Computer Physics Communications 265 (2021) 108025

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

GSvit — An open source FDTD solver for realistic nanoscale optics simulations ☆,☆☆

Petr Klapetek [a,b,*], Petr Grolich [a], David Nezval [a,b], Miroslav Valtr [a,b], Radek Šlesinger [a], David Nečas [b,c]

[a] Czech Metrology Institute, Okružní 31, 638 00 Brno, Czech Republic
[b] CEITEC BUT, Purkyňova 123, 612 00 Brno, Czech Republic
[c] RG Plasma Technologies, CEITEC, Masaryk University, Kamenice 5, 625 00 Brno, Czech Republic

ARTICLE INFO

Article history:
Received 29 February 2020
Received in revised form 1 April 2021
Accepted 26 April 2021
Available online 6 May 2021

Keywords:
FDTD
Plasmonics
Optics
Roughness

ABSTRACT

Surface and volume imperfections can significantly affect the performance of nanoscale or microscale devices used in photonics, optoelectronics or scientific instrumentation. In this article we present an open source software package for Finite-Difference Time-Domain electromagnetic field calculations suitable for calculations on graphics cards. Its special features include handling realistic models of imperfect nanoscale objects, such as treatment of arbitrary geometries including addition of random roughness to any geometrical object. The method is compared to conventional optical approach represented by Rayleigh-Rice theory. Practical applicability is demonstrated using a calculation of variation of field enhancement at proximity of a rough nanoscale antenna and rough particle scattering. It is shown that such approach can be namely useful in the areas where many repeated calculations are necessary, e.g. when studying how the optical response of nanoscale objects can vary when they are rough.

Program summary
Program Title: GSvit
CPC Library link to program files: https://doi.org/10.17632/k424zbsxnk.1
Licensing provisions: GPLv2
Programming language: C
Nature of problem: If we want to analyse impact of random imperfections, in particular surface roughness, on optical response of nanoscale and microscale objects, we need to run many calculations with different random realisations. GSvit is a general electromagnetic field solver optimized for running such calculations, via fast computing on graphics cards and algorithms for loading of arbitrary data and modification of their geometry to construct randomly rough surfaces and interfaces.
Solution method: Finite-Difference Time-Domain method implemented on computer processor and on graphics card, with advanced pre-processing in order to add realistic roughness to 3D objects.
Additional comments including restrictions and unusual features: The current version of software, including the documentation and downloadable examples can be found on http://gsvit.net/.

1. Introduction

Phenomena related to interaction of light with artificially formed micro- and nanostructures are related to many scientific areas, with rapidly increasing number of potential industrial applications. Advanced manufacturing techniques allow the creation of new materials and functional devices, e.g. in the field of plasmonics and optoelectronics. There are also many novel measuring instruments which employ local electromagnetic field scattering, like tip-enhanced Raman spectroscopy, surface-enhanced Raman spectroscopy, or near field infrared spectroscopy. Various nanoantennas, antenna arrays and localized scattering centres are being produced for these purposes. In order to increase reliability of manufacturing processes it is important to be able to analyse potential effects of imperfections and irregularities on device

☆ The review of this paper was arranged by Prof. Stephan Fritzsche.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail address: pklapetek@cmi.cz (P. Klapetek).

https://doi.org/10.1016/j.cpc.2021.108025

performance. This can be rarely done analytically and a numerical solution capable of incorporating imperfections is therefore needed.

In order to treat the interaction of electromagnetic field with structures similar to its wavelength, we need to utilise fully its wave nature in order to incorporate all the scattering, diffraction and interference processes. Several techniques can handle this problem and are used for nanoscale optical calculations, for example Direct Moment Method [1], Green's tensor technique [2], or Reciprocal-space Perturbative Method [3]. Each approach has its strengths and weaknesses and is best suited for particular types of problems. Here we present a Finite-Difference Time-Domain based approach (FDTD) [4].

FDTD is a well-established computational method and is already widely used in the field of nanoscale optics. It is capable of solving nearly arbitrary problems in electromagnetics, at least in principle. Its main drawback is the requirement of a fine spatial grid (compared to the wavelength), which limits applicability to conventional optical phenomena (e.g. with visible wavelength and millimetre or centimetre size optical elements) because it then leads to excessive computational demands. However, the natural range of scales in nanoscale and microscale optics is favourable for FDTD. Hence, it is becoming a common technique for calculations in plasmonics [5,6], optoelectronics [7], microscopy [8–11] and similar areas. Many computational tools already available for nanoscale optics calculations, both commercial and open source [12], employ FDTD.

FDTD calculations presented in scientific literature are frequently based on simple geometrical models [6,9,13] which are not capable of treating realistic material imperfections, such as non-ideal geometries, surface roughness, or various inhomogeneities. One possible reason may be that setting up complex models involving deviation from ideal geometries is time-consuming and that more detailed models in FDTD still require significant computational time, as a dense space discretisation needs to be used and all the algorithms related to FDTD then become prohibitively slow.

Of all the possible imperfections, we concentrate on surface roughness as this error source is frequently present in many nanoscale and microscale systems. All manufacturing processes result in some surface roughness, which can be controlled only to some extent. Usually roughness only decreases device performance, but in a few specific cases it can also be helpful, for instance with hot spots in tip-enhanced Raman spectroscopy [14] or in surface-enhanced Raman spectroscopy [15].

In this article we present algorithms developed for including the random roughness easier in the framework of FDTD. For statistical analysis we namely need methods generating rough nanoobjects with known statistical properties in the computational domain, and a FDTD solver fast enough to run many calculations with different realisations of the rough object. On the basis of our proof-of-concept studies [16,17] we have developed the open source GSvit solver [18], focusing on topics that we found missing or inadequate in other software packages:

1. Graphics card computing support for calculation speedup.
2. Treatment of realistic geometries determined experimentally, e.g. loading data like surface topography or form from different measuring instruments.
3. Capability to create objects with realistic roughness, or alter surface properties of loaded objects, running multiple calculations for different random seeds.

The performance of data modification algorithms is validated by comparison to classical optical theory, and its practical applicability is demonstrated on calculation of local field enhancement at proximity of a rough single rod metallic nanoantenna and on scattering by a rough sphere. Although this work focuses on distinctive functions of GSvit, it has to be noted that it is an universal computational package which can also deal with typical tasks in computational electromagnetics, similarly to the other FDTD packages.

2. GSvit overview

2.1. FDTD basics

The Finite-Difference Time-Domain method is a widely used method for complete numerical solution of Maxwell equations in discretized space. For a linear, isotropic, non-dispersive and possibly lossy material the Maxwell equations describing the time evolution of electric and magnetic field intensity vectors **E** and **H** can be written in the following form

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu}\nabla \times \mathbf{E} - \frac{1}{\mu}(\mathbf{M} + \sigma^* \mathbf{H}) \tag{1}$$

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\varepsilon}\nabla \times \mathbf{H} - \frac{1}{\varepsilon}(\mathbf{J} + \sigma \mathbf{E}) \tag{2}$$

where μ is the magnetic permeability, ε is the electrical permittivity, σ is the electric conductivity, σ^* is the equivalent magnetic loss, and **M** and **J** are the magnetic and electric source current densities. These equations provide the basis of the FDTD method, calculating electric and magnetic field intensities in a staggered grid (**E** and **H** are shifted by a half of the grid spacing).

Using a dense discretisation (at least $\lambda/10$ where λ is the wavelength of the incident light) and directly solving the electric and magnetic field in a leap-frog scheme leads to a stable algorithm suitable for many problems in physics and engineering. Combined with the many related additional algorithms that have been developed in the last fifty years, FDTD can be used for nearly any material. If treated carefully, it can be accurate down to 1% in the area of nanoplasmonics as reported in Ref. [19]. Its main disadvantages are the necessity of a dense discretisation and complicated handling of smooth boundaries with an arbitrary orientation (staircasing effect). On the other hand, due to its simple formulation it can be easily implemented. Furthermore, it results, such as electromagnetic field distribution in the simulation domain, can be readily interpreted. It is also highly parallelisable as it consists of relatively simple equations repeatedly solved in all grid points, which is also the reason why it is convenient for graphics card (GPU) computing [16,17,20,21].

2.2. Structure of the program

The package consists of two main executables, GSvit and XSvit. GSvit is the backend numerical solver, which can be run as a command line program. It is written in C and optionally uses graphics cards based on the Nvidia CUDA computing interface. XSvit is the graphical interface and also serves as a parameter file editor, either via visual definition of the model or direct editing of the parameter file. XSvit can be used for creating or altering the parameter files, running the numerical solver and running Gwyddion [33] which serves as the viewer for most of the results. It is written in C, with Gtk+ graphical toolkit.

Apart from the basic FDTD computational approach known as Yee's algorithm [22], which is used for electromagnetic field propagation, the software implements also the following extensions:

- Point and dipole sources, total/scattered field (TSF) and pure scattered field (SF) source formulations, including sources in an infinitely extending layered medium [23] and focused sources based on the plane wave decomposition [24].

- Periodic boundary conditions, simple second order boundary conditions [25] and convolutional perfectly matched layer boundary (CPML) conditions [4].
- Near-field-to-far-field (NFFF) transform using the time domain approach [28] and field teleportation [27] for crossing empty spaces.
- Staircasing reduction via effective medium and local subgridding [26].
- Dispersive media treatment using the piecewise linear recursive convolution (PLRC) approach and auxiliary differential equations (ADE) approach [4].

Full details about functionality of different FDTD extensions and their relation to the basic Yee's algorithm can be also found in the literature, e.g. the monograph [4].

2.3. Parameter and media files

The command line part, GSvit, is controlled by a parameter file passed as the only command line argument. It is a plain text file defining all the aspects of the computation as a sequence of two-line directives. The first line always consists of a keyword defining the directive type, the second depends on the keyword and specifies the values. As an example consider the following text file:

```
POOL
200 200 200 1e-8 1e-8 1e-8

COMP
1000

SOURCE_TSF
10 10 10 190 190 190 0 0 0 1 633e-9 1

MEDIUM_VECTOR
material_file.txt

MEDIUM_SPECTRAL
5 20 -1 42

OUT_IMAGE
Ey 150 100 -1 -1 ey_snapshot
```

It provides all the basics for running a simulation on an artificially modified sphere as discussed in section 4.5. `POOL` specifies the computational domain of $200\times200\times200$ voxels with voxel spacing of 10 nm (10^{-8} m) and `COMP` runs 1000 steps of the Yee's algorithm to propagate the planar electromagnetic wave, injected via Total/Scattered field source boundary condition defined by `SOURCE_TSF`. This wave interacts with an object described by the material file given by `MEDIUM_VECTOR` and modified with `MEDIUM_SPECTRAL` modifier by adding roughness with 5 voxels RMS, 20 voxels correlation length and using 42 as the random seed. The output, given by `OUT_IMAGE`, will consist of electric field E_y images taken in the yz plane with $x = 100$ voxels. First 150 steps will not be saved.

The only missing piece is the object the light wave interacts with. A germanium sphere with radius of 40 voxels centred in the domain at $(100, 100, 100)$ can be defined by a single-line `material_file.txt`

```
4 100 100 100 40 99 Ge
```

In practice multiple outputs are usually set, absorbing boundary conditions are added by `BOUNDARY_ALL` and a near-field to far-field transform (requested by a `NFFF` directive) is used to get the far-field response of the optical system. The full documentation of parameter files is provided on the project pages [18], alongside with many examples how to run different calculations, including complete examples of object boundary modification using methods presented in section 3.2.

2.4. GPU calculations

By default, all the algorithms are executed on the main processor (CPU), using double precision arithmetic with parallelisation handled by OpenMP [29]. Optionally, some algorithms can be run on the GPU instead of the CPU, based on the use of NVIDIA CUDA tools [30]. GPU can be utilised in two different ways:

1. At each step (i.e. Yee step) all the data are synchronized back to the CPU to perform operations with higher precision arithmetic, output results, run external sources, or simply to save graphics card memory (allocating only data necessary for each kernel call). This approach typically leads to a three- to six-times speedup compared to pure CPU execution, namely due to limited bandwidth of data transfer between the main memory and graphics card memory.
2. All the computations are performed on the GPU and the data are synchronised only for outputs (i.e. not every step). Time values of NFFF data are produced at the end of the computation. This can lead to more than seventy-times speedup as shown below even for quite a straightforward FDTD implementation on the GPU.

For GPU kernel implementations and for nearly all the algorithms, a one-to-one correspondence between GPU threads and computational space points is preserved. The basic data structures are stored in the GPU global memory. Each thread takes the neighbour values (or whichever are necessary for the computation), and after the computation it updates the value in the global memory. On one hand, this approach does not utilise the device memory optimally as shared memory is not used at all. On the other hand, it avoids code variants for different memory accesses, keeping the code simple—while still providing a significant speedup as discussed below.

In some cases, for instance NFFF computations, it is not possible to establish a one-to-one correspondence between GPU threads and computational space points as this could lead to memory access conflicts in the accumulators, where the integration of surface currents over the NFFF boundary is being performed. A set of auxiliary accumulators was used in this case to split the integration between more GPU kernels without accessing the same global memory.

3. Data loading and generation methods for realistic geometry treatment

3.1. Data input

The basic way to define the material distribution in the FDTD computational domain is to add solid geometrical shapes—such as boxes, spheres or cylinders—with given material properties. By means of Constructive Solid Geometry, i.e. using unions and intersections of these basic entities, even quite complicated structures can be created.

Even more general structures can be defined as tetrahedral meshes, as produced for example by Tetgen [31] from input created in a 3D modeller such as Blender [32]. The data can also be entered as a voxel-by-voxel array of material properties in the .vtk format, allowing a detailed control of the material distribution, e.g. producing gradients in optical properties.

Surface topographies can also be input as Gwyddion height fields (height maps), i.e. 2D arrays of height values. Gwyddion [33] is a modular multiplatform software for scanning probe microscopy (SPM) data analysis which can process most of native SPM manufacturer data formats (plus several data formats used in confocal microscopy, scanning electron microscopy and other microscopy techniques). It also features many data synthesis algorithms, so artificial surfaces can be prepared in Gwyddion and loaded into GSvit. Direct use of the height fields is useful namely in virtual SPM applications of FDTD in which we simulate the interaction of light with a surface at every pixel of the image. The height field can be then shifted arbitrarily with respect to the beam in order to produce data from the requested position.

Using a combination of different geometry loading approaches, we can model structures representing many real experimental situations. However, for the handling of statistical phenomena like surface roughness, this is still often only the starting point. The loaded objects can be further modified within GSvit to perturb their boundaries.

3.2. Object modifiers

In order to add inhomogeneities such as roughness or variable layer thickness to inputs prepared for computation, one could use an external software and prepare such data in a 3D modeller (for instance) and load them as a tetrahedral mesh. This approach turned out to be impractical as the control of surface parameters is quite limited in 3D modelling software. Moreover, to perform statistical analysis of electromagnetic field propagation, we need to run many simulations with different realisations of a particular roughness that would all have to be created in the 3D modeller.

A better approach is to alter the surface topography of selected objects within the FDTD package at the beginning of the computation. In GSvit this is achieved by object modifiers, allowing an automatic creation of different instances of the rough object with given roughness parameters. We can then run multiple calculations of the light-object interaction and evaluate the results statistically. Here we describe three basic algorithms for object modification developed for GSvit: vector displacement modifier (sec. 3.2.1), random Gaussians and arbitrary function modifiers (sec. 3.2.2), and the growth modifier (sec. 3.2.3). The second method was developed entirely for purposes of this work. The first is analogous to displacement mappings used in computer graphics; the third was based on existing models of surface growth [35].

Recently there has been growing interest in application of polynomial chaos and stochastic FDTD methods to the problem of modelling systems with rough boundaries [37,36]. They can be considerably more efficient than the classical Monte Carlo approach implemented in GSvit. However, Monte Carlo can be immediately used with any simulation of physical roughening process (added as a roughening operator to GSvit), whereas these approaches require efficient parametric representations—and the path from the former to the latter is seldom short or straight. The choice made in GSvit enables the accommodation of experimental roughening procedures. For maximum efficiency with well-studied roughness models one may have to look elsewhere.

3.2.1. Vector displacement modifier

In this method voxel values are altered by replacing them with values taken from displaced positions, defined by the vector displacement field. This has no effect within a homogeneous region but perturbs object boundaries—and can also be used to perturb material gradients or other properties. The vector displacements consist of three 3D arrays (x, y, z) of the same volume as the computational domain. They are filled by correlated noise using a spectral synthesis method. To do this, Fourier transform coefficients are set to define suitable spectral properties of the noise and an inverse Fourier transform is used to get the displacement fields.

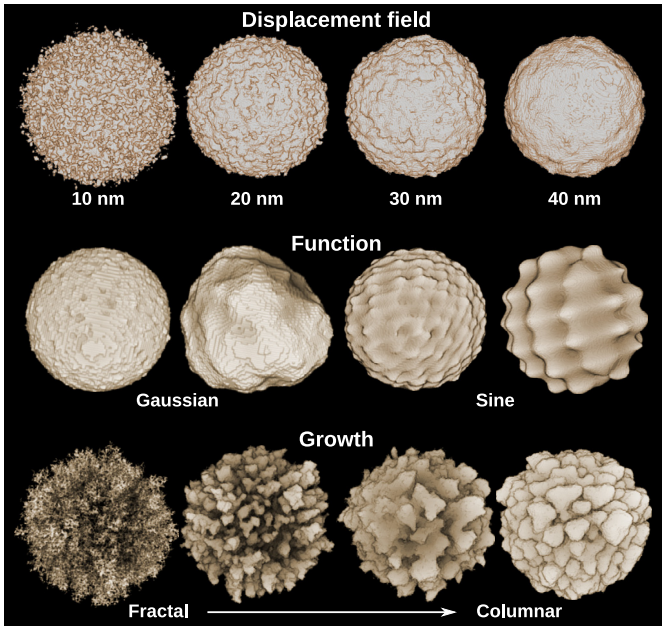

Fig. 1. The effect of the various surface roughness modifier types applied to a sphere of 1 µm diameter. In the case of the vector displacement field (section 3.2.1) the correlation length is from 10 nm to 40 nm as indicated. The function modifier (section 3.2.2) is shown for two random Gaussian function modifiers and two function modifiers constructed as a mix of sines. For the material growth modifier (section 3.2.3), the relaxation jump probability and upsampling factor is gradually increased resulting in a progression from fractal-like to columnar roughness.

In the case of Gaussian roughness, the most common roughness model, the Fourier transform coefficients moduli are set to the product of three Gaussian functions, in x, y and z as follows:

$$F(\mathbf{K}) = \exp\left[-(\pi T/2)^2 |\mathbf{K}|^2/2\right], \tag{3}$$

where $\mathbf{K} = 2(i/x_{\mathrm{res}}, j/y_{\mathrm{res}}, k/z_{\mathrm{res}})$, T is the desired correlation length and x_{res}, y_{res} and z_{res} are the voxel resolutions in respective directions. Phases are set as uniform random numbers in the range of $[0, 2\pi)$. This produces a displacement field with the correct correlation length. The variance parameter S_q is set by normalising the integral of $|F|^2$ to S_q^2. The material in each voxel is then replaced by the material from a nearby voxel determined by the x, y and z shifts and rounded to an integer. An advantage of this procedure is that its natural parameters are the output roughness parameters S_q and T. An example of the results obtained for Gaussian roughness and different correlation lengths is shown in Fig. 1.

3.2.2. Random Gaussians and arbitrary function modifiers

To add roughness with shape controlled in the direct space (as opposed to frequency space) we created an alternative procedure, based on blurring the object boundary, altering the blurred arrays and thresholding. The following steps are performed (see Fig. 2 for a schematic illustration):

1. First the object is formed by a set of voxels with values set to one; the rest of the computational domain is set to zero (void).
2. The object is blurred in 3D to create a smooth transition between the material and void regions.
3. The object is altered by adding 3D Gaussians to random locations in the computational domain. They are added with both

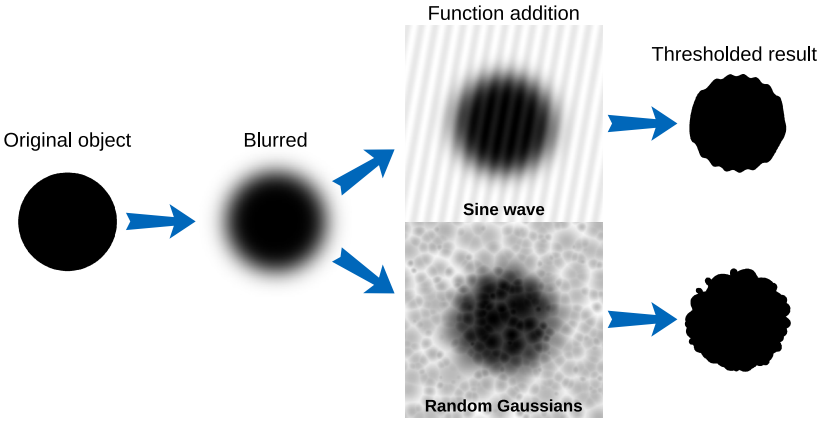

Fig. 2. Scheme of object boundary modification by function addition, illustrated for a sine function and random Gaussians.

positive and negative signs and a selected distribution of half-widths.
4. The resulting floating point array is thresholded, producing again in an object with sharp boundaries, now perturbed by the Gaussians.

The procedure has three parameters: half-width of the Gaussians, the number of iterations and the probability of placing a Gaussian at a voxel. By varying the parameters and/or running the whole algorithm multiple times, complex roughness can be created on the object surface. An example is given in Fig. 1.

A major disadvantage of this procedure is that the relation between its parameters and resulting roughness parameters is complex and not known analytically. Obtaining roughness with prescribed properties requires solving an inverse problem. Practically, this is done by roughening flat surfaces and evaluation of their statistical parameters, e.g. using the surface analysis available in Gwyddion. This is discussed in more detail in sec. 4.2. A benefit of the method is that the range of surfaces that can be generated is much broader than for spectral synthesis.

Instead of random roughness addition, an arbitrary 3D function can be used instead to perform the object modification. The procedure is almost identical, only the blurred object is now modified by a user-defined function evaluated at every voxel. An example is shown in Fig. 1 for modification by sine functions.

3.2.3. Growth modifier

To create more realistic rough surfaces that would be statistically similar to surfaces of deposited materials (e.g. by evaporation or sputtering), we implemented a Monte Carlo growth simulation that can be applied to any object or set of objects. The procedure is based on ballistic particle deposition with limited relaxation to local energy minima (maximum number of bonds). Shadowing during the growth leads to the formation of a variety of columnar structures. The procedure is similar to those presented in works dealing with the simulation of sputter deposition of thin films [34,35] and was used for 2D material modification in our previous work [38]:

1. First the selected object and its surroundings are upsampled (usually by a factor of 2 or 3) to get a higher resolution for the growth process, as generation of realistic surface roughness by the growth procedure needs sufficiently large objects (in terms of the number of voxels).
2. A predefined number of particles is sequentially simulated, coming from all the faces of the computational volume with a random distribution of direction and hitting the object.
3. When a particle hits the object, it can relax to locations in its neighbourhood if it is energetically favourable (more bonds) and when the probability of such a jump is above some threshold value (that can be understood as a temperature factor in the context of deposition techniques).

The process is repeated for the requested number of particles, typically in the order of millions. Instead of letting particles to hit the object from all directions, the particle source can be limited for instance to selected faces of the computational domain, matching more closely geometrical conditions in various deposition techniques. The procedure has three parameters: relaxation probability, maximum relaxation radius, and the number of particles.

The growth method can simulate a wide variety of randomly rough surfaces, from locally smooth surfaces (high jump probability, large radius), to columnar structures, to very porous surfaces with a self-affine fractal structure (low jump probability, small radius). The range of roughness parameters that can be obtained is discussed in the next section—but again, the relation between simulation parameters and surface parameters is complex. An example of structures grown on a sphere for different algorithm parameters is shown in Fig. 1.

4. Results and discussion

In this section we present the results of software performance tests to show and discuss the achievable simulation speeds and sizes of the computational domain. We also verify the roughness addition approach by comparing its results with a classical optical theory, which is for this range of surface parameters the Rayleigh–Rice theory. Finally, we present two examples of running a simulation on a statistical ensemble of rough object realisations to show the variation of the output field values related to different roughness realisations They were selected to demonstrate practical applications of the two modifiers most relevant in nanoscale optics problems—vector displacement for a well-defined roughness and growth modifier to simulate a deposition process.

4.1. CPU and GPU performance

In Fig. 3 we present the Yee's algorithm speed as evaluated on a number of available computer processors (both single thread and multithreaded) and graphics cards. The figure shows the time taken by 400 steps of the Yee's algorithm in a $200\times200\times200$ voxels large computational domain. In Fig. 4 the time needed for a single calculation is plotted as a dependence on the number of threads used on a CPU. To see the dependence for a large number of threads we used a high performance computing system for

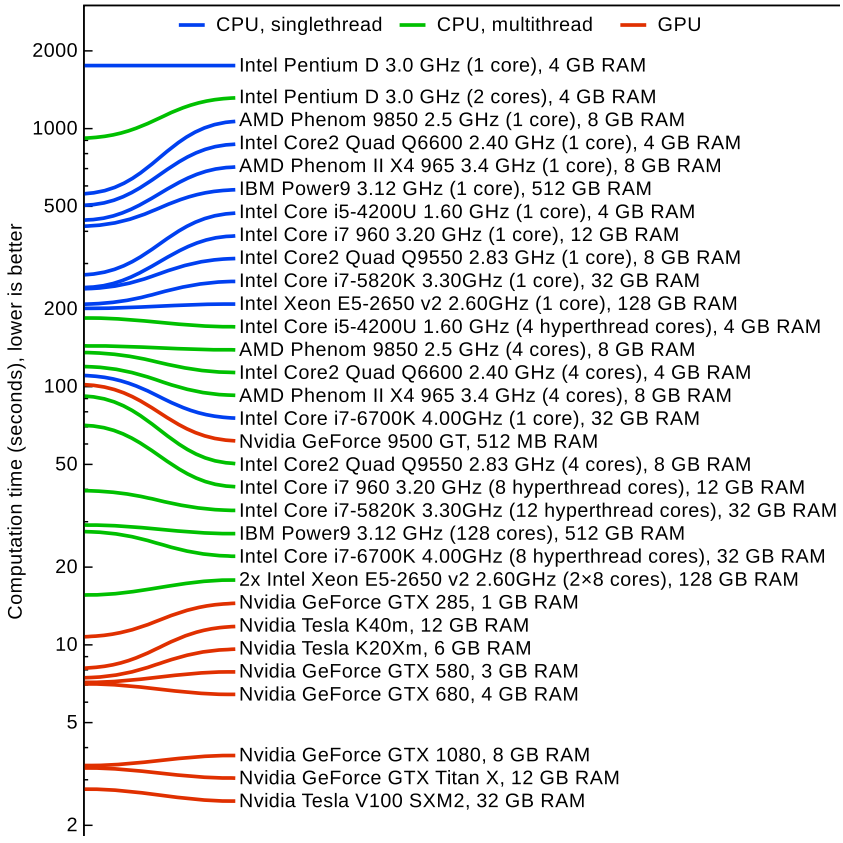

Fig. 3. FDTD implementation performance on different computer processors and graphics cards (time necessary for calculating the same problem on different systems). (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

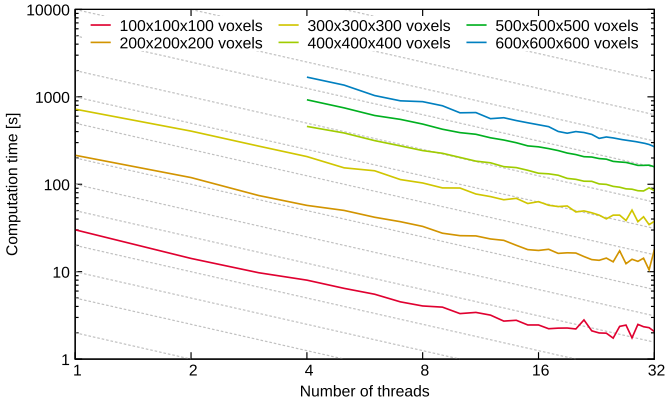

Fig. 4. FDTD performance on a 32-core AC922 (8335-GTG) system up to 32 threads shown as time necessary for calculating the same problem in all the cases. Dashed lines correspond to ideal linear scaling.

this test. The IBM® Power® System AC922 (8335-GTG) server is a 2-socket server that offers 32 cores. We can see that the algorithm scales almost linearly with increasing number of available cores. The deviations for large number of cores and large problem sizes indicate that it ceases to be purely CPU-bound and becomes limited also by memory bandwidth in these cases.

An important aspect of FDTD is also the required memory. The computation domain size is usually limited by the size of the computer memory. Therefore, even if we remove the speed bottleneck by using GPU, the computed volume bottleneck persists. As an example, on Geforce GTX Titan X with 12 GB RAM, the limit is 328 million voxels, which, at space discretisation of $10\times10\times10$ nm, which we typically use for calculations in the visible range, means a volume of approximately $7\times7\times7$ micrometres.

4.2. Roughness generation algorithms

While the vector displacement modifier creates surfaces whose statistical properties can be controlled by the modifier parameters—roughness value and correlation length—this is not the case for the growth modifier. A connection between algorithm parameters and resulting statistical properties had to be established.

In order to test performance of the ballistic deposition-based growth algorithm we carried out a numerical study of the dependence of roughness statistical properties on algorithm parameters. All the methods described in this paper can be applied to objects with any geometry. However, for the evaluation a flat base object is preferable because a complex base object geometry makes the separation of geometry and roughness more difficult and increases the uncertainty of obtained roughness parameters. A flat substrate allows utilising common statistical analysis techniques for surface measurements. Therefore, the following methodology was used (see Fig. 5A):

1. A thin parallelepiped object was placed in the centre of the computational volume.
2. A growth object modifier was applied to the object, using parameters chosen from a predefined set.
3. A central part was cut from the modified object to avoid border effects. The central part was used in all following evaluations.
4. The fractal dimension was calculated using the cube-counting method.
5. A height field was constructed from the central part (see Fig. 6 for few examples).
6. The thickness of the deposited (or subtracted) material was evaluated from the average surface value.

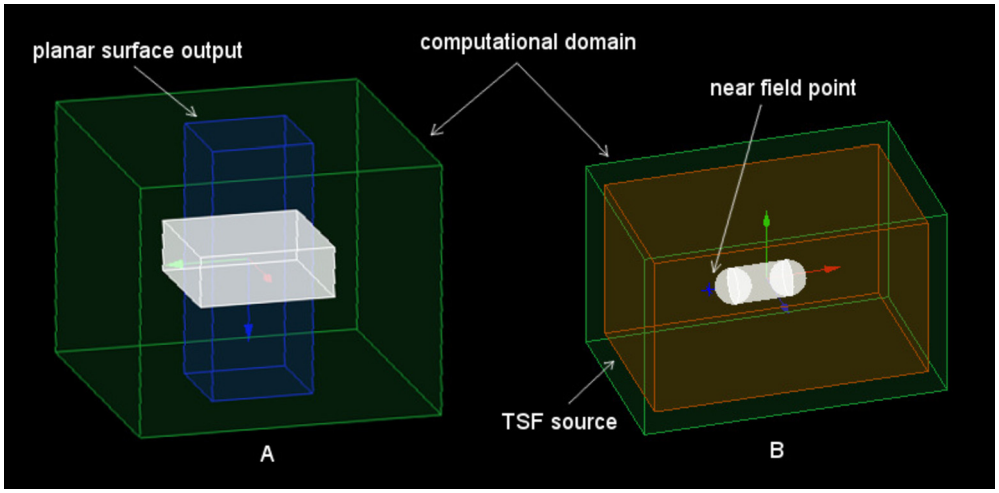

Fig. 5. Computational domain geometry for (A) roughness evaluation and (B) rod antenna field enhancement calculation.

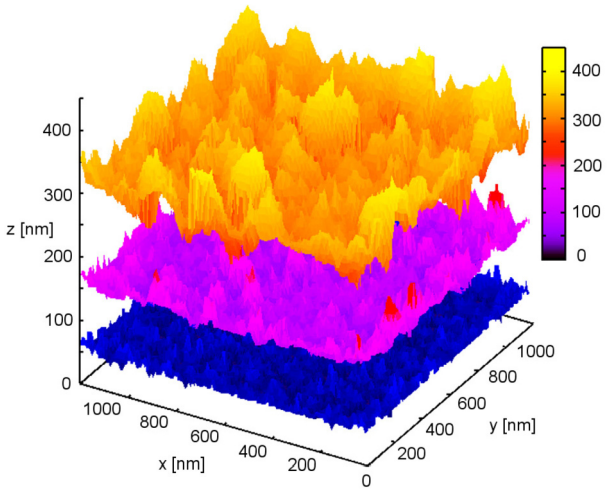

Fig. 6. Series of snapshots of the growth process for the same random seed and increasing number of particles. The z axis placement of the surfaces corresponds to real layer thickness.

7. The root mean square value S_q of roughness was evaluated from the surface.
8. The power spectral density function (PSDF) was calculated and averaged from the individual surface profiles and fitted by Gaussian PSDF model:

$$W_1(K_x) = \frac{{S_q}^2 T}{2\sqrt{\pi}} \exp(-K_x^2 T^2/4)\,, \tag{4}$$

where T is the correlation length. Note that this is based on assumption that the surface correlation function is Gaussian, which is not necessarily true for many of the surfaces.

The parameters of the growth object modifier were chosen to cover the widest possible spectrum of statistical properties the methods can produce (based on a preliminary parametric survey). A snapshot of a single growth process, taken for different number of particles is shown in Fig. 6. By combining all the results, a database of surface properties was created, referring to particular object modifier parameters. By searching in the database we can choose roughening parameters leading to the desired surface roughness.

Fig. 7 shows the summary of all the surface statistical parameters that were achieved within the growth modifier study using various probability factors, numbers of particles, etc. We can see that although the three statistical parameters are correlated (which is expected as it follows from the roughness growth theory [35]), a wide range of roughness parameters can be simulated.

Since the surface roughness generation is a statistical process, every surface has slightly different statistical properties even if the input parameters are identical. We repeated the surface modification with the same parameters fifty times to assess the variation of resulting statistical parameters with random seed used for the calculation. For the fifty runs the following results and variances were obtained: $S_q = 1.92 \pm 0.05$ nm, $T = 2.3 \pm 0.1$ nm, $D_f = 2.33 \pm 0.06$. From this we can conclude that the individual randomly rough surface realisations are sufficiently similar for practical simulations.

Checking what surface properties are obtained for a flat object with selected modifier parameters can be seen as a general technique for using the object modifiers in practice. However, it still has some disadvantages. First, such preparation can be time-consuming even if we normally do not need to build an entire database similar to the one used here. Second, if the objects are concave, with deep trenches, caverns or similar complicated geometry, the resulting statistical properties may differ from the predicted ones. However, flat geometry is the only case for which roughness parameters such as S_q can be evaluated in a standard manner.

4.3. Comparison to theoretical calculations

Adding well-controlled imperfections to objects in FDTD calculations is useful for studying their impact on the optical response for geometries that are more complex than what could be handled by classical optical theories. For the same reason, however, a verification of FDTD results by comparing them with explicit theoretical calculations must be done using a simpler model system which can be described by the theory. Therefore, we compared the FDTD-computed specular reflection from a rough surface with second-order Rayleigh–Rice theory.

Rayleigh–Rice theory expresses the Fresnel coefficient r_q of light specularly reflected from a rough system as the sum of the corresponding coefficient for a smooth system and perturbation term Δr_q:

$$r_{q,\mathrm{rough}} = r_{q,\mathrm{smooth}} + \Delta r_q\,. \tag{5}$$

Index q distinguishes the polarisation (p or s). The perturbation term is expressed using an integral with spectral density of roughness $W(\mathbf{K})$ over the space of spatial frequencies $\mathbf{K} \in \mathcal{R}^2$:

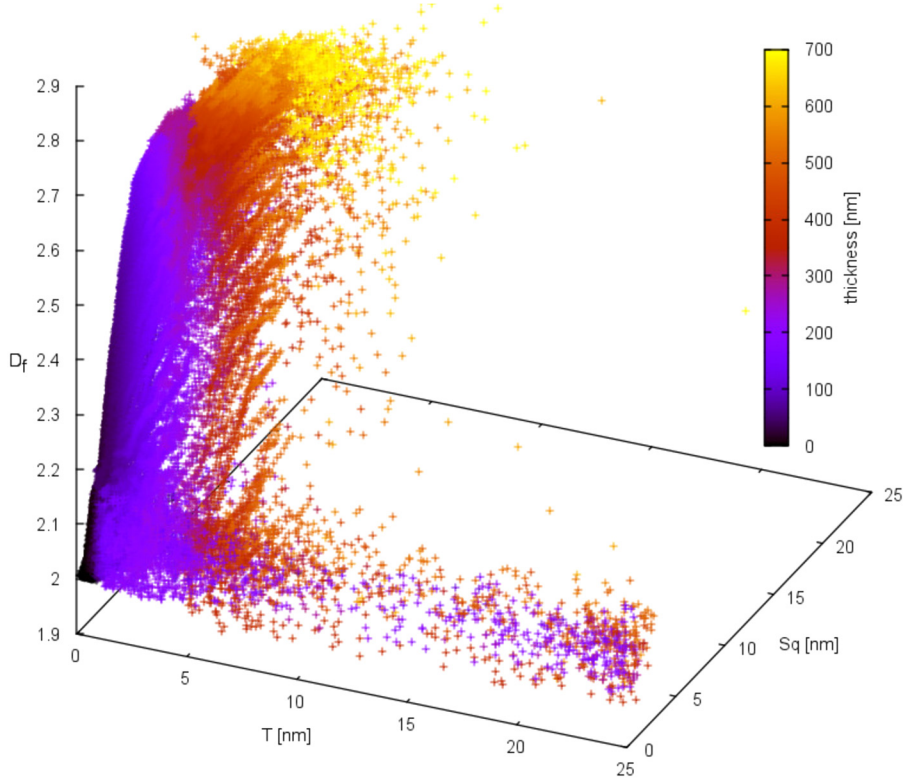

Fig. 7. Database of resulting statistical parameters for various parameters of growth object modifier.

$$\Delta r_q = \int_{\mathcal{R}^2} f_q(\mathbf{K})\, W(\mathbf{K}')\, \mathrm{d}\mathbf{K}\,, \tag{6}$$

where f_q is a complicated function of $\mathbf{K}$, optical constants and the incidence angle ϑ_0 [39–41]. Vector $\mathbf{K}'$ is equal to $\mathbf{K}' = \mathbf{K} - n_0 k_0 \sin\vartheta_0 \mathbf{e}_x$, where n_0 is the refractive index of the ambient and k_0 is the corresponding wave vector. Other approaches for modelling of light specularly reflected from rough systems exist and are widely used in optics, for instance diffraction theories or effective medium approximations. However, Rayleigh–Rice theory is valid for larger S_q/λ ratios than effective medium approximations (EMA) and does not require locally smooth surfaces, unlike diffraction theories. It is, therefore, a suitable theoretical approach for the following comparison.

The model system was a germanium prism with varying Gaussian roughness modelled by spectral synthesis for $S_q = 0$ to 50 nm and fixed $T = 100$ nm. The FDTD calculation was run in a computational volume of 500×500×150 voxels spaced equally in all directions by 10 nm and containing the $5 \times 5 \times 1.5$ µm^3 large Ge prism. A Gaussian beam with wavelength of 633 nm was incident on the prism in the z axis direction. The specular reflection was evaluated using NFFF transformation. The results were normalised to the reflectance of smooth surface in order isolate the effect of roughness by eliminating the differences between the analytically known reflection from a smooth plane and FDTD results. In this case the difference was about 5% because of the combination of large extinction coefficient of germanium and FDTD discretisation that is fine enough to represent the chosen roughness, but not fine enough to represent the spatial field decay in germanium with high accuracy. If the voxel size is 4 times smaller the error in smooth germanium reflectance drops below 1%, however such voxel size would already need an extremely large mesh if the roughness should be added.

The results are compared in Fig. 8a. It can be seen that FDTD slightly underestimates the impact of roughness. It is most likely an effect of staircasing, i.e. discretisation of simulated roughness. In particular, FDTD gives almost no effect for S_q small compared to voxel size—such small roughness cannot be represented correctly in the voxel grid. This could be improved either by conformal modelling (at the cost of computation speed) or by using an EMA, weighing the materials in each voxel.

Fig. 8a shows the results of using the Bruggeman EMA [42], frequently employed to model roughness, for representing the effective medium in voxels filled only partially by Ge. The addition of EMA improved the agreement for large S_q, where it could help representing fine roughness details not captured by the voxel grid. However, for small roughness the results in fact became worse—the curve has non-zero derivative at $S_q = 0$. This is not surprising; it is well known that the relation between EMA effective roughness layer and S_q should be quadratic [43], but voxel material averaging gives a contribution linear in S_q.

There could be also other factors contributing to the differences. For instance the displacement field modifier can generate overhangs, whereas the Rayleigh–Rice theory assumes that surface is described by a (single valued) function. Therefore, the assumptions in the two approaches do not match exactly. Nevertheless, we can conclude that the FDTD results agree well with theoretical calculations for the reflectance of rough surfaces—provided that one takes care to choose a voxel size which can represent well the rough material boundary. In addition to keeping the voxel size below tenth of the wavelength, as needed by the Yee algorithm, it must also be kept small enough to incorporate the fine details of the roughness. In Fig. 8b we show how the result depends on voxel size for one of the roughnesses shown in Fig. 8a, with mean square roughness of 32 nm and correlation length of 100 nm. It can be seen that when the voxel size increases, the reflectance increases towards the smooth surface value (which would be 1). When EMA is used, the effect is less pronounced as the staircasing effect is reduced. The range in which we can prove it is however limited—by the amount of computer memory on the lower end and discreti-

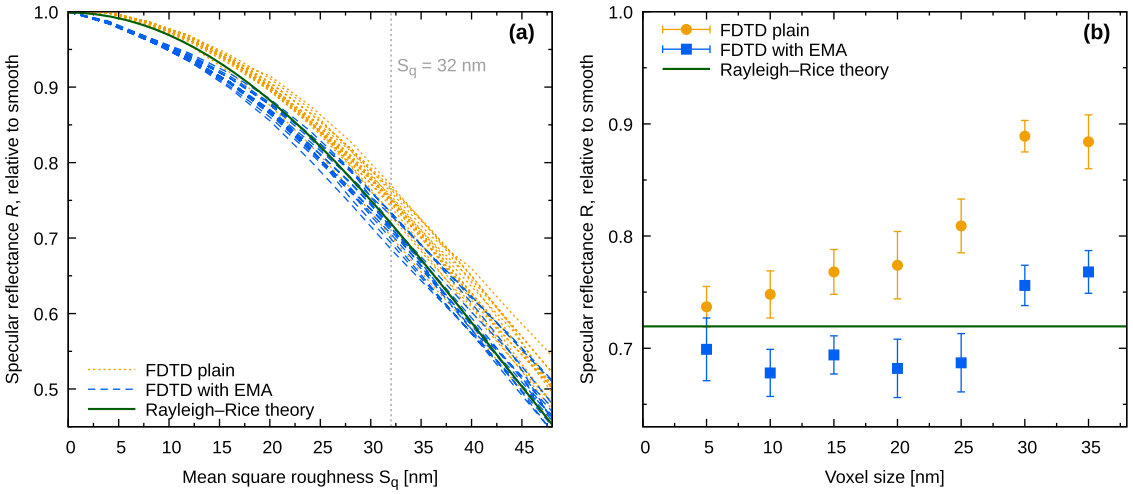

Fig. 8. (a) Verification of the roughness addition by comparison to Rayleigh–Rice theory. (b) Dependence of the rough surface reflectance on the voxel size. Each value is an average of 15 FDTD runs, error bars correspond to the standard deviation.

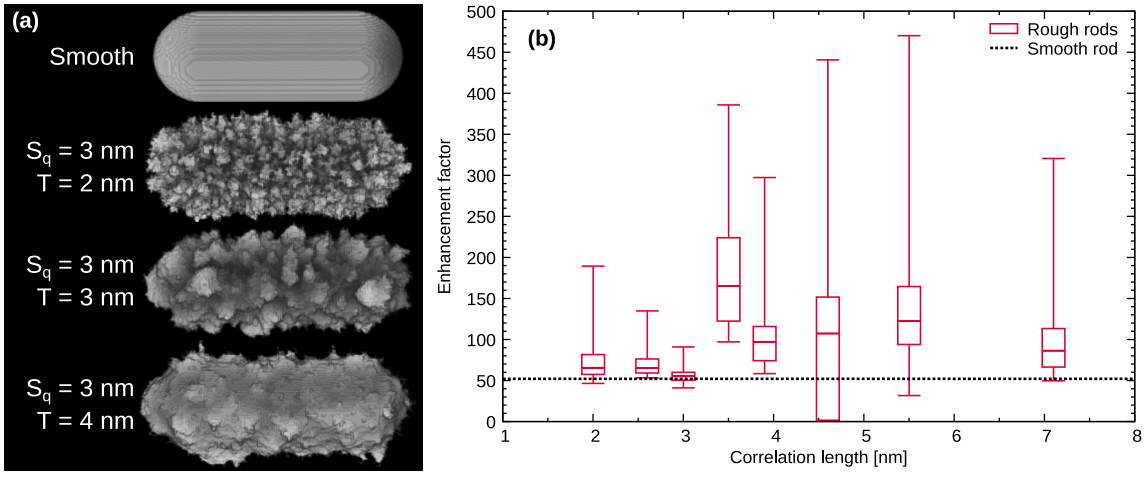

Fig. 9. (a) examples of generated rough rod antennas; (b) statistical results for the peak value of field enhancement as a function of roughness correlation length.

sation limit required by the Yee algorithm on the upper end. It should be noted that the Gaussian roughness created by spectral synthesis is locally smooth, which is certainly a benefit when we discretize the surface. For other roughness models, providing surfaces with higher amount of high spatial frequency components it might be more difficult to find a suitable voxel size. A convergence test similar to the one shown in Fig. 8b can help with such decision.

4.4. Application example: a rod antenna

Here we present an example of interaction of light with a more complex rough nanoobject and a statistical evaluation of the influence of surface roughness. The example is still one of the simplest possible—an isolated nanoantenna formed by a single cylindrical rod capped by hemispheres. Using the approach presented above, in particular the growth object modifier, we created 60 different rough rod realisations for each set of parameter values. We then evaluated the resulting local field enhancement close to the rod apex. Spectral dependences of the field enhancement were calculated, and for a larger statistical ensemble also the enhancement value for the peak in this spectrum was evaluated.

Details of the numerical experiment are as follows: the computational domain was a box with sides of 410 nm × 260 nm × 260 nm, with a voxel spacing of 1 nm (see Fig. 5B for schematics). It contained an aluminium rod of length approximately 120 nm (varying with different roughness realisations), placed in vacuum. We used the growth modifier to add a roughness to the rod, the parameters were chosen the database of the growth modifier parameters generated on a flat surface as described in section 4.2. All rods prepared for this calculation had the same mean square roughness $S_q = 3$ nm, but different correlation lengths T from 2 to 8 nm. From the resulting thicknesses we tested that the growth modifier produces results are consistent with the simulations of flat surface that were used for database creation. A few examples of the generated rods are shown in Fig. 9a.

Within the FDTD calculation the rod was illuminated by a plane wave using the total/scattered field approach with wavelengths ranging from 250 to 800 nm. The calculation was performed wavelength by wavelength. A second order absorbing boundary condition was used to let the light leave the computational domain. To reach a steady state, 12000 time steps were calculated, using the time step value of $0.8 \cdot \Delta t$, where Δt is the time step evaluated from FDTD stability criterion. Piecewise linear recursive convolution approach was used to treat the rod, formed of aluminium. The time step was adjusted by the factor 0.8 to keep the metal treatment approach stable. The computation output consisted of time dependences of field values in a defined position from the ideal rod apex in direction of its axis. Field amplitudes were evaluated

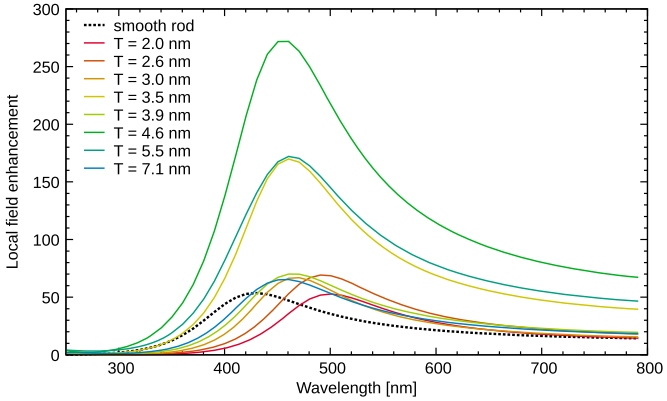

Fig. 10. Spectral dependence of the field enhancement close to smooth and rough rod apex. Roughness parameter S_q was 3 nm for all the rough rods.

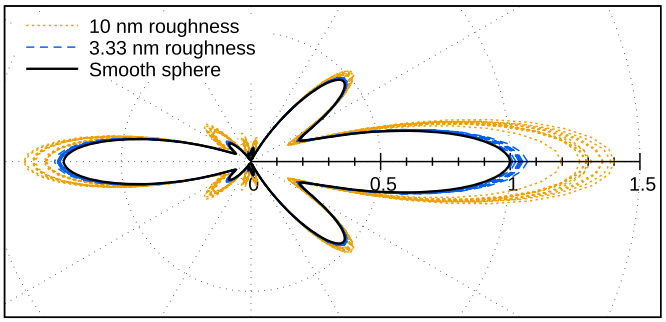

Fig. 11. The effect of roughness on Mie scattering on a spherical particle. Results for two different roughness values, 3.3 nm (5 voxels) and 10 nm (20 voxels), are compared with smooth sphere (shown as solid black curve). The light intensities are plotted as relative.

after reaching a steady state after a transient period at the start of the computation.

The spectral dependences of the field enhancement in the vicinity of the rod are plotted in Fig. 10 for rods with different correlation lengths. We can see that the overall enhancement varies in an order of magnitude, which is an effect of randomness in the rod surface shape. In some cases the roughness increases the field at the apex, namely due to lighting rod effect at sharp features pointing towards the point where the field is evaluated. In some cases the field intensity is even slightly smaller than for a smooth rod. We can also see shifts in the peak wavelength (maximum enhancement), which is mostly related to the rod length—despite taking care to prepare rods with the same mean length. Although this could still be caused by imperfections in our rod modification approach, it is possible mean rod length is no longer the parameter determining the peak wavelength when a relatively large roughness is present.

The 60 different realisations of each rough rod were used to observe how the peak value of the field enhancement varies among different instances. To prevent a systematic error in the effective rod length, we searched the maximum value of the enhancement for each rod type independently. In Fig. 9b results of statistical analysis of the peak value of the field enhancement at the vicinity are shown, as evaluated from this statistical ensemble. We can see that there is a large variation in the field enhancement factor, particularly for larger surface correlation lengths. Yet there is no clear dependence of the mean enhancement factor on the correlation length. We could expect an influence of the correlation length on the enhancement for much longer rod antennas and longer correlation lengths, which will be the subject of a following study. The presented approach gives us an estimate of how much the real rod enhancements can vary in contrast to an ideal rod shape. A possible explanation is the presence of some hot spots close to the rod end, similarly to what was observed in tip-enhanced Raman spectroscopy [14] or in surface-enhanced Raman spectroscopy [15].

The capability of evaluating the local variances of the electromagnetic field intensity caused by roughness is the most important feature of the presented approach. Results like those shown in 9b can help us understand how the imperfections caused by the manufacturing process in nanoscale optics propagate to the optical performance of the device. By coupling the FDTD solver with algorithms for local data modifications we can perform such studies in an efficient way.

4.5. Application example: a rough particle Mie scattering

In the second example we show an application of the vector displacement modifier to light scattering by rough spheres. Light scattering on an ideal spherical particle is described by the well known Mie solution [44]. This infinite series-based solution of the scattering problem is employed in many practical applications, e.g. in aerosol particle counters. Real particles are, however, not ideally spherical. Many works have proposed methods for inclusion of shape imperfections, both using more complex analytical models and numerical simulations [45]. Surface roughness is one of the common possible distortions and we can use the surface modifiers in GSvit to explore its impact on the angular scattering diagrams. Since the angular dependence of scattering depends heavily on particle radius, it is important to do such analysis for particles of the same size, which can be achieved using the vector displacement modifier.

A sphere of 400 nm radius (equal to 60 voxels for 6.667 nm voxel size) and relative permittivity of 4 was used. The computational domain was $360\times360\times360$ voxels large and the sphere was located in its centre. It was illuminated using the Total/scattered field formalism by monochromatic light with 633 nm wavelength, with 2nd order absorbing boundary conditions. NFFF transformation was used for calculating field amplitudes of light scattered to different directions and perpendicular polarisation components were evaluated.

The reference solution for an ideal sphere was calculated both using FDTD and Mie's scattering. Since staircasing, present in FDTD calculations, leads itself to the introduction of imperfections, we fitted the FDTD result by the Mie's solution to obtain the effective optical radius. For the smooth sphere it was 400.2 nm. We also evaluated a voxel-based effective radius by counting all the voxels occupied by the material and calculating the radius of a sphere with the same volume, which was 400.02 nm for the smooth sphere.

Rough spheres were created by applying the spectral synthesis modifier with correlation length 20 nm and two different roughness values $S_q = 3.33$ and 10 nm. For each roughness value, ten different curves were calculated for different random seeds of the modifier. The results are plotted in Fig. 11.

We can see that for the smaller roughness $S_q = 3.33$ nm the curves match quite closely the smooth sphere solution, with only minor deviations from the reference curve. The corresponding voxel-based radius was 400.18 ± 0.4 nm while the best fit from Mie solution was 400.04 ± 0.17 nm. For the larger roughness $S_q =$ 10 nm (see also the example in Fig. 1) the result deviates from the Mie's solution considerably more, in particular backscattering is increased significantly. Even though the voxel-based radius is nearly the same (400.2 ± 0.9 nm), the best fit from Mie solution is 398.2 ± 0.3 nm, so the effective optical radius seems to become smaller with increasing roughness.

Similarly to the first example the possible variants of calculation are numerous and a systematic study will be topic of a

forthcoming paper; here we wanted to illustrate the benefits of having a combination of a set of surface modifiers with a FDTD solver.

5. Conclusion

We have presented a graphics card-compatible Finite-Difference Time-Domain solver with a set of extensions allowing local modifications of surface properties of objects in the computational domain. By combining fast computing on graphics cards and random surface topography modifications we can perform systematic studies of the effect of surface roughness on various nanoscale optics systems performance. This was validated by comparison to Rayleigh–Rice theory and illustrated by two examples, a study of roughness impact on variations of a rod nano-antenna field enhancement and angular scattering by rough spheres. As the surface modifiers can be applied to basically any object, the range of potential applications of the presented approach is wide.

The benefit of performing the object modifications directly in the framework of the FDTD solver is that the additional information obtained during the roughness addition can be further used, e.g. to set up a local higher density grid or to calculate the effective material properties in the main grid. Even when the parameters used for object topography modification need to be determined via a trial-and-error approach, this allows substantially better control of the roughening process than in standard 3D modelling packages. Surface modification via ballistic deposition also introduces the basic physical phenomena known from diffusion-limited growth studies, and the created roughness should therefore better represent the topography of real objects in the micro- and nanotechnology area. All the presented algorithms are part of a publicly available open source software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The GSvit software development was supported by projects “BeCOMe”, “Advent”, and “MetVBadBugs” in the EMPIR programme co-financed by the Participating States (The Ministry of Education, Youth and Sports of the Czech Republic project numbers 8B18001, 8B17001, and 8B16001) and from the European Union’s Horizon 2020 research and innovation programme; projects “Scatterometry” and “Raman” receiving funding from the ERA-NET Plus programme of the European Commission under Grant Agreement No. 217257 and from The Ministry of Education, Youth and Sports of the Czech Republic (project numbers 7AX12105 and 7AX13005); project CEITEC 2020 funded by The Ministry of Education, Youth and Sports of the Czech Republic under number LQ1601; and by Institutional Subsidy for Long-Term Conceptual Development of a Research Organization granted to the Czech Metrology Institute by the Ministry of Industry and Trade of the Czech Republic.

References

[1] L. Alvarez, M. Xiao, Opt. Commun. 260 (2006) 727–732.
[2] O.J. Martin, Ch. Girard, A. Dereux, J. Opt. Soc. Am. A 13 (1996) 1801–1808.
[3] D. Barchiesi, Ch. Girard, O.J. Martin, D. Van Lebeke, D. Courjon, Phys. Rev. E 54 (1996) 4285–4292.
[4] A. Taflove, S.C. Hagness, Computational Electrodynamics: The Finite-Difference Time-Domain Method, 2nd ed., Artech House, Norwood, MA, 2000.
[5] Q.N. Luu, J.M. Doorn, M.T. Berry, C. Jiang, C. Lin, P.S. May, J. Colloid Interface Sci. 356 (2011) 151–158.
[6] J.-M. Duan, X.-F. Li, L. Yao, S. Pan, M.-D. Chen, Opt. Commun. 282 (2009) 4005–4008.
[7] N. Zhu, J. Song, L. Shao, Opt. Commun. 266 (2006) 117–121.
[8] S.H. Simpson, S. Hanna, Opt. Commun. 256 (2005) 476–488.
[9] F. Festy, A. Demming, D. Richards, Ultramicroscopy 100 (2004) 437–441.
[10] J.L. Kann, T.D. Milster, F.F. Froehlich, R.W. Ziolkowski, J.B. Judgins, Appl. Opt. 36 (1997) 5951–5958.
[11] T.J. Antosiewicz, T. Szoplik, Opt. Express 17 (2007) 10,920–10,928.
[12] A.F. Oskooi, D. Roundy, M. Ibanescu, P. Bermel, J.D. Joannopoulos, S.G. Johnson, Comput. Phys. Commun. 181 (2010) 687–702, http://ab-initio.mit.edu/wiki/index.php/Meep.
[13] P. Zhang, S. Yang, L. Wang, Jun Zhao, Z. Zhu, B. Liu, J. Zhong, X. Sun, Nanotechnology 25 (2014) 245301.
[14] W. Zhang, X. Cui, B-S. Yeo, T. Schmid, Ch. Hafner, R. Zenobi, Nano Lett. 7 (2007) 1401–1405.
[15] J-A. Sánchez-Gill, J.V. García Ramos, Chem. Phys. Lett. 367 (2003) 361–366.
[16] P. Klapetek, M. Valtr, Surf. Interface Anal. 42 (2010) 1109–1113.
[17] P. Klapetek, M. Valtr, A. Poruba, D. Nečas, M. Ohlídal, Appl. Surf. Sci. 256 (2010) 5640–5643.
[18] http://gsvit.net/.
[19] A.C. Lesina, A. Vaccari, P. Berini, L. Ramunno, Opt. Express 23 (2015) 10481.
[20] V. Demir, A.Z. Elsherbeni, Appl. Comput. Electromagn. Soc. J. 25 (2010) 303–314.
[21] S. Adams, J. Payne, R. Boppana, in: DoD High Performance Computing Modernization Program Users Group Conference, 2007, pp. 334–338.
[22] K. Yee, IEEE Trans. Antennas Propag. 14 (1966) 302–307.
[23] I.R. Capoglu, G.S. Smith, IEEE Trans. Antennas Propag. 56 (2008) 158–169.
[24] I.R. Capoglu, A. Taflowe, V. Backman, Opt. Express 16 (2008) 19,208–19,220.
[25] Z.P. Liao, H.L. Wong, B.P. Yang, Y.F. Yuan, Sci. China Ser. A 1 (1980) 063.
[26] M.W. Chevalier, R.J. Luebbers, V.P. Cable, IEEE Trans. Antennas Propag. 45 (1997) 411–421.
[27] R.E. Diaz, I. Scherbatko, J. Comput. Phys. 203 (2005) 176–190.
[28] O.M. Ramahi, IEEE Trans. Antennas Propag. 45 (1997) 753–759.
[29] https://www.openmp.org/.
[30] NVIDIA CUDA Programming Guide, Version 2.0, http://www.nvidia.com/.
[31] H. Si, TetGen. A quality tetrahedral mesh generator and three-dimensional Delaunay triangulator, http://tetgen.org/, 2007.
[32] http://www.blender.org/.
[33] D. Nečas, P. Klapetek, Cent. Eur. J. Phys. 10 (2012) 181–188, http://gwyddion.net/.
[34] J.T. Drotar, Y.-P. Zhao, T.-M. Lu, G.-C. Wang, Phys. Rev. B 62 (2000) 2118–2125.
[35] A.-L. Barabási, H.E. Stanley, Fractal Concepts in Surface Growth, Cambridge University Press, Cambridge, 1995.
[36] K. Masumnia-Bisheh, K. Forooraghi, M. Ghaffari-Miab, C.M. Furse, IEEE Trans. Antennas Propag. 67 (2019) 7466–7475.
[37] Z. Zubac, D. De Zutter, D. Vande Ginste, IEEE Trans. Antennas Propag. 62 (2014) 4852–4856.
[38] P. Klapetek, I. Ohlídal, Ultramicroscopy 94 (2003) 19–29.
[39] S.O. Rice, Commun. Pure Appl. Math. 4 (1951) 351–378.
[40] D. Franta, I. Ohlídal, J. Mod. Opt. 45 (1998) 903–934.
[41] J. Vohánka, M. Čermák, D. Franta, I. Ohlídal, Phys. Scr. 94 (2019) 045502.
[42] D.A.G. Bruggeman, Ann. Phys. 416 (1935) 636–664.
[43] B. Fodor, P. Kozma, S. Burger, M. Fried, P. Petrik, Thin Solid Films 617 (2016) 20–24.
[44] G. Mie, Ann. Phys. 330 (3) (1908) 377–445.
[45] T. Nousiainen, E. Zubko, H. Lindqvist, M. Kahnert, Tyynelä, J. Quant. Spectrosc. Radiat. Transf. 113 (2012) 2391–2405.