Top-down extended meshing algorithm and its applications to Green's tensor nano-optics calculations

Joan Alegret[*] and Mikael Käll
Department of Applied Physics, Chalmers University of Technology, S-412 96 Göteborg, Sweden

Peter Johansson
Department of Natural Sciences, Örebro University, S-701 82 Örebro, Sweden

(Received 6 November 2006; revised manuscript received 17 January 2007; published 6 April 2007)

We present a computational algorithm which speeds up Green's tensor nano-optics calculations by means of optimizing the mesh that represents the system we want to investigate. The algorithm automates the process of creating a variable-size mesh that describes an arbitrary nanostructure. The total number of elements of this mesh is smaller than that of a regular mesh representing the same structure, and thus the Green's tensor calculations can be performed faster. Precision, however, is kept at a similar level than for the regular mesh. Typically, the algorithm yields a mesh that speeds up Green's tensor calculations by a factor of 4, while giving a maximum error in the field magnitude of about 5%. The speed-up factor makes it very suitable for otherwise lengthy calculations, and the error should be acceptable for most applications.

DOI: 10.1103/PhysRevE.75.046702 PACS number(s): 02.60.Pn, 42.25.Bs, 42.25.Fx

I. INTRODUCTION

Recent developments in measurement and fabrication techniques have permitted an enormous advancement in the field of nano-optics. Methods such as near-field scanning optical microscopy (NSOM) [1] and surface-enhanced Raman scattering (SERS) [2–4], applied to well-defined samples prepared by, for example, electron beam lithography (EBL) [5] or focused ion-beam etching (FIB) [6], have opened the possibility to probe optical near-field magnitudes, like electric field or local optical density of states [7,8], with great precision. We do not have, however, a practical theory that satisfactorily explains all behaviors observed by such experimental techniques. Apart from some highly idealized cases, such as spheres, ellipsoids, or cylinders, it is, in general, impossible to find analytical solutions to Maxwell's equations for arbitrary nano-optical systems, and it usually becomes necessary to utilize numerical methods to make theoretical predictions or corroborate experimental results. A wide variety of such methods have been developed: the discrete-dipole approximation (DDA) [9,10], Finite-difference time domain (FDTD) [11,12], Multiple multipole (MM) [13], and Green's tensor (GT) [14,15], are some of the most common; for a more complete overview, see, for example, Ref. [16].

The Green's tensor method is particularly interesting because of its adaptability and range of applications. Unlike the other methods mentioned above, GT is based on the exact solution for a propagating wave in a homogeneous background. The idea is to solve Helmholtz's vector equation for the given background, and then insert the scatterer or scatterers we want to investigate as a perturbation. The system is divided into small pieces, usually cubes, in a process called *meshing*, and each of those pieces is called a mesh element. GT assumes that the field inside each mesh element is constant, adds all mesh elements to the background as a perturbation, and then numerically solves the total system (background plus scatterers). The solution is exact in the sense that it can be found to arbitrary precision.

The layered Green's tensor (LGT) method [17,18] is a generalization of the above scheme to a situation where the background needs not be homogeneous, but is built up by a stack of homogeneous layers. In this case, the reference system is thus still translationally invariant in two of the coordinate directions, but not in the third. It can therefore be applied to many experimental systems, for example, waveguides or nanoparticles on a substrate. The advantage of LGT, in comparison to other methods, is that the perturbation is the only part we have to discretize with a mesh since the background is already accounted for in the exact solution of Helmholtz's equation. This permits a good precision, and also better speed than if we had to discretize the layers as well.

Unfortunately, the mesh in GT calculations often consists of a number of elements N so large that those calculations are simply too slow to be practical. For instance, with our FORTRAN implementation of GT and LGT, GT calculations of a system with just $N=10^4$, with an allowed relative error of 10^{-5}, take about six hours per wavelength on a modern computer (AMD 64-bit 3800+GHz processor, 1 GB RAM), while LGT calculations for the same system can take much longer depending on the chosen amount of layers. Calculations of spectral properties will have to be repeated for different wavelengths, so for just 20 wavelengths it will take 5 days with GT, and more with LGT.

DDA tries to circumvent this time problem by solving the system in Fourier space, which is a well-known method to speed up the solution [19]. Unfortunately, this step requires the background to be homogeneous and the mesh elements to be of equal size. While in principle we can limit ourselves to constant mesh sizes, the background in GT methods is in general not homogeneous (LGT has by definition a heterogeneous background), so we cannot, in general, resort to this

[*]Electronic address: alegret@chalmers.se

way of speeding calculations up. Another choice would then be to run these calculations within a set of parallel computers, but this solution is out of reach in many cases, as parallel computing facilities are not widely available. Instead, our goal was to optimize GT and LGT calculations so that they would be feasible even on single-processor computers. We therefore had to find a way to decrease the time it takes to make GT or LGT calculations, without noticeably affecting the precision of the results.

The next obvious choice to decrease calculation times is to reduce the number of mesh elements N. There are a number of so-called adaptive mesh refinement (AMR) methods [20,21] that try to find the optimum mesh by starting with a coarse mesh, solving the system with the chosen numerical method, estimating the error associated to each grid point and creating a new mesh with more elements where the error is larger, then repeating the process until the error is below a specified threshold.

For the case at hand, there are two problems with the AMR approach: first of all, one needs a good way of estimating the error to be able to implement it; and second, it is not a fast algorithm for calculation-heavy methods such as GT: since each AMR iteration adds more elements to the mesh that one has to solve with GT, it pays off to try to find an algorithm that creates an optimized mesh from the start, as we avoid all intermediate calculations to reach a final mesh. Hence our source for a faster GT method lies elsewhere.

There are two main obstacles in trying to find an optimum meshing algorithm for GT calculations. First, as mentioned above, GT assumes that the electric field is constant inside each mesh element. If we decrease the number of elements, then they will have to be larger in order to simulate the same system, but if we make them too large, our assumption of a magnitude being constant inside them will not be accurate enough. Second, for purely geometrical reasons, large elements may not, in general, represent the shape of our nanostructure well enough. If our mesh elements are cubic and we want to represent a curved object, the discretization will produce a staircasing effect all along the curved surface. This effect, of course, will be greater the larger the mesh elements are.

Our solution to these problems is based on the observation that the geometrical effect is of more importance than the effect of assuming a constant field inside the mesh elements. In other words, if the geometrical effect would not exist, we could have larger mesh elements without compromising the precision of the result. The reason that the geometrical effect is larger is that the staircasing effect produces pointy edges at the surface of the particles. Since electromagnetic field concentrates around pointy edges, this effect introduces a large error at or near the surface of a rounded particle, so near-field calculations could be seriously affected. This implies that we should create a finer mesh near the surfaces of particles, and a coarser mesh in the bulk. In this way, the near field will remain approximately the same, while N will be reduced and thus the calculations will take less time.

We present here an algorithm that solves the need for an automated mesh generator for nano-optics calculations. We call it the "top-down extended meshing algorithm" (TEMA). It takes a regular square or cubic mesh and transforms it into a variable-size mesh suitable for nano-optics GT or LGT simulations of arbitrary systems. TEMA greatly speeds up the calculations without compromising the precision of the result.

II. THEORETICAL BACKGROUND

Before looking at the TEMA method itself, we will give a summary of the Green's tensor method for nano-optics, a step that is important in order to clarify some of the issues that TEMA addresses. Further details can be obtained elsewhere [14,15,17,18].

Consider an incident field $\mathbf{E}^0(\mathbf{r},t)$, periodic in time,

$$\mathbf{E}^0(\mathbf{r},t) = \mathbf{E}^0(\mathbf{r})e^{-i\omega t}, \tag{1}$$

and propagating in a nonmagnetic background with permittivity ϵ_B. We add a scattering system (a particle or collection of particles) to the background, by means of introducing a dielectric function $\epsilon(\mathbf{r})$ that has the value corresponding to the material of the scatterer(s) at the places where a scatterer is present, and is equal to the background value ϵ_B otherwise. The GT method yields the optical properties of the total system (background plus scatterers) when illuminated by the incident field.

We start with the vectorial Helmholtz equation that describes the electric field of the total system, $\mathbf{E}(\mathbf{r})$:

$$\nabla \times \nabla \times \mathbf{E}(\mathbf{r}) - k_0^2\epsilon(\mathbf{r})\mathbf{E}(\mathbf{r}) = \mathbf{0}, \tag{2}$$

where k_0 is the vacuum wave number. By defining the function

$$\Delta\epsilon(\mathbf{r}) = \epsilon(\mathbf{r}) - \epsilon_B \tag{3}$$

we may rewrite Eq. (2) as

$$\nabla \times \nabla \times \mathbf{E}(\mathbf{r}) - k_0^2\epsilon_B\mathbf{E}(\mathbf{r}) = k_0^2\Delta\epsilon(\mathbf{r})\mathbf{E}(\mathbf{r}). \tag{4}$$

By assumption, $\mathbf{E}^0(\mathbf{r})$ is a solution of the homogeneous equivalent of Eq. (4):

$$\nabla \times \nabla \times \mathbf{E}^0(\mathbf{r}) - k_0^2\epsilon_B\mathbf{E}^0(\mathbf{r}) = \mathbf{0}. \tag{5}$$

This equation may be used to define a Green's tensor for the background, $\mathbf{G}(\mathbf{r},\mathbf{r}')$, by introducing a point source at $\mathbf{r}=\mathbf{r}'$:

$$\nabla \times \nabla \times \mathbf{G}(\mathbf{r},\mathbf{r}') - k_0^2\epsilon_B\mathbf{G}(\mathbf{r},\mathbf{r}') = \mathbf{1}\delta(\mathbf{r}-\mathbf{r}'). \tag{6}$$

The explicit form of $\mathbf{G}(\mathbf{r},\mathbf{r}')$ for an homogeneous background is [15]

$$\mathbf{G}(\mathbf{r},\mathbf{r}') = \left(\mathbf{1} + \frac{\nabla\nabla}{k^2}\right)\frac{\exp(ikR)}{4\pi R}, \tag{7}$$

where $k^2 = k_0^2\epsilon_B$ and $R = |\mathbf{r}-\mathbf{r}'|$.

Equations (4)–(6) yield the following expression for the total field $\mathbf{E}(\mathbf{r})$:

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}^0(\mathbf{r}) + \int_V \mathrm{d}\mathbf{r}' G(\mathbf{r},\mathbf{r}') \cdot k_0^2 \Delta\epsilon(\mathbf{r}')\mathbf{E}(\mathbf{r}'), \tag{8}$$

where V denotes the region in space where scatterers are present. The equation has a singularity for $\mathbf{r}'=\mathbf{r}$, which can be avoided by removing the singularity point from the integration volume and compensating the value of the integral through a source dyadic $\mathbf{L}$ [15,22] that depends on the geometry of the excluded infinitesimal volume. Explicitly,

$$\mathbf{E}(\mathbf{r}) = \mathbf{E}^0(\mathbf{r}) + \left(\int_{V'} \mathrm{d}\mathbf{r}' G(\mathbf{r},\mathbf{r}') \cdot k_0^2 \Delta\epsilon(\mathbf{r}')\mathbf{E}(\mathbf{r}') \right) - \mathbf{L} \cdot \frac{\Delta\epsilon(\mathbf{r})}{\epsilon_B}\mathbf{E}(\mathbf{r}), \tag{9}$$

where V' is the original volume V minus an infinitesimal volume around the singular point $\mathbf{r}'=\mathbf{r}$: $\int_{V'} \equiv \lim_{\delta V\to 0}\int_{V-\delta V}$.

This equation may be discretized in a way that becomes useful for numerical calculations. We first subdivide (i.e., *mesh*) the volume V into N pieces, which we will assume to be of cubic shape. The resulting mesh elements are centered at positions $\{\mathbf{r}_1,\ldots,\mathbf{r}_N\}$. The elements do not necessarily have to be of the same size, so we will keep track of their individual volumes as well: $\{V_1,\ldots,V_N\}$. For simplicity, we introduce the subindex notation $\mathbf{E}_i=\mathbf{E}(\mathbf{r}_i)$, $\mathbf{G}_{ij}=\mathbf{G}(\mathbf{r}_i,\mathbf{r}_j)$, and analogously for other quantities. Equation (9) becomes

$$\mathbf{E}_i = \mathbf{E}_i^0 + \left(\sum_{j=1, j\neq i}^{N} \mathbf{G}_{ij} \cdot k_0^2 \Delta\epsilon_j V_j \mathbf{E}_j \right) + \mathbf{M}_i \cdot k_0^2 \Delta\epsilon_i \mathbf{E}_i - \mathbf{L} \cdot \frac{\Delta\epsilon_i}{\epsilon_B}\mathbf{E}_i, \tag{10}$$

where

$$\mathbf{M}_i = \int_{V_i'} \mathrm{d}\mathbf{r}' \mathbf{G}(\mathbf{r}_i,\mathbf{r}') \tag{11}$$

is the self-interaction term [23], which must be calculated explicitly due to the infinitesimal volume δV removed around the singularity of the Green's tensor.

To calculate $\mathbf{M}_i$, we should in principle evaluate Eq. (11). However, $\mathbf{M}_i$ is often a small quantity, so that we may calculate it approximately, without incurring too large an error, by assuming the volumes V_i and δV to be spherical. Namely, we define an effective radius R_i for each element i as follows:

$$R_i \equiv \left(\frac{3}{4\pi} V_i \right)^{1/3}, \tag{12}$$

and we map our cubic mesh into a spherical one with elements of radii R_i. For an integration volume V_i' consisting of a spherical volume V_i minus a spherical infinitesimal volume δV removed from its center, we have [15]

$$\mathbf{M}_i = \frac{2}{3k_0^2}[(1 - ik_0R_i)\exp(ik_0R_i) - 1]\mathbf{1}. \tag{13}$$

This approximation is of course optional and may be skipped if rigorous results are desired.

On the other hand, for δV either cubic or spherical, we have [22]

$$\mathbf{L} = \frac{1}{3}\mathbf{1}. \tag{14}$$

Equations (10)–(14) are the basic equations of the homogeneous Green's tensor method.

Notice that Eq. (10) approximates the integral in Eq. (9) by a discrete sum: the field at the center of each element times the volume of that element. Thus the field inside each individual mesh element is assumed to be constant. This is a good approximation if the mesh element sides are much smaller than the wavelength of the incoming field, so that the field varies slowly inside all elements. Note, however, that this size restriction only holds for the individual mesh elements. The scatterer as a whole can of course be of arbitrary size.

Equation (10) can be written as a set of linear equations that has to be solved for all components of $\mathbf{E}_i$, so if the number of mesh elements N is large, the equations will take a long time to be solved.

Furthermore, the field outside the scattering region V is

$$\mathbf{E}(\mathbf{r}_{out}) = \mathbf{E}^0(\mathbf{r}_{out}) + \sum_{j=1}^{N} \mathbf{G}_{oj} \cdot k_0^2 \Delta\epsilon_j V_j \mathbf{E}_j, \tag{15}$$

with $\mathbf{G}_{oj} \equiv \mathbf{G}(\mathbf{r}_{out},\mathbf{r}_j)$ and $\mathbf{r}_{out} \notin V$. It is clear from Eq. (15) that a large N will have an impact on the calculation time of fields.

For a layered background, the central equations are essentially the same, although the permittivity of the background is not a constant, but a piecewise constant function that only depends on the coordinate perpendicular to the layers. Calculating $\mathbf{G}_{ij}$ for a layered background is in itself a lengthy procedure, so it is even more important to optimize the mesh for this case, to minimize the number of such calculations.

III. ALGORITHM

TEMA assumes that we have a certain matrix M that represents a regular mesh. The matrix can have dimension 2 or 3 depending on whether we want to simulate a two- or a three-dimensional system. The cells of the matrix represent the material present at the corresponding position of the mesh. The value of the cell is the code for the material present in that position: for instance, it may be set to zero if the element is occupied—the code of element number j is therefore related to the value of $\Delta\epsilon_j$ in Eq. (10). All mesh elements that are occupied by the same material are represented by the same value in the matrix M. Notice that if a cell is "empty," it means in our case that the position is occupied by a certain background material, which can indeed be vacuum but could also be any other substance, because our background may consist of layers of different materials. By assigning a certain length d to the side of each mesh element that M represents, we map M into a region of the physical space. We then have a regular mesh that represents the system we want to study.

Our goal now is to create a second mesh with a smaller number of mesh elements, thereby speeding up further calculations, without reducing the precision of the result. We will impose that all mesh elements we create have sides that are multiples of d. This choice allows us to simplify the algorithm, but it may also speed up the Green's tensor method, as will be discussed in Sec. IV D 2. For the sake of simplicity, we will refer to a mesh element with side ad as a "size a element."

We must first decide which is the largest size for a mesh element that we allow the algorithm to create. This size naturally depends on the problem considered, and on the precision we would like to have for our calculations. We will choose the side s of this largest element to be a multiple of d: $s=bd$, $b \in \mathbb{N}$. Our algorithm must then be able to create mesh elements with sizes from 1 to b. Larger sizes are preferred, since they will reduce the total amount of mesh elements N, but for near-field calculations, we must keep size-1 elements at the surface of the particles to avoid precision loss, as we will see later. We therefore need an automatic way to ensure that the outer part of the structure contains only size-1 elements. For this purpose, let us now introduce the concept of an *extended mesh element*. To each mesh element with size h, we will assign a second element, centered at the same position but with size $h+f(h)$, where $f(h) \in \mathbb{N}, \forall h$, and $f(1)=0$. We will call this second element the "extended mesh element." Notice that, by definition, the size of the extended element is greater than h for $h>1$. We will also call the original mesh element with size h the "proper" mesh element. Finally, we will refer to the collection of all points in the extended mesh element and not in the proper mesh element as the "edge" of the element of size h.

By means of this extended element, we can solve the problem of automatically producing a variable-size mesh with significantly less elements than the original mesh with constant sizes. We scan throughout our space M in sequence, and check if the *extended* mesh element we are considering will fit at a given position. By fit, we mean that it must sit in its entirety inside a subspace that is not empty, i.e., which is occupied by some material different from the background. If this is the case, we assign the corresponding *proper* mesh element to that position, and mark the space it occupies as full. Also, we cannot allow mesh elements to overlap, so the proper mesh element must only occupy a space that previously was not full, that is, which was only occupied by size-1 elements. Notice that this procedure ensures that the external part of any objects in our mesh contains only size-1 elements. This is depicted in Fig. 1.

It is worth noting that the method we use to scan through M is not unambiguously defined. Let us say we begin at point (1, 1, 1). We may try to fit the extended element there by taking (1, 1, 1) to be any of the eight corners of the extended element, or its center, for instance. Additionally, if the element does not fit, we can proceed to point (2, 1, 1), or point (1, 2, 1), or (1, 1, 2), and so on. However, all these options will yield very similar results. Only for the case of symmetric objects it will be favorable to choose a specific way of re-meshing; we will further discuss this point in Sec. IV D 4.

Since we want the algorithm to be general, it must also handle the case when several materials are present in our system. As mesh elements have to consist of only one material, we must also add the condition that the whole space occupied by a proper element must contain only one kind of material. This is not necessary for the whole extended element, so the edge can actually consist of different materials, as long as none of them is background.

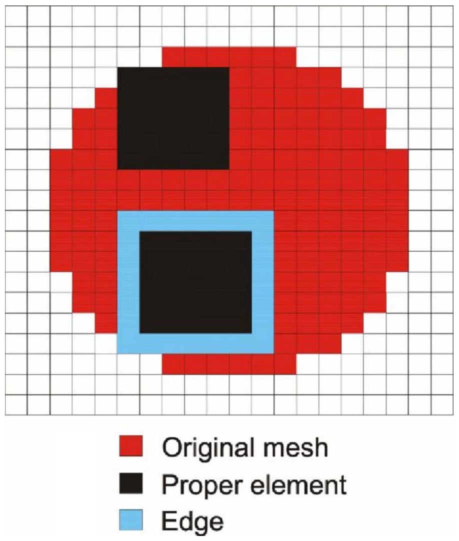

FIG. 1. (Color online) Comparison between proper and extended mesh elements. A proper mesh element with no edge (top black square) can sit at the surface of the object. By introducing an edge, the proper mesh element (bottom black square) is forced to sit at a distance from the surface.

Once we decide the largest size b and the individual sizes of the extended elements, the algorithm already has enough information to create the optimized mesh. It will start with the largest size b and try to fit the extended element wherever possible. Then it will do the same for sizes $b-1$, $b-2\ldots$, down to 2. Each successfully positioned mesh element of size a will reduce the total number of mesh elements N by a^D-1, where D is the dimension of the space M, so it is more favorable to start with the largest elements. This is why we call this a top-down algorithm.

Notice, however, that we cannot assert that taking b to be the largest mesh size will produce the best results in terms of minimizing N, as this critically depends on the geometry of our system. In order to further improve our mesh, we should repeat the whole process, but changing the largest allowed size from b to $b-1$, $b-2\ldots$, down to 2. We should store the corresponding numbers of mesh elements N_b, $N_{b-1},\ldots$, N_2, and choose the one that yields the smallest number. The last step is to output the mesh on a file that can be read by the GT/LGT program.

IV. RESULTS AND DISCUSSION

A. TEMA example

Figure 2 presents an example of a three-dimensional mesh created by TEMA. The original mesh represents a sphere of 45 mesh units in diameter, with an irregular internal void built up by three overlapping vacuum spheres. The maximum allowed size for the mesh elements is initially taken to

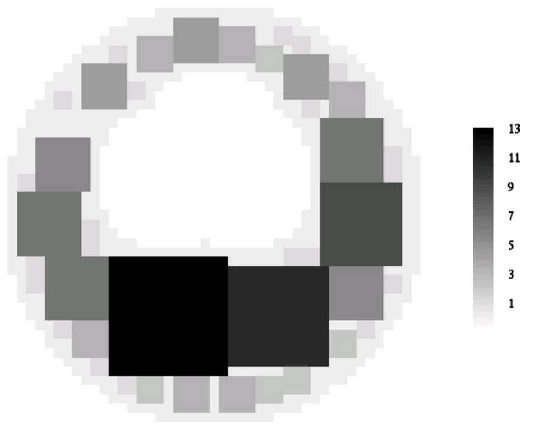

FIG. 2. (Color online) Example of a mesh created by TEMA. A diameter 45 sphere with an asymmetric void is re-meshed with mesh sizes of 13 and below. The figure shows a cut through the center of the sphere. The mesh elements are color-coded according to their sizes.

be $b=15$. All elements are required to have a size-1 edge. The algorithm found an optimal maximum size of $b=13$. The figure represents a cut through the center of the object. By applying TEMA, the total number of elements goes from $N=46\,588$ for the original mesh to $N=18\,113$, making it possible to calculate the same sample with just 40% of the initial mesh elements. If a maximum size of 13 is too large for our purposes, decreasing b does not drastically increase the total amount of mesh elements. For example, a maximum size of $b=7$ yields $N=18\,212$, and $b=3$ leads to $N=19\,753$.

B. Calculation times

Figure 3 shows a comparison between LGT calculation times with and without TEMA. The samples considered are truncated spheres of different sizes, in a two-layer system. TEMA has been applied with a constant edge of size 1 for all elements, and taking the largest allowed element to be $r-2$,

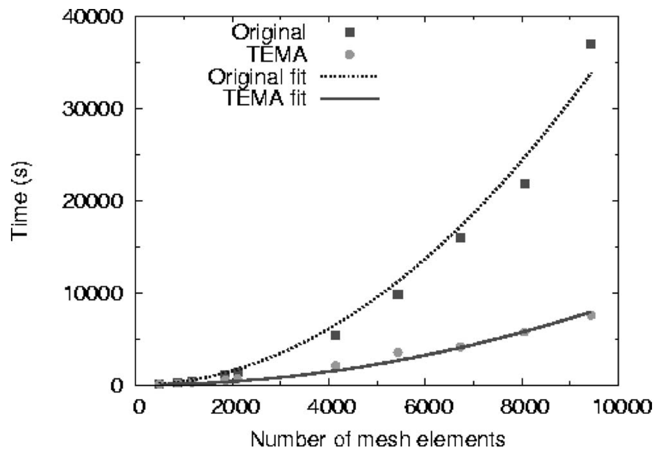

FIG. 3. Comparison between calculation times for the original, regular mesh, and the mesh created by TEMA. The two sets of calculation times are fitted to a quadratic function $t=AN^2$, where N is the number of mesh elements in the original mesh.

TABLE I. Calculation times with the original mesh ($t_{original}$) and with the TEMA mesh (t_{TEMA}), according to the fitting functions from Fig. 3. N is the number of elements in the original mesh.

N	$t_{original}$	t_{TEMA}
10^3	378 s	89 s
10^4	10.5 h	2.5 h
10^5	43.7 days	10.3 days

where r is the radius of the sphere. All calculations have been carried out on a Pentium 4 3.6-GHz computer. Our LGT program is a FORTRAN code which uses a variant of the bi-conjugate gradient method, the BiCGstab(2) method [24,25], to solve the linear system of equations that yields the fields inside each mesh element.

We can extrapolate the calculation times by fitting the two sets of results to a quadratic function, $t=AN^2$. This gives us an approximation to the calculation times with and without TEMA. The resulting A coefficient is $A=(3.78\pm0.12)\times10^{-4}$ s for the original mesh and $A=(8.9\pm0.3)\times10^{-5}$ s for the TEMA mesh.

Table I presents a comparison between times for different values of N. As the results indicate, GT calculations using the TEMA mesh are about four times faster than those using a regular mesh. The time required to run the TEMA program itself is negligible compared to the rest of the calculations: the re-meshing has been carried out in all cases in 5 s or less.

C. Precision

It is also necessary to make sure that the precision is not noticeably affected when utilizing the TEMA mesh. Figure 4 shows an example of the same near-field enhancement calculation with the original mesh and with the TEMA mesh. We have taken a gold structure shaped like a crescent moon with a diameter of 24 nm and a height of 12 nm, embedded in vacuum. The scatterer has been created by starting with a gold cylinder of 26 nm diameter and 12 nm height with its base centered at the origin, and then putting a “negative” or vacuum cylinder of equal dimensions but with the base centered at $(0,-7,0)$ nm. The resulting shape is a crescent-like structure, but due to the mesh discretization, it presents a sharp edge on each side, and is slightly smaller than the original cylinders. The presence of sharp edges is ideal to test the precision of TEMA, since the electric field is concentrated around them. We look at the field enhancement in a plane located 2 nm below the base of the crescent-like structure.

The near-field presents few differences between the TEMA calculations and the original mesh. The most apparent one is a quantitatively different enhancement at the hot spots, but a numerical comparison between the two plots reveals that the maximum relative error for the field magnitude is only 3.8%. This is consistent with our requirement that the meshing method should not significantly affect the precision. The differences between the two plots around the curved black lines are mostly due to the different scale of the two graphs.

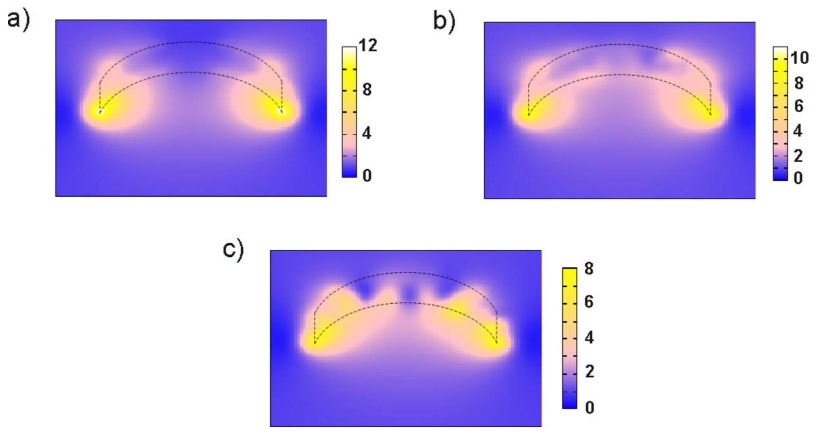

FIG. 4. (Color online) GT calculations of a "crescent moon"-like gold structure in vacuum. The wavelength is $\lambda=600$ nm, and the gold dielectric constant is $\epsilon=-9.42+1.50i$. (a) Field enhancement, 2 nm below the surface, with the original mesh. (b) Same plot, but with the TEMA mesh. (c) Variable size mesh plot with no mesh edges. Notice the big difference between this plot and the previous two.

Figure 4(c), is the same near-field plot, but with a mesh with edge sizes equal to zero in all cases. It is apparent from the graph that the result is quite different from the original plot. This behavior demonstrates the need for mesh edges, as we have discussed previously.

For a general case, the quality and accuracy of the results will, of course, depend both on the largest size allowed for the TEMA elements and on the desired precision, so there is no general recipe to decide when the TEMA mesh is satisfactory. On the other hand, TEMA is made in such a way that it is difficult to reach a point where the mesh created by the algorithm yields results that do not resemble the ones obtained with a regular mesh. Even if we allow very large elements to be created, the algorithm will surely find that a smaller value for the largest element gives a more optimized mesh, as placing a very large element inside a particle leaves little room for the rest. Our tests indicate that TEMA meshes rarely contain less than 40% of the original elements, except when the particles are close to being parellelepipeds, but in that case the TEMA mesh still gives good results. As a rule of thumb, the maximum error for the field magnitude with an arbitrary TEMA mesh is of about 5%, which should be acceptable for most calculations. There is an exception to this small error when calculating local fields that we will discuss and give a solution to in Sec. IV D 5

Further improvement in the precision of TEMA calculations can be achieved by utilizing the symmetry of the sample, as we will show in Sec. IV D 4

D. Other improvements

1. Convex particles

In its general form, this is already a fast algorithm, but its speed can be improved when we are considering convex particles. If the system contains no concavities, and the distance between surfaces is larger than the largest edge of the extended elements, then a necessary and sufficient condition for any extended element to fit inside the system (in the sense discussed above) is that its vertexes be in a position occupied by a size-1 element. That reduces the amount of tests to see if an extended element is in an allowed position to just four (in two dimensions) or eight (in three dimensions), thereby speeding up the algorithm. Notice, however, that once the edge is found to be in an allowed location, we still have to go through the tests for the proper element.

2. GT symmetries

One key optimization lies in the GT method itself. Since we have taken care in positioning the centers of the mesh elements on a regular matrix of points, we can greatly decrease the amount of calculations involved in solving the GT linear system by means of the symmetries of the dyadic Green's tensor.

Let us first concentrate on the homogeneous case. Because of the symmetry of the problem (rotational and translational invariance in all three coordinates), we may calculate the individual Green's tensor for two given mesh elements, $\mathbf{G}(\mathbf{r}_i,\mathbf{r}_j)$, through the Green's tensor of their distance:

$$\mathbf{G}(\mathbf{r}_i,\mathbf{r}_j)=M^{-1}\mathbf{G}(\mathbf{d},\mathbf{0})M. \qquad (16)$$

Here, $\mathbf{d}=(|\mathbf{r}_i-\mathbf{r}_j|,0,0)$ and M is the rotation matrix that brings the $\mathbf{d}$ vector parallel to $\mathbf{r}_i-\mathbf{r}_j$.

Equation (16) means that we may reconstruct the whole GT matrix from the distances between elements and the angles that all combinations of $\mathbf{r}_i-\mathbf{r}_j$ make with the $+x$ axis. As the elements all lie on a regular matrix of points, the total number of different distances between points, D, should be much smaller than the number of possible combinations of two elements, N^2. We can first scan through all combinations of mesh positions $\mathbf{r}_i$ and $\mathbf{r}_j$ and create a look-up table of their respective distances d_i, $i=1,\ldots,D$. The next step is to calculate the D different Green's tensors $\mathbf{G}_{(i)}(\mathbf{d}_i,0)$ for all possible distances, and store them. Finally, we can reconstruct the dyadic Green's tensor $\mathbf{G}(\mathbf{r},\mathbf{r}')$ for any two mesh elements by means of Eq. (16) and our look-up table.

This way of proceeding will prove particularly useful in the LGT case. While it is fast and easy to obtain the dyadic Green's tensor of two given mesh elements in homogeneous media, the calculations in the layered case are much more time-consuming. Therefore if we first make D calculations, we can then get any of the individual Green's tensors by means of two simple matrix multiplications. As mentioned before, $D\ll N^2$ in most practical cases, thus the calculation time decreases significantly. Moreover, it is not always possible to store N^2 3×3 complex matrices in the computer memory if N^2 is a very large number, but it may be possible to store D matrices instead. This is very useful if we solve the large $3N\times 3N$ matrix defined by the system in Eq. (10) via iterative methods, because we can store all information we need in memory, instead of having to calculate over and over the individual matrices on-the-fly with each iteration.

In the case of the layered Green's tensor, there is no longer full rotational and translational invariance due to the layered background. We may still simplify the problem, though, by utilizing the remaining symmetries of the system. Let us assume that the layers are parallel to the XY plane. Then, the system still possesses full translational symmetry in the X and Y directions, and rotational symmetry around the Z axis. Given two elements at $\mathbf{r}_i=(x,y,z)$ and $\mathbf{r}_j=(x',y',z')$, respectively, we can reconstruct the Green's tensor as follows:

$$\mathbf{G}(\mathbf{r}_i,\mathbf{r}_j)=M^{-1}\mathbf{G}(d,0,z;0,0,z')M. \qquad (17)$$

This time, $d=[(x-x')^2+(y-y')^2]^{1/2}$ because it is only in the XY plane that we can utilize the symmetry. Obviously, M will also be just a rotation in the XY plane.

For the layered case, the look-up table must include separate entries, not only for each value of d, but for each combination of d, z, and z'. The total number D of different 3×3 matrices to store will be larger than for the homogeneous case, but still significantly smaller than N^2. As an example, for a sphere of diameter equal to 45 mesh elements, the TEMA mesh with edge 1 yields $N=15\,162$ elements. To solve the system, we need $N^2\approx 2.3\times 10^8$ matrices. The LGT look-up table, however, consists of $D=2,311,599$ elements, which is two orders of magnitude less. The typical time necessary to calculate a two-layer LGT dyadic matrix is $t_{LGT}\sim 1$ ms. To prepare the full LGT matrix will thus take about one hour using the look-up table elements, and about 100 h without. Notice that this way of proceeding will speed up the solving of the system of equations by *more* than two orders of magnitude, because the matrices can be stored in memory and will not be have to be constructed every time from scratch. Thus the conjugate gradient iterations become much faster.

The extra time needed for the creation of the look-up table itself, that is, finding all different combinations of (d,z,z') and writing the necessary information to a file, is of the order of a minute or less in all cases considered here ($N\leq 10^5$), therefore negligible in comparison to the total gains in time.

All times reported in Sec. IV B are obtained with this symmetry algorithm implemented into our LGT program.

3. Odd-sized mesh

GT symmetries are very useful in order to decrease the calculation time. However, notice that by allowing our mesh sizes to be both odd and even, the centers of the mesh will in general lie on a matrix of points of the form $(k/2,l/2,m/2)$, $k,l,m\in\mathbb{Z}$. This increases the amount of unique distances between elements, thus also increasing the number of elements in the look-up table, D.

If we restrict our proper mesh elements to have odd sizes, the mesh centers will lie on the same regular matrix as the original, size 1 mesh: (k,l,m), $k,l,m\in\mathbb{Z}$. For the example mentioned in Sec. IV A of a 45-diameter sphere, D goes down to 888 898, while N increases to 21 972.

For near-field plots, N is a crucial value because we must create N LGT dyadics for each point in the plot. That can easily amount to 10^4N LGT calculations or more. However, if we just need far-field cross sections, the number of dyadics needed is N for each direction considered, so it will be just N for forward or backward scattering, and pN for scattering in all directions, where p is the number of points in which we discretize the integration in solid angle (a typical value of p in DDA calculations is $p=200$). Thus it may be useful to restrict the mesh sizes to odd numbers if we are only interested in far-field calculations.

4. Symmetric and nearly symmetric objects

If the objects we want to study have a certain symmetry, say they are spherical, we can use this information to improve the precision. The algorithm in its general form does not know where the centers of the objects are located. It starts at one edge of the space to re-mesh and sweeps through all points in sequence: for example, first looking at all points along x, then increasing y one step and looking again through all x, and so on. This will in general produce asymmetric meshes even for symmetric objects. If we know, however, that the object we consider is a sphere, we can force the algorithm to start re-meshing the object from the center outwards. By utilizing this method, larger elements will preferentially be located in the inner part of the sphere; the resulting mesh will be more symmetrical and will represent the original object better, leading to a smaller error. The number of elements will still be optimized.

In general, we can improve the precision at the expense of a little speed for any type of object, by changing the individual edge sizes. If we choose a larger edge size for larger elements, they will only be located where there is more space available, thus the inner part of the object. This is particularly useful for those objects that are nearly symmetrical, like ellipsoids, spheres with a void, etc., where the full symmetry cannot be applied directly.

Figure 5 shows the same sample discussed in Sec. IV A, but with variable size edges according to the function $f(h)=\text{int}(h/2)$ (cf. Sec. III). Notice how the maximum mesh size goes down to 7, and how only the smaller elements are close to the surfaces. This is a good idea if we need a higher precision for near-field calculations. The price to pay is that the higher constraints make it more difficult to place the elements. Therefore the total number of elements will go up. In this case, it has increased from $N=18\,113$ to $N=19\,558$.

5. Near-field plots

One last improvement that we will present here is the following: for near-field plots, if we want to visualize the fields inside the mesh elements, having too big elements may hamper the results, since, as we mentioned, the field inside the mesh elements is taken to be constant. However, we may force the algorithm to keep size 1 mesh elements on the plane or planes we want to visualize, while re-meshing the rest of the system. Therefore we may add a condition to our

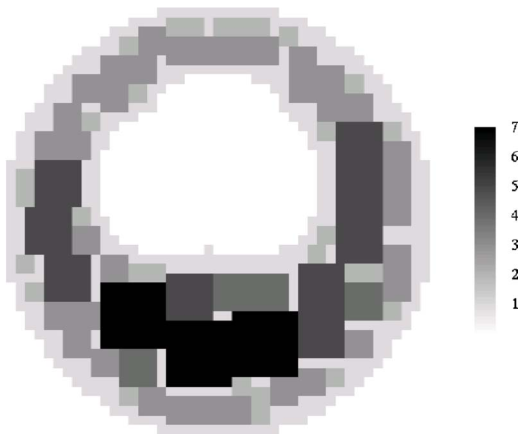

FIG. 5. (Color online) The same sample as in Fig. 2, this time with variable edges so that larger mesh elements are placed in the inner part of the sample.

algorithm, which states that only the edges can intersect the visualization planes, not the proper elements. This will again ensure an automatic re-meshing of the system while keeping size-1 elements all over the visualization planes.

The criteria for an acceptable result will vary depending on the specific needs of the calculations we are performing. If we need high precision, it may be necessary to decrease the size of the smallest element (that is, to increase the number of elements in the original mesh) before applying TEMA.

V. CONCLUSIONS

We have presented the top-down extended mesh algorithm (TEMA) that takes a regular mesh and automatically creates a second mesh with variable size elements. This second mesh is optimized to be used in the Green's tensor method. We have shown that the TEMA mesh leads to faster Green's tensor calculations, typically about four times faster for many practical problems. We have also shown that the precision achieved by the calculations is not significantly decreased by the variable size mesh: typical field magnitude errors are 5% or below.

Since TEMA creates regularly spaced meshes, further gains in speed can be achieved by utilizing the symmetries of the tensor. We have seen that this step also greatly reduces the needs for memory storage, typically by two orders of magnitude.

Finally, we have also shown how to adapt TEMA to specific problems: near-field plots, far-field cross sections, symmetric or quasisymmetric systems, and convex objects. The algorithm itself and its optional improvements are fast and simple to implement, and the gains in calculation speed make it possible to run large Green's tensor problems in a reasonable amount of time, even on single-processor computers.

ACKNOWLEDGMENTS

This work was supported by the Swedish Research Council and the Swedish Foundation for Strategic Research.

[1] U. Dürig, D. W. Pohl, and F. Rohner, J. Appl. Phys. **59**, 3318 (1986).
[2] D. L. Jeanmaire and R. P. Van Duyne, J. Electroanal. Chem. Interfacial Electrochem. **84**, 1 (1977).
[3] M. G. Albrecht and J. A. Creighton, J. Am. Chem. Soc. **99**, 5215 (1977).
[4] M. Moskovits, Rev. Mod. Phys. **57**, 783 (1985).
[5] C. Vieu, F. Carcenac, A. Pepin, Y. Chen, M. Mejias, A. Lebib, L. Manin-Ferlazzo, L. Couraud, and H. Launois, Appl. Surf. Sci. **164**, 111 (2000).
[6] J. Melngailis, A. A. Mondelli, I. L. B. III, and R. Mohondro, J. Vac. Sci. Technol. B **16**, 927 (1998).
[7] C. Girard and E. Dujardin, J. Opt. A, Pure Appl. Opt. **8**, S73 (2006).
[8] G. Colas des Francs, C. Girard, J.-C. Weeber, and A. Dereux, Chem. Phys. Lett. **345**, 512 (2001).
[9] E. M. Purcell and C. R. Pennypacker, Astrophys. J. **186**, 705 (1973).
[10] B. T. Draine, Astrophys. J. **333**, 848 (1988).
[11] K. Yee, IEEE Trans. Antennas Propag. **14**, 302 (1966).
[12] C. F. Lee, *Finite Difference Method for Electromagnetic Scattering Problems* (Massachusetts Institute of Technology, Cambridge, MA, 1990).
[13] E. Moreno, D. Erni, C. Hafner, and R. Vahldieck, J. Opt. Soc. Am. A **19**, 101 (2002).
[14] O. J. F. Martin, A. Dereux, and C. Girard, J. Opt. Soc. Am. A **11**, 1073 (1994).
[15] O. J. F. Martin and N. B. Piller, Phys. Rev. E **58**, 3909 (1998).
[16] C. Girard, Rep. Prog. Phys. **68**, 1883 (2005).
[17] M. Paulus, P. Gay-Balmaz, and O. J. F. Martin, Phys. Rev. E **62**, 5797 (2000).
[18] M. Paulus and O. J. F. Martin, J. Opt. Soc. Am. A **18**, 854 (2001).
[19] J. Goodman, B. Draine, and P. Flatau, Opt. Lett. **16**, 1198 (1991).
[20] F. Fernandez, Y. Yong, and R. Ettinger, IEEE Trans. Magn. **29**, 1882 (1993).
[21] R. Verfürth, in *Proceedings of the 5th International Conference on Computational and Applied Mathematics* (Elsevier, Amsterdam, 1994), pp. 67–83.
[22] A. D. Yaghjian, Proc. IEEE **68**, 248 (1980).
[23] N. Piller and O. Martin, IEEE Trans. Antennas Propag. **46**, 1126 (1998).
[24] G. L. G. Sleijpen and D. R. Fokkema, Electron. Trans. Numer. Anal. **1**, 11 (1993).
[25] G. L. G. Sleijpen, H. A. van der Vorst, and D. R. Fokkema, Numer. Algorithms **7**, 75 (1994).