Progress In Electromagnetics Research B, Vol. 64, 171–193, 2015

Speeding Beyond FDTD, Perforated Finite Element Time Domain Method for 3D Electromagnetics

S. M. Raiyan Kabir*, B. M. A. Rahman, and Ken T. V. Grattan

Abstract—A three-dimensional (3D) time domain approach can be particularly valuable for the analysis of many different types of practical structures. In this regard, the finite difference time domain (FDTD) method is a popular technique, being used successfully to analyze the electromagnetic properties of many structures, including a range of optical or photonic devices. This FDTD method offers several major advantages: a minimum level of calculation is required for each of the cells into which the structure is divided, as well as data parallelism and explicit and easy implementation: however, the use of a cuboid grid makes the method very resource intensive for large simulations, especially those in 3D. Although the finite element (FE) approach is superior for the discretization of two-dimensional (2D) and 3D structures, most of the FE-based time domain approaches reported so far suffer from limitations due to the implicit or iterative form or the mass matrix formulation, for example. This paper presents an FE based time domain technique for 3D structures which uses a unique perforated mesh system. It calculated the numerical dispersion characteristics for the FDTD and the proposed method and compared. This paper finally discusses how to utilize the improved numerical dispersion characteristics of the proposed method to increase the simulation speed beyond the FDTD 3D method by using Intel micro-processors.

1. INTRODUCTION

For the better analysis of any 'real world' object, a three-dimensional (3D) analysis is clearly the most appropriate, as is evident in most areas of engineering and science. Applying this approach to studying the electromagnetic (EM) properties of a range of photonic devices, a 3D time domain simulation can be particularly valuable to determine their key characteristics, such as their transient response and the scattering in the device and thus achieve an effective analysis of the overall broadband response of the devices. Although a two-dimensional (2D) approximation can, in some cases, provide satisfactory results, to obtain the total response of a system most accurately, a 3D model is often preferable.

The Finite Difference Time Domain (FDTD) method is a popular technique for analyzing the time domain properties of such electromagnetic devices — this method is well-suited because of the simplicity of the [1–4]. The other advantages of the FDTD method are the explicit and data parallel natures of the governing equations. These three features make it highly suitable for simulation and design when compared to the use of other time domain techniques. In addition, it also uses the minimum computational resources needed to calculate the characteristics of a single computational cell.

However, the FDTD method also has several serious deficiencies related to the cuboid grid used to discretize the computational domain. Any curved or slanted structure will show an effect known as staircasing at the edge of the structure. One way to reduce the improper discretization that results from that is to increase the resolution used. In a 2D system, increasing to the resolution may be a feasible solution but in a 3D system, the memory requirement increases as a cubic function of the

Received 19 August 2015, Accepted 9 November 2015, Scheduled 9 December 2015
* Corresponding author: S. M. Raiyan Kabir (Raiyan.Kabir.1@city.ac.uk).
The authors are with the City University London, EC1V 0HB, London, United Kingdom.

resolution selected. Therefore, increasing the resolution results in a much higher memory requirement and CPU time to carry out these simulations on many occasions these resource requirements become unacceptably high. The FDTD method also requires the entire domain to be discretized with multiple grids of the same resolution. Therefore, on many occasions, the use of a discretization process for the FDTD method becomes very inefficient. In 3D, with the FDTD method and using Yee's lattice [5], this will stagger the 6 field components into 6 different points. Hence, this approach requires 3 different boundaries for the 3 components of the electric field, $\mathbf{E}$ (assuming $\mu_r = 1$ throughout the computational domain) when non-magnetic materials are used.

As an alternative to this approach, the mesh used in a FE approach can allow for a better approximation than does using the FD grid. An exact representation of the interface is also possible as a 3D FE-based mesh uses polyhedral elements to discretize the computational domain. Thus in this way, curved and slanted structures can also be approximated with smooth linear edges. Although, many attempts have been made to develop an FE-based alternative to the FDTD method, but unfortunately, some of the methods reported so far require a large matrix solution [4], or an implicit and iterative solution [6] or indeed the implementation of a higher order mesh [7–9]. As a result, all these methods need a greater level of computational resources and as a consequence these approaches have not been as widely adopted as has the FDTD method. To increase the popularity of any method it is essential to improve the speed of simulation and possible take the speed beyond the speed of simulation of the FDTD method. Unfortunately none of the FE based time domain methods to date managed to achieve speed close to the FDTD method, let alone beating it in speed.

In our previous work, a new FE-based time domain technique using a unique perforated space mesh system has been proposed for solving 2D problems [10]. The method has shown a very low numerical dispersion when the structure was discretized with an equilateral mesh system. The low numerical dispersion thus allows a reduction of the resolution and as a consequence a welcome reduction in the computational resource requirement. In 3D, the resource constraint is the main barrier for solving many time domain problems. Developing a code that can reduce the resource usage can therefore make possible the simulation of many devices, in 3D, which would be a major advantage in the design and analysis of optical and photonic devices.

This paper reports the 3D formulation of the perforated mesh FETD method. It derives the numerical dispersion relation of the proposed 3D method and calculates the "Resolution Reduction Factor (RRF)" (RRF was introduced in [10] for 2D formulation) by comparing the numerical characteristics with that of the FDTD 3D method. The work carried out a theoretical CPU performance analysis using latest Intel based x86 general instruction set and the 'Single Instruction Multiple Data' (SIMD) instruction set on simulation speed of both the FDTD 3D method and the proposed perforated FETD 3D method. It also shows how the use of SIMD instructions set could favour the proposed method over the FDTD 3D method. Finally, it shows how the use of RRF could take the execution speed of the proposed method beyond that of the FDTD 3D method.

2. GOVERNING EQUATIONS

The governing equations arise from Maxwell's equations for the source free and isotropic region and therefore,

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu}\nabla \times \mathbf{E} \tag{1a}$$

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\epsilon}\nabla \times \mathbf{H} \tag{1b}$$

where, $\mathbf{H} = \hat{x}H_x + \hat{y}H_y + \hat{z}H_z$, $\mathbf{E} = \hat{x}E_x + \hat{y}E_y + \hat{z}E_z$, μ and ϵ are the vector magnetic field, the vector electric field, the permeability and the permittivity of the medium respectively with $\hat{x}$, $\hat{y}$ and $\hat{z}$ being the unit vectors in the x, y and z directions, respectively.

The x, y and z directional components from the Eqs. (1) can be separated and 6 governing equations can be obtained as shown below

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y}E_z - \frac{\partial}{\partial z}E_y\right) \tag{2a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x}E_z - \frac{\partial}{\partial z}E_x\right) \tag{2b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x}E_y - \frac{\partial}{\partial y}E_x\right) \tag{2c}$$

$$\frac{dE_x}{dt} = \frac{1}{\epsilon}\left(\frac{\partial}{\partial y}H_z - \frac{\partial}{\partial z}H_y\right) \tag{2d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\epsilon}\left(\frac{\partial}{\partial x}H_z - \frac{\partial}{\partial z}H_x\right) \tag{2e}$$

$$\frac{dE_z}{dt} = \frac{1}{\epsilon}\left(\frac{\partial}{\partial x}H_y - \frac{\partial}{\partial y}H_x\right) \tag{2f}$$

Equations (2) are coupled equations and these six equations can be solved to calculate the time evolution of the EM wave. It can be observed in all the six equations, that the **E** components can always be calculated using only the **H** field components and *vice versa*.

3. SPACE-TIME DISCRETIZATION

To solve Eq. (2) using the finite element technique, the space-time domain has to be discretized by using the finite elements. In Eq. (2), all the time calculations are on the left hand side and the space calculations are on the right hand side. This split in the calculation enables us to avoid the four dimensional form and makes the calculation easier by allowing the discretization of the domains separately using two 3D meshes for the spatial dimensions and two one-dimensional (1D) meshes for the time dimension.

3.1. Space Discretization

To discretize a 3D structure, linear tetrahedrons are chosen. This approach is implemented to reduce both the computational load and the memory requirement of each element to a minimum. The shape function for a tetrahedron is the equation of the surface plane of the tetrahedron, which is given by

$$N_i = a_i x + b_i y + c_i z + d_i \tag{3}$$

where, N is the shape function, i is the index of local node for an element and a, b, c and d are constants.

The variation of the field inside each element can be expressed by,

$$\Phi(x, y, z) = \sum_{i=1}^{M} N_i \phi_i \tag{4}$$

where Φ can be any field component (any one of H_x, H_y, H_z, E_x, E_y and E_z components) inside the element; ϕ_i is the field component at the ith node of the element (any one of h_x, h_y, h_z, e_x, e_y, e_z); M is total number of nodes associated with the element. For a linear element (i.e., four node tetrahedral elements) $M = 4$.

3.2. Time Discretization

In a way similar to what was shown in the literature [10], the time axis can be discretized with 1D finite elements. For a linear element the shape function can be written as shown,

$$Q_j = p_j t + q_j \tag{5}$$

where p_j and q_j are the coefficients of the line passing through the nodes of the time element.

The variation of field between two time nodes can be expressed as,

$$\Psi(t) = \sum_{j=1}^{P} Q_j \psi^{(j)} \tag{6}$$

where Ψ can be any field component (any one of the H_x, H_y, H_z, E_x, E_y, E_z components) inside the element, $\psi^{(j)}$ is the field component at jth time node, P is the number of the time nodes in the time element and for linear elements, $P = 2$.

3.3. Discretized Governing Equations

Applying both the discretizations on Eq. (2), the discretized form of the governing equations may be obtained as follows

$$h_x^{(n+1)} = -\frac{1}{\frac{dQ_2}{dt}}\left[\frac{1}{\mu}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial y}e_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}e_{yi}^{(n)}\right) + \frac{dQ_1}{dt}h_x^{(n-1)}\right] \tag{7a}$$

$$h_y^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}}\left[\frac{1}{\mu}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}e_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}e_{xi}^{(n)}\right) - \frac{dQ_1}{dt}h_y^{(n-1)}\right] \tag{7b}$$

$$h_z^{(n+1)} = -\frac{1}{\frac{dQ_2}{dt}}\left[\frac{1}{\mu}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}e_{yi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial y}e_{xi}^{(n)}\right) + \frac{dQ_1}{dt}h_z^{(n-1)}\right] \tag{7c}$$

$$e_x^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}}\left[\frac{1}{\epsilon}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial y}h_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}h_{yi}^{(n)}\right) - \frac{dQ_1}{dt}e_x^{(n-1)}\right] \tag{7d}$$

$$e_y^{(n+1)} = -\frac{1}{\frac{dQ_2}{dt}}\left[\frac{1}{\epsilon}\left[\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}h_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}h_{xi}^{(n)}\right) + \frac{dQ_1}{dt}e_y^{(n-1)}\right]\right] \tag{7e}$$

$$e_z^{(n+1)} = \frac{1}{\frac{dQ_2}{dt}}\left[\frac{1}{\epsilon}\left[\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}h_{yi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial y}h_{xi}^{(n)}\right) - \frac{dQ_1}{dt}e_{zk}^{(n-1)}\right]\right] \tag{7f}$$

where the field components with the $(n+1)$, (n) and $(n-1)$ superscripts are the future, current and the past values, respectively. It can be noted that each of the equations, when applied on one element, produces only one future value of the field. For this reason, this one value of the field cannot be placed on any of the corner nodes of the element. Therefore, the future field calculated will be stored at the centroid of each element, which is unique. It can be observed from Eq. (7) that the formulation is explicit and data parallel for the calculation of the field components for each time step.

4. THE MESH

As mentioned in the previous section, the space and time dimensions can be represented through two different mesh systems. The space mesh will be a 3D mesh with tetrahedral elements and the time mesh will contain the 1D line elements.

4.1. The Space Mesh

For 3D space discretization, polyhedral elements can be used. Here linear elements take the minimum memory and the minimum computation time and the 4-node tetrahedron is the linear element in a 3D FE discretization. Therefore, a tetrahedron can be chosen as the basic element type for space discretization. A basic cube, given by $ABCGFHDE$ in Fig. 1 can be taken as the initial building block and thus a tetrahedron can be obtained from this cube by drawing a plane through the points A, B and C. The tetrahedron $ABCD$ thus generated is shaded (in red) in the figure while it should be noted that the unshaded part of the cube will be left unused in the calculation.

A mesh can then be generated using the basic tetrahedron by adding further tetrahedra in all directions. Fig. 2(a) shows a $4 \times 4 \times 4$ mesh generated using the basic element presented in Fig. 1. As can be seen, there are hollow spaces inside the mesh, giving an overall perforated mesh discretization, in a way similar to the mesh presented in previous work [10].

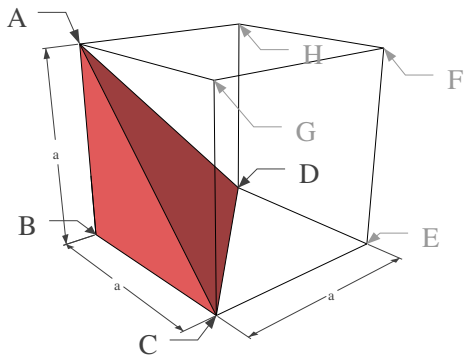


Figure 1. A basic tetrahedral element inside a 3D cubic cell.

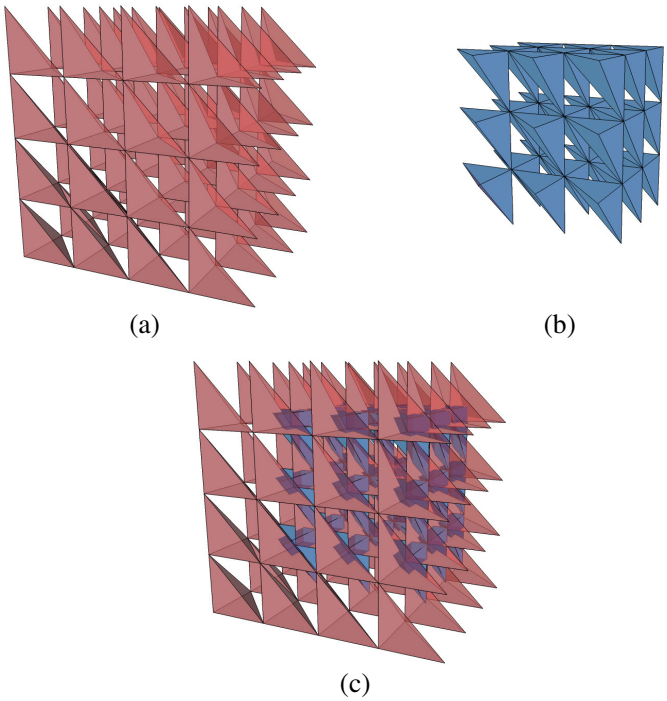


Figure 2. (a) $4 \times 4 \times 4$ main mesh generated by using the basic element of Fig. 1, (b) the $3 \times 3 \times 3$ auxiliary mesh generated using the centroid of the main mesh elements in Fig. 2(a), (c) both meshes together.

At this stage if all the **E** components are placed in the corner nodes of all the elements of the mesh, Eqs. (7a), (7b) and (7c) will produce one value for all the **H** field components at the centroid of each of the elements. To calculate the **E** field components from these **H** field components stored at the centroid, an auxiliary mesh is required which contains only the centroids of the main mesh as the corner nodes and the corner nodes of the main mesh as centroids. Fig. 2(b) shows the auxiliary mesh for the main mesh presented in Fig. 2(a). It can be observed that the mesh in Fig. 2(b) is also a perforated and Fig. 2(c) shows both the meshes together.

It should be noted that, unlike the situation in the FDTD method where the components of both fields are all staggered in space at different points, the meshing system proposed puts all components of the same field at the same node. This implies, if all components of the **E** field are stored in the corner nodes of the main mesh, that all components of the **H** field will be stored at the corner nodes of the auxiliary mesh.

This proposed meshing technique will allow a more accurate representation of structures consisting of non-magnetic materials. The reason for this is that in the EM time domain and non-magnetic materials, the device structure may be generated with the appropriate permittivity and the interface between the materials. As all the **E** field components are at the same point, only one material interface on the mesh exists to represent the physical boundary of the device. By contrast, with the FDTD method all the field components are staggered at different points in space, so for any 3D structure with non-magnetic materials, there must be 3 different interfaces for each physical boundary. As a result, the FDTD always makes a representation of the physical boundaries that is inaccurate.

Although the mesh presented in this section is uniform, a more accurate approximation of the structure can be obtained by moving the nodes on the interfaces and this can be performed if necessary. This technique does not require increased resolution. For greater efficiency and accuracy, an advance meshing scheme can be developed. For the FDTD method, moving a node is more difficult as it is a grid based method. Therefore, a finer grid is needed to increase the accuracy of the representation of the structure, causing much higher memory requirement and increased CPU time.

Another major advantage of the proposed technique is the use of the perforated mesh. To discretize a cube with tetrahedral elements, a minimum of 5 tetrahedra are required. The method discussed considers only one tetrahedron and therefore, compared to a full mesh finite element approach, this method could be up to 5 times faster. This makes the method better suited to 3D calculations when compared to any other finite element time domain method so far reported. As the number of elements that needs to be calculated is very large for 3D structures and for time domain analysis all the elements have to be calculated for each time step, these advantages play a vital role in the adoption of the 3D FETD as a more appropriate approach.

4.2. Time Mesh

In a way similar to that reported previously [10], the time mesh system consists of two coupled 1D meshes which allows the calculation of the **E** and the **H** fields at different points in time.

As shown in Fig. 3, if all the **E** field components are stored at time 0 and the **H** field components at time $t/2$ then according to Eqs. (7d), (7e) and (7f) the next value for the **E** field components can be calculated at time t. Here, time 0, $t/2$, and t are considered as the past, present and future times.

In the same way, to calculate the next values of the **H** field components, times $t/2$, t and $3t/2$ are taken as the past, present and future times, respectively. Eqs. (7a), (7b) and (7c) are used to calculate the next **H** field components at time $3t/2$. In this way, the time propagation can be calculated by using the staggered 1D time meshes **M** and **N**, as shown in Fig. 3.

Like all other explicit time domain methods, proposed method is also subject to a time stability restriction similar to Courant-Friedrichs-Lewy (CFL) condition [11]. As mentioned in [7], the method when used with an element which resembles a Finite difference rectangular are subject to the same limit. The phenomenon can also be observed by experimentation using the proposed method. A more generic form of stability condition for explicit FE method solving Maxwell's equations directly has been presented in [12].

5. PERFECTLY MATCHED LAYER

To perform a simulation within a finite computational domain, fields approaching the boundary should be handled. If they are allowed to reach the computational domain boundaries they will reflect back

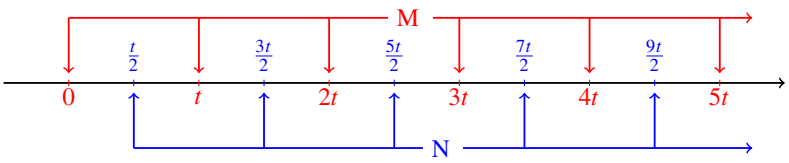


Figure 3. Arrangement of time mesh system for equal time spacing.

into the domain and destroy the solution. This reflection is non-physical and due to the truncation only. To prevent the wave to reach the boundaries the wave should be absorbed at the boundary without reflection. One of the best ways to absorb the wave and prevent reflection for a EM time domain method is to use Perfectly Matched Layers (PML) [27, 28]. To implement the PML boundaries for the proposed method the we followed the Auxiliary Differential Equation (ADE) approach. For this paper an un-split formulation is used.

For 3D there are 7 different types of PML as follows,

(i) X PML
(ii) Y PML
(iii) Z PML
(iv) XY PML
(v) YZ PML
(vi) ZX PML
(vii) XYZ PML

The names of the PMLs are given based on the field they absorb.

To implement the 7 PMLs modified vector differential operators are taken respectively as follows,

$$\tilde{\nabla}_x = \hat{x}\frac{\partial}{\partial x}\left(1+j\frac{\sigma_x}{\omega}\right)^{-1} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z} \tag{8a}$$

$$\tilde{\nabla}_y = \hat{x}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y}\left(1+j\frac{\sigma_y}{\omega}\right)^{-1} + \hat{z}\frac{\partial}{\partial z} \tag{8b}$$

$$\tilde{\nabla}_z = \hat{x}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z}\left(1+j\frac{\sigma_z}{\omega}\right)^{-1} \tag{8c}$$

$$\tilde{\nabla}_{xy} = \hat{x}\frac{\partial}{\partial x}\left(1+j\frac{\sigma_x}{\omega}\right)^{-1} + \hat{y}\frac{\partial}{\partial y}\left(1+j\frac{\sigma_y}{\omega}\right)^{-1} + \hat{z}\frac{\partial}{\partial z} \tag{8d}$$

$$\tilde{\nabla}_{yz} = \hat{x}\frac{\partial}{\partial x} + \hat{y}\frac{\partial}{\partial y}\left(1+j\frac{\sigma_y}{\omega}\right)^{-1} + \hat{z}\frac{\partial}{\partial z}\left(1+j\frac{\sigma_z}{\omega}\right)^{-1} \tag{8e}$$

$$\tilde{\nabla}_{zx} = \hat{x}\frac{\partial}{\partial x}\left(1+j\frac{\sigma_x}{\omega}\right)^{-1} + \hat{y}\frac{\partial}{\partial y} + \hat{z}\frac{\partial}{\partial z}\left(1+j\frac{\sigma_z}{\omega}\right)^{-1} \tag{8f}$$

$$\tilde{\nabla}_{xyz} = \hat{x}\frac{\partial}{\partial x}\left(1+j\frac{\sigma_x}{\omega}\right)^{-1} + \hat{y}\frac{\partial}{\partial y}\left(1+j\frac{\sigma_y}{\omega}\right)^{-1} + \hat{z}\frac{\partial}{\partial z}\left(1+j\frac{\sigma_z}{\omega}\right)^{-1} \tag{8g}$$

Here, σ_x, σ_y and σ_z are the coefficient of absorption in the x, y and z directions, respectively, and ω is the angular frequency.

The governing equations for the X PML can be derived by applying Eq. (8a) on Eq. (2) and are as follows,

$$\frac{dH_x}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial y}E_z - \frac{\partial}{\partial z}E_y\right) \tag{9a}$$

$$\frac{dH_y}{dt} = \frac{1}{\mu}\left(\frac{\partial}{\partial x}E_z - \frac{\partial}{\partial z}E_x - \mathcal{E}_{x[z]}\right) - \sigma_x H_y \tag{9b}$$

$$\frac{dH_z}{dt} = -\frac{1}{\mu}\left(\frac{\partial}{\partial x}E_y - \frac{\partial}{\partial y}E_x - \mathcal{E}_{x[y]}\right) - \sigma_x H_z \tag{9c}$$

$$\frac{dE_x}{dt} = \frac{1}{\epsilon}\left(\frac{\partial}{\partial y}H_z - \frac{\partial}{\partial z}H_y\right) \tag{9d}$$

$$\frac{dE_y}{dt} = -\frac{1}{\epsilon}\left(\frac{\partial}{\partial x}H_z - \frac{\partial}{\partial z}H_x - \mathcal{H}_{x[z]}\right) - \sigma_x E_y \tag{9e}$$

$$\frac{dE_z}{dt} = \frac{1}{\epsilon}\left(\frac{\partial}{\partial x}H_y - \frac{\partial}{\partial y}H_x - \mathcal{H}_{x[y]}\right) - \sigma_x E_z \tag{9f}$$

$$\frac{d\mathcal{E}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z} E_x \tag{9g}$$

$$\frac{d\mathcal{E}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y} E_x \tag{9h}$$

$$\frac{d\mathcal{H}_{x[z]}}{dt} = \sigma_x \frac{\partial}{\partial z} H_x \tag{9i}$$

$$\frac{d\mathcal{H}_{x[y]}}{dt} = \sigma_x \frac{\partial}{\partial y} H_x \tag{9j}$$

Governing equations for other types of PML can be derived by applying their respective partial differential operator on Eq. (2) in the similar manner.

6. SIMULATION RESULTS

C++ language and OpenMP technology were used to develop a shared memory parallel code. Perfectly Matched Layers (PMLs) were also implemented using the auxiliary differential equation (ADE) approach in the code to support the simulation for longer times in a truncated computational domain. In this section, results from several simulations performed by using the code are presented and results discussed. to maintain the stability condition discussed in Section 4.2, the time step size for all the simulations are maintained at $\Delta/2c$. Here, Δ is the size of the element and c is normalised speed of light.

6.1. Free Space Propagation

The simplest possible simulation which can be performed using the code is the free space propagation from a point source. To do so, an H_z point source was placed at the center of cubic computational domain and the wavelength chosen was 1 µm. The PML was placed around the computational domain to truncate and absorb all the reflections from the boundary of the computational domain. To illustrate the output of the simulation, three slices were taken from the volume after 1000 time steps where the center of each slice was placed at the center of the computational domain. The 'slicing planes' used were parallel to the xy, yz and zx planes of the computational domain and Fig. 4 shows the field distribution on these planes.

As can be seen, all the fields radiating from the center of the computational domain have been absorbed at the outer boundary and no sign of reflection is visible inside the domain. Fig. 4(b) shows the distribution of the H_z field and although the field is inserted directly by the source at the center, the field propagates uniformly in xy-plane. The field intensity decreases gradually when the propagation direction changes from the xy-plane towards the z axis and there is no propagation of the H_z field in the z direction.

The propagations of the H_x and the H_y fields are mostly confined in the zx and the yz-planes, respectively. However, the E_x and the E_y fields do not propagate in the zx and the yz, planes respectively. For this setup with the H_z source, the E_z field does not develop and Fig. 4(g) shows no presence of a E_z field in that domain.

Equation (2c) calculates the H_z field from variation in y direction of E_x field and variation in x of E_y field components respectively. In Figs. 4(e) and (f), E_x and E_y fields components are visible. Therefore, the calculated H_z field is dominant in both x and y direction and and is not present in z exact z direction. The H_z slowly reduces when the direction of propagation approaches z direction. This explanation is consistent for all 6 field components.

6.2. Nanowires

Next, to illustrate the guiding of an EM wave in a waveguide, a silicon nanoguide is simulated and the cross section of the nanowire can be seen in Fig. 5(a). The Si core of the waveguide was fabricated on a SiO_2 buffer layer where the waveguide considered here is 240 nm in height and 500 nm in width. An H_y continuous wave point source at a wavelength of 1.55 µm wavelength was placed at the center of the guide cross section at $z = 1.1$ µm (z-direction). The 1.55 µm has been chosen as it is the communication

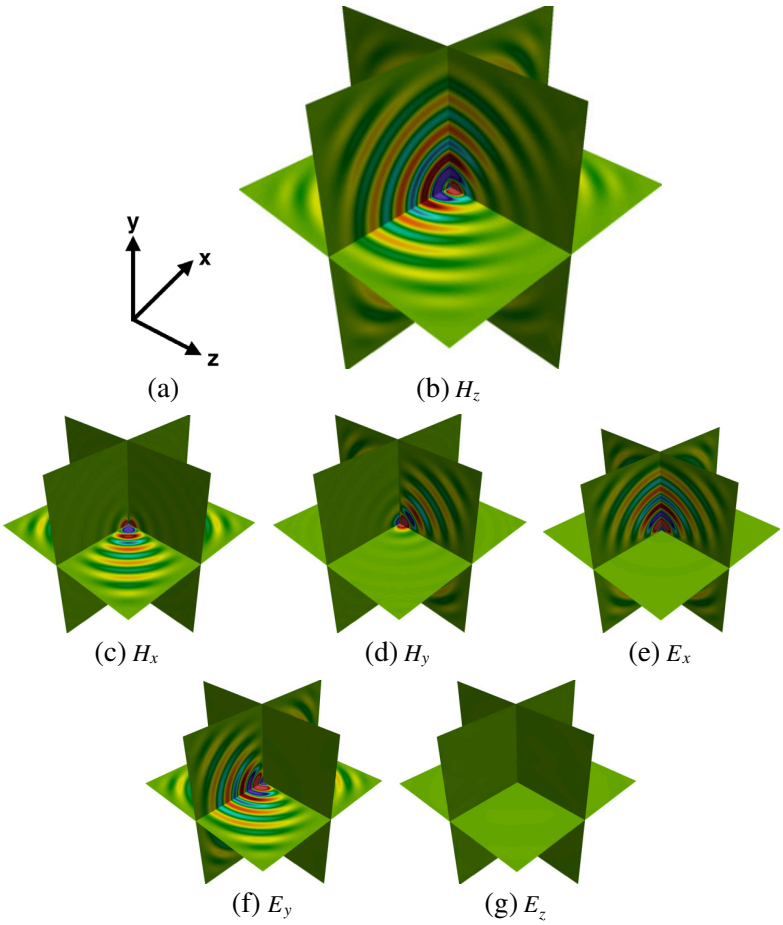


Figure 4. (a) Axes of all plots are shown in this figure, (b) H_z, (c) H_x, (d) H_y, (e) E_x, (f) E_y and (g) E_z field distribution in space after 1000 time steps for a H_z point source at the middle of the computational domain.

frequency used in most of todays optical communication devices. The resolution for the simulation was 50 nodes/ µm where the time-step size was taken as $\Delta/2c$ and Δ is the size of a cell in µm with c being the speed of light in µm/s.

As the radiation from the point source propagates along the guide, the mode profile of the guide then develops. Fig. 5(b) shows the propagation of the H_y field injected inside the waveguide after 5000 time steps. Figs. 5(c) and 5(d) show the mode profiles for the H_y and H_x fields respectively. These mode profiles were captured at a distance of 3 µm from the source on a plane parallel to xy-plane. It is encouraging that this result agrees well with the mode profile of the same silicon nanowire, obtained by using the full-vectorial finite element method (VFEM) [13]. The dimensions of the SiO_2 substrate was chosen in such a way that, when using VFEM the propagating mode profile do not interact with the computational boundaries. This is not a concern for the proposed method. As the proposed method uses PML. The effective index of the guide was calculated using Eq. (10) [14], as shown below

$$n_{eff} = \frac{1}{k_0 d}\left(\phi|_b - \phi|_a\right) \tag{10}$$

where, k_0 is the wave vector for free space propagation in a vacuum; $\phi|_a$ and $\phi|_b$ are the phases of the wave at two different observation points a and b at the same time and d is the distance between the two points. It should be noted that the Eq. (10) calculates the effective index from the propagating wave inside the guide. Therefore, contains the total effect of the refractive indices of Si, Silica and Air for the propagation mode.

To calculate the effective index, the field distribution along the central axis of the nanowire was determined after 5000 time-steps. The phases were recorded at distances of 3 μm and 4 μm away from the source on the central axis where the effective index of the structure was calculated to be 2.4721.

To benchmark this result obtained for the effective index, the same nanowire design was simulated using the VFEM [15, 16]. To compare the results, the cross sectional resolution of was kept exactly the same as the earlier simulation with the proposed FETD method. Similar H_x and H_y field patterns were obtained for the fundamental mode from the VFEM and as a result, the effective index was calculated to be equal to 2.4751. It is pleasing that the results obtained are very close, considering that the method described in this paper uses discrete time and propagation axes, whereas for the VFEM method, the propagation axis and time were considered to be continuous.

6.3. Nano Power Splitter

To evaluate further the code developed in this work, a structure which is non-uniform along the axial direction was considered and its characteristics simulated. For this purpose, a compact MMI-based power splitter was considered [17] where the core of the power splitter was considered to be fabricated from Si and the buffer layer was SiO_2. The structure consists of three parts, as discussed below

(i) The input guide

(ii) The splitter section

(iii) The two output guides

The height of the Si structure was taken as 200 nm in all the sections. The widths of the input and output guides were taken to be 400 nm. The multimoded splitter section was considered to be 6 μm in width and 30 μm in length. In the simulation an H_y field point source at a 1.55 μm wavelength was

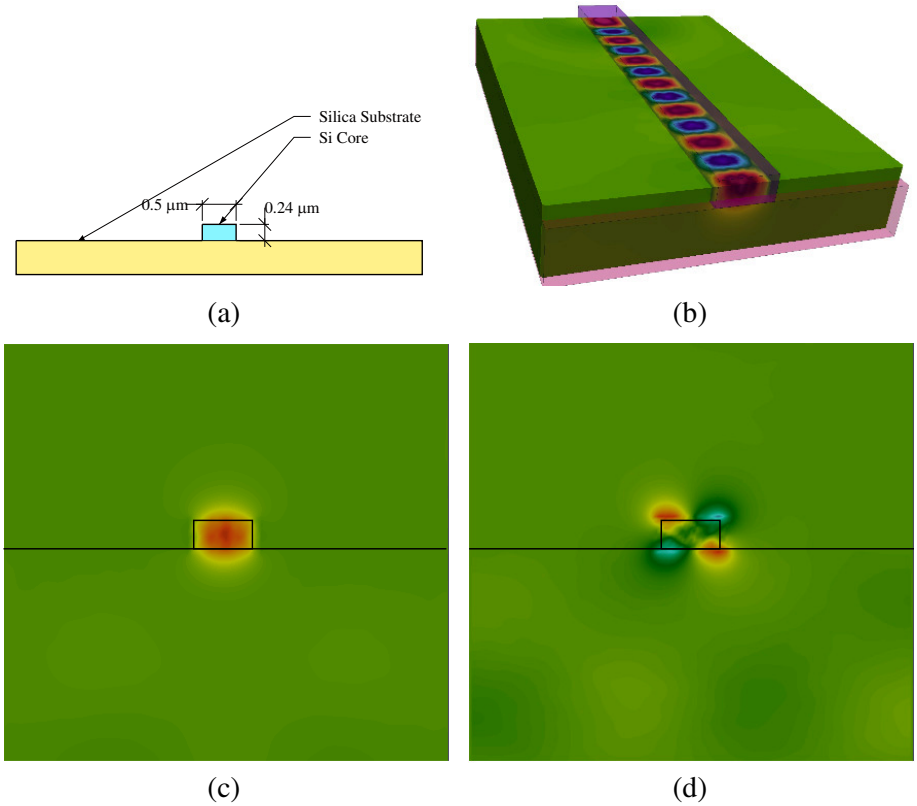


Figure 5. (a) Cross section of the nanowire, (b) propagation of H_y field inside the guide, (c) mode profile of the dominant H_y field, (d) mode profile of the non-dominant H_x field.

placed at the center of the input guide where the length of the input guide was chosen to be 5 µm to form a mode before entering the splitter section. These output waveguides were placed 1.5 µm away from the center of the splitter and the resolution taken for the simulation was 20 nodes/ µm, with the time-step size being $\Delta/2.5c$. Fig. 6(a) (page 11) shows a 3D schematic diagram of the structure where Fig. 6(b), shows the H_y field distribution after 9000 time-steps. It can clearly be seen from Fig. 6(c) that the output waves are both equal in amplitude and phase. Therefore, the structure divides the input into two coherent and equal amplitude parts at the output waveguides and the mode profiles of the H_y field both at the input and output of the nanowires are shown in Fig. 7(a) (page 12) and 7(b), respectively. The field plot of Fig. 7(a) was taken at 1.5 µm away from the source on the input nanowire.

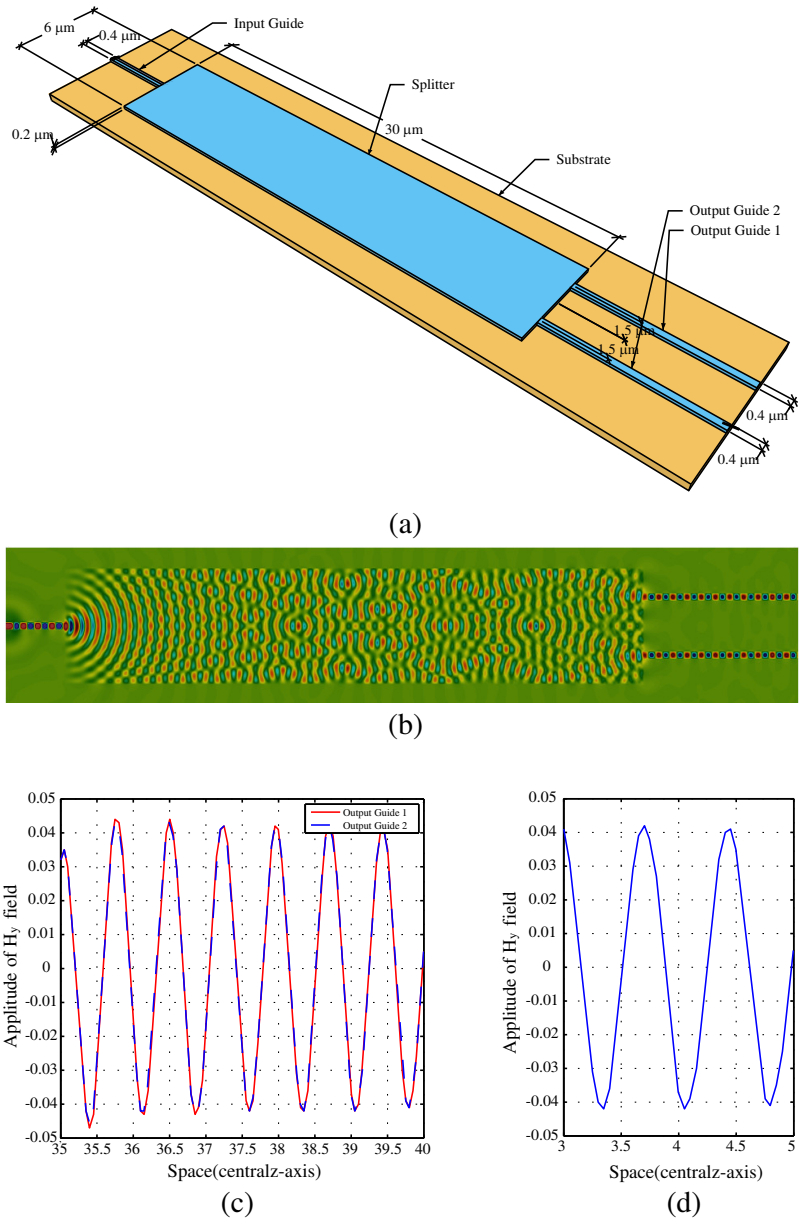
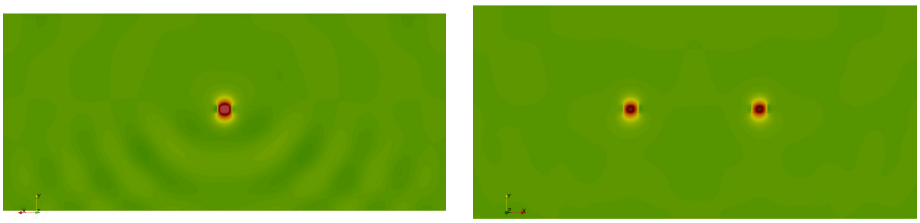


Figure 6. (a) Schematic 3D diagram of the nano power splitter (sky blue part is the Si core and orange coloured part is SiO_2 substrate), (b) H_y field profile at the central zx-plane of the power splitter after 9000 time-steps, (c) comparison of the H_y field of the output waves in output guide 1 and 2 at the central z-axis after 9000 time-steps, (d) the input H_y field in the input guide.



(a) (b)

Figure 7. (a) H_y field profile propagating in the nano input wire, (b) H_y field profile after splitting the input power into two at the output nanowires.

The Fig. 7(b) was captured at 5 µm away from the splitter section on the output guides.

7. NUMERICAL DISPERSION

For the numerical simulation, the computational domain has to be discretized. However, due to the discretisation, phase error can be introduced into the propagating plane wave. As a result, the speed of propagation may be slower than the actual speed of the wave. The phase lag with the actual wave increases with the length of its propagation in the computational domain. The issue with the phase error gets worse when the error varies with the direction of propagation. The result of this is different speeds of propagation in different directions, which is equivalent to an artificial anisotropy imposed on the wave by the discretisation even when the material is actually isotropic. This phenomenon is known as **"Numerical Dispersion"** [4] or **"Numeric Anisotropy"** [18]. This dispersion can be minimised by increasing the resolution of the discretisation [1], but to do this would require more memory and more computations.

Technique to calculate the numerical dispersion for the 2D perforated method mesh system has been described in [10]. For 3D, similar steps could be followed to obtain the numerical dispersion relation from Eq. (7).

7.1. Formulation of Numerical Dispersion Relation

A monochromatic source was assumed for the propagation where **E** and **H** field components can be expressed as

$$e^{(n)}_{x\langle k\rangle} = E_{x0}e^{j(\omega t^{(n)}-\tilde{\kappa_x}x_k-\tilde{\kappa_y}y_k-\tilde{\kappa_z}z_k)} \tag{11a}$$

$$e^{(n)}_{y\langle k\rangle} = E_{y0}e^{j(\omega t^{(n)}-\tilde{\kappa_x}x_k-\tilde{\kappa_y}y_k-\tilde{\kappa_z}z_k)} \tag{11b}$$

$$e^{(n)}_{z\langle k\rangle} = E_{z0}e^{j(\omega t^{(n)}-\tilde{\kappa_x}x_k-\tilde{\kappa_y}y_k-\tilde{\kappa_z}z_k)} \tag{11c}$$

$$h^{(m)}_{x\langle l\rangle} = H_{x0}e^{j(\omega t^{(m)}-\tilde{\kappa_x}x_l-\tilde{\kappa_y}y_l-\tilde{\kappa_z}z_l)} \tag{11d}$$

$$h^{(m)}_{y\langle l\rangle} = H_{y0}e^{j(\omega t^{(m)}-\tilde{\kappa_x}x_l-\tilde{\kappa_y}y_l-\tilde{\kappa_z}z_l)} \tag{11e}$$

$$h^{(m)}_{z\langle l\rangle} = H_{z0}e^{j(\omega t^{(m)}-\tilde{\kappa_x}x_l-\tilde{\kappa_y}y_l-\tilde{\kappa_z}z_l)} \tag{11f}$$

where, $\underline{\tilde{\kappa}} = \hat{x}\tilde{\kappa}_x + \hat{y}\tilde{\kappa}_y + \hat{z}\tilde{\kappa}_z$ is the numerical wave vector; ω is the frequency of the source; E_{x0}, E_{y0}, E_{z0}, H_{x0}, H_{y0} and H_{z0} are the amplitudes of the E_x, E_y, E_z, H_x, H_y and H_z field components, respectively.

Applying the Eq. (11) to the 3D governing equations Eq. (7) and eliminating all amplitudes in a similar way followed in [10], produces the numerical dispersion relation for the 3D FETD with perforated

mesh system as,

$$\left\{e^{j\omega\Delta t^{(n)}}\right\}\cdot\left\{\frac{dQ^{(n)}}{dt}\right\}\cdot\left\{e^{j\omega\Delta t^{(m)}}\right\}\cdot\left\{\frac{dQ^{(m)}}{dt}\right\}=v_p^2\cdot\left[\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial x}\right\}\cdot\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial x}\right\}+\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial y}\right\}\cdot\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial y}\right\}+\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_k}\right\}\left\{\frac{\partial N_k}{\partial z}\right\}\cdot\left\{e^{-j\tilde{\kappa}\cdot\underline{\Theta}\cdot\Delta\mathbf{r}_l}\right\}\left\{\frac{\partial N_l}{\partial z}\right\}\right] \quad (12)$$

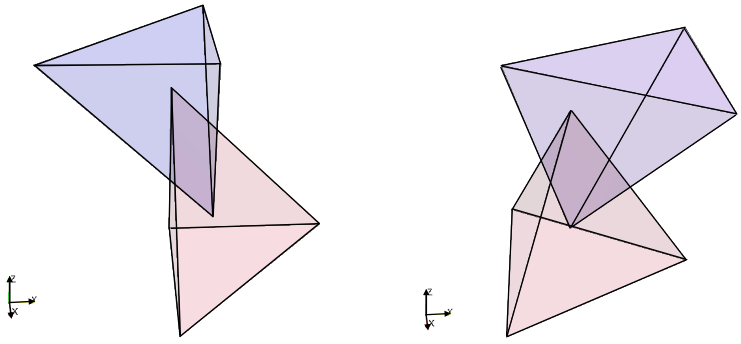
Here, $v_p=\frac{1}{\sqrt{\mu\epsilon}}$, $\Delta x_{k(i)}=x_{k(i)}-x_{\langle l\rangle}$, $\Delta y_{k(i)}=y_{k(i)}-y_{\langle l\rangle}$, $\Delta z_{k(i)}=z_{k(i)}-z_{\langle l\rangle}$, $\Delta\mathbf{r}_k=(\Delta x_k,\Delta y_k,\Delta z_k)$, $\Delta x_{l(i)}=x_{l(i)}-x_{\langle k\rangle}$, $\Delta y_{l(i)}=y_{l(i)}-y_{\langle k\rangle}$, $\Delta z_{l(i)}=z_{l(i)}-z_{\langle k\rangle}$, $\Delta\mathbf{r}_l=(\Delta x_l,\Delta y_l,\Delta z_l)$, $\Delta t_\tau^{(m)}=t_\tau^{(m)}-t^{[n]}$, $\Delta t_\tau^{(n)}=t_\tau^{(n)}-t^{[m]}$, $\underline{\tilde{\kappa}}=(\tilde{\kappa}_x,\tilde{\kappa}_y,\tilde{\kappa}_z)$, $\underline{\tilde{\kappa}}=(\tilde{\kappa}_x,\,\tilde{\kappa}_y,\,\tilde{\kappa}_z)=(\tilde{\kappa}\cos\theta\sin\phi,\,\tilde{\kappa}\sin\theta\sin\phi,\,\tilde{\kappa}\cos\phi)=\tilde{\kappa}\cdot(\cos\theta\cdot\sin\phi,\,\sin\theta\cdot\sin\phi,\,\cos\phi)=\tilde{\kappa}\cdot\underline{\Theta}$ and $\underline{\Theta}=(\cos\theta\cdot\sin\phi,\,\sin\theta\cdot\sin\phi,\,\cos\phi)$. N and Q are the space and time shape functions respectively.

Equation (12) can be used with Newton's iterative method to obtain the numerical wave vector $\tilde{\kappa}$. The normalised propagation velocity, $v_p/c=2\pi/\tilde{\kappa}_{final}$ can be calculated using the final converged value for specific angles of propagation θ and ϕ.

The numerical dispersion calculation using Eq. (12) has been used to calculate the resolution reduction factor in Section 8.

7.2. Numerical Dispersion Results

The **"Isosceles Right-Angled Tetrahedral" (IRT3D)** mesh (Fig. 8(a)) and **"Equilateral Tetrahedral (ET3D)"** mesh (Fig. 8(b)) was considered to find out the numerical dispersions for them.



(a) (b)

Figure 8. Two coupled element of the IRT3D and ET3D mesh systems, respectively. The element from the main mesh is shown in red and the element from the auxiliary mesh is shown in blue colours, respectively. (a) Two elements of the coupled the IRT3D mesh, (b) two elements of the coupled the ET3D mesh.

The elements of IRT3D mesh has tetrahedrons where 2 for the 3 adjacent planes are in right angle with the base plane. On the other hand all the edges of the ET3D mesh are equal in length. As mentioned in Section 4.1 for both type of meshes nodes of the auxiliary mesh (blue elements in Fig. 8) are the centroid of the main mesh elements (red elements in Fig. 8).

A computer program was written in Python language to find the numerical dispersion of both the meshes by solving Eq. (12) for a specific range of angles θ and ϕ. To obtain the results applicable for all wavelengths the resolution of these calculations were normalized by wavelength. From this point all resolutions in this paper are considered as samples per wavelength (number of samples/λ).

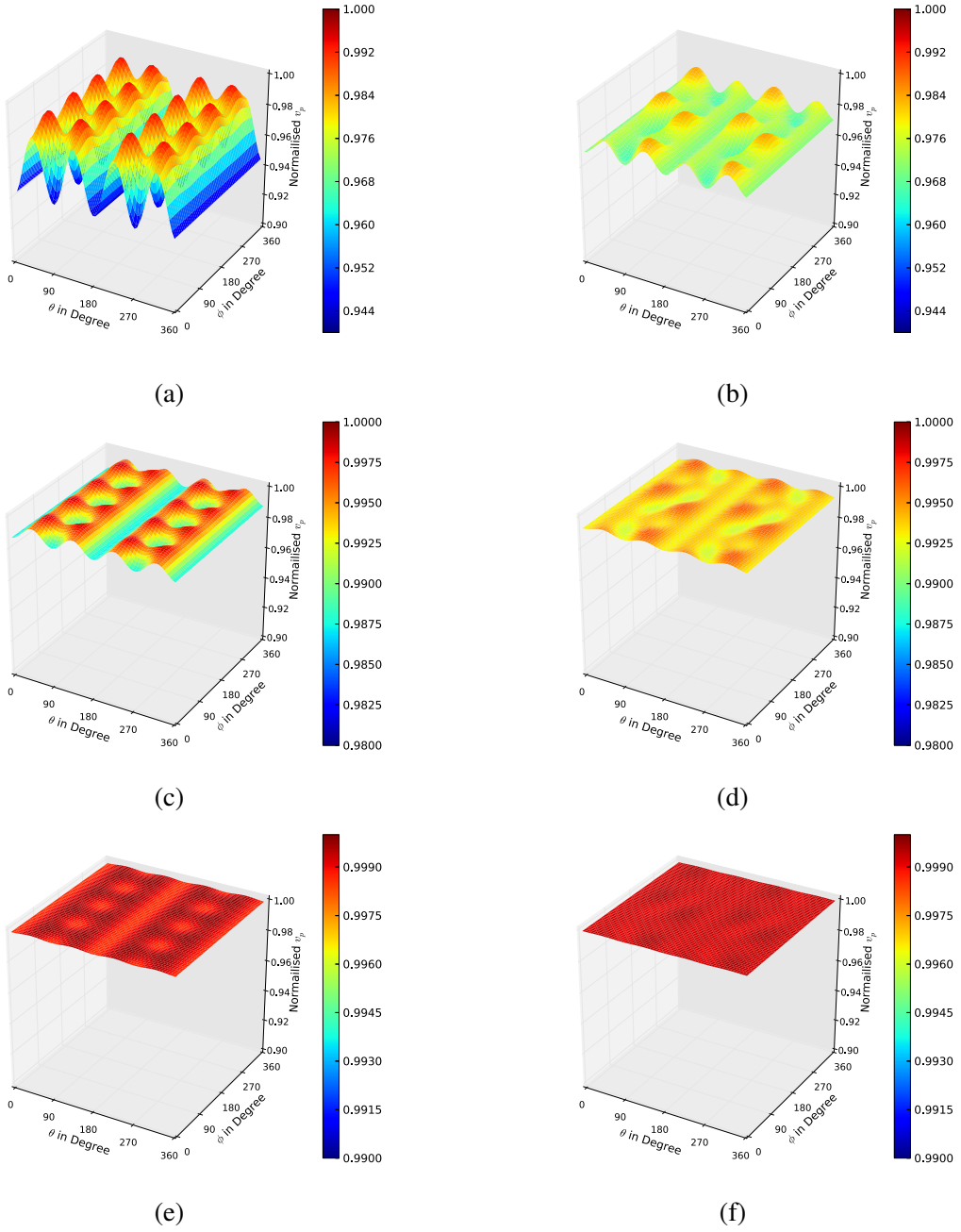


Figure 9. Side by side comparison of normalised v_p of the IRT3D and the ET3D meshes for resolution $5/\lambda$, $10/\lambda$ and $30/\lambda$, respectively. (a) Normalised $v_p(\theta, \phi)$ distribution for $5/\lambda$ resolution for the IRT3D mesh, (b) Normalised $v_p(\theta, \phi)$ distribution for $5/\lambda$ resolution for the ET3D mesh, (c) normalised $v_p(\theta, \phi)$ distribution for $10/\lambda$ resolution for the IRT3D mesh, (d) normalised $v_p(\theta, \phi)$ distribution for $10/\lambda$ resolution for the ET3D mesh, (e) normalised $v_p(\theta, \phi)$ distribution for $30/\lambda$ resolution for the IRT3D mesh, (f) normalised $v_p(\theta, \phi)$ distribution for $30/\lambda$ resolution for the ET3D mesh.

Figure 9 presents a side by side comparison of normalised phase velocity v_p distribution for the IRT3D and the ET3D meshes for $5/\lambda$, $10/\lambda$ and $30/\lambda$, respectively. It can be noted that the Figs. 9(a), 9(c) and 9(e) are showing the normalised $v_p(\theta, \phi)$ distributions of the IRT3D mesh for the resolutions $5/\lambda$, $10/\lambda$ and $30/\lambda$, respectively. Figs. 9(b), 9(d) and 9(f) are showing the normalised $v_p(\theta, \phi)$ distributions of the ET3D mesh for the resolutions $5/\lambda$, $10/\lambda$ and $30/\lambda$, respectively.

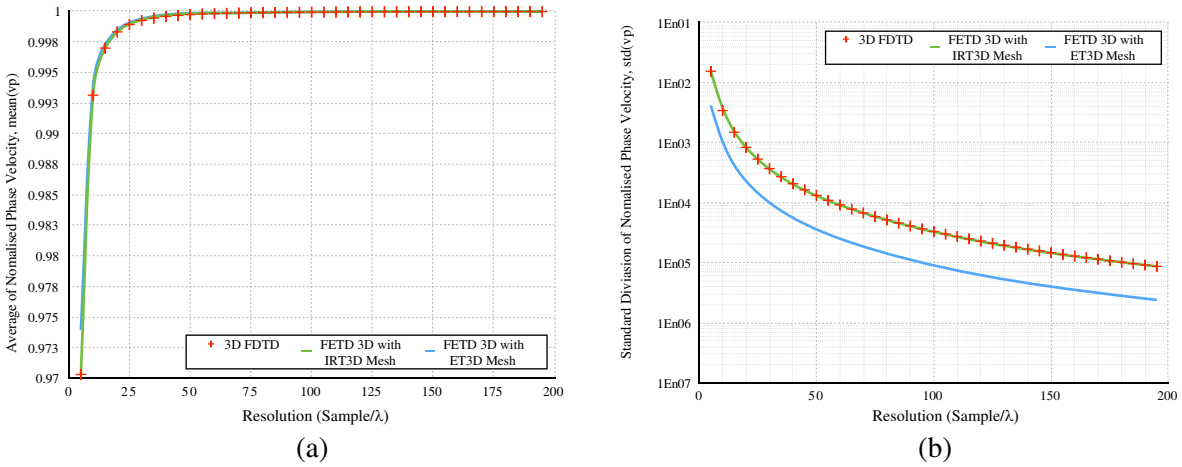


Figure 10. Mean and Standard Deviation of the 3D FDTD method and the proposed 3D FETD with the IRT3D and the ET3D meshes for $5/\lambda$ to $200/\lambda$ resolutions. (a) Comparison of the mean of normalised v_p for 3D FDTD method and proposed FETD3D method with IRT3D and ET3D mesh, (b) comparison of the standard deviation of normalised v_p for 3D FDTD method and proposed FETD3D method with IRT3D and ET3D meshes.

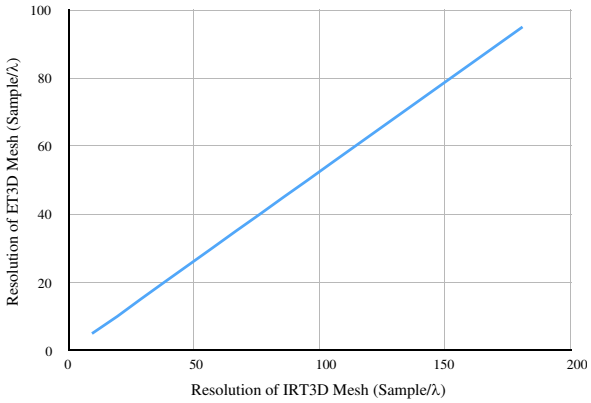


Figure 11. Resolution Relation of the ET3D and the IRT3D meshes for resolution $5/\lambda$ to $500/\lambda$.

It can also be seen from figure pairs (9(a), 9(b)), (9(c), 9(d)) and (9(e), 9(f)) that each of the pairs are plotted with the same scale in amplitude. Which also kept the colour scheme of the surface plot to be the same. All the figure pairs presented in Fig. 9 shows that for all resolution the variation of normalised v_p in ET3D mesh is lower than the IRT3D mesh. Hence, the numerical dispersion of the ET3D mesh is lower that the IRT3D mesh for all resolution. This result is consistent with the finding of [10] for the 2D formulation.

7.3. Comparison with the FDTD3D Method

To compare the numerical dispersion of the FDTD3D method with the proposed method another Python program was written to solve the numerical dispersion relation of the FDTD3D method (presented in [1]). Calculations were performed to obtain the normalized speed propagation of the FDTD3D method for all the resolution presented in Fig. 9.

Figure 10 shows the comparison among the mean and standard deviation of the normalized speed of propagation with FDTD3D, proposed FETD3D with IRT3D mesh and ET3D mesh respectively. It should be noted that all numerical dispersion results for the IRT3D mesh are exact match with numerical dispersion results of the FDTD3D method for the all resolutions. This is also consistent with the finding

of [10] for the 2D formulation. When the $v_p(\theta, \phi)$ distribution surfaces for the FDTD3D were plotted for $5/\lambda$, $10/\lambda$ and $30/\lambda$ resolutions, they produced exactly the same plots as shown in Figs. 9(a), 9(c) and 9(e). From the above results, it is clear that the numerical dispersion property of the FDTD3D method and the proposed method with IRT3D mesh are identical. From this point in this paper they are considered to be synonyms.

However, the ET3D mesh follows a significantly lower standard deviation curve compared to the IRT3D mesh and the FDTD3D. Therefore, for all resolutions the numerical dispersion for the ET3D mesh with the proposed FETD3D is significantly lower than that of the FDTD3D and IRT3D mesh with the proposed method. The mean for all three cases are comparable.

8. RESOLUTION REDUCTION FACTOR CALCULATION

To calculate the resolution reduction factor for IRT3D and ET3D mesh, first the functional relation between the resolutions of IRT3D and ET3D mesh should be evaluated. To find the relation of the resolutions of the IRT3D mesh and the ET3D mesh, the resolutions of each mesh time can be described as two sets $\mathbf{RES_{IRT3D}}$ and $\mathbf{RES_{ET3D}}$ respectively.

$$\mathbf{RES_{ET3D}} = \{i | i \in \mathbb{N}\} \tag{13a}$$

$$\mathbf{RES_{IRT3D}} = \{i | i \in \mathbb{N}\} \tag{13b}$$

Here, i is the number of points per wavelength on the mesh (i/λ).

Two mapping sets for the ET3D and the IRT3D mesh can be taken as $\mathbf{STD_{V_{ET3D}}}$ and $\mathbf{STD_{V_{IRT3D}}}$ in Eqs. (14a) and (14b), respectively as

$$\mathbf{STD_{V_{ET3D}}} = \{s | s = \mathcal{S}_{V_{ET3D}}(i),\ i \in \mathbf{RES_{ET3D}},\ s \in \mathbb{R}\} \tag{14a}$$

$$\forall i,\ \mathcal{S}_{V_{ET3D}} : \mathbf{RES_{ET3D}} \mapsto \mathbf{STD_{V_{ET3D}}}$$

$$\mathbf{STD_{V_{IRT3D}}} = \{s | s = \mathcal{S}_{V_{IRT3D}}(i),\ i \in \mathbf{RES_{IRT3D}},\ s \in \mathbb{R}\} \tag{14b}$$

$$\forall i,\ \mathcal{S}_{V_{IRT3D}} : \mathbf{RES_{IRT3D}} \mapsto \mathbf{STD_{V_{IRT3D}}}$$

$\mathbf{STD_{V_{ET3D}}}$ and $\mathbf{STD_{V_{IRT3D}}}$ are respectively the sets of standard deviations of the standard deviations calculated using Eq. (12) for different resolutions for ET3D and IRT3D mesh respectively.

The **"Resolution Relation ($\mathcal{R}$)"** is considered as the mapping from $\mathbf{RES_{IRT3D}}$ to $\mathbf{RES_{ET3D}}$ when $\mathcal{S}_{V_{ET3D}}(i)$ and $\mathcal{S}_{V_{IRT3D}}(j)$ are almost equal and $i \in \mathbf{RES_{IRT3D}}$ and $j \in \mathbf{RES_{ET3D}}$. Fig. 11 shows the $\mathcal{R}$. As can be seen, the relation is a linear function.

$$\forall i \in \mathbf{RES_{IRT3D}},\ \mathcal{R} : i \mapsto j | \mathcal{S}_{V_{IRT3D}}(i) \cong \mathcal{S}_{V_{ET3D}}(j), \tag{15}$$

$$i \in \mathbf{RES_{IRT3D}} \text{ and } j \in \mathbf{RES_{ET3D}}$$

Unlike the resolution relation of 2D formulation in [10] (where the relation was a second order parabolic) the relation of the 3D formulation is linear in nature. The difference might be because the number of points used in the 3D for 1 cell did not increase proportionally compared to 2D.

Therefore, the **"Resolution Reduction Factor (RRF)"** is a constant for all resolutions when IRT3D and ET3D meshes are considered. From the line in Fig. 11 the **RRF** was calculated using slop inverse to be **1.903** for all the resolutions.

9. THEORETICAL CPU PERFORMANCE

For any numerical method speed of execution is a crucial matter. The speed of design, development and analysis using the method depends on the speed on execution. For a time domain analysis tool like the FDTD [5] and the proposed FETD method, the CPU performance is one of the most important factor that directly relates to its commercial success.

This section theoretically compares the CPU performance of the proposed method with the FDTD3D method by deriving formulations with minimum number of CPU operations. Then calculating the minimum number of CPU cycles needed to obtain any field component for the next time step in one 'discretization element' (for FDTD one single Yee's lattice and for the proposed FETD method a

single tetrahedron). The CPU cycles needed to for each computational operations were taken from the instruction sets of Intel's 64 bit x86 based Haswell Chips [19].

This analysis ignores memory latency and all other matters which may be subject to the implementation, compiler etc.. It also ignores multi-threading although both the methods are data parallel and explicit in formulation. The analysis only focus on operations related to execution of the governing equations only. This is to exclude any influence of other external matters like software design, compiler, operating system etc..

This section also utilised the Single Instruction Multiple Data (SIMD) instructions provided in the Intel 64-bit x86 Haswell microprocessors. SMID instructions are special instructions developed by CPU Intel which performs the same operation on multiple sets of data with the same latency of a general purpose instruction (applies it on only one set of data). For example, general purpose ADD operation could add a pair of integers in 3 CPU cycles. On the other hand the SIMD ADD operation could perform the same operation on 8 pairs of integers with the same 3 CPU cycles on a Haswell chip. This is advantageous for a computationally heavy method like the proposed method as the SIMD instructions allow further parallelisation of the method in sub equation level. SIMD instructions are widely used for performance enhancement of computationally heavy algorithms [20–26].

9.1. The Computation Minimized Form of The FDTD3D Method

The computation minimized form of the FDTD3D method is as follows considering equal discretisation step size Δ in x, y and z directions. Δt is the time step size.

$$H_x|_{i,j,k}^{n+1/2} = \mathbb{A}\left[\left(E_y|_{i,j,k+1/2}^{n} - E_y|_{i,j,k-1/2}^{n}\right) - \left(E_z|_{i,j+1/2,k}^{n} - E_z|_{i,j-1/2,k}^{n}\right)\right] + H_x|_{i,j,k}^{n-1/2} \tag{16a}$$

$$H_y|_{i,j,k}^{n+1/2} = \mathbb{A}\left[\left(E_z|_{i+1/2,j,k}^{n} - E_z|_{i-1/2,j,k}^{n}\right) - \left(E_x|_{i,j,k+1/2}^{n} - E_x|_{i,j,k-1/2}^{n}\right)\right] + H_y|_{i,j,k}^{n-1/2} \tag{16b}$$

$$H_z|_{i,j,k}^{n+1/2} = \mathbb{A}\left[\left(E_x|_{i,j+1/2,k}^{n} - E_x|_{i,j-1/2,k}^{n}\right) - \left(E_y|_{i+1/2,j,k}^{n} - E_y|_{i-1/2,j,k}^{n}\right)\right] + H_z|_{i,j,k}^{n-1/2} \tag{16c}$$

$$E_x|_{i,j,k}^{n+1/2} = \mathbb{B}\left[\left(H_z|_{i,j+1/2,k}^{n} - H_z|_{i,j-1/2,k}^{n}\right) - \left(H_y|_{i,j,k+1/2}^{n} - H_y|_{i,j,k-1/2}^{n}\right)\right] + E_x|_{i,j,k}^{n-1/2} \tag{16d}$$

$$E_y|_{i,j,k}^{n+1/2} = \mathbb{B}\left[\left(H_x|_{i,j,k+1/2}^{n} - H_x|_{i,j,k-1/2}^{n}\right) - \left(H_z|_{i+1/2,j,k}^{n} - H_z|_{i-1/2,j,k}^{n}\right)\right] + E_y|_{i,j,k}^{n-1/2} \tag{16e}$$

$$E_z|_{i,j,k}^{n+1/2} = \mathbb{B}\left[\left(H_y|_{i+1/2,j,k}^{n} - H_y|_{i-1/2,j,k}^{n}\right) - \left(H_x|_{i,j+1/2,k}^{n} - H_x|_{i,j-1/2,k}^{n}\right)\right] + E_z|_{i,j,k}^{n-1/2} \tag{16f}$$

Here, $\mathbb{A} = \frac{\Delta t}{\mu_{i,j,k}\Delta}$ and $\mathbb{B} = \frac{\Delta t}{\epsilon_{i,j,k}\Delta}$ can be stored in the memory for each node in the grid.

9.1.1. Latency for General Purpose Instructions

Table 1 shows the add, subtract and multiply instructions required for each equation in Eq. (16) when general purpose instruction of Intel Haswell architecture. It also sums the total number of cycles needed for each equation and also to calculate the next time step. As can be seen the total number of CPU cycles needed for the FDTD cycle for one element is 102.

From this point onwards in all tables **Late.** has been used to denote "Latency", number of cycles needed to execute one instruction and **Ins.** has been used to denote "CPU Instruction".

9.1.2. Latency for General Purpose and SIMD Instructions

Table 2 presented the latency distribution for the 3D FDTD method. Compared to latency distribution presented in Table 1 for only general purpose instructions, every equation in Table 2 gets a latency reduction for the use of 1 SIMD subtraction instruction. Therefore, each of them gets a small latency reduction. The total impact is significant as the total latency for calculating 1 time-step reduces from 102 to **84**.

Table 1. Compute operations and latencies for 3D FDTD method with general purpose instructions.

Eqs.	Add		Sub		Mult		Total	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
(16a)	1	3	3	3	1	5	5	17
(16b)	1	3	3	3	1	5	5	17
(16c)	1	3	3	3	1	5	5	17
(16d)	1	3	3	3	1	5	5	17
(16e)	1	3	3	3	1	5	5	17
(16f)	1	3	3	3	1	5	5	17
Total for $\mathbf{E} \rightarrow \mathbf{H}$:							**15**	**51**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:							**15**	**51**
Total for One Time-step:							**30**	**102**

Table 2. Compute operations and latencies for 3D FDTD method with general purpose and SIMD Instructions.

Eqs.	Add		Sub		SIMD Sub		Mult		Total	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
(16a)	1	3	1	3	1	3	1	5	4	14
(16b)	1	3	1	3	1	3	1	5	4	14
(16c)	1	3	1	3	1	3	1	5	4	14
(16d)	1	3	1	3	1	3	1	5	4	14
(16e)	1	3	1	3	1	3	1	5	4	14
(16f)	1	3	1	3	1	3	1	5	4	14
Total for $\mathbf{E} \rightarrow \mathbf{H}$:									**12**	**42**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:									**12**	**42**
Total Operation for One Time-step:									**24**	**84**

9.2. The Computation Minimized Form of The Proposed Perforated FETD3D Method

Although to calculate the optimized form of FDTD3D fixed step size in all direction considered, for the proposed method no assumption is taken for the shape of the tetrahedral element. This is to preserve the flexibility to use any kind of tetrahedron. Ultimately the method will be used with adoptive or arbitrary mesh where any shape and size of tetrahedron can be used. This flexibility will require additional computation for the proposed. But as the method is expected to use the flexibility in practical scenarios, it is important to compare the proposed method with the flexibility with FDTD3D.

Therefore computation minimized form of the proposed method is as follows,

$$h_x^{(n+1)} = \mathbb{C}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial z}e_{yi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial y}e_{zi}^{(n)}\right) + h_x^{(n-1)} \tag{17a}$$

$$h_y^{(n+1)} = \mathbb{C}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}e_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}e_{xi}^{(n)}\right) + h_y^{(n-1)} \tag{17b}$$

$$h_z^{(n+1)} = \mathbb{C}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial y}e_{xi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial x}e_{yi}^{(n)}\right) + h_z^{(n-1)} \tag{17c}$$

$$e_x^{(n+1)} = \mathbb{D}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial y}h_{zi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial z}h_{yi}^{(n)}\right) + e_x^{(n-1)} \tag{17d}$$

$$e_y^{(n+1)} = \mathbb{D}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial z}h_{xi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial x}h_{zi}^{(n)}\right) + e_y^{(n-1)} \tag{17e}$$

$$e_z^{(n+1)} = \mathbb{D}\left(\sum_{i=1}^{4}\frac{\partial N_i}{\partial x}h_{yi}^{(n)} - \sum_{i=1}^{4}\frac{\partial N_i}{\partial y}h_{xi}^{(n)}\right) + e_{zk}^{(n-1)} \tag{17f}$$

Here, $\mathbb{C} = 1/\left(\mu\frac{dQ_2}{dt}\right)$ and $\mathbb{D} = 1/\left(\epsilon\frac{dQ_2}{dt}\right)$ can be stored in the memory for each associated element.

9.2.1. Latency for General Purpose Instructions

Table 3 list the latency distribution for the computation optimised formulation for the proposed 3D FETD governing equations in Eq. (17). As can be seen in the table, each of the equations cause same latency similar to the 3D FDTD method presented in Table 1. But each of the equations for the 3D FETD requires **69** CPU cycles compared to **17**. The main reason for the excessive use of the addition and multiplication operations. Excessive use of general instructions made the total performance for each time-step much slower than the 3D FDTD performance with general purpose instructions. The 3D FDTD theoretically requires **102** CPU cycles compared to the proposed 3D method requiring **414**. Therefore, the performance of the proposed 3D method with general purpose instructions are almost **4** times slower than the 3D FDTD with general purpose instructions.

9.2.2. Latency for General Purpose and SIMD Instructions

Table 4 presents the latencies associated with the proposed method with general propose and SIMD instructions together. It can be noted that when SIMD instructions are used for additions and multiplications, the latency of all the equations in Eq. (17) decrease significantly from 69 CPU cycles to **27** CPU cycles. Therefore, the total reduction of latency is more than **60**% from 414 CPU cycles in Table 3 to **162** in Table 4.

Although use of SIMD instructions significantly reduces the CPU cycle count for the proposed method. It falls short of the FDTD3D's 102 (using only general purpose instructions) and 84 (using both general purpose and SIMD instructions). To take the speed of the proposed method beyond FDTD3D the RRF advantage of the proposed FETD3D method could be used.

Table 3. Compute operations and latencies for 3D FETD method with general purpose instructions.

Eqs.	**Add**		**Sub**		**Mult**		**Total**	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
(17a)	7	3	1	3	9	5	17	69
(17b)	7	3	1	3	9	5	17	69
(17c)	7	3	1	3	9	5	17	69
(17d)	7	3	1	3	9	5	17	69
(17e)	7	3	1	3	9	5	17	69
(17f)	7	3	1	3	9	5	17	69
Total for $\mathbf{E} \rightarrow \mathbf{H}$:							**51**	**207**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:							**51**	**207**
Total Operation for One Time-step:							**102**	**414**

Table 4. Compute operations and latencies for 3D FETD method with general purpose and SIMD instructions.

Eqs.	Add		SIMD Add		Sub		Mult		SIMD Mult		Total	
	Ins.	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**	**Ins.**	**Late.**
(17a)	1	3	2	3	1	3	1	5	2	5	7	27
17b)	1	3	2	3	1	3	1	5	2	5	7	27
(17c)	1	3	2	3	1	3	1	5	2	5	7	27
(17d)	1	3	2	3	1	3	1	5	2	5	7	27
(17e)	1	3	2	3	1	3	1	5	2	5	7	27
(17f)	1	3	2	3	1	3	1	5	2	5	7	27
Total for $\mathbf{E} \rightarrow \mathbf{H}$:											**21**	**81**
Total for $\mathbf{H} \rightarrow \mathbf{E}$:											**21**	**81**
Total Operation for One Time-step:											**42**	**162**

9.3. Comparing Proposed FETD3D and FDTD3D Considering RRF

As the numerical characteristics of the IRT3D mesh the same as the FDTD3D there is added advantage of using it as the mesh of choice with the proposed method. The ET3D mesh has a RRF of 1.903 for all resolutions. Therefore, the resolution of the mesh can be reduced by 1.903 times keeping the numerical error the same. This could allow the proposed method to run faster then the FDTD3D method as it will have to process significantly less number of elements then the FDTD3D method.

To analyse the effect of the RRF on the speed of the proposed method a Python code was developed. Since the RRF is insensitive of resolution, for this analysis the resolution for the proposed 3D FETD was considered to be $10/\lambda$. Therefore, the resolution for the 3D FDTD method was $19/\lambda$. The domain size was determined interns of number of division in each direction. For simplicity, a cubic domain was considered.

The program generated required number of elements necessary for a cubic domain with ET3D mesh and calculated the latency using the information presented in Tables 3 and 4. It also generated the 3D cubic grid for the FDTD3D with necessary number of cells and calculated the CPU latency using information from Tables 1 and 2. A theoretical CPU performance ratio can be calculated by dividing the CPU latency of the FDTD3D method by the CPU latency of the proposed FETD3D method. Fig. 12 shows the outputs of the Python code[†]. As can be seen in Fig. 12(a), the CPU performance ratio is much higher at when the number of division is much lower. It decrease with a saturation curve and settles down around **1.71** at higher resolutions when only general purpose instructions are considered with the RRF.

Figure 12(b) shows the performance with SIMD instructions. Similar to Fig. 12(a), the CPU performance ratio is higher at the beginning and settles around **3.6** at higher resolution. Therefore, the proposed FETD3D method is theoretically 1.71 times faster then the FDTD3D method when only general purpose computation is considered. But when the SIMD instructions are considered the method is 3.6 times faster then the FDTD3D method regardless of the resolution of choice. This is because RRF is insensitive to resolution (Section 8).

10. FUTURE PROSPECT

The proposed method is explicit in formulation and data parallel in nature. Therefore, it is very much suitable for multi-processing, distributed processing, GPGPU (General Purpose Computing on Graphics

[†] The small increase in performance at the FETD resolution 170 is due to a round off issue. As resolutions are integers the Python code used `round()` function from the numpy package to round off the equivalent resolution for the FDTD calculation. As the RRF is 1.903, the round off function started to chose higher value from the FETD resolution 170. Hence, both the plot shows a small performance increase at 170.

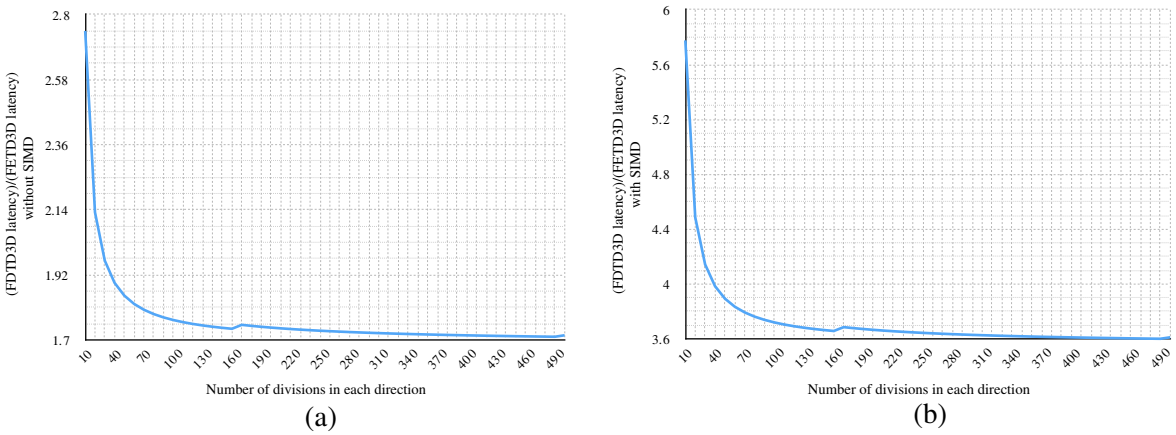


Figure 12. CPU latency comparison between FDTD3D and FETD3D with ET3D mesh. (a) FDTD3D latency over FETD3D latency vs size of computational domain without SIMD instructions, (b) FDTD3D latency over FETD3D latency vs size of computational domain with SIMD instructions.

Processing Units)[‡] and heterogeneous processing implementations. For the simulations of this paper the proposed method has already been implemented in multi-processing. However, its performance can be improved by implementing the method for GPGPU computing or heterogeneous computing. A distributed computing implementation could allow simulate much larger domain, using many computers in parallel.

The method is FE based and the nodes of the elements of the meshes are movable, which allows a better approximation for any curved or slanted structure than that can be obtained using the FDTD method. The movability of the nodes could also allow quick modification of the structure and/or the computational domain in between two time steps. The much superior numerical dispersion property seen will allow more accurate simulations than through the use of the FDTD at a much lower resolution using the proposed FE-based method. If used with an optimized and efficient meshing scheme, the efficiency of the time domain simulation could be improved further.

Due to the use of grids the FDTD3D simulations were always performed mostly in cuboid computational domain. A cuboid domain is efficient for straight guide simulations. But a cuboid domain could be wasteful for modelling bent, L-shaped structures and for complicated integrated designs. Since the proposed method uses FE mesh system, an arbitrary shape computational domain is a possibility like any other FE based method. This could take the speed of simulation of the proposed method far beyond the speed offered by the FDTD method.

But, both the efficient meshing scheme and the arbitrary shaped computational domain are subject to further research. This is because the proposed method uses a unique performed double mesh technique which was never used before. The shape of the computational domain depends on Perfectly Matched Layer (PML) boundary condition which was designed keeping FDTD in mind [27, 28]. Since the method could allow none uniform and arbitrary shaped computational domain which might change its shape with time, a more robust PML is required. Therefore, a new PML could also be a future area of research.

11. CONCLUSION

In this paper a three-dimensional implementation of the perforated FE-based time domain technique has been presented. The approach presented uses an innovative perforated mesh system, which, as discussed earlier, reduces the computational load significantly. The effectiveness of the proposed method has been illustrated by performing several benchmarking simulations.

The distribution of normalised numerical speed of propagation has been shown by deriving the numerical dispersion relation from the 3D governing equations. It has also been proven that the

[‡] Graphics processors are designed as to be massively parallel computation processors which can perform hundreds and thousands of instructions parallel fashion. As the proposed method is data parallel and explicit, a GPGPU is implementation could accelerate the execution time many times.

numerical dispersion characteristics of the proposed method with the IRT3D mesh and the FDTD3D method are exact match and the proposed method with the ET3D mesh has significantly lower standard deviation in normalised numerical speed. Thus propagation of electromagnetic wave is much stable in the ET3D mesh. Using the standard deviation data the resolution relation between the proposed method with ET3D mesh and the FDTD3D method (or the proposed method with IRT3D mesh) has been determined. From the resolution relation the Resolution Reduction Factor (RRF) has been calculated. It has been shown that it is resolution invariant and a constant.

The computation optimised version of the FDTD3D governing equations and the governing equations of the proposed method have been presented. Using the Intel Heswell instruction sets [19], the minimum number of CPU cycles were calculated for both the methods. It has been shown that the use of SIMD instructions could significantly favor the proposed method over the FDTD3D method.

Finally, it has been shown that proposed method can increase the execution speed beyond the FDTD3D method for both general purpose and SIMD implementation when the RRF calculated in Section 8 is considered.

ACKNOWLEDGMENT

A special thanks to U. S. Army Communications-Electronics Research, Development & Engineering Center (CERDEC) for partically funding the development of the method.

REFERENCES

1. Hagness, S. and A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method (Second Edition)*, 2 Edition, Artech House, 2000.
2. Taflove, A. and M. Brodwin, "Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 23, No. 8, 623–630, 1975.
3. Sun, G. and C. Trueman, "Some fundamental characteristics of the one-dimensional alternate-direction-implicit finite-difference time-domain method," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 52, No. 1, 46–52, 2004.
4. Lee, J., R. Lee, and A. Cangellaris, "Time-domain finite-element methods," *IEEE Transactions on Antennas and Propagation*, Vol. 45, No. 3, 430–442, 1997.
5. Yee, K., "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, Vol. 14, No. 3, 302–307, 1966.
6. Gedney, S. and U. Navsariwala, "An unconditionally stable finite element time-domain solution of the vector wave equation," *IEEE Microwave and Guided Wave Letters*, Vol. 5, No. 10, 332–334, 1995.
7. Cangellaris, A., C. Lin, and K. Mei, "Point-matched time domain finite element methods for electromagnetic radiation and scattering," *IEEE Transactions on Antennas and Propagation*, Vol. 35, No. 10, 1160–1173, 1987.
8. Hesthaven, T. W. J. S., "High-order/spectral methods on unstructured grids i. Time-domain solution of Maxwell's equations," Tech. Rep. 2001-6, ICASE NASA Langley Research Center, Hampton, Virginia, March 2001.
9. Songoro, H., M. Vogel, and Z. Cendes, "Keeping time with Maxwell's equations," *IEEE Microwave Magazine*, Vol. 11, No. 2, 42–49, 2010.
10. Raiyan Kabir, S. M., B. M. A. Rahman, A. Agrawal, and K. T. V. Grattan, "Elimination of numerical dispersion from electromagnetic time domain analysis by using resource efficient finite element technique," *Progress In Electromagnetics Research*, Vol. 137, 487–512, 2013.
11. Courant, R., K. Friedrichs, and H. Lewy, "Über die partiellen differenzengleichungen der 568 mathematischen physik," *Mathematische Annalen*, Vol. 100, No. 1, 32–74, 1928.
12. Cangellaris, A., "Time-domain finite methods for electromagnetic wave propagation and scattering," *IEEE Transactions on Magnetics*, Vol. 27, No. 5, 3780–3785, 1991.

13. Leung, D., N. Kejalakshmy, B. M. A. Rahman, and K. Grattan, "Rigorous modal analysis of silicon strip nanoscale waveguides," *Optics Express*, Vol. 18, No. 8, 8528–8539, 2010.
14. Kirby, E., J. Hamm, K. Tsakmakidis, and O. Hess, "FDTD analysis of slow light propagation in negative-refractive-index metamaterial waveguides," *Journal of Optics A: Pure and Applied Optics*, Vol. 11, No. 11, 114027, 2009.
15. Rahman, B. M. A. and J. B. Davies, "Finite-element solution of integrated optical waveguides," *Journal of Lightwave Technology*, Vol. 2, No. 5, 682–688, 1984.
16. Rahman, B. M. A. and J. B. Davies, "Finite-element analysis of optical and microwave waveguide problems," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 32, No. 1, 20–28, 1984.
17. Themistos, C. and B. M. A. Rahman, "Design issues of a multimode interference-based 3-dB splitter," *Applied optics*, Vol. 41, No. 33, 7037–7044, 2002.
18. Juntunen, J. and T. Tsiboukis, "Reduction of numerical dispersion in FDTD method through artificial anisotropy," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 48, No. 4, 582–588, 2000.
19. Intel Corporation, *Intel® 64 and IA-32 Architectures Optimization Reference Manual*, 2014.
20. Jamieson, L. H., P. T. Mueller, and H. J. Siegel, "FFT algorithms for simd parallel processing systems," *Journal of Parallel and Distributed Computing*, Vol. 3, No. 1, 48–71, 1986.
21. Agaian, S. and D. Gevorkian, "Synthesis of a class of orthogonal transforms. Parallel simd-algorithms and specialized processors," *Pattern Recognition and Image Analysis*, Vol. 2, No. 4, 394–408, 1992.
22. Ben-Asher, Y., D. Egozi, and A. Schuster, "2-D simd algorithms for perfect shuffle networks," *Journal of Parallel and Distributed Computing*, Vol. 16, No. 3, 250–257, 1992.
23. Apostolakis, J., P. Coddington, and E. Marinari, "New simd algorithms for cluster labeling on parallel computers," *International Journal of Modern Physics C*, Vol. 4, No. 04, 749–763, 1993.
24. Chen, H., N. S. Flann, and D. W. Watson, "Parallel genetic simulated annealing: A massively parallel simd algorithm," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 2, 126–136, 1998.
25. Hong, I., S. Chung, H. Kim, Y. Kim, Y. Son, and Z. Cho, "Ultra fast symmetry and simd-based projection-backprojection (ssp) algorithm for 3-D pet image reconstruction," *IEEE Transactions on Medical Imaging*, Vol. 26, No. 6, 789–803, 2007.
26. Goualard, F., "Fast and correct simd algorithms for interval arithmetic," *PARA'08*, Springer, 2010.
27. Berenger, J., "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of Computational Physics*, Vol. 114, No. 2, 185–200, 1994.
28. Berenger, J., "Perfectly matched layer for the FDTD solution of wave-structure interaction problems," *IEEE Transactions on Antennas and Propagation*, Vol. 44, No. 1, 110–117, 1996.