Efficient Implementation of Operators on Semi-Unstructured Grids

Christoph Pflaum, David Seider
Institut für Angewandte Mathematik und Statistik,
Universität Würzburg
pflaum@mathematik.uni-wuerzburg.de,
seider@mathematik.uni-wuerzburg.de

Abstract. Semi-unstructured grids consist of a large structured grid in the interior of the domain and a small unstructured grid near the boundary. It is explained how to implement differential operators and multigrid operators in an efficient way on such grids. Numerical results for solving linear elasticity by finite elements are presented for grids with more than 10^8 grid points.

1 Introduction

One main problem in the discretization of partial differential equation in 3D is the large storage requirement. Storing the stiffness matrix of a FE-discretization of a system of PDE's on unstructured grids is very expensive even in case of relative small grids. To avoid this problem, one prefers fixed structured grids, since these kind of grids lead to fixed stencils independent of the grid point. Another interesting property of structured grids is that several numerical algorithms can be implemented more efficient on structured grids than on unstructured grids (see [7]). On the other hand, fixed structured grids do not have the geometric flexibility of unstructured grids (see [3], [10], [8], and [9]). Therefore, we apply semi-unstructured grids (see [6] and [5]). These grids consist of a large structured grid in the interior of the domain and a small unstructured grid, which is only contained in boundary cells. Since semi-unstructured grids can be generated automatically for general domains in 3D, they have the geometric flexibility of unstructured grids. In this paper, we explain how to implement discrete differential equations in an efficient way on semi-unstructured grids. It will be shown that algorithms based on semi-unstructured grids require much less storage than on unstructured grids. Therefore, these grids are nearly as efficient as structured grids.

2 Semi-Unstructured Grids

Assume that $\Omega \subset \mathbb{R}^3$, is a bounded domain with a piecewise smooth boundary. Semi-unstructured grids are based on the following infinite structured grids with

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2331, pp. 622–631, 2002.
© Springer-Verlag Berlin Heidelberg 2002

meshsize $h > 0$:

$$\Omega_h^\infty := \{(ih, jh, kh) \mid i, j, k \in \mathbb{Z}\} \quad .$$

The set of cells of this grid is

$$\mathcal{Z}_h^\infty := \{[ih, (i+1)h] \times [jh, (j+1)h] \times [kh, (k+1)h] \mid i, j, k \in \mathbb{Z}\}\,.$$

Theses cells are called *interior*, *exterior*, or *boundary cells*. The classification in these three types of cells has to be done carefully (see [6]). Otherwise, it is very difficult to construct a semi-unstructured grid. Let us denote the set of *boundary cells* by $\mathcal{Z}_h^{\text{boundary}} \subset \mathcal{Z}_h^\infty$ and the set of *interior cells* by $\mathcal{Z}_h^{\text{interior}} \subset \mathcal{Z}_h^\infty$.

To obtain a finite element mesh, we subdivide every interior and every boundary cell by tetrahedra, such that suitable properties are satisfied (see [6]). One of these properties is that the interior angles ϕ of every tetrahedron are smaller than a fixed maximal angle $\phi_{max} < 180°$, which does not depend on the meshsize h and the shape of the domain. This means

$$\phi \leq \phi_{max} < 180° \tag{1}$$

for every interior angle. A suitable subdivision of the boundary cells is described in [5]. Here, we briefly explain the subdivision of the interior cells. To this end, let us mark the corners of an interior cell Z as in Figure 2. Then, this cell is subdivided as follows:

Regular Subdivision

```
Tet. 1: WNT, WND, WST, EST
Tet. 2: EST, WND, WST, ESD
Tet. 3: WND, WSD, WST, ESD
Tet. 4: EST, WND, ESD, END
Tet. 5: ENT, WNT, EST, END
Tet. 6: WNT, WND, EST, END
```

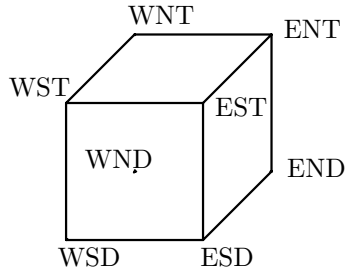

Fig. 1. Subdivision of an Interior Cell Z.

The subdivision τ_h of the interior cells and boundary cells by tetrahedra leads to a discretization domain

$$\Omega_{dis,h} = \bigcup_{\Lambda \in \tau_h} \overline{\Lambda}$$

and a set of nodal points

$$\mathcal{N}_h = \bigcup_{\Lambda \in \tau_h} \mathcal{E}(\Lambda),$$

where $\mathcal{E}(\Lambda)$ is the set of corners of a tetrahedron Λ.

We have to distinguish three types of nodal points:
I. The regular interior points:

$$\mathcal{N}_{h,I} := \{P \in \mathcal{N}_h \mid P \text{ is corner point of eight interior cells.}\}.$$

II. The boundary points and the points contained in boundary cells:

$$\mathcal{N}_{h,II} := \{P \in \mathcal{N}_h \mid P \text{ is not adjacent or contained in an interior cell.}\}.$$

III. The regular points near the boundary:

$$\mathcal{N}_{h,III} := \{P \in \mathcal{N}_h \mid P \text{ is corner point of at least one interior cell and one boundary cell.}\}.$$

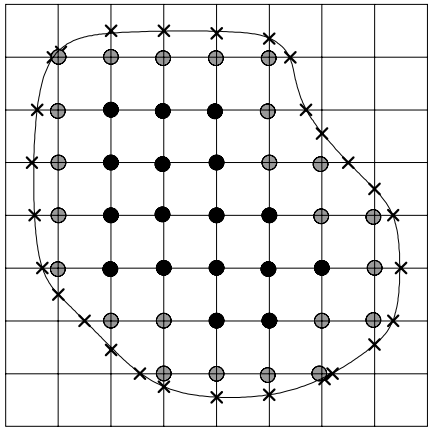

Fig. 2. Grid points $\mathcal{N}_{h,I}$, $\mathcal{N}_{h,II}$, and $\mathcal{N}_{h,III}$ for a semi-unstructured grid in 2D.

Observe that

$$\mathcal{N}_h := \mathcal{N}_{h,I} \cup \mathcal{N}_{h,II} \cup \mathcal{N}_{h,III}.$$

Figure 2 depicts the three types of nodal points for semi-unstructured grid in 2D.

For the discretization of partial differential equations, we apply the finite element space with linear elements on τ_h:

$$V_h := \left\{ v \in H^1(\Omega) \;\middle|\; v\big|_\Lambda \text{ is linear on every tetrahedron } \Lambda \in \tau_h \right\}.$$

3 Implementation of Differential Operators with Constant Coefficients

To avoid a large storage requirement, we want to implement operators like discrete differential operators without storing the stiffness matrix or the local stiffness matrices. This always can be done on a finite element grid by a recalculation

of the local stiffness matrices when they are need. But such an implementation is very time consuming. On structured grids one can get a more efficient implementation by fixed stencils.

In this section, we explain how to obtain an efficient implementation of discrete differential operators with constant coefficients on semi-unstructured grids. For reasons of simplicity, we restrict ourselves to the discrete differential operator corresponding to the bilinear form

$$(u, v) \to a(u, v) = \int_{\Omega} \nabla u \, \nabla v \, d\mu. \tag{2}$$

To obtain an efficient implementation of the discrete differential operator corresponding to this bilinear form, one has to distinguish three cases:

I. Implementation at regular interior points $M \in \mathcal{N}_{h,I}$.
The eight cells next to the regular interior points $M \in \mathcal{N}_{h,I}$ are interior cells. Therefore, we can implement the operator corresponding to (2) by the following fixed stencil

$$\begin{aligned}
&(20.0 * u_M \\
&-4.0 * (u_T + u_D + u_E + u_W) \\
&-2.0 * (u_N + u_S) \\
&-(u_{ND} + u_{WN} - u_{WT} - u_{ED} + u_{ST} + u_{ES} \\
&-u_{EST} - u_{WND})) * h/3.0
\end{aligned}$$

Here, E means the next point in the east direction: $E = (1, 0, 0) + M$. The other directions W, N, S, D, T are defined analogously.

II. Implementation at points $P_1 \in \mathcal{N}_{h,II}$.
To calculate a discrete differential operator at a point $P_1 \in \mathcal{N}_{h,II}$, we need the local stiffness matrix of every tetrahedron $T \in \tau_h$ adjacent to P_1. These tetrahedra are contained in boundary cells, since $P_1 \in \mathcal{N}_{h,II}$. We do not want to store the local stiffness matrix of T completely and we do not want to recalculate them each time they are needed. Therefore, we use the following approach, which stores 13 values for each tetrahedron $T \in \tau_h$ contained in a boundary cell. Using these 13 values one can calculate the local stiffness matrix easily for different bilinear forms. To explain the definition of the 13 values $\xi_0, \xi_{x,1}..., \xi_{x,4}, \xi_{y,1}..., \xi_{y,4}, \xi_{z,1}..., \xi_{z,4}$, let $T \in \tau_h$ be a tetrahedron contained in a boundary cell and let us denote P_1, P_2, P_3, P_4 the corners of T. Furthermore, let v_p be the nodal basis function at the corners $p = P_1, P_2, P_3, P_4$ of T and

$$M = \frac{1}{4}(P_1 + P_2 + P_3 + P_4).$$

Then, let

$$\begin{aligned}
\xi_0 &:= \mathrm{vol}(T)/6, \\
\xi_{x,i} &:= \xi_0 \cdot \frac{\partial v_{P_i}}{\partial x}(M), \quad \text{for } i = 1, 2, 3, 4,
\end{aligned}$$

$$\xi_{y,i} := \xi_0 \cdot \frac{\partial v_{P_i}}{\partial y}(M), \quad \text{for } i = 1,2,3,4,$$
$$\xi_{z,i} := \xi_0 \cdot \frac{\partial v_{P_i}}{\partial z}(M), \quad \text{for } i = 1,2,3,4.$$

Using these values, one can calculate the stiffness matrix corresponding to the bilinear form $a(u,v)$. For example, the entry $a(v_{P_1}, v_{P_2})$ of the local stiffness matrix is

$$\int_\Omega \nabla v_{P_1} \nabla v_{P_2}\, d\mu = (\xi_{x,1}\xi_{x,2} + \xi_{y,1}\xi_{y,2} + \xi_{z,1}\xi_{z,2})/(6\xi_0).$$

III. Implementation at points $P_1 \in \mathcal{N}_{h,III}$.
The cells adjacent to a point $P_1 \in \mathcal{N}_{h,III}$ are boundary cells and interior cells. Therefore, we have to distinguish to cases:

a) Let c be a boundary cell adjacent to P_1. Then, the local stiffness matrix of c can be calculated using the local stiffness matrices of the tetrahedra $T \in \tau_h$ contained in c.
b) Let c be an interior cell adjacent to P_1. Then, the local stiffness matrix corresponding to (2) is a fixed 8×8 matrix, which does not depend on c. Therefore, storing this matrix is very inexpensive.

4 Implementation of Operators with Variable Coefficients

Now, let us explain the implementation of discrete differential operators with variable coefficients.

For reasons of simplicity, let us restrict ourselves to the bilinear form

$$(u,v) \to a(u,v) = \int_\Omega b \nabla u \nabla v\, d\mu, \tag{3}$$

where $b \in \mathcal{C}(\Omega)$ and $\min_{p\in\Omega}(b(p)) > 0$. It is well-known, that, in general, one cannot directly implement the differential operator corresponding to a. Therefore, one has to approximate a by a bilinear form a_h. The standard way to approximate a is to apply a numerical integration with one Gauss integration point for every $T \in \tau_h$. Here, we use another approach. To explain this approach, let us classify the tetrahedra of τ_h in the following way:

$$\tau_{h,i} := \{T \in \tau_h \mid T \text{ is contained in an interior cell}\}$$
$$\tau_{h,b} := \{T \in \tau_h \mid T \text{ is contained in a boundary cell}\}.$$

Obviously, it is $\tau_h = \tau_{h,i} \cup \tau_{h,b}$. Let us define an interpolation operator I_h as follows:

$$I_h : \mathcal{C}(\Omega) \to L^\infty(\Omega)$$
$$I_h(b)(p) := b(p) \quad \text{if } p \text{ is the middle point of an interior cell.}$$
$$I_h(b)(p) := b(p) \quad \text{if } p \text{ is the middle point of a tetrahedron } T \in \tau_{h,b}.$$
$$I_h(b) \quad \text{is constant on every interior cell.}$$
$$I_h(b) \quad \text{is constant an every tetrahedron } T \in \tau_{h,b}.$$

Now, let the approximation of a be the following bilinear form

$$(u, v) \to a_h(u, v) = \int_\Omega I_h(b)\, \nabla u\, \nabla v\, d\mu$$

By the Bramble-Hilbert Lemma and by some calculations one can show (see [2] or [1]),

$$\begin{aligned} |a(u, v) - a_h(u, v)| &\le Ch\|b\|_{C^1}\|u\|_{H^1}\|v\|_{H^1} \\ |a(u, v) - a_h(u, v)| &\le Ch^2\|b\|_{C^2}\|u\|_{H^2}\|v\|_{H^2} \\ |v|^2_{H^1} &\le Ca_h(v, v). \end{aligned}$$

Using these inequalities one can prove an $O(h^{2-\delta})$ convergence with respect to the L^2-norm for the finite element solution corresponding to the bilinear form (3) for every $\delta > 0$ (see [6] and [4]) . The advantage of the above construction of a_h is that one can implement the differential operator corresponding to a_h similar to the construction in section 3.

I. Implementation at regular interior points $M \in \mathcal{N}_{h,I}$.
The discrete differential operator corresponding to (3) can be implemented by the following stencil at points $M \in \mathcal{N}_{h,I}$

$(b_{cellWSD} * (3.0 * u_M - u_W - u_S - u_D)+$
$b_{cellESD} * (5.0 * u_M - 3.0 * u_D - u_E - u_S - u_{ES} + u_{ED})+$
$b_{cellWND} * (7.0 * u_M - 3.0 * (u_W + u_D) - u_N - u_{ND} - u_{WN} + 2.0 * u_{WND})+$
$b_{cellEND} * (5.0 * u_M - 3.0 * u_E - u_N - u_D - u_{ND} + u_{ED})+$
$b_{cellWST} * (5.0 * u_M - 3.0 * u_W - u_T - u_S - u_{ST} + u_{WT})+$
$b_{cellEST} * (7.0 * u_M - 3.0 * (u_E + u_T) - u_S - u_{ST} - u_{ES} + 2.0 * u_{EST})+$
$b_{cellWNT} * (5.0 * u_M - 3.0 * u_T - u_N - u_W - u_{WN} + u_{WT})+$
$b_{cellENT} * (3.0 * u_M - u_E - u_N - u_T)$
$) * h/6.0.$

Here, $cellWSD$ means the middle point of the next cell point of M in the west-south-down direction.

II. and III. Implementation at points $P_1 \in \mathcal{N}_{h,II} \cup \mathcal{N}_{h,III}$.
The discrete differential operator at points $P_1 \in \mathcal{N}_{h,II} \cup \mathcal{N}_{h,III}$ can be implemented for variable coefficients similar to the case of constant coefficients. To explain this, let $\mathcal{M}_{const}$ be the local stiffness matrix for a constant coefficient 1.0 at a tetrahedron $T \in \tau_{h,b}$ or an interior cell c. Then, $I_h(b)$ is a constant value β on this tetrahedron T or interior cell c, respectively. Therefore, the local stiffness matrix $\mathcal{M}_{var}$ in case of variable coefficients is

$$\mathcal{M}_{var} = \beta \mathcal{M}_{const}.$$

The above implementation of discrete differential operators for variable coefficients is not much more time and storage consuming than for constant coefficients. To explain this, one has to study the implementation at regular interior

points $P_1 \in \mathcal{N}_{h,I}$, since the computational time at these points dominates the total computational time on semi-unstructured grids. Counting the number of operations shows that the above discrete differential operator for variable coefficients requires $71/19 \approx 3.7$ more operations than in case of constant coefficients. Since, the computational time often is strongly influenced by problems with a small cache in the computer architecture, the computational time will increase by a factor smaller than 3.7 (see numerical results in section 6).

5 Implementation of Multigrid Operators and Coarse Grid Operators

For reasons of simplicity, let us assume that $\Omega \subset]0,1[^3$ and that $h = 2^{-n}$. A multigrid algorithm is based on a sequence of fine and coarse grids

$$\mathcal{N}_h = \mathcal{N}_n, \mathcal{N}_{n-1}, \cdots, \mathcal{N}_2, \mathcal{N}_1.$$

To define these grids, we recursively define sets of coarse grid cells:

$$\begin{aligned}
\mathcal{Z}_n^{\text{boundary}} &:= \{c \in \mathcal{Z}_{2^{-n}}^{\infty} \mid c \text{ is a boundary cell on the finest grid.}\} \\
\mathcal{Z}_k^{\text{boundary}} &:= \{c \in \mathcal{Z}_{2^{-k}}^{\infty} \mid \text{there exists a cell } c_b \in \mathcal{Z}_{k+1}^{\text{boundary}} \text{ such that } c_b \subset c\} \\
\mathcal{Z}_n^{\text{interior}} &:= \{c \in \mathcal{Z}_{2^{-n}}^{\infty} \mid c \text{ is an interior cell on the finest grid.}\} \\
\mathcal{Z}_k^{\text{interior}} &:= \{c \in \mathcal{Z}_{2^{-k}}^{\infty} \mid \text{there exist 8 cells } c_i \in \mathcal{Z}_{k+1}^{\text{interior}} \text{ such that } c_i \subset c\}
\end{aligned}$$

Furthermore, we define the maximal and minimal coarse grid

$$\begin{aligned}
\mathcal{N}_{k,\max} &:= \{p \mid p \text{ is a corner of a cell } c \in \mathcal{Z}_k^{\text{interior}} \cup \mathcal{Z}_k^{\text{boundary}}\}, \\
\mathcal{N}_{k,\min} &:= \{p \mid p \text{ is a corner of exactly 8 cells } c \in \mathcal{Z}_k^{\text{interior}} \cup \mathcal{Z}_k^{\text{boundary}}\}.
\end{aligned}$$

The coarse grid $\mathcal{N}_k$ has to be constructed such that the following inclusions are satisfied:

$$\mathcal{N}_{k,\min} \subset \mathcal{N}_k \subset \mathcal{N}_{k,\max}.$$

Here, we do not want to explain in detail how to choose $\mathcal{N}_k$. But let us mention, that the construction of $\mathcal{N}_k$ depends on the boundary conditions. The coarse grid $\mathcal{N}_k$ is the union of two disjoint types of grids:

$$\begin{aligned}
\mathcal{N}_k &= \mathcal{N}_{I,k} \cup \mathcal{N}_{II,k}, \\
\mathcal{N}_{I,k} &= \{p \mid p \text{ is a corner of exactly 8 cells } c \in \mathcal{Z}_k^{\text{interior}}\}.
\end{aligned}$$

To obtain a multigrid algorithm, we need a function space V_k on the coarse grid $\mathcal{N}_k$ such that

$$\begin{aligned}
V_1 \subset V_2 \subset \ldots \subset V_n \\
\dim(V_k) = |\mathcal{N}_k|.
\end{aligned}$$

On the interior coarse cells, this function space is given in a natural way by finite elements. On the boundary coarse grid cells one has to construct the function

space V_k such that the boundary conditions of the fine grid space are preserved. A detailed description of the construction of V_k will be given in a subsequent paper. The coarse spaces V_k lead to natural restriction and prolongation operators R_k and P_k and to coarse grid differential operators L_k. The relation between these operators is

$$L_k = R_k L_{k+1} P_{k+1}.$$

At the grid points $\mathcal{N}_{I,k}$, the restriction and prolongation operators are operators with fixed stencils.

For the implementation of the coarse grid differential operators, we distinguish to cases:

I. Implementation at regular interior points $P \in \mathcal{N}_{I,k}$.

- If the fine grid operator L_h is an operator with constant coefficients, then the corresponding coarse grid differential operator L_k at interior points $P \in \mathcal{N}_{I,k}$ is

  $$L_k(P) = L_{2^{-k}}(P).$$

  Therefore, we do not need additional storage to evaluate this operator.
- If the fine grid operator L_h is an operator with variable coefficients, then one has to restrict the local stiffness matrices and one has to store the local stiffness matrices on all coarse cells. A differential operator can be evaluated using the local coarse grid stiffness matrices.

II. Implementation at points near the boundary $P \in \mathcal{N}_{II,k}$.

- If the fine grid operator L_h is an operator with constant coefficients, then the local stiffness matrices only at the coarse cells $\mathcal{Z}_k^{\text{boundary}}$ have to be stored.
- If the fine grid operator L_h is an operator with variable coefficients, then the local stiffness matrices at all coarse cells $\mathcal{Z}_k^{\text{boundary}} \cup \mathcal{Z}_k^{\text{interior}}$ have to be stored.

To store a local stiffness matrix, one has to store 8×8 values. Since these local stiffness matrices are stored only on coarse grids, storing the local stiffness matrices costs only a small percentage of the total storage.

6 Numerical results.

In this section, we present two kinds of numerical results.

Numerical result 1: Differential operator with variable coefficients. Table 1 shows the computational time for the evaluation of one discrete Laplace operator on a Sun Ultra 1 workstation. t_c is the computational time of the operator with a constant coefficient corresponding to the following bilinear form

$$a(u, v) = \int_\Omega \nabla u \nabla v \; d\mu.$$

t_v is the computational time of the operator with a variable coefficient $\alpha(x, y, z)$ corresponding to the bilinear form

$$a(u, v) = \int_\Omega \nabla u \nabla v \ \alpha(x, y, z) \ d\mu.$$

One can see that the computational time increases only by a factor less than 2 in case of a variable coefficient.

Table 1. Parallel solution of linear elasticity equation.

grid size	t_c in sec	t_v in sec	t_v/t_c in sec
21 247	0.23	0.31	1.33
154 177	1.24	1.98	1.59
1 164 700	8.92	16.1	1.81

Numerical result 2: Multigrid for linear elasticity with traction free (Neumann) boundary conditions. We want to solve the linear elasticity equations with traction free boundary conditions. The solver is a cg-solver with a multigrid algorithm as a preconditioner. The multigrid algorithm is a stable solver even for traction free boundary conditions, since the coarse grid space is constructed in such a way that it contains the rigid body modes. Table 2 shows the computational time for one cg-iteration with multigrid preconditioning with respect to the number of grid points. The number of unknowns is the number of grid points multiplied by 3. The calculations were done on ASCI Pacific Blue.

Table 2. Parallel solution of linear elasticity equation.

processors	time in sec	grid size	number of unknowns
600	120	121 227 509	363 682 527
600	26	15 356 509	46 069 527
88	18	1 973 996	5 921 988
98	3.8	259 609	778 827
88	1.1	35 504	106 512
12	19.6	259 609	778 827
4	8.9	35 504	106 512

Acknowledgment. The author Christoph Pflaum would like to thank the Center for Applied Scientific Computing for supporting his research during his stay at Lawrence Livermore National Laboratory.

References

1. S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*. Springer, New York, Berlin, Heidelberg, 1994.
2. Ch. Großmann and H.-G. Roos. *Numerik partieller Differentialgleichungen*. Teubner, Stuttgart, 1992.
3. W.D. Henshaw. Automatic grid generation. *Acta Numerica*, 5:121–148, 1996.
4. C. Pflaum. Discretization of second order elliptic differential equations on sparse grids. In C. Bandle, J. Bemelmans, M. Chipot, J. Saint Jean Paulin, and I. Shafrir, editors, *Progress in partial differential equations, Pont-á-Mousson 1994, vol. 2, calculus of variations, applications and computations*, Pitman Research Notes in Mathematics Series. Longman, June 1994.
5. C. Pflaum. The maximum angle condition of semi-unstructured grids. In *Proceedings of the Conference: Finite Element Methods, Three-dimensional Problems (Jyväskylä, June 2000)*, Math. Sci. Appl., pages 229–242. GAKUTO Internat. Series, Tokio, 2001. to appear.
6. C. Pflaum. Semi-unstructured grids. *Computing*, 67(2):141–166, 2001.
7. L. Stals, U. Rde, C. Wei, and H. Hellwagner. Data local iterative methods for the efficient solution of partial differential equations. In *In proceedings of the eighth biennial computational techniques and applications conference, Adelaide, Australia*, 1997.
8. J.F. Thompson. *Numerical Grid Generation*. Amsterdam: North–Holland, 1982.
9. C. Yeker and I. Zeid. Automatic three-dimensional finite element mesh generation via modified ray casting. *Int. J. Numer. Methods Eng.*, 38:2573–2601, 1995.
10. M. A. Yerry and M. S. Shephard. Automatic three-dimensional mesh generation by the modified-octree technique. *Int. J. Numer. Methods Eng.*, 20:1965–1990, 1984.