# A Vector Level Control Function for Generalized Octree Mesh Generation

Tao Chen, Jeffery Johnson†, Robert W. Dutton

AEL 231D, Integrated Circuits Laboratory, Dept. of Electrical Engineering,
Stanford University, Stanford, CA 94305
†IBM Microelectronics Division, Essex Junction, VT 05402

Abstract

Due to the complexity of 3D geometry structure, regular gridding algorithms often generate a very large amount of mesh points which requires an enormous amount of computational power for later device and process simulations. A vector level control function and related generalized octree method is presented in this paper, and it allows mesh refinement in selected directions only. Thus the grid points can be placed more optimally and that enables efficient device and process simulations.

## 1. Introduction

Some quadtree/octree based mesh generation schemes have been employed for finite element methods used in device [1] and process simulations [3] because of the simplicity of the tree data structure and its advantage in faciliating adaptive mesh refinement. The quadtree mesh generation scheme recursively partitions a square into four smaller, equal size squares; the octreetakes a cube and partitions it into eight smaller cubes. This set of squares or cubes form a tree. Usually a one-level difference rule is enforced for any two neighboring cells.

While this regular quadtree method works well in 2D, in part because of the limited complexity of geometry, the analogous 3D octree method runs into problems in many cases and generating excessive grid. For example, suppose one has a BJT with base dimension of $1 \times 1 \times 0.1 \mu m^3$. Using the regular octree method, more than $50 \times 50 \times 5$ grid points are needed for merely five grid planes in the z direction for this region alone. Clearly the amount of grid points required for more complex structures can quickly become excessive. A generalized octree method which solves this problem nicely is presented in this work. The new method is able to refine in only the required directions. At each refinement stage, there are seven choices. An octant can be refined in all xyz directions, or only in xy, or xz, or yz directions, or only in x, or y, or z directions.

## 2. Algorithm

To achieve this generalized refinement scheme, a vector level control function is defined. Like the scalar function in [2], an integer valued 3D vector function is now defined on the solid volume. It indicates the directions for which the refinement will be performed. For example, in the previous BJT example, a vector with a large z component and moderate x, y components can be assigned to the base region. This vector level control function also can be computed based on simulation errors. For example, if the derived solution error shows that denser mesh is needed in x direction, but not in y direction, a refinement with a level control function coupled with appropriate error estimator can readily be performed.

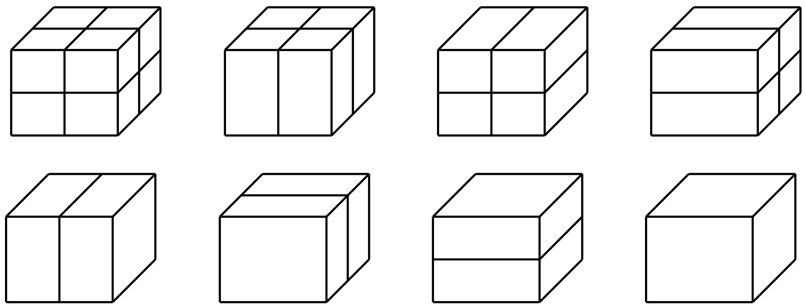

Figure 1: An octant can be refined in xyz, or xy, or xz, or yz, or only x, or y, or z directions.

Every octant also has a level vector ($L_{oct}$) associated with it. (The octant usually is no longer a cube.) For a given octant, to determine which directions are to be refined, the level control function ($L_c$) is evaluated on the center of the octant. If $L_{oct,x} < L_{c,x}$, then the octant will be refined in x direction, and similarly for the y, z directions. An octant will then be refined in one of the seven cases as shown in Fig. 1, or not at all. Each child octant will have a level vector $L_{child}$ satisfying

$$L_{child,i} = L_{oct,i} + 1 \qquad if\ i\ direction\ is\ refined, \qquad (for\ i = x, y, z). \tag{1}$$

To abide the one-level difference rule, for every octant refined, its neighbors are found using the tree structure and updated. If an octant is refined in y direction, we update the four (six if refined in more than one direction) neighboring (left, right, bottom, top) octants. The level control function sets values on the centers of the neighboring octants by the following rule (this example assumes only refinement in y direction, similar rules follows easily).

$$\begin{gathered} if\ (L_{neighbor,c,y} < L_{oct,y} - 1),\ L_{neighbor,c,y} = L_{oct,y} - 1; \\ else\ do\ nothing. \end{gathered}$$

After the initial tree generation, the entire octree is traversed again so that additional refinements can be performed if some values of the level control function are updated after the octants are refined. The octree generation process finishes when no more octants need to be generated. The final tree structure guarantees the one level difference in every direction and has the refinements in the directions required. Fig. 2 is a simple mesh using the generalized octree method.

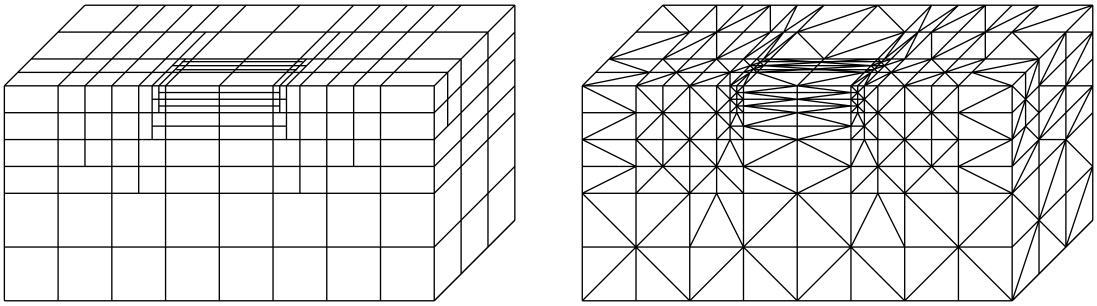

Figure 2: A simple 3D example using the generalized octree method.

Triangulation/Tetrahedralization is finally performed on the quadtree/octree. Extreme aspect ratio elements can often be generated. A local topological Delaunay transformation similar to the one proposed in [4] is then performed, where some pairs of tetrahedra are transformed into groups of three tetrahedra and the reverse process. This is analogous to the 2D case where edge swapping will produce a constrained Delaunay mesh for a convex hull. The overall mesh quality thus is improved by these topological transfromations.

## 3. Conclusion

This generalized octree method provides great flexibility for mesh refinement. It makes simulations with complicated structures easier due to its ability to place grids optimumly. It also provides a vehicle for the development of vector error estimators, where the refinements it performs could enhance the simulation accuracy efficiently while not adding too many points. Overall it reduces many grid points, and still generates a quality mesh.

Fig. 3 shows a mesh for a BJT used in high frequency RF circuits. It has very long but shallow base and emitter regions. The generalized octree mesh gives a very good fit according to the doping variation, and a large amount of grid points are concentrated along the junctions. Fig. 4 gives a cross section view of the mesh.

## References


[1] P. Conti, N. Hitschfeld and W. Fichtner, "An Octree-Based Mixed Element Grid Allocator For Adaptive 3D Device Simulation", *NUPAD III, Technical Digest, 1991.*

[2] D. Yang, K. Law and R. Dutton, "An Automated Mesh Refinement Scheme Based On Level-Control Function", *NUPAD IV, Technical Digest, 1992.*

[3] Z. Sahul, R. Dutton and M. Noell, "Grid and Geometry Techniques for Multi-Layer Process Simulation", *Proc. SISDEP 1993.*

[4] N. Golias and T. Tsiboukis, "Three-Dimensional Automatic Adaptive Mesh Generation", *IEEE Trans. on Magnetics, Vol. 28, No. 2, March 1992.*

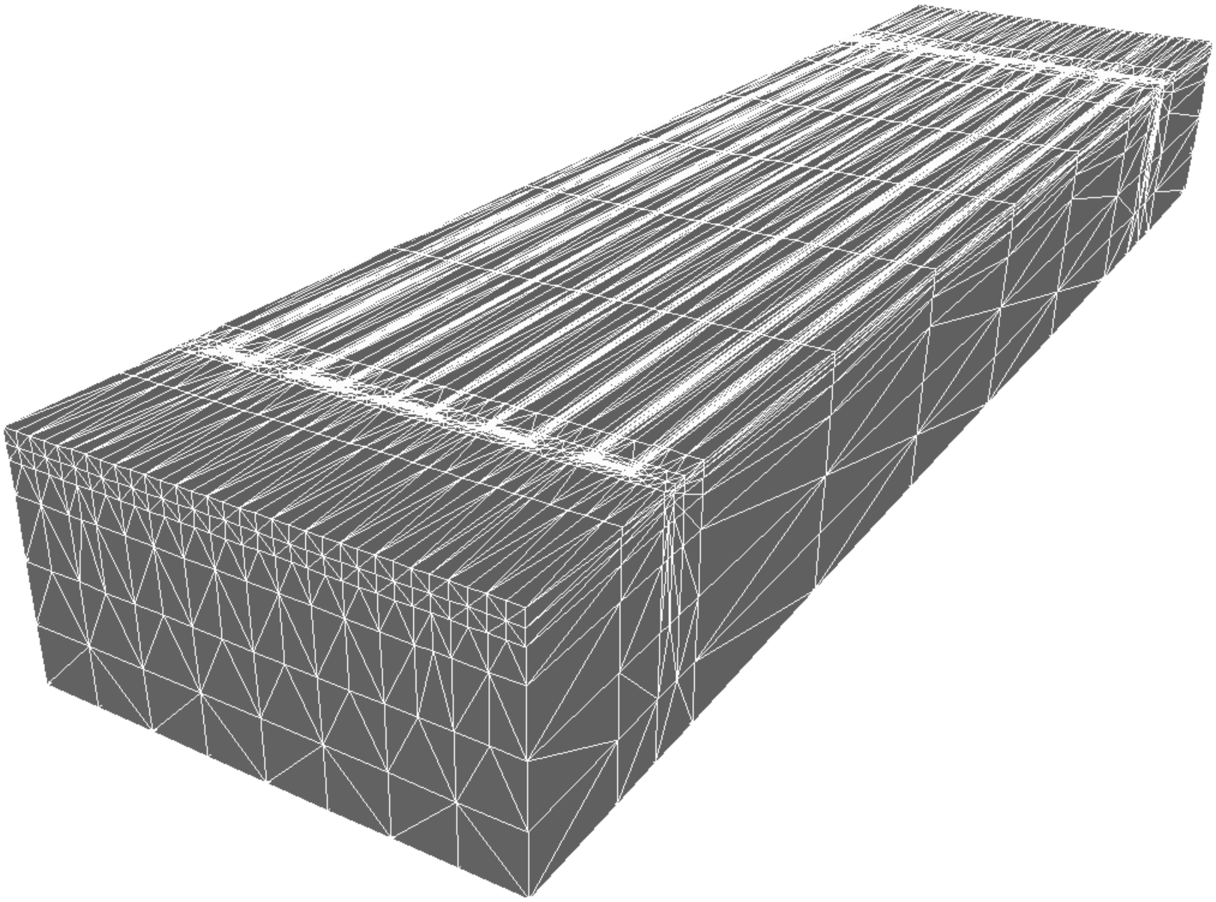

Figure 3: Mesh for a BJT used in high frequency RF circuits. The three base and two emitter regions are very long, but also very shallow. The generalized octree mesh concentrates a large amount of grid points along the junctions.

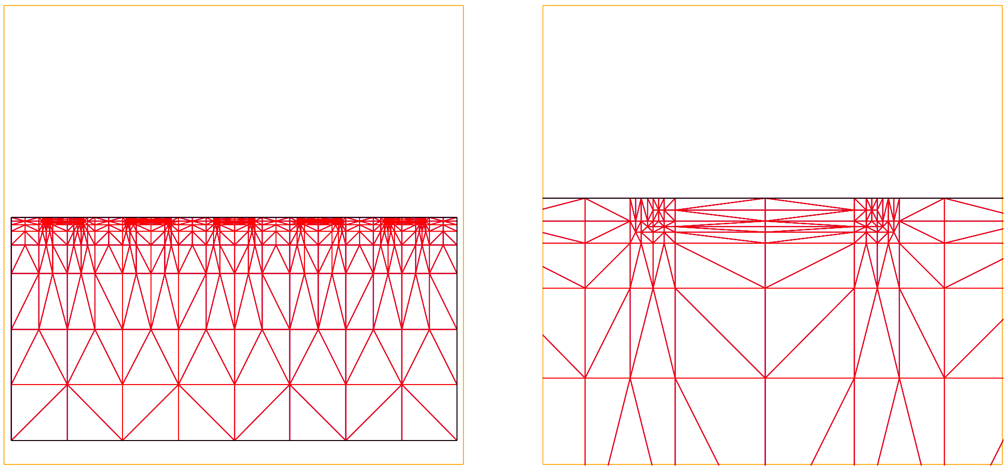

Figure 4: 2D cross section mesh of the BJT and its zoom in view of the emitter region.