Published in final edited form as:
*J Comput Phys*. 2008 ; 227(11): 5778–5798. doi:10.1016/j.jcp.2008.02.008.

# Fast intersections on nested tetrahedrons (FINT): An algorithm for adaptive finite element based distributed parameter estimation

**Jae Hoon Lee***, **Amit Joshi**, and **Eva M. Sevick-Muraca**
*Department of Radiology, Baylor College of Medicine, Photon Migration Laboratory, One Baylor Plaza, BCM 360, Houston, TX 77030, USA*

## Abstract

A variety of biomedical imaging techniques such as optical and fluorescence tomography, electrical impedance tomography, and ultrasound imaging can be cast as inverse problems, wherein image reconstruction involves the estimation of spatially distributed parameter(s) of the PDE system describing the physics of the imaging process. Finite element discretization of imaged domain with tetrahedral elements is a popular way of solving the forward and inverse imaging problems on complicated geometries. A dual-adaptive mesh-based approach wherein, one mesh is used for solving the forward imaging problem and the other mesh used for iteratively estimating the unknown distributed parameter, can result in high resolution image reconstruction at minimum computation effort, if both the meshes are allowed to adapt independently. Till date, no efficient method has been reported to identify and resolve intersection between tetrahedrons in independently refined or coarsened dual meshes. Herein, we report a fast and robust algorithm to identify and resolve intersection of tetrahedrons within nested dual meshes generated by 8-similar subtetrahedron subdivision scheme. The algorithm exploits finite element weight functions and gives rise to a set of weight functions on each vertex of disjoint tetrahedron pieces that completely cover up the intersection region of two tetrahedrons. The procedure enables fully adaptive tetrahedral finite elements by supporting independent refinement and coarsening of each individual mesh while preserving fast identification and resolution of intersection. The computational efficiency of the algorithm is demonstrated by diffuse photon density wave solutions obtained from a single- and a dual-mesh, and by reconstructing a fluorescent inclusion in simulated phantom from boundary frequency domain fluorescence measurements.

### Keywords

Intersection; Tetrahedral mesh; Dual-mesh; Refinement; Derefinement; Tomography; Adaptive finite element; Fluorescence; Photon diffusion

## 1. Introduction

Over the past few decades, electrical impedance, microwave computed, diffuse optical, and fluorescence enhanced optical tomographies have been sought as medical imaging techniques that use model-based, iterative reconstruction algorithms to reconstruct distinct tissue properties for identification of interior, diseased tissue from boundary value measurements. In these imaging modalities, the finite element method (FEM) is used to represent arbitrary tissue volumes for solution of the forward problem, i.e., prediction of the boundary measurements from a model and a given (or guessed) tissue property map, and for solution of the inverse

* Corresponding author. Current address: Department of Medical Research, Korea Institute of Oriental Medicine, Expo-ro 483, Yuseong-gu, Daejeon 305-811, Korea. Tel.: +1 713 798 9195; fax: +1 713 798 8050. E-mail address: daniel-jhlee@kiom.re.kr (J.H. Lee).

problem, i.e. recovery of the interior tissue property map from a model and set of boundary measurements. The inverse imaging solution depends upon optimization procedures that seek to minimize the error between forward FEM predictions and actual measurements by iteratively adjusting the spatially distributed parameter of interest. In tomography applications, unknown parameter is typically discretized by nodal basis functions. However, other discretization schemes such as piecewise discontinuous parameter maps are also possible. The accuracy of the forward imaging problem solution along with the computation expense, increases with the refinement of the finite element mesh. For the inverse problem, increasing mesh refinement results in better potential image resolution, while at the same time increasing the number of unknown variables which can create numerical instability. Goal-based adaptive local refinement of finite element meshes can produce highly resolved images at a low computational cost.

The basis for mesh refinement for optimal forward and inverse problems differs and hence using a single mesh for solving the forward problem accurately as well as limiting the number of unknowns in the inverse problem is not possible. Therefore, the use of separate meshes, i.e., a refined mesh for accurate forward solutions and a coarse inverse mesh for parameter recovery enables a complete decoupling of the two problems. Based upon this recognition, fixed dual-mesh-based approaches have been proven to be successful [1–7]. However, when there is no *a priori* information available on the spatial distribution of tissue properties, the discretization of forward and inverse meshes becomes arbitrary rather than optimal.

Adaptive mesh refinements based upon *a posteriori* error estimates have been tailored to specific models [8–16] but have not been widely applied to distributed parameter estimation problems of the type encountered in tomography. The few reports that employ adaptive mesh refinement in two- and three-dimensional electrical impedance tomography [17–19] are based upon a single mesh used for both forward and inverse problems. Although the adaptive dual-mesh techniques employing refinement both in the forward and the inverse meshes were attempted for optical tomography [20], the work was limited to the two-dimensional problems. Recently, Joshi et al. [21,22] reported the use of a fully adaptive, three-dimensional fluorescence enhanced optical tomography technique that used a hexahedron-based dual-mesh scheme to improve image resolution at feasible computational cost. The approach employed a forward mesh that was refined/coarsened based upon the spatial gradient of excitation and emission fluences and of tissue properties while the inverse mesh was independently refined/coarsened based upon the spatial gradient of tissue fluorophore concentration. The use of hexahedral elements, however, limits application to simple rectilinear geometries. Curved geometries require higher order or isoparametric brick shaped elements resulting in complex algorithms. As tetrahedral meshes can be relatively easily generated for complicated geometries encountered in medical imaging, optimal, three-dimensional adaptive tomography employing tetrahedron-based dual-adaptive mesh scheme will positively impact multiple biomedical imaging modalities. However, independent refinement of tetrahedral elements in the two meshes requires solution to a computationally intensive intersection problem between two tetrahedrons on the different meshes, which prohibits its deployment in iterative parameter estimation algorithms. In principle, the intersection between two tetrahedrons can be found using a series of triangle-line piercing tests conducted in three-dimensional real object space. However, the object space intersection scheme is not robust due to the finite precision floating point arithmetic and thus sophisticated implementation is required to circumvent the precision errors. Furthermore, geometrical searching is required to pick candidate tetrahedrons for intersection in the two related forward and inverse meshes. However, the simple brute force algorithm employing intersection check for all possible pairs of tetrahedrons in the two meshes will cost $O(N^2)$ operations where $N$ is the number of tetrahedrons. For static tetrahedral dual-mesh tomography, the intersection is computed once and stored at the preprocessing step. For iterative reconstruction phase, the intersection information thus stored is repetitively used.

However, the adaptive dual-mesh tomography requires intersection be resolved whenever the geometric environment changes due to refinement/coarsening of forward or inverse mesh. Therefore, for adaptive tetrahedron-based dual-mesh tomography, a fast and robust solution to the intersection problem is essential to cope with repetitive and independent refinement/derefinement of the two related tetrahedral meshes. To our knowledge, there is no demonstrated method for handling intersection in this dynamically varying situation, and consequently, no adaptive tomography technique using tetrahedral meshes is yet available for iterative tomographic parameter estimation problems.

In this contribution we present a fast and robust algorithm to handle intersection in the nested conforming tetrahedral elements that enables the adaptive dual-mesh-based three-dimensional tomography. The demonstrated intersection algorithm performs refinement of a parent mesh using the 8-subtetrahedron subdivision scheme [23] but does not perform intersection based upon real object space. Instead, the algorithm uses the volume coordinates or linear tetrahedral element basis functions, leading to dramatically simplified, fast and robust intersection outputs. The resolved intersection results in a set of vertices and weight function values on each vertex of disjoint tetrahedron pieces that completely cover up the intersection region of two tetrahedrons.

This paper is organized as follows: In Section 2, the 8-subtetrahedron subdivision scheme [23] used as the basis of this work is briefly reviewed before the intersection handling algorithm FINT is detailed and applied towards the finite element assembly procedure on dual mesh intersection outputs. In Section 3, first the efficiency of FINT is validated by demonstrating that the finite element assembly time scales linearly with the number of elements, followed by the application of FINT to a simulated fluorescence image reconstruction problem. Finally, we conclude by summarizing our work and commenting upon its future implications.

## 2. Theory and approach

For a dual-adaptive mesh-based finite element scheme, we first define two meshes: $M$ and $M'$. For the inverse problem, $M$ will be the forward mesh for solutions of the forward variables and $M'$ will be the inverse mesh for the unknown parameter recovery, or for coupled field problems like fluorescence optical imaging, $M$ can be used for discretizing the excitation field while $M'$ is used for solving for the fluorescence emission field. For the applications considered in this paper, $M'$ will be created by duplicating an initial coarse mesh $M$. Both meshes will be then allowed to adaptively refine and derefine independently. Any tetrahedron in the initial dual-mesh is called a *root* tetrahedron. If any tetrahedron $\mathbf{T}$ is subdivided into subtetrahedrons, then each subtetrahedron is called a *child* of $\mathbf{T}$ and $\mathbf{T}$ is called the *parent* of the child. A tetrahedron with no children is called a *leaf* tetrahedron. In the following subsection, the mesh refinement and derefinement procedure is detailed.

### 2.1. Refinement and derefinement in 8-similar subtetrahedron subdivision scheme

**2.1.1. Mesh refinement—**The 8-similar subtetrahedron subdivision scheme [23] is illustrated in Fig. 1. The scheme allows only 0, 1, or 3 split points on each triangle face, leading to four different types of tetrahedron subdivisions. Initial tetrahedrons are called *regular*. When any *regular* tetrahedron is refined, it is always subdivided into eight subtetrahedrons as shown in Fig. 1(a). The subdivision is created by the basic operation termed $\mathrm{SUB}_8$. Any tetrahedrons generated by $\mathrm{SUB}_8$ are called *regular* and of the type $S_8$. All the other subdivision configurations shown in Fig. 1(b)–(d) are introduced to ensure conformity. The subdivisions are created by operations termed $\mathrm{SUB}_2$, $\mathrm{SUB}_{4a}$, and $\mathrm{SUB}_{4b}$ and produce two, four, and four child tetrahedrons, respectively. The resulting subtetrahedrons are called *non-regular* and of type $S_2$, $S_{4a}$, and $S_{4b}$, respectively. If any non-regular tetrahedron $\mathbf{T}$ is chosen for refinement, its regular parent $\mathbf{T}_p$ is always refined first by $\mathrm{SUB}_8$. Therefore, the non-regular tetrahedrons

exist only at leaf levels with no children. In the sequence of nested tetrahedral meshes, only tetrahedrons of types $S_8$, $S_2$, $S_{4a}$, and $S_{4b}$ are present. In accordance with the 8-subtetrahedron subdivision for tetrahedrons, triangle faces in the tetrahedral networks are refined by 4-similar subtriangle subdivision [23]. It can be noted that another tetrahedron subdivision scheme can be generated by dividing the tetrahedron into four subtetrahedrons about a single point in the center. However, poorly shaped elements thus created will require face-edge swapping to ensure good quality, and the new elements will not be nested, which makes the intersection problems harder to handle for implementing dual-adaptive mesh refinement.

**2.1.2. Mesh derefinement—**Mesh coarsening or derefinement is another operation that must be considered in fully adaptive finite element methods. With the combination of refinement and derefinement operations, the sequence of nested mesh becomes more efficient than when refinement operations are used alone. The efficiency arises because coarsening operations prevent the exponential increase in the number of nodes that would otherwise occur from refinement operations. To date, derefinement operations have been developed for nested meshes generated from edge bisection [24] and none exist for 8-subtetrahedron subdivision. A local derefinement algorithm in the nested meshes created by the 8-subtetrahedron subdivision has been developed by the authors and implemented also in this work. Fig. 2 illustrates examples of derefinement in a two-dimensional triangle mesh that can be extended to three-dimensional tetrahedral mesh in a straightforward manner. Initially, all nodes present in the tetrahedral mesh are marked as *alive*. Then, the nodes of the tetrahedral element **T** to be derefined are marked as *dead* except those shared by their parent. Next, we iterate the marking process until all triangle facets of the tetrahedral elements lying at **T**'s parent level have only 0, 1, or 3 *alive* split points. However, uncontrolled iteration leads to arbitrary disentanglement of the nested mesh. Mesh derefinement is restricted to a local region by employing the following procedure:

- **i.** Mark a tetrahedron **T** of the subdivision level $l$ as dead.
- **ii.** Pick the neighbor regular tetrahedron $\mathbf{T}_n$ of the level $l-1$ sharing the nodes of the parent tetrahedron $\mathbf{T}_p$ of the tetrahedron **T** to be derefined.
- **iii.** Set the split point of any $\mathbf{T}_n$'s edge connected to any $\mathbf{T}_p$'s node as dead.
- **iv.** If any edge has an endpoint marked as dead, mark its split point as dead.
- **v.** Mark any children of $\mathbf{T}_n$ and $\mathbf{T}_p$ with dead nodes as dead; and recurs (ii)–(v) for dead children of $\mathbf{T}_n$ and $\mathbf{T}_p$ until no dead point appears.
- **vi.** Any edge **E** any one of whose endpoint is marked as dead, and any triangle face **F** any one of whose vertex is marked as dead, are marked as dead.

The procedure completes marking up to **T**'s subdivision level $l$ and may result in dead endpoints of edges of the subdivision level $l+1$, triggering marking procedure for elements of the level $l+1$ outside the region occupied by $\mathbf{T}_p$ and $\mathbf{T}_n$. The marking front propagates outward symmetrically with decreasing sequence of the subdivision level until no marking procedure is necessary. The above procedure ensures the triangles of the subdivision level $l-1$ on the boundary of the region occupied by the tetrahedrons $\mathbf{T}_p \cup \{\mathbf{T}_n\}$ to be unaffected and retain their original 0, 1, or 3 split points, and the triangles faces shared between $\mathbf{T}_n$'s to have either 0 or 1 *alive* split point lying only on the boundary of the region occupied by $\mathbf{T}_p \cup \{\mathbf{T}_n\}$. Therefore, the tetrahedral network of the subdivision level $l$ lying outside the region occupied by $\mathbf{T}_p \cup \{\mathbf{T}_n\}$ are not affected. When tetrahedrons initially chosen for derefinement have different subdivision levels, we do marking processes for tetrahedrons lying at the deepest level first and then proceed to the next deepest level. Finally, deletion of all dead nodes, edges, faces, and tetrahedrons, followed by reapplication of regular 8-subtetrahedron subdivision according to the number of remaining split points, recovers conformity of the tetrahedral mesh.

In the two-dimensional cases as illustrated in Fig. 2, the element **T** in the above (i)–(v) corresponds to a triangle and therefore the geometrical entity that should have 0, 1, or 3 split points is the **T** itself; accordingly, we have only an edge **E** to mark as dead in (vi). For the derefinement of a triangle **T** of the subdivision level $l$, the edges of the subdivision level $l - 1$ on the boundary of the region occupied by the triangles $\mathbf{T}_p \cup \{\mathbf{T}_n\}$ are unaffected and thus retain their original 0 or 1 split point, whereas the edges shared between $\mathbf{T}_n$'s have 0 *alive* split point. Therefore, the triangular network of the subdivision level $l$ lying outside the region occupied by the triangles $\mathbf{T}_p \cup \{\mathbf{T}_n\}$ are not affected. Similarly, deletion of all dead nodes, edges, and triangles followed by reapplication of regular 4-similar subtriangle subdivision according to the number of remaining split points, recovers conformity of the triangle mesh.

## 2.2. Solution of intersection problem

Finite element-based solution of PDE's on an adaptive mesh $M$, when the parameters are discretized on another independently adapting mesh $M'$, requires the resolution of intersections between $M$ and $M'$. Refinement/coarsening operations in a dual-mesh environment require efficient solution to tracking tetrahedron partners in the dual-meshes, finding the plane of intersection and solving the intersection problem. In this work, we solve the intersection problem by taking advantage of (i) the partnership of the twin tetrahedrons in the nested dual-mesh environments that enables fast searching of candidates of intersecting pairs of tetrahedrons and (ii) the local linear finite element basis functions or weight functions, otherwise called volume coordinates that make intersection computations simple and concise.

**2.2.1. Partnership—**The partnership in the dual-mesh scheme is created as follows. A *partner* data field is introduced for each tetrahedron of the type $S_8$. An input parent mesh $P$ is duplicated. $P$ and its duplicate will be denoted as $M$ and $M'$, respectively. Then each tetrahedron in $M$ has its *twin* in $M'$ and vice versa. Initially, all tetrahedrons in $M$ and $M'$ are marked as *regular*, i.e., of the type $S_8$. The *partnership* is defined in Definition 1.

**Definition 1:** The *partner* of any *regular* tetrahedron **T** in $M$ is **T**'s *twin* $T'$ in $M'$ if $\mathbf{T}'$ exists, and vice versa.

If **T**'s partner is $\mathbf{T}'$, then $\mathbf{T}'$'s partner is **T**. Any twin tetrahedrons are always mutually related to each other by the *partner* field. Therefore, each *root* tetrahedron in $M$ at subdivision level 0 has its partner in $M'$. Whenever a new tetrahedron **T** of $S_8$ is created in $M$ by refinement, its twin $\mathbf{T}'$ is assigned as the partner of **T** if $\mathbf{T}'$ exists in $M'$. Likewise, whenever a new tetrahedron $\mathbf{T}'$ of $S_8$ is created in $M'$ by refinement, its twin **T** is assigned as the partner of $\mathbf{T}'$ if **T** exists in $M$. When **T** is the $i$th child among its parent $\mathbf{T}_p$'s eight children, then its partner $\mathbf{T}'$ is the $i$th child of $\mathbf{T}'_p$ if (i) $\mathbf{T}_p$ has its partner $\mathbf{T}'_p$ and (ii) $\mathbf{T}'_p$ also have eight children. Otherwise if the condition (i) or (ii) fails, **T** has no partner in $M'$. Any tetrahedron **T** of the type $S_2$, $S_{4a}$, or $S_{4b}$ that is not regular has no partner. However, **T**'s parent $\mathbf{T}_p$ may have its partner because $\mathbf{T}_p$ is regular. Whenever any **T** in $M$ has its partner $\mathbf{T}'$ in $M'$ and **T** is deleted by derefinement, $\mathbf{T}'$ will lose its twin and consequently will lose its partner, and vice versa. With these rules, the partnership is updated after each cycle of refinement/derefinement.

**2.2.2. Determination of intersection candidates—**Whether a parameter or a variable, consider a quantity $Q$ and another quantity $Q'$ discretized on $M$ and on $M'$, respectively, where $Q$ and $Q'$ are coupled in the governing equations. Since constant or piecewise continuous finite element basis functions are used to describe a continuously varying $Q$ and $Q'$, finite element assembly involving both $Q$ and $Q'$ must be done on the intersection between a *leaf* tetrahedron **T** on $M$ and a *leaf* tetrahedron $\mathbf{T}'$ on $M'$. Because of partnership, the assembly can be done either on **T** or on $\mathbf{T}'$ basis. Suppose that the assembly is performed using **T**. There are three cases that need to be considered.

- case 1: **T** has its partner **T**′.
- case 2: **T** is of the type $S_8$ and **T** has no partner.
- case 3: **T** is of the type $S_2$, $S_{4a}$ or $S_{4b}$.

In the first case in which **T** is of type $S_8$, we determine whether its partner **T**′ has children. If **T**′ has children, then all descendant tetrahedrons of **T**′ including **T**′ are completely inside **T**. Therefore, the intersection is trivial, and **T** and **T**′ map directly.

In the second case, if **T** of type $S_8$ has no partner, then any leaf tetrahedron in $M'$ that intersects with **T** either (i) has a lower level of subdivision than **T** or (ii) has the same level as **T** and is of type $S_2$, $S_{4a}$, or $S_{4b}$. In both cases, we traverse up the hierarchy through parents until any one of **T**'s ancestors $\mathbf{T}_a$ that has its partner $\mathbf{T}'_a$ in $M'$ is visited. $\mathbf{T}'_a$ must either (i) have only non-regular child tetrahedrons of type $S_2$, $S_{4a}$, or $S_{4b}$ at the leaf level or (ii) be a leaf tetrahedron of type $S_8$: If $\mathbf{T}'_a$ should have eight regular children of type $S_8$, then the first **T**'s ancestor that has its partner would be found at the deeper refinement level than $\mathbf{T}_a$, which contradicts that $\mathbf{T}_a$ is the first ancestor with its partner. Therefore $\mathbf{T}_a$ cannot have eight regular children. If **T** is determined to be a candidate for intersection with $\mathbf{T}'_a$'s children, then the intersections are computed with respect to the internal triangle faces shared by $\mathbf{T}'_a$'s children if they exist, as shown in Fig. 3. Otherwise, the intersection is trivial.

In the third case in which **T** is a non-regular tetrahedron of the type $S_2$, $S_{4a}$, or $S_{4b}$, we traverse up the hierarchy through parents until any one of **T**'s ancestors $\mathbf{T}_a$ that has its partner $\mathbf{T}'_a$ in $M'$ is visited. There are three possible scenarios regarding $\mathbf{T}'_a$: (i) If $\mathbf{T}'_a$ is a leaf tetrahedron, the intersection is trivial because **T** is inside $\mathbf{T}'_a$. (ii) If $\mathbf{T}'_a$ has non-regular child tetrahedrons of type $S_2$, $S_{4a}$, or $S_{4b}$, then **T** is observed as a candidate for intersections with $\mathbf{T}'_a$'s children. In this case, the intersections are computed with respect to the internal triangle faces shared by $\mathbf{T}'_a$'s children. (iii) Otherwise, $\mathbf{T}'_a$ must have eight regular child tetrahedrons of type $S_8$ which is possible only if $\mathbf{T}_a$ is **T**'s parent $\mathbf{T}_p$. If we suppose that $\mathbf{T}_a$ is not $\mathbf{T}_p$, then $\mathbf{T}_a$ is an ancestor of $\mathbf{T}_p$ and $\mathbf{T}_p$ must be inside one of $\mathbf{T}'_a$'s eight children of type $S_8$. Consequently, one of $\mathbf{T}'_a$'s eight children has one of **T**'s ancestors as its partner which contradicts by *mutual partnership* that $\mathbf{T}_a$ is the first ancestor of **T** with its partner. Therefore, $\mathbf{T}_a = \mathbf{T}_p$ holds. Consequently, $\mathbf{T}'_a$'s leaf-level descendants are the candidates for intersection with $\mathbf{T}_p$'s children and the intersections are computed with respect to the internal triangle faces shared by $\mathbf{T}_p$'s children.

From the above scenarios, one can summarize with the following general rule. If a leaf tetrahedron **T** has a partner, then **T** works as a geometric *container* covering **T**'s partner and all its descendants. Otherwise, the first ancestor of **T** which has a partner **P** in $M'$ can be found by traversing up the parents. If **P** has non-regular children, **P** works as a container of covering **P**'s partner and all its descendants. Otherwise, **P** has eight regular children and **T**'s parent works as a container of **P** and **P**'s children. Therefore, any non-trivial intersecting situations arise from the internal triangle faces in some container tetrahedron **C** that is regular and the tetrahedrons embedded therein which are all leaf-level descendants of **C**'s partner. The container tetrahedron **C** is classified into four different types by Definition 2:

**Definition 2:** If a container **C** is a leaf tetrahedron, **C** is called a type $C_8$-container. If **C** has two children of type $S_2$, **C** is called a type $C_2$-container. If **C** has four children of type $S_{4a}$, **C** is called a type $C_{4a}$-container. Finally, if **C** has four children of type $S_{4b}$, **C** is called a type $C_{4b}$-container.

In Fig. 3, $\mathbf{T}_{ijkl}$ corresponds to the container and an edge *pq* is an edge of one of the leaf-level descendants of $\mathbf{T}_{ijkl}$'s partner. The algorithm handling intersections in nested adaptive tetrahedral dual-mesh is given by Algorithm 1 FINT which is detailed in pseudocode below.

**Algorithm 1:** Fast intersections of nested tetrahedrons (FINT)

Given the nested two tetrahedral meshes *M* and *M*′,

For each leaf tetrahedron **T** on *M* not yet handled,

If **T** has a partner,

Handle intersection using **T** as $C_8$-container.

Else

Search the first ancestor of **T** which has a partner **P** by traversing up the parents.

If **P** has no child,

Handle intersection using **P** as $C_8$-container.

Else If **P** has two children of type $S_2$,

Handle intersection using **P** as $C_2$-container.

Else If **P** has four children of type $S_{4a}$,

Handle intersection using **P** as $C_{4a}$-container.

Else If **P** has four children of type $S_{4b}$,

Handle intersection using **P** as $C_{4b}$-container.

Else If **P** has eight children of type $S_8$,

Denoting **T**'s parent by **Q**,

If **T** is of type $S_2$,

Handle intersection using **Q** as $C_2$-container.

Else if **T** is of type $S_{4a}$,

Handle intersection using **Q** as $C_{4a}$-container.

Else if **T** is of type $S_{4b}$,

Handle intersection using **Q** as $C_{4b}$-container.

End If

End If

End If

End For.

**2.2.3. Computation of intersection points—**Tetrahedron–tetrahedron intersections in nested meshes generated by the 8-subtetrahedron subdivision algorithm provide relatively simple configurations in comparison with non-nested arbitrarily configured tetrahedral meshes. Unfortunately, the accuracy of the intersection computations in real object space is dependent upon the tetrahedron shapes, which results in non-robust computation in real object space. However, we have utilized that the volume coordinate is independent of the tetrahedron shapes and can be used to provide a robust solution to the intersection problem. For a tetrahedron **T** with nodes 0, 1, 2, and 3, the volume coordinate $\phi_i$ of a point $p$ for node $i$ inside the tetrahedron is the volume of the subtetrahedron $\mathbf{T}_{pjkl}$ normalized with the volume $V_{\mathbf{T}}$ of **T**:

$$\phi_i(p)=\text{volume}(p,j,k,l)/V_{\mathrm{T}} \tag{1}$$

for $i, j, k, l = 0,1,2,3$. The volume coordinate is identical to the local linear finite element basis function or *weight*. For fast and robust intersection handling, we define a 4-vector by Definition 3 as

**Definition 3:** For any point $p$ inside a tetrahedron **T**, a 4-vector **p** given by

$$\mathbf{p}=|w_i(p),w_j(p),w_k(p),w_l(p)|_{\mathbf{T}} \tag{2}$$

is called **T**-based representation of $p$, where $w_i(p)$, $w_j(p)$, $w_k(p)$, and $w_l(p)$ are weights of **T**'s node $i, j, k,$ and $l$, respectively, at $p$'s location.

The 4-vector is linear in the space occupied by **T** and four basis vectors $\mathbf{i} = |1, 0, 0,0|_{\mathbf{T}}$, $\mathbf{j} = |0, 1, 0,0|_{\mathbf{T}}$, $\mathbf{k} = |0, 0, 1,0|_{\mathbf{T}}$, and $\mathbf{l} = |0, 0, 0,1|_{\mathbf{T}}$ span the 4-vector space in **T**-based representation. Suppose that four nodes of a container **C** are $i, j, k,$ and $l$. Without loss of generality, we can associate nodes $i, j, k,$ and $l$ with four basis vectors $\mathbf{i} = |1, 0, 0,0|_{\mathbf{C}}$, $\mathbf{j} = |0, 1, 0,0|_{\mathbf{C}}$, $\mathbf{k} = |0, 0, 1,0|_{\mathbf{C}}$, and $\mathbf{l} = |0, 0, 0,1|_{\mathbf{C}}$ in **C**-based representation. Furthermore, the following planes of intersection can be defined:

- **C** of type $C_2$ has a plane of intersection passing the triangle $\mathrm{F}_{jk}$ with the nodes $i, m,$ and $l$ where $m$ is the midpoint of the edge $jk$.
- **C** of type $C_{4a}$ has two planes of intersection, passing the triangle $\mathrm{F}_{jk}$ with the nodes $i, m,$ and $l,$ and the triangle $\mathrm{F}_{il}$ with the nodes $j, n,$ and $k,$ respectively, where $m$ and $n$ are mid-points of the edges $jk$ and $il$.
- **C** of type $C_{4b}$ has three planes of intersection passing the triangle, $\mathrm{F}_j$ with the nodes $i, o,$ and $m,$ the triangle $\mathrm{F}_k$ with the nodes $i, m,$ and $n,$ and the triangle $\mathrm{F}_l$ with the nodes $i, n,$ and $o,$ respectively, where $m, n,$ and $o$ are mid-points of the edge $jk, kl,$ and $lj,$ respectively.

All triangles of interest are illustrated with shaded triangles and node locations in Fig. 3. In the following, the approach to find the intersection for each container type is described:

***2.2.3.1. Intersections in $C_8$-container:*** The intersection is trivial. Each intersection is each leaf tetrahedron **T** embedded in the container **C. T**- and **C**-based 4-vector representations for each node of **T** are used directly for finite element assembly related two leaf tetrahedrons separately from $M$ and $M'$.

***2.2.3.2. Intersections in $C_2$-container:*** The child tetrahedron of a type $C_2$-container **C** with nodes $i, j, m,$ and $l$ located in the back of $\mathbf{F}_{jk}$ is denoted as **L**. As shown in Fig. 3(b), the mid-point $m$ of the edge $jk$ is associated with a 4-vector $\mathbf{m} = |0, 1/2, 1/2,0|_{\mathbf{C}}$ and hence $|1, 0, 0,0|_{\mathbf{C}}$, $|0, 1, 0,0|_{\mathbf{C}}$, $|0, 1/2, 1/2,0|_{\mathbf{C}}$, and $|0, 0, 0,1|_{\mathbf{C}}$ span 4-vectors in the space occupied by **L**. Any point $p$ inside **L** is also associated with a 4-vector $|a,b, c,d|_L$ where $a, b, c,$ and $d$ are weights of **L**'s four nodes $i, j, m,$ and $l$. The point $p$ is represented as $|w_i,w_j,w_k,w_l|_{\mathbf{C}}$ in **C**-based

representation as well. The 4-vector $|w_i,w_j,w_k,w_l|_{\mathbf{C}}$ must be a linear combination of |1, 0, 0,0|$_{\mathbf{C}}$, |0, 1, 0,0|$_{\mathbf{C}}$, |0, 1/2, 1/2,0|$_{\mathbf{C}}$, and |0, 0, 0,1|$_{\mathbf{C}}$, which means that

$$|w_i,w_j,w_k,w_l|_{\mathbf{C}}=a|1,0,0,0|_{\mathbf{C}}+b|0,1,0,0|_{\mathbf{C}}+c|0,1/2,1/2,0|_{\mathbf{C}}+d|0,0,0,1|_{\mathbf{C}}. \tag{3}$$

Therefore, we have $p$'s 4-vector $\mathbf{p}$ as

$$\mathbf{p}=|w_i(p),w_j(p),w_k(p),w_l(p)|_{\mathbf{C}}, \tag{4}$$

$$\mathbf{p}=|w_i(p),d_{jk}(p),2w_k(p),w_l(p)|_{\mathbf{L}}, \tag{5}$$

with $d_{jk}$ defined by

$$d_{jk}(p)=w_j(p)-w_k(p). \tag{6}$$

For the other child tetrahedron **R** with four nodes $i$, $m$, $k$, and $l$ in the front of $\mathbf{F}_{jk}$, |1, 0, 0,0|$_{\mathbf{C}}$, |0, 1/2, 1/2,0|$_{\mathbf{C}}$, |0, 0, 1,0|$_{\mathbf{C}}$, and |0, 0, 0,1|$_{\mathbf{C}}$ span 4-vectors in the space occupied by **R**. In the same manner, a 4-vector $\mathbf{q}$ of any point $q$ inside **R** is obtained in the same way;

$$\mathbf{q}=|w_i(q),w_j(q),w_k(q),w_l(q)|_{\mathbf{C}}, \tag{7}$$

$$\mathbf{q}=|w_i(q),2w_j(q),d_{kj}(q),w_l(q)|_{\mathbf{R}}. \tag{8}$$

The intersection point $r$ between the edge $pq$ and the triangle $\mathbf{F}_{jk}$ can be represented by

$$\mathbf{r}=(1-t)\mathbf{p}+t\mathbf{q}, \quad 0<t<1 \tag{9}$$

because of the linearity. From (5), (6), and (8), one can show that

$$d_{kj}(r)=0 \tag{10}$$

if $r$ is on $\mathbf{F}_{jk}$. Eq. (10) then requires $(1-t)w_j(p)+tw_j(q)=(1-t)w_k(p)+tw_k(q)$ and $t$ becomes

$$t=\frac{d_{kj}(p)}{d_{kj}(q)-d_{kj}(p)}. \tag{11}$$

Eq. (10) expresses the plane equation of $\mathbf{F}_{jk}$ in 4-vector space and any edge with endpoints $p$ and $q$ will intersect if and only if $d_{kj}(p)d_{kj}(q)<0$. Combination of (9) and (11) gives rise to the 4-vector $\mathbf{r}$ in **C**-based representation at the intersection point $r$. At the final output of the presented algorithm, any points inside **C** will be switched into **L** or **R**-based representation using (5) or (8) for finite element assembly related two leaf tetrahedrons separately from $M$ and $M'$.

***2.2.3.3. Intersections in $C_{4a}$-container:*** As shown in Fig. 3(c), there exist two mid-points $m$ and $n$ represented by $\mathbf{m}=|0, 1/2, 1/2,0|_{\mathbf{C}}$ and $\mathbf{n}=|1/2, 0, 0, 1/2|_{\mathbf{C}}$. The four subtetrahedrons of **C** are denoted by **P** with nodes $i$, $j$, $m$, and $n$; **Q** with nodes $i$, $k$, $n$, and $m$; **R** with nodes $l$, $j$, $n$, and $m$; and **S** with nodes $l$, $k$, $m$, and $n$. Then, any 4-vector $\mathbf{p}=|w_i(p), w_j(p), w_k(p), w_l(p)|_{\mathbf{C}}$ inside **P**, **Q**, **R**, or **S** is represented by one of the following 4-vectors, respectively;

$$\mathbf{p}=|d_{il}(p),d_{jk}(p),2w_k(p),2w_l(p)|_{\mathbf{P}}, \tag{12}$$

$$\mathbf{p}=|d_{il}(p),d_{kj}(p),2w_l(p),2w_j(p)|_{\mathbf{Q}}, \tag{13}$$

$$\mathbf{p}=|d_{li}(p),d_{jk}(p),2w_i(p),2w_k(p)|_{\mathbf{R}}, \tag{14}$$

$$\mathbf{p}=|d_{li}(p),d_{kj}(p),2w_j(p),2w_i(p)|_{\mathbf{S}}. \tag{15}$$

The intersection condition can be derived similarly to (10) and reads;

$$d_{li}(r)=0, \tag{16}$$

$$d_{jk}(r)=0, \tag{17}$$

with respect to the triangles $\mathbf{F}_{il}$ and $\mathbf{F}_{jk}$, respectively. Then the intersection point $r$ is obtained using (9) and (11) with $d_{li}$ and $d_{jk}$ defined with respect to $\mathbf{F}_{il}$ and $\mathbf{F}_{jk}$. At the end, any points inside **C** will be switched into **P**, **Q**, **R**, or **S**-based representation using (12)–(15) for finite element assembly related two leaf tetrahedrons separately from $M$ and $M'$.

***2.2.3.4. Intersections in $C_{4b}$-container:*** As shown in Fig. 2(d), there exist three mid-points $m$, $n$, and $o$ represented by $\mathbf{m} = |0,1/2, 1/2,0|_{\mathbf{C}}$, $\mathbf{n} = |0, 0, 1/2, 1/2|_{\mathbf{C}}$, and $\mathbf{o} = |0, 1/2, 0, 1/2|_{\mathbf{C}}$. The four subtetrahedrons of **C** are denoted by **P** with nodes $i$, $j$, $m$, and $o$; **Q** with nodes $i$, $k$, $n$, and $m$; **R** with nodes $i$, $l$, $o$, and $n$; and **S** with nodes $i$, $m$, $n$, and $o$. Following the same convention as above, $|1, 0,0,0|_{\mathbf{C}}$, $|0, 1, 0,0|_{\mathbf{C}}$, $|0, 1/2, 1/2,0|_{\mathbf{C}}$, and $|0, 1/2, 0, 1/2|_{\mathbf{C}}$ span 4-vectors in the space occupied by **P**; $|1, 0, 0,0|_{\mathbf{C}}$, $|0, 0, 1,0|_{\mathbf{C}}$, $|0, 0, 1/2, 1/2|_{\mathbf{C}}$, and $|0, 1/2, 1/2,0|_{\mathbf{C}}$ span 4-vectors in **Q**; $|1, 0, 0,0|_{\mathbf{C}}$, $|0, 0, 0,1|_{\mathbf{C}}$, $|0, 1/2,0, 1/2|_{\mathbf{C}}$, and $|0, 0, 1/2, 1/2|_{\mathbf{C}}$ span 4-vectors in **R**; and finally, $|1, 0, 0,0|_{\mathbf{C}}$, $|0, 1/2, 1/2,0|_{\mathbf{C}}$, $|0, 0, 1/2, 1/2|_{\mathbf{C}}$, and $|0, 1/2, 0, 1/2|_{\mathbf{C}}$ span 4-vectors in **S**.

Using the same analyses carried out above, any 4-vector $\mathbf{p} = |w_i(p), w_j(p), w_k(p), w_l(p)|_{\mathbf{C}}$ inside **P**, **Q**, **R**, or **S** is represented by one of the following 4-vectors, respectively;

$$\mathbf{p}=|w_i(p),d_{j,kl}(p),2w_k(p),2w_l(p)|_{\mathbf{P}}, \tag{18}$$

$$\mathbf{p}=|w_i(p),d_{k,lj}(p),2w_l(p),2w_j(p)|_{\mathbf{Q}}, \tag{19}$$

$$\mathbf{p}=|w_i(p),d_{l,jk}(p),2w_j(p),2w_k(p)|_{\mathbf{R}}, \tag{20}$$

$$\mathbf{p}=|w_i(p), -d_{j,kl}(p), -d_{k,lj}(p), -d_{l,jk}(p)|_{\mathbf{S}}, \tag{21}$$

where $d_{i,jk}$ is defined by

$$d_{i,jk}(p)=w_i(p) - w_j(p) - w_k(p). \tag{22}$$

The plane equations for the triangles $\mathbf{F}_j$, $\mathbf{F}_k$, and $\mathbf{F}_l$ become

$$d_{\alpha,\beta\gamma}(p)=0, \tag{23}$$

with $\alpha, \beta, \gamma = j, k, l$ cyclic, respectively. 4-vectors $\mathbf{r}_j$, $\mathbf{r}_k$, and $\mathbf{r}_l$ for intersection points with respect to $\mathbf{F}_j$, $\mathbf{F}_k$, and $\mathbf{F}_l$ can be derived from (9) and (23), and the resulting interpolation coefficients $t_j$, $t_k$, and $t_l$ are given by

$$t_\alpha=\frac{d_{\alpha,\beta\gamma}(p)}{d_{\alpha,\beta\gamma}(p) - d_{\alpha,\beta\gamma}(q)} \tag{24}$$

with $\alpha, \beta, \gamma = j, k, l$ cyclic. At the end, any points inside **C** will be switched into **P**, **Q**, **R**, or **S**-based representation using (18)–(21) for finite element assembly related two leaf tetrahedrons separately from $M$ and $M'$.

**2.2.4. Weight registering—**The weight-based intersection computation requires 4-vectors in container-based representation. Therefore, each node of any leaf tetrahedron $\mathbf{T}_l$ in $M$ embedded in its container **C** in $M'$ must be registered with a 4-vector in **C**-based representation. The 4-vectors are computed from the partner $\mathbf{T}_a$ of the container **C** where $\mathbf{T}_a$ is one of $\mathbf{T}_l$'s ancestors in $M$. It is noteworthy that direct computations of weights using nodal coordinates are avoided. All leaf tetrahedrons including $\mathbf{T}_l$, inside the volume occupied by **C**, are accessible by downward *preorder traversal* of the tetrahedron subtree rooted at the tetrahedron $\mathbf{T}_a$. If the tetrahedron $\mathbf{T}_c$ currently visited is not a leaf, the *preorder work* is to register weights for split points on edges of $\mathbf{T}_c$. Otherwise if $\mathbf{T}_c$ is a leaf, the preorder work is to resolve intersection and gather disjoint tetrahedron pieces including weight maps and nodal indices that are coupled. The details are as follows: $\mathbf{T}_a$'s four nodes $i_a$, $j_a$, $k_a$, and $l_a$ are represented by $\mathbf{i}_a = |1, 0, 0,0|_{\mathbf{C}}$, $\mathbf{j}_a = |0, 1, 0,0|_{\mathbf{C}}$, $\mathbf{k}_a = |0, 0, 1,0|_{\mathbf{C}}$, and $\mathbf{l}_a = |0, 0, 0,1|_{\mathbf{C}}$, because $\mathbf{T}_a$ is **C**'s twin. Then the recursive weight registering procedure starts with $\mathbf{T}_c = \mathbf{T}_a$ where $\mathbf{T}_c$ denotes the tetrahedron currently visited. Whenever any edge $pq$ of $\mathbf{T}_c$ has a mid-point $m$, a 4-vector **m** for $m$ is registered using (9) with $t = 1/2$, that is, $\mathbf{m} = (\mathbf{p} + \mathbf{q})/2$ and then each child of $\mathbf{T}_c$ is visited. If $\mathbf{T}_c$ is a leaf, intersection is resolved and disjoint tetrahedron pieces including coupling information related with the pieces are collected using the method described in the Section 2.2.3. Weight registering terminates when all leaf tetrahedrons in $M$ embedded in **C** are visited and returns with all intersections in **C** fully resolved.

Fig. 4 illustrates the process of weight registering in 2D triangle meshes that is directly applicable to 3D tetrahedral meshes except that in 2D, we work with *3-vector*s instead of 4-vectors. Assuming that $\mathbf{i} = |1, 0,0|_{\mathbf{C}}$, $\mathbf{j} = |0, 1,0|_{\mathbf{C}}$, and $\mathbf{k} = |0, 0,1|_{\mathbf{C}}$, then 3-vector representation of $l$ is given by $\mathbf{l} = |0, 1/2, 1/2|_{\mathbf{C}}$. Any point $p$ in the space occupied by **C** is associated by a 3-vector $\mathbf{p} = |w_i(p),w_j(p),w_k(p)|_{\mathbf{C}}$ where $w_i(p)$ is $p$'s weight or area coordinate with respect to $i$, etc. We also know that the equation of the line of intersection is given by $d_{jk}(p) = w_j(p) - w_k(p) = 0$ in the 3-vector space and any edge $pq$ intersects with the edge $il$ if $d_{jk}(p)d_{jk}(q) < 0$. Because **C** is the partner of $\mathbf{T}_a = \{n_1,n_2,n_3\}$, we have $\mathbf{n}_1 = |1, 0,0|_{\mathbf{C}}$, $\mathbf{n}_2 = |0, 1,0|_{\mathbf{C}}$, and $\mathbf{n}_3 = |0, 0,1|_{\mathbf{C}}$. Next, we start the preorder traversal down the subtree rooted at $\mathbf{T}_a$. The firstly visited triangle is $\mathbf{T}_a$. Because it is not a leaf, the 3-vectors $\mathbf{n}_4 = |0, 1/2,1/2|_{\mathbf{C}}$, $\mathbf{n}_5 = |1/2, 0, 1/2|_{\mathbf{C}}$, and $\mathbf{n}_6 = |1/2, 1/2,0|_{\mathbf{C}}$ are registered. The secondly visited triangle is the first $\mathbf{T}_a$'s child $\mathbf{T}_1 = \{n_1,n_6,n_5\}$. Because $\mathbf{T}_1$ is a leaf, we check whether it intersects with the edge $il$. We see that the edge $n_6n_5$ intersects with $il$ since $d_{jk}(n_6)d_{jk}(n_5) = -1/4 < 0$ and therefore we resolve intersection for $\mathbf{T}_1$ and obtain disjoint triangle pieces and weight maps for $(\mathbf{T}_1,\mathbf{L})$ and for $(\mathbf{T}_1,\mathbf{R})$. The third visited triangle is $\mathbf{T}_2 = \{n_2,n_4,n_6\}$ which is not a leaf and therefore $\mathbf{n}_7 = |1/4, 1/2, 1/4|_{\mathbf{C}}$, $\mathbf{n}_8 = |1/4, 3/4,0|_{\mathbf{C}}$, and $\mathbf{n}_9 = |0, 3/4, 1/4|_{\mathbf{C}}$ are registered. The fourth visited triangle is a leaf triangle $\mathbf{T}_5 = \{n_6,n_8,n_7\}$ and therefore intersection is checked. Since $d_{jk}(n_6) = 1/2 > 0$, $d_{jk}(n_8) = 3/4 > 0$, and $d_{jk}(n_7) = 1/4 > 0$, $\mathbf{T}_5$ does not intersect with $il$ and is inside **L** since all $d_{jk} \geqslant 0$. We collect $\mathbf{T}_5$ as a triangle piece and obtain weight maps between $\mathbf{T}_5$ and **L**, etc. In the same manner, the next visited three leaf triangles $\mathbf{T}_6$, $\mathbf{T}_7$, and $\mathbf{T}_8$ can be found not to intersect with $il$ and satisfy $d_{jk} \geqslant 0$ for all nodes. Since all children of $\mathbf{T}_2$ have been visited, we next move to $\mathbf{T}_3 = \{n_3,n_5,n_4\}$. Since $\mathbf{T}_3$ is a leaf, we check intersection and find that $d_{jk}(n_3) = -1 < 0$, $d_{jk}(n_5) = -1/2 < 0$, and $d_{jk}(n_4) = 0$. Since $d_{jk} \leq 0$ for all nodes of $\mathbf{T}_3$, $\mathbf{T}_3$ does not intersect $il$. The negative sign denotes that $\mathbf{T}_3$ is inside **R** and we obtain $\mathbf{T}_3$ as a triangle piece and weight maps between $\mathbf{T}_3$ and **R**. Next we visit $\mathbf{T}_4 = \{n_4,n_5,n_6\}$ which is not a leaf, and register $\mathbf{n}_7$; we allow overwriting though $\mathbf{n}_7$ was already registered. Now, only leaf triangles $\mathbf{T}_9$ and $\mathbf{T}_{10}$ are left, and in the same manner as described above, we see that they intersect $il$ and obtain triangle pieces, etc. One should note that the whole procedure described so far is applied to 3D tetrahedral meshes in exactly the same manner.

**2.2.5. Tetrahedron piece collection and weight mapping—**Whenever a leaf tetrahedron **T** on $M$ is visited during weight registration inside the container **C** on $M'$, intersections are considered between **T** and **T**′ on $M'$, where **T**′ is **C** or one of **C**'s children if they exist. Two 4-vectors **u** and **v** in **T**- and **T**′-based representation, respectively, are required to couple **T** and **T**′. If all edges of **T** are non-intersecting, then all four nodes of **T** belong to the space occupied by some **T**′, and **u** and **v** are collected for each node of **T**. If any edge of **T** is intersecting one of the internal triangles of **C**, the 4-vectors **u** and **v** in **T**- and **C**-based representation, respectively, are computed at the intersection point, and two tetrahedron pieces are generated by *bisection* of **T** using the intersection point as a split point. Intersection computation and tetrahedron bisection are repeated for each tetrahedron piece thus generated and new pieces are created until no intersections are encountered. Any resulting tetrahedron piece $\mathbf{T}_r$ belongs to the space occupied by either **T**′ or **T**′'s siblings. After each 4-vector **v** for $\mathbf{T}_r$'s vertices is switched into **T**′ or into **T**′'s sibling-based representation from **C**-based one by using (5), (8), (12)–(15) or (18)–(21), then **u**, **v**, spatial coordinates for each vertex of the tetrahedron piece and two index arrays indicating nodes of **T** and **T**′ (or one of **T**′'s siblings) that are coupled are gathered together. Finally, the gathered information is stored in computer memory, or if required, finite element assembly is performed using the information. Fig. 5 illustrates examples of the set of disjoint tetrahedron pieces generated by FINT.

**2.2.6. Finite element assembly—**FINT gives rise to a set of vertices, 4-vectors **u** and **v** on each vertex of disjoint tetrahedron pieces that completely cover up the intersection of two leaf tetrahedrons **T** and **T**′ from the meshes $M$ and $M'$, respectively. Suppose that $\{\phi_n(\mathbf{x})|n = 0,1,2,\ldots\}$ and $\{\psi_n(\mathbf{x})|n = 0,1,2,\ldots\}$ are linear finite element basis functions in $M$ and $M'$, respectively, where **x** is a real space coordinate vector. For each vertex $\alpha$ of a tetrahedron piece $\Delta$ in the intersection of **T** and **T**′, 4-vectors **u** and **v** are given by

$$\mathbf{u}_\alpha = |\phi_i(\mathbf{x}_\alpha), \phi_j(\mathbf{x}_\alpha), \phi_k(\mathbf{x}_\alpha), \phi_l(\mathbf{x}_\alpha)|_{\mathbf{T}}, \tag{25}$$

$$\mathbf{v}_\alpha = |\psi_{i'}(\mathbf{x}_\alpha), \psi_{j'}(\mathbf{x}_\alpha), \psi_{k'}(\mathbf{x}_\alpha), \psi_{l'}(\mathbf{x}_\alpha)|_{\mathbf{T}'}, \tag{26}$$

where $\{i, j, k, l\}$ and $\{i', j', k', l'\}$ are nodes of **T** and **T**′, respectively, and $\mathbf{x}_\alpha$ is the coordinate vector of $\alpha$. For each vertex $\alpha$ of the piece $\Delta$, a *virtual* linear basis function $\pi_\alpha(\mathbf{x})$ is introduced. $\pi_\alpha(\mathbf{x})$ subtends $\Delta$ and behaves like a usual basis function with

$$\pi_\alpha(\mathbf{x}_\beta) = \delta_{\alpha\beta}, \tag{27}$$

$$\int_\Delta \pi_p^a \pi_q^b \pi_r^c \pi_s^d \mathrm{d}v = 6V_\Delta \frac{a!b!c!d!}{(a+b+c+d+3)!}, \tag{28}$$

where vertices $\{p, q, r, s\}$ and $V_\Delta$ are the vertices and the volume of $\Delta$, respectively; that is, $\pi_\alpha$ is the volume coordinates with respect to $\Delta$. Linear property of $\phi_n(\mathbf{x})$, $\psi_n(\mathbf{x})$, and $\pi_\alpha(\mathbf{x})$ gives rise to the relationships, $\phi_n(\mathbf{x}) = \Sigma_\alpha \phi_n(\mathbf{x}_\alpha)\pi_\alpha(\mathbf{x})$ and $\psi_n(\mathbf{x}) = \Sigma_\alpha \psi_n(\mathbf{x}_\alpha)\pi_\alpha(\mathbf{x})$, enabling two $4 \times 4$ transformation matrices **U** and **V** to be introduced:

$$U_{n\alpha} = \phi_n(\mathbf{x}_\alpha), \tag{29}$$

$$V_{n\alpha} = \psi_n(\mathbf{x}_\alpha). \tag{30}$$

The matrices **U** and **V** perform the change of basis from the virtual basis $\pi$ to the actual basis $\phi$ and $\psi$, respectively, and inside $\Delta$ result in:

$$\phi_i(\mathbf{x}) = \sum_\alpha U_{i\alpha}\pi_\alpha(\mathbf{x}), \tag{31}$$

$$\psi_i(\mathbf{x}) = \sum_{\alpha} V_{i\alpha}\pi_\alpha(\mathbf{x}). \tag{32}$$

Integrations of products that $\phi$, $\psi$, or both are involved in are performed over $\Delta$ for finite element assembly by treating $\pi$ as the usual linear finite element basis function. For example, $\int_\Delta \nabla\phi_i(\mathbf{x}) \cdot \nabla\ \psi_j(\mathbf{x})dv = \Sigma_{\alpha\beta}U_{i\alpha}V_{j\beta}\int_\Delta \nabla\pi_\alpha(\mathbf{x}) \cdot \nabla\pi_\beta(\mathbf{x})dv$, and $\int_\Delta \phi_i(\mathbf{x})\psi_j(\mathbf{x})dv = \Sigma_{\alpha\beta}U_{i\alpha}V_{j\beta}\int_\Delta \pi_\alpha(\mathbf{x})\pi_\beta(\mathbf{x})dv$, and so forth.

## 3. Computational experiments

### 3.1. Forward imaging problem in fluorescence optical tomography

A coupled system of PDE for fluorescence enhanced optical imaging provides an excellent example to assess the finite element assembly cost and the capability for independent refinement/derefinement in the dual-mesh scheme using FINT. The coupled governing equations describing excitation and fluorescence light propagation can be described as [25]

$$[-\nabla \cdot D_x(\mathbf{x})\nabla + k_x]\Phi_x(\mathbf{x},\omega) = 0, \tag{33}$$

$$[-\nabla \cdot D_m(\mathbf{x})\nabla + k_m]\Phi_m(\mathbf{x},\omega) = \beta_{xm}\Phi_x(\mathbf{x},\omega), \tag{34}$$

where $D_{x,m} = 1/[3(\mu_{axi,ami} + \mu_{axf,amf} + \mu'_{sx,sm})]$, $k$, $k_{x,m} = i\omega/c + \mu_{axi,ami}(\mathbf{x}) + \mu_{axf,amf}(\mathbf{x})$, and $\beta_{xm} = q\mu_{axf}/[1 - i\omega\tau\ (\mathbf{x})]$. An index $x$ denotes the excitation field and $m$ denotes the emission field; $\Phi_x, \Phi_m$ are photon fluence fields at excitation and emission wavelengths, respectively; $D_{x,m}$ are photon diffusion coefficients; $\mu_{axi,ami}$ is the absorption coefficient due to endogenous chromophores; $\mu_{axf,amf}$ is the absorption coefficient due to exogenous fluorophores; $\mu'_{sx,sm}$ is the reduced scattering coefficient; $\omega = 2\pi f$ where $f$ is the modulation frequency; $q$ is the quantum efficiency of the fluorophore; finally, $\tau$ is the lifetime of fluorescence. Eqs. (33) and (34) are solved under the Robin-type boundary conditions

$$2D_x\mathbf{n} \cdot \nabla\Phi_x + \gamma\Phi_x + S(\mathbf{x}) = 0, \tag{35}$$

$$2D_m\mathbf{n} \cdot \nabla\Phi_m + \gamma\Phi_m = 0, \tag{36}$$

where $\mathbf{n}$ is the unit outward surface normal vector, $\gamma$ is a constant depending only upon the refractive index mismatch on the boundary, and $S(\mathbf{x})$ is the excitation boundary source distribution.

In order to evaluate FINT, we considered excitation and emission fluence solutions for two cases: (1) adaptive single mesh discretizing both $\Phi_x$ and $\Phi_m$ in which the combination of the spatial gradients of $\Phi_x$ and $\Phi_m$ determined refinement/derefinement and (2) two adaptive meshes separately discretizing $\Phi_x$ and $\Phi_m$ in which the spatial gradients of $\Phi_x$ and $\Phi_m$ independently determined refinement/derefinement. The geometry considered was an otherwise optically homogeneous sphere (chosen to challenge curvilinear surface representation) embedded with two fluorescent spheres. The “background” sphere was 5 cm in diameter and centered at the ($x$,$y$, $z$) coordinate origin. The embedded two spheres were 1 cm in diameter and centered at (1.25 cm, 0, 1.25 cm) and (− 1.25 cm, 0, 0), respectively. A point source of modulated excitation light was at the surface location of (0, 0, 2.5 cm). For simplicity, we assumed optical properties of $\mu_{axi} = \mu_{ami} = \mu_{amf} = 0.01\ \text{cm}^{-;1}$, $\mu'_{sx} = \mu'_{sm} = 10.0\ \text{cm}^{-1}$, and $\mu_{axf} = 1.0\ \text{cm}^{-1}$. The other parameters were assumed to be $f = 100$ MHz, $q = 0.1$, $\tau = 0.1$ ns, and refractive index $n = 1.33$. As an *a posteriori* error estimator for refinement/derefinement of tetrahedral meshes, we used an error estimator based upon Kelly’s flux jump criterion [26]

$$\varepsilon[f]=h\int_A \left|\frac{\partial f}{\partial n}\right|_{\pm}^{2} da, \tag{37}$$

across the triangle facet of the tetrahedral element, where $h$ is the height of the tetrahedron, $A$ is the area of the triangle, and $|\partial f/\partial n|_{\pm}$ is the jump of the normal derivative of $f$ across the triangle facet. Eq. (37) measures the spatial curvature of the variable $f$ of interest. For the present case, we relate $f$ with a fluence field $\Phi$. In the single-mesh, we took the element error with $\varepsilon_s = \varepsilon[\Phi_x/\max\{|\Phi_x|\}] + \varepsilon_m[\Phi_m/\max\{|\Phi_m|\}]$ to put an equal significance to $\Phi_x$ and $\Phi_m$, by which the element is chosen for refinement if $\varepsilon_s > \delta_{s,r}\max\{\varepsilon_s\}$ or derefinement if $\varepsilon_s < \delta_{s,d}\max\{\varepsilon_s\}$. In the dual-mesh, we took the element error in the first mesh (in which $\Phi_x$ is solved) with $\varepsilon_x = \varepsilon[\Phi_x]$, and the element error in the second mesh (in which $\Phi_m$ is solved) with $\varepsilon_m = \varepsilon[\Phi_m]$, by which the element in the first mesh is chosen for refinement if $\varepsilon_x > \delta_{x,r}\max\{\varepsilon_x\}$ or derefinement if $\varepsilon_x < \delta_{x,d}\max\{\varepsilon_x\}$, and the element in the second mesh is chosen for refinement if $\varepsilon_m > \delta_{m,r}\max\{\varepsilon_m\}$ or derefinement if $\varepsilon_m < \delta_{m,d}\max\{\varepsilon_m\}$. The refinement/derefinement factors chosen are $\delta_{s,r} = 10^{-4}$, $\delta_{s,d} = 10^{-8}$, $\delta_{x,r} = 10^{-8}$,$\delta_{x,d} = 10^{-14}$, $\delta_{m,r} = 10^{-1}$, and $\delta_{m,d} = 10^{-8}$. In both single- and dual-mesh schemes, we solved the excitation equation first, passed the value of the excitation fluence field to the emission equation and solved the emission field. In the single-mesh, the finite element assembly was executed directly using the elements in excitation and emission problems, respectively. In the dual-mesh scheme, finite element assembly in the excitation problem was executed by passing elements in the first mesh to FINT as input tetrahedrons, while elements in the second mesh were passed to FINT as input tetrahedrons for finite element assembly in the emission problem. All computations were done using the PC with 3.6 GHz Pentium 4 processor and 3.25 GB RAM.

### 3.2. Inverse problem in fluorescence optical tomography

FINT will be applied primarily for inverse imaging problems. The second computational experiment illustrates the application of FINT toward dual-adaptive mesh-based fluorescence tomography. A cylindrical phantom with 5 cm diameter and 6 cm height was simulated with a 5 mm diameter spherical fluorescent target located at the off-center position, (1.0 cm, 0.0, 1.0 cm). Synthetic frequency domain fluorescence measurements were collected for 24 sources and 24 detectors positioned in three concentric rings with eight sources and eight detectors each. The source-detector rings were positioned 1.5 cm apart in the axial direction. Simulation geometry is depicted in Fig. 6. The optical properties were assumed to resemble that of 1% Liposyn solution and the fluorescence contrast corresponded to a 0.5 μM indocyanine green dye in the tumor equivalent to $\mu_{axf} = 0.1\ \text{cm}^{-1}$. For more details on optical properties, we refer the reader to [27]. A trust region Gauss–Newton image reconstruction algorithm was used to invert the synthetic measurements.

## 4. Results and discussion

### 4.1. Forward problem simulation results

For the forward simulation experiment described in Section 3.1, starting from an initial coarse mesh with 3760 tetrahedral elements and 791 nodes, five refinement/derefinement iterations were executed. The initial mesh and iteratively adapted meshes after three and five refinements/derefinements are illustrated in Fig. 7. The changes in the number of tetrahedral elements and nodes for the (i) single mesh-based discretization of both excitation/emission fields, (ii) dual-mesh discretization of excitation field, and (iii) dual-mesh discretization of the emission field are summarized in Tables 1–3, respectively. At the first adaptation, global refinements were applied for both the single and the dual meshes, and the selective refinement/derefinement according to the error estimates were performed from the second adaptation onwards. For the single mesh, the mesh obtained after five iterations of refinement/derefinement had 376,565 tetrahedral elements and 64,820 nodes. After five iterations of the dual-mesh adaptation, the

first mesh which was refined/derefined on the basis of $\Phi_x$ and discretizing $\Phi_x$, $\mu_{axi}$, and $\mu'_{sx}$ had 426,331 tetrahedral elements and 73,990 nodes and the second mesh which was refined/derefined on the basis of $\Phi_m$ and discretizing $\Phi_m$, $\mu_{ami}$, $\mu_{amf}$, $\mu_{axf}$, and $\mu'_{sm}$ had 367,847 tetrahedrons and 63,200 nodes.

The FEM solutions in single-mesh and dual-mesh method for the phase and amplitude of $\Phi_x$ and $\Phi_m$ are illustrated in Fig. 8 with alpha-blended 3D isosurface plots. The solutions from the single-mesh and dual-mesh approaches shown in Fig. 8 are identical, demonstrating that we can successfully decouple and solve the coupled field problems using two independently self-adapting dual meshes if FINT is employed. The results show that FINT gives rise to the exact coupling between the two independently self-adapting meshes and therefore enables the separate adaptive finite element analysis for each field, bringing flexibility into the adaptive finite element methods using tetrahedral elements.

The finite element assembly time cost is illustrated in Fig. 9, where the assembly cost includes all operations in FINT, such as the identification of the container, tree traversal, tetrahedron piece generation, and actual finite element assembly. For the refined/derefined dual-mesh, the overall finite element assembly times in the dual-mesh scheme for $\Phi_x$ and $\Phi_m$ were approximately linear in the total number of tetrahedral elements involved in the assembly as shown in Fig. 9. The patterns of assembly cost in the first and in the second mesh shown in Fig. 9(a), with respect to number of elements in the first and the second mesh, respectively, match the patterns of the number of tetrahedron pieces shown in Fig. 9(b). It is noteworthy that the assembly time required for $\Phi_x$ and $\Phi_m$ in the dual-mesh is proportional to the number of disjoint tetrahedron pieces generated based upon the first and the second mesh, respectively, as illustrated in Fig. 9(c), which indicates the cost of FINT operations including container identification, tree traversal, and tetrahedron piece collection is marginal and the whole computational cost is dominated by the actual finite element assembly step.

### 4.2. Inverse image reconstruction results

For the inverse imaging computational experiment described in Section 3.2, the initial parameter mesh consisted of 187 nodes and 754 tetrahedrons. This mesh was used for discretizing the $\mu_{axf}$. For forward simulations the initial parameter mesh was duplicated and refined once toward the cylinder boundary to provide 643 nodes and 2927 tetrahedrons for solving the coupled photon diffusion equations. A trust region Gauss–Newton algorithm was used to iteratively update the parameter map. Both the forward and parameter meshes were adaptively refined along the Gauss–Newton iterations.

To drive adaptive mesh refinement, we introduced a smoothness measure of the spatial distribution of $\mu_{axf}$ where the measure is checked after every five successive iterative reconstructions to determine whether or not to trigger refinements [28]. If a significant change in $\mu_{axf}$ was detected, the dual-mesh adaptation was performed where the two meshes discretizing fluence fields and the unknown parameter $\mu_{axf}$ were locally refined using Kelly's criterion (37) for the fluence fields and $\mu_{axf}$, respectively. In practice, derefinement was found to be unnecessary since the overly refined situations were avoided using the smoothness check. However it must be noted that these image reconstructions have been performed with simulated measurements, under realistic conditions of low SNR fluorescence measurements, and the absence of prior knowledge about the interior of the imaged domain, derefinement might be necessary in obtaining suitable meshes, and for avoiding spurious mesh refinements in the first few Newton steps. The image reconstruction algorithm and the image smoothness measure used for mesh refinements are detailed elsewhere in the description of dual-adaptive mesh-based fluorescence optical tomography [28].

One hundred Gauss–Newton iterations were carried out during which the forward mesh refinement was triggered 12 times, while the parameter mesh was locally refined 10 times. Final forward and parameter meshes are depicted in Fig. 10. Reconstructed fluorescent target is depicted via slice plots and isosurfaces drawn at FWHM(50% of the maximum contour levels) are illustrated in Fig. 11. The variations in the number of nodes and elements due to refinements in the forward and the parameter meshes are illustrated in Fig. 12. Total image reconstruction time was 7.28 min on a PC with 3.6 GHz Pentium 4 processor and 3.25 GB RAM. The total computational cost of the image reconstruction algorithm is dominated by the forward problem solution cost [28]. Final parameter mesh consisted of only 608 nodes, while still providing diffusion length limited resolution in the vicinity of the reconstructed target demonstrating the power of adaptive mesh refinement- based inverse imaging.

## 5. Summary and conclusions

We have presented and demonstrated an algorithm called FINT for handling intersections in the nested conforming tetrahedral environments that enables the adaptive dual-mesh-based finite element methods. The motivation of this work is to obtain an efficient dual-mesh coupling scheme for development of fully adaptive three-dimensional fluorescence enhanced optical tomography using two tetrahedral meshes generated by 8-subtetrahedron subdivision. The key strength of FINT algorithm lies in using nested meshes and weight registering using linear finite element basis functions that significantly facilitates the intersection handling and weight mapping between tetrahedral elements from the two separate meshes. FINT can be applied not only to the polyhedral regions but also to the geometry with smooth curved surfaces, a benefit that arises from 4-vector- based intersection and interpolation schemes. FINT produces information on a set of disjoint tetrahedrons with weight maps of tetrahedral elements from two separate meshes. Finite elements are assembled on the tetrahedron piece by piece basis using virtual linear basis functions defined on each piece. We have also demonstrated that FINT provides a fast and robust tool to couple two meshes while allowing them to be independently refined or derefined as evidenced in the coupled system of excitation/emission photon diffusion equations. The finite element assembly time on dual meshes with FINT increased linearly with the number of elements against the quadratic increase which will result from a brute force intersection computation. We also demonstrated the application of FINT for a simple fluorescence optical tomography problem. Detailed discussions of mesh refinement criteria and tomography algorithm development have been reported elsewhere [28]. This work represents the first step for incorporating tetrahedral FEM strategies in an array of parameter estimation problems, including that of fluorescence optical tomography and other medical, model-based tomography approaches. The dual-adaptive strategies enabled by FINT algorithm can increase the computational performance of recently proposed shape-based inverse imaging algorithms [29,30].

## Acknowledgements

This work was supported by National Institute of Health (NIH R01 CA112679).

## References

1. Arridge SR, Schweiger M, Hiraoka M, Delpy DT. A finite element approach for modeling photon transport in tissue. Med Phys 1993;20:299–309. [PubMed: 8497214]
2. Woo EJ, Hua P, Webster JG, Tompkins WJ. Finite element method in electrical impedance tomography. Med Biol Eng Comput 1994;32:530–536. [PubMed: 7845069]
3. Paulsen KD, Meaney PM, Moskowitz MJ, Sullivan JM Jr. A dual mesh scheme for finite element based reconstruction algorithms. IEEE Trans Med Imaging 1995;14:504–514. [PubMed: 18215855]

4. Jiang H, Paulsen K, Osterberg U, Patterson M. Frequency-domain near-infrared photo diffusion imaging: initial evaluation in multi-target tissue-like phantoms. Med Phys 1998;25:183–193. [PubMed: 9507478]
5. Schweiger M, Arridge SR. Optical tomographic reconstruction in a complex head model using a priori region boundary information. Phys Med Biol 1999;44:2703–2721. [PubMed: 10588279]
6. Dehghani H, Pogue BW, Shudong J, Brooksby B, Paulsen KD. Three dimensional optical tomography: resolution in small-object imaging. Appl Opt 2003;42:3117–3129. [PubMed: 12790463]
7. Huang M, Zhu Q. Dual-mesh tomography reconstruction method with a depth correction that uses a priori ultrasound information. Appl Opt 2004;43:1654–1662. [PubMed: 15046168]
8. Verfürth, R. A Review of A Posteriori Error Estimation and Adaptive Mesh Refinement Techniques. Wiley/Teubner; New York, Stuttgart: 1996.
9. Johnson CR, MacLeod RS. Nonuniform spatial mesh adaptation using a posteriori error estimates: applications to forward and inverse problems. Appl Numer Math 1994;14:311–326.
10. Verfürth R. A posteriori error estimators for convection–diffusion equations. Numer Math 1998;80:641–663.
11. Becker, R.; Rannacher, R. A Feed-back Approach to Error Control in Finite Element Methods: Basic Analysis and Examples. IWR; 1996.
12. Kaltenbacher B. Regularization by projection with a posteriori discretization level choice for linear and nonlinear ill-posed problems. Inv Prob 2000;16:1523–1539.
13. Li R, Liu W, Ma H, Tang T. Adaptive finite element approximation for distributed elliptic optimal control problems. SIAM J Control Optim 2002;41:1321–1349.
14. Yeung F, Levinson SF, Fu D, Parker KJ. Feature-adaptive motion tracking of ultrasound image sequences using a deformable mesh. IEEE Trans Med Imaging 1998;17:945–956. [PubMed: 10048851]
15. Beilina L. Adaptive finite element method for an inverse scattering problem. J Inverse Problems Inform Technol 2003;1
16. Rivara MC, Levin C. A 3-d refinement algorithm suitable for adaptive and multigrid techniques. Commun Appl Numer Methods 1992;8:281–290.
17. Molinari M, Cox SJ, Blott BH, Daniell GJ. Adaptive mesh refinement techniques for electrical impedance tomography. Physiol Meas 2001;22:91–96. [PubMed: 11236895]
18. Molinari M, Blott BH, Cox SJ, Daniell GJ. Optimal imaging with adaptive mesh refinement in electrical impedance tomography. Physiol Meas 2002;23:121–128. [PubMed: 11876225]
19. Molinari M, Cox SJ, Blott BH, Daniell GJ. Comparison of algorithms for non-linear inverse 3D electrical tomography reconstruction. Physiol Meas 2002;23:95–104. [PubMed: 11876245]
20. Joshi, A.; Sevick-Muraca, EM. Adaptive finite element methods for distributed parameter system identification. Proceedings of the 2004 American Control Conference, American Automatic Control Council; Boston, MA. 2004. p. 2263-2266.
21. Joshi A, Bangerth W, Sevick-Muraca EM. Adaptive finite element based tomography for fluorescence enhanced optical imaging in tissue. Opt Express 2004;12:5402–5417.
22. Joshi A, Bangerth W, Sevick-Muraca EM. Non-contact fluorescence optical tomography with scanning patterned illumination. Opt Express 2006;14:6516–6535.
23. Liu A, Joe B. Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. Math Comput 1996;65:1183– 1200.
24. Plaza A, Padron MA, Carey GF. A 3D refinement/derefinement algorithm for solving evolution problems. Appl Numer Math 2000;32:401–418.
25. Sevick-Muraca, EM.; Kuwana, E.; Godavarty, A.; Houston, JP.; Thompson, AB.; Roy, R. Biomedical Photonics Handbook. CRC Press; 2003. Near infrared fluorescence imaging and spectroscopy in random media and tissues. Chapter 33
26. Kelly DW, De JP, Gago SR, Zienkiewicz OC, Babuska I. A posteriori error analysis and adaptive processes in the finite element method: Part I – Error analysis. Int J Numer Methods Eng 1983;19:1593–1619.

27. Joshi A, Bangerth W, Hwang K, Rasmussen JC, Sevick-Muraca EM. Fully adaptive FEM based fluorescence optical tomography from time-dependent measurements with area illumination and detection. Med Phys 2006;33:1299–1310. [PubMed: 16752565]

28. Lee JH, Joshi A, Sevick-Muraca EM. Fully adaptive finite element based tomography using tetrahedral dual-meshing for fluorescence enhanced optical imaging in tissue. Opt Express 2007;15:6955–6975.

29. Dorn O, Miller EL, Rappaport CM. Shape reconstruction method for electromagnetic tomography using adjoint fields and level sets. Inv Prob 2000;16:1119–1156.

30. Dorn O, Lesselier D. Level set methods for inverse scattering. Inv Prob 2006;22:R67–R131. 5798.

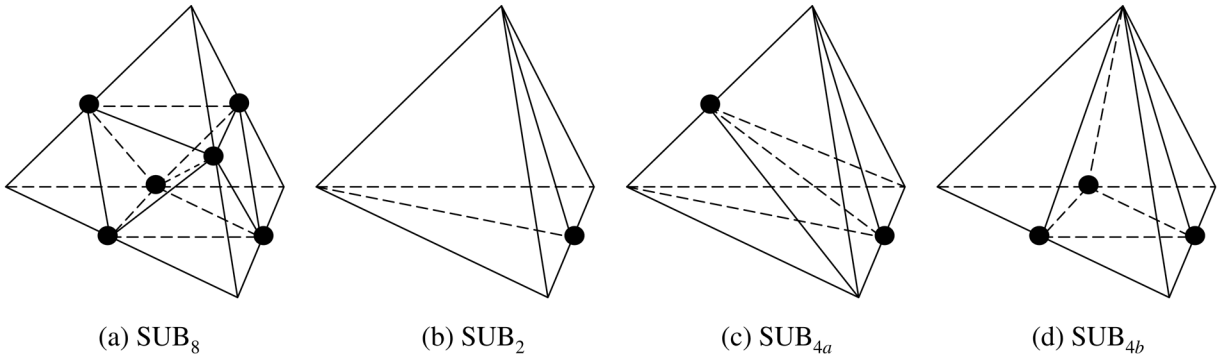


**Fig. 1.**

Four types of tetrahedron subdivisions permitted in the 8-regular subtetrahedron subdivision algorithm [15]. Filled circles are termed split points. (a) Regular 8-subtetrahedron subdivision of a tetrahedron with one split point on each edge; (b) a non-regular subdivision of a tetrahedron with one split point; (c) a non-regular subdivision of a tetrahedron with two split points on a pair of opposite edges; and (d) a non-regular subdivision of a tetrahedron with three split points on the same face.

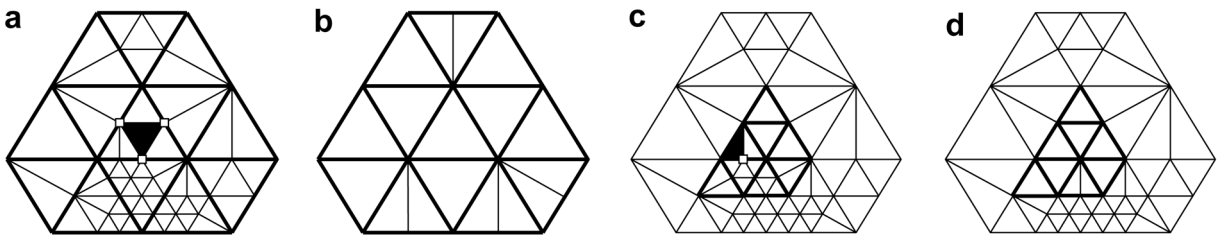


**Fig. 2.**

A 2D example providing an overview of our derefinement algorithm. In the figure, a shaded triangle denotes the element to be derefined and the open symbols mark *dead* nodes which trigger derefinement. When the shaded triangle element is to be derefined, the affected elements are the children of the elements with thick edges. (a) A shaded leaf triangle to derefine in the center; (b) the derefined mesh created by deletion of the shaded triangle in (a). (c) A shaded leaf triangle chosen for derefinement in the lower left from the center element, and (d) the derefined mesh resulting by deletion of the shaded triangle in (c). The derefinement algorithm for 2D triangle meshes can be extended to the derefinement algorithm for 3D tetrahedral meshes.

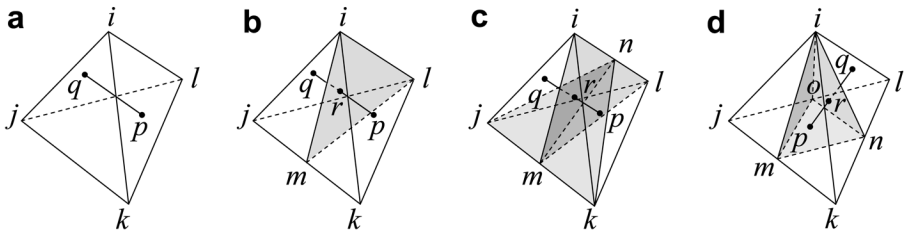


**Fig. 3.**

Configurations of the intersection between a specific internal triangle plane and a edge *pq* where *r* is an intersection point. The regular tetrahedron $\mathbf{T}_{ijkl}$ is a container tetrahedron and (a) a leaf tetrahedron of type $S_8$, (b) a parent of type $S_2$ tetrahedrons, (c) a parent of type $S_{4a}$ tetrahedrons, and (d) a parent of type $S_{4b}$ tetrahedrons. Intersections are checked with respect to the shaded internal triangles (a) triangle $\mathbf{F}_{jk}$ passing *i*, *m*, and *l*, (b) triangle $\mathbf{F}_{jk}$ passing *i*, *m*, and *l*, and triangle $\mathbf{F}_{il}$ passing *j*, *n*, and *k*, (c) triangle $\mathbf{F}_j$ passing *i*, *o*, and *m*, triangle $\mathbf{F}_k$ passing *i*, *m*, and *n*, and triangle $\mathbf{F}_l$ passing *i*, *n*, and *o*, of the tetrahedron $\mathbf{T}_{ijkl}$.

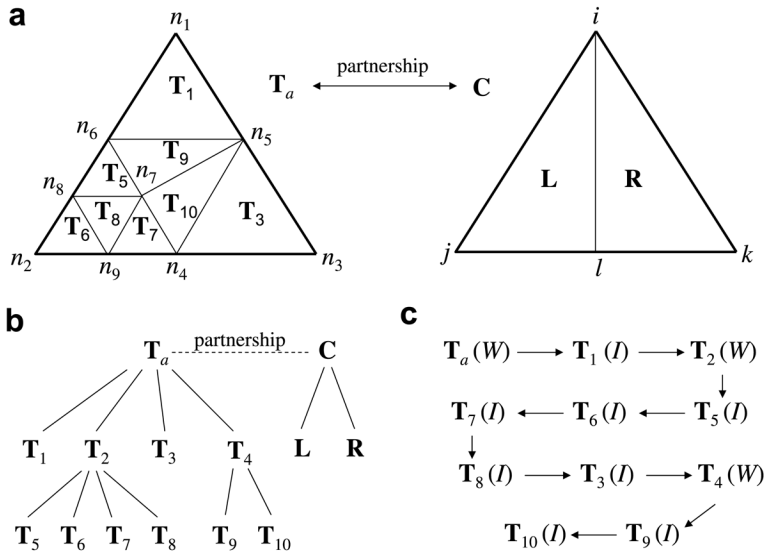


**Fig. 4.**

Illustration of weight registering in 2D triangle meshes which is extended directly to 3D tetrahedral meshes. (a) Two intersecting sub-meshes in *M* (left) and *M′* (right); (b) the triangle subtree rooted at $\mathbf{T}_a$ where $\mathbf{T}_a$ is the ancestor with its partner **C** that is first visited when traversing upward the triangle tree through parents from a leaf triangle, for example, $\mathbf{T}_3$, or $\mathbf{T}_9$, etc. (c) The sequence of visiting triangles in *preorder traversal* of the subtree. **L** and **R** are two *non-regular* children of the container triangle **C** and the edge *il* lies on the line of intersection. *W*and *I* indicate the *preorder work* that is performed when visiting each triangle, where *W* registers weight on the mid-point of edges of the triangle currently visited and *I* resolves intersection and obtains disjoint triangle pieces including weight maps and nodal indices of the two triangles that are coupled.

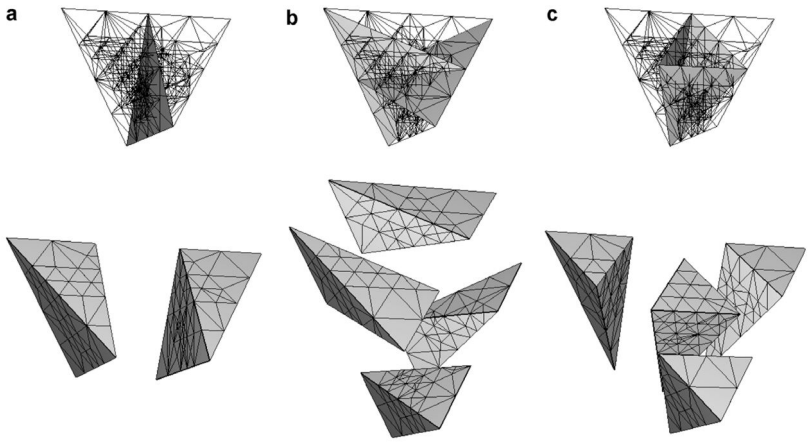


**Fig. 5.**

Illustration of three sets of disjoint tetrahedron pieces generated by FINT for a given set of tetrahedrons embedded in its container (a) of type $C_2$, (b) of type $C_{4a}$, and (c) of type $C_{4b}$, respectively.

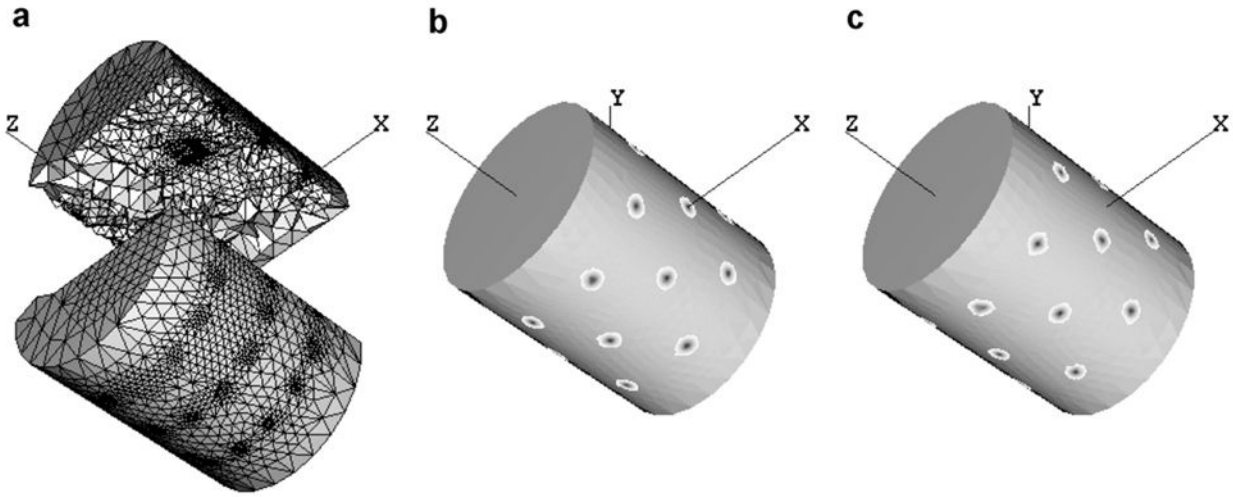


**Fig. 6.**
Simulated frequency domain fluorescence measurements to assess the inverse imaging application of FINT. (a) The mesh used to create simulated boundary measurement; (b) 24 source locations; and (d) 24 detector locations.

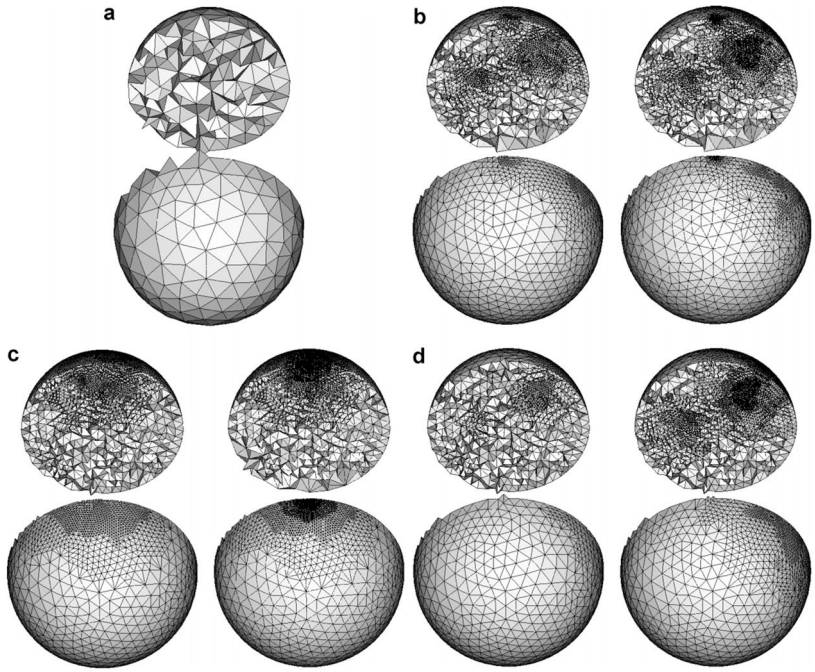


**Fig. 7.**

Tetrahedral mesh refinement/derefinement evolutions from (a) an initial mesh. For each two meshes shown in (b)–(d), the left and right meshes have been obtained after the third and fifth adaptive iteration, respectively. The illustrated in (b)–(d) are the evolutions: in (b) a single mesh in which gradients of $\{\Phi_x, \Phi_m\}$ were used as criteria for refinement/derefinement; in (c) the first of a dual-mesh in which the excitation Eq. (33) was solved and gradients of $\Phi_x$ were used as a criteria for refinement/derefinement; and in (d) the second of the dualmesh in which the emission Eq. (34) was solved and gradients $\Phi_m$ were used as a criteria for refinement/derefinement. The meshes were globally refined using the regular subdivision at the first iterations. In (b) and (c), regular elements of the subdivision level zero or non-regular elements of the subdivision level one are visible around the opposite side of the illuminated position, illustrating that small portions of the meshes were derefined.

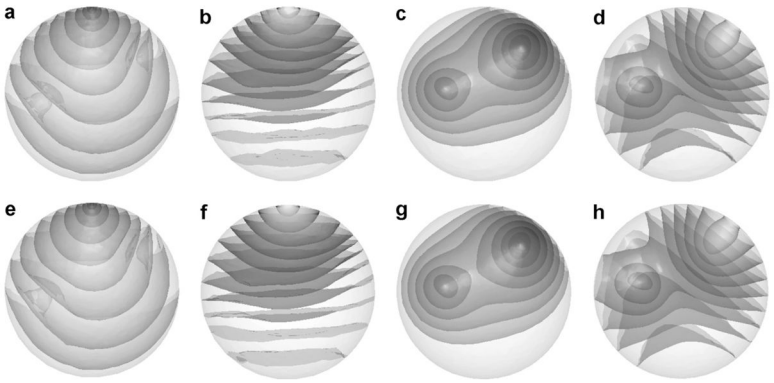


**Fig. 8.**

Isosurface plots for amplitude and phase of $\Phi_x$ and $\Phi_m$ in the single- and dual–mesh schemes using the fifth refined/derefined meshes where (a)–(d) are results from the single-mesh scheme, and (e)–(h) are results from the dual-mesh scheme. The figures shown are the plots of (a) $\log_{10}|\Phi_x|$, (b) phase$[\Phi_x]$, (c) $\log_{10}|\Phi_m|$, and (d) phase$[\Phi_m]$, (e) $\log_{10}|\Phi_x|$, (f) phase$[\Phi_x]$, (g) $\log_{10}|\Phi_m|$, and (h) phase$[\Phi_m]$.

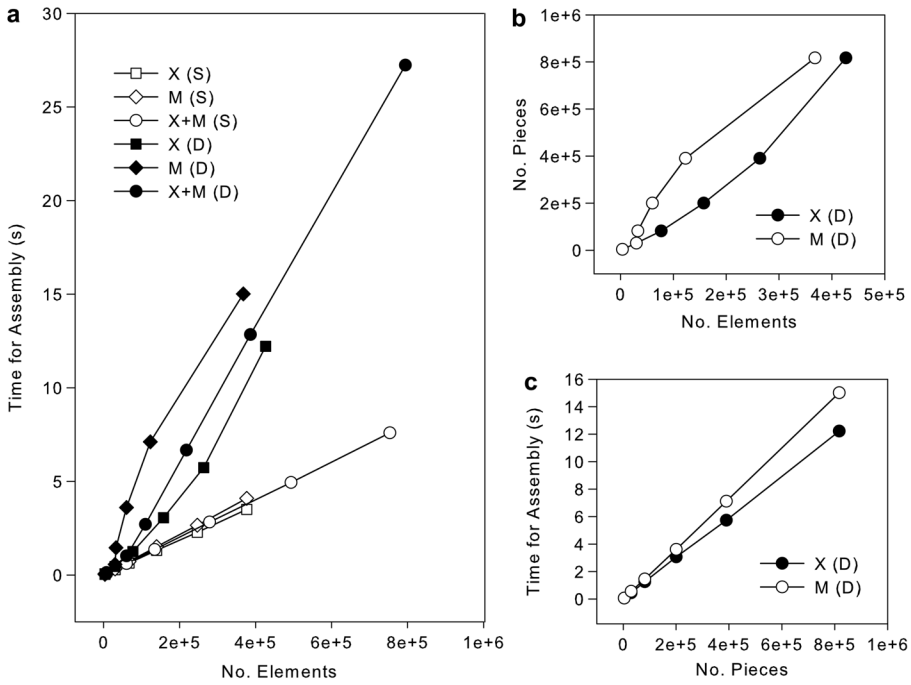


**Fig. 9.**

Time required for finite element assembly in single-mesh (S) and dual-mesh (D) scheme: (a) assembly times with respect to the number of elements, (b) the number of disjoint tetrahedron pieces generated by FINT with respect to the number of elements, and (c) the assembly time with respect to the number of the tetrahedron pieces obtained by FINT. X andM mean the excitation field and the emission field, respectively. X +M indicates that elements and times are counted by adding the counts from X and from M.

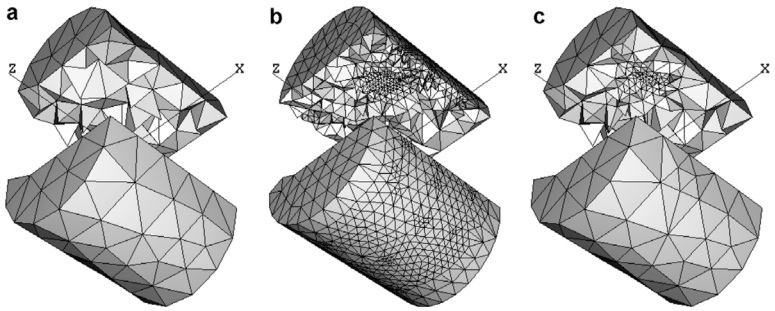


**Fig. 10.**
Final forward and parameter meshes obtained from refinements of the initial coarse mesh shown in (a) through 100 Gauss–Newton iterations where (b) and (c) show the final forward and parameter meshes, respectively.

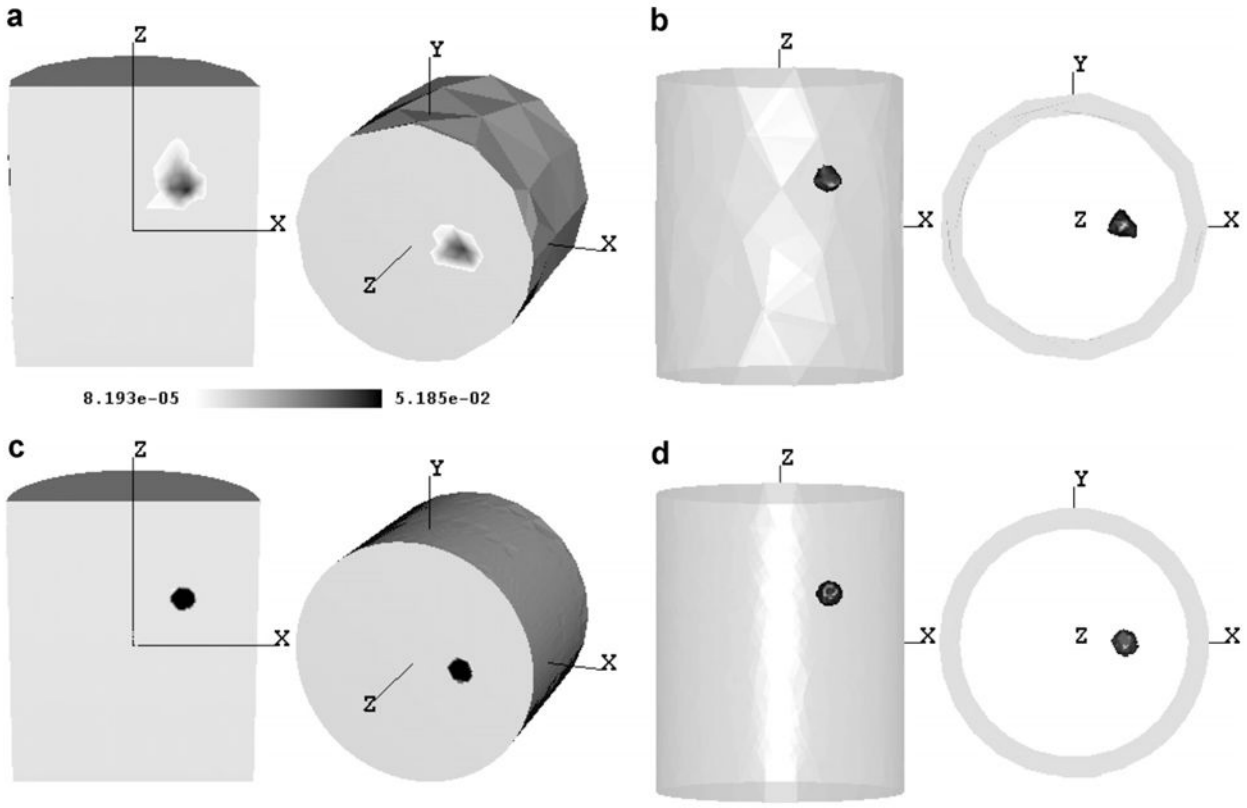


**Fig. 11.**

Reconstructed and actual fluorescent targets in slice and isosurface plot, where (a) and (b) show reconstructed target while (c) and (d) show actual target. The isosurface represents FWHM (50% of the maximum contour levels).

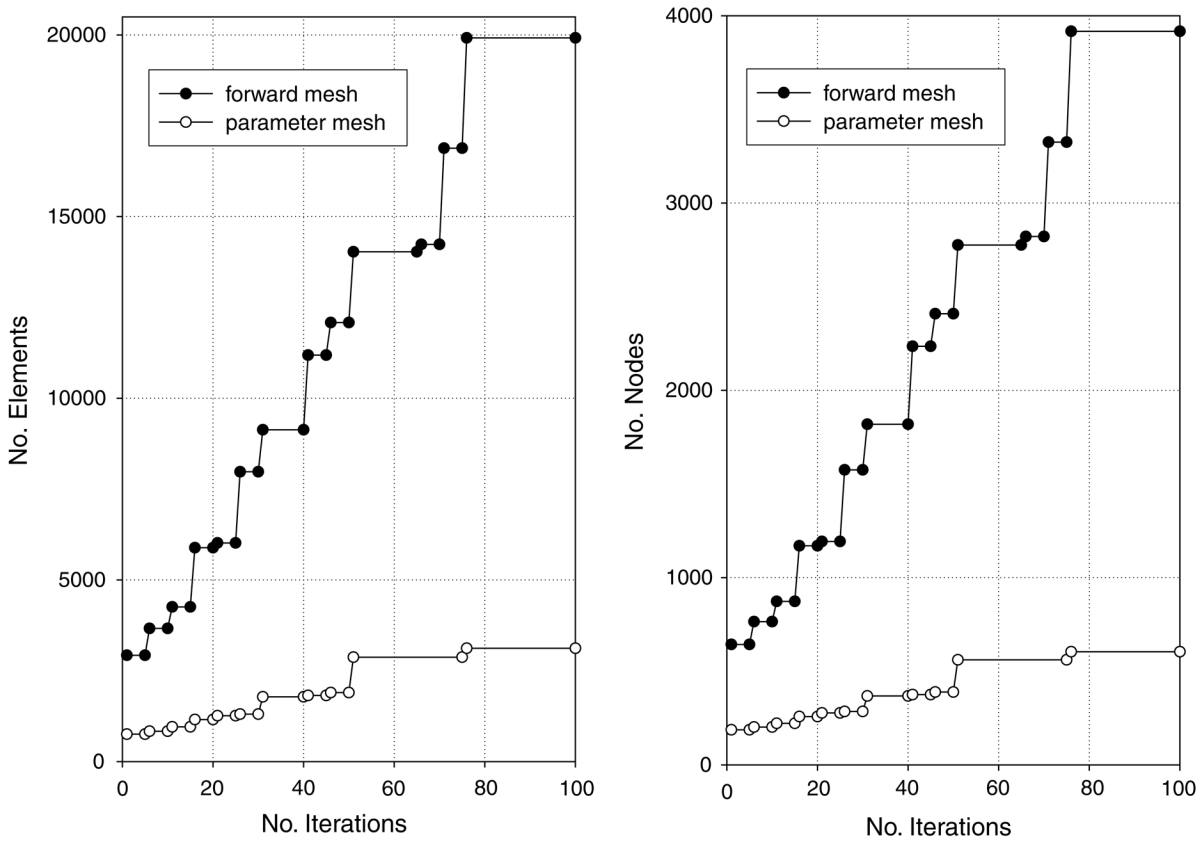


**Fig. 12.**

The variations in the number of elements and nodes in the forward and the parameter mesh with respect to the number of the iterative reconstructions performed.

**Table 1**

Change in the number of elements and nodes in the adaptive single mesh used for solutions of the coupled photon diffusion problem (E = elements, N = nodes, N/A = not applied)

| Iteration | Start | | Refinement | | Derefinement | |
|---|---|---|---|---|---|---|
| | E | N | E | N | E | N |
| 1 | 3760 | 791 | 30,080 | 5600 | N/A | N/A |
| 2 | 30,080 | 5600 | 69,822 | 12,481 | 67,158 | 11,971 |
| 3 | 67,158 | 11,971 | 139,441 | 24,319 | N/A | N/A |
| 4 | 139,441 | 24,319 | 246,480 | 42,600 | N/A | N/A |
| 5 | 246,480 | 42,600 | 376,565 | 64,820 | NA | N/A |
| 6 | 376,565 | 64,820 | N/A | N/A | NA | N/A |

**Table 2**

Change in the number of elements and nodes in the first mesh used for solution of the excitation fluence field in coupled photon diffusion problem with the adaptive dual mesh scheme (E = elements, N = nodes, N/A = not applied)

| Iteration | Start | | Refinement | | Derefinement | |
|---|---|---|---|---|---|---|
| | E | N | E | N | E | N |
| 1 | 3760 | 791 | 30,080 | 5600 | N/A | N/A |
| 2 | 30,080 | 5600 | 77,254 | 13,816 | N/A | N/A |
| 3 | 77,254 | 13,816 | 157,765 | 27,765 | 157,601 | 27,731 |
| 4 | 157,601 | 27,731 | 263,924 | 46,103 | 263,582 | 46,035 |
| 5 | 263,582 | 46,035 | 427,382 | 74,190 | 426,331 | 73,990 |
| 6 | 426,331 | 73,990 | N/A | N/A | NA | N/A |

**Table 3**

Change in the number of elements and nodes in the second mesh used for solution of the emission fluence field in the coupled photon diffusion problem with the adaptive dual mesh scheme (E = elements, N = nodes, N/A = not applied)

| Iteration | Start | | Refinement | | Derefinement | |
|---|---|---|---|---|---|---|
| | E | N | E | N | E | N |
| 1 | 3760 | 791 | 30,080 | 5600 | N/A | N/A |
| 2 | 30,080 | 5600 | 32,820 | 6070 | N/A | N/A |
| 3 | 32,820 | 6070 | 60,384 | 10,824 | N/A | N/A |
| 4 | 60,384 | 10,824 | 122,863 | 21,480 | N/A | N/A |
| 5 | 122,863 | 21,480 | 367,847 | 63,200 | N/A | N/A |
| 6 | 367,847 | 63,200 | N/A | N/A | N/A | N/A |