2

More on One-Dimensional Simulation

Before moving on to two- and three-dimensional problems, we will stay with one-dimensional simulation in order to introduce some advanced concepts. First we will change the formulation slightly and introduce the use of the flux density into the simulation. This may initially seem like an unnecessary complication. However, as we get to frequency-dependent materials in section 2.3, the advantages will become apparent. Then in section 2.2 we introduce the use of the discrete Fourier transform in FDTD simulation. This is an extremely powerful method to quantify the output of the simulation.

It should become very apparent in this chapter how closely signal processing is linked to time-domain EM simulation. This will be most obvious in section 2.4, where we use Z transforms to simulate complicated media.

2.1 REFORMULATION USING THE FLUX DENSITY

Up to now, we have been using the form of Maxwell's equations given in Eq. (1.1), which uses only the E and the H fields. However, a more general form is

$$\frac{\partial \boldsymbol{D}}{\partial t} = \nabla \times \boldsymbol{H} \tag{2.1a}$$

$$\boldsymbol{D}(\omega) = \varepsilon_0 \cdot \varepsilon_r^*(\omega) \cdot \boldsymbol{E}(\omega) \tag{2.1b}$$

$$\frac{\partial \boldsymbol{H}}{\partial t} = -\frac{1}{\mu_0} \nabla \times \boldsymbol{E}, \tag{2.1c}$$

where $\boldsymbol{D}$ is the electric flux density. Notice that Eq. (2.1b) is written in the frequency domain. The reason for this will be explained later. We will begin by normalizing these equations, using

$$\tilde{E} = \sqrt{\frac{\varepsilon_0}{\mu_0}} \cdot E \tag{2.2a}$$

$$\tilde{D} = \sqrt{\frac{1}{\varepsilon_0 \cdot \mu_0}} \cdot D, \tag{2.2b}$$

which leads to

$$\frac{\partial \tilde{\boldsymbol{D}}}{\partial t} = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \nabla \times \boldsymbol{H} \tag{2.3a}$$

$$\tilde{\boldsymbol{D}}(\omega) = \varepsilon_r^*(\omega) \cdot \tilde{\boldsymbol{E}}(\omega) \tag{2.3b}$$

$$\frac{\partial \boldsymbol{H}}{\partial t} = -\frac{1}{\sqrt{\varepsilon_0 \mu_0}} \nabla \times \tilde{\boldsymbol{E}}. \tag{2.3c}$$

We saw in Chapter 1 that this form of Eqs. (2.3a) and (2.3c) will lead to the very simple finite difference equations, Eq. (1.4a) and (1.4b). The only change is the use of D instead of E. However, we still have to get Eq. (2.3b) into a time domain difference equation for implementation into FDTD. The first task is to get it from the frequency domain to the time domain. We will assume we are dealing with a lossy dielectric medium of the form

$$\varepsilon_r^*(\omega) = \varepsilon_r + \frac{\sigma}{j\omega\varepsilon_0} \tag{2.4}$$

and substitute Eq. (2.4) into (2.3b):

$$D(\omega) = \varepsilon_r \cdot E(\omega) + \frac{\sigma}{j\omega\varepsilon_0} E(\omega). \tag{2.5}$$

Taking the first term into the time domain is not a problem because it is simple multiplication. In the second term, Fourier theory tells us that $1/j\omega$ in the frequency domain is integration in the time domain, so Eq. (2.5) becomes

$$D(t) = \varepsilon_r \cdot E(t) + \frac{\sigma}{\varepsilon_0} \int_0^t E(t') \cdot dt'.$$

We will want to go to the sampled time domain, so the integral will be approximated as a summation over the time steps Δt:

$$D^n = \varepsilon_r \cdot E^n + \frac{\sigma \cdot \Delta t}{\varepsilon_0} \sum_{i=0}^{n} E^i. \tag{2.6}$$

Note that E and D are specified at time $t = n \cdot \Delta t$. There is one problem remaining: looking back at Eq. (2.3b), we see that we have to solve for E^n given the value D^n. Unfortunately, the value E^n is needed in the calculation of the summation. We will circumvent this by separating the E^n term from the rest of the summation:

$$D^n = \varepsilon_r \cdot E^n + \frac{\sigma \cdot \Delta t}{\varepsilon_0} E^n + \frac{\sigma \cdot \Delta t}{\varepsilon_0} \sum_{i=0}^{n-1} E^i$$

Now we can calculate E^n from

$$E^n = \frac{D^n - \dfrac{\sigma \cdot \Delta t}{\varepsilon_0} \displaystyle\sum_{i=0}^{n-1} E^i}{\varepsilon_r + \dfrac{\sigma \cdot \Delta t}{\varepsilon_0}}. \tag{2.7}$$

We can calculate E^n, the *current* value of E, from the current value of D and *previous* values of E. It will prove advantageous to define a new parameter for the summation

$$I^n = \frac{\sigma \cdot \Delta t}{\varepsilon_0} \sum_{i=0}^{n} E^i$$

so Eq. (2.7) can be reformulated with the following two equations:

$$E^n = \frac{D^n - I^{n-1}}{\varepsilon_r + \dfrac{\sigma \cdot \Delta t}{\varepsilon_0}} \tag{2.8a}$$

$$I^n = I^{n-1} + \frac{\sigma \cdot \Delta t}{\varepsilon_0} E^n. \tag{2.8b}$$

Note that the summation is calculated by Eq. (2.8b), which, at every time step n, simply adds the value E^n times the constant term to the previous values of the summation at $n - 1$. It is not necessary to store all the value of E^n from 0 to n! Now the entire FDTD formulation is

```
dx[k] = dx[k] + .5*( hy[k-1] - hy[k] )          (2.9a)
ex[k] = gax[k]*( dx[k] - ix[k] )                (2.9b)
ix[k] = ix[k] + gbx[k]*ex[k]                    (2.9c)
hy[k] = hy[k] + .5* ( ex[k] - ex[k+1] )         (2.9d)
```

where

```
gax[k] = 1/(epsilon+(sigma*dt/epsz))            (2.10a)
gbx[k] = sigma*dt/epsz.                         (2.10b)
```

The important point is this: all of the information regarding the media is contained in Eqs. (2.9b) and (2.9c). For free space, `gax = 1 and gbx = 0.`; for lossy material, `gax` and `gbx` are calculated according to Eqs. (2.10a) and (2.10b). In calculating `ex[k]` at the point k, it uses only values of `dx[k]` and previous values of `ex[k]` in the time domain. Equations (2.9a) and (2.9d), which contain the spatial derivatives, do not change regardless of the media!

It may seem as though we have paid a high price for this fancy formulation compared to the formulation of Chapter 1. We now need D_x as well as E_x and an auxiliary parameter ix. The real advantage comes when we deal with more complicated materials, as we'll see in the following sections.

PROBLEM SET 2.1

1. The program fd1d_2.1.c implements the reformulation using the flux density. Get this program running and repeat the results of problem 1.7.1.

2.2 CALCULATING THE FREQUENCY DOMAIN OUTPUT

Up until now, the output of our FDTD programs has been the E field itself, and we have been content to simply watch a pulse or sine wave propagate through various media. Needless to say, before any such practical applications can be implemented, it will be necessary to quantify the results. Suppose now that we are asked to calculate the E field distribution at every point in a dielectric medium subject to illumination at various frequencies. One approach would be to use a sinusoidal source and iterate the FDTD program until we observe that a steady state has been reached, and determine the resulting amplitude and phase at every point of interest in the medium. This would work, but then we must repeat the process for every frequency of interest. System theory tells us that we can get the response to every frequency if we use an impulse as the source. We could go back to using the Gaussian pulse, which, if it is narrow enough, is a good approximation to an impulse. We then iterate the FDTD program until the pulse has died out, and take the Fourier transform of the E fields in the slab. If we have the Fourier transform of the E field at a point, then we know the amplitude and phase of the E field that would result from illumination by any sinusoidal source. This, too, has a very serious

drawback: the E field for all the time domain data at every point of interest would have to be stored until the FDTD program is through iterating so the Fourier transform of the data could be taken, presumably using a fast Fourier transform algorithm. This presents a logistical nightmare.

Here is an alternative. Suppose we want to calculate the Fourier transform of the E field $E(t)$ at a frequency f_1. This can be done by the equation

$$E(f_1) = \int_0^{t_T} E(t) \cdot e^{-j2\pi f_1 t}\, dt. \tag{2.11}$$

Notice that the lower limit of the integral is 0 because the FDTD program assumes all causal functions. The upper limit is t_T, the time at which the FDTD iteration is halted. Rewriting (2.11) in a finite difference form,

$$E(f_1) = \sum_{n=0}^{T} E(n \cdot \Delta t) \cdot e^{-j2\pi f_1 (n \cdot \Delta t)}, \tag{2.12}$$

where T is the number of iterations and Δt is the time step, so $t_T = T \cdot \Delta t$. Equation (2.12) may be divided into its real and imaginary parts

$$\begin{aligned} E(f_1) = &\sum_{n=0}^{T} E(n \cdot \Delta t) \cdot \cos(2\pi f_1 \cdot \Delta t \cdot n) \\ &- j \sum_{n=0}^{T} E(n \cdot \Delta t) \cdot \sin(2\pi f_1 \cdot \Delta t \cdot n), \end{aligned} \tag{2.13}$$

which may be implemented in computer code by

```
real_pt[m,k] = real_pt[m,k] + ex[k]*cos(2*pi*freq(m)*dt*n)      (2.14a)
imag_pt[m,k] = imag_pt[m,k] + ex[k]*sin(2*pi*freq(m)*dt*n).     (2.14b)
```

For every point k, in the region of interest, we require only two buffers for every frequency of interest f_m. At any point k, from the real part of $E(f_i)$, `real_pt[m,k]`, and the imaginary part `imag_pt[m,k]`, we can determine the amplitude and phase at the frequency f_m:

```
amp[m,k] = sqrt(pow(real_pt[m,k],2.)+pow(imag_pt[m,k],2.))      (2.15a)
phase[m,k] = atan2(imag_pt[m,k],real_pt[m,k]).                  (2.15b)
```

Note that there is an amplitude and phase associated with every frequency at each cell [1, 2]. The program fd1d_2.2.c at the end of the chapter calculates the frequency response at three frequencies throughout the problem space. Figure 2.1 is a simulation of a pulse hitting a dielectric medium with a dielectric constant of 4, similar to Fig. 1.4. The frequency response at 500 MHz is also displayed. At $T = 200$, before the pulse has hit the medium, the frequency response is 1 through that part of the space where the pulse has traveled. After 400 time steps, the pulse has hit the medium, and some of it has penetrated into the medium and some of it has been reflected. The amplitude of the transmitted pulse is determined by Eq. (1.A.2) at the end of Chapter 1,

$$\tau = \frac{\sqrt{4} \cdot 1}{1 + \sqrt{4}} = .667,$$

which is the Fourier amplitude in the medium. The Fourier amplitude outside the medium varies between $1 - .333$ and $1 + .333$. This is in keeping with the pattern formed by the standing wave that is created from a sinusoidal signal, whose reflected wave is interacting with the original incident wave.

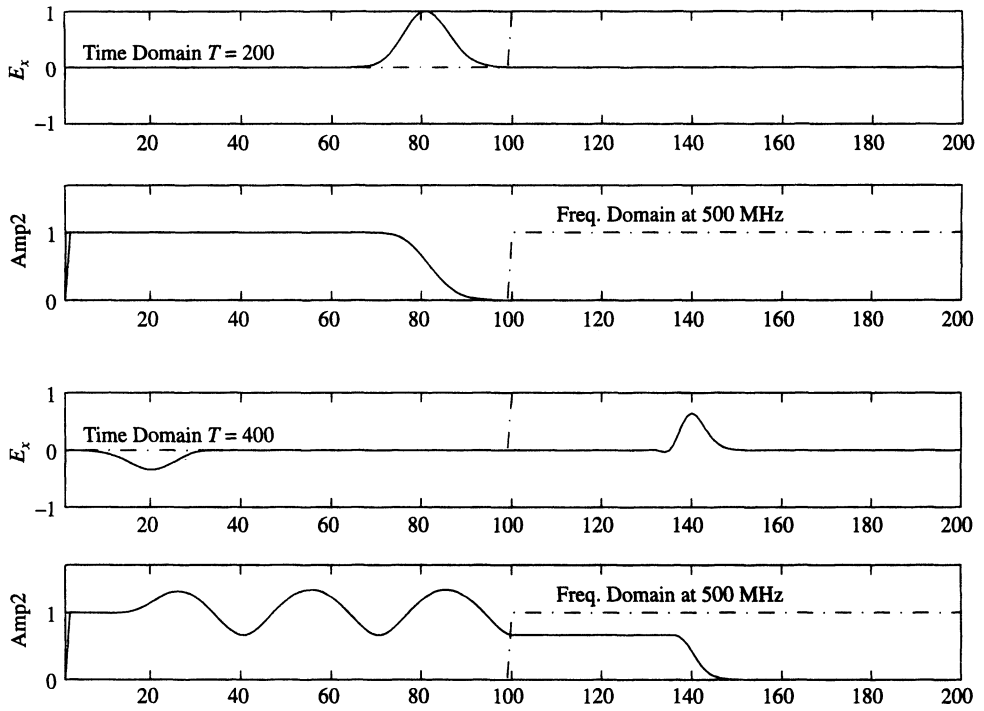

Figure 2.1 Simulation of a pulse striking a dielectric medium with $\varepsilon_r = 4$. The top figure is the pulse after 200 time steps. Notice that the Fourier amplitude is 1 in that part of the space where the pulse has traveled, but 0 elsewhere. After 400 time steps, the pulse has struck the medium, and part of it has been transmitted and part reflected. The Fourier amplitude in the medium is 0.667, which is the percentage that has been transmitted.

PROBLEM SET 2.2

1. The program fd1d_2.2.c implements the discrete Fourier transform with a Gaussian pulse as its source. Get this program running. Duplicate the results in Fig. 2.1.

2.3 FREQUENCY-DEPENDENT MEDIA

The dielectric constant and conductivity of most media vary at different frequencies. The pulses we have been using as a source in Chapters 1 and 2 contain a spectrum of frequencies. In order to simulate frequency-dependent material, we will need a way to account for this. One of the most significant developments in the FDTD method was a means to simulate frequency-dependent materials [3].

We will start with a very simple example to illustrate the ideas. Suppose we have a medium whose dielectric constant and conductivity vary over the frequency range of 10 to 1000 MHz as shown in Fig. 2.2. A material like this can be adequately represented by the following formulation:

$$\varepsilon_r^*(\omega) = \varepsilon_r + \frac{\sigma}{j\omega\varepsilon_0} + \frac{\chi_1}{1 + j\omega t_0}. \tag{2.16}$$

This is referred to as the Debye formulation. In this formulation, there is a dielectric constant ε_r and a conductivity σ, but there is also a frequency-dependent term. The following parameters

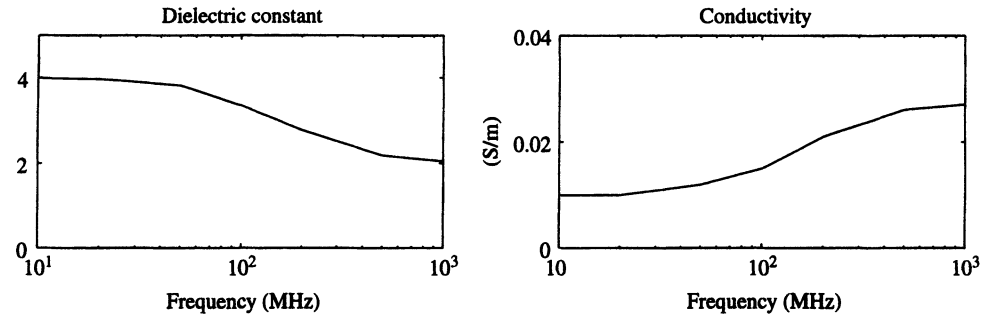

Figure 2.2 Relative dielectric constant and conductivity as functions of frequency for a Debye medium with the following properties: $\varepsilon_r = 2$, $\sigma = .1$, $\chi = 2$, $\tau = .001$ μsec.

represent the medium of Fig. 2.2:

$$\varepsilon_r = 2, \quad \sigma = .01, \quad \chi_1 = 2, \quad t_0 = .001 \ \mu \text{ sec}.$$

In order to simulate this medium in FDTD, Eq. (2.16) must be put into the sampled time domain. Let us define the last term times the E field as

$$S(\omega) = \frac{\chi_1}{1 + j\omega t_0} E(\omega). \tag{2.17}$$

The inverse Fourier transform of the Debye term is $(\chi_1/t_0)e^{-(t/t_0)}u(t)$, where $u(t)$ is the Heavyside, or rectangular function, which is 0 for $t < 0$ and 1 thereafter. (Remember we said that implicit to FDTD is the fact that all functions are causal, i.e., zero for $t < 0$, because our computer programs initialize the field values to zero.) Equation (2.17) in the frequency domain becomes the convolution

$$S(t) = \frac{\chi_1}{t_0} \int_0^t e^{-(t'-t)/t_0} E(t') \cdot dt'$$

in the time domain. We now have to approximate this as a summation in the sampled time domain:

$$\begin{aligned} S^n &= \chi_1 \cdot \frac{\Delta t}{t_0} \sum_{i=0}^{n} e^{-\cdot \Delta t(n-i)/t_0} \cdot E^i \\ &= \chi_1 \cdot \frac{\Delta t}{t_0} \left(E^n + \sum_{i=0}^{n-1} e^{-\Delta t(n-i)/t_0} \cdot E^i \right). \end{aligned} \tag{2.18}$$

Notice that

$$\begin{aligned} S^{n-1} &= \chi_1 \cdot \frac{\Delta t}{t_0} \sum_{i=0}^{n-1} e^{-\Delta t(n-1-i)/t_0} \cdot E^i \\ &= \chi_1 \cdot \frac{\Delta t}{t_0} e^{\Delta t/t_0} \sum_{i=0}^{n-1} e^{-\Delta t(n-i)/t_0} \cdot E^i. \end{aligned}$$

Substituting this value into Eq. (2.18) above gives

$$S^n = \chi_1 \cdot \frac{\Delta t}{t_0} \cdot E^n + e^{-\cdot \Delta t/t_0} S^{n-1}. \tag{2.19}$$

Similar to the way we handled the lossy dielectric, we can write

$$\begin{aligned} D^n &= \varepsilon_r \cdot E^n + I^n + S^n \\ &= \varepsilon_r \cdot E^n + \left[\frac{\sigma \cdot \Delta t}{\varepsilon_0} \cdot E^n + I^{n-1}\right] + \left[\chi_1 \cdot \frac{\Delta t}{t_0} \cdot E^n + e^{-t/t_0} \cdot S^{n-1}\right], \end{aligned} \tag{2.20}$$

and solving for E^n

$$E^n = \frac{D^n - I^{n-1} - e^{-\Delta t/t_0} S^{n-1}}{\varepsilon_r + \dfrac{\sigma \cdot \Delta t}{\varepsilon_0} + \chi_1 \cdot \dfrac{\Delta t}{t_0}} \tag{2.21a}$$

$$I^n = I^{n-1} + \frac{\sigma \cdot \Delta t}{\varepsilon_0} \cdot E^n \tag{2.21b}$$

$$S^n = e^{-\Delta t/t_0} S^{n-1} + \chi_1 \cdot \frac{\Delta t}{t_0} \cdot E^n. \tag{2.21c}$$

This formulation is implemented by the following computer code:

```
dx[k]= dx[k] + .5*( hy[k-1] - hy[k] )                    (2.22a)
ex[k] = gax[k]*( dx[k] - ix[k] - del_exp*sx[k] )          (2.22b)
ix[k] = ix[k] + gbx[k]*ex[k]                             (2.22c)
sx[k] = del_exp*sx[k] + gcx[k]*ex[k]                     (2.22d)
hy[k] = hy[k] + .5* ( ex[k] - ex[k+1] )                  (2.22e)
```

where

```
gax[k] = 1/(epsr + (sigma*dt/epsz) + (chi1*dt/t0) )      (2.23a)
gbx[k] = sigma*dt/epsz                                   (2.23b)
gbc[k]= chi1*dt/t0                                       (2.23c)
```

and

```
del_exp = exp(-dt/t0).
```

Once again, note that everything concerning the medium is contained in Eqs. (2.22b) through Eq. (2.22d); Equations (2.22a) and (2.22e), the calculation of the flux density and magnetic field, are unchanged.

The program fd1d_2.3.c calculates the frequency domain amplitude and phase for three frequencies. Figure 2.3 shows a simulation of a pulse going into a frequency-dependent dielectric material with the properties

$$\varepsilon_r = 2, \quad \sigma = .01 \quad , \chi_1 = 2, \quad t_0 = .001\ \mu \text{ sec}.$$

This set of parameters has the following effective dielectric constants and conductivities at the three frequencies:

Frequency (MHz)	ε_r	σ (S/m)
50	6.55	.024
200	3.94	.047
500	2.46	.06

Notice that the Fourier amplitude attenuates more rapidly at 200 MHz than at 50 MHz, and more rapidly still at 500 MHz. This is because the conductivity is higher at these frequencies. At the same time, the higher relative dielectric constant at the lower frequencies means the amplitude just inside the medium is smaller.

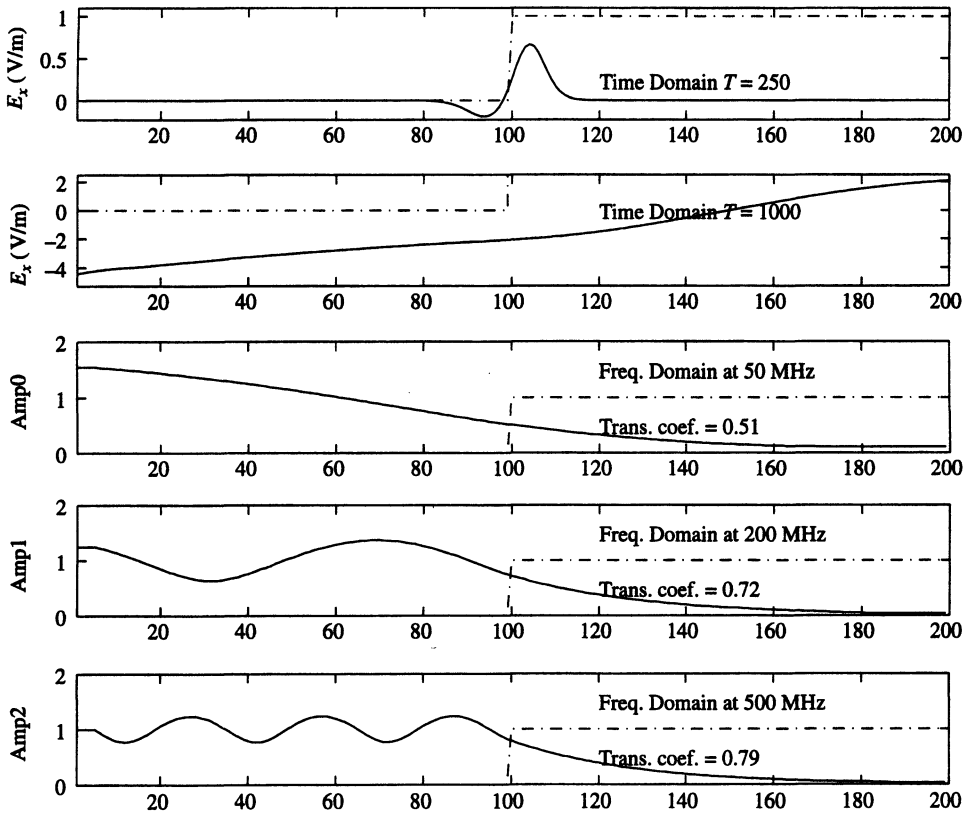

Figure 2.3 Simulation of a pulse striking a frequency-dependent dielectric medium (Debye medium) with the following properties: $\varepsilon_r = 2$, $\sigma = 0.01$, $\chi_1 = 2$, $t_0 = .001$ μsec. After 250 time steps (top), the pulse has struck the medium, and part of it has been transmitted and part reflected. After 1000 steps (second from top) the pulse has penetrated into the medium, but has spread. Notice the different percentages of transmittance and the different rates of attenuation within the medium at each frequency due to the different effective dielectric constants and conductivities.

2.3.1 Auxiliary Differential Equation Method

We could have taken a somewhat different approach to the simulation of the dispersive medium described by Eq. (2.16). There is another method, usually referred to as the auxiliary differential equation (ADE) method [4, 5]. Let's look at Eq. (2.17), but rewrite it in the following manner:

$$(1 + j\omega t_0)\, S(\omega) = \chi_1 E(\omega). \tag{2.24}$$

Once again, we must find a way to take this to the discrete time domain for implementation in the FDTD formulation. We will start by going to the continuous time domain, so (2.24) becomes

$$s(t) + t_0 \frac{d\, s(t)}{dt} = \chi_1 e(t). \tag{2.25}$$

In the sampled time domain, this becomes

$$\frac{S^n + S^{n-1}}{2} + t_0 \frac{S^n - S^{n-1}}{\Delta t} = \chi_1 E^n.$$

Notice that we approximated the $s(t)$ term over two time steps. We did this because we needed two time steps to approximate the derivative. As before, we solve for S^n:

$$S^n = \frac{\left(1 - \frac{\Delta t}{2 \cdot t_0}\right)}{\left(1 + \frac{\Delta t}{2 \cdot t_0}\right)} S^{n-1} + \frac{\left(\frac{\Delta t}{t_0}\right) \cdot \chi_1 \cdot E^n}{\left(1 + \frac{\Delta t}{2 \cdot t_0}\right)}. \tag{2.26}$$

We can use this instead of Eq. (2.19) to calculate E^n in Eq. (2.21). How can we be sure these will give equivalent answers? You are probably familiar with the following approximations:

$$1 - \delta \cong e^{-\delta} \quad if\ \delta << 1$$

$$\frac{1}{1+\delta} \cong e^{-\delta} \quad if\ \delta << 1.$$

Putting the two together gives

$$\frac{1-\delta}{1+\delta} \cong e^{-2\delta} \quad if\ \delta << 1.$$

In this case

$$\delta = \frac{\Delta t}{2 \cdot t_0},$$

so we have

$$\frac{\left(1 - \frac{\Delta t}{2 \cdot t_0}\right)}{\left(1 + \frac{\Delta t}{2 \cdot t_0}\right)} \cong e^{-\Delta t / t_0}. \tag{2.27}$$

This leaves only the following question: How do we know that $\Delta t / t_0$ is small enough? Recall earlier that the cell size has to be small enough to get about ten points per wavelength for the smallest wavelength in the simulation. This is a similar situation. If the medium that we are trying to simulate has a Debye term with a time constant of t_0, we must be sure that our time steps are small compared to t_0, say $\Delta t \leq t_0/10$. This insures that Eq. (2.27) is a fairly good approximation.

PROBLEM SET 2.3

1. The program fd1d_2.3.c implements the frequency-dependent formulation. Get this program running and repeat the results of Fig. 2.3.

2.4 FORMULATION USING Z TRANSFORMS

If you've been studying the Z transform theory in the appendix, or if you are already familiar with Z transforms, you will now see the advantage of using Z transforms for the FDTD formulation of frequency-dependent media [6]. Returning to the problem of calculating E in a Debye media, we begin with our frequency domain equation

$$D(\omega) = \left(\varepsilon_r + \frac{\sigma}{j\omega\varepsilon_0} + \frac{\chi_1}{1 + j\omega t_0}\right) \cdot E(\omega).$$

We can avoid dealing with troublesome convolution integrals in the time domain because we will go immediately to the Z domain,

$$D(z) = \varepsilon_r \cdot E(z) + \frac{\sigma \cdot \Delta t/\varepsilon_0}{1 - z^{-1}} \cdot E(z) + \frac{\chi_1 \cdot \Delta t / t_0}{1 + z^{-1}} \cdot E(z). \tag{2.28}$$

Note that the factor Δt, which is the time step, had to be added to the last two terms in going from the time domain to the Z domain. (See section A.2 in the appendix at the end of book). Similar to what we did above, we will define some auxiliary parameters:

$$I(z) = \frac{\sigma \cdot \Delta t/\varepsilon_0}{1 - z^{-1}} \cdot E(z) = z^{-1} I(z) + \frac{\sigma \cdot \Delta t}{\varepsilon_0} E(z) \tag{2.29a}$$

$$S(z) = \frac{\chi_1 \cdot \Delta t/t_0}{1 - e^{-\Delta t/t_0} z^{-1}} \cdot E(z) = e^{-\Delta t/t_0} z^{-1} S(z) + \frac{\chi_1 \cdot \Delta t}{t_0} E(z). \tag{2.29b}$$

Equation (2.28) then becomes

$$\begin{aligned} D(z) = \varepsilon_r \cdot E(z) \; + \; z^{-1} I(z) \; + \; & \frac{\sigma \cdot \Delta t}{\varepsilon_0} E(z) \\ + \; e^{-\Delta t/t_0} z^{-1} S(z) \; + \; & \frac{\chi_1 \cdot \Delta t}{t_0} E(z), \end{aligned} \tag{2.30}$$

from which we can solve for $E(z)$ by

$$E(z) = \frac{D(z) - z^{-1} I(z) - e^{-\Delta t/t_0} z^{-1} S(z)}{\varepsilon_r + \dfrac{\sigma \cdot \Delta t}{\varepsilon_0} + \dfrac{\chi_1 \cdot \Delta t}{t_0}}. \tag{2.31}$$

Here is the advantage of the Z transform formulation: to get to the sampled time domain, replace $E(z)$ with E^n, $z^{-1}E(z)$ with E^{n-1}, and make a similar replacement with the other parameters in Eqs. (2.31), (2.29a), and (2.29b). What you get is

$$E^n = \frac{D^n - I^{n-1} - e^{-\Delta t/t_0} S^{n-1}}{\varepsilon_r + \dfrac{\sigma \cdot \Delta t}{\varepsilon_0} + \dfrac{\chi_1 \cdot \Delta t}{t_0}} \tag{2.32a}$$

$$I^n = I^{n-1} + \frac{\sigma \cdot \Delta t}{\varepsilon_0} E^n \tag{2.32b}$$

$$S^n = e^{-\Delta t/t_0} S^{n-1} + \frac{\chi_1 \cdot \Delta t}{t_0} E^n, \tag{2.32c}$$

which is *exactly* what we got in the previous section. The difference is, we didn't have to do anything with integrals and their approximations. As we move to formulations that are more complicated, the advantage of the Z transform will become evident.

2.4.1 Simulation of an Unmagnetized Plasma

In this section, we will demonstrate the versatility of the methods we have learned in this chapter by simulating a completely different medium from those we have been working with so far. The permittivity of unmagnetized plasmas is given as [7]

$$\varepsilon^*(\omega) = 1 + \frac{\omega_p^2}{\omega(j\nu_c - \omega)}, \tag{2.33}$$

where $\omega_p = 2\pi f_p$
f_p is the plasma frequency
ν_c is the electron collision frequency.

By using partial fraction expansion, Eq. (2.33) can be written as

$$\varepsilon^*(\omega) = 1 + \frac{\omega_p^2/\nu_c}{j\omega} - \frac{\omega_p^2/\nu_c}{\nu_c + j\omega}. \tag{2.34}$$

This is the value we will use for the complex dielectric constant in Eq. (2.3b). Notice that the form resembles Eq. (2.16), the expression for a lossy material with a Debye term. There are several ways of approaching this problem, but let's start by taking the Z transform of (2.34) to obtain

$$\varepsilon^*(z) = \frac{1}{\Delta t}1 + \frac{\omega_p^2/\nu_c}{1 - z^{-1}} - \frac{\omega_p^2/\nu_c}{1 - e^{-\nu_c \cdot \Delta t} z^{-1}}. \tag{2.35}$$

By the convolution theorem, the Z transform of Eq. (2.3b) is

$$D(z) = \varepsilon^*(z) \cdot E(z) \cdot \Delta t. \tag{2.36}$$

By inserting Eq. (2.35) into (2.36) we obtain

$$\begin{aligned} D(z) &= E(z) + \frac{\omega_p^2 \Delta t}{\nu_c}\left[\frac{1}{1 - z^{-1}} - \frac{1}{1 - e^{-\nu_c \cdot \Delta t} z^{-1}}\right] E(z) \\ &= E(z) + \frac{\omega_p^2 \Delta t}{\nu_c}\left[\frac{(1 - e^{-\nu_c \cdot \Delta t}) z^{-1}}{1 - (1 - e^{-\nu_c \cdot \Delta t}) z^{-1} + e^{-\nu_c \cdot \Delta t} z^{-2}}\right] E(z). \end{aligned}$$

Notice that we cross multiplied the term in the brackets. An auxiliary term will be defined as

$$S(z) = \frac{\omega_p^2 \Delta t}{\nu_c}\left[\frac{(1 - e^{-\nu_c \cdot \Delta t})}{1 - (1 - e^{-\nu_c \cdot \Delta t}) z^{-1} + e^{-\nu_c \cdot \Delta t} z^{-2}}\right] E(z).$$

$E(z)$ can be solved for by

$$E(z) = D(z) - z^{-1} S(z) \tag{2.37a}$$

$$S(z) = (1 + e^{-\nu_c \cdot \Delta t}) z^{-1} S(z) - e^{-\nu_c \cdot \Delta t} z^{-2} S(z) + \frac{\omega_p^2 \Delta t}{\nu_c}(1 - e^{-\nu_c \cdot \Delta t}) E(z). \tag{2.37b}$$

Therefore, the FDTD simulation becomes

```
ex[k] = dx[k] - sx[k];                                          (2.38a)
sx[k] = (1 + exp(-vc*dt)*sxm1[k]
      - exp(-vc*dt)*sxm2[k]
      + (pow(omega,2.)*dt/vc)*(1-exp(-vc*dt))*ez[k];            (2.38b)
sxm2[k] = sxm1[k];                                              (2.38c)
sxm1[k] = sx[k];                                                (2.38d)
```

Notice from Eq. (2.38b) that we need the two previous values of S in our calculation. This is accomplished by Eqs. (2.38c) and (2.38d).

We will do a simulation of a pulse propagating in free space that comes upon a plasma. Plasma is a very interesting medium. At relatively low frequencies, it looks like a metal, and at higher frequencies, it becomes transparent. By lower or higher frequencies, I mean below or above the plasma resonance frequency f_p. This simulation will use the properties of silver: $\nu_c = 57$ THz, $f_p = 2000$ THz. Of course, since we are simulating much higher frequencies, we will need a much smaller cell size. In the course of this problem, it will be necessary to simulate EM waves of 4000 THz (tera => 10^{12}). At this frequency, the free space wavelength is

$$\lambda = \frac{3 \times 10^8}{4 \times 10^{15}} = 0.75 \times 10^{-7}.$$

In following our rule of thumb of at least ten points per wavelength, a cell size of one nanometer, i.e., $\Delta x = 10^{-9}$, will be used. Figure 2.4 shows our first simulation at 500 THz, well below the plasma frequency. The incident pulse is a sine wave inside a Gaussian envelope. Notice that it interacts with the plasma almost as if it were a metal barrier and is almost completely reflected. Figure 2.5 is a similar simulation at 4000 THz, well above the plasma frequency. A small portion of it is reflected, but the majority of the pulse passes right through.

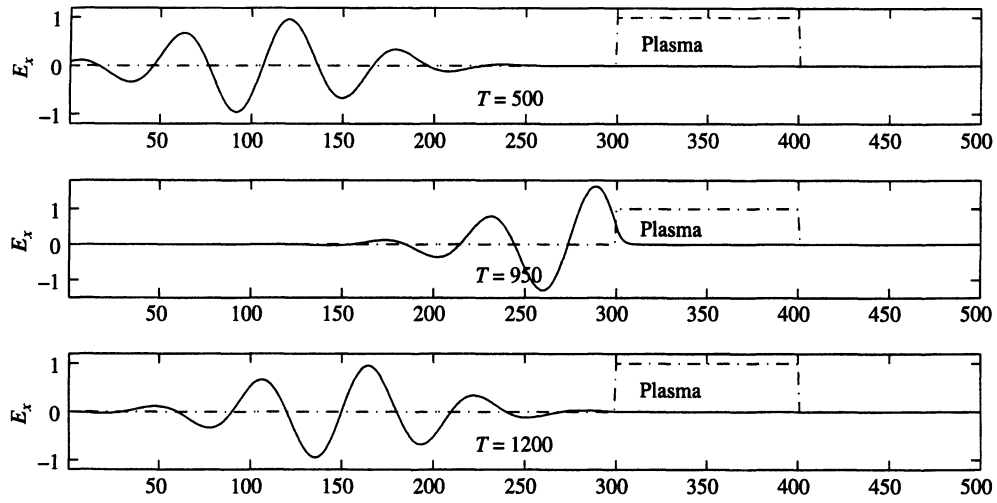

Figure 2.4 Simulation of a wave propagating in free space and striking a plasma medium. The plasma has the properties of silver; $f_p = 2000$ THz and $\nu_c = 57$ THz. the propagating wave has a center frequency of 500 THz. After 1200 time steps, it has been completely reflected by the plasma.

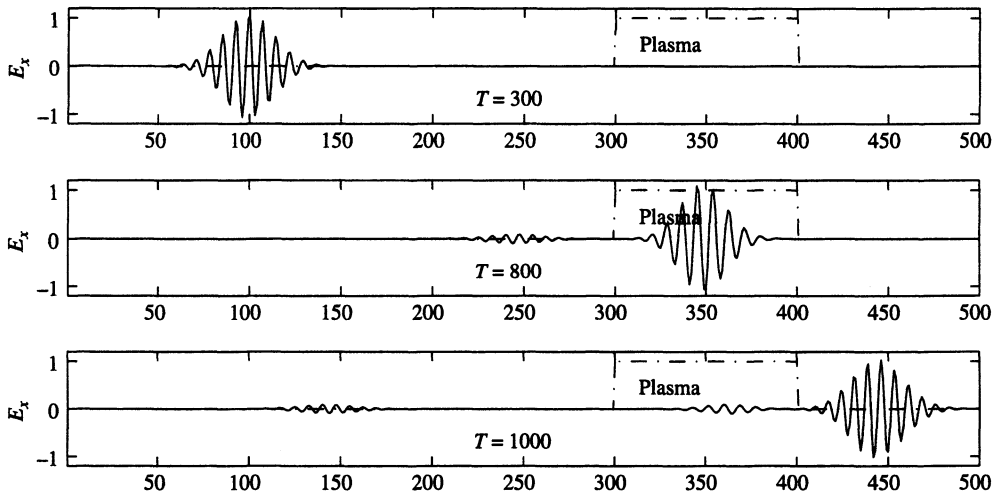

Figure 2.5 Simulation of a wave propagating in free space and striking a plasma medium. The plasma has the properties of silver; $f_p = 2000$ THz and $\nu_c = 57$ THz. The propagating wave has a center frequency of 4000 THz. After 1000 time steps, it has completely passed through the plasma.

PROBLEM SET 2.4

1. Modify the program fd1d_2.3.c to simulate plasma and duplicate the results of Figs. 2.4 and 2.5. This is much easier than it might look. First change your cell size to 1 nanometer. Then replace the calculation of the E field for a lossy Debye medium with that of Eqs. (2.38). After doing the simulations at 500 THz and 4000 THz, repeat at 2000 THz. What happens?
2. Repeat problem 2.4.1, but use a narrow Gaussian pulse as your input and calculate the frequency response at 500, 2000, and 4000 THz. Does the result look the way you would expect, particularly at 2000 THz?

2.5 FORMULATING A LORENTZ MEDIUM

The Debye model described a single-pole frequency dependence. We move now to the next level, which is a two-pole dependence referred to as a Lorentz formulation:

$$\varepsilon_r^*(\omega) = \varepsilon_r + \frac{\varepsilon_1}{1 + j2\delta_0\left(\frac{\omega}{\omega_0}\right) - \left(\frac{\omega}{\omega_0}\right)^2}. \tag{2.39}$$

Figure 2.6 is a graph of the dielectric constant and conductivity of a material with the following Lorentz parameters: $\varepsilon_r = 2$, $\varepsilon_1 = 2$, $f_0 = 100$ MHz, $\delta = .25$ ($\omega_0 = 2\pi f_0$). To simulate a Lorentz medium in the FDTD formulation, we first put Eq. (2.39) into Eq. (2.3b),

$$\begin{aligned} D(\omega) &= \varepsilon_r E(\omega) + \frac{\varepsilon_1}{1 + j2\delta_0\left(\frac{\omega}{\omega_0}\right) - \left(\frac{\omega}{\omega_0}\right)^2} E(\omega) \\ &= \varepsilon_r E(\omega) + S(\omega), \end{aligned} \tag{2.40}$$

where we have again defined an auxiliary term,

$$S(\omega) = \frac{\omega_0^2 \varepsilon_1}{\omega_0^2 + j\omega 2\delta_0\omega_0 + (j\omega)^2} E(\omega).$$

We will use the ADE method to move this to the time domain. Start by rewriting it in the following manner:

$$\left(\omega_0^2 + j\omega 2\delta_0\omega_0 + (j\omega)^2\right) S(\omega) = \omega_0^2 \varepsilon_1 E(\omega).$$

Finally, we proceed to the finite difference approximations

$$\omega_0^2 S^{n-1} + 2\delta_0\omega_0 \frac{S^n - S^{n-2}}{2\Delta t} + \frac{S^n - 2S^{n-1} + S^{n-2}}{\Delta t^2} = \omega_0^2 \varepsilon_1 E^{n-1}.$$

A few things are worth noting: the second-order derivative generated a second-order differencing

$$\frac{d^2 s(t)}{dt^2} \cong \frac{S^n - 2S^{n-1} + S^{n-2}}{\Delta t^2}.$$

The first-order equation is taken over two time steps instead of one:

$$\frac{ds(t)}{dt} \cong \frac{S^n - S^{n-2}}{2\Delta t}.$$

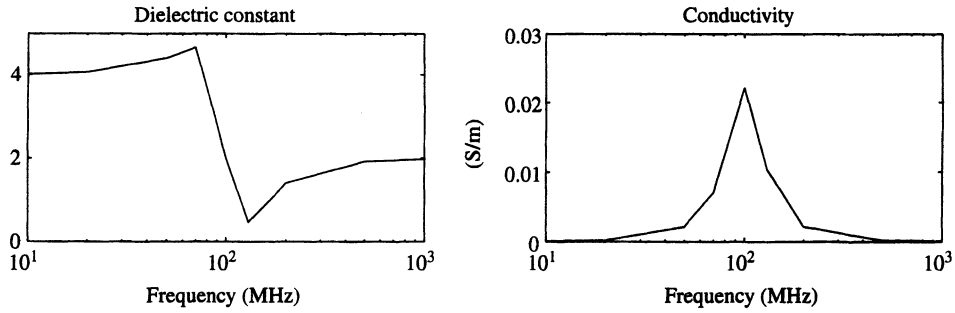

Figure 2.6 Relative dielectric constant and conductivity as functions of frequency for a Lorentz medium with the following properties:

$$\varepsilon_r = 2, \varepsilon_1 = 2, f_0 = 100 \text{ MHz}, \delta = .25.$$

This was done because the second-order derivative spanned two time steps. Next, we solve for the newest value of S:

$$S^n\left(\frac{\delta_0\omega_0}{\Delta t}+\frac{1}{\Delta t^2}\right)+S^{n-1}\left(\omega_0^2-\frac{2}{\Delta t^2}\right)+S^{n-2}\left(-\frac{\delta_0\omega_0}{\Delta t}+\frac{1}{\Delta t^2}\right)=\omega_0^2\varepsilon_1E^{n-1}$$

$$S^n=-\frac{\left(\omega_0^2-\frac{2}{\Delta t^2}\right)}{\left(\frac{\delta_0\omega_0}{\Delta t}+\frac{1}{\Delta t^2}\right)}S^{n-1}-\frac{\left(-\frac{\delta_0\omega_0}{\Delta t}+\frac{1}{\Delta t^2}\right)}{\left(\frac{\delta_0\omega_0}{\Delta t}+\frac{1}{\Delta t^2}\right)}S^{n-2}+\frac{\omega_0^2\varepsilon_1}{\left(\frac{\delta_0\omega_0}{\Delta t}+\frac{1}{\Delta t^2}\right)}E^{n-1}$$

$$S^n=\frac{\left(2-\Delta t^2\omega_0^2\right)}{(1+\Delta t\delta_0\omega_0)}S^{n-1}-\frac{(1-\Delta t\delta_0\omega_0)}{(1+\Delta t\delta_0\omega_0)}S^{n-2}+\frac{\Delta t^2\omega_0^2\varepsilon_1}{(1+\delta_0\omega_0\Delta t)}E^{n-1}. \tag{2.41}$$

Now let's go back to Eq. (2.40) and take it to the sampled time domain

$$D^n=\varepsilon_rE^n+S^n;$$

and since we are interested in solving for E, we'll write it as

$$E^n=\frac{D^n-S^n}{\varepsilon_r}.$$

Notice that we already have a solution for S^n in Eq. (2.41). And since we need only the previous value of E, i.e., E^{n-1}, we are in business! We don't need to worry about expanding the auxiliary term to make sure they don't require values of E that we don't have yet.

Let's try a different approach. Go back to Eq. (2.39). An alternative form of the Lorentz formulation is

$$S(\omega)=\frac{\gamma\beta}{\left(\alpha^2+\beta^2\right)+j\omega2\alpha+(j\omega)^2}\varepsilon_1E(\omega), \tag{2.42}$$

where

$$\gamma=\frac{\omega_0}{\sqrt{1-\delta_0^2}}$$

$$\alpha=\delta_0\omega_0$$

$$\beta=\omega_0\sqrt{1-\delta_0^2}.$$

At this point, we can go to Table A.1 and look up the corresponding Z transform, which gives

$$S(z)=\frac{e^{-\alpha\cdot\Delta t}\cdot\sin(\beta\cdot\Delta t)\cdot\Delta t\cdot z^{-1}}{1-2e^{-\alpha\cdot\Delta t}\cdot\cos(\beta\cdot\Delta t)\,z^{-1}+e^{-2\alpha\cdot\Delta t}}\gamma\varepsilon_1E(\omega).$$

The corresponding sampled time domain equation is

$$S^n=2e^{-\alpha\cdot\Delta t}\cdot\cos(\beta\cdot\Delta t)\,S^{n-1}-e^{-2\alpha\cdot\Delta t}S^{n-2}+e^{-\alpha\cdot\Delta t}\cdot\sin(\beta\cdot\Delta t)\,\gamma\cdot\Delta t\cdot\varepsilon_1E^{n-1}. \tag{2.43}$$

Well and good. But if both methods are correct, they should both yield the same answers, which means the corresponding terms in Eqs. (2.41) and (2.43) should be the same. Let's see if that's true.

S^{n-2} term: We saw earlier that

$$e^{-2\delta_0\omega_0\Delta t}\cong\frac{(1-\Delta t\delta_0\omega_0)}{(1+\Delta t\delta_0\omega_0)}.$$

E^{n-1} term: If $\beta \cdot \Delta t \ll 1$, then $\sin(\beta \cdot \Delta t) \cong (\beta \cdot \Delta t)$, and

$$e^{-\alpha \cdot \Delta t} \sin(\beta \cdot \Delta t) \cdot \gamma \cdot \Delta t = \frac{\beta \cdot \Delta t}{1 + \alpha \cdot \Delta t} \cdot \gamma \cdot \Delta t$$
$$= \frac{\left(\omega_0 \sqrt{1 - \delta_0^2} \cdot \Delta t\right)}{1 + \delta_0 \omega_0 \cdot \Delta t} \cdot \frac{\omega_0}{\sqrt{1 - \delta_0^2}} \cdot \Delta t \cong \frac{\omega_0^2 \cdot \Delta t^2}{1 + \cdot \delta_0 \omega_0 \Delta t}.$$

S^{n-1} term: By using a Taylor series expansion of the cosine function, we get

$$2e^{-\alpha \cdot \Delta t} \cos(\beta \cdot \Delta t) \cong 2\left(\frac{1}{1 + \alpha \cdot \Delta t}\right)\left(1 - \frac{(\beta \cdot \Delta t)^2}{2}\right)$$
$$= \frac{2 - (\beta \cdot \Delta t)^2}{1 + \alpha \cdot \Delta t} = \frac{2 - \omega_0^2 \cdot \left(1 - \delta_0^2\right) \cdot \Delta t^2}{1 + \omega_0 \delta_0 \cdot \Delta t}.$$

The last step is perhaps our weakest one. In depends upon δ being small compared to 1 so δ^2 will be negligible.

2.5.1 Simulation of Human Muscle Tissue

We will end with the simulation of human muscle tissue. Muscle tissue can be adequately simulated over a frequency range of about two decades with the following formulation:

$$\varepsilon_r^*(\omega) = \varepsilon_r + \frac{\sigma}{j\omega\varepsilon_0} + \varepsilon_1 \frac{\omega_0}{\left(\omega_0^2 + \alpha^2\right) + j2\alpha\omega + \omega^2}. \tag{2.44}$$

(Problem 2.5.2 addresses how one determines the specific parameters, but we will not be concerned with that here.) Inserting Eq. (2.44) into Eq. (2.1b) and taking the Z transforms we get

$$D(z) = \varepsilon_r E(z) + \frac{\sigma \cdot \Delta t/\varepsilon_0}{1 - z^{-1}} \cdot E(z) + \varepsilon_1 \frac{e^{-\alpha \cdot \Delta t} \cdot \sin(\omega_0 \cdot \Delta t) \cdot \Delta t \cdot z^{-1}}{1 - 2e^{-\alpha \cdot \Delta t} \cos(\omega_0 \cdot \Delta t) \cdot z^{-1} + e^{-2\alpha \Delta t} z^{-2}} \cdot E(z). \tag{2.45}$$

We will define two auxiliary parameters:

$$I(z) = \frac{\sigma \cdot \Delta t/\varepsilon_0}{1 - z^{-1}} \cdot E(z) \tag{2.46a}$$

$$S(z) = \frac{\varepsilon_1 \cdot e^{-\alpha \cdot \Delta t} \cdot \sin(\omega_0 \cdot \Delta t) \cdot \Delta t \cdot E(z)}{1 - 2e^{-\alpha \cdot \Delta t} \cos(\omega_0 \cdot \Delta t) \cdot z^{-1} + e^{-2\alpha \cdot \Delta t} z^{-2}}. \tag{2.46b}$$

Now Eq. (2.45) becomes

$$D(z) = \varepsilon_r E(z) + I(z) + z^{-1} S(z). \tag{2.47}$$

Once we have calculated $E(z)$, $I(z)$ and $S(z)$ can be calculated from Eqs. (2.46a) and (2.46b):

$$I(z) = z^{-1} I(z) + \frac{\sigma \cdot \Delta t}{\varepsilon_0} E(z) \tag{2.48a}$$

$$S(z) = 2e^{-\alpha \cdot \Delta t} \cos(\omega_0 \cdot \Delta t) \cdot z^{-1} S(z) - e^{-2\alpha \cdot \Delta t} z^{-2} S(z) + \varepsilon_1 \cdot e^{-\alpha \cdot \Delta t T} \cdot \sin(\omega_0 \cdot \Delta t) \cdot \Delta t \cdot E(z). \tag{2.48b}$$

Remember that in calculating $E(z)$, we need the previous value of $S(z)$ (as indicated by $z^{-1}S(z)$), the present value of $D(z)$, and the present value of $I(z)$. The present value of $D(z)$

is not a problem because, in the order in which the algorithm is implemented, it has already been calculated in Eq. (2.1a). However, we will need the expanded version of $I(z)$. Equation (2.47) becomes

$$D(z) = \varepsilon_r E(z) + z^{-1} I(z) + \frac{\sigma \cdot \Delta t}{\varepsilon_0} \cdot E(z) + z^{-1} S(z),$$

from which $E(z)$ can be calculated by

$$E(z) = \frac{D(z) - z^{-1} I(z) - z^{-1} S(z)}{\varepsilon_r + \sigma \cdot \Delta t / \varepsilon_0} \varepsilon_r E(z) \tag{2.49a}$$

$$I(z) = z^{-1} I(z) + \frac{\sigma \cdot \Delta t}{\varepsilon_0} \cdot E(z) \tag{2.49b}$$

$$\begin{aligned} S(z) = {} & 2e^{-\alpha \cdot \Delta t} \cos(\omega_0 \cdot \Delta t) \cdot z^{-1} S(z) \\ & - e^{-2\alpha \cdot \Delta t} z^{-2} S(z) + \varepsilon_1 \cdot e^{-\alpha \cdot \Delta t} \cdot \sin(\omega_0 \cdot \Delta t) \cdot \Delta t \cdot E(z). \end{aligned} \tag{2.49c}$$

Note that $S(z)$ did not have to be expanded out because it already had a z^{-1} in the numerator.

PROBLEM SET 2.5

1. Modify fd1d_2.3.c to simulate a Lorentz medium with the properties used in Fig. 2.6. Calculate the Fourier amplitudes at 50, 100, and 200 MHz. In light of Fig 2.6, do you get the results you expect?
2. Quantify the results of problem 2.5.1 by comparing them with analytic calculations of the reflection coefficient and transmission coefficient at the three frequencies using Appendix 1A.
3. FDTD simulation has been used extensively to model the effects of electromagnetic radiation on human tissue both for safety [8] and for therapeutic applications [9]. Human muscle tissue is highly frequency dependent. Table 2.1 shows how the dielectric constant and conductivity of muscle vary with frequency [10]. Other tissues display similar frequency dependence [11].

TABLE 2.1 Properties of Human Muscle Tissue

Frequency (MHz)	Dielectric Constant	Conductivity (S/m)
10	160	.625
40	97	.693
100	72	.89
200	56.5	1.28
300	54	1.37
433	53	1.43
915	51	1.60

Write a program to calculate values of ε, σ, χ_1, and t_0 that would give an adequate Debye representation of the data in Table 2.1. You probably will not be able to fit the values exactly. (Hint: Do *not* try for a purely analytic solution; do it by trial and error. In other words, have your program prompt you for the parameters ε, σ, χ_1, and t_0 and then calculate and display the effective dielectric constant and conductivity at each frequency of interest.) Then write a program that will do a similar calculation for the Lorentz parameters. Which one is better?

4. Using a Debye or Lorentz formulation, use the parameters you found from problem 2.5.3 to do a simulation of a pulse striking a medium of muscle tissue at 50, 100, and 433 MHz.

REFERENCES

[1] C. M. Furse, S. P. Mathur, and O. P. Gandhi, Improvements to the finite-difference time-domain method for calculating the radar cross section of a perfectly conducting target, *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-38, July 1990, pp. 919–927.

[2] D. M. Sullivan, Mathematical methods for treatment planning in deep regional hyperthermia, *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-39, May 1991, pp. 864–872.

[3] R. Luebbers, F. Hunsberger, K. Kunz, R. Standler, and M. Schneider, A frequency-dependent finite-difference time-domain formulation for dispersive materials, *IEEE Trans. Electromag. Compat.*, vol. EMC-32, Aug. 1990, pp. 222–227.

[4] R. M. Joseph, S. C. Hagness, and A. Taflove, Direct time integration of Maxwell's equations in linear dispersive media with absorption for scattering and popagation of femtosecond electromagnetic pulses, *Optics Letters*, vol. 16, Sept. 1991, pp. 1412–1414.

[5] O. P. Gandhi, B. Q. Gao, and Y. Y. Chen, A frequency-dependent finite-difference time-domain formulation for general dispersive media, *IEEE Trans. Microwave Theory Tech.*, vol. 41, April 1993, pp. 658–665.

[6] D. M. Sullivan, Frequency-dependent FDTD methods using Z transforms, *IEEE Trans. Antenna Prop.*, vol. AP-40, Oct. 1992, pp. 1223–1230.

[7] A. Ishimaru, *Electromagnetic Wave Propagation, Radiation, and Scattering*, Englewood Cliffs, NJ: Prentice Hall, 1991.

[8] D. M. Sullivan, Use of the finite-difference time-domain method in calculating EM absorption in human tissues, *IEEE Trans. Biomed.*, vol. BME-34, Feb. 1987, pp. 148–157.

[9] D. M. Sullivan, Three dimensional computer simulation in deep regional hyperthermia using the finite-difference time-domain method, *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-38, Feb. 1990, pp. 204–211.

[10] C. C. Johnson and A. W. Guy, Nonionizing electromagnetic wave effects in biological materials and systems, *Proc. IEEE*, vol. 60, June 1972, pp. 692–718.

[11] M. A. Stuchly and S. S. Stuchly, Dielectric properties of biological substances—tabulated, *J. Microwave Power*, vol. 15, 1980, pp. 19–16.

```
/* FD1D_2.1.c.  1D FDTD simulation of a pulse hitting
   a dielectric medium */
/* New formulation using flux density */

# include <math.h>
# include <stdlib.h>
# include <stdio.h>

#define KE  200

main ()
{
    float dx[KE],ex[KE],hy[KE],ix[KE];
    float ga[KE],gb[KE];
    int n,k,kc,ke,kstart,nsteps;
    float ddx,dt,T,epsz,epsilon,sigma;
    float t0,spread,pi,freq_in,arg,pulse;
    FILE *fp, *fopen();
    float ex_low_m1,ex_low_m2,ex_high_m1,ex_high_m2;

    pi  = 3.14159;
    kc = KE/2;                  /* Center of the problem space */
    ddx = .01;                  /* Cell size */
    dt = ddx/6e8;               /* Time steps */
    epsz = 8.8e-12;

    for ( k=0; k < KE; k++ ) {  /* Initialize to free space *
      ga[k] = 1.;
      gb[k] = 0.;
      ex[k] = 0.;
      dx[k] = 0.;
      hy[k] = 0.;
    }
       printf( "Dielectric starts at --> ");
       scanf("%d", &kstart);
       printf( "Epsilon --> ");
       scanf("%f", &epsilon);
       printf( "Conductivity --> ");
       scanf("%f", &sigma);
       printf("%d %6.2f  %6.2f  \n", kstart,epsilon, sigma);

    for ( k=kstart; k <= KE; k++ ) {
      ga[k] = 1./(epsilon + sigma*dt/epsz) ;
      gb[k] =  sigma*dt/epsz ;
    }

       for ( k=1; k <= KE; k++ )
       { printf( "%2d   %4.2f %4.2f\n",k,ga[k],gb[k]); }

    /* These parameters specify the input pulse */
```

```
    t0 = 50.0;
    spread = 20.0;
    T = 0;
    nsteps = 1;

    /* Main part of the program */

while ( nsteps > 0 ) {
       printf( "nsteps --> ");
       scanf("%d", &nsteps);
       printf("%d \n", nsteps);

    for ( n=1; n <=nsteps ; n++)
    {
       T = T + 1;

       /* Calculate the Dx field */
       for ( k=1; k < KE; k++ )
       { dx[k] = dx[k] + 0.5*( hy[k-1] - hy[k] ) ; }

       /* Put a Gaussian pulse at the low end */

       freq_in = 3e8;
       pulse =  exp(-.5*(pow((t0-T)/spread,2.0)));
        dx[kc] =  dx[kc] + pulse;
       printf( "%5.1f %6.2f %6.2f\n",T,pulse,dx[kc]);

          /* Calculate Ex from Dx */
          for ( k=0; k < KE-1; k++ )
        {   ex[k] = ga[k]*(dx[k] - ix[k] );
            ix[k] = ix[k] + gb[k]*ex[k];  }

       /* Boundary conditions */
       ex[0]       = ex_low_m2;
       ex_low_m2   = ex_low_m1;
       ex_low_m1   = ex[1];

       ex[KE-1]     = ex_high_m2;
       ex_high_m2   = ex_high_m1;
       ex_high_m1   = ex[KE-2];


       /* Calculate the Hy field */
       for ( k=0; k < KE-1; k++ )
       { hy[k] = hy[k] + .5*( ex[k] - ex[k+1] ) ; }

    }

       for ( k=0; k < KE; k++ )
       { printf( "%2d   %6.2f %6.2  \n",k,dx[k],ex[k]); }
```

```
        /* Write the E field out to a file "Ex" */
        fp = fopen( "Ex","w");
        for ( k=0; k < KE; k++ )
        { fprintf( fp,"  %6.3f \n",ex[k]); }
        fclose(fp);
        printf( "%5.1f \n",T);
}
}
```

```
/* FD1D_2.2.c. The Fourier Tranform has been added.*/

# include <math.h>
# include <stdlib.h>
# include <stdio.h>

#define KE 200

main ()
{
    float dx[KE],ex[KE],hy[KE],ix[KE];
    float ga[KE],gb[KE];
    int n,m,k,kc,ke,kstart,nsteps;
    float ddx,dt,T,epsz,epsilon,sigma;
    float t0,spread,pi,pulse;
    FILE *fp, *fopen();
    float ex_low_m1,ex_low_m2,ex_high_m1,ex_high_m2;
    float real_pt[5][KE],imag_pt[5][KE];
    float freq[5],arg[5],ampn[5][KE],phasen[5][KE];
    float real_in[5],imag_in[5],amp_in[5],phase_in[5];
    float mag[KE];

    kc = KE/2;                  /* Center of the space */
    pi  = 3.14159;
    epsz = 8.8e-12;
    ddx = .01;                  /* Cells size */
    dt = ddx/6e8;               /* Time steps */
       printf(" %6.4f  %10.5e \n",ddx,dt);

    for ( k=1; k < KE; k++ ) {  /* Initialize to free space */
      ga[k] = 1.;
      gb[k] = 0.;
      dx[k] = 0.;
      ex[k] = 0.;
      hy[k] = 0.;
      ix[k] = 0.;
      mag[k] = 0.;

        for ( m=0; m <= 2; m++ ) {
           real_pt[m][k]   = 0.;        /* Real and imaginary parts    */
           imag_pt[m][k]   = 0.;        /*    of the Fourier Transform */
           ampn[m][k]      = 0.;        /* Amplitude and phase of the  */
           phasen[m][k]    = 0.;        /*    Fourier Transforms       */
           }
      }
        for ( m=0; m <= 2; m++ ) {
           real_in[m] = 0.;       /* Fourier Trans. of input pulse */
           imag_in[m]  = 0.;
       }
```

```
    ex_low_m1 = 0.;
    ex_low_m2 = 0.;
    ex_high_m1 = 0.;
    ex_high_m2 = 0.;

       /* Parameters for the Fourier Transform */

       freq[0] = 100.e6;
       freq[1] = 200.e6;
       freq[2] = 500.e6;

       for ( n=0; n<= 2; n++ )
       {  arg[n] = 2*pi*freq[n]*dt;
       printf( "%2d %6.2f %7.5f \n",n,freq[n]*1e-6,arg[n]);
       }

       printf( "Dielectric starts at --> ");
       scanf("%d", &kstart);
       printf( "Epsilon --> ");
       scanf("%f", &epsilon);
       printf( "Conductivity --> ");
       scanf("%f", &sigma);
       printf("%d %6.2f  %6.2f  \n", kstart,epsilon, sigma);

    for ( k=kstart; k <= KE; k++ ) {
      ga[k] = 1./(epsilon + sigma*dt/epsz ) ;
      gb[k] =  sigma*dt/epsz ;
    }

       for ( k=1; k <= KE; k++ )
       { printf( "%2d   %6.2f %6.4f \n",k,ga[k],gb[k]); }

    /* These parameters specify the input pulse */
    t0 = 50.0;
    spread = 10.0;

    T = 0;
    nsteps = 1;

    /* Main part of the program */

while ( nsteps > 0 ) {
       printf( "nsteps --> ");
       scanf("%d", &nsteps);
       printf("%d \n", nsteps);

    for ( n=1; n <=nsteps ; n++)
    {
       T = T + 1;
```

```
      /* Calculate the Dx field */
      for ( k=0; k < KE; k++ )
      { dx[k] = dx[k] + 0.5*( hy[k-1] - hy[k] ) ; }

      /* Initialize with a pulse */

       pulse =  exp(-.5*(pow((t0-T)/spread,2.0)));
      dx[5] =  dx[5] + pulse;
      printf( "%5.1f %6.2f %6.2f \n",T,pulse,dx[5]);

      /* Calculate Ex from Dx */
      for ( k=0; k < KE-1; k++ )
      { ex[k] = ga[k]*(dx[k] - ix[k] )  ;
      ix[k] = ix[k] + gb[k]*ex[k]     ;
      }

         /* Calculate the Fourier transform of Ex. */
        for ( k=0; k < KE; k++ )
       {  for ( m=0; m <= 2; m++ )
             { real_pt[m][k] = real_pt[m][k] + cos(arg[m]*T)*ex[k] ;
               imag_pt[m][k] = imag_pt[m][k] - sin(arg[m]*T)*ex[k] ;
        }   }

          /* Fourier Transform of the input pulse */
        if   (T < 100 ) {
               for ( m=0; m <= 2 ; m++ )
               { real_in[m] = real_in[m] + cos(arg[m]*T)*ex[10] ;
                 imag_in[m] = imag_in[m] - sin(arg[m]*T)*ex[10] ;
        }     }

      /* Boundary conditions */
      ex[0]       = ex_low_m2;
      ex_low_m2   = ex_low_m1;
      ex_low_m1   = ex[1];

      ex[KE-1]     = ex_high_m2;
      ex_high_m2   = ex_high_m1;
      ex_high_m1   = ex[KE-2];

      /* Calculate the Hy field */
      for ( k=0; k < KE-1; k++ )
      { hy[k] = hy[k] + .5*( ex[k] - ex[k+1] ) ; }

   }

/* End of the main loop */

   /*  for ( k=0; k < KE; k++ )
     { printf( "%2d   %6.2f %6.2  \n",k,dx[k],ex[k]); }  */
```

```
        /* Write the E field out to a file "Ex" */
      fp = fopen( "Ex","w");
      for ( k=0; k < KE; k++ )
      { fprintf( fp,"  %6.3f \n",ex[k]); }
      fclose(fp);

  /* Calculate the amplitude and phase of each frequency */

  /* Amplitude and phase of the input pulse */
    for (m=0; m <= 2; m++ )
     {      amp_in[m] = sqrt( pow(imag_in[m],2.)
                            + pow(real_in[m],2.) );
            phase_in[m] = atan2( imag_in[m],real_in[m]);
       printf( "%d  Input Pulse :  %8.4f %8.4f %8.4f  %7.2f\n",
       m,real_in[m],imag_in[m],amp_in[m],(180.0/pi)*phase_in[m]);

        for ( k=1; k < KE; k++ )
         {  ampn[m][k] = (1./amp_in[m])*sqrt( pow(real_pt[m][k],2.)
                            + pow(imag_pt[m][k],2.) );
            phasen[m][k] = atan2( imag_pt[m][k],real_pt[m][k])
                         - phase_in[m];
          }

       /*  for ( k=1; k < KE; k++ )
              { printf( "%d  %6.3f %6.3f %6.3f \n",
            k,ampn[0][k]),ampn[1][k],ampn[2][k]; } */

       /* Write the amplitude field out to a files "Amp" */
          fp = fopen( "Amp0","w");
        for ( k=0; k < KE; k++ )
              { fprintf( fp,"  %8.5f \n",ampn[0][k]); }
      fclose(fp);
          fp = fopen( "Amp1","w");
        for ( k=0; k < KE; k++ )
              { fprintf( fp,"  %8.5f \n",ampn[1][k]); }
      fclose(fp);
          fp = fopen( "Amp2","w");
        for ( k=0; k < KE; k++ )
              { fprintf( fp,"  %8.5f \n",ampn[2][k]); }
      fclose(fp);

        printf( "%5.1f \n",T);
}

}
```

```
/* FD1D_2.3.  1D FDTD simulation of a frequency dependent material */

# include <math.h>
# include <stdlib.h>
# include <stdio.h>

#define KE 200

main ()
{
    float dx[KE],ex[KE],hy[KE],ix[KE],sx[KE];
    float ga[KE],gb[KE],gc[KE];
    int n,m,k,kc,ke,kstart,nsteps;
    float ddx,dt,T,epsz,epsilon,sigma;
    float t0,spread,pi,pulse;
    FILE *fp, *fopen();
    float ex_low_m1,ex_low_m2,ex_high_m1,ex_high_m2;
    float tau,chi1,del_exp;
    float real_pt[5][KE],imag_pt[5][KE];
    float freq[5],arg[5],ampn[5][KE],phasen[5][KE];
    float real_in[5],imag_in[5],amp_in[5],phase_in[5];
    float mag[KE];

    kc = KE/2;                  /* Center of the space */
    pi  = 3.14159;
    epsz = 8.8e-12;
    ddx = .01;                  /* Cells size */
    dt = ddx/6e8;               /* Time steps */
       printf(" %6.4f  %10.5e \n",ddx,dt);

    for ( k=1; k < KE; k++ ) {  /* Initialize to free space */
      ga[k] = 1.;
      gb[k] = 0.;
      gc[k] = 0.;
      dx[k] = 0.;
      ex[k] = 0.;
      hy[k] = 0.;
      ix[k] = 0.;
      sx[k] = 0.;

      for ( m=0; m <= 4; m++ ) {
          real_pt[m][k] = 0.;      /* Real and imaginary parts    */
          imag_pt[m][k] = 0.;      /*    of the Fourier Transform */
          ampn[m][k]    = 0.;      /* Amplitude and phase of the  */
          phasen[m][k]  = 0.;      /*    Fourier Transforms       */
      }
    }
      for ( m=0; m <= 4; m++ ) {
```

```
        real_in[m] = 0.;      /*    Fourier Trans. of input pulse */
        imag_in[m] = 0.;
    }

ex_low_m1 = 0.;
ex_low_m2 = 0.;

ex_high_m1 = 0.;
ex_high_m2 = 0.;

    /* Parameters for the Fourier Transform */

    freq[0] = 50.e6;
    freq[1] = 200.e6;
    freq[2] = 500.e6;

    for ( n=0; n<= 4; n++ )
    {  arg[n] = 2*pi*freq[n]*dt;
    printf( "%2d %6.2f %7.5f \n",n,freq[n]*1e-6,arg[n]);
    }

    printf( "Dielectric starts at --> ");
    scanf("%d", &kstart);
    printf( "Epsilon --> ");
    scanf("%f", &epsilon);
    printf( "Conductivity --> ");
    scanf("%f", &sigma);
    printf( "chi1 --> ");
    scanf("%f", &chi1);
    tau = 1000.;                  /* Make sure tau is > 0. */
    if(chi1 > 0.0001 ) {
    printf( "tau (in microseconds) --> ");
    scanf("%f", &tau);
       del_exp = exp(-dt/tau); }
    printf("%d %6.2f  %6.2f  %6.2f  %6.2f\n", kstart,epsilon,
       sigma,tau,chi1);
    tau = 1.e-6*tau;
    { printf( "del_exp =  %8.5f \n",del_exp); }

for ( k=kstart; k <= KE; k++ ) {
  ga[k] = 1./(epsilon + sigma*dt/epsz + chi1*dt/tau) ;
  gb[k] =  sigma*dt/epsz ;
  gc[k] =  chi1*dt/tau ;
}

    for ( k=1; k <= KE; k++ )
    { printf( "%2d   %6.2f %6.4f %6.4f \n",k,ga[k],gb[k],gc[k]); }

/* These parameters specify the input pulse */
t0 = 50.0;
```

```
    spread = 10.0;

    T = 0;
    nsteps = 1;

    /* Main part of the program */

while ( nsteps > 0 ) {
       printf( "nsteps --> ");
       scanf("%d", &nsteps);
       printf("%d \n", nsteps);


    for ( n=1; n <=nsteps ; n++)
    {
       T = T + 1;

       /* Calculate the Dx field */
       for ( k=0; k < KE; k++ )
       { dx[k] = dx[k] + 0.5*( hy[k-1] - hy[k] ) ; }

       /* Initialize with a pulse */

       pulse =  exp(-.5*(pow((t0-T)/spread,2.0)));
       dx[5] =  dx[5] + pulse;
       printf( "%5.1f %6.2f %6.2f \n",T,pulse,dx[5]);

       /* Calculate Ex from Dx */
       for ( k=0; k < KE-1; k++ )
       { ex[k] = ga[k]*(dx[k] - ix[k] - sx[k])  ;
       ix[k] = ix[k] + gb[k]*ex[k]    ;
       sx[k] = del_exp*sx[k] + gc[k]*ex[k]    ;  }

       /* Calculate the Fourier transform of Ex. */
       for ( k=0; k < KE; k++ )
           {  for ( m=0; m <= 2; m++ )
             { real_pt[m][k] = real_pt[m][k] + cos(arg[m]*T)*ex[k] ;
               imag_pt[m][k] = imag_pt[m][k] - sin(arg[m]*T)*ex[k] ; }
       }

        /* Fourier Transform of the input pulse */
        if  (T < 100 ) {
             for ( m=0; m <= 2 ; m++ )
             { real_in[m] = real_in[m] + cos(arg[m]*T)*ex[10] ;
               imag_in[m] = imag_in[m] - sin(arg[m]*T)*ex[10] ; }
        }

       /* Boundary conditions */
       ex[0]      = ex_low_m2;
       ex_low_m2  = ex_low_m1;
```

```
      ex_low_m1  = ex[1];

      ex[KE-1]     = ex_high_m2;
      ex_high_m2  = ex_high_m1;
      ex_high_m1  = ex[KE-2];

      /* Calculate the Hy field */
      for ( k=0; k < KE-1; k++ )
      { hy[k] = hy[k] + .5*( ex[k] - ex[k+1] ) ; }

   }

/* End of the main loop */

   /*  for ( k=0; k < KE; k++ )
     { printf( "%2d   %6.2f %6.2  \n",k,dx[k],ex[k]); }  */


      /* Write the E field out to a file "Ex" */
     fp = fopen( "Ex","w");
     for ( k=0; k < KE; k++ )
     { fprintf( fp,"  %6.3f \n",ex[k]); }
     fclose(fp);

/* Calculate the amplitude and phase of each frequency */

/* Amplitude and phase of the input pulse */
  for (m=0; m <= 2; m++ )
  {     amp_in[m] = sqrt( pow(imag_in[m],2.)
                        + pow(real_in[m],2.) );
        phase_in[m] = atan2( imag_in[m],real_in[m]);
        printf( "%d  Input Pulse :  %8.4f %8.4f %8.4f  %7.2f\n",
        m,real_in[m],imag_in[m],amp_in[m],(180.0/pi)*phase_in[m]);

      for ( k=1; k < KE; k++ )
          { ampn[m][k] = (1./amp_in[m])*sqrt( pow(real_pt[m][k],2.)
                        + pow(imag_pt[m][k],2.) );
    phasen[m][k] = atan2( imag_pt[m][k],real_pt[m][k])-phase_in[m];
    printf( "%d  %8.4f %8.4f %8.5f  %8.2f  \n",
    k,real_pt[m][k],imag_pt[m][k],ampn[m][k],(180.0/pi)*phasen[m][k]) ;
      }
  }

     /*  for ( k=kstart; k < KE; k++ )
           { printf( "%d  %6.3f %6.3f %6.3f \n",
          k,ampn[1][k]),ampn[2][k],ampn[3][k]; } */

    /* Write the amplitude field out to a file "Amp" */
       fp = fopen( "Amp0","w");
       for ( k=1; k < KE; k++ )
```

```
            { fprintf( fp,"  %8.5f \n",ampn[0][k]); }
        fclose(fp);
          fp = fopen( "Amp1","w");
          for ( k=1; k < KE; k++ )
            { fprintf( fp,"  %8.5f \n",ampn[1][k]); }
        fclose(fp);
          fp = fopen( "Amp2","w");
          for ( k=1; k < KE; k++ )
            { fprintf( fp,"  %8.5f \n",ampn[2][k]); }
        fclose(fp);

        printf( "%5.1f \n",T);
}

}
```