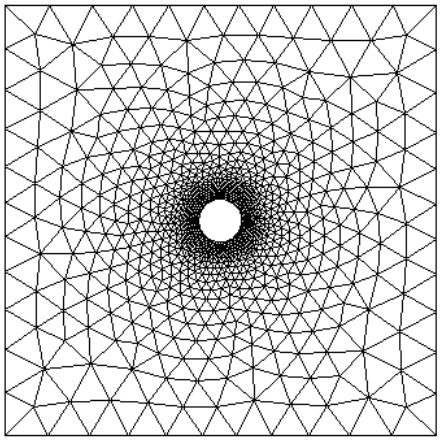
In the mesh generation field, the mesh size control is very critical to mesh quality and to the successful field simulations using the generated mesh. The mesh sizes need to catch local details in areas of the geometry where small features exist. On the other hand, in non-critical areas of the geometry, the mesh size can be large as long as the mesh transition is smooth enough. However, it is tedious to manually determine the local features of the geometry and mesh these entities by desired sizes. Premeshing boundaries of the domain with the desired size is a standard way of obtaining size transition and gradation. However, the user has no direct control over the mesh grading on the geometry. Local refinement of an existing mesh is another option. Unfortunately, unrefined meshes will form a fixed constraint to the refined areas and results are not always satisfactory.

As a simple example of the importance of size functions, consider Figure 1. The geometry is a 10x10 square with a circular hole of radius 0.5. In Figure 1(a) the inner circle is pre-meshed with a size of 0.05 and rest of the face is meshed with a size of 1.0 by an advancing front triangle meshing algorithm. Transitions are handled by the algorithm itself, with no reliance on a size function. A total of 5,656 elements are generated. In (b), the same algorithm is used, but a size function is prescribed which guides the meshing. The size function used prescribes a size of 0.05 at the hole boundary and a geometric growth rate of 1.2 based on the distance from the hole. This growth is limited by a maximum size specified as 1.0. There are only 1,950 elements generated in this case. You can see that the mesh gradation is well controlled by the growth rate in (b) compared with the mesh pattern in (a). When meshing the same face with a quadrilateral/paving algorithm [1], no mesh could be obtained without a size function because of the extreme gradation difference and the lack of interior gradation control. With the size function it can be meshed nicely, as shown in Figure 1(c).

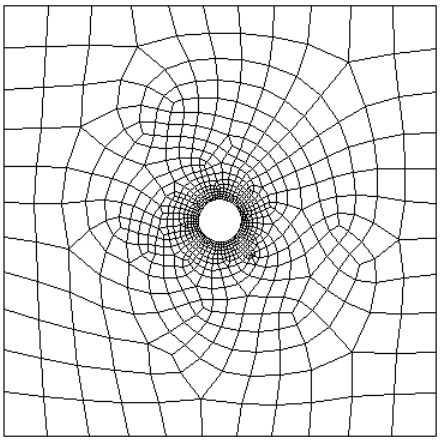
During the meshing process it is highly desirable that some guidance be provided to the mesh tools to specify the size of elements to be defined and the variation of size from one part of the domain to another. Sizing and gradation control can be determined during the meshing process or more commonly as an a priori procedure. As an a priori procedure, a size function is defined over the entire domain. The sizing function, d=f (x), where d is the target element size and x is the location in the domain, can be customized for specific geometric or physical prosperities. The sizing function may take into account surface features as well as physical



(a) Circle pre-meshed with size 0.05, face meshed with size 1.0



(b) Face directly meshed with fixed size function.



(c) Face meshed with quads

Figure 1. Comparison of meshing results from: (a) pre-meshed inner circle with no growth control and (b) from size function with growth controlled

properties in determining local element sizes. Such surface features as proximity to other surfaces and/or surface curvature can be used to control surface mesh density distribution. Physical properties such as boundary layers, surface loads or error norms from a previous solution may be considered. For instance, in an adaptive finite element scheme, a size specification in the simulated field is deduced from simulation results, usually via an error estimate. This may then be combined with face geometric constraints being considered. The size specification is then normalized by metrics and this metric map that defines a control space is used to control the mesh gradation [2].

Many authors have described the use of some form of element size control in the literature for a specific meshing algorithm. Based on the spatial decomposition approach for meshing purposes as pioneered about two decades ago by Yerry, Shepard [3] and surveyed by Thacker [4] and Shepard [5], a size-governed quadtree triangle mesh generation method was presented by Frey and Marechal [6] to deal with planar domains of arbitrary shape. The domain is first decomposed into a set of cells. The size of the these tree cells are adjusted to match the element sizes at boundaries of the domain prescribed by a given size map, and the mesh gradation is controlled by the level of refinement of the cells using the [2:1] rule. Therefore, these cells have a size distribution compatible with the desired mesh gradation and so can provide a convenient control space which can be used to determine the element size. Secondly, the quadrants are triangulated accordingly to get full triangle elements. Finally the triangles are optimized (i.e. smoothed).

Currently, a background mesh appears to be the most commonly used means of defining an element sizing function. In the background mesh method, collections of vertices containing the sizing information are first selected. Then Delaunay triangulation is performed with them, inserting additional interior nodes. Finally the meshing tool retrieves a target size at any location within the domain by linear interpolating in a certain background triangle (for 2D) or tetrahedra (for 3D).

Shahyar Pirzadeh [7] introduced an approach that adopted uniform Cartesian grid and the elliptic grid point distribution for generation of 2D unstructured mesh using the advancing front technique. It was analogous to solving a steady-state heat conduction problem with discrete heat sources. The spacing parameters of grid points were distributed over the nodes of the Cartesian background grid by solving a Poisson equation. To increase the control over the grid point distribution, a directional clustering approach was also implemented. However, there will be some mathematical difficulties when it is used for general 3D problems and/or with non-nodal and non-linear sources.

More recently, Owen and Saigai [8] presented the method of controlling element size on parametric surfaces, taking into account boundary layers, surface curvature and anisotropy, and using natural neighborhood interpolation. Related works using background mesh can be found in [9 - 11].

Although the algorithms discussed above are effective and useful in many aspects, neither gives a general and versatile way of size control for all kinds of geometry and all types of element. The goal of this work was to create a general way of defining mesh size for all element types and for different kinds of geometric features. The size function had to provide very rapid evaluators that would be general for any meshing algorithm. Also, local geometric effects had to be able to radiate, or influence size on a more non-local area. For example, tight curvature on one surface should affect other edges/surfaces in close proximity to ensure a controlled transition rate. This paper describes how these objectives were met using a background overlay grid. The work will be described by first defining the size functions provided to the user and the size function initializations. Then details of the use of a background grid are documented and examples of its use given.

2. DEFINITIONS OF SIZE FUNCTIONS

2.1 Terminology

Our size functions are based on a distance controlled radiation. To understand this definition, the parameters that are common to all size function must be defined.

- Source entities: Source entities are a set of geometric entities on which the mesh sizes are specified and from which the mesh size is grown into affected areas. Source entities can be any general geometric type including vertex, edge, face or volume.
- Attached entities: the geometric entities on which the size functions will have influence as the entities are meshed. These include edge, face or volume. The attached entity can be the same entity as the source.

  When a size function is attached to an upper topology, all lower topologies of the attached entity will be influenced. When a size function is attached to a lower topology, its upper topologies will not be affected.
- Growth rate: This parameter controls the geometric pace with which the mesh size progresses from the source. It is based on the distance of the point being evaluated from the source.

  In cases where the elements of significantly different sizes are immediately adjacent to each other, both the meshing tool and the simulation tool cannot perform well. In order to maintain a desired growth ratio, the target size is adaptively adjusted by applying a geometric growth formula. This parameter specifies the rate of this geometric progression.
- Distance limit: This variable specifies the range in which the size function is valid. It is the distance for the source mesh size to grow up to the size limit, but it is not user controlled.
- Size limit: This is the maximum mesh size. When the grown size at the given location exceeds the size limit, this limit is used instead. Therefore, if the distance from a given point to a source is larger than the distance limit, we do not need to test the grown size and the size limit is directly used.

Figure 2 demonstrates a single radiating size function. One edge of a cube (upper right edge) is used as the source entity. The spherically tipped cylinder, whose axis is the source edge and whose radius is the distance limit, indicates an iso-surface of the prescribed size function within which the size function is valid. In the remaining areas of the cube that are outside the cylinder shown, the size radiating growth has no effect and size limit is used instead.

2.2 Definition

Based on practical applications and experience, the following size functions have been provided in our algorithms:

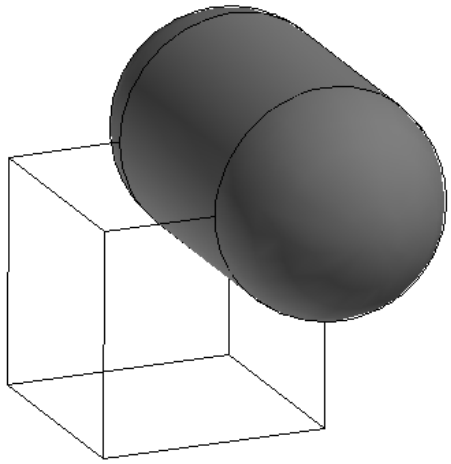


Figure 2. Demonstration of the effective domain of size function

2.1.1 Fixed size function

For a fixed size function, the mesh size on the source entity is a constant value.

To define a fixed size function, all the parameters introduced in section 2.1 are used, along with a "start size" specifying the constant size of the mesh on the source entities.

2.2.2 Curvature size function

A curvature based size function specifies the mesh sizes on the source entities relative to the degree of surface curvature, i.e. finer mesh sizes in highly curved regions and coarser mesh sizes in regions of low curvature. Curvature based size functions can only use faces as the source entity. This feature provides a convenient means of controlling the geometric approximation of the mesh elements. This varying size on the surface can then radiate outward at the specified growth rate.

To define a curvature size function, all the parameters introduced in section 2.1 are used, along with an angle. This angle specifies the maximum angle between adjacent facets of geometric faces. Using angle as an input specification makes the curvature size function purely dependent on the curvature and independent of the size of the model. For example, a big sphere and a small sphere will have the same number of elements generated if they have the same curvature size function angle.

2.2.3 Proximity size function

A proximity size function controls density based on geometric closeness of entities. The mesh sizes on source entities are determined by the gap between faces (3D) or edges (2D) of the source entities and the required number of elements in the gap.

The first additional parameter for a proximity size function is "cells per gap" which specifies the minimum number of elements that should be put in the gap between any two closest opposing faces (volumetric mesh) or two opposing edges (surface mesh). Source entities for a proximity size function can be faces or volumes. When a volume is used as the source, all faces of the volume become source faces. The proximity check for all source faces includes a check of the proximity of edges on the face.

3. SIZE FUNCTION INITIALIZATION

In preparation for the generation of the background grids, the three types of size function definitions must be initialized differently. This initialization establishes the desired sizes everywhere on the sources.

3.1 Fixed size function

All the source entities have a constant mesh size on the entity, and no special initialization is needed.

3.2 Curvature size function

Initialization of curvature size functions requires the generation of a faceted representation of the face that meets the curvature requirements.

The facets on the source faces are created according to the specified maximum normal angle deviation. This means that the angle of rotation of the normal vectors of any two adjacent facets on a common edge does not exceed the specified maximum angle. This insures that the curvature of the face can be accurately captured. The mesh size, S_n, at a node n of a facet of the source face can be computed by:

$$S_n = 2 * \sin(\theta_{\max} / 2) / \rho_{\max}$$

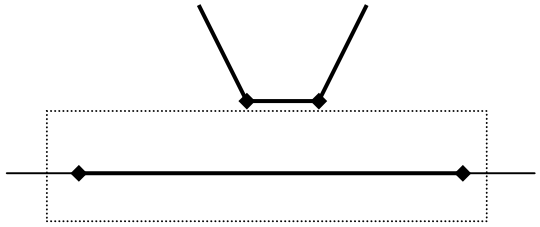
Here $\rho_{\max}$ is the larger curvature along two orthogonal axes. If the computed size is larger than the size limit, or if a face is flat (i.e. no curvature), then the specified size limit is used.

It is possible for the local size, S_n, to be larger than the radiation from a nearby node would permit. Thus, if the radiated size of node m at node n, S_{mn}, is less than S_n, the radiated size S_{mn} is used.

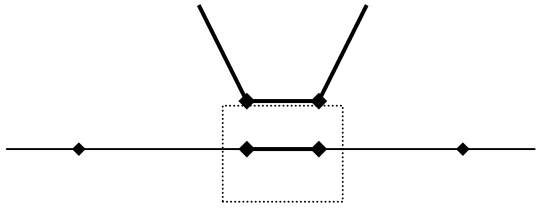
3.3 Proximity size function

For initialization of the proximity size function, we also need to create facets for the faces. First, a set of coarse facets is created for each face according to the maximum normal angle specified for the proximity facet. Next, the distance of each facet center to another "visible" facet is calculated and stored with the facet. Because these coarse facets are often long and slender, such facet-based calculations may over-extend the influence of gaps (see Figure 3a). To avoid this undesirable effect, these coarse facets are further refined as follows.

Within some maximum level of facet subdivisions, if the maximum edge length of the current facet is larger than several times its distance (or gap) to the target facet, then the current facet will be split on the longest edge into two smaller ones. The sub-facets will then be compared with the target facet and iteratively refined as needed. Finally this gap value is stored in each facet. This process is optimized by computing the distance between the bounding box of current facet to the bounding boxes of other target facets and comparing the distance to the stored minimum distance. If the computed distance is beyond the stored minimum range of current facet, remaining calculations will be skipped. This significantly reduces the amount of distance calculations needed. Fig. 3b demonstrates that the refined facets localize the gap influence.



(a) Unrefined gap influence



(b) Refined gap influence

Figure 3. Refinement of proximity facets

Also, since we are often only concerned about the gap within volumes, if two facets are from faces that belong to the same volume, we can make use of the relation of the facet normal vectors to avoid unnecessary comparisons. If their normal vectors, whose positive directions are defined as pointing outward the volume, are pointing toward each other, which means there is void space in-between, we can omit the proximity check. Even if no volume is provided as the source entity, we have to check whether the given source faces belong to the same volume. If they do, internally we still establish the volume pointer and compare the relation of face facets normals in order to speed up the initialization.

If any face that owns the facet in the facet pair being compared is a dangling face of the volume, the normal of its facets is ambiguous and full calculations are needed.

Beside face proximity check (i.e. 3D proximity), an option of performing edge proximity check (i.e. 2D proximity) is

given. During edge checking, all the edges of a face need to be faceted into line segments. The distance between each pair of opposing edge segments on the same face will be computed and the shortest distance used to determine the mesh size on each line segment. To speed up computation, only the segment pairs within a mutually visible range will be checked. To guarantee accuracy, a minimum number of edge segments are created (50 in our implementation), especially for very short edge loops.

For both the volume and face proximity controls, the mesh size on the source entity is determined by:

$$S_n = d_{gap} / N_{cells}$$

Where d_{gap} is the smallest gap distance associated with a facet, N_{cells} is the number of cells in the gap area.

4. BACKGROUND GRID GENERATION

As a result of the size function initialization, the desired size on all sources is known. The next step is to establish the complete background grid. The background grid provides the radiation mechanism for all the size functions. It also allows multiple size functions to be combined into a global function. The background grid is an axis-aligned octree-based mesh, the size of the bounding box of all attached entities. The background grid values are derived at all cell corners and refined as needed to capture the size function gradients.

4.1 Background grid initialization

Setting up the background grid starts by generating a bounding box of the domain of the attached entities.

To ensure resolution is not excessive, we find the dominant direction which is the longest range of the bounding box and divide it by a given number of lines in each grid (e.g. 3) to obtain a unit length. We ensure all other directions are scaled accordingly by this unit length, but require at least 2 grid lines in each direction. This way, the resulting background grids will be equi-sided cubic cells.

If a group of entities have exactly identical size functions attached, a single united bounding box is used. This can save time in cases where the bounding boxes of individual attachment entities overlap. In a few cases where the attachment entities are far apart from each other, it can increase the time instead. However, an increase in speed is noted for almost all practical geometries.

4.2 Establishing values at the background grid nodes

Because the size on all sources, S_{ent}, is now known, we can use the same approach to handle all size functions when obtaining the mesh size at the background grid nodes. The only difficulty is to identify which source mesh size to use when growing to a given background grid node. This can be determined by projecting the given grid point onto the closest facet.

When growing the size from the closest facet of the source entity to the given corner point of the background cell, the mesh size is successively progressed from the size on the source entity. It is controlled by the defined growth rate, g, and the distance of the point of the background cell to the source. The progression is iterative. During the progress, an incremental distance expands step by step until the desired point is within the region between two neighboring distances. Suppose R is the distance of the given point to the source entity, R_n and R_{n+1} are the two said distances, respectively. Then the condition can be expressed as:

condition-exit: $R_n <= \mathrm{R} <= R_{n+1}$

Let S_n be the mesh size at the previous distance and S_{n+1} be the mesh size at the subsequent distance. The initial values for these variables are

$$S_n = S_{n+1} = R_{n+1} = S_{ent}$$

$$R_n = 0$$

The following loop, once completed, will give the size and radius of the two distances bounding the given point:

while (!condition-exit) {

$S_n = S_{n+1}$;

$R_n = R_{n+1}$;

$S_{n+1} = S_n * g$;

$R_{n+1} = R_n + S_{n+1}$;

}

A linear interpolation between the two bounding distances is accomplished by this equation

$$\gamma = (\mathrm{R} - R_n) / (R_{n+1} - R_n)$$

Here ($0 <= \gamma <= 1$). The actual size, S_p, at the given point, P, is computed as:

$$S_p = (1 - \gamma) * S_n + \gamma * S_{n+1}$$

However, the final size is the smaller of the computed size and the defined size limit.

$$S_p = \mathrm{MIN}(S_p, S_{\max})$$

If a corner point is affected by several size functions, the smallest mesh size will be taken for it.

4.3 Linear interpolation

Once the background grids are created, the mesh size at any given point can be found by interpolation in the background grids for any meshing processes. Since the background grids are axis-aligned and well shaped cubes, finding the correct

background grid cell is trivial. Simple linear interpolation can work well to give the mesh size at any point P(x,y,z):

$$S_p = \sum_{i=1}^{8} N_i S_i$$

Where S_i is the mesh size at 8 corners of the cell which the point falls into and N_i is the tri-linear interpolation function for each corner point. Suppose the local coordinates (α, β, γ) of the given point inside the background cell can be expressed as

$$\left.\begin{aligned} \alpha &= (x - x_{\min})/(x_{\max} - x_{\min}) \\ \beta &= (y - y_{\min})/(y_{\max} - y_{\min}) \\ \gamma &= (z - z_{\min})/(z_{\max} - z_{\min}) \end{aligned}\right\}$$

Where $(x_{\min}, y_{\min}, z_{\min})$ and $(x_{\max}, y_{\max}, z_{\max})$ define the range of the cell. Then the mesh size at point P can be expanded as

$$\begin{aligned} S_1 &= (1-\alpha)(1-\beta)(1-\gamma) \bullet S\big|_{x\min, y\min, z\min} \\ S_2 &= \alpha(1-\beta)(1-\gamma) \bullet S\big|_{x\max, y\min, z\min} \\ S_3 &= (1-\alpha)\beta(1-\gamma) \bullet S\big|_{x\min, y\max, z\min} \\ S_4 &= \alpha\beta(1-\gamma) \bullet S\big|_{x\max, y\max, z\min} \\ S_5 &= (1-\alpha)(1-\beta)\gamma \bullet S\big|_{x\min, y\min, z\max} \\ S_6 &= \alpha(1-\beta)\gamma \bullet S\big|_{x\max, y\min, z\max} \\ S_7 &= (1-\alpha)\beta\gamma \bullet S\big|_{x\min, y\max, z\max} \\ S_8 &= \alpha\beta\gamma \bullet S\big|_{x\max, y\max, z\max} \end{aligned}$$

4.4 Background grid refinement

Background grid generation is the most expensive part of the overall process. To speed up this bottleneck, some steps have been taken.

One of the most important considerations is when and how to stop background grid refinement. A criterion has to be set to ensure that an almost linear relation of mesh size has been reached within a cell and so there is no need to refine the cell again, no matter what the actual ratio of mesh sizes within a cell is. This is because the meshing size at a point is computed by linear interpolation in the background grid using mesh sizes at its eight corner points. The linear relation can be tested by comparing the deviation of the linearly interpolated mesh size at the center of a background grid S_{linear} (i.e. averaged size at 8 corner points of a cell) from the defined mesh size S_{def} (i.e. the actual size computed from size functions), and take the relative percent as the error estimate. The background grid will be refined if the relative error, δ, is larger than the specified level of accuracy, Δ_{tol}, which can be expressed by the following equation:

$$\delta = \frac{\left|S_{linear} - S_{def}\right|}{S_{def}} \times 100\% < \Delta_{tol}$$

Where Δ_{tol} is a given error tolerance and is controllable by the user.

This seems a reasonable way of stopping the refining process, but potentially non-linear distributions in other areas of the cell cannot be caught nicely, especially in the earlier stage of the refining process. This will lead to corruption of grid generation, unless a constraint of one level difference is applied for neighboring cells. As shown in Figure 4, if the cell contains source entities whose smallest size, $B_{\min}$, is less than the minimum size at the 8 corner points of the cell, $C_{\min}$, then the centric size, S_{linear}, computed from the 8 corners is not accurate, so another refinement to the cell is triggered.

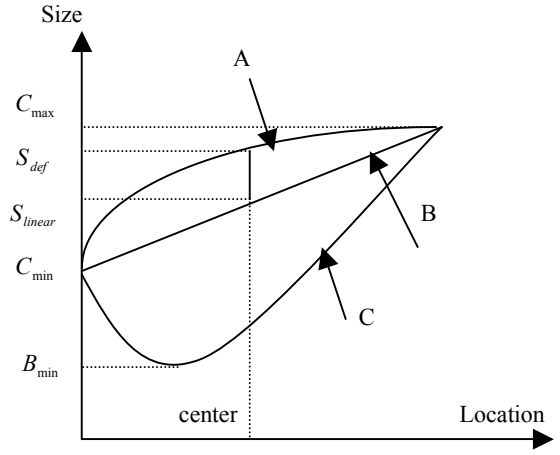


Figure 4. Refining criterion for a background cell (A - actual size distribution from defined size functions, B – size by linear interpolation from 8 corner points, C – source entities possibly with smaller size inside the cell)

In any case, if the maximum range of the bounding box of a background grid cell becomes smaller than the minimum local size in the cell (at eight corner nodes or inside the cell), stop refining the cell to avoid over-refinement.

4.5 Speed/Memory issues

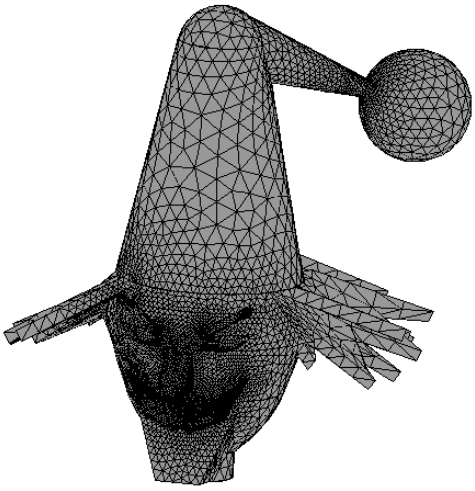
Storage of the mesh size information in the background grids can require a lot of memory. To speed up grid generation, we save the computed mesh size at each corner point of the cell, so that its neighboring cells can directly use the size at shared corner points. This option improves the speed, but sacrifices memory. An option is given to let the user decide whether memory or speed is more important. Thus, they can choose either saving the mesh size at unique grids or re-computing the mesh sizes for each cell.

5. EXAMPLES

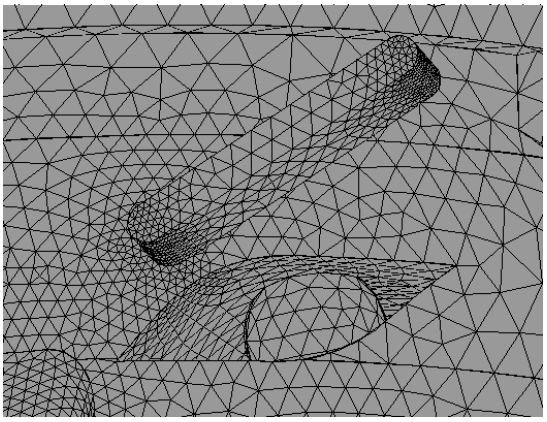
A few examples are given below to show the application of a single type size function or a combination of them in the meshing process. High quality meshes have been generated using the defined size functions with very little effort.

5.1 Meshing the Clown Head

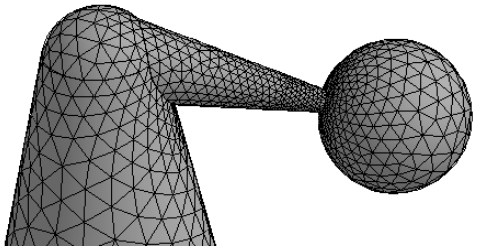
A single curvature size function is defined for meshing the clown head. Normal angle = 20, growth rate = 1.2, size limit = 2, and all faces are used as source and the size function is attached to the whole volume. Figure 5(a) is the meshing results of the whole head. Figure 5(b) and (c) shows the eyeball and hat-tail, respectively. You can see the meshes are nicely transitioned according to the curvature of the surface. Also, the radiation effect of the tight curvature on neighboring surfaces is shown.



(a) Whole head



(b) Eyeball



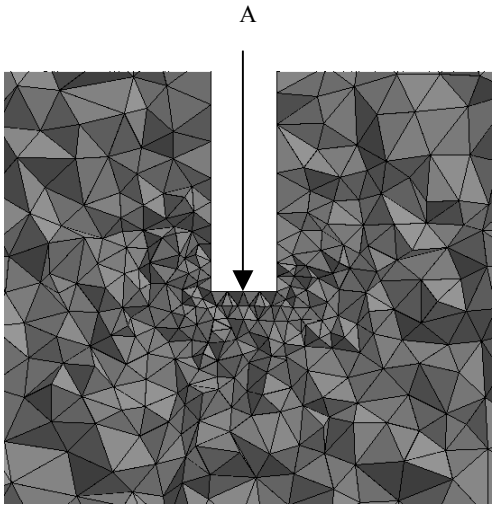
(c) Hat-tail

Fig. 5 Meshing the clown using a single curvature size function

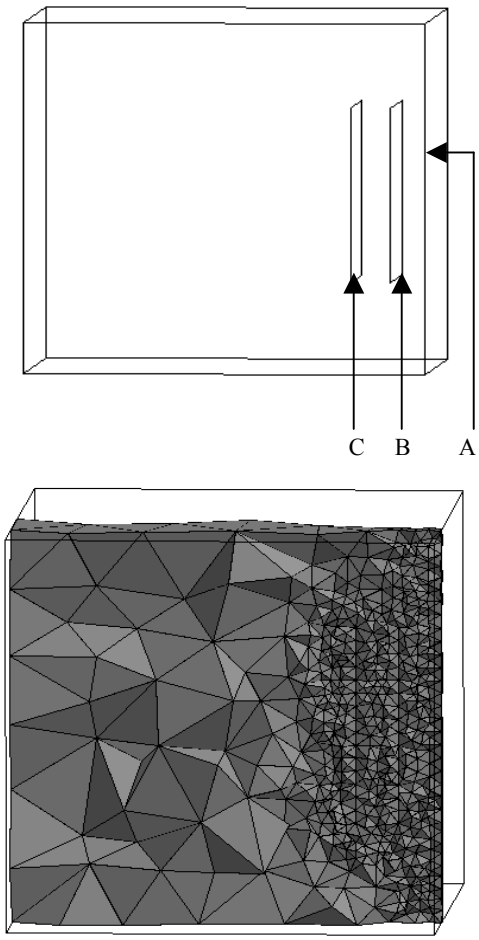
5.2 Use of 2D and 3D Proximity size functions

Figure 6 shows how the 2D and 3D proximity size functions work. In Figure 6(a) the proximity size function is defined as follows: cells-per-gap =4, growth-rate =1.2, size-limit = 20 which is big enough so that the mesh size can grow without any restrictions until hitting the boundary. The source face is face A. The sizes are radiating from the source face into the rest of the volume.

Figure 6(b) shows the shape of a volume with two dangling faces (upper) and the meshing results (lower) of a volume 3D proximity size function. Parameters are specified as follows: cells-per-gap = 3, growth-rate = 1.2, size-limit = 2, three faces (one side face A and two interior dangling faces B and C) are used as source entities. The size function is attached to the whole volume.



(a) 2D proximity



(b) 3D proximity including dangling faces: shape of the geometry (upper) and the meshing results (lower)

Figure 6. Use of proximity size functions in volume meshing

5.3 Use of combined size functions

Figure 7 uses a combination of size functions to mesh the volume. A fixed size function (start-size = 0.05, growth-rate = 1.2, size-limit = 0.5, two flat planes A and B as source), a curvature based size function (normal-angle = 10, growth-rate = 1.2, size-limit = 0.5, two circular faces C and D as source) and a proximity size function (cells-per-gap = 3, growth-rate = 1.2, size-limit = 0.5, whole volume as source) are defined and attached to the same volume. In common areas where three size functions are effective, the smallest size among the three size functions is chosen to set the local mesh size.

Our last example in Figure 8 shows another model meshed using combined size functions. Two curved faces are used as source for both the curvature size function (normal-angle =

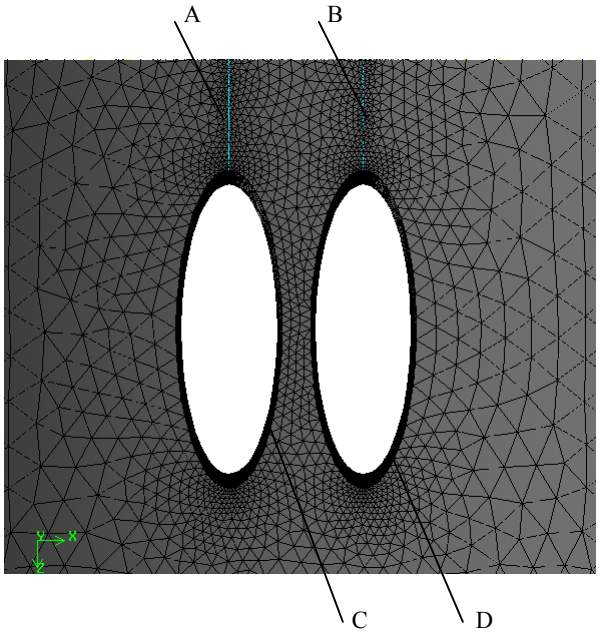
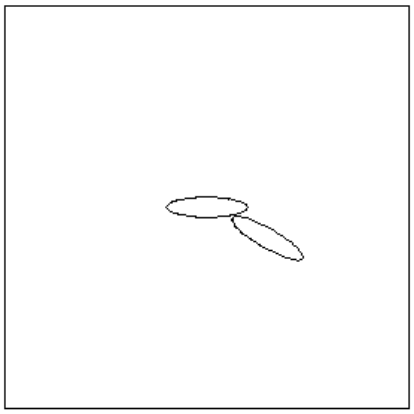
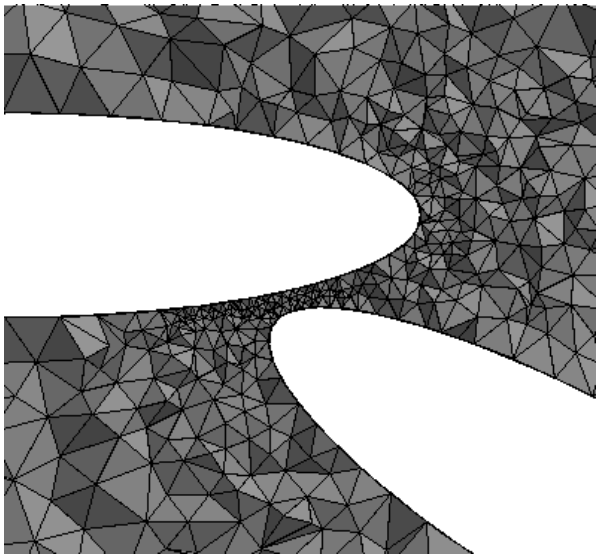


Figure 7. Meshing results using composite size functions. Three kinds of size functions are attached to the volume.

40, growth-rate = 1.2, size-limit = 1.5) and the proximity size function (cells-per-gap = 2 growth-rate = 1.2, size-limit = 2). Figure 8(a) is the outline of the geometry from which you can see the two airfoils that are close to each other. It would be difficult to mesh the areas between them if proximity size functions were not used to specify the number of element in the tiny gap. Figure 8(b) is the enlarged local mesh patterns between the two airfoils after the volume is meshed. The elements are generated exactly as specified and grown nicely from the gap toward neighboring regions. The effect of the curvature size function can also be seen around the wing tips.



(a) Shape of whole geometry



(b) Local detailed elements

Figure 8. Use of proximity and curvature size functions in meshing a volume with airfoil voids

CONCLUSION

A general method of controlling mesh sizes and radiation for all element types and for different kinds of geometric features has been created. The defined size functions have provided rapid evaluators that would be general for any meshing algorithm. Local geometric effects have been radiated to influence size on a more non-local area. The basic algorithms of constructing the background grid and creating fixed, curvature and proximity size functions have been put forward. The criterion of refining the background grid has been shown. Local mesh size at any point in the domain can be interpolated from the pre-determined sizes at corner points of the background cell into which the given point falls. The proposed sizing method has been implemented in Gambit product, and successfully tested on a wide variety of models with excellent results.

In the future, other types of size functions can be added to meet specific user needs. For example, we can add a size function that catches the exterior proximity of the volume if this is desirable. Also, we can add a size function that uses pre-meshed entities as sources and uses the size of the existing mesh on the sources to radiate. For some applications it is also beneficial to have the directional size functions with anisotropic properties.

Speed improvement for background grid refinement is also a focus in the future work.

ACKNOWLEDGEMENTS

The authors express gratitude to Yongheng Shao for his support to the work, and to Young Kyu Lee who has given helpful suggestions in improving the background grid refinement criterion.

REFERENCES

[1] Ted D. Blacker, Michael B. Stepheson, “Paving: A new approach to automated quadrilateral mesh generation”, Int. J. Numer. Methods Eng. Vol 32, pp. 811-847 (1991)

[2] Houman Borouchaki, Frederic Hecht and Pascal Frey, Mesh gradation control, Proceedings of 6th International meshing roundtable. Oct. 13-15, 1997. Park City, Utah, USA.

[3] M.A. Yerry and M.S. Shepard, “A modified-quadtree approach to finite element mesh generation”, IEEE Computer Graphics Appl., Vol 3(1), pp.39-46 (1983)

[4] W. C. Tracker, “A brief review of techniques for generating irregular computational grids”, Int. J. Numer. Methods Eng. Vol 15, pp. 1335-1341 (1980)

[5] M. S. Shepard, Approaches to the automatic generation and control of finite element meshes, Applied Mechanics Reviews, Vol 41, pp. 169-185 (1988)

[6] Pascla J. Frey and Loic Marechal, “Fast adaptive quadtree mesh generation”, Proceedings of 7th International meshing roundtable. Oct. 26-28, 1998. Dearborn, MI. USA.

[7] Shahyar Pirzadeh, “Structured background grids for generation of unstructured grids by advancing-front method”, AIAA Journal. Vol 31(2), pp. 257-265(1993)

[8] Steven Owen and Sunil Saigal, “Surface mesh sizing control”, Int. J. Numer. Meth. Engng. Vol 47, pp. 497-511(2000)

[9] J.Z. Zhu,, O.C. Zienkiewicz,, E. Hinton, J. Wu, “A new approach to the development of automatic quadrilateral mesh generation”, Int. J. Numer. Meth. Engng. Vol 32, pp. 849-866(1991)

[10] S.A. Canann,, Y.C. Liu, A.V. Mobley, “Automatic 3D surface meshing to address today's industrial needs”, Finite Elements in Analysis and Design. Vol 25. 185-198 (1996)

[11] R. Lohner, “Extention and improvements of the advancing front grid generation technique”, Communications in Numer. Method in Engng. Vol 12. pp. 683-702(1996).