A NEW TYPE OF SIZE FUNCTION RESPECTING PREMESHED ENTITIES

Jin Zhu

Fluent, Inc. 1007 Church Street, Evanston, IL, U.S.A. jz@fluent.com

ABSTRACT

This paper describes the creation of a new type of size function – the mesh size function that honors the existing mesh on premeshed geometry entities and radiates the mesh sizes from the premeshed source entities to the attached entities – from the technology of using background overlay grids. The creation of faceted meshes from premeshed source entities (i.e. edges or faces) is presented in a more general way, which allows the use of existing procedure of size function implementations. The introduction of the mesh size function has greatly enhanced the capabilities of the three types of size functions that were already available (including a fixed size function, a curvature size function and a proximity size function) and provided nice solutions to the situations where the old size functions did not work desirably. Meshing results of the new size function with controlled mesh sizes are given.

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

1. INTRODUCTION

As everyone knows, 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 small geometry features, and then are smoothly transitioned into the nearby areas of the geometry unless they reach the given size limit. Various methods have been used by different researchers to set up size functions to automatically detect the geometric features and put appropriate mesh size at desired locations, thus eliminate the need of manually locate the local features of the geometry and mesh these entities by desired sizes [1-7]. The background overlay grid size functions that were developed in our previous work illustrated satisfactory performance in mesh size control [8]. However, sometimes creating a mesh that is radiated in a controlled manner from some premeshed boundaries of the domain can also be an efficient way of obtaining desired mesh transition and gradation. The mesh on premeshed boundaries can come from manual operations as desired, but more often it comes from the meshing results of other size functions, or even from imported geometry. In the following, we will list some problems that would be encountered during the meshing processes by using the size function capabilities that had existed, and demonstrate the necessity of creating a new mesh size function.

In the first place, let's take the case of meshing a 2D airfoil as an example, as displayed in Figure 1. The initial mesh size of a fixed size function is defined as a constant value specified by the user. This works for many cases where uniform sizes at the location of the sources are desired. However, for other cases such as the one in Figure 1, it requires that non-uniform mesh sizes be used at the location

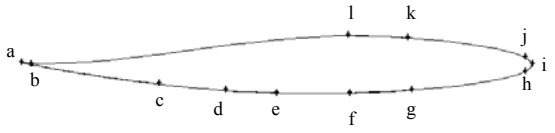


Figure 1 An airfoil geometry split and used as the sources of a fixed size function

of the sources. It is not acceptable to define a constant size along all the airfoil edges. It is necessary for this problem to cluster meshes at the leading edge, at an approximate shock location along the upper surface, and at the trailing edge. Using the current size function implementation, it is required to define at least 6 size functions to cluster elements at the desired locations and then grow the elements away from the airfoil surfaces. One size function uses vertex a as source, and other 5 size functions use edges $\overline{bc}$, $\overline{de}$, $\overline{fg}$, $\overline{hij}$ and $\overline{kl}$, respectively, as sources. These edges can be part of the airfoil surfaces. This is not a very convenient way and takes some trial and error to get desirable mesh clustering at the airfoil surfaces.

A much more convenient and better approach will be to mesh the edges (that represent the surfaces of the airfoil) separately and cluster the edge nodes using the standard edge meshing bunching functionality. Then, have a size function to use those edges as sources and the existing (and varying) mesh sizes at those edges as the initial mesh size. The mesh elements are then allowed to grow using the user specified

ratio and size limit. Note that this concept can be extended into 3D meshing, in which case the size function will take the existing mesh on source faces as the initial size.

For another example, the geometry in Figure 2(a) contains two volumes, exterior volume volume.1 and interior volume volume.2. Four size functions are created (growth rate = 1.2, size limit = 2, cells-per-gap = 3 for proximity size function and angle = 25 for curvature size function) and attached to the geometry as follows:

Proximity size function sfunc.1:

source: volume.1

attachment: volume.1

Curvature size function sfunc.2:

source: all faces of volume.1

attachment: volume.1

Proximity size function sfunc.3:

source: volume.2

attachment volume.2

Curvature size function sfunc.4:

source: all faces of volume.2

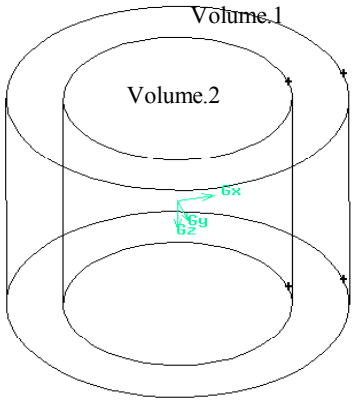
attachment: volume.2

Suppose volume.1 is meshed first and volume.2 second. Then the common face between volume.1 and volume.2 is meshed according to the size functions attached to both volumes. Since the size functions attached to volume.1 give smaller mesh sizes than the size functions attached to volume.2, so the mesh on the common face is dominated by sfunc.1 and sfunc.2 (actually by proximity size function sfunc.1), instead of sfunc.3 and sfunc.4. However, when meshing volume.2, the mesh size is purely controlled by the two size functions attached to volume.2, which will conflict with the meshes generated on the common face, thus causes size jump inside volume.2 or even generates un-usable meshes. (See Figure 2 (b))

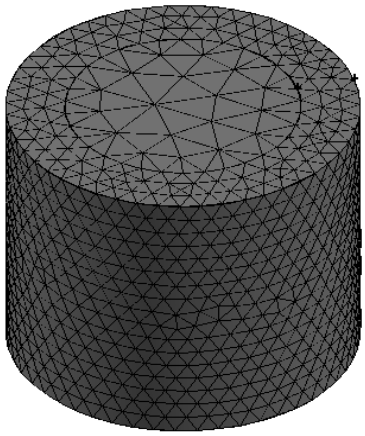
Since size functions attached to the upper topology will also affect its lower topologies, same problem will occur even if volume.2 is meshed first. Here the key issue is that the mesh size on the common face is controlled by four size functions from two sharing volumes, whereas the mesh size in each volume is controlled only by two size functions associated with it.

The mesh size on the common face of above model in Figure 2 is nearly constant. For the similar geometry shown in Figure 3 where the common face has varying sizes due to the changing curvature and gap distance from volume.2, same mismatching of mesh sizes will be encountered.

As one can expect that it is hard to determine which size functions are the dominant ones that give the smallest mesh size on the entities to be meshed. To avoid a poor mesh being generated, a workaround for the above scenario is to attach the size functions in a crossing way, that is, also attach size functions sfunc.1 and sfunc.2 to volume.2 and, similarly,



(a) Geometry containing two volumes



(b) Meshes with big size variation

Figure 2 Problems with old size functions for two similar volumes, one being enclosed by another

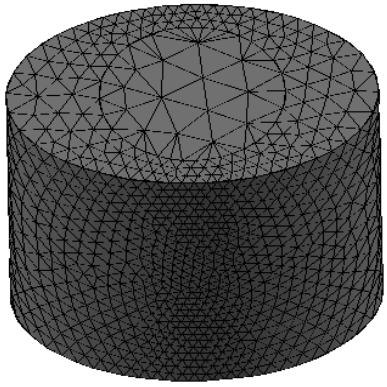


Figure 3 Problems with old size functions for two connected volumes that are not similar

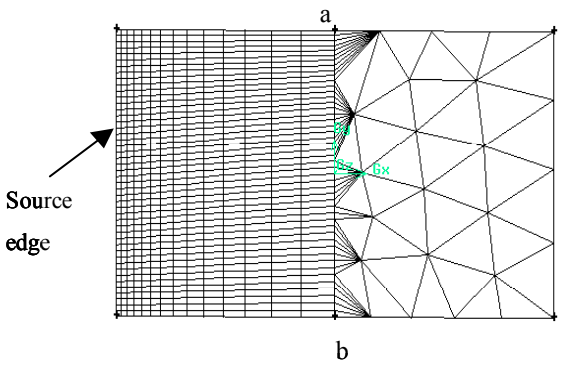
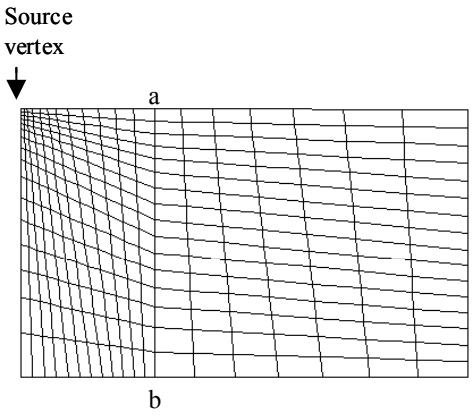
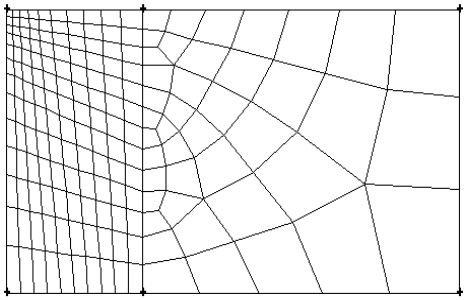


Figure 4 Size jump on the right face from adjusted uniform edge mesh



(a) Mapped mesh on right face



(b) Quad-paved mesh on right face

Figure 5 Mesh size inconsistencies from premeshed edge with non-uniform adjusted mesh

sfunc.3 and sfunc.4 to volume.1. The drawback of this method will be that the bounding box for the background grid will be inevitably larger, which usually uses longer time and larger memory for the background grid to be generated.

For some mesh schemes, the mesh sizes determined by size function have to be adjusted so that the scheme can work. For example in a mapped face, the mesh sizes on opposite paired edges have to be increased or decreased so that their mesh intervals match each other. In Figure 4, there are two faces, face.1 and face.2, connected through common edge $\overline{ab}$. The edge at the left-most side is used as source edge of a fixed size function, and a start size of 0.1 and growth rate of 1.2 are specified. The fixed size function is attached to both the left face face.1 and right face face.2. When face.1 is meshed with the map scheme, the mesh sizes on the common edge of the two faces are decreased (i.e. smaller mesh size than computed by the fixed size function) in order to match the mesh intervals on the opposite paired edge which is also the source edge of the fixed size function. Later, when face.2 that is adjacent to face.1 is meshed with the triangle/pave scheme, the mesh size obtained from the defined size function will be very different from the existing mesh on the common edge, causing big size jump near the common edge.

Figure 5 is similar to Figure 4, but now the source entity is the upper-left vertex, therefore the mesh distributions on the common edge $\overline{ab}$ are non-uniform. Suppose face.1 on the left side is first map meshed. When the right face is meshed with either the map scheme (Figure 5 (a)) or the quad/pave scheme (Figure 5 (b)), great size variation can be observed.

From above illustrations it can be seen that the face or volume meshed first will have great (and usually adverse) impact on the mesh quality of the face or volume across the common boundary that is meshed later and whose attached size functions can not match or smoothly transition the mesh sizes on the premeshed common boundary. It is difficult, if not impossible, to handle the mesh size conflict across the common boundary by using existing size functions, nor by specifying additional sources to them, because existing size functions can only measure the mesh sizes of their sources based on curvature, proximity and fixed sizes, but they can not evaluate their sources by means of existing meshes on the sources that may have arbitrary and variable mesh size distributions that are non-predictable before hand.

Suppose we can define a new size function for the case in Figure 4 and Figure 5 in such a way that it can use the common edge as source entity and be attached to the face.2. This size function is valid only when its source entity has existing mesh when being evaluated. Then this new size function will dominate its attached face and give smaller sizes than previously defined fixed size function in its affected area, so that the mesh size from the existing mesh on the source edge will be grown into the neighboring face.2 and the mesh on face.2 will be improved significantly. This fact suggests that we should create a new type of size function to address this awkward situation to ensure smooth transition across the common edge of different mesh domains. More than just growing the mesh size from boundary into the interior of mesh domain, the definition of this new size function will help to resolve the mesh size conflict that may occur across the common boundaries of adjacent mesh domains.

Sometimes, imported face or edge meshes that were generated from outside the meshing product need to be

preserved and used in the creation of a mesh for a geometry model. Figure 6 demonstrates such a case where face "A" has imported mesh that needs to be taken into account in the generation of volume mesh. It is required that the new mesh grows smoothly from the imported mesh into the rest of the domain of the geometry. None of the previously implemented size functions in our meshing product can satisfy this kind of requirement, thus the mesh size function to be presented in this paper is indispensable for this purpose.

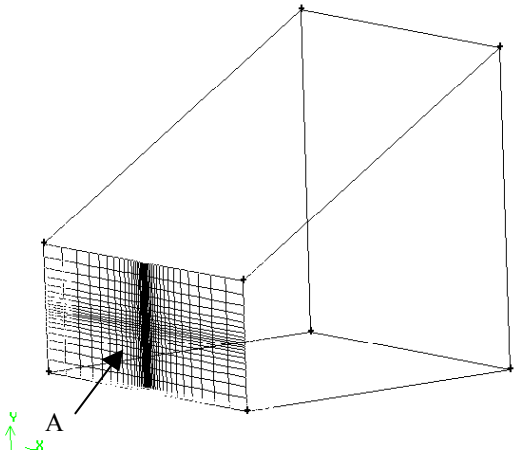


Figure 6 Mesh creation from imported mesh

The goal of this work was to create a new type of size function, named as ***mesh size function***, which respects the existing mesh on premeshed source entities, controls the mesh size growth from premeshed source entities to the attached entities and at the same time, like other size functions we already had, provides very rapid evaluators that would be general for any meshing algorithm.

This paper describes how this new type of size function is implemented using a background overlay grid. The work will be presented by comparing this new size function with old ones, and their differences being emphasized. Application examples of this new size function are given.

2. DEFINITIONS OF SIZE FUNCTIONS

As in our previous implementations, the new mesh size function is also based on a distance-controlled radiation. The parameters that are common to all size functions are used for the new mesh size function too. They are:

- Source entities: Edges or faces that have existing meshes are used as geometric entities. When the mesh size function is defined, the source entities may not have meshes, but they should have meshes available in order to be valid in meshing the attached entities.
- Attached entities: The attached entities on which the mesh size function will have influence include edge, face or volume. For mesh size function, usually the attached entity is different from the source.
- Growth rate: This parameter controls the geometric pace with which the mesh on the premeshed source entities is grown into affected areas.
- 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.

In our previous implementation, all the size functions require a specific parameter, respectively, to define the mesh sizes on the source entities for initialization purpose. In the definition of the mesh size function, the existing meshes on the source entities are directly used as starting sizes, so only the common parameters list above will be enough for its definition.

3. SIZE FUNCTION INITIALIZATION

In preparation for the generation of the background grids, all types of size functions must be initialized differently. This initialization establishes the desired sizes everywhere on the sources. For old size functions, it is needed to generate a reasonable faceted representation of the source entities and then an ideal mesh size is computed for each piece of facet and stored in it.

For the mesh size function, however, we directly use the meshes on the source edge or source face as input. For a meshed edge source, each element of the edge mesh is converted into an edge segment and the length of the segment represents the local mesh size on that edge. An edge segment holder is used to store all the edge mesh segments associated with the premeshed geometry edge. For a meshed face source, we convert each triangle element of the face into a facet and pass the size information of that triangle element to the facet. If the source face has quadrilateral elements, each quad element is split into two triangle elements each of which is converted into a facet of the source face. The mesh size of a face facet is computed as the averaged length of the three sides of its original triangular element from which it is converted. A face facet holder is used to store all the face mesh facets associated with the premeshed geometry face.

When evaluating the mesh size at a point in the space, the point is first projected to a selected edge segment (for edge source case) or face facet (for face source case). If the projection is valid, the mesh size stored in that edge segment or face facet is taken as the start size and then grown to the given point, according to specified growth rate of the mesh size function.

4. BACKGROUND GRID GENERATION

4.1 Improved procedures of establishing mesh sizes at nodes of background grid

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, realized by the refining process. This procedure was described in detail in our previous work [8] and will not be listed here. The only difference is that at each corner node, the background grid will also use the mesh size radiated from the mesh size functions and compare it

with all other mesh sizes obtained from old size functions, when applied together.

However, when establishing values at the background grid nodes, an improvement can be made regarding the approach of growing the mesh size from the source entity to a given point. Previously, we used an interactive procedure to determine the spacing at a given point as influenced by a particular source. Using the prescribed geometric growth factor, 'g', we essentially "march" to the desired point from the source, applying the growth factor at each interval and summing the result. Rather than iterate in this fashion, the desired mesh size can be obtained by analytically summing the terms of the geometric series given as

$$S_n = S_0 \cdot g^n \quad (S_0 \text{ is the spacing at the source, } n \geq 0)$$

The distance from the source entity to the given point is the sum of mesh sizes at incremental intervals except for the first mesh size on the source, and then the proper terms of the series can be listed as:

$$R_n = R_{n-1}\ g \quad (n > 0)$$

$$R_0 = 0 \ (n = 0)$$

Or in full expression:

$$R_0 = 0,\ R_1 = S_0 \cdot g\ ,\ R_2 = S_0 \cdot g^2\ , \ldots\ R_n = S_0 \cdot g^n$$

Knowing the Euclidean distance (R) from the source to the node in question, we can sum all the items in the series until R_n, so that we can then directly solve for the exponent as follows:

$$R_n = S_0\ (g^n - 1) / (g - 1) - S_0$$

$$g^n = R\ (g-1) / S_0 + g$$

$$n = \ln (R\ (g-1) / S_0 + g) / \ln (g)$$

Finally we take the integer part of the obtained n value.

$$n = (\text{int})\ n$$

which can then be used to immediately evaluate the spacing at the node without the need for iteration. The desired point will locate within the region between two subsequent distances R_n and R_{n+1} from source that are measured at incremental steps n and n+1, respectively. Then the following condition can be satisfied:

$$R_n \leq R \leq R_{n+1}$$

This would simplify the evaluation of S_n and speed up the calculation.

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

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

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

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

The final size is the smallest one of the defined size limit and the all computed sizes (if a corner point is affected by several size functions).

4.2 Improvement to projections to source entities under way

According to our statistics obtained from timing the profile of size function creation, it is found that the bottleneck of the size function speed is the projection of the corner nodes of background grid to the faceted source entities, which counts for about 90% of the total time. The remaining time is spent for other operations such as growing the initial mesh size on source entities to a given point along the distance, inserting newly computed mesh size into a sorted list, and getting mesh size at a shared point from the list. The most time used for projection is spent in searching the best facet to project the node. An investigation in improving the projection process is being under way which tries to project a list of nodes in one background grid to the best facets at the same time. This approach will significantly reduce the time in background grid generation once successfully realized.

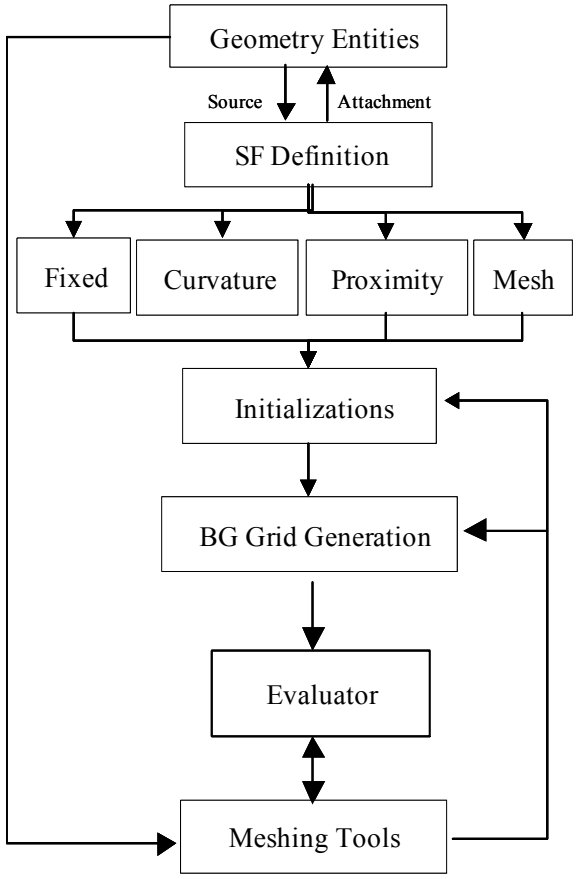


Figure 7 Flow chart of the size function applications

4.3 Flow chart of the background grid size function approach

No matter what types of the size functions to use, except for the differences in initializations, the same procedures will be followed when apply these size functions to the meshing processes. The following chart in Figure 7 illustrates the general procedures we used in the meshing processes for all meshing schemes. After defined, the size functions are attached to the geometric entities via a specially designed data structure. Initialization of size functions is triggered if any of the attached entities or their lower topologies is being meshed. The background grid generation for the attached entities follows the initialization process, creating a specific set of background grid for each group of entities that have identical size functions attached. The established background grid serves as an evaluator providing mesh size information to the meshing process. After entering the meshing session, the mesh size at a given point is evaluated quickly through tri-linear interpolations in a background cell into which the point falls. By the returned mesh size value, the next mesh node is placed along certain direction.

5. EXAMPLES

A few examples are given below to show the application of the new mesh size function or its combination with other types of size functions in the meshing process. Smooth meshes that were unable to be generated before have been generated due to the introduction of the new mesh size function.

5.1 Updated meshing results for co-centric volumes

Figure 8 is the updated meshing results of the example in the beginning of this paper (see Figure 2). To prevent the size jump in the interior volume.2, a mesh size function is created that uses the common face as source and is attached to the interior volume. In order for the mesh size function to be useful to its attachment, the exterior volume.1 should be meshed first in this case, so that the common face inherits its meshes from the meshing process of exterior volume before interior volume.2 is meshed, thus the mesh size function using the common face as source can be valid for use in meshing volume.2.

Figure 9 gives the new meshing results corresponding to Figure 3. Similarly to Figure 8, the meshes of the interior volume are radiated nicely from the common face, although the meshes on common face have varying degrees of sizes than in the previous case.

5.2 Remeshed results for connected faces

Next, referring to Figure 4, we have defined a second mesh size function that uses the common edge of the two connected faces as source and attach it to the face on the right side (see Figure 10). Since the mesh size function has smaller size distributions everywhere in the right face than the original fixed size function and so will dominate the mesh size selection in the whole domain of the right face, the tri/pave algorithm will use the mesh size from the mesh size function to position nodes, forming smooth mesh transitions from the common edge and across the whole face on face.2. The adjustment to the mesh distributions on the common edge does not deteriorate the mesh quality on the right face any more.

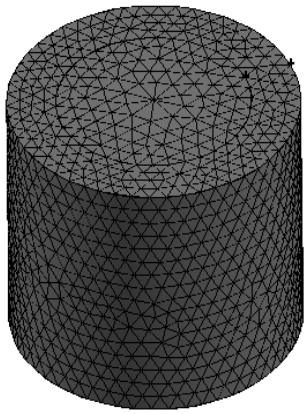


Figure 8 Remeshed results from Figure 2 when mesh size function is applied

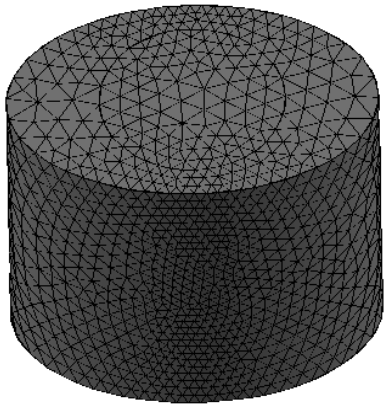


Figure 9 Remeshed results from Figure 3 when mesh size function is applied

Similar results to the above have been obtained in Figure 11 for the case displayed in Figure 5 where the initial size function start from the upper-left vertex, instead of the left-most edge as in Figure 4. No matter the right face is meshed with the map scheme (Figure 11 (a)) or the quad/pave scheme (Figure 11 (b), the meshes on the face are grown in such a way that you will not notice any sudden changes of mesh sizes near the common edge and it looks like the whole meshes are smoothly radiated from the same upper-left vertex without size jumping.

5.3 New meshing results for volumes with imported face mesh

We discussed the impossibility of generating a volume mesh that is required to radiate from the imported face mesh and concluded that there was no easy way of doing it with the old size function capabilities in our mesh sizing tool. However, with the realization of the mesh size function, this task becomes very easy. Simply specify the face "A" having imported mesh as the source face (see Figure 6) and specify the volume to be meshed as attachment entity of the mesh size function, and then start meshing the volume. The volume, which could be imported together with the meshed face or created from the imported face (e.g. by sweeping the given face along specified path) within Gambit product, will be meshed according to the user's specification. Figure 12(a) shows the mesh on the boundary surface of the volume after the meshing process is finished, and Figure 12(b) is the internal mesh patterns of the volume. A growth rate of 1.2 is used in the definition and the size limit is large enough not to be reached within the domain of the model.

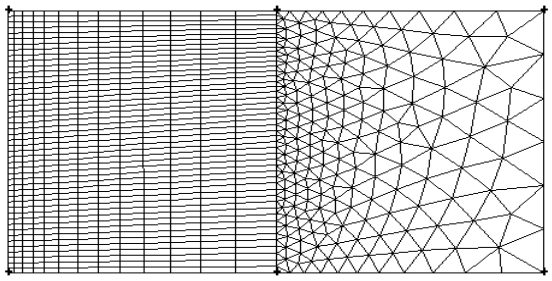
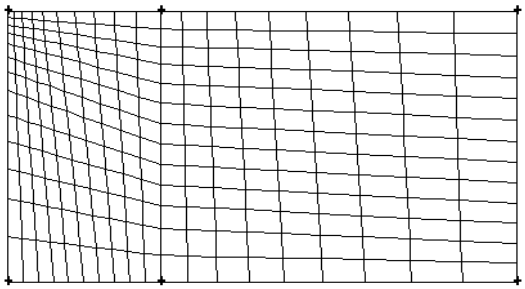
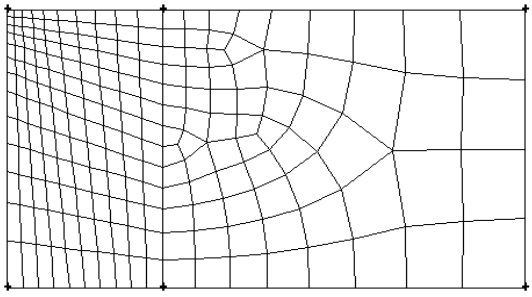


Figure 10 Mesh distributions on right face when the common edge is used for mesh size function

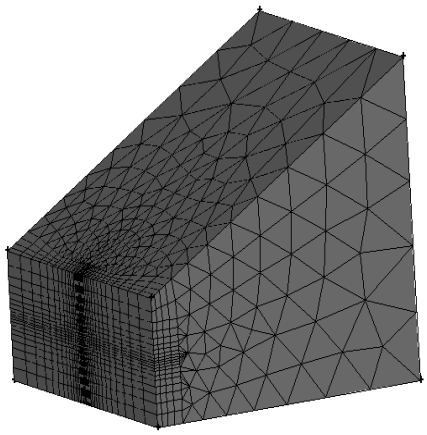


(a) Mapped mesh on right face

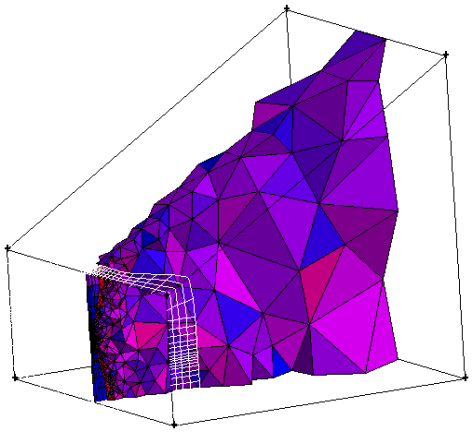


(b) Quad/paved mesh of right face

Figure 11 Mesh distributions on right face using a mesh size function from the common edge



(a) Mesh on boundary surface



(b) Internal volume mesh

Figure 12 Meshing results of volume using imported face mesh as source

CONCLUSION

From the established method of size functions using the background overlay grids, a new ***mesh size function*** has been set up for controlling mesh sizes and radiation from premeshed geometric entities (i.e. edges and/or faces). The defined mesh size function has provided supplemental means to assist all the meshing tools where the size functions that were implemented previously sometimes could not meet the

special needs. The details on how to construct the new mesh size function has been described and its comparison with other size functions presented. The proposed mesh size function has been implemented in Gambit product, and its efficiency has been illustrated by successful meshing examples with satisfactory results.

REFERENCES

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

[2] 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)

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

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

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

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

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

[8] J Zhu, Ted Blacker, Rich Smith, Background Overlay Grid Size Functions, Proceedings of 11th International Meshing Roundtable. pp65-73 (2002). Sept. 15-18, 2002. Ithaca, New York, USA.