Phys Enph 479/879 Assignment #5: The Finite-Difference Time-Domain Method

Josh Dykstra[1,*]

[1] *Department of Physics, Engineering Physics and Astronomy, Queen's University, Kingston, ON K7L 3N6, Canada*

(Dated: March 30, 2020)

This assignment discusses the use of the Finite-Difference Time-Domain Method for solving Electromagnetic problems. A model of free space propagation of electric fields is created with absorbing boundary conditions, total scattered-field with reflection and transmission from a simple dielectric. The model will then be extended to involve the displacement field using flux density in which the medium of propagation will have loss and frequency-dependent characteristics using Drude and Lorentz models formulated by properties of Z-transforms. The program will then be updated to accomodate for two dimensions and then optimized using python package numba, slicing and vectorization and MPI parallelization, comparing and contrasting computational efficiency.

I. INTRODUCTION

A. Finite-Difference Time Domain Method

The finite difference time domain method provides numerical solutions to electromagnetic problems. This method is arguably the simplest, both conceptually and in terms of implementation, but the accuracy is contingent upon the implementation and is generally rather computationally expensive. The characteristic dimensions of the the domain of interest have to be within the order of a wavelength in size otherwise "quasi-static" approximations or ray-based methods generally provide more efficient solutions depending on the physical features of the system [V].
Electromagnetism is the study of interactions between electrically charged particles, the forces that are exerted are governed by Maxwell's Time-Varying Field Equations:

$$\nabla \cdot \mathbf{D} = \rho_v,$$

$$\nabla \cdot \mathbf{B} = 0,$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t},$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}.$$

The flow of Maxwell's Equations creates the following loop (All these processes happen simultaneously in nature but in FDTD and in a computable program it follows this pattern):

1. A circulating **E** field induces a change in the **B** field at the center of circulation.
2. A **B** field induces an **H** field in proportion to the permeability.
3. A circulating **H** field induces a change in the **D** field at the center of circulation.
4. The **D** field induces an **E** field in proportion to the permittivity. (Cycle restarts at 1.)

The Finite-Difference Time-Domain method employs finite differences as approximations to both the spatial and temporal derivatives in Maxwell's equations, most specifically Ampere's and Faraday's laws.

B. The Yee Algorithm

The first algorithm for FDTD was proposed by Kane Yee in 1966 which uses second order central-differences, this case works in a single dimension but can be extended for higher dimensions. The key points of the algorithm are:

1. Replace all the derivatives in Ampere's and Faraday's laws with finite differences. Discretize space and time so that the electric and magnetic fields are staggered in both space and time.
2. Solve the resulting difference equations to obtain "update equations" that express the (unknown) future fields in terms of (known) past fields.
3. Evaluate the magnetic fields one time-step into the future so they are now known (effectively they become past fields).
4. Evaluate the electric fields one time-step into the future so they are now known (effectively they become past fields).
5. Repeat the previous two steps until the fields have been obtained over the desired duration.

II. THEORY AND EQUATIONS

A. Finite-Difference Time-Domain Algorithm

The time-dependent curl equations in free space that follow from the Maxwell relations are given by the equations:

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\epsilon_0}\nabla \times \mathbf{H}, \quad \frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu_0}\nabla \times \mathbf{E}. \tag{1a}$$

In one dimension these collapse to the following, with E_x and H_y propagating in the z direction:

$$\frac{\partial E_x}{\partial t} = -\frac{1}{\epsilon_0}\frac{\partial H_y}{\partial z}, \quad \frac{\partial H_y}{\partial t} = -\frac{1}{\mu_0}\frac{\partial E_x}{\partial z}. \tag{2a}$$

* 19jjd1@queensu.ca

The central difference approximations for the condition $(\Delta t, \Delta x \equiv \Delta z)$ are then:

$$\frac{E_x^{n+1/2}(k) - E_x^{n-1/2}(k)}{\Delta t} = -\frac{1}{\epsilon_0}\frac{H_y^n(k+\frac{1}{2}) - H_y^n(k-\frac{1}{2})}{\Delta x}, \tag{3a}$$

$$\frac{H_y^{n+1}(k+\frac{1}{2}) - H_y^n(k+\frac{1}{2})}{\Delta t} = -\frac{1}{\mu_0}\frac{E_x^{n+1/2}(k+1) - E_x^{n+1/2}(k)}{\Delta x}. \tag{3b}$$

Rearranging for these equations gives us our iterative FDTD algorithm for a single dimension, which can be computed by a machine:

$$E_x^{n+1/2}(k) =$$

$$E_x^{n-1/2}(k) - \frac{\Delta t}{\epsilon_0 \Delta x}\left[H_y^n\left(k+\frac{1}{2}\right) - H_y^n\left(k-\frac{1}{2}\right)\right], \tag{4}$$

For the E field, and for the H field in the y-direction:

$$\mathrm{H}_y^{n+1}\left(k+\tfrac{1}{2}\right) =$$

$$H_y^n\left(k+\tfrac{1}{2}\right) - \tfrac{\Delta t}{\mu_0 \Delta x}\left[E_x^{n+1/2}(k+1) - E_x^{n+1/2}(k)\right], \tag{5}$$

It is preferable to have E and H to be similar in size, with an injected plane wave. To do this the following re-scaling relation is required:

$$\tilde{E} = \sqrt{\frac{\epsilon_0}{\mu_0}}\, E = \epsilon_0 c\, E, \tag{6}$$

The Courant condition states that the time step of a numerically explicit time integration scheme must be less than a certain time, otherwise the simulation produces incorrect results. The condition for this case is as follows:

$$\Delta t < \frac{\Delta x}{\sqrt{n} c} \tag{7}$$

B. Absorbing Boundary Condition (ABC)

For the wave not to be reflected backwards at the edges of the simulation the following absorbing boundary condition (ABC) must be implemented:

$$E_x^n(0) = E_x^{n-2}(1), \tag{8}$$

$$E_x^n(kend-1) = E_x^{n-2}(kend-2). \tag{9}$$

C. Total Scatter Field

Injecting a source will lead to a field that is propagating in both directions, we preferably want to limit the propagation in the backwards direction otherwise it will cause inconveniences when measuring reflected wave for scattering problems within a dielectric medium. To do this we subtract off sections of the source wave at the scattered only region as so:

$$E_x(isource) = E_x(isource) - 0.5 H_{source}(isource)$$
$$H_y(isource-1) = H_y(isource-1) - 0.5 E_{source}(isource)$$

D. Dielectric

We can now implement a 1fs pulse to propagate through a thin film with dielectric constant $\epsilon = 9$. We can measure the reflected fields (before source) and transmitted fields (after source and thin film). Numerically these are found by performing a fast fourier transform (FFT) on the fields R and T converting them from the original domain into its frequency domain.

$$T = \left|\frac{E_\mathrm{t}(\omega)}{E_\mathrm{in}(\omega)}\right|^2, \tag{10}$$

$$R = \left|\frac{E_\mathrm{r}(\omega)}{E_\mathrm{in}(\omega)}\right|^2, \tag{11}$$

Here $E_{in}(\omega)$ represents the incident electric field in its frequency domain and $E_r(\omega), E_t(\omega)$ the reflected electric field and transmitted electric field respectively. The analytical solution in frequency space can be formulated as follows:

$$n = \sqrt{\epsilon}, \quad r_1 = \frac{1-n}{1+n}, \quad r_2 = \frac{n-1}{n+1}, \quad k_0 = \frac{\omega}{c}, \tag{12}$$

$$r(\omega) = \frac{r_1 + r_2 e^{2ik_0 Ln}}{1 + r_1 r_2 e^{2ik_0 Ln}}, \quad t(\omega) = \frac{(1+r_1)(1+r_2)e^{2ik_0 Ln}}{1 + r_1 r_2 e^{2ik_0 Ln}}. \tag{13}$$

This yields the total analytical solution for the transmitted and reflected fields,

$$T_\mathrm{an} = |t(\omega)|^2, \quad R_\mathrm{an} = |r(\omega)|^2. \tag{14}$$

E. Reformulating using Flux Density

We can now introduce the electric flux density into our FDTD model to make it more general. We know from Maxwell's equations the displacement field relation:

$$\frac{\partial \mathbf{D}}{\partial t} = \nabla \times \mathbf{H}, \tag{15}$$

With the electric flux density:

$$\mathbf{D} = \epsilon_0 \epsilon(\omega)\mathbf{E}(\omega). \tag{16}$$

With a similar rescaling as previously done with the electric field:

$$\tilde{\mathbf{E}} = \epsilon_0 c\mathbf{E}, \quad \tilde{\mathbf{D}} = c\mathbf{D}. \tag{17}$$

The full set of rescaled equations are then:

$$\frac{\partial \tilde{\mathbf{D}}}{\partial t} = c\nabla \times \mathbf{H}, \tag{18}$$

$$\tilde{\mathbf{D}} = \epsilon(\omega)\tilde{\mathbf{E}}(\omega), \tag{19}$$

$$\frac{\partial \mathbf{H}}{\partial t} = -c\nabla \times \tilde{\mathbf{E}}. \tag{20}$$

F. Fourier and Laplacian Transformations

The following Fourier and Laplacian transformations can be very useful in simplifying differential equations, a convolution in time is a simple multiplication in frequency space, it is important to note that these transforms work for analogue signals which are different to the digital signals most commonly used by numerical time solvers.

$$x(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(t) e^{-i\omega t} dt \tag{21}$$

$$x(s) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} x(t) e^{-st} dt \tag{22}$$

The Z transform is useful for dealing with digital signals and can be thought of as a delay operator. It converts a discrete-time signal into a complex frequency-domain representation.

$$\mathcal{Z}\{x[n]\} = x(z) = \sum_{n=0}^{\infty} x[n] z^{-n} \tag{23}$$

G. Lossy Plasma and the Drude Model

The Drude model is an application of kinetic theory that explains the transport properties of electrons in metals assuming the material has motionless positive ions and a non-interacting electron gas. Drude's dielectric function is as follows:

$$\epsilon_{\mathrm{D}}(\omega) = 1 - \frac{\omega_p^2}{\omega^2 + i\omega\alpha}, \tag{24}$$

The Lorentz model is deduced from the classical theory solution to the problem of an electron bound to a nucleus driven by an oscillating electric field similar to the response of a mass on a spring with dampening and an external driving force.

$$\epsilon_{\mathrm{L}}(\omega) = 1 + \frac{f_0\, \omega_0^2}{\omega_0^2 - \omega^2 - i2\omega\alpha}, \tag{25}$$

$$D_x(\omega) = \epsilon(\omega) E_x(\omega), \tag{26}$$

$$\mathbf{D}(\omega) = \epsilon(\omega)\mathbf{E}(\omega) \rightarrow \Delta t \cdot \epsilon(z)\mathbf{E}(z), \tag{27}$$

H. Numba

Numba is a Python package specially built for scientific computing that translates functions into optimised machine code at runtime using industy-standard LLVM compiler library allowing numerical Python code to reach the speeds most commonly associated with C or FORTRAN.

I. Slicing, Vectorisation and MPI

Array slicing and vectorisation is the process which optimises the repetition of similar tasks on a subset of data by performing the operations simultaneously. The technique utilises index manipulation to greatly speed up the compuation time for large scaled repetitve processes.
The Message Passing Interface (MPI) is a messaging protocol system widely used for solving significant scientific and engineering problems on multiple processor nodes within a computer. Like slicing and vectorisation it is particularly useful when a computer is tasked with performing the same calculation multiple times on different data sets. It splits the job into different sections and divides them out between a set number of slave processors to share the workload and improve efficiency. Generally there is a master processor which is in charge of communicating to the other slaves the tasks that are required, and once the tasks completed it receives the information from each individual slave processor and collates it all together. A downside to using MPI is that the communication between processors and nodes is a time sensitive process and can cause delays, especially when large numbers of processors are used. More time is then spent sending and receiving data than is actually spent computing mathematical problems.

III. IMPLEMENTATION

To implement the time varying fields we create arrays for each type of field and have them update after each time step using finite-difference time-domain method as discussed. A two femtosecond pulse is used to in this model characterised as follows:

$$\Omega_{pulse} = exp(-\frac{(t-t_0)^2}{2(\frac{2fs}{\Delta t})^2}) \tag{28}$$

A. Absorbing Boundary Condition and Scatter-Field

To implement the absorbing boundary condition (ABC) we invoke the equations 9 after the second time iteration and after the electric field has been updated. Implementing the Scatter field approximation we substitute in equations II C in for our original E and H fields, which now appear as follows:

$$E_x[isource] = E_x[isource] - 0.5 * \Omega_{pulse}(t + 1/2)$$
$$H_y[isource - 1] = Hy[isource - 1] - 0.5 * \Omega_{pulse}(t)$$

This causes the pulse to be launched in only a single forward moving direction (to the right) so that later when we introduce the dielectric our reflected wave measured is purely from the reflection in the dielectric and not the back scattered injected pulse.

B. Dielectric

Adding the thin film to our model is the next step, it is important to make sure that the diectric is not placed somewhere that will disrupt the initial pulse before it is full estabilised, for this reason the dielectric is chosen to start ahead of the source halfway across the gird. A thin film of $1\mu m$ corresponds to 50 grid points, we select the dielectric constant to be $\epsilon = 9$ at all points within the dielectric and 1 everywhere else (free space).

C. Transmission and Reflection

To measure the transmitted wave and reflected wave through the dielectric a point is chosen on the grid behind the source for the reflected wave and after the source and dielectric for the transmitted wave, this point then records the amplitude of the electric field as a function of time, the numerical transmission and reflection coefficients can be calculated from equations 11. The analytical solution is also calculated and the two compared together.

D. Reformulation With Displacement Field

To make a more general model we update our existing simulation to add in the displacement field, this is generally quite simple as we just need to add an extra array and implement the flux density curl equation from Maxwell's equations. By doing this it enables us to create dispersive media for our pulse to travel through so we can implement both the Drude and Lorentz dispersion models.

E. Drude and Lorentz Dispersion Model

To implement the Drude and Lorentz models we introduce a new array S that will perform the space transformation into the sampled time-domain. This array will have to store two time steps before. The relation for the electric field then becomes:

$$\mathbf{E}^n = \mathbf{D}^n - \mathbf{S}^n$$

$$\mathbf{S}^n = (1 + e^{\alpha\Delta t})\mathbf{S}^{n-1} - e^{-\alpha\Delta t}\mathbf{S}^{n-2} + \frac{\Delta t\omega_0^2}{\alpha}(1 - e^{-\alpha\Delta t})\mathbf{E}^{n-1}$$

With the following input parameters:

$$\alpha = 1.4 \times 10^{14}\text{rads/s}$$
$$\omega_p = 1.26 \times 10^{15}\text{rads/s}$$

The Lorentz model is slightly different and to find the S relation some algebraic manipulation of the properties of table II must be done, it is then found that:

$$\mathbf{S}^n = (1 + e^{\alpha\Delta t})\mathbf{S}^{n-1} - e^{-\alpha\Delta t}\mathbf{S}^{n-2} + \frac{\Delta t\omega_0^2}{\alpha}(1 - e^{-\alpha\Delta t})\mathbf{E}^{n-1}$$
$$\beta = \sqrt{\omega_0^2 - \alpha^2}$$

F. Two Dimensions

Using the flux density formulation the second dimension can be easily implemented using the same Maxwell's equations as before but using the transverse magnetic staggered grid. Our animation will appear as a contour map to include the two spatial dimensions and the Electric field amplitude dimension.

G. Numba, Slicing and Vectorisation

The Numba Python package is implemented with the following command above the function to be converted into optimised machine code:

```
@numba.jit(nopython=True)
```

We expect to see a significant improvement in computation time using the Numba package, the run-times for 3 tests will be recorded and averaged to compare with other optimisation operations.
To implement slicing we use array index manipulation to perform the same operation on each of the elements for variables D_z , E_z , H_x and H_y. The array manipulation is as follows:

```
Dz[1:-1,:] -=   0.5*(Hy[:-2,:])
Dz[:,1:-1] += 0.5*(Hx[:,:-2])
Dz[:,:] += 0.5*(Hy[:,:] -Hx[:,:])
Ez[:,:] =   ga[:,:]*Dz[:,:]

Dz[isource,jsource]= ...
Dz[isource,jsource] + pulse
Ez[isource,jsource]= ...
ga[isource,jsource]*(Dz[isource,jsource])

Hx[:,1:-1] -= 0.5 * Ez[:,2:]
Hx[:,:] += 0.5 * Ez[:,:]
Hy[1:-1,:] += 0.5 * Ez[2:,:]
Hy[:,:] -= 0.5*Ez[:,:]
```

The array *ga* is the dielectric value at each spatial grid point. Also, isource and jsource are the x,y source locations of the initial pulse.

H. MPI

To implement MPI a few minor changes must be made to the structure of the code, firstly we choose the initial processor, id=0 to be the master, this means that it will be

main communication node of the system and will compile together all the data once each of the slave nodes has performed all its operations. We start by adjusting the size of the arrays for the fields to be truncated in the x-dimension, we divide the maximum value of x by the number of processors running the job. For example if we had a 160x160 grid and we wanted to run on four processors, each node would have its own 40x160 chunk of the field. When communicating data for the next processor to be able to perform its task it needs to know two pieces of critical information: The first element in its array and the last. To do this we set up something called a dead layers, slots of data that will be passed upon completion to the next or previous processor that are assigned specifically to hold the data to be sent. We then alter the dielectric conditions to make sure that our new dielectric is not repeated for each processor and only occurs once in total, the following adjustment to the code is as follows:

```
for j in range (0,Ymax):
    for i in range (0,n):
        if (i+((Xmax*id)/p)>X1
        and i+((Xmax*id)/p)<X2+1
        and j>Y1 and j<Y2+1):
            ga[i,j] = 1./eps2
```

We now implement the boundary conditions, these are only applicable to the first and last processors and are the same as before. To perform the communication between nodes we have it set up that the first processor only sends its dead-layer information to the processor next in line. The last only sends data "upwards", to the previous processor and all the rest send data both up and down. The first processor then collects all the data and collates it together to reproduce the graph results achieved originally.

IV. CONCLUSIONS

A. Absorbing Boundary Condition, Scatter-field and Simple Dielectric

As can be seen from Figures 4 and 3 the implementation of the ABCs has been successful, the edges of the simulation grid do not reflect back the initial wave and the injected pulse only travels in the forward moving direction. The dynamics of the dielectric can be seen most clearly in the animation, when the pulse travels through it part of the wave is reflected at the boundary, the strength of which depending on the dielectric constant. The reflected waves get trapped constantly reflecting at both boundaries of the dielectric causing a drawn out resonant response depending on the length of the dielectric this behaviour can be seen by the electric field amplitudes for the transmitted and reflected waves shown in figure 5. Finding the transmission and reflection coefficients for the electric fields required us to perform a FFT to change the domain into frequency space. Plots 6 and 7 show the coefficients as a function of the frequency for a truncated frequency domain, both graphs show that the sum of the two coefficients is constant from which we can conclude that our medium has no loss associated with it, as the fractions of the reflected and transmitted fields constitutes the total electric field. The analytical solution superimposed over figure 7 shows excellent agreement to the numerical solution.

B. Drude and Lorentz Dispersion Model

Implementing the media with loss capacity first required us to reformulate the program to include the displacement fields. We consider this to be correct as we obtain the same set of results as we did for the first section. Plot 9 shows the sum of the transmission and reflection coefficients is no long constant at 1, instead we see loss in the system, an amount which depends on the length of the dielectric. The greater the size of the dielectric the more loss we see in the system, with substantial effect within the range $200 \rightarrow 150$THz for $1\mu m$ and $2\mu m$. The analytical solution shown in plot 10 shows again excellent agreement to our numeric solution. For the Lorentz dispersion we see again in 12 that the sum of the reflected and transmitted waves is no longer constant. As the frequency approaches 200THz the central frequency of the injected plane wave and the resonant frequency of the oscialltor the total electric field falls to zero if the size of the dielectric is large enough. The transmitted field which is the most of the contribution towards the sum then returns back to its original position close to 100%. The analytical solution shown in figure 13 shows once again amazing agreement towards the numerical solution.

C. Two Dimensions, Numba, Slicing and Vectorisation

We extend our program into two dimensions, to visualise this we create a contour map as shown in figure 14, this shows snapshots of a 2000 step simulation with a grid size of 160×160.

Figure 1. Computation times for 161×161 grid for 2000 steps.

Original Code (s)	Numba (s)	Sliced (s)	Numba + Sliced (s)
311.9	1.49	0.91	6.21

Figure 2. Computation times for 1610×1610 grid for 200 steps.

Original Code (s)	Numba (s)	Sliced (s)	Numba + Sliced (s)
3095	19.04	37.8	61.43

Tables 2 and 1 show the computational times for each of the different optimisation tasks, clearly showing that for 2000 steps and a grid size of 161×161 the fastest method is the slicing on its own shortly followed by Numba. The base code is as expected the longest to run, which shows that all of the methods used to optimise are in fact making the program more efficient computationally. For a 1610×1610 we see the exact mirror of scaling for the different processes, once again with Numba being the fastest to perform its operations in 19.04 seconds.

V. REFERENCES

[1] Schneider, J. B., 2017. Introduction to the Finite-Difference Time Domain Method: FDTD in 1D. In: *Understanding the Finite-Difference Time-Domain.* s.l.:s.n., p. 33.

Table I. Selected properties of the Z Transform

Property	Time Domain	Z Domain
Definition	$x[n]$	$x(z)$
Time shift	$x[n-m]$	$z^{-m}x(z)$
Convolution	$y[m] = \Delta t \cdot \sum_{n=0}^{\infty} h[m-n]x[n]$	$\Delta t \cdot h(z)x(z)$

Table II. Example Z Transforms

Frequency	Sampled-Time	Z Domain
1	$\delta[n]$ (impulse func)	1
$\frac{1}{i\omega}$	$u[n]$ (step func)	$\frac{1}{1-z^{-1}}$
$\frac{\beta}{(\alpha^2+\beta^2)-\omega^2-i2\omega\alpha}$	$e^{-\alpha\Delta t\cdot n}\sin(\beta\Delta t)u[n]$	$\frac{e^{-\alpha\Delta t}\sin(\beta\Delta t)z^{-1}}{1-2e^{-\alpha\Delta t}\cos(\beta\Delta t)z^{-1}+e^{-2\alpha\Delta t}z^{-2}}$
$-\frac{\omega_p^2}{\omega^2+i\omega\alpha}$	$\frac{\omega_p^2}{\alpha}(1-e^{-\alpha\Delta t\cdot n})u[n]$	$\frac{\frac{\omega_p^2}{\alpha}(1-e^{-\alpha\Delta t})z^{-1}}{1-(1+e^{-\alpha\Delta t})z^{-1}+e^{-\alpha\Delta t}z^{-2}}$

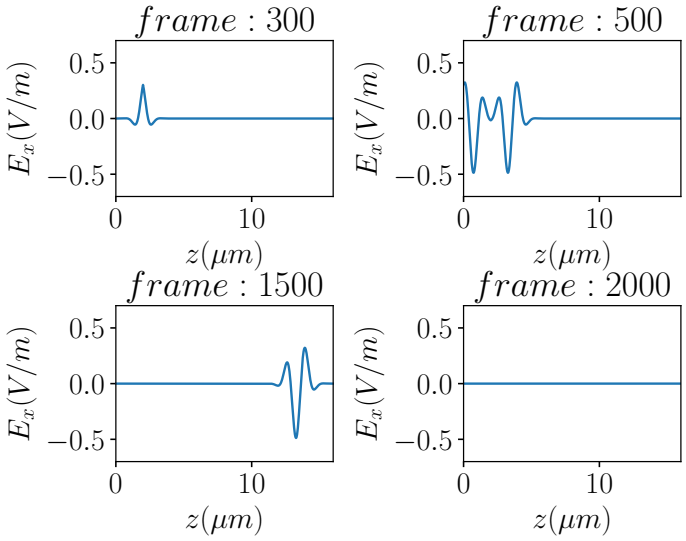

Figure 3. Snapshots of the animation displaying the absorbing boundary conditions (ABC), Pulse is completely absorbed at the boundary, no reflection can be seen.

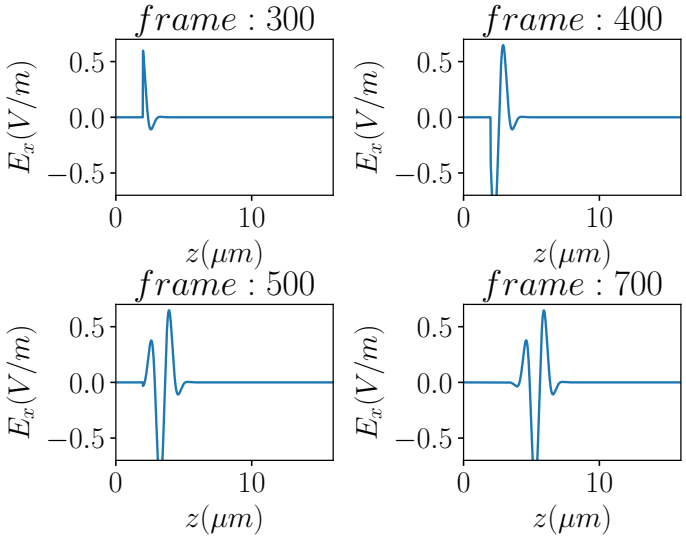

Figure 4. Animation snapshots displaying the scatter field with only a forward moving pulse, none of the wave travels backwards.

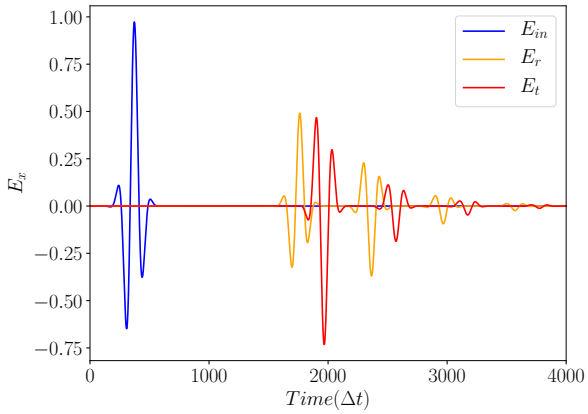

Figure 5. Plot showing the electric field for the incident (blue), reflected (orange) and transmitted waves (red) as a function of time.

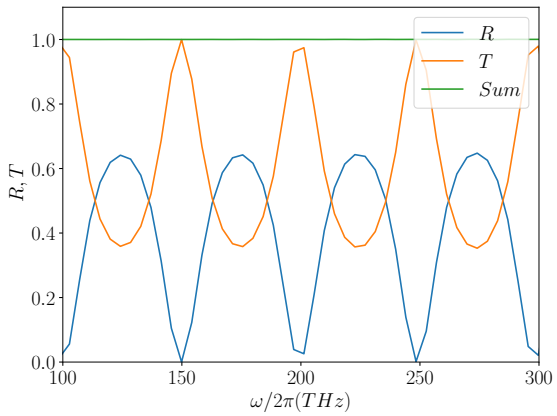

Figure 6. Plot showing the Fast Fourier Transforms (FFT) of the reflected and transmitted fields combined with the sum of the two.

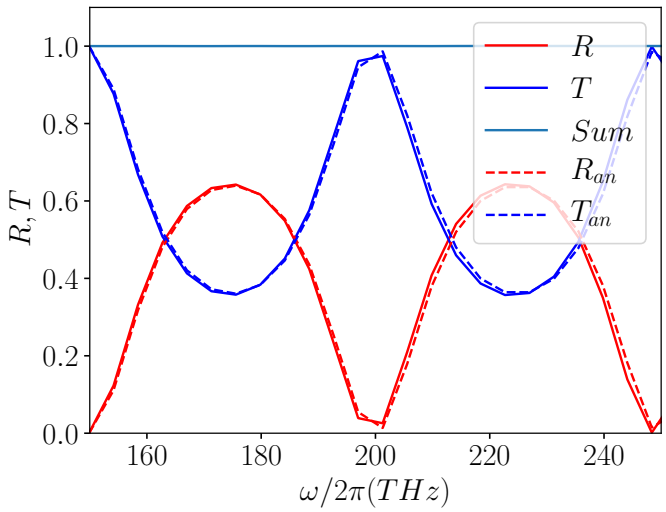

Figure 7. Analytical solution to the fourier transforms of the transmitted and reflected fields superimposed on top of the numerical solution.

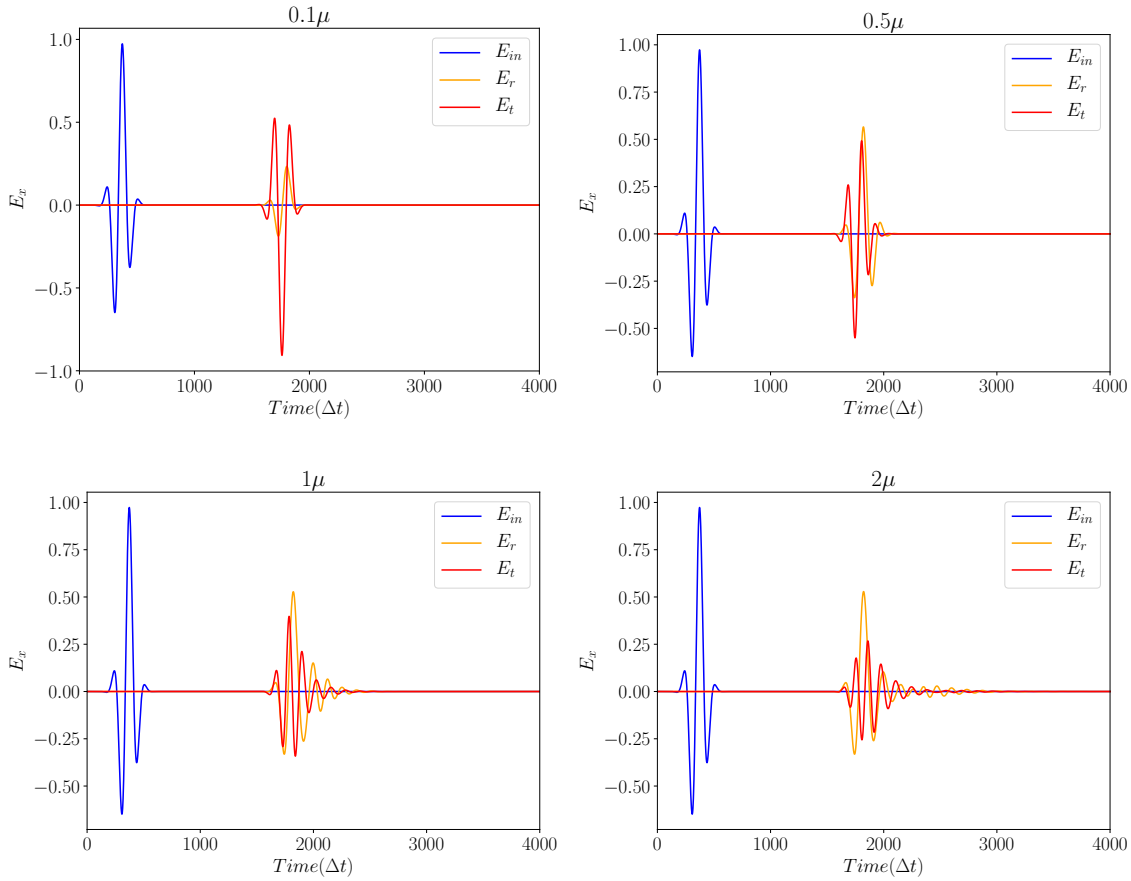

Figure 8. Drude dispersion model for $5, 25, 50, 100\mu m$ dielectric, reflected and transmitted fields.

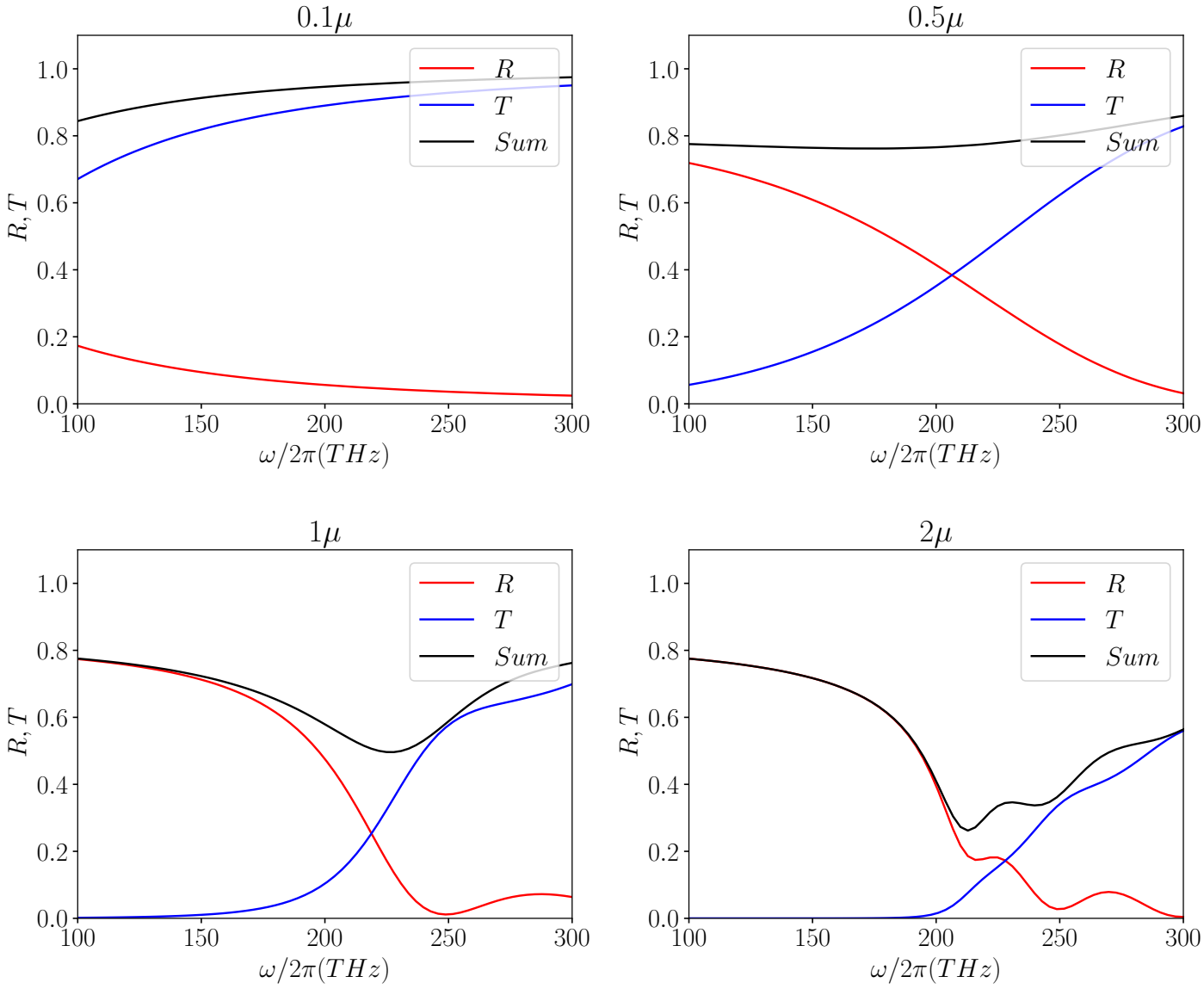

Figure 9. Drude dispersion model for $5, 25, 50, 100\mu m$ dielectric, Fourier transform.

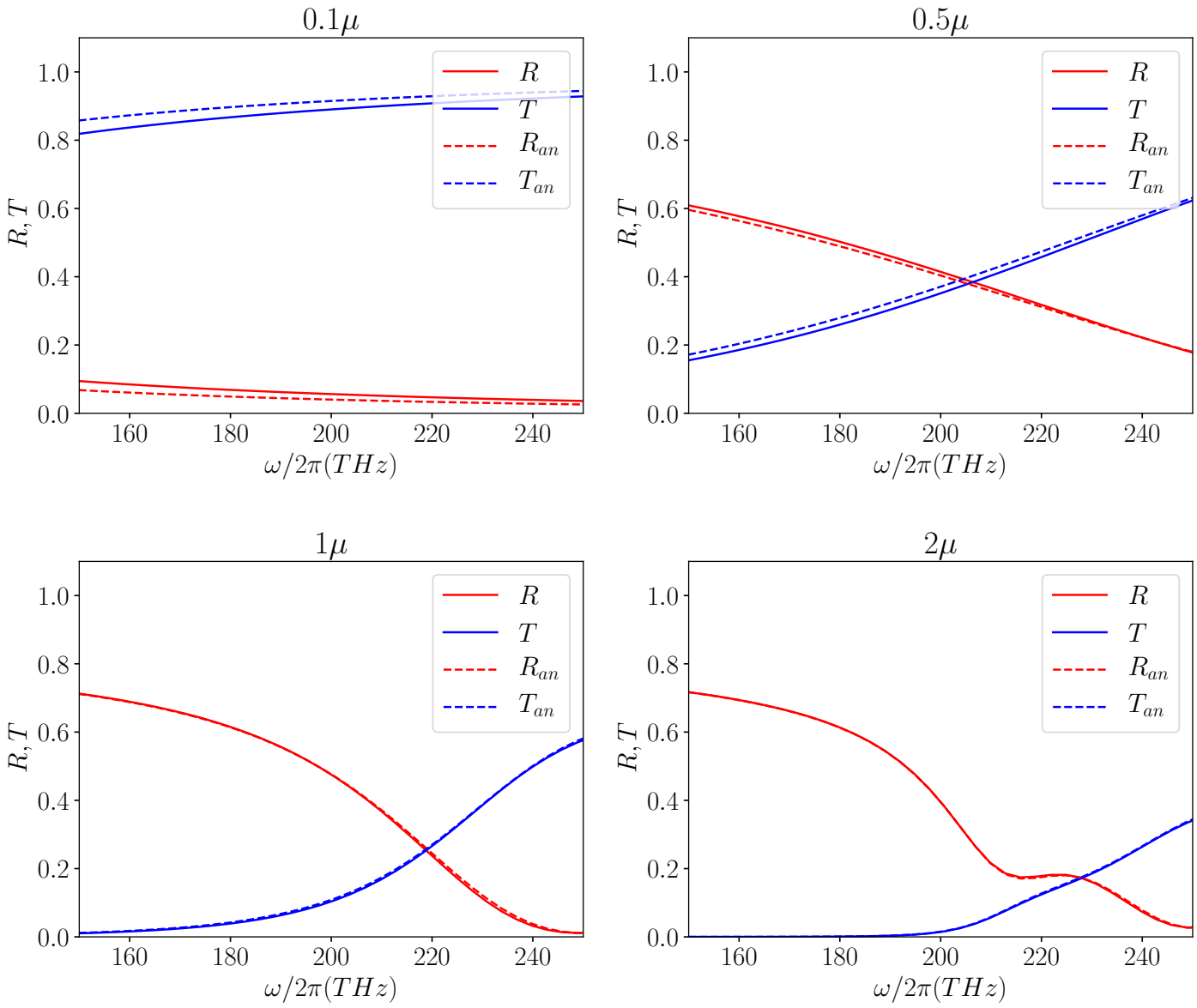

Figure 10. Drude dispersion model for $5, 25, 50, 100\mu m$ dielectric, superimposed analytical solution.

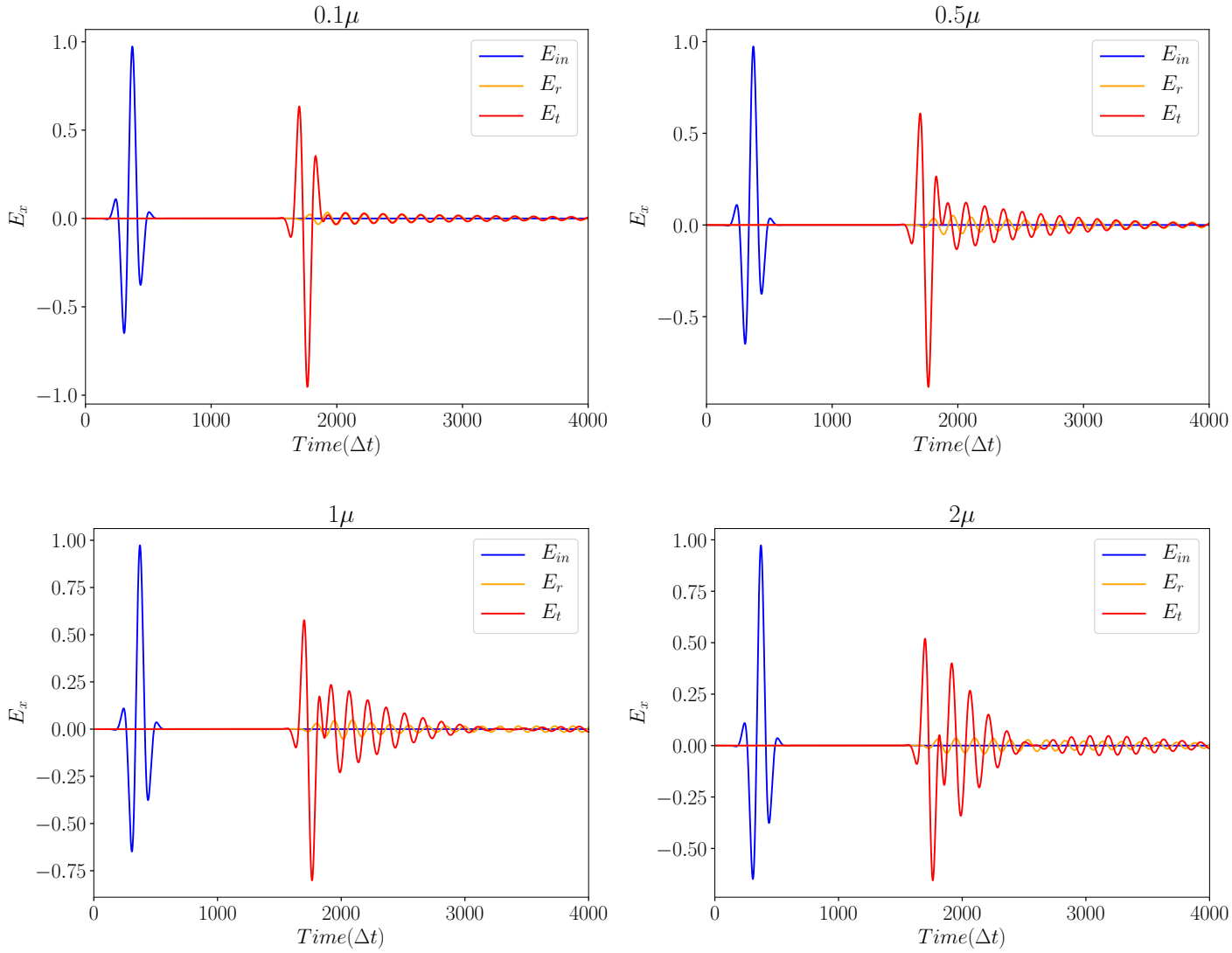

Figure 11. Lorentz dispersion model for $5, 25, 50, 100\mu m$ dielectric, transmitted and reflected fields.

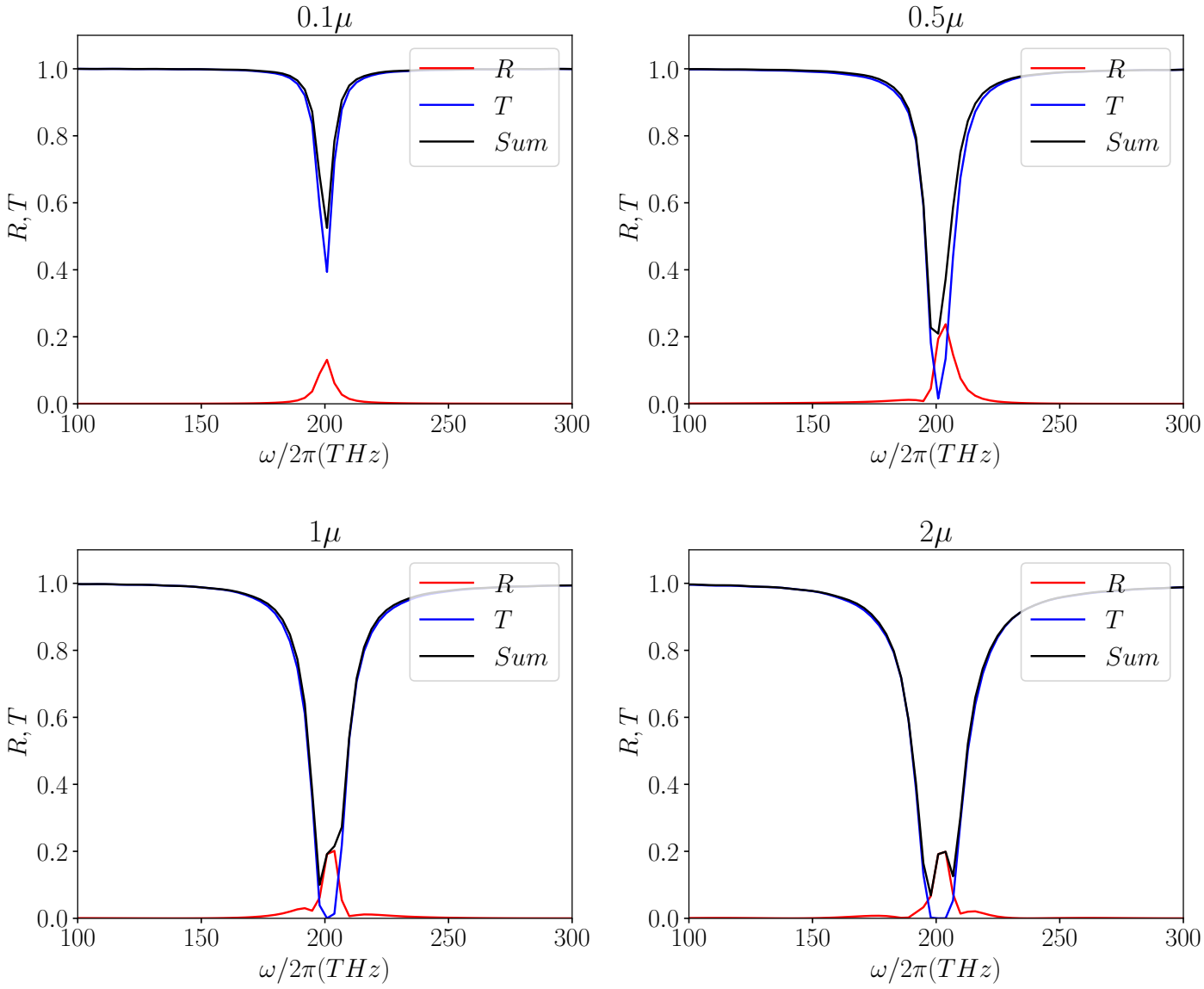

Figure 12. Lorentz dispersion model for $5, 25, 50, 100\mu m$ dielectric, Fourier transforms.

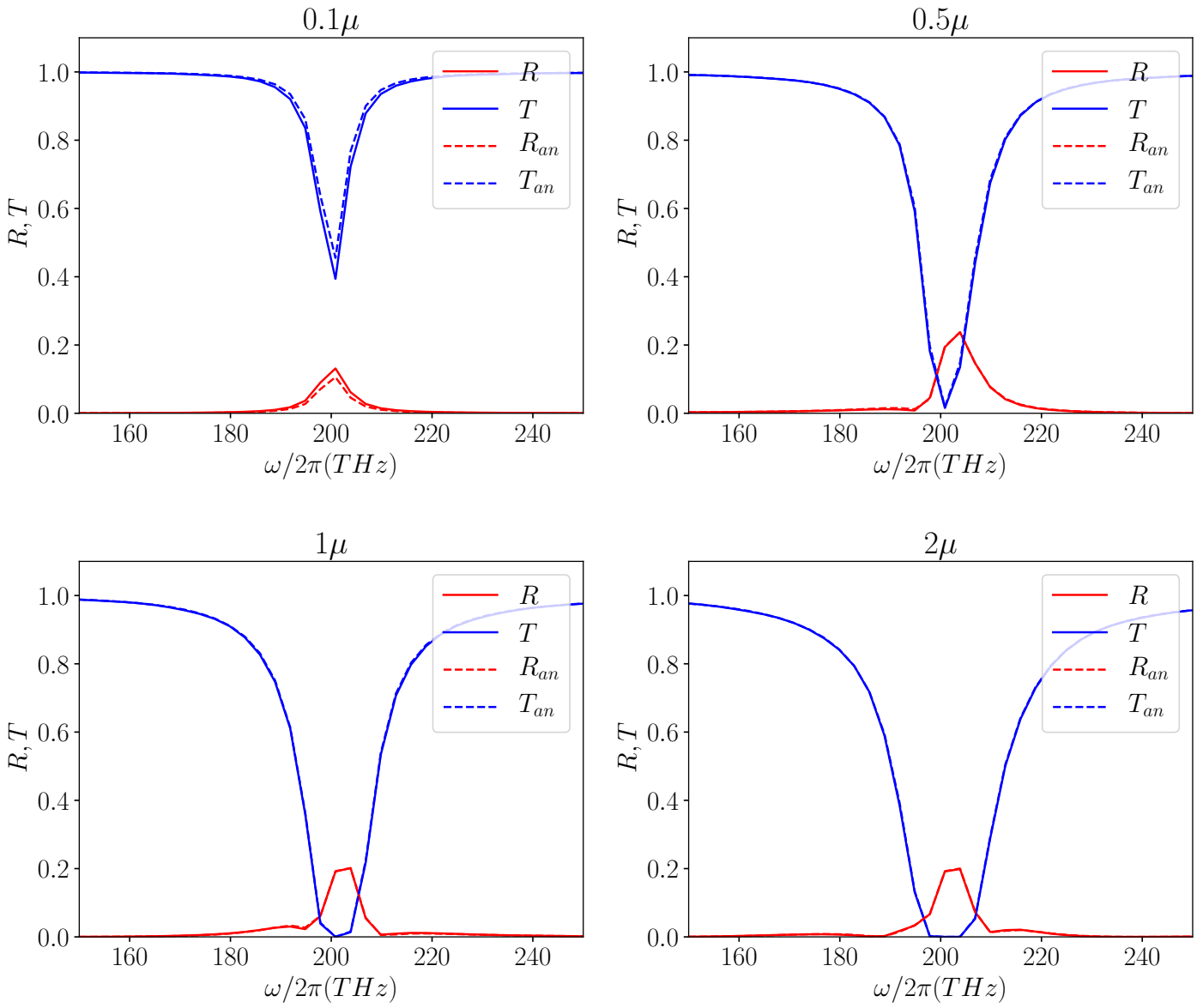

Figure 13. Lorentz dispersion model for $5, 25, 50, 100\mu m$ dielectric, with analytical solution superimposed.

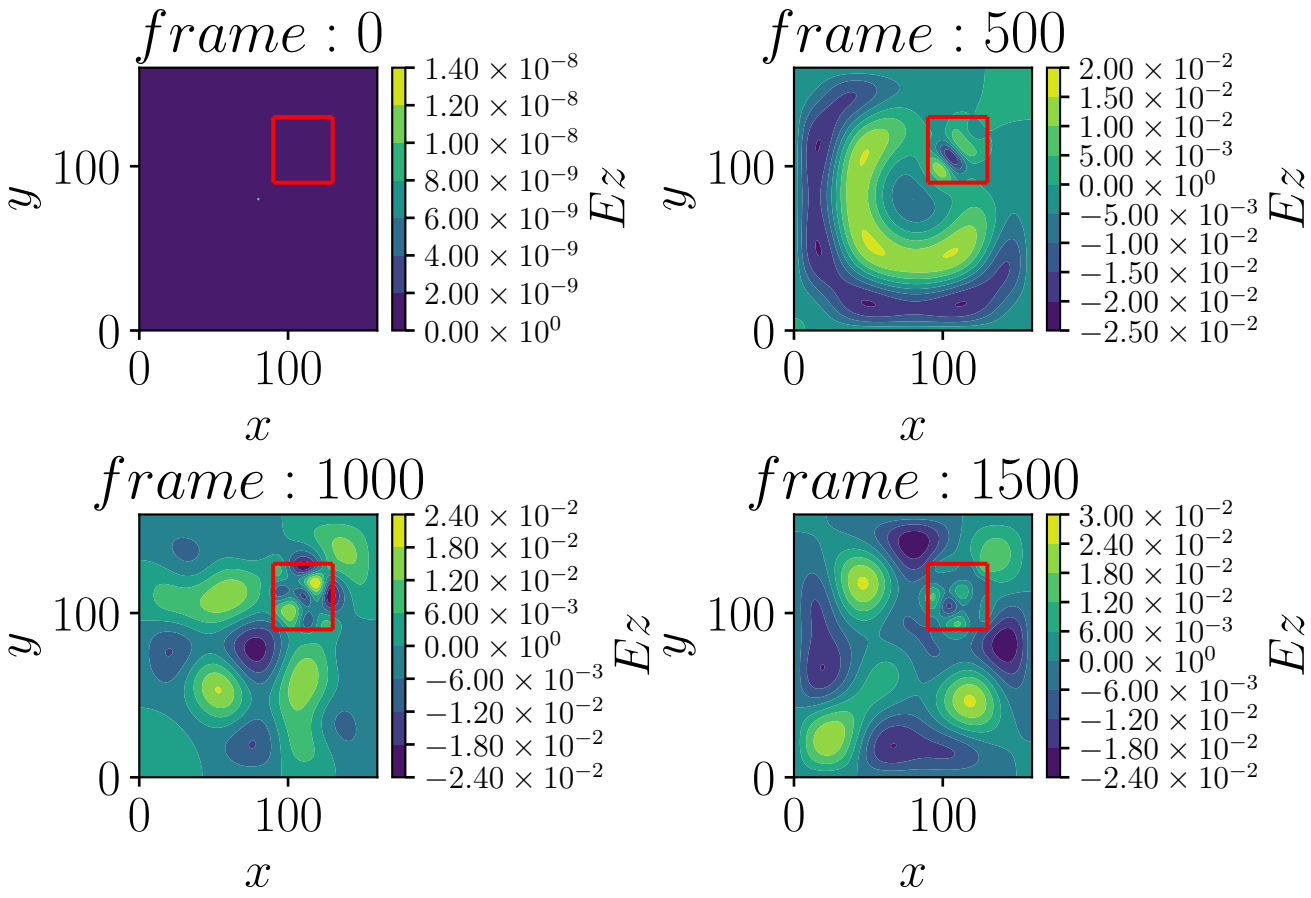

Figure 14. Two dimensional contour map of the electric field.