

# Object-oriented implementation of 3D DC adaptive finite-element method

**Journal Article****Author(s):**

Ren, Zhengyong; Tang, Jingtian; Wang, Feiyan; Xiao, Xiao; Liu, Changsheng; Guo, Rongwen

**Publication date:**

2010

**Permanent link:**

<https://doi.org/https://doi.org/10.3929/ethz-b-000422684>

**Rights / license:**

[In Copyright - Non-Commercial Use Permitted](#)

**Originally published in:**

Frontiers of Earth Science in China 4(2), <https://doi.org/10.1007/s11707-009-0065-x>

# Object-oriented implementation of 3D DC adaptive finite-element method

Zhengyong REN (✉)<sup>1,2</sup>, Jingtian TANG<sup>2</sup>, Feiyan WANG<sup>2</sup>, Xiao XIAO<sup>2</sup>, Changsheng LIU<sup>2,4</sup>, Rongwen GUO<sup>2,3</sup>

<sup>1</sup> Institute of Geophysics, ETH Zurich, Zurich 8092, Switzerland

<sup>2</sup> School of Info-physics and Geomatics Engineering, Central South University, Changsha 410083, China

<sup>3</sup> School of Earth and Ocean Sciences, University of Victoria, Victoria 32100, Canada

<sup>4</sup> Changsha Aeronautical Vocational and Technical College, Changsha 410014, China

© Higher Education Press and Springer-Verlag 2009

**Abstract** In this paper, we introduced a clear object-oriented framework to implement the complicated adaptive procedure with C++ programming language. In this framework, it consisted of the unstructured mesh generation, a-posteriori error estimating, adaptive strategy, and the postprocessing. Unlike the procedure-oriented framework, which is commonly used in DC resistivity modeling with FORTRAN language, the object-oriented one, which is famous for its characteristic of encapsulation, could be used for a class of problems that would be executed by only making some changes on the user interface. To validate its flexibility, two synthetic DC examples were tested here.

**Keywords** object-oriented strategy, adaptive finite-element method, C++ framework, unstructured mesh

## 1 Introduction

Adaptive finite element method (AFEM) that obtained better accuracies and efficiencies than finite element method (FEM) has become a focused topic in electromagnetic computations. However, an AFEM procedure for 3D case was more complex than the standard FEM program. Generally, it included a 3D unstructured mesh generation, a-posteriori error estimating, adaptive mesh refinement, and postprocessing procedure, which agreed well with the essence of object-oriented philosophy (OOP). AFEM associated with OOP technique was already studied in potential flow (Akin and Singh, 2002), structural mechanics (Niekamp and Stein, 2002), multi-physics applications (Stewart and Edwards, 2004), electrochemistry (Ludwig and Speiser, 2006), astrophysical

fluid dynamics (Rosenberg et al., 2006), and crack propagation (Phongthanapanich and Dechaumphai, 2004), etc.

However, little attention was paid to its applications in geophysical fields of earth sciences. In current geophysical fields, the FEM procedure (Nguyen and Mardon, 1995; Folch et al., 1999; Haber, 2000; Braun, 2003; Axness et al., 2004) was widely studied and the AFEM approach was still in elementary development. Key and Weiss (2006) and Li and Key (2007) applied the AFEM to 2-D magnetotelluric and controlled-source electromagnetic modeling. However, their codes were written in the procedure-oriented language FORTRAN. The objective of this study was trying to implement the AFEM approach using OOP technique, especially in 3D geophysical DC cases.

The paper was organized as follows. First, the brief mathematical formulas of an AFEM approach for the second-order elliptic problem were introduced. Second, the C++ framework of the AFEM procedures was briefly depicted. Then, several key classes were presented in detail. Last, two geophysical DC examples were tested to validate our framework.

## 2 Elliptic partial differential equations

Generally, many geophysical problems such as seismic, gravity, and magnetic all belonged to a second-order elliptic problem that usually had the following form:

$$\begin{aligned} \nabla(a\nabla u) + bu &= f && \text{in } \Omega, \\ u &= g_1 && \text{on } \partial\Omega_D, \\ \frac{\partial u}{\partial n} &= g_2 && \text{on } \partial\Omega_N, \\ a\frac{\partial u}{\partial n} + ru &= g_3 && \text{on } \partial\Omega_M, \end{aligned} \quad (1)$$

where  $\Omega \in R^3$  was a computational domain;  $u(x,y,z)$  was the unknown;  $a(x,y,z)$ ,  $b(x,y,z)$ ,  $r(x,y,z,a)$ , and  $f(x,y,z)$  were coefficient functions;  $\partial\Omega_D$  was the essential boundary with Dirichlet condition  $g_1$ ;  $\partial\Omega_N$  was the nature boundary with Neumann condition  $g_2$ ;  $\partial\Omega_M$  was the boundary with Mixed condition on which condition functions  $\beta$  and  $g_3$  were defined, and  $\mathbf{n}$  was the outward unit normal on the corresponding boundary  $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N \cup \partial\Omega_M$ .

In FEM approximations (Brenner, 2002), the integral weak form of Eq. (1) was to find a  $u$  which belonged to  $V$  satisfying

$$B(u,v) = F(v), \quad \text{for all } v \in V, \quad (2)$$

where  $B(\cdot, \cdot)$  and  $F(\cdot)$  were the bilinear and linear form, respectively, defined by

$$\begin{aligned} B(u,v) &= \int_{\Omega} (a \nabla u \nabla v - buv) d\Omega + \int_{\partial\Omega_M} ruv dS, \\ f(v) &= - \left( \int_{\Omega} fvd\Omega - \int_{\partial\Omega_N} ag_2 v dS - \int_{\partial\Omega_M} g_3 v dS \right), \end{aligned} \quad (3)$$

where  $U \in H^1(\Omega)|_{\partial\Omega_D} = g_1$ ,  $V \in H^1(\Omega)|_{\partial\Omega_D} = 0$ , and  $H^1(\Omega)$  is the Sobolev space.

By the Galerkin method, the FEM numerical solution was to find  $u_h$ , satisfying the following formula,

$$B(u_h, v_h) = F(v_h), \quad \text{for all } v_h \in V_h, \quad (4)$$

where  $u_h \in U^h$ ,  $v_h \in V^h$ ,  $H_h^1(\Omega)$  is the Sobolev finite-element space.

After solving the large matrix system derived from Eq. (4), we employed a gradient-based posteriori error estimator (Zienkiewicz and Zhu, 1992a, b; Zienkiewicz and Taylor, 2000) to adaptively refine the mesh. We assumed that  $e_K$  represented the  $K$ th element error,  $\nabla u_h$  the numerical gradient, and  $G(u_h)$  the recovered gradient, respectively. Then,  $e_K$  of element  $K$  in  $L_2$  norm could be obtained by

$$e_K = \|G(u_h) - \nabla u_h\|_{L_2(K)}, \quad (5)$$

where  $G(u_h)$  was defined in element patch  $\tilde{K} = K \cup_{i=0}^{m-1} J_i$ :  $\{\partial J_i \cap \partial K \neq \emptyset, a_{J_i} = a_K\}$ ,  $m$  was the element path size, and  $J_i$  was the neighboring element of  $K$ . In 3D case,  $G(u_h)$  was an interpolation polynomial, owning the same order with unstructured linear element

$$G(u_h) = a_0 + b_0x + c_0y + d_0z. \quad (6)$$

What we should note is that the least-square fitting was respectively required to calculate the coefficients in Eq. (6) for each component of the gradients

$$\sum_{s \in S} [G(u_h)(s) - \nabla u_h(s)]^2 \rightarrow \min, \quad (7)$$

where  $s$  was the Gauss integral point in each element belonging to  $\tilde{K}$ .

In the AFEM approach, a global relative error  $\eta_{\text{goal}}$  was specified, which generally could not be satisfied in the initial mesh. Thus, the element goal error was defined by assuming that an optimal mesh there would have an equal error distribution in each element:

$$\tilde{e}_M = \eta_{\text{goal}} (\|G(u_h) - \nabla u_h\|_{L_2(\Omega)}^2 + \|\nabla u_h\|_{L_2(\Omega)}^2)^{1/2} / \sqrt{M}, \quad (8)$$

where  $M$  was the number of elements. Through comparing  $e_K$  with  $\tilde{e}_M$  and using the well-known convergence of FEM approximation  $e \propto h^{\min(1,\lambda)}$ , the new element size for the  $K$ th element in the current mesh can be predicated (Zienkiewicz and Taylor, 2000),

$$\begin{aligned} h_{\text{new}} &= h_{\text{old}} \times \varepsilon^{\frac{1}{\min(1,\lambda)}}, \\ \varepsilon &= \frac{\tilde{e}_M}{e_K}, \end{aligned} \quad (9)$$

where  $h_{\text{old}}$  was the current element size,  $\lambda \in [0.5, 1.0]$  was the strength of singularities such as source point and anomaly bodies. For the detail adaptive refinement process, please refer to Ren and Tang (2009).

### 3 An object-oriented framework

In Fig. 1, the object-oriented framework was displayed by the united model language (UML) of the adaptive FEM procedures. For practical geophysical problems, the only thing that should be done by artificial work was to offer a geometric model with known boundary conditions on the elliptic Eq. (1). After that, the adaptive computation would automatically terminate with outputting the final high accurate numerical solutions  $u_h$ .

#### 3.1 Mesh Class

Mesh was a key component in our framework. It was solely defined by nodes and elements with neighboring relationships, which was shown in Fig. 2(b). Moreover, the Element class was abstracted as the father class of all finite elements. Additionally, the neighboring vector connected all elements in mesh for correctly assembling finite-element equation. In terms of *virtual polymorphism*, general operators on abstract Element class should be transformed into derived finite elements. In the Element class, all common operators were defined as *virtual or pure virtual* functions, which were redefined in son derived classes to perform correct actions. In this study, a linear tetrahedron finite element was employed. The 4-node element owned four faces and six edges, which was abstracted as Tri3 class and Edge2 class, respectively. Unlike the way of being member data in other literatures, Tri3 class and Edge2 class were derived from a father Element class to most possibly reuse codes and keep it

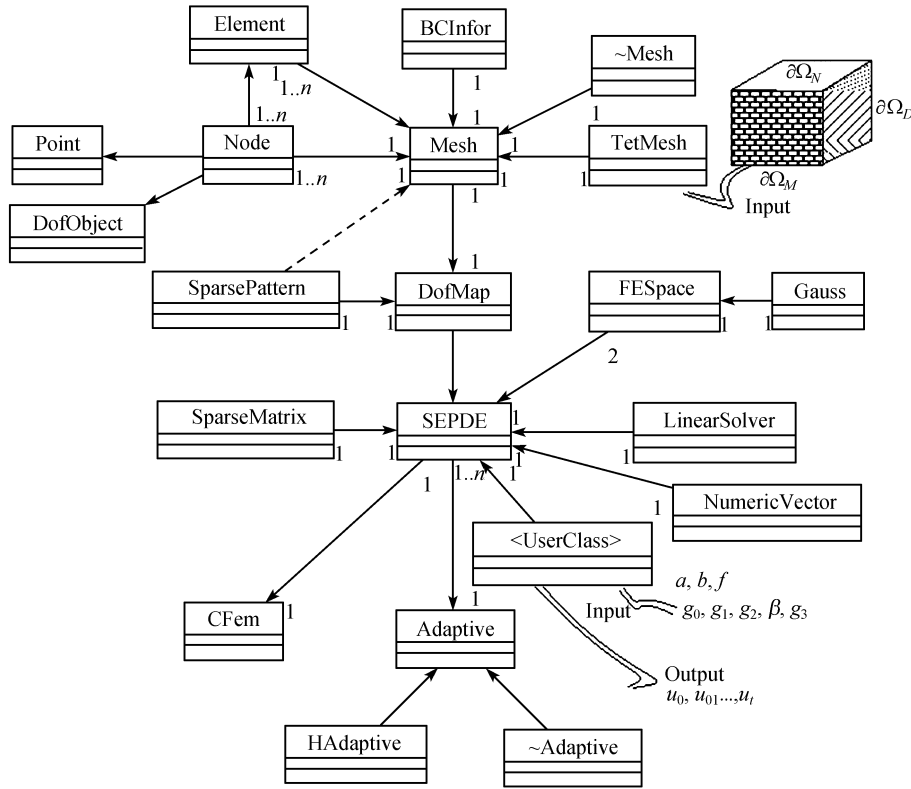


Fig. 1 Object-oriented framework

uniform. The key job of the mesh was to assemble the final sparse large linear equation by looping each element with proper boundary conditions. For the essential boundary condition, a Tri3 class face could be renewed from an element and then applying it to each node of this Tri3 face. To fulfill the data structure of mesh, a mesh generation TetMesh class was defined. At present, TetMesh class could generate unstructured grids by calling open source library TetGen (Si, 2003).

### 3.2 DofMap Class

As it was shown in Fig. 1, a DofMap class was defined to manage the unknowns on a mesh, which contained a renumbering of unknowns and mappings of element indices. After a mesh and a PDE system were given, the DofMap class could prepare the unknown indices for assembly. Here, a DofObject class was introduced on each node associated with the DofMap class. In this class, an INT number was used to identify the index of unknowns on each node. After DofObject class was initialized, all unknowns would be renumbered by the outstanding RCM algorithm. In the DofMap class, another interior SparsePattern class was presented in which a `std::vector<std::vector<unsigned int>>` semimatrix was used to store the sparse pattern of unknowns on a mesh. The size of this matrix was the number of unknowns, and the size of each

row  $i$  was the number of unknowns connected with the  $i$ th unknown. For an unknown pair  $\langle i, j \rangle$  on a given element  $e$ , the index  $j$  was pushed back in the  $i$ th row. Once this linear time complexity ( $O(m)$ ,  $m$  was the number of elements) process was done on all elements, the sparse pattern of a mesh was immediately available. Since huge temporary storage allocation of sparse matrix would seriously decrease the performances of linear solver defined in LinearSolver class, in the SparsePattern class, a function was defined, which used this computed sparse pattern to preallocate the memory for SparseMatrix class to fix it.

### 3.3 FESpace Class

Being separated from Mesh class, all operations and data associated with finite element space were abstracted into the FESpace class. To cooperate with the general framework, the principle was designed by surrounding the element type, which required all necessary data of finite-element space that were prepared when finite element encountered. However, the geometrical shape of any finite element was so irregular that it was hard to design this general framework on the real element. Fortunately, the Isoparametric element projection mapping could easily fix it. In the mapping process shown in Fig. 2(c), the real element  $R(x)$  was transformed into a standard reference element  $S(x')$  by a projecting  $F$ , which was denoted by

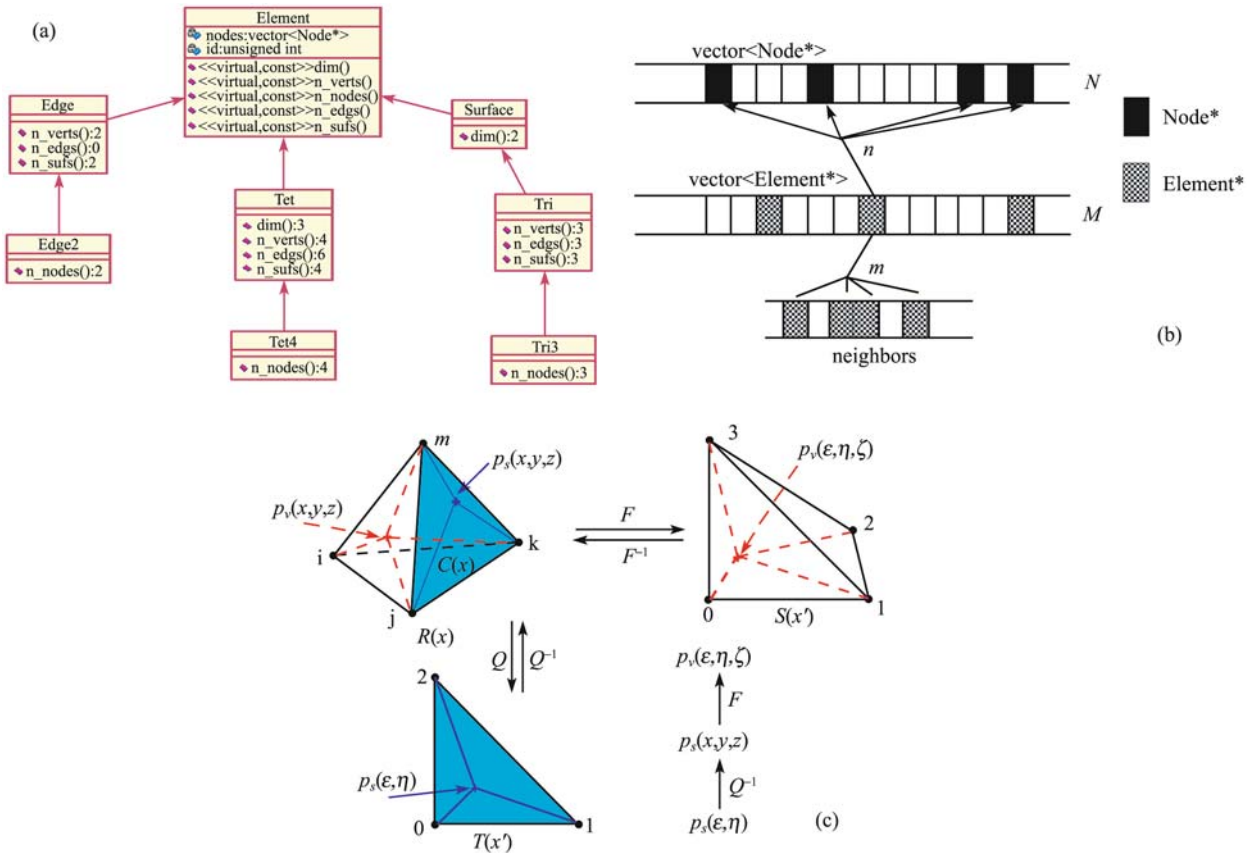


Fig. 2 Element class. (a) Class hierarchical structure; (b) underlying data structures; (c) isoperimetric unstructured element projecting

$$R(x) \xrightleftharpoons[F^{-1}]{F} S(x'), \quad (10)$$

where  $F^{-1}$  was the inverse projecting from  $S(x')$  to  $R(x)$ . And  $x, x'$  could be solely mapped in  $S(x')$  and  $R(x)$ , respectively. The volume integral term derived from Eq. (4) could be easily implemented on the reference element  $S(x')$ . In addition, by means of Gauss integral rule (Zienkiewicz and Taylor, 2000), the expression of volume term  $V(\varphi)$  could be deduced as

$$V(\varphi) = \frac{1}{V_s} V(\varphi)_{q_v} \times W_{q_v} \times \det|J_V|_{q_v}, \quad (11)$$

where  $q_v$  was the volume integral points in  $S(x')$ ,  $V_s$  was the volume of unit reference element  $S(x')$ ,  $W_{q_v}$  was the numerical weights at points  $q_v$ ,  $J_V$  denoted the Jacobin transform matrix, which could easily be calculated by the projecting map  $F$  (Zienkiewicz and Taylor, 2000).

With regard to surface integrals, they owned the similar expression,

$$S(\varphi) = \frac{1}{A_s} S(\varphi)_{q_s} \times W_{q_s} \times \det|J_s|_{q_s}, \quad (12)$$

where  $A_s$  was the area of an unit 2D reference element  $T(x')$ ,  $q_s$  was the surface integral points in  $T(x')$ ,  $W_{q_s}$  was

the numerical weights at points  $q_s$ , and  $J_s$  was the Jacobin transform matrix that projected a 3D surface of  $R(x)$  to a unit 2D reference element  $T(x')$  in which only two volume coordinates  $(\epsilon, \eta)$  were involved. It could be realized that shape functions related term  $S(\varphi(\epsilon, \eta, \zeta))$  at the surface integral point  $q_s$  could be calculated by its corresponding local coordinates  $(\epsilon, \eta)$ . Therefore, the  $(x, y, z)$  coordinates of surface integral points  $q_s$  could be first calculated by the surface projecting map  $T$ . Moreover, by the 3D inverse volume projecting  $F^{-1}$ , the local  $(\epsilon, \eta, \zeta)$  coordinate of  $q_s$  could be accurately estimated by Newton iterative method (Brenner and Scott, 2002) so that the value of  $S(\varphi)_{q_s}$  could be calculated.

In our implementation, the surface integrals were treated as the volume integrals that were incorporated in FESpace class since the only difference between the two kinds of integrals was that surface integrals were to calculate the  $(x, y, z)$  coordinates of surface integral points by the 2D surface projecting. If the values of  $S(\varphi)$  in the reference element  $S(x')$  were computed, the remaining surface integral weights and corresponding determinant of the 2D surface projecting Jacobian matrix were multiplied into  $S(\varphi)_{q_s}$  so that the surface integral terms  $S(\varphi)$  were easily available. To avoid being too much memory consuming, the values of shape functions were preallocated, its

gradients and also the numerical integral points and weights by storing them as the fast indexed caches, since they were independent of the practical geometrical shape of element type. This way could dramatically accelerate the computational speed and decreased the needs of memory. Second, this independent design of FESpace could offer us a way of only involving in two FESpace objects in the finite-element assembling process, with one for the volume integrals and another for surface integrals.

### 3.4 SEPDE Class

An abstract SEPDE class was defined for a second-order elliptic partial differential equation (SEPDE). Two maps were involved to fulfill the coefficient functions of Eq. (1), which were shown in Fig. 3. The first `std::map<ID, BCType>` projected the boundary type `BCType` (such as Neumann) from boundary `ID`. The second one `std::map<ID, BCFunction [2]>` was used to map the boundary function pointers `BCFunction` from the boundary `ID`. In most engineering cases, the source coefficient function  $f$  such as the Delta function was not smooth enough for the Gauss integral rule. Therefore, an additional routine was offered for users to assemble the right-hand side vector  $\mathbf{B}$ . There were also several classes such as `SparseMatrix` class for global matrix  $\mathbf{A}$  and `NumVector` class for RHS vector  $\mathbf{B}$  and solution vector  $\mathbf{X}$ . To solve the linear equation  $\mathbf{AX} = \mathbf{B}$ , the `LinearSolver` class was defined based on the `LASPack` (Scalicky, 1996) package that offered some Krylov subspace iterative methods such as CG-type ones.

std::map<ID, BCType>		std::map<ID, BCFunction[2]>		
key	value	key	value	
1	Neumann	1	$g_1$	
2	Dirichlet	2	$g_2$	
3	Mixed	3	$r$	$g_3$

(a)
(b)

Fig. 3 Map of boundary ID and conditions

### 3.5 Adaptive Class

A CFEM class was designed to implement the conventional finite-element procedure and an AFEM class was proposed for the adaptive process. From the AFEM class, h-version adaptive process class HAFEM was derived. The HAFEM class was shown on the top level, which contained the `Mesh` class, `DofMap` class, and `SEPDE` class. In the HAFEM class, a `SuperPR` class that adapted the gradient-recovery-based a-posteriori error estimator was adopted to drive the whole adaptive algorithm. To terminate the adaptive process, two flags were set, which were respectively the max iterations  $T_{\max} = 5$  and the

percent goal error threshold  $\eta_{goal} = 10\%$ . The key difference between the adaptive process and the traditional one was that no matter how complex the models were, the AFEM method could also offer us a way that the numerical solution would asymptotically converge to the exact solutions. For example, in a 3D linear element case, the convergent ratio of CFEM algorithm was less than 1/6 in contrast to a value of 1/3 of AFEM algorithm. The superior performances of adaptive process were shown in the following tests.

## 4 DC examples

### 4.1 Model 1

Model 1 was a conducting cube buried in earth, as shown in Fig. 4. The solution domain was  $1000 \text{ m} \times 1000 \text{ m} \times 1000 \text{ m}$  with positive z-axis down. A 10 m-length dipole-dipole measuring line was located along x-axis. A DC class derived from SEPDE class was defined to finish our jobs. By using the structure of our framework shown in Fig. 1, the inputted boundary conditions and flags could be expressed as follows:

$$\begin{aligned}
 b &= 0, \\
 f &= -\delta(\mathbf{p} - \mathbf{p}_A) + \delta(\mathbf{p} - \mathbf{p}_B), \\
 g_2 &= 0 \text{ on } \partial\Omega_N \text{ (with id = 1),} \\
 r &= a \left( \frac{\cos(\mathbf{r}_{pA}, \mathbf{n}) |\mathbf{r}_{pB}|}{|\mathbf{r}_{pB}| - |\mathbf{r}_{pA}|} - \frac{\cos(\mathbf{r}_{pB}, \mathbf{n}) |\mathbf{r}_{pA}|}{|\mathbf{r}_{pB}| - |\mathbf{r}_{pA}|} \right), \\
 g_3 &= 0 \text{ on } \partial\Omega_M \text{ (with id = 3),}
 \end{aligned} \tag{13}$$

where  $\mathbf{p}_A$  and  $\mathbf{p}_B$  was respectively positive and negative source point,  $\mathbf{p}$  was any potential point in the model, and  $a$  denoted the conductivity of model that had a value of 0.05 S/m in the cube and a value of 0.01 S/m in the homogenous half-space.  $\mathbf{r}_{pA}$  was a vector connecting  $\mathbf{p}_A$  and  $\mathbf{p}$ . Correspondingly,  $\mathbf{r}_{pB}$  denoted a vector connecting  $\mathbf{p}_B$  and  $\mathbf{p}$ , and  $\mathbf{n}$  was an outer unit vector on  $\partial\Omega_M$ .

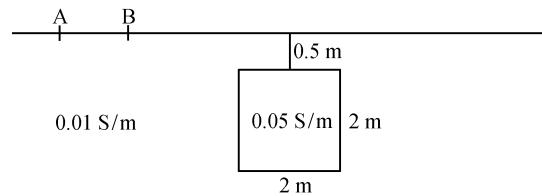
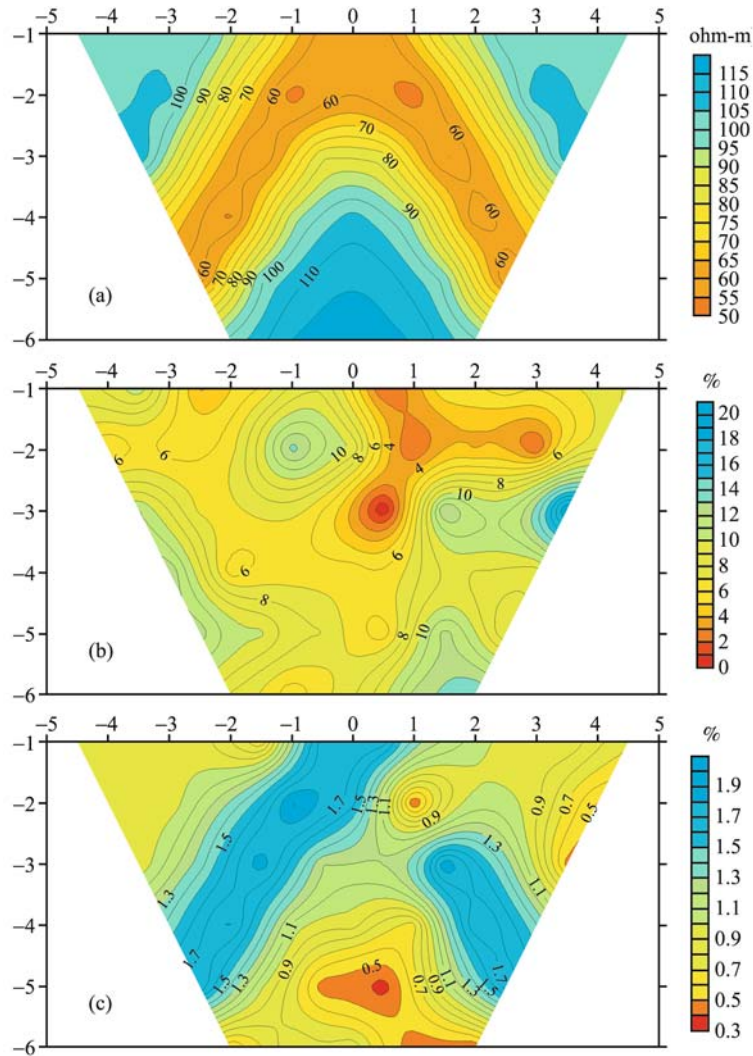


Fig. 4 Cube model

In Fig. 5, the relative errors computed using the singularity removal technique proposed by Wu (2002) with finite element as reference are displayed. Our algorithm has been dramatically validated by the decreased



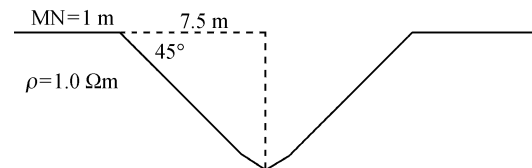
**Fig. 5** Comparison of singularity-removal method (Wu, 2003) and our adaptive finite-element method. (a) Apparent resistivities from the final adaptive iteration (third mesh); (b) relative errors on the first mesh; (c) relative errors on the third mesh

relative errors produced from final third generated mesh depicted in Fig. 5(c). In the first mesh with 5980 nodes and 15093 tetrahedrons in which the relative errors are displayed in Fig. 5(b), the average relative error has a value more than 15%; after two adaptive iterations, on the third mesh with 40902 nodes and 20292 tetrahedrons, it is dramatically decreased to a value of less than 1.0% by which the superior properties of adaptive process are presented clearly.

#### 4.2 Model 2

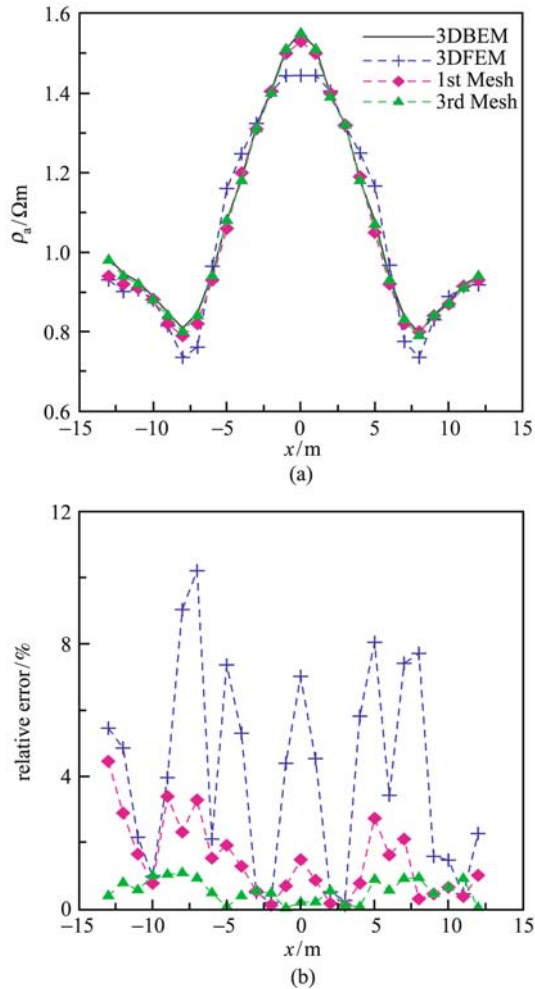
A valley model was tested, which was shown in Fig. 6. The solution domain was  $2000\text{ m} \times 2000\text{ m} \times 2000\text{ m}$  with positive z-axis down. Two current source points A and B were located with two measuring electrodes M and N moving in the range of  $AB/3$ . All inputted boundary conditions and flags were as same as those in Model 1.

Computational results are compared with both 3D



**Fig. 6** Valley model

boundary element method (3DBEM) proposed by Xu and Zhao (1985) and 3D finite-element method (3DFEM) using structured mesh introduced by Qiang and Luo (2007). In Fig. 7(a), it is clear to see that our results are much closer to that obtained from 3D boundary element method. By contrast, it produces big deviations from Qiang and Luo's results (3DFEM). The relative errors using the Xu and Zhao's results as the reference are presented in Fig. 7(b). From this figure, it can be seen that large instability exists in the Qiang and Luo's results, which may



**Fig. 7** (a) Apparent resistivities obtained from 3D FEM (Qiang and Luo, 2007), 3D BEM (Xu and Zhao, 1985), the first mesh and third mesh of our adaptive algorithm; (b) relative errors of apparent resistivities that are calculated using the results of 3D BEM (Xu and Zhao, 1985) as reference

be caused by the not enough approximation to the surface boundaries. Our algorithm based on unstructured tetrahedrons can flexibly fix this problem. Even at the initial mesh with 7830 nodes, more stable results with an average relative error of 1.45% are obtained. Moreover, in the final mesh (the 3rd adaptively refined mesh) with 49392 nodes, the average relative error is dramatically reduced to 0.55%. This convergent phenomenon of errors not only offers us more accurate results on the final mesh but also proves the ability of dealing with complicated problems in our algorithm.

## 5 Conclusions

In this study, the powerful properties of OOP method in designing complex adaptive finite-element procedures are shown. The OOP technique can divide the whole complex

adaptive process into several rather separated modules by which modification is easily done to deal with more problems. These advantages offer us an easy way to set up a general framework for the common elliptic differential equation. Practical engineering problems can also be easily deduced with a few extra works.

Two 3D DC resistivity examples have shown these excellent performances through being compared with other techniques. From these results, it is obvious to see that oriented-object-based adaptive finite element method can solve the DC problems with remarkably high accuracy and efficiency. These are really essential for large-scaled forward modeling problem that is the basis for the inversion problem.

**Acknowledgements** This project was supported by grants from the National Natural Science Foundation of China (Grant No. 40874072), Technology Development Program of Hunan, China (No. 2008FJ4181, 2008), Scientific Research Project of the Education Department of Hunan Province (No. 08D008, 2008), and China Scholarship Council (CSC) 2007 and 2008 for financial supports. The first author wanted to give great thanks to Yan YAN (Central South University, China) for valuable discussion on finite-element methodology and dealing with coding difficulties during two and half years.

## References

- Akin J E, Singh M (2002). Object-oriented Fortran 90 P-adaptive finite element method. *Advances in Engineering. Software*, 33, (7–10): 461–468
- Anxness C, Carrera J, Bayer M (2004). Finite-element formulation for solving the hydrodynamic flow equation under radial flow conditions. *Computers & Geosciences*, 30(6): 663–670
- Braun J (2003). Pecube: a new finite-element code to solve the 3D heat transport equation including the effects of a time-varying, finite amplitude surface topography. *Computers & Geosciences*, 29(6): 787–794
- Brenner S C, Scott L R (2002). *The Mathematical Theory of Finite Element Methods*. Berlin: Springer
- Folch A, Vázquez M, Codina R, Martí J. (1999). A fractional-step finite-element method for the Navier-Stokes equations applied to magma-chamber withdrawal. *Computers & Geosciences*, 25(3): 263–275
- Haber E (2000). A mixed finite element method for the solution of the magnetostatic problem with highly discontinuous coefficients in 3D. *Computational Geosciences*, 4(4): 323–336
- Key K, Weiss C (2006). Adaptive finite-element modeling using unstructured grids: The 2D magnetotelluric example. *Geophysics*, 71 (6): G291–G299
- Li Y G, Key K (2007). 2D marine controlled-source electromagnetic modeling: Part 1- An adaptive finite-element algorithm. *Geophysics*, 72(2): WA51–WA62
- Ludwig K, Speiser B (2006). EChem ++ —An object-oriented problem solving environment for electrochemistry: Part 4. Adaptive multi-level finite elements applied to electrochemical models Algorithm and benchmark calculations. *Journal of Electroanalytical Chemistry*,

- 588(1): 74–87
- Nguyen S H, Mardon D (1995). A p-version finite-element formulation for modeling magnetic resonance relaxation in porous media. *Computers & Geosciences*, 21(1): 51–60
- Niekamp R, Stein E (2002). An object-oriented approach for parallel two- and three-dimensional adaptive finite element computations. *Computers & Structures*, 80(3–4): 317–328
- Phongthanapanich S, Dechaumphai P (2004). Adaptive Delaunay triangulation with object-oriented programming for crack propagation analysis. *Finite Elements in Analysis and Design*, 40(13–14): 1753–1771
- Qiang J K, Luo Y Z (2007). The resistivity FEM numerical modeling on 3-D undulating topography. *Chinese J Geophys*, 50(5): 1606–1613. (in Chinese with English abstract)
- Ren Z Y, Tang J T (2009). 3D direct current resistivity modeling with an unstructured mesh by an adaptive finite-element method. *Geophysics* (in press)
- Rosenberg D, Fournier A, Fischer P, Pouquet A (2006). Geophysical-astrophysical spectral-element adaptive refinement (GASpAR): Object-oriented h-adaptive fluid dynamics simulation. *Journal of Computational Physics*, 215(1): 59–80
- Scalicky T (1996). *LASPack Reference Manual*. Dresden: Dresden University of Technology, 39, <http://www.netlib.org>, accessed July 26, 2006
- Si H (2003). Tetgen: a quality tetrahedral mesh generator and 3D delaunay triangulation, <http://tetgen.berlios.de>, accessed July 01, 2006
- Stewart J R, Edwards H C (2004). A framework approach for developing parallel adaptive multiphysics applications. *Finite Elements in Analysis and Design*, 40(12): 1599–1617
- Wu X P (2003). A 3-D finite-element algorithm for DC resistivity modeling using the shifted incomplete Cholesky conjugate gradient method. *Geophys J Int*, 154: 947–956
- Xu S Z, Zhao S K (1985). The boundary element method calculating electric field of a point source on three-dimension topography. *Journal of Guilin College of Geology*, 5 (2): 163–168 (in Chinese with English abstract)
- Zienkiewicz O C, Zhu J Z (1992a). The superconvergent patch recovery and a posteriori error estimates. Part 1: the recovery technique. *Int j numer. methods eng*, 33(7): 1331–1364
- Zienkiewicz O C, Zhu J Z (1992b). The superconvergent patch recovery and a posteriori error estimates. Part 2: error estimates and adaptivity. *Int j numer. methods eng*, 33(7): 1365–1382
- Zienkiewicz O C, Taylor R L (2000). *The Finite-element Method (fifth edition) Volume I: The basic*. Woburn MA: Butterworth-Heinemann