INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2009; **60**:55–78
Published online 24 July 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1878

A particle swarm optimization approach for hexahedral mesh smoothing

A. Egemen Yilmaz[1,*,†,‡] and Mustafa Kuzuoglu[2,§]

[1]*Havelsan A.S. Eskisehir Yolu 7.km. Mustafa Kemal Mah. Ankara 06530, Turkey*
[2]*Department of Electrical and Electronics Engineering, Middle East Technical University, Ankara 06531, Turkey*

SUMMARY

There are some approaches for all-hexahedral mesh quality improvement by means of node-movement while preserving the connectivity. Among these methods, the most easily implemented and well known one is the Laplacian smoothing method; however, for this method mesh quality improvement is not guaranteed in all cases, and this approach might cause inverted elements especially in concave regions. In this work, a method for the improvement of hexahedral mesh shape-quality without causing inverted elements is proposed; which is based on optimization of an objective function calculated by means of the individual qualities of hexahedral elements in the mesh. The shape-quality for each hexahedral element is defined via the condition number of the relevant element. The numerical optimization scheme is the particle swarm optimization method, which originated from observations related to the social behaviors of bird, insect, or fish colonies. The purpose of this paper is to discuss the applicability of this approach to mesh smoothing. Some examples are given in order to demonstrate the applicability. Copyright © 2008 John Wiley & Sons, Ltd.

Received 2 August 2007; Revised 2 June 2008; Accepted 3 June 2008

KEY WORDS: mesh smoothing; hexahedral mesh; condition number; particle swarm optimization; mesh quality; finite element method

1. INTRODUCTION

In recent years, several algorithms have been developed for all- or dominant-hexahedral mesh generation [1–5]. For these algorithms, high quality of the resultant mesh is not always guaranteed; and skewed and/or inverted elements are likely. On the other hand, mesh quality is known to be

*Correspondence to: A. Egemen Yilmaz, Havelsan A.S. Eskisehir Yolu 7.km. Mustafa Kemal Mah. Ankara 06530, Turkey.
†E-mail: asimegemenyilmaz@yahoo.com, aeyilmaz@havelsan.com.tr
‡SW Engineer.
§Professor.

critical for the accuracy in finite element method (FEM) solutions. Hence, mesh quality improvement can be crucial in practice. Quality improvement methods can be classified as node movement (including smoothing and optimization), node swapping, insertion, and topological cleanup [6].

Smoothing can be defined as repositioning of the nodes in a mesh without changing the connectivity. There exist multiple ways of smoothing: Laplacian smoothing and its variations [7, 8], optimization-based smoothing [9–12], physics-based smoothing [13, 14], and hybridization of different techniques. The optimization-based unstructured mesh smoothing methods are preferred over the Laplacian methods, since they do not create inverted elements. On the other hand, generally they are computationally more expensive. Sometimes, combining these methods may result in efficient and fast quality improvement tools [6].

Much research has been conducted, and many papers have been published in the field of unstructured mesh smoothing and optimization [7, 9–12]. Among these papers, to the authors' knowledge, [15–20] are examples which dealt with unstructured hexahedral mesh smoothing. The method described in [15] is based on maximizing a variant of the scaled-Jacobian metric. This approach does not guarantee that the improved mesh will consist only of untangled elements; however the method in [16] achieves this goal. The aim of this paper is to propose another method, which is as successful as [16].

2. HEXAHEDRAL ELEMENT QUALITY METRIC

Invertibility, size, and shape are the three important factors of the element quality:

- If an element has positive local volume (everywhere in the element, not just at the eight corners), then it is considered to be *invertible*.
- For a mesh without any inverted elements, element *size* becomes the factor under consideration. An element must be small enough to have small discretization errors, and on the other hand, it should be sufficiently large such that available computer resources (CPU time, memory) are efficiently used.
- The last metric is element *shape*, which is a function of element aspect ratios and skew [21]. Skew gives information about the angles within an element regardless of the aspect ratio. Accuracy decreases if an element contains very large and small angles (i.e. close to 0 or 180°). In [16], Knupp showed that the shape can be expressed as a function of the *condition number*, which can be improved by means of:

  (i) aspect ratio improvement, and/or
  (ii) element skew reduction.

Definitions about element quality are directly taken from [16]. For clarification, these definitions are repeated in this section. The eight nodes of a hexahedral element ($k=0,1,\ldots,7$) can be numbered such that the nodes 0, 1, 2, 3 are at the bottom, and the nodes 4, 5, 6, 7 are at the top surface; this numbering scheme and the node coordinates of hexahedral element transformed to the $\xi\eta\zeta$-space are listed in Table I.

Let $\mathbf{x}$ be the position vector of one of the eight nodes; and $\mathbf{x}_1, \mathbf{x}_2$, and $\mathbf{x}_3$ be the coordinates of the three neighbor nodes. Their order for proper orientation is again listed in Table I. Three edge vectors $\mathbf{e}_1=\mathbf{x}_1-\mathbf{x}, \mathbf{e}_2=\mathbf{x}_2-\mathbf{x}, \mathbf{e}_3=\mathbf{x}_3-\mathbf{x}$ and the matrix $\mathbf{A}$ can be formed from the three column vectors, $\mathbf{A}=[\mathbf{e}_1|\mathbf{e}_2|\mathbf{e}_3]$. This means that if the vector $\mathbf{x}$ is (x, y, z); and for $i=1,2,3$ if the vectors

DOI: 10.1002/fld

Table I. Nodal ordering for a hexahedral element.

k	(ξ, η, ζ)	x_1	x_2	x_3
0	(0, 0, 0)	1	3	4
1	(1, 0, 0)	2	0	5
2	(1, 1, 0)	3	1	6
3	(0, 1, 0)	0	2	7
4	(0, 0, 1)	7	5	0
5	(1, 0, 1)	4	6	1
6	(1, 1, 1)	5	7	2
7	(0, 1, 1)	6	4	3

$\mathbf{x}_i$ are (x_i, y_i, z_i), then $\mathbf{A}$ can be written as follows:

$$\mathbf{A} = \begin{bmatrix} x_1 - x & x_2 - x & x_3 - x \\ y_1 - y & y_2 - y & y_3 - y \\ z_1 - z & z_2 - z & z_3 - z \end{bmatrix} \tag{1}$$

For each node of the hexahedron, eight such matrices, $\mathbf{A}_k$ can be constructed. It is assumed that the element is untangled; namely $\det(\mathbf{A}_k) \geqslant 0$ for all k. If there is a tangled element inside the mesh, then prior to the optimization it should be untangled. For this purpose, the method described in [22], or the one described in [23] (if applied to 3D) can be used.

Knupp also has defined weight matrices $\mathbf{W}_k$ in order to specify the ideal element shape in [21]. By means of $\mathbf{A}_k$ and $\mathbf{W}_k$, the matrices $\mathbf{T}_k = \mathbf{A}_k \mathbf{W}_k^{-1}$ are formed and used in order to represent the shape metrics. The shape metric is defined as follows: the matrices $\mathbf{T}_k$ resemble an orthogonal matrix if the objective function gets optimized. In case that the element becomes an ideal element (most probably with a different orientation), $\mathbf{T}_k$ becomes orthogonal, and eventually $\mathbf{A}_k = \mathbf{T}_k \mathbf{W}_k$. For hexahedra, the ideal shape is a cube; and the weight matrix is the identity matrix.

Let the condition number be $\kappa(\mathbf{T}) = |\mathbf{T}||\mathbf{T}^{-1}|$, where the matrix norm is the Frobenius norm. $f = 8/\sum_k (\kappa(\mathbf{T}_k)/3)^2$ is an algebraic shape metric for hexahedral elements. The proof is given in [16]. With this definition, f is a non-simplicial algebraic shape metric since it has the following properties:

- the domain of f is the set of matrices $\mathbf{T}_k = \mathbf{A}_k \mathbf{W}_k^{-1}$, $k = 0, 1, \ldots, K-1$, with $\det(\mathbf{T}_k) \geqslant 0$;
- f is size invariant;
- f is orientation invariant;
- For all $\mathbf{T}_k$, $0 \leqslant f(\{\mathbf{T}_k\}) \leqslant 1$;
- $f(\{\mathbf{T}_k\}) = 1$ if and only if $\mathbf{T}_k$ is a scalar multiple of an orthogonal matrix for all k;
- $f(\{\mathbf{T}_k\}) = 0$ if and only if $\det(\mathbf{T}_k) = 0$ for some k; i.e. the three edges having a common node are coplanar;
- It should be noted that for tangled elements, shape is not defined.

(The notation $f(\{\mathbf{T}_k\})$ stands for f being a function of all $\mathbf{T}_k$ matrices.)

It is obvious that other functions satisfying the definition of a shape metric can also be defined and used in order to define an objective function for element quality improvement.

DOI: 10.1002/fld

3. THE OBJECTIVE FUNCTION

Based on the shape quality metric mentioned in the previous section, an objective function is described in this section. Again, the definition is reused from [16]. The objective function considers the shape quality of all elements in a hexahedral mesh. Let $\mathbf{T}_{n,k}$ be the matrix corresponding to the kth node of the nth hexahedral element ε_n. We can define

$$f_n = f(\varepsilon_n) = \frac{1}{1/8\sum_k (\kappa(\mathbf{T}_{n,k})/3)^2} \tag{2}$$

as the quality metric of the nth hexahedral element in the mesh (where $n = 1, \ldots, N$).

For the definition of the objective function, it is better to work with $1/f_n$ instead of f_n because it provides a greater numerical range and a steeper gradient; as well as a metric creating a barrier. The objective function is the sum over all elements (ε_n's) inside the hexahedral mesh (ε)

$$F = \frac{1}{N}\sum_n (1/f_n) - 1 = \frac{1}{8N}\sum_n \sum_k (\kappa(\mathbf{T}_{n,k})/3)^2 - 1 \tag{3}$$

This is nothing but the sum of the squares of the element condition numbers. The objective function is scaled so that the minimum value of F is 0.

4. PARTICLE SWARM OPTIMIZATION

The particle swarm optimization (PSO) method is an effective optimization algorithm, which has been applied successfully to some difficult multidimensional continuous/discontinuous problems in various fields so far [24]. Moreover, this technique has been shown to be outperforming other optimization methods such as genetic algorithms (GAs) [25–30].

PSO, which has been developed in 1995 by Kennedy and Eberhart [31], can be described through its leading example: Assume that there is a swarm of bees whose main aim is to find the location with the highest density of flowers in a field without any a priori knowledge; starting at random locations with random velocities. Each bee can remember its previsited successful locations (cognitive behavior), and also it can feel the best locations found by the swarm (social behavior). When a bee finds a better place than previously found places, then it would have tendency to go to this new location in addition to the best location found by the swarm. Eventually, the whole swarm would be attracted towards that location.

Each member of the swarm is referred to as a *particle*; which corresponds to a solution candidate. All the particles accelerate toward the best personal and best overall location; meanwhile they continuously check the value of their current locations.

Each member of the swarm remembers the best location of own discovery. This location is called the personal best or p_{best}. On the other hand, each member feels the best location discovered by the swarm. This is called the global best or g_{best} of the swarm.

The necessary steps for the PSO algorithm are as follows: After the definition of the fitness/objective function; and the definition of the solution space; the particles (i.e. locations and velocities) in the swarm are initialized. Then the particles are moved inside the solution space. For each particle, the fitness is evaluated at the relevant particle's location. If this value is greater than the value calculated at p_{best} of the relevant particle, or the global g_{best} of the swarm, then these values are updated.

DOI: 10.1002/fld

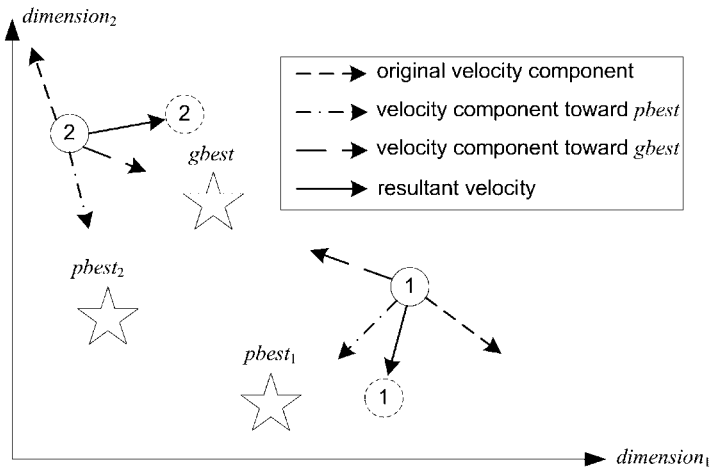

Figure 1. Pictorial description of PSO in 2D.

The velocity manipulation is the main key to convergence in PSO. Locations of p_{best} and g_{best} are major factors for the new velocity value of a particle during this step. A particle gets accelerated in the directions of p_{best} and g_{best} as follows:

$$v_n = w \cdot v_n + c_1 \cdot u_1 \cdot (p_{\text{best},n} - x_n) + c_2 \cdot u_2 \cdot (g_{\text{best},n} - x_n) \tag{4}$$

where x_n is the particle's coordinate in the nth dimension and v_n is the velocity of the particle in the nth dimension. This operation is performed at each dimension in an N-dimensional problem. A pictorial description in two-dimensions can be seen in Figure 1. It can clearly be seen in this equation that the new velocity is the summation of the current velocity scaled by w and increased in the direction of g_{best} and p_{best} for that dimension.

c_1 and c_2 are scaling factors representing the attraction powers of p_{best} and g_{best}. c_1 is a factor showing the memory/history influence on a particle's movement (i.e. a metric of cognitivity), and c_2 is a factor showing the swarm's influence on a particle's movement (i.e. a metric of sociality). Increasing c_1 increases a particle's tendency to its own p_{best}; whereas increasing c_2 increases a particle's tendency to the assumed global maximum.

u_1 and u_2 are random numbers between 0.0 and 1.0 obeying uniform distribution. In most PSO implementations, two independent random numbers are used in order to control the attraction powers of g_{best} and p_{best}. The main reason for this is to add a flavor of unpredictability to the behavior of the swarm. w is known as the inertial weight, and this number (chosen to be between 0.0 and 1.0) determines how much the particle remains along its original direction regardless of the g_{best} and p_{best} attraction. This is a factor adding diversity, and setting up a balance between exploration and exploitation. Detailed discussions about the ideal choices of c_1, c_2 and w can be found in [32].

After the velocity has been calculated, the movement of the particle is straightforward. The velocity is applied during a given time-step, which is usually chosen to be unity; and the new coordinate is calculated at each dimension as follows:

$$x_n = x_n + \Delta t . v_n \tag{5}$$

DOI: 10.1002/fld

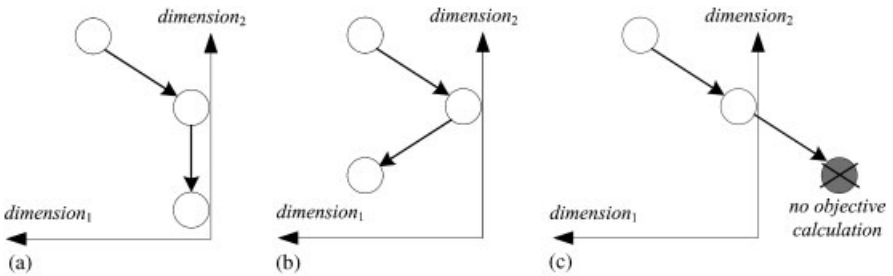

Figure 2. Boundary condition/wall concept in PSO: (a) absorbing walls; (b) reflecting walls; and (c) invisible walls.

After these operations are completed for all particles in the swarm, the whole movement and fitness evaluation process is repeated. Hence, the particles are moved for discrete time intervals as if their snapshots are taken at the end of each time-interval. This is carried on until the termination criterion (criteria) is (are) met. There might be several termination criteria, such as maximum iteration number, achievement of target fitness, saturation in improvement of g_{best}, etc.

In most applications, it is usually desired to put constraints on the search domain. Due to the movements of the particles, there is always a possibility that particles fall outside the solution space during the iterations. In order to prevent/avoid this problem, three different boundary conditions can be imposed [32] as seen in Figure 2:

(1) If a particle exceeds the boundary of the solution space at one dimension, the velocity in that dimension is set to zero; and the relevant particle is implicitly pulled back toward the allowed solution space. This case can be considered as an *absorbing boundary condition*.
(2) If a particle exceeds the boundary of the solution space at one dimension, the velocity is reversed in that dimension; and hence the particle is directly reflected back; which can be considered as a *reflecting boundary condition*.
(3) Without any constraints, the particles are moved to everywhere; but for a, particle falling outside, fitness is not evaluated; which is interpreted as an *invisible boundary condition*.

5. ADAPTATION OF PSO AND DERIVATIVES TO MESH SMOOTHING

Adaptation of PSO to the mesh smoothing will fall into the optimization-based smoothing category. Certainly, optimization-based smoothing methods are computationally expensive compared to methods like Laplacian methods; on the other hand, for assured success they do not have any restrictions (such as the convexity of the region to be smoothed) like Laplacian methods. The main motivations for the usage of PSO in this work can be summarized as follows:

- First of all, PSO is a young but promising, flexible, easy-to-implement global optimization algorithm which is suitable for multidimensional continuous optimization problems by its definition. Due to these facts, it is appropriate for the mesh smoothing problems.
- The method is open to modifications, variations, and hybridizations for performance improvement purposes.

DOI: 10.1002/fld

- Unlike some other optimization algorithms, it does not require any a priori information about the gradient of the objective function. Evaluation of the objective function at a given point is sufficient for the implementation.
- So far, it has been demonstrated that PSO outperforms (in terms of accuracy, convergence speed, CPU and memory requirements) most of the nature-inspired optimization algorithms [25–30].
- By its well-known rapid convergence feature, it can provide quick results from highly distorted mesh. Moreover, unlike most mesh smoothing techniques, it provides global optimization rather than yielding local extrema; which are highly probable to be encountered in the mesh smoothing problems.
- Since it is a population-based search method, PSO provides not only the best solution, but also a large set of good solutions. There might be some applications for mesh smoothing, where one would like to get advantage of this property. For example, with a set of different meshes for the same geometry, it might be possible to investigate the effect of mesh shape (especially some particular details of the mesh) on the solution accuracy.
- The composite nature of PSO makes it especially conducive to implementation on parallel processors.

For the adaptation of PSO (and its derivatives such as NPSO [33]) to mesh smoothing problems, in the most general case it should be noted that the objective function F is a function of n variables, where n is the total number of node coordinates to be adjusted for mesh quality improvement. More specifically, if the nodes to be adjusted in the problem are $P_1, P_2, \ldots, P_k$ where $P_i = (x_i, y_i, z_i)$; then the objective function F will be a function of $n = 3k$ variables, and it can be written as $F(x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_k, y_k, z_k)$.

Considering that the number of nodes in a mesh is usually very large, at first glance it can be said that the problem will yield a function F with a large number of variables. The following paragraphs give an idea about how large the dimension or the degree of freedom (DOF) is.

The mesh seen in Figure 3 has a total of $(N+1)(L+1)(M+1)$ nodes assuming that all elements are of first order. During the smoothing of this mesh; if all nodes are allowed to move, then the DOF of this problem will be:

$$\mathrm{DOF}_{\max} = 3(N+1)(L+1)(M+1) \tag{6}$$

The value in (6) represents the worst case; and hence it is called as $\mathrm{DOF}_{\max}$. Obviously by allowing all the nodes to move, the shape of the volume of interest will most probably change after mesh smoothing. However in most cases, the shape of the volume of interest is preserved, which means:

- The corner nodes are fixed,
- The surface nodes are allowed only to move along the surface, and
- The edge nodes are allowed only to move along the edges.

With such restrictions, it can be shown that the DOF might reduce down to

$$\mathrm{DOF}_{\mathrm{nom}} = 3(N-1)(L-1)(M-1) + 4M(L-1) + 4L(N-1) + 4N(M-1) \tag{7}$$

The proof is straightforward depending on the brute force count of the nodes. This value represents the typical case, and it is a nominal value; hence it is called as $\mathrm{DOF}_{\mathrm{nom}}$. This means that $\mathrm{DOF}_{\max}$ is only a theoretical upper bound, which is not encountered in practice.

DOI: 10.1002/fld

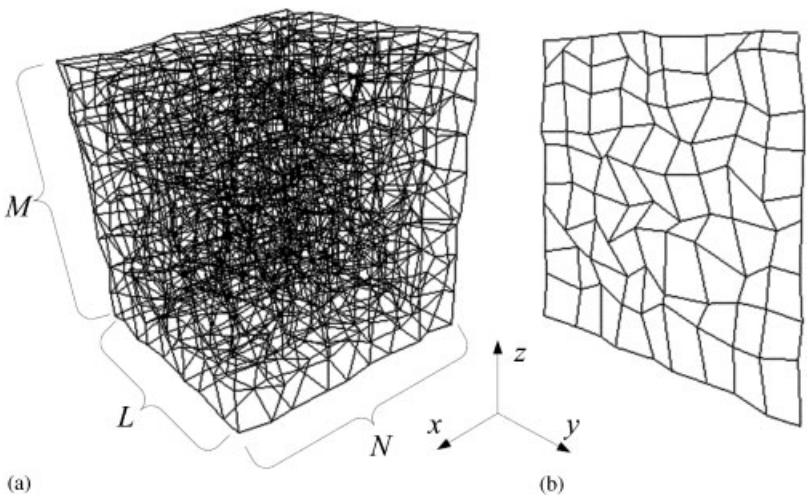

Figure 3. (a) 3D isometric view of the computational domain consisting of *NLM* hexahedral elements and (b) 3D isometric view of only one layer.

Another extreme case is to preserve the surface mesh completely. In such a case, the DOF reduces to

$$\mathrm{DOF}_{\min} = 3(N-1)(L-1)(M-1) \tag{8}$$

In summary, $\mathrm{DOF}_{\min}$ is a lower bound, whereas $\mathrm{DOF}_{\max}$ is an upper bound; and $\mathrm{DOF}_{\mathrm{nom}}$ is a typical value for the smoothing of such volume of interest.

It is sure that high DOF will increase the computation efforts during the mesh smoothing. The following paragraphs discuss some techniques in order to reduce the DOF, and to have a more efficient PSO solution:

1. *Domain decomposition* (*The divide and conquer method*): During the improvement of the mesh quality, there might be opportunities to decompose the main domain into r independent subdomains. Via this manipulation, instead of trying to solve a PSO problem with n-DOF, one can try to solve r independent problems with n_i-DOF where $\Sigma n_i = n$. Certainly, it should be noted that with such an approach, the optimization will not be global (i.e. the smoothed mesh will be sub-optimal).

Assume that the volume of interest (i.e. the problem domain) is divided into subdomains. For mesh continuity, two adjacent subdomains (say V_1 and V_2) should have the same surface mesh at their shared surface (S_{12}). Assume that the mesh of V_1 is smoothed initially; and assume that the surface mesh at S_{12} is preserved during the smoothing of V_2. This means that in such a case, the expected degree-of-freedom (say $\mathrm{DOF}_{\mathrm{exp}}$) for the smoothing operation of V_2 mesh will be even lower than $\mathrm{DOF}_{\mathrm{nom}}$. Moreover, if a volume is surrounded by other volumes in all directions; and if all of its surface meshes have already been smoothed, then $\mathrm{DOF}_{\mathrm{exp}}$ will be reduced down to $\mathrm{DOF}_{\min}$. Consequently, for a non-isolated volume (i.e. surrounded by other volumes) the following can be said about $\mathrm{DOF}_{\mathrm{exp}}$:

$$\mathrm{DOF}_{\min} \leqslant \mathrm{DOF}_{\mathrm{exp}} \leqslant \mathrm{DOF}_{\mathrm{nom}} \tag{9}$$

DOI: 10.1002/fld

Table II. Performance measures for PSO-mesh smoothing of various domains.

					PC-1		PC-2	
N	L	M	Number of elements	Maximum degree of freedom	Total elapsed time (s)	Elapsed time per element (s)	Total elapsed time (s)	Elapsed time per element (s)
2	2	2	8	81	0.06	0.00750	0.048	0.00600
3	3	3	27	192	0.09	0.00333	0.065	0.00241
4	4	4	64	375	0.17	0.00266	0.113	0.00177
5	5	5	125	648	0.27	0.00216	0.193	0.00154
6	6	6	216	1029	0.461	0.00213	0.321	0.00149
7	7	7	343	1536	0.711	0.00207	0.513	0.00150
8	8	8	512	2187	1.122	0.00219	0.786	0.00154
9	9	9	729	3000	1.753	0.00240	1.268	0.00174
10	10	10	1000	3993	2.644	0.00264	2.022	0.00202
11	11	11	1331	5184	3.925	0.00295	3.146	0.00236
12	12	12	1728	6591	5.888	0.00341	4.414	0.00255
13	13	13	2197	8232	8.472	0.00386	5.838	0.00266
14	14	14	2744	10125	12.067	0.00440	8.015	0.00292
15	15	15	3375	12288	16.424	0.00487	10.997	0.00326
16	16	16	4096	14739	22.342	0.00545	14.894	0.00364
17	17	17	4913	17496	29.923	0.00609	20.274	0.00413

Number of Particles = 20.
Number of Iterations = 50.
PC1: Intel Pentium M 1.4 GHz 512 MB RAM Laptop.
PC2: Intel Pentium 4 3.4 GHz 100 GB RAM Desktop.

The optimum computation size (in terms of number of elements) is tried to be investigated by means of a simple analysis. A PSO setup with 20-particle population and 50 iterations is executed. The DOF is chosen to be worst case; i.e. all nodes are allowed to be floating, which yields $\mathrm{DOF}_{\max}$. Table II shows the elapsed time (both total and per element) during the PSO-smoothing for meshes with for various N, L, and M values. All preprocessing (memory allocations, PSO population setup, etc.) and postprocessing (memory deallocations, result displays, etc.) operations are included in the total elapsed time.

The performance measurement is performed by means of PSO mesh generation script executed at Matlab 6.5 on two separate PCs, where:

- PC1 is a laptop Windows PC with Intel Pentium M 1.4 GHz CPU and 512 MB RAM.
- PC2 is a desktop Windows PC with Intel Pentium 4 3.4 GHz CPU and 1 GB RAM.

Investigation of Table II yields the following observations:

- The elapsed time per element is minimum (about 0.00150 s on PC-2) for $6\times6\times6$ or $7\times7\times7$-element-sized domains.
- For domains smaller than $6\times6\times6$, the overhead for the problem setup (operations like swarm generation, and updates) seem to be dominant in terms of CPU time. Hence, due to such overheads, elapsed time per element is high for small domains.
- For domains larger than $7\times7\times7$, PSO-related operations seem to be dominant. As the allocated memory and the particle sizes increase, the updates and objective function evaluations take longer times.

DOI: 10.1002/fld

By using the results of this analysis, the following divide-and-conquer strategy can be proposed:

- For the most efficient PSO mesh smoothing, a mesh can be considered as a collection of $7\times7\times7$-element (or comparable sized) subdomains.
- Mesh smoothing can be performed at each subdomain one by one.

By using the divide-and-conquer strategy, a mesh of 100 000 elements is smoothed by PSO (with 50 iterations, 20 particles) about 84 s on PC-2 by using $7\times7\times7$-element subdomains. During this experiment;

- The shape of the main domain is preserved; i.e. the nodes along the outer surfaces are allowed to be moving along the surfaces. There is no other specific restriction.
- For the interior subdomains, eventual DOF reduction is performed by getting use of the shared surface mesh, if the relevant adjacent subdomain is already smoothed. For such subdomains, the DOF is $\mathrm{DOF}_{\mathrm{exp}}$, where its lower and upper bounds are given in Equation (9).

It should be noted that the manipulation of the interior subdomains is the main weakness of this approach. Holding the surface mesh for a subdomain yields a sub-optimal solution. In addition, changing the order of the subdomains yields a differently smoothed mesh.

The optimality of $7\times7\times7$-element subdomain size can be observed in the same problem numerically. By using the same setup described above, the solution of the same problem takes 123 s on PC-2 if $10\times10\times10$-element subdomains are used.

Such a strategy will dramatically reduce the complexity and the computation time. Certainly, the number of iterations necessary for convergence highly depends on the DOF. Moreover, the population size should be increased for high DOF problems. Nevertheless, the results of this analysis give an idea of optimum subdomain size (which is $7\times7\times7$ or equivalent) for fixed population size and fixed number of iterations.

Figure 4 shows the PSO-smoothed version (via divide-and-conquer method) of the mesh seen in Figure 3.

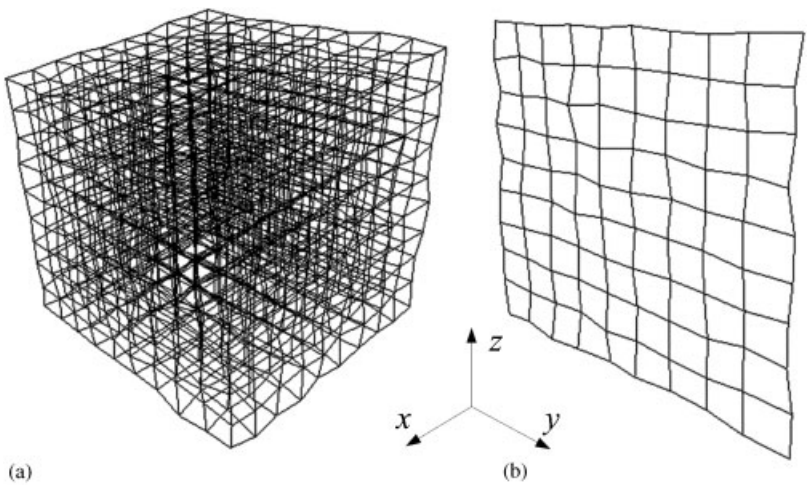

Figure 4. (a) 3D isometric view of the PSO-smoothed version of the mesh in Figure 3 and (b) 3D isometric view of only one layer.

DOI: 10.1002/fld

2. *Fixing some nodes*: Instead of trying to move all the nodes, some nodes (especially the ones on the boundaries) might be considered to be fixed. As an example, if the ith and jth nodes are defined to be fixed; then the function $F(x_1, y_1, z_1, \ldots, x_i, y_i, z_i, \ldots, x_j, y_j, z_j, \ldots, x_k, y_k, z_k)$ will be simplified to $F(x_1, y_1, z_1, \ldots, x_k, y_k, z_k)$.

3. *Imposing nodes dependencies*: The movement of some nodes can be defined to be dependent on each other. For example, if the movement of the ith node in x direction is set to be dependent to the movement of the jth node in x direction; then the function $F(x_1, y_1, z_1, \ldots, x_i, y_i, z_i, \ldots, x_j, y_j, z_j, \ldots, x_k, y_k, z_k)$ will be simplified to $F(x_1, y_1, z_1, \ldots, x_i, y_i, z_i, \ldots, y_j, z_j, \ldots, x_k, y_k, z_k)$.

4. *Setting rules to individual node movements*: The movement of a node inside a mesh might be defined to be in some specific direction; to yield a dependent movement in two directions. For example, the movement of a node might be defined to be in r direction of the cylindrical coordinates where $r=(x^2+y^2)^{1/2}$. If the movement of the ith node is set to be in r direction; then the function $F(x_1, y_1, z_1, \ldots, x_i, y_i, z_i, \ldots, x_k, y_k, z_k)$ will be simplified to $F(x_1, y_1, z_1, \ldots, r_i, z_i, \ldots, x_k, y_k, z_k)$.

5. *Reduction by means of symmetry*: For symmetric problems, instead of trying to optimize the whole mesh, only a subset can be optimized and the whole mesh can be reconstructed. For some problems, this might cause the dimension of F to reduce to $\frac{1}{8}$ or even $\frac{1}{16}$ of the original; if a solution in an octant or half octant is sufficient.

Certainly, manipulations as fixing some nodes, imposing some nodes to be dependent, setting rules to individual node movement reduce the level of quality improvement. There is a trade-off between the quality of the final mesh and the computation time. On the other hand, increasing the DOF does not always guarantee better improvement. Moreover, manipulations as reduction by means of symmetry might not be applicable in most of the problems in practice. Practically, methods other than domain decomposition (divide-and-conquer) might not be applicable in most cases.

The adaptation of PSO to mesh smoothing is slightly different than the normal PSO procedure. Instead of initializing all the particles in n-dimensional space in a totally random manner, an automatically generated mesh is used for this purpose. All particles are positionally initialized by superimposing Gaussian noise (with zero mean and a user defined variance) to the automatically generated mesh at each dimension. The initial velocities of the particles at each dimension are generated as in the ordinary PSO. The Δt value is chosen to be unity; and the initial random velocities of the particles are assigned so that a node can move a distance of at most l_i along one direction due to this velocity component at first iteration. Here, l_i is a user-defined parameter; chosen to be comparable to average edge length along one direction.

Obviously, the choice of the step size (i.e. both Δt and v_n) in the optimization has great impact on the convergence. So far, the effects of the step size and its selection/computation have not been specifically investigated in the mesh smoothing problem. This is a potential subject of further research.

The fitness evaluation throughout the algorithm is achieved by means of minimization of F in Equation (3); i.e. the fitness of a mesh increases as F decreases. All p_{best}, g_{best} computations are performed by using this definition.

With manipulations, initializations and definitions described above, it is possible to apply PSO and its derivatives to the mesh smoothing problems.

For the simple examples in the present work, a population size of 15 is chosen (for most applications it is shown that a population size of 10–20 is efficient); and the number of iterations

DOI: 10.1002/fld

is taken as 50. c_1 and c_2 are chosen to be 2.0 as usual, and w is chosen to be 1.0. Moreover, for all nodes reflecting walls are defined.

As a practical application of above manipulations, two problems in engineering electromagnetics are given as examples. As will be noticed, there are so numerous manipulations in the examples that; one can think whether these reductions are worth to apply rather than solving the problem with high DOF. Certainly, it is wiser to solve these problems directly rather than spending effort to decrease the DOF; but the examples are just given to demonstrate the application of the suggested reduction methods.

First, the circular microstrip patch antenna, which is a well-known structure both for scattering or radiation problems in engineering electromagnetics, is considered. The mesh generated for this problem should be conformal to the circular patch internally; and it should be conformal to the rectangular prism substrate externally. An automatically generated all-hexahedra mesh for this structure can be seen in Figure 5.

First, the problem domain can be reduced by means of symmetry; where the problem can be solved in a quadrant. Then, domain decomposition can be performed by considering the sub-domain inside the circular patch; and the sub-domain between the circular patch and the outer boundary of the substrate separately. More dimension reduction can be achieved if further symmetry is considered in each sub-domain. These manipulations are illustrated in Figure 6. Moreover in each sub-domain, the nodes at the boundaries can be defined to be fixed; and the movements of some nodes can be defined in specific directions only. These operations can be seen in Figure 7.

Another advantage of PSO, when applied to mesh smoothing problem, is the imposure of the boundary conditions. The nodes can easily be classified as 'on the boundary', 'adjacent to the boundary', 'completely inside' during the mesh generation process (at least, the mesh generation algorithm developed for this work performs this operation). By using this information, the boundary

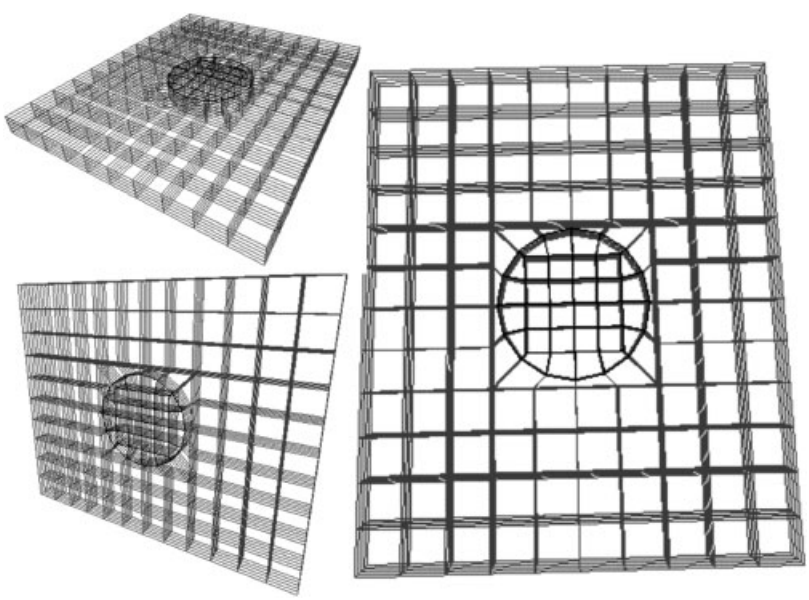

Figure 5. Automatically generated all-hexahedral mesh for circular microstrip patch antenna.

DOI: 10.1002/fld

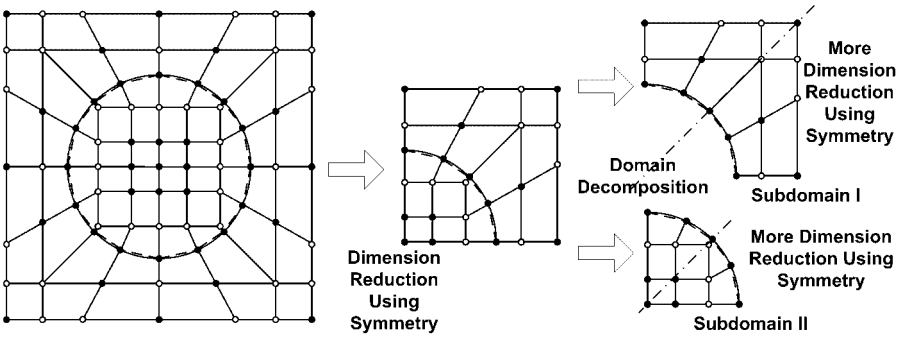

Figure 6. Symmetry reduction and domain decomposition.

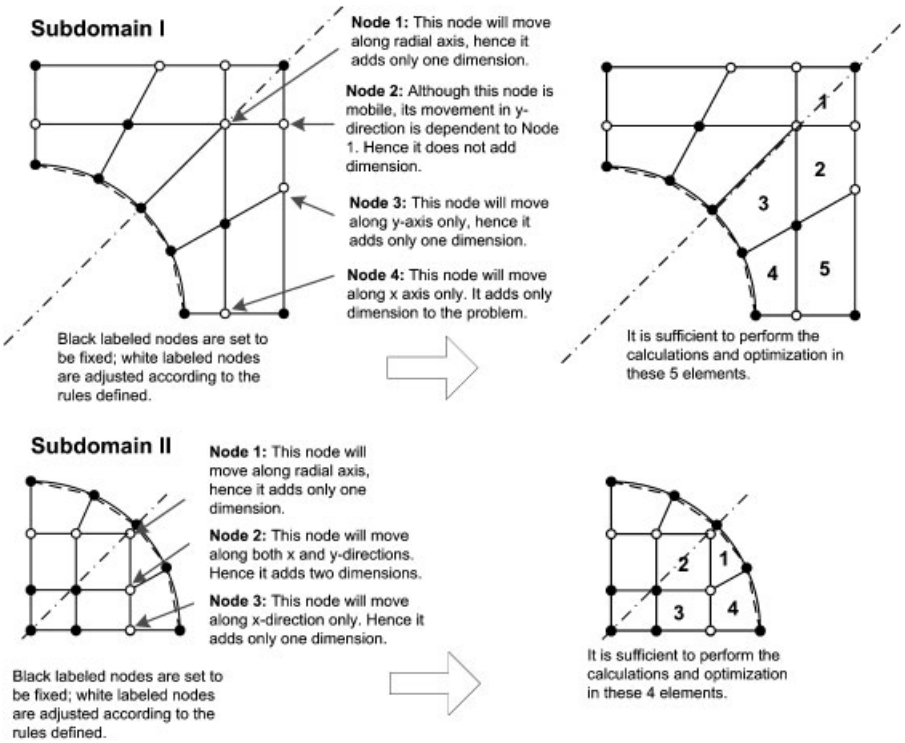

Figure 7. Fixing some nodes, setting rules to individual node movements, and imposing node dependencies.

condition applied to each node can be decided automatically, rather than considering them one by one.

For a moving node, a reflecting boundary condition (wall) can be applied so that the relevant element is kept untangled (although the condition number metric used in this work already controls untangling, this method can be used as a general mechanism for prevention of untangling in general).

DOI: 10.1002/fld

In order to keep the nodes inside a boundary (e.g. the boundary of the computational domain), absorbing or invisible walls can be defined on the boundaries as well. For the circular microstrip patch antenna problem, the reflecting walls defined for all floating nodes are illustrated in Figure 8.

The overall quality improvement in the circular microstrip patch antenna mesh improvement is achieved by means of two separate PSO schemes; a 2-DOF scheme for Subvolume I (Figure 9), and a 2-DOF scheme for Subvolume II (Figure 10). Finally, the whole procedure is summarized and illustrated in Figure 11. In the figures, only the top views of the meshes are given for better and simpler visualization rather than 3D views. It should not be misinterpreted as if the volume mesh is reduced to surface mesh; and hexahedral elements are treated as quadrilateral elements. Without losing generality, all operations (metric and objective function calculation, PSO setup) are performed exactly as defined above.

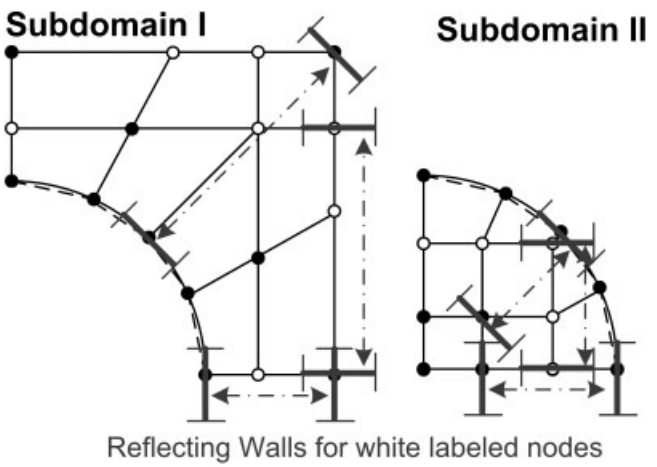

Figure 8. Setting boundary conditions (reflecting walls) to floating nodes.

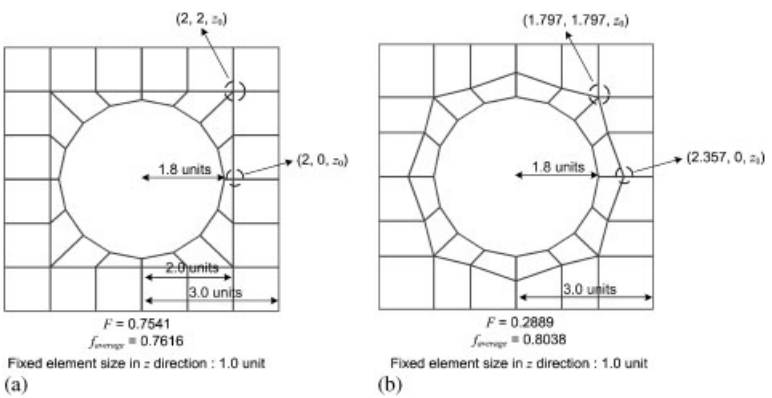

Figure 9. An example of PSO mesh smoothing (circular microstrip patch—outer subdomain; problem dimension reduced to 2-DOF): (a) automatically generated mesh and (b) PSO smoothed mesh.

DOI: 10.1002/fld

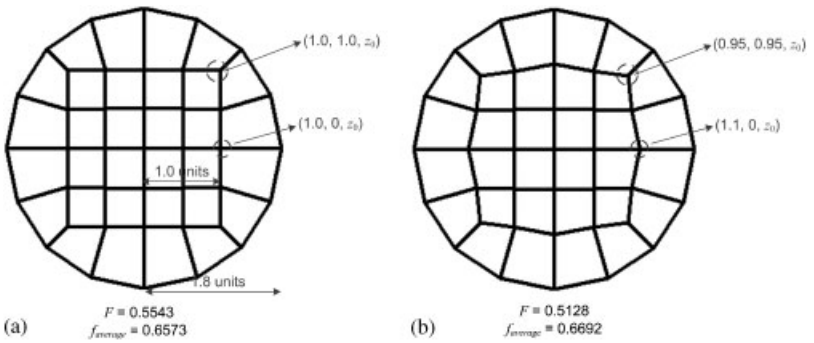

Figure 10. An example of PSO mesh smoothing (circular microstrip patch—inner subdomain; problem dimension reduced to 2-DOF): (a) automatically generated mesh and (b) PSO smoothed mesh.

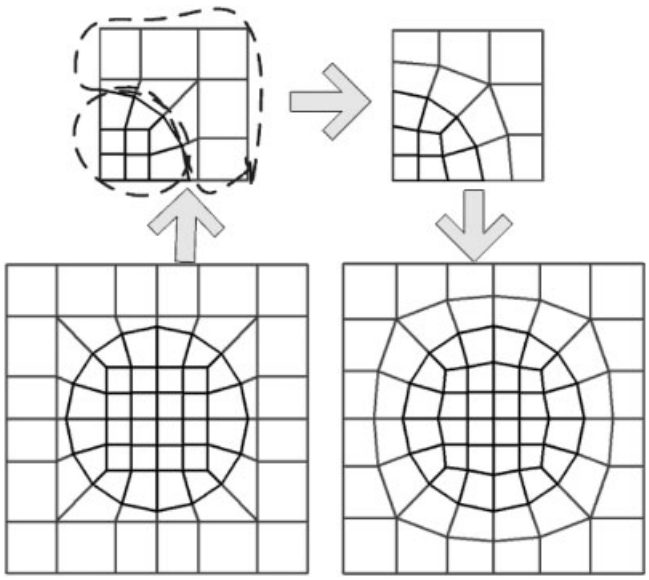

Figure 11. Main steps followed in PSO mesh smoothing of circular microstrip patch (domain decomposition, symmetry reduction, and reconstruction).

Another application considered in the present work is the scattering of a perfectly conducting sphere with a radius of one wavelength (λ), which is an engineering electromagnetics application again. For this problem, it is certainly known that the total electric field inside the perfectly conducting sphere is zero. Hence, there is no need to consider the sphere; and there is no need to generate mesh for this volume. This means that the computational domain is a thick spherical shell. For automatic all-hexahedral mesh generation, the computational domain shall be decomposed into three sub-domains; two top hats and the remaining surrounding region as seen in Figure 12.

DOI: 10.1002/fld

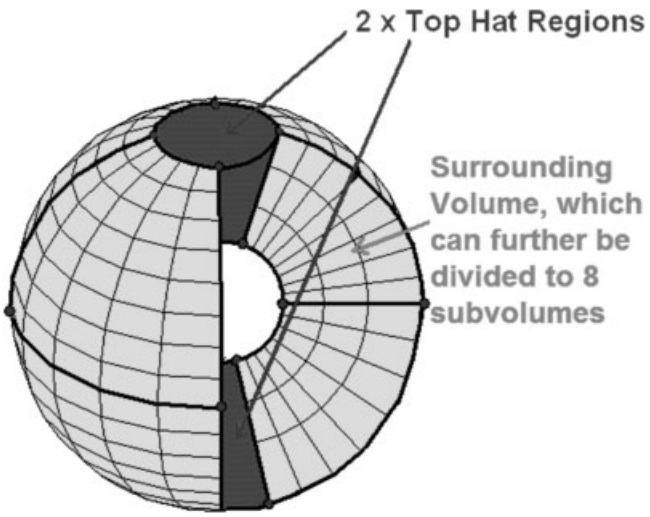

Figure 12. Subdomains necessary for all-hexahedral meshing of a spherical shell.

Automatically generated mesh for the top hat is usually of poor quality; whose cross section for a constant R surface can be seen in Figure 13. Again, by employing symmetry the dimension of the objective function F can be reduced. Moreover, instead of trying to improve the whole top hat mesh, only one layer can be improved and then whole top hat can be reconstructed.

The improvement in the top hat mesh can be seen in Figures 14 and 15 with different views. It should be noted that this improvement is achieved by means of only a 3-DOF PSO scheme.

Since this problem is an open domain problem, in order to be able to apply the FEM, an artificial absorber shall be defined for the simulation of extension to infinity and mesh truncation. Perfectly matched layers (PMLs) defined by Berenger [34] is implemented in this work for this purpose by means of the complex coordinate stretching [35]. Hence, the cross section of the mesh is classified into regions as free space and PML as seen in Figure 16. The calculated electric field for the PML region is physically meaningless, and hence ignored throughout the error norm analysis described in the following paragraphs.

For this problem, the effect of the mesh quality on the Finite Element Solution accuracy is investigated. First, the exact area of the surface S_T of the top hat (as seen in Figure 17) has been compared to the calculated areas of the automatically generated and PSO-smoothed meshes. In other words, for $\mathbf{G}'(x, y, z) = G'(x, y, z)\mathbf{a}_R$ where $G'(x, y, z) = 1$, the following surface integral is computed.

$$S_{\text{exact}} = \int\int_{S_T} \mathbf{G}'(x, y, z) \cdot \mathbf{d}s = \int\int_{S_T} (\mathbf{a}_R G'(x, y, z)) \cdot (\mathbf{a}_R \, \mathrm{d}s) = \int\int_{S_T} \mathrm{d}s \tag{10}$$

By using the isoparametric hexahedral elements (i.e. assuming that each hexahedral element is transformed to a cube in $\xi\eta\zeta$-space extending from $(-1, -1, -1)$ to $(1, 1, 1)$; for any function $G'(x, y, z)$, the surface integral on the surface of an element

$$\int\int_{S_e} G'(x, y, z) \, \mathrm{d}s \tag{11}$$

DOI: 10.1002/fld

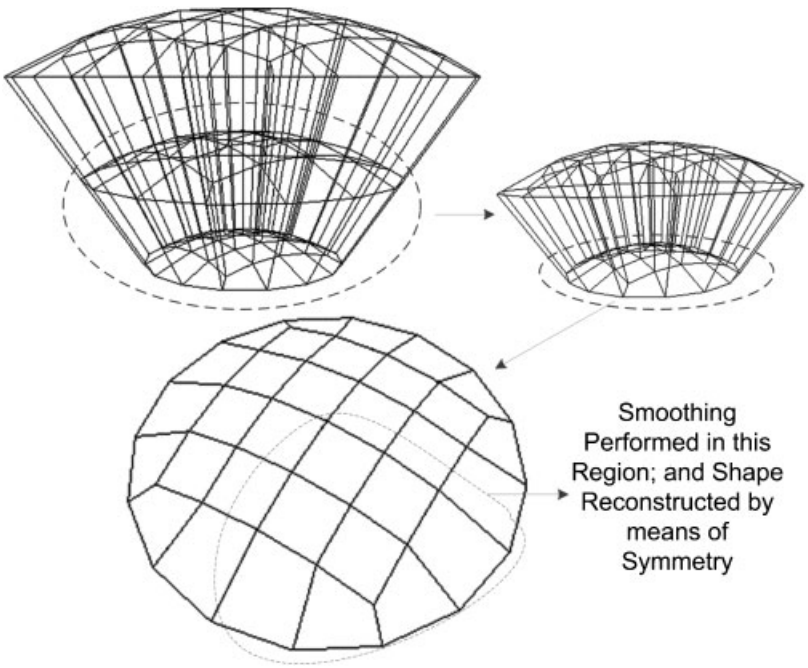

Figure 13. Top hat mesh viewed and investigated in different levels of detail.

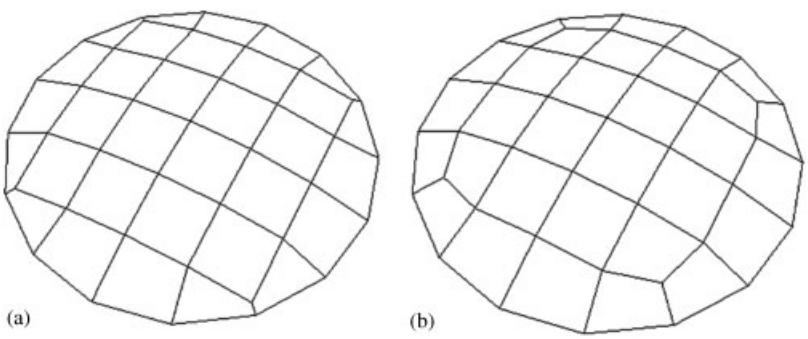

Figure 14. Arbitrary constant R surface extracted from the top hat mesh: (a) automatically generated mesh (3D isometric view of a constant R surface) and (b) PSO smoothed mesh (3D isometric view of a constant R surface.

in the xyz-space can be stated as

$$\int_{-1}^{1}\int_{-1}^{1} G(\xi,\eta,\zeta)\left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| \mathrm{d}\xi\,\mathrm{d}\eta, \quad \zeta \text{ constant} \tag{12}$$

DOI: 10.1002/fld

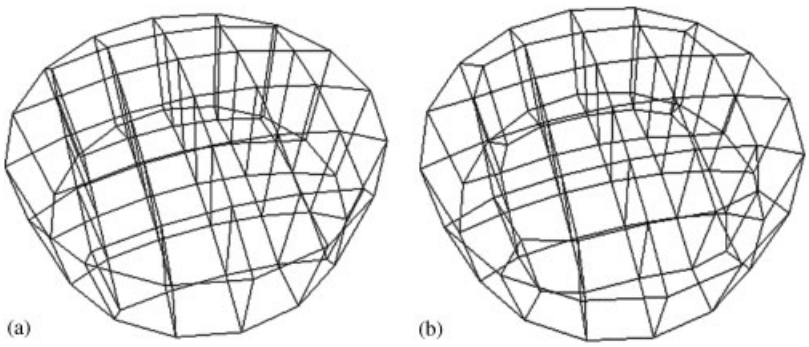

Figure 15. One layer of mesh extracted from the top hat mesh: (a) automatically generated mesh (3D isometric view of one layer) and (b) PSO smoothed mesh (3D isometric view of one layer).

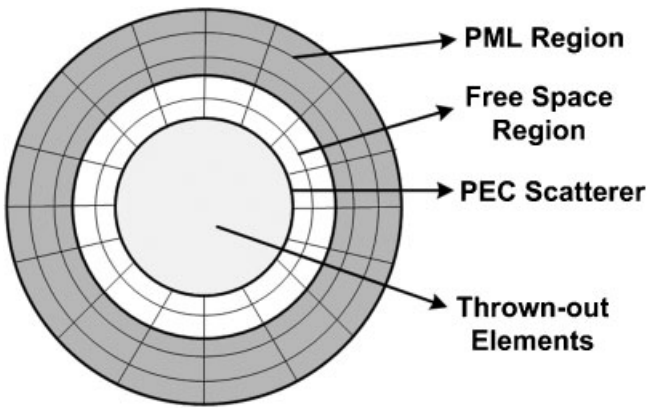

Figure 16. Cross section of the mesh generated for the perfectly electric conductor sphere problem.

in the $\xi\eta\zeta$-space. In Equation (12),

$$\left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| = \left[\left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\right)^2 + \left(\frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \eta} - \frac{\partial z}{\partial \eta}\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \xi}\right)^2\right]^{1/2} \tag{13}$$

Or, in other words,

$$\mathrm{d}s = \left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| \mathrm{d}\xi\,\mathrm{d}\eta \quad \text{where } \zeta \text{ is constant} \tag{14}$$

Certainly, $S_{\text{calculated}}$ can be stated as $\sum_n \iint_{S_e} G'(x,y,z)\,\mathrm{d}s$ where the summation traces all elements on the surface of the top hat. For the error in S, we define the following error norm:

$$\mathrm{err}(S) = \frac{|S_{\text{calculated}} - S_{\text{exact}}|}{S_{\text{exact}}} \tag{15}$$

DOI: 10.1002/fld

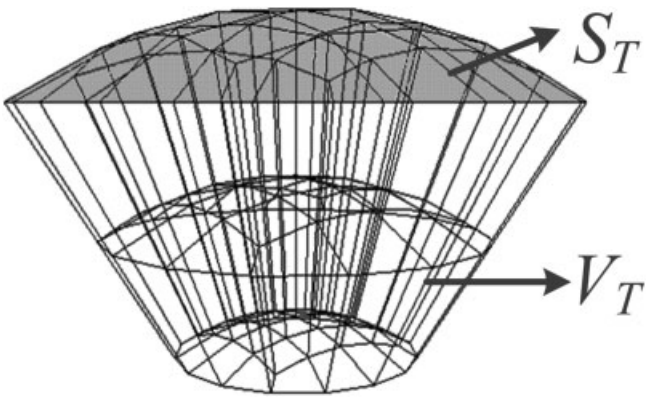

Figure 17. Surface and volume definitions of the top hat.

Second, the exact volume of the top hat (V_T as seen in Figure 17) has been compared to the calculated volumes of the automatically generated and PSO-smoothed meshes. In other words, for $H'(x, y, z) = 1$, the following volume integral is computed

$$V_{\text{exact}} = \iiint_{V_T} H'(x, y, z)\,\mathrm{d}v = \int\int\int_{V_T} \mathrm{d}v \tag{16}$$

By using the isoparametric hexahedral elements; for any function $H'(x, y, z)$, the volume integral in an element

$$\int\int\int_{V_e} H'(x, y, z)\,\mathrm{d}v \tag{17}$$

in the *xyz*-space can be stated as

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} H(\xi, \eta, \zeta)\det(\mathbf{J})\,\mathrm{d}\xi\,\mathrm{d}\eta\,\mathrm{d}\zeta \tag{18}$$

in the $\xi\eta\zeta$-space, where $\mathbf{J}$ is the Jacobian matrix of the xyz to $\xi\eta\zeta$ transformation. Certainly, $V_{\text{calculated}}$ can be stated as $\sum_n \iiint_{V_e} H'(x, y, z)\,\mathrm{d}v$ where the summation traces all elements inside the top hat. For the error in V, we define the following error norm:

$$\text{err}(V) = \frac{|V_{\text{calculated}} - V_{\text{exact}}|}{V_{\text{exact}}} \tag{19}$$

A low quality mesh usually implies that the relevant surface mesh is also of low quality. This implies that both the surface and volume representations are bad; i.e. err(S) and err(V) values are high. On the other hand, having low err(S) and err(V) values does not guarantee mesh quality. These are just indicators about the low quality but not the high quality of a mesh. Hence, more reliable error norms should be defined if possible.

DOI: 10.1002/fld

Table III. Reduction in the error norms after smoothing.

		err(S)	err(V)	err($\mathbf{E}$)
3400 total elements	Automatically generated mesh	0.0398	0.0336	0.0968
	PSO-smoothed mesh	0.0367	0.0302	0.0881
28 800 total elements	Automatically generated mesh	0.0026	0.0022	0.0131
	PSO-smoothed mesh	0.0024	0.0020	0.0117

Moreover, solution of the scattered electric field results obtained by both the automatically generated and the PSO-smoothed meshes are compared to the analytical results, which can be found by using Mie Series. For the electric field, we define the following error norm:

$$\mathrm{err}(\mathbf{E}) = \frac{1}{K} \sum_{i=1}^{K} \frac{\|\mathbf{E}_{\mathrm{calculated}}(P_i) - \mathbf{E}_{\mathrm{exact}}(P_i)\|}{\|\mathbf{E}_{\mathrm{exact}}(P_i)\|} \tag{20}$$

where $\mathbf{E}_{\mathrm{exact}}(P_i)$ is the exact electric field calculated via the Mie Series at the centroid (P_i) of an element lying in free space; whereas $\mathbf{E}_{\mathrm{calculated}}(P_i)$ is the value calculated by FEM at the same point. Certainly, the summation traces all elements lying in free space; K is the number of such elements; and err($\mathbf{E}$) is therefore the mean normalized error over the free space portion of the computational domain.

Table III demonstrates the improvement in the solutions (i.e. reduction in the error norms) after PSO smoothing. The reduction in err($\mathbf{E}$) is about 8–10% for this example. At first sight, one can consider that the effort spent might not be worth to have such reduction. But it should be considered that the initial mesh is also not very low in quality; and this is just a simple example for demonstration of the concept. Certainly for the cases where the initial mesh is very low in quality, the improvement in err($\mathbf{E}$) will be drastic.

In order to demonstrate the effectiveness of the proposed method, more complicated problems are investigated. Figure 18(a) shows an object having a mesh with 2000 hexahedral elements and noisy internal nodes (i.e. noise is superimposed to the internal nodes; the surface nodes are preserved). For the solution, the mesh is smoothed by dividing this mesh into seven subdomains (6 of which with 300 elements, 1 of which with 200 elements). The population size is chosen as 100, and the number of iterations is chosen as 1000; yielding 100 000 objective function computations during optimization. Figure 18(b) shows the smoothed version of the mesh. The Matlab implementation of the smoothing takes 605 s (10 min 5 s CPU time) on PC1; and 402 s (6 min 42 s CPU time) on PC2 with these settings. From the figure, it can be observed that the smoothed mesh is of good quality but not perfect. The main reasons are the sub-optimality caused by the divide-and-conquer approach (as mentioned before), and the termination of the PSO at some point (i.e. 1000th iteration).

With similar settings, the mesh shown in Figure 19 is also smoothed with the proposed method (again, for the generation of the bad mesh noise is superimposed to the internal nodes; the surface nodes are preserved). The mesh with 2000 elements is smoothed at about 609 s (10 min 9 s CPU time) on PC1; and 404 s (6 min 44 s CPU time) on PC2.

As all optimization-based smoothing techniques, the proposed method is computationally expensive compared to Laplacian smoothing and its derivatives. For the execution time of the main PSO routine; the DOF, the population size, and the number of iterations are the driving factors. In addition, the number of elements (thus DOF) is the driving factor for the elapsed time during the fitness/objective function computation.

DOI: 10.1002/fld

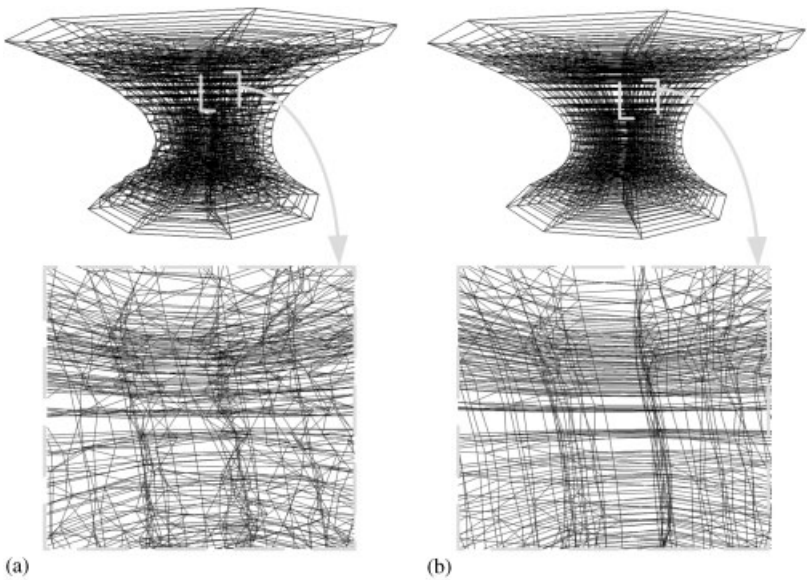

Figure 18. A shape homeomorphic to a cylinder: (a) intentionally badly generated (noise added) mesh and (b) PSO-smoothed mesh with some details.

Moreover, as all population-based optimization techniques, the memory requirements of the proposed method are higher than other optimization techniques; since it requires the current and personal best states of all particles in the swarm (i.e. all population) together with the global best solution to be stored in memory. Assuming that for each node, the coordinates (real numbers) together with the indices (positive integers) of the connected nodes are stored; for an efficient C/C++ implementation, the approximate memory consumption of the proposed method during the execution time is given as

$$N_{\text{memory}} \text{ (bytes)} \cong 18 \times (N_{\text{pop}} + 1) \times \text{DOF} \tag{21}$$

where N_{pop} is the population size; which means about 2.5-megabyte memory usage for the problems seen in Figures 18 and 19. It should be emphasized that the memory usage of the current Matlab implementation is much more than this value due to the nature of Matlab.

In summary, the DOF, the population size, and the number of iterations are the key factors determining the resource (memory and CPU time) usage. For given examples (even though the global-optimality is sacrificed due to the divide-and-conquer approach, and sub-optimality is accepted), the resource usage is tolerable and the results are promising.

6. CONCLUSIONS AND FUTURE WORK

A method for mesh improvement by means of local node repositioning based on the condition number-related quality metric and Particle Swarm Optimization is proposed in this work. Meshes that can be encountered in practical situations are smoothed with this method. As an example, the

DOI: 10.1002/fld

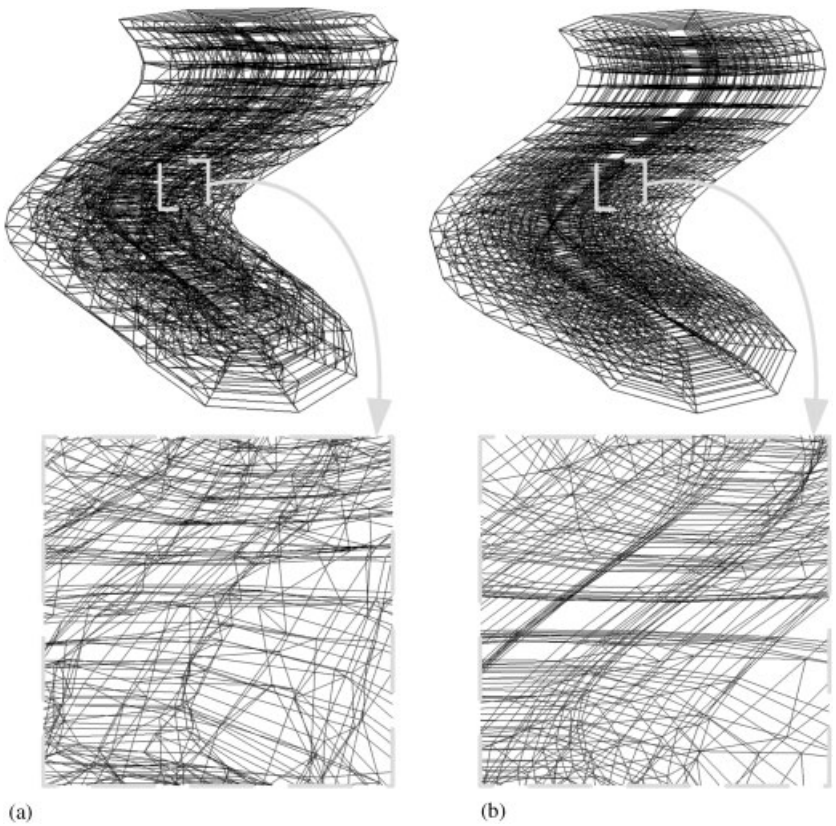

Figure 19. Another shape homeomorphic to a cylinder: (a) intentionally badly generated (noise added) mesh and (b) PSO-smoothed mesh with some details.

impact of smoothing to the finite element solution accuracy is observed when H(*curl*)-conforming hexahedral elements are used. On the other hand, the method puts no restriction on the type of the hexahedral element; i.e. quality of any other hexahedral element type (H(*grad*)-conforming, H(*div*)-conforming) can be improved by means of this method.

This paper demonstrates the applicability of the PSO to mesh smoothing problem as a conceptual proof. Obviously, this yields potential future studies such as:

- investigation of the computation of ideal step size;
- the ideal population size and number of iterations for given DOF;
- improvement in convergence by means of modifications to the particle behaviors.

As future work, mesh improvement by means of PSO might also be extended to other types of finite elements (e.g. triangular and quadrilateral elements in 2D, tetrahedral, prismatic elements in 3D). The method can also be extended to any type of element with higher order if applied to appropriate quality metrics or combined to appropriate validity criteria. Moreover, with the definition of several objective functions depending on various quality metrics; multiobjective mesh smoothing can be performed by means of multi objective particle swarm optimization [36]. And

DOI: 10.1002/fld

finally, for very high DOF problems (in the order of millions), instead of sacrificing the global-optimality by DOF reduction techniques; parallelization of the PSO at distributed-computing environments can also be considered.

REFERENCES

1. Knupp P. Next-generation sweep tool: a method for generating all-hex meshes on two-and-one-half dimensional geometries. *7th International Meshing Roundtable*, Detroit, MI, 1998; 505–513.
2. Folwell N, Mitchell S. Reliable whisker weaving via curve contraction. *7th International Meshing Roundtable*, Dearborn, MI, 1998; 365–378.
3. Jankovich S, Benzley S, Shepherd J, Mitchell S. The graft tool. *8th International Meshing Roundtable*, S. Lake Tahoe, CA, 1999; 387–392.
4. Owen S, Saigal S. H-Morph: an indirect approach to advancing front hex-meshing. *International Journal for Numerical Methods in Engineering* 2000; **49**(1–2):289–312.
5. Schneider R. Automatic generation of hexahedral finite element meshes. *4th International Meshing Roundtable*, Albuquerque, NM, 1995; 103–114.
6. Kovalev K. Unstructured hexahedral non-conformal mesh generation. *Ph.D. Thesis*, Vrije Universtiteit Brussel, 2005.
7. Canann S, Tristano J, Staten M. An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. *7th International Meshing Roundtable*, Dearborn, MI, 1998; 479–494.
8. Mukherjee N. A Hybrid, Variational 3D smoother for orphaned shell meshes. *11th International Meshing Roundtable*, Ithaca, NY, 2002; 379–390.
9. Freitag L, Jones M, Plassmann P. An efficient parallel algorithm for mesh smoothing. *4th International Meshing Roundtable*, Albuquerque, NM, 1995; 47–58.
10. Hansbo P. Generalized Laplacian smoothing of unstructured grids. *Communications in Numerical Methods in Engineering* 1995; **11**:455–464.
11. Parthasarathy N, Kodiyalam S. A constrained optimization approach to finite element mesh smoothing. *Finite Elements in Analysis and Design* 1991; **9**:309–320.
12. Zavattier P. Optimization strategies in unstructured mesh generation. *International Journal for Numerical Methods in Engineering* 1996; **39**:2055–2071.
13. Farhat C, Degand C, Koobus B, Lesoinne M. Torsional springs for two-dimensional unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**:231–245.
14. Degand C, Farhat C. A three-dimensional torsional spring analogy method for unstructured dynamic meshes. *Computers and Structures* 2002; **80**:305–316.
15. Calvo N, Idelsohn S. All-hexahedral mesh smoothing with a node-based measure of quality. *International Journal for Numerical Methods in Engineering* 2001; **50**(8):1957–1967.
16. Knupp P. A method for hexahedral mesh shape optimization. *International Journal for Numerical Methods in Engineering* 2003; **58**:319–332.
17. Branets L, Carey GF. A local cell quality metric and variational grid smoothing algorithm. *12th International Meshing Roundtable*, Sante Fe, NM, 2003; 371–390.
18. Brewer M, Freitag Diachin L, Knupp P, Leurent T, Melander D. The Mesquite mesh quality improvement toolkit. *12th International Meshing Roundtable*, Sante Fe, NM, 2003; 239–250.
19. Brewer M, Folwell N, Knupp P. Increasing Tau3P abort-time via mesh quality improvement. *12th International Meshing Roundtable*, Sante Fe, NM, 2003; 379–392.
20. Knupp P. Mesh quality improvement for SciDAC applications. *Journal of Physics*: *Conference Series* 2006; **46**:458–462.
21. Knupp P. Algebraic mesh quality measures. *SIAM Journal on Scientific Computing* 2001; **23**(1):193–218.
22. Knupp P. Hexahedral and tetrahedral mesh untangling. *Engineering with Computers* 2001; **17**(3):261–268.
23. Vachal P, Garimella RV, Shashkov MJ. Untangling of 2D meshes in ALE simulations. *Journal of Computational Physics* 2004; **196**:627–644.
24. Eberhart RC, Shi Y. Evolving artificial neural networks. *Proceedings of 1998 International Congress on Neural Networks and Brain*, Beijing, P.R.C., 1998.

DOI: 10.1002/fld

25. Kennedy J, Spears WM. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on multi modal problem generator. *Proceedings of IEEE International Congress on Evolutionary Computation*, Anchorage, AK, 1998.
26. Hassan R, Cohanim B, de Weck O. A comparison of particle swarm optimization and the genetic algorithm. *Proceedings of 46th AIAA/ASME/ASCE/AHS/ASC Structures*, *Structural Dynamics and Materials Conference*, Austin, TX, 2005.
27. Jones KO. Comparison of genetic algorithms and particle swarm optimisation for fermentation feed profile determination. *Proceedings of International Conference on Computer Systems and Technologies* (*CompSysTech*), Tarnovo, Bulgaria, 2006.
28. Lukeš Z, Raida Z. Multi-objective optimization of wire antennas: genetic algorithms versus particle swarm optimization. *Radioengineering* 2005; **14**:91–97.
29. Mouser CR, Dunn SA. Comparing genetic algorithms and particle swarm optimisation for an inverse problem exercise. *Australia and New Zealand Industrial and Applied Mathematics* (*ANZIAM*) *Journal* 2005; **46**:C89–C101.
30. Pasupuleti S, Battiti R. The gregarious particle swarm optimizer (G-PSO). *Proceedings of 8th Annual Conference on Genetic and Evolutionary Computing*, Seattle, WA, 2006; 67–74.
31. Kennedy J, Eberhart RC. Particle swarm optimization. *Proceedings of IEEE Congress on Neural Networks IV*, Piscataway, NJ, 1995.
32. Robinson J, Rahmat-Samii Y. Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation* 2004; **52**(2):397–407.
33. Yang C, Simon D. A new particle swarm optimization technique. *Proceedings of 18th International Conference on Systems Engineering*, Las Vegas, NV, 2005; 164–169.
34. Berenger JP. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics* 1994; **114**:185–200.
35. Chew WC, Jin JM, Michielssen E. Complex coordinate stretching as a generalized absorbing boundary condition. *Microwave and Optical Technology Letters* 1997; **15**:363–369.
36. Coello CAC, Lechunga MS. MOPSO: a proposal for multiple objective particle swarm optimization. *Proceedings of the Congress on Evolutionary Computation* 2002; 1051–1056.

DOI: 10.1002/fld