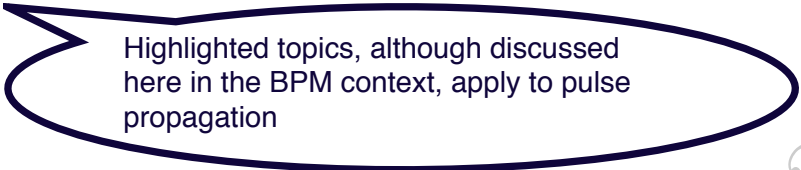


---

## Contents



Highlighted topics, although discussed here in the BPM context, apply to pulse propagation

<b>1</b>	<b>Maxwell's equations: Numerical simulation perspective</b>	<b>1</b>
1.1	Maxwell's equations and the Beam Propagation Method	1
1.1.1	Propagation equations in the BPM context	4
1.1.2	Divergence equations in the BPM context	5
1.1.3	Medium models in the BPM context	6
1.2	Maxwell detour: Illustration of numerical issues	8
1.2.1	Three sources of difficulties in large-scale Maxwell simulations	8
1.2.2	Direct Maxwell solver example - FDTD Yee scheme	10
1.2.3	Dispersion relation as a solvability condition	14
1.2.4	Numerical dispersion relation as a solvability condition	15
1.2.5	3D FDTD Yee scheme: numerical wave properties	18
1.3	Practice track: Numerical properties of FD Maxwell solvers	21
1.3.1	Initial conditions, stability, and numerical dispersion	21
1.3.2	Absorbing boundary conditions	30
1.3.3	Numerical dispersion curve extracted from noise	36
<b>2</b>	<b>Beam propagation techniques in homogeneous media</b>	<b>41</b>
2.1	Expansion into Plane Waves	41
2.1.1	Basic DFT-based method	43
2.1.2	Paraxial approximation	44
2.1.3	Propagation step length and sampling	45
2.1.4	Stationary phase approximation	46
2.1.5	Beam profile evaluation in the far field	47
2.2	Practice track: Fourier transform technique	48
2.2.1	Gaussian beam solution propagating at large angles	48
2.2.2	Implementation of FFT-BPM, paraxial and non-paraxial regimes	50
2.2.3	Simulation of beam propagation in the Fraunhofer regime	53
2.2.4	Strongly non-paraxial regime: Poisson's bright spot	55
2.2.5	Calculation of the longitudinal vector component	58
2.3	Expansion into Bessel Beams	60
2.3.1	Basic DHT-based method	61
2.3.2	Connection to Wave and Helmholtz Equations	62
2.3.3	Paraxial and non-paraxial solutions	63
2.3.4	Discrete Hankel Transform	63
2.3.5	Application of DHT to free-space propagation	66

## 1.2 Maxwell detour: Illustration of numerical issues

In many situations the Beam Propagation Method is an inevitable replacement for the full Maxwell's equations. The purpose of this Section is to support this claim with facts, and look closer at the roots of practical limitations in numerical modeling of optical-frequency electromagnetic waves and their propagation. We also take a brief guided tour through a typical implementation of a Maxwell simulator. While Maxwell simulators are not our main objective in this course, their conceptual simplicity offers a very intuitive “play-field” to illustrate important elements of the theory that underlines the Beam Propagation Method. This section is also meant to provide an opportunity to make the first contact with important numerical issues such as accuracy, stability and dispersion. These are notions of fundamental importance for numerical solution of partial differential equations in general.

### 1.2.1 Three sources of difficulties in large-scale Maxwell simulations

It is certainly true that Maxwell's equations can be efficiently solved by numerical solvers and applied to many areas of computational electromagnetics. However, they are rather ill suited for computer simulations of nonlinear phenomena in optical pulses which propagate for significant distances. This Section explains the origins of this limited utility in the general field of nonlinear optics (thus, not all arguments will apply to contexts in which the BPM is typically used). Roughly speaking there are three areas where problems show up when one attempts to apply a numerical Maxwell solver to a large problem:

1. **Grid resolutions** imply memory and computational time requirements that are only acceptable for sufficiently small problems.
2. **Numerical dispersion** is very strong. For example, even the “discretized computational vacuum” is more dispersive than real water!
3. **Realistic medium response** models require lots of additional memory and computational time. They also suffer from numerical dispersion problems.

In what follows we briefly review the general properties of direct Maxwell solvers. We will restrict our attention to the so-called Finite Difference Time-Domain (FDTD) solvers as the most common representatives of direct Maxwell solution methods. However, the discussed issues are relevant for Maxwell's equations simulators in general.

#### Grid resolutions and memory requirements

First, let us consider the memory requirements. A direct solver works over a fixed spatial domain, and evolves the grid-based representation of the electric and magnetic fields in discrete time steps. As a rule of thumb, for accurate simulations one typically needs about 30 grid points per wavelength in space. About the same number of steps is needed in time per single wave-cycle. Such a resolution may be practically achievable for radio- or micro-waves and small simulated volumes (measured in units of cubic wavelengths). But in the optics context, the resolution requirements translate into sub-micrometer spatial, and sub-femtosecond temporal resolutions. If an optical pulse beam is only one centimeter wide, and propagates over a laboratory-scale distance of only a few meters, a single snapshot of the field will require of the order of  $10^{20}$  field-variables to store in the memory. Moreover, for a mere meter of propagation, each of these variables must be updated through tens of millions of integration steps! Taken that a typical computer memory and performance increased about three orders of magnitude over a decade, it becomes clear that a brute-force approach in the optics context is not going to be feasible in any foreseeable future.

### Numerical dispersion

The second property which practically disqualifies direct solvers for the simulation of long-distance nonlinear wave propagation is the numerical dispersion. Similar to the natural light waves, the phase and group velocities of the “numerical waves” depend on their frequency - they exhibit the so-called numerical dispersion. It is a general property of each and every numerical method which mimics wave propagation, that the true wave properties are “warped,” or modified through the discrete nature of the simulation. More precisely, the real relation between the wave’s propagation velocity and its frequency, which is usually called the dispersion relation, is replaced by an artificial one which is often drastically different. As a rule, a discrete numerical method can mimic the correct dispersion properties of waves only for small frequencies. Consequently, a very small fraction of the bandwidth which is available to the numerics can actually be utilized. In case of the typical discretization scheme for the Maxwell’s equations, this artificial deformation of the natural relation between the wave velocity and its frequency is very restricting. This is because the numerical dispersion depends on the grid resolution, and is in general extremely strong from the point of view of nonlinear optics: For example, a “numerical vacuum” simulated by a direct solver will typically exhibit the chromatic dispersion orders of magnitude larger than water or other transparent condensed media. Surely, these unwanted effects decrease in magnitude as we increase the grid resolution, but to get them fully under control would require enormous grid resolution. Because in the computational nonlinear optics it is absolutely crucial to capture the chromatic dispersion *very accurately* this problem is extremely serious.

### Realistic medium models

Last but not least, it turns out that in direct solvers it is quite difficult to implement models of nonlinear and dispersive media. The origin of the problem is that dispersion and often also the nonlinearity is connected to some kind of memory in the medium. This does not mesh well with the fact that a direct solver scheme is usually designed to store only a single temporal snapshot of the field configuration. If the reaction of the medium at any given point depends on the history of the local field, we must keep sufficient information about this history available to the numerical solver. This can easily multiply the memory needs. The problem gets even worse once we consider that the frequency-dependent properties of the model media are also plagued by the numerical dispersion. The artificial deformation of the actual numerical medium response can be significantly different from the targeted frequency-dependent properties for the same reason we pointed out for the linear wave propagation.

To summarize, it is clear that the direct numerical solution of Maxwell’s equations is not feasible for many nonlinear and long-distance propagation optical phenomena. The Beam Propagation Method, which is in fact a whole family of approaches, has been developed mainly to overcome the difficulty of propagation over long (in comparison with the light wavelength) distances.

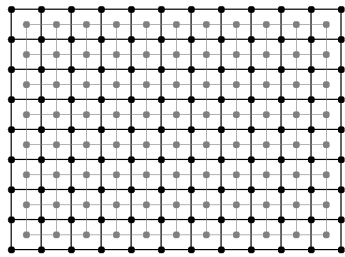
### 1.2.2 Direct Maxwell solver example - FDTD Yee scheme

The following subsection is devoted to one of the most useful and frequently applied algorithms for direct numerical solution of Maxwell equations. We say "direct" to emphasize that nothing is done to the equations themselves such as their transformation into a system that would be perhaps easier to solve. Thus, the equations we analyze in this section are not those of a typical BPM method. Nevertheless, it will be a useful warm-up exercise, with take-away lessons to be applied later in the course. Our detour from the BPM territory into the Maxwell realm is motivated and informed by the following:

- when studying a given method, it is important to understand "competitor techniques;"
- working with a Maxwell solver allows to appreciate frequently occurring numerical issues in a relatively simple setting;
- in particular, we will get a first taste of how Finite-Difference (FD) techniques work, and ...
- we work out an example of numerical dispersion and stability calculations, which are crucial ingredients in all BPM methods we study in this course

Finite-Difference Time-Domain (FDTD) Maxwell solvers represent electromagnetic fields on discrete computational grids. There are many choices for how the grids can be organized, or how the discrete sampling points for different fields and their vectorial components are distributed in space.

One big family of solvers utilizes the so-called dual grids for electric and magnetic fields. What are dual grids or lattices? Roughly speaking they have the following property: Assume one lattice is given and let us visualize a square lattice for simplicity. Its dual grid is also a square lattice but it is shifted diagonally such that where the original (or primal) grid has an edge (i.e. a connector of two vertices), the dual has a facet (which is a side of a cell).



Square lattice (dark vertices), and its dual (gray vertices). Note that gray vertices surround each of the dark ones and vice-versa. In a 2D Maxwell solver, vertical and horizontal components of electric fields could be located at the corresponding edges of the dark grid, while the magnetic fields could be placed on the gray nodes of the dual grid.

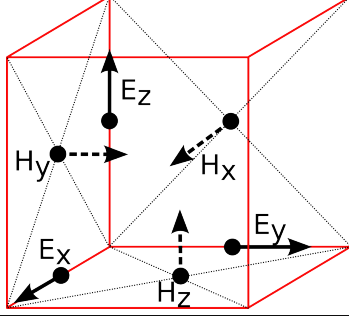
Between the duals, any given vertex of the original grid is surrounded by neighbors that belong to the dual grid, which is a desirable property: If one places the electric field sampling points at vertices or, alternatively, along the edges of one grid, the magnetic fields will occupy the dual lattice. This reflects the very property of Maxwell equations in which the time-derivatives of the electric field components are expressed as function of the magnetic fields and vice versa. The reason we may want to do this is because of the way we approximate partial derivatives. This will become clearer soon.

---

**Exercise:** Research a proper definition of dual lattices. What is the dual grid for a triangular lattice? What is it for a cubic lattice?

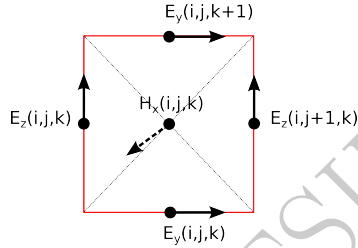
---

The placement of the electric and magnetic fields on mutually dual lattices is a characteristic feature of the Yee scheme. In three dimensions it is illustrated in the following figure:



Elementary cell of a cubic lattice supporting the computational domain for a direct Maxwell solver using the Yee scheme. Each cell carries the electric field vector components placed along its edges. The magnetic vector components are located in the centers of the cell's facets.

The figure shows six vector components per single cell of the computational “box”. Each of the components are thus sampled in space in the same way but at mutually shifted locations. Indeed, the locations of each vector component constitute six interwoven cubic grids (considering equal lattice constants  $\Delta x, \Delta y, \Delta z$ ). As a result, each component of the electric field is surrounded by four samples of the magnetic field and the directions of the latter are perpendicular to the electric field. The situation is completely “symmetric” from the point of view of the magnetic field. You can see this in the above figure by extending the magnetic arrows into lines — you should see how the lines you get form facets of a dual grid and that the electric field samples are placed in the centers of these facets. This facilitates an efficient, symmetric in space, finite-difference approximation of the curl operations needed in the Maxwell equations.



One facet of the cubic grid used in Yee method. The magnetic field component placed in its center is surrounded by the perpendicular components of the electric field. It turns out that these are exactly the quantities that are coupled in the Ampere's law: a temporal derivative of the magnetic field in the facet center is connected to the curl of the electric field.

The figure above illustrates how the grid arrangement facilitates discrete approximation of the curl operations that appear in the Maxwell's equations. The  $x$ -component of  $\nabla \times \mathbf{E}$  can be approximated in terms of the electric field quantities around the perimeter of the facet shown in the figure as

$$(\nabla \times \mathbf{E})_x \approx \frac{1}{\Delta y} (E_z(i, j+1, k) - E_z(i, j, k)) - \frac{1}{\Delta z} (E_y(i, j, k+1) - E_y(i, j, k)) \quad (1.17)$$

This estimate is used in the equation for the temporal derivative of the field variable in the center, namely  $H_x(i, j, k)$ . The situation is completely analogous for other components and also for the electric field.

The arrangement of computational grids described here is called staggered. What we have just illustrated is staggering in space. But the Yee method staggers the electric and magnetic

grids also in time. Of course, time can be viewed as just another dimension, very much like the spatial dimensions. We will look at this closer first in the simple, one-dimensional case.

In a space with a single spatial dimension (i.e. a line) there are only two electromagnetic field components, one electric and one magnetic, both perpendicular to the spatial direction and to each other. An alternative and more physical way to visualize a 1-D Maxwell system is to consider a 3-D configuration of the electromagnetic field, which all quantities depend on only one spatial variable which we choose to be  $x$ .

---

**Exercise:** Starting from the Maxwell equations, reduce the system to one spatial dimension, e.g. by removing all spatial derivatives with respect to  $y$  and  $z$ . Consider only radiation fields with no free charges or currents. Also show that  $E$  and  $cB$  could be natural “units” to represent the electric and magnetic components. Note that in an electromagnetic plane wave propagating in free space the electric and magnetic field amplitudes are related by  $cB = E$ . So the choice is not only convenient but also quite physical.

---

If one chooses suitable units then the one-dimensional Maxwell equations reduce to

$$\partial_t E(x, t) = +\partial_x H(x, t) \quad , \quad \partial_t H(x, t) = -\partial_x E(x, t) \quad (1.18)$$

Next we decide on how these equations are approximated through finite difference derivative expressions. Recall that in general the first-order derivative of a function  $f(x)$  at a point  $x = x_0$  can be approximated as

$$f'(x_0) \approx \frac{f(x_0 + dx/2) - f(x_0 - dx/2)}{dx} \quad (1.19)$$

This expression is second-order accurate. It simply means that the error decreases as, or is proportional to  $(dx)^2$ . Note that the similar expression

$$f'(x_0) \approx \frac{f(x_0 + dx) - f(x_0)}{dx} \quad (1.20)$$

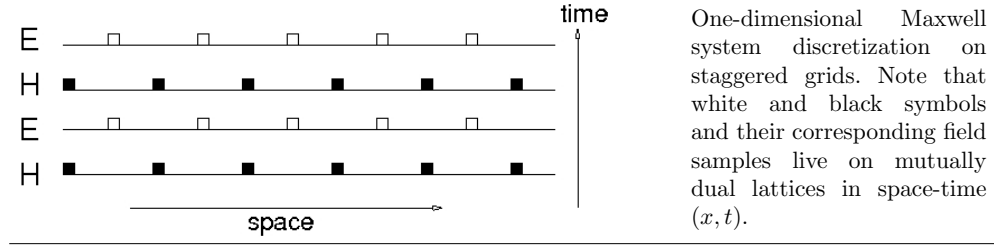
is only first-order accurate. The crucial difference between the two is that the first one is *symmetric* with respect to the point at which the derivative is sought.

---

**Exercise:** Write a Matlab or Mathematica program to demonstrate the order of accuracy for both of the above finite-difference formulas. Choose a function for which the exact answer can be computed and plot the discretization error of the derivative as a function of  $dx$ . Plot the results on log scale, compare the resulting slopes, then relate them to the order of accuracy.

---

The lesson here is that if we could ensure that our finite difference approximations for all partial derivatives in the problem can be obtained from *symmetric* differences, we gain an order of accuracy for free (i.e. we get the second order instead of just first). The grids staggered in space take care of this for the spatial derivatives. However, we also need temporal derivatives and that is why we will stagger our grids in time, too. Of course, the numerical algorithm does not keep in the computer memory samples of electromagnetic field for all discrete times simulated. Instead, it is sufficient to store one time slice of each the magnetic and the electric fields, provided that they are 'spaced' in time by one half of the temporal integration step. This is the main idea behind the success of the Yee method. The one-dimensional version of the Yee grid including four temporal slices is shown in the following diagram:



Now we can return to equations (1.18) and design their discrete approximation. First we need to choose a point with respect to which derivatives will be calculated. Since we aim to end up with symmetric differences, this point should be located in between the field samples. Assume that all field samples in the above picture are known with the exception of the last one (top row) which stores the electric field at time  $t + dt$ . What we can use at this point in time is the previous two rows with the magnetic field at time  $t + dt/2$  and the “old” electric field at time  $t$ . The temporal derivative of the electric field can be approximated as

$$\partial_t E(x, t + dt/2) \approx \frac{E(x, t + dt) - E(x, t)}{dt} \quad (1.21)$$

which is second-order accurate in  $dt$ , and could be applied to all  $x$  where the grid points are populated by the electric fields. It is important to keep in mind that the second-order accuracy in this formula is only reached for the point in between the two discrete values. This point is therefore at time  $t + dt/2$  and this is good since it is exactly the time for which we have previously calculated the magnetic field. The spatial derivative of the magnetic field can be evaluated for the same point as:

$$+\partial_x H(x, t + dt/2) \approx \frac{H(x + dx/2, t + dt/2) - H(x - dx/2, t + dt/2)}{dx} \quad (1.22)$$

These finite-difference derivative approximations can be used to replace the partial derivatives in the first of the two equations in (1.18), to obtain:

$$\frac{E(x, t + dt) - E(x, t)}{dt} = \frac{H(x + dx/2, t + dt/2) - H(x - dx/2, t + dt/2)}{dx}, \quad (1.23)$$

and from this follows the update scheme for the electric field:

$$E(x, t + dt) = E(x, t) + \frac{dt}{dx} [H(x + dx/2, t + dt/2) - H(x - dx/2, t + dt/2)]. \quad (1.24)$$

Note that the values of  $x$  and  $t$  run only over the discrete points in space and time where the sampled fields exist.

The discretization of the second Maxwell equation (1.18) proceeds along similar lines. This time we think of an update that takes the magnetic field from the time, say,  $t - dt/2$  to the time  $t + dt/2$ . In the process, electric fields at  $t$  are utilized to express the spatial derivative. The counterpart of (1.23) becomes

$$\frac{H(x - dx/2, t + dt/2) - H(x - dx/2, t - dt/2)}{dt} = \frac{E(x - dx, t) - E(x, t)}{dx} \quad (1.25)$$

and the update scheme for the magnetic field reads

$$H(x - dx/2, t + dt/2) = H(x - dx/2, t - dt/2) + \frac{dt}{dx} [E(x - dx, t) - E(x, t)] \quad (1.26)$$

Several points are worthwhile to note in the equations for the Yee method update (1.24), (1.26). First, the method is kind of a leap-frog scheme when calculations of new magnetic field samples take alternate turns with the evaluations of electric fields. At all times, the algorithm must keep in the memory one time slice for the magnetic and one time-slice for the electric field. They each correspond to integration times spaced apart by  $\Delta t/2$ . Thus, different types of field samples never share the same location in space and in time. This is something to keep in mind when calculating physical observables, for example energy, or when preparing initial condition for simulations.

---

**Exercise:** Consider a one-dimensional Maxwell system and an initial condition given by  $E(x, t = 0) = E_0(x)$  and  $H(x, -\Delta t/2) = 0$ , with  $E_0(x)$  being an arbitrary smooth function.

- a) Explain why the subsequent numerical propagation will generate two wave-packets with the spatial shape of  $E_0(x)/2$  propagating in opposite directions.
  - b) Explain how would you create an initial condition that represents a pulsed waveform that propagates in the positive  $x$ -direction, with no waves propagating in the negative  $x$ -direction.
- 

Another very important point about the numerical representation of the Maxwell system is the appearance of the ratio  $\Delta t/\Delta x$  between the integration step in time and the spacing of the computation grid. Here we have used dimensionless units, but it should be obvious that if we were just a little more careful keeping track of all constants and units, the relevant dimensionless ratio would end up to be  $c\Delta t/\Delta x$ .

Imagine that this number is large because of how we have chosen our numerical grid resolution and the integration time step. It would mean that at each update the change in either magnetic or electric field would be large. Hence it is natural to expect that such a numerical algorithm would not be very accurate. Worse, it would not even survive for very long, because of what we call numerical instability. This is the topic we will investigate next.

### 1.2.3 Dispersion relation as a solvability condition

In preparation for the following analysis of numerical dispersion, let us briefly recall the derivation of plane-wave solution properties to Maxwell equations in a homogeneous non-magnetic medium. The point to be emphasized is that the physical dispersion relation, a characteristics of the medium, can be viewed as a solvability condition for a homogeneous linear system of equations. The unknowns in this system are the wave-amplitudes of electric and magnetic fields. The determinant of the system must vanish in order for the non-trivial solution to exist and the dispersion relation ensures just that. The same will become evident for discrete numerical wave solutions in the section that follows.

To derive these conditions, consider an infinite, homogeneous, and isotropic medium, characterized by a frequency-dependent permittivity  $\epsilon(\omega)$ , and a magnetic permeability equal to that of vacuum. In such a medium, plane waves are Maxwell solutions that comprise representation of translation symmetry in space and time. Mathematics tells us that this fact alone ensures that such solutions must take the following special form

$$\begin{aligned} \mathbf{E} &= \mathbf{E}_a \exp[i\mathbf{k} \cdot \mathbf{r} - i\omega t] \\ \mathbf{H} &= \mathbf{H}_a \exp[i\mathbf{k} \cdot \mathbf{r} - i\omega t] \\ &\dots \end{aligned} \quad (1.27)$$

At this point, the parameters  $\omega$  and  $\mathbf{k}$  are to be specified such that the above can satisfy the Maxwell equations. In other words, these equations are our ansatz. The next question is what are admissible values for  $\mathbf{k}$  and  $\omega$  such that we can find nontrivial solution for  $\mathbf{E}_a, \mathbf{H}_a$ ? Inserting the ansatz into

$$\begin{aligned}\nabla \times \mathbf{H} &= +\epsilon_0 \epsilon \partial_t \mathbf{E} \\ \nabla \times \mathbf{E} &= -\mu_0 \partial_t \mathbf{H},\end{aligned}\tag{1.28}$$

one obtains a homogeneous linear system for the wave amplitudes

$$\begin{aligned}\mathbf{k} \times \mathbf{H}_a &= -\epsilon_0 \epsilon \omega \mathbf{E}_a \\ \mathbf{k} \times \mathbf{E}_a &= +\mu_0 \omega \mathbf{H}_a\end{aligned}\tag{1.29}$$

This is a system with zero right-hand-side (keep in mind that both  $\mathbf{E}_a$  and  $\mathbf{H}_a$  are vector “unknowns”). As such, it must have a zero determinant to have a non-trivial solution. It is straightforward to set up the corresponding  $6 \times 6$  matrix, and evaluate its determinant. However, an easier way to calculate the solvability condition is by elimination of one of the unknown amplitudes, for example  $\mathbf{E}_a$ . One can do this by applying  $\mathbf{k} \times$  from left onto the first row and then combining the two resulting equations to remove  $\mathbf{E}_a$ :

$$\mathbf{k} \mathbf{k} \cdot \mathbf{H}_a + \left[ \frac{\omega^2 \epsilon}{c^2} - k^2 \right] \mathbf{H}_a = 0.\tag{1.30}$$

The first term must vanish as a consequence of the magnetic divergence equation. The second gives the dispersion relation

$$\frac{\omega^2 \epsilon(\omega)}{c^2} - k^2 = 0\tag{1.31}$$

as a *solvability condition* for amplitudes of the field solutions. Of course, should one eliminate the magnetic amplitude first, the corresponding solvability condition turns out to be the same. The reason for this is because the elimination approach is equivalent to the one based on the determinant of the system.

In what follows, the above will be referred to as the physical, or continuum dispersion relation for plane wave solutions of Maxwell equations. All numerical solutions will have their own dispersion relations. Although they will not only reflect the physical model (i.e. Maxwell) but also the properties of the concrete numerical scheme to produce the wave solutions. It will be important to obtain the numerical dispersion relation and compare it to that above as a means to judge the accuracy and other important numerical properties.

#### 1.2.4 Numerical dispersion relation as a solvability condition

Next we demonstrate a procedure that we will repeat, at least in broad strokes, time and again when we develop various beam-propagation schemes. What we aim to do next is to understand two issues related to the given numerical algorithm. Namely, accuracy and stability.

The notion of accuracy is quite intuitive. In more quantitative terms it means how the error (i.e. the gap between exact and numerical solutions) depends on the numerical parameters  $\Delta t$  and  $\Delta x$ . What we usually look for is the determination of the so-called accuracy order; For example the Yee scheme described above is second order accurate both in space and in time. This simply means that errors “suffered” by the method during a single-step update are proportional to, or scale as  $\Delta x^2$  (or  $\Delta t^2$ ).

As for the numerical stability, it may seem a little artificial notion at first. However, suffice a few encounters with it in real numerical work, and even beginners will become painfully aware

of its importance. It is said that a numerical scheme is stable if local errors do not accumulate and rather remain bounded as the algorithm continues to evolve the solution. If the scheme happens to be unstable then sooner or later (indeed very soon in practice!) errors start to grow exponentially. This error amplification is so fast that all one may have a chance to notice is that the solution suddenly turns into numerical junk and often exhibiting arbitrarily large values.

Notion that is closely related to accuracy and stability is numerical dispersion. Dispersion in real waves is behavior dependent on their wavelength or frequency, and this is similar in numerically calculated waves. However, in numerical waves it comes in two flavors. First, it originates in the physical model and is in that sense real. The second source is purely numerical. It depends on the grid resolution, integration step and the type of the numerical method.

Numerical dispersion can have quite serious consequences. It can even determine whether a given method is applicable under certain conditions. For example, numerical dispersion in direct Maxwell solvers is so strong that their applications in nonlinear optics that require accurate capture of phase-matching relations is utterly impractical.

One effective method to investigate accuracy, stability and numerical dispersion is the so-called plane-wave approach. This is applicable to wave-propagation problems in homogeneous media. We will first collect our Yee-scheme update equations in the following form

$$\begin{aligned} \frac{E(x, t) - E(x, t - dt)}{dt} &= \frac{+H(x + dx/2, t - dt/2) - H(x - dx/2, t - dt/2)}{dx} \\ \frac{H(x - dx/2, t/2) - H(x - dx/2, t - dt/2)}{dt} &= \frac{-E(x, t) + E(x - dx, t)}{dx}, \end{aligned} \quad (1.32)$$

and will seek their solution in the form of plane waves. Plane waves are basis for representation of the translational symmetry in space and time, which means that if a solution is a plane-wave initially, it will remain to be a plane-wave for all subsequent times. This is true for real-life waves as well as for the numerical ones.

An important issue to keep in mind is the linearity of the above equations; A particular solution may be considered to be a part of the total solution. So the solutions investigated next can be viewed as a perturbations on the background of the complete physical solution. In other words, they can represent an "error". Our task is then to understand how this error will propagate and if it will stay bounded or will grow.

The above equations are translationally invariant, as one should expect, since they were derived for a homogeneous one-dimensional medium. To verify this symmetry note that the update for a given spatial and temporal grid-point only depends on the values in the neighborhood. The dependence is characterized by constant coefficients. This is why one can postulate a plane-wave solution as an initial condition for a numerical wave, and parametrize it by two parameters, namely wavenumber and angular frequency. The wavenumber  $k$  controls the wavelength in space regardless of how it is chosen. The translational invariance of the algorithm guarantees that the wavelength remains preserved upon numerical propagation. Similarly, if the solution was harmonic with an angular frequency  $\omega$  up to a certain point in time, it must remain as such also later. This is the manifestation of the invariance with respect to the shift in time (which in turn is related to the energy conservation).

Thus, we seek solutions in a form that reflects the translation symmetry in space and time. These are plane waves for the electric and magnetic fields with amplitudes  $E_0$  and  $H_0$ , respectively:

$$E = E_0 \exp[-i\omega t + ikx] \quad H = H_0 \exp[-i\omega t + ikx]. \quad (1.33)$$

The amplitudes are unknown constants for now, and we need to find their values. Inserting this ansatz in the above numerical update equations, one obtains

$$\begin{aligned} \frac{E_0 - E_0 e^{+i\omega dt}}{dt} &= \frac{+H_0 e^{+i\omega dt/2 + ikdx/2} - H_0 e^{+i\omega dt/2 - ikdx/2}}{dx} \\ \frac{H_0 e^{-i\omega dt/2 - ikdx/2} - H_0 e^{+i\omega dt/2 - ikdx/2}}{dt} &= \frac{-E_0 + E_0 e^{-ikdx}}{dx} . \end{aligned} \quad (1.34)$$

This is nothing but a homogeneous system of linear equations for the amplitudes  $E_0, H_0$ . For this linear homogeneous system to have a non-zero solution the determinant of the corresponding matrix must be equal to zero, and the latter evaluates to:

$$\sin\left(\frac{\omega dt}{2}\right)^2 = \left(\frac{dt}{dx}\right)^2 \sin\left(\frac{kdx}{2}\right)^2 . \quad (1.35)$$

This is a condition which selects compatible pairs of  $\omega, k$ . It is a *numerical dispersion relation* for plane-wave solutions on the grid. It carries information about the accuracy and stability of the Yee scheme, which will be analyzed next.

### Stability

Suppose there is a perturbation of the solution in a form of a plane-wave with some given wavenumber. The question is whether this perturbation will diminish, or if it will grow. The dispersion relation determines the angular frequency that the numerical evolution algorithm will generate spontaneously. It is obtained from the above equation as

$$\omega(k) = \pm \frac{2}{\Delta t} \arcsin\left[\frac{c\Delta t}{\Delta x} \sin\left(\frac{k\Delta x}{2}\right)\right] . \quad (1.36)$$

This will give a real value for  $\omega(k)$  only if the coefficient  $c\Delta t/\Delta x$  is less or equal to one. Indeed, if it was not,  $\omega$  would acquire imaginary part for a sufficiently short wavelength  $\lambda = 2\pi/k$ . The shortest wavelength representable on a discrete grid is equal to  $2\Delta x$  — at least for wavelengths in its vicinity the above solutions would be imaginary. An angular frequency with a negative imaginary part would generate a numerical wave with an exponentially increasing amplitude. Numerical noise alone would be sufficient to “seed” or start this to develop and soon this perturbation would take over the whole solution. This is the essence of numerical instability. Therefore in order to have a stable numerical scheme the ratio  $c\Delta t/\Delta x$ , also called the Courant ratio, must be smaller than one. It is to be emphasized that this threshold for stability depends on the dimension of the space in which waves propagate. Also, stability alone does not guarantee accuracy, it is rather a necessary condition for having a solution at all. An important point that beginning practitioners must appreciate is that manifestations of instability can not be avoided even if the initial solution does not contain the offending waves that tend to grow without bounds. If the numerical scheme is unstable it *will* degrade the solution sooner or later.

### Accuracy

One useful way to assess the accuracy of the numerical scheme is to compare the numerical dispersion to that of the continuum physical model. In the case discussed here, i.e. a one-dimensional electromagnetic wave propagation in vacuum, the continuum dispersion relation reads

$$\omega(k) = \pm ck . \quad (1.37)$$

This has to be compared with Eq.(1.36) which is a discrete version that reflects the fact that the wave propagates through a discrete grid in discrete time-steps. At a first sight the two counterparts may look rather different. However, taking the continuum limit means that the wavelength  $2\pi/k$  must be large in comparison with the grid spacing  $\Delta x$ . Then, the right hand side of the dispersion relation becomes

$$\omega(k) = ck + \frac{k^3}{24}(c^3\Delta t^2 - c\Delta x^2) + \dots, \quad (1.38)$$

so the continuum limit is properly reproduced by the Yee scheme, as the first term is the vacuum dispersion and the rest vanishes when the wavelength is large. The correction term gives a simple way to estimate how big the deviation between the behavior of the real and numerical waves is. It also shows how these deviations depend on the choice of the grid resolution and the integration step.

Yet another way to measure accuracy of numerical wave propagation algorithm is to evaluate the phase and group velocities and compare them to those of the continuum system. Recall that the phase and group velocities are related to the dispersion relation expressed through the wave-number as a function of angular frequency as follows:

$$(v_p)^{-1} = \frac{k(\omega)}{\omega}, \quad (v_g)^{-1} = \frac{\partial k(\omega)}{\partial \omega}. \quad (1.39)$$

---

**Exercise:** Express the dispersion relation of the one-dimensional Maxwell system as  $k(\omega) = \dots$

- Derive analytic expressions for the phase and group velocities.
  - Evaluate and compare these with the continuum limit(s) as functions of the grid spacing for several choices of Courant ratios.
- 

It is left to the reader as an exercise to explore how these quantities compare to their continuum limit counterparts in the simple one-dimensional case. The rest of this section will concentrate on the three-dimensional Maxwell solver algorithm based on the Yee scheme and will also briefly review results for stability and accuracy. An additional notion we need to look into is the numerical anisotropy.

### 1.2.5 3D FDTD Yee scheme: numerical wave properties

There are many textbooks detailing the derivation of the Yee scheme equations for the three-dimensional Maxwell solver. The following equations represent one complete integration step:

$$\begin{aligned} H_{x(i,j,k)}^{n+1/2} &= H_{x(i,j,k)}^{n-1/2} - \frac{\Delta t}{\mu} \left( \frac{E_{z(i,j+1,k)}^n - E_{z(i,j,k)}^n}{\Delta y} - \frac{E_{y(i,j,k+1)}^n - E_{y(i,j,k)}^n}{\Delta z} \right) \\ H_{y(i,j,k)}^{n+1/2} &= H_{y(i,j,k)}^{n-1/2} - \frac{\Delta t}{\mu} \left( \frac{E_{z(i,j,k+1)}^n - E_{z(i,j,k)}^n}{\Delta z} - \frac{E_{x(i+1,j,k)}^n - E_{x(i,j,k)}^n}{\Delta x} \right) \\ H_{z(i,j,k)}^{n+1/2} &= H_{z(i,j,k)}^{n-1/2} - \frac{\Delta t}{\mu} \left( \frac{E_{x(i+1,j,k)}^n - E_{x(i,j,k)}^n}{\Delta x} - \frac{E_{y(i,j+1,k)}^n - E_{y(i,j,k)}^n}{\Delta y} \right) \end{aligned} \quad (1.40)$$

$$E_{x(i,j,k)}^{n+1} = \frac{1 - \frac{\sigma \Delta t}{2\epsilon}}{1 + \frac{\sigma \Delta t}{2\epsilon}} E_{x(i,j,k)}^n + \frac{\frac{\Delta t}{\epsilon}}{1 + \frac{\sigma \Delta t}{2\epsilon}} \left( \frac{H_{z(i,j,k)}^{n+1/2} - H_{z(i,j-1,k)}^{n+1/2}}{\Delta y} - \frac{H_{y(i,j,k)}^{n+1/2} - H_{y(i,j,k+1)}^{n+1/2}}{\Delta z} \right)$$

$$\begin{aligned}
E_{y(i,j,k)}^{n+1} &= \frac{1 - \frac{\sigma \Delta t}{2\epsilon}}{1 + \frac{\sigma \Delta t}{2\epsilon}} E_{y(i,j,k)}^n + \frac{\frac{\Delta t}{\epsilon}}{1 + \frac{\sigma \Delta t}{2\epsilon}} \left( \frac{H_{x(i,j,k)}^{n+1/2} - H_{x(i,j,k-1)}^{n+1/2}}{\Delta z} - \frac{H_{z(i,j,k)}^{n+1/2} - H_{z(i-1,j,k)}^{n+1/2}}{\Delta x} \right) \\
E_{z(i,j,k)}^{n+1} &= \frac{1 - \frac{\sigma \Delta t}{2\epsilon}}{1 + \frac{\sigma \Delta t}{2\epsilon}} E_{z(i,j,k)}^n + \frac{\frac{\Delta t}{\epsilon}}{1 + \frac{\sigma \Delta t}{2\epsilon}} \left( \frac{H_{x(i,j,k)}^{n+1/2} - H_{x(i-1,j,k)}^{n+1/2}}{\Delta x} - \frac{H_{z(i,j,k)}^{n+1/2} - H_{z(i,j-1,k)}^{n+1/2}}{\Delta y} \right)
\end{aligned}$$

This update-scheme is written in a general form that allows for different grid resolutions  $\Delta x, \Delta y, \Delta z$  in the three space dimensions. As shown, the scheme also accounts for a medium characterized by the dielectric permittivity  $\epsilon$ , electric conductivity  $\sigma$ , and magnetic permeability  $\mu$ . The triplets in the brackets are array indices that indicate the cell within a cubic computational box. Whilst the upper index labels the time integration step.

These update-scheme equations are the basis from which one can deduce all properties of numerical waves. The method is essentially the same as one we have used in the one dimensional case.

### Numerical dispersion

The three-dimensional counterpart of the (vacuum) dispersion formula

$$\sin\left(\frac{\omega \Delta t}{2}\right)^2 = \left(\frac{c \Delta t}{\Delta x}\right)^2 \sin\left(\frac{k_x \Delta x}{2}\right)^2 + \left(\frac{c \Delta t}{\Delta y}\right)^2 \sin\left(\frac{k_y \Delta y}{2}\right)^2 + \left(\frac{c \Delta t}{\Delta z}\right)^2 \sin\left(\frac{k_z \Delta z}{2}\right)^2 \quad (1.41)$$

says that each spatial dimension adds the same term (to the dispersion relation) as the one we have derived for the one-dimensional case.

---

**Exercise:** Starting from the three-dimensional Yee scheme equations, derive the dispersion relation (1.41). Hint: This is a lengthy calculation and you may want to use some computer algebra system. The procedure is the same as in one dimension, at least in principle. The dispersion relation can be obtained as a condition for vanishing determinant of the linear system for vector amplitudes of the electric and magnetic field. To set up this system, start with an ansatz for a vector plane wave characterized by an angular frequency  $\omega$  and a wave vector  $\mathbf{k} = (k_x, k_y, k_z)$ . The unknowns of the linear system will be the six amplitudes  $E_x, E_y, E_z, H_x, H_y, H_z$ .

---

The dispersion relation now implies that the condition for stability reads

$$\Delta t < \frac{1}{c \sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}}} \quad (1.42)$$

Thus, in three dimensions, the constraint on the temporal integration step is somewhat steeper than in one dimension. It also depends on different resolutions in the three spatial dimensions with the finest affecting the stability criterion most.

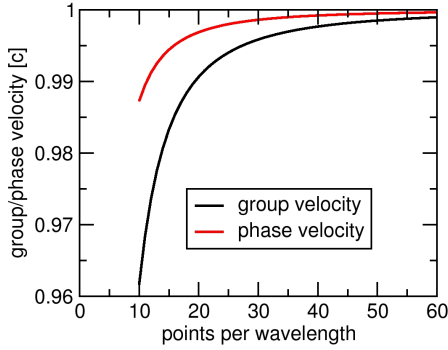
---

**Exercise:** Starting from the three-dimensional dispersion relation for the Yee scheme, derive the stability condition. Hint: it is sufficient to consider the “most extreme” waves which become unstable first.

---

Next, let us look briefly at the accuracy of the Yee scheme in three dimensions. The figure below shows how the numerical phase and group velocity depends on the grid resolution. The latter is expressed as number of grid points that span a length equal to the wavelength of a plane

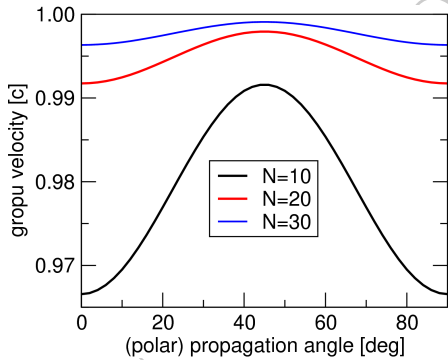
wave. The two curves are calculated for propagation in numerical vacuum and the continuum limit values are both equal to  $c$ . It is a common rule of thumb that a well-resolved Maxwell simulation would use some thirty points per wavelength. Note that the picture shows that even for a much better resolution there remains quite a gap between the continuum limits and what the numerical solver will produce in terms of wave velocities.



Numerical phase and group velocity of waves simulated by the Yee scheme. In the ideal case both of these curves would be constant, equal to one. The difference between the group and phase velocity indicates that the “simulated vacuum” acquires strong chromatic dispersion.

Naturally, it is a matter of context whether or not the accuracy demonstrated in this illustration is sufficient. One particular application in which the artificial numerical dispersion introduced by the Yee-scheme based direct Maxwell solver is unacceptable is nonlinear optics. This is especially true in situations in which phase matching plays a role, such as in harmonic radiation or in four-wave mixing. In such regimes, the outcome of nonlinear interactions depends sensitively on the mutual phase relations between interacting waves. If the solver modifies them by “only” a fraction of a percent the simulated result may be far from a reliable model of reality.

**Numerical anisotropy** Having seen that the numerical wave velocity depends so sensitively on the grid resolution, it should not come as a big surprise that the computational grid acts as an anisotropic medium. Indeed, the effective discreteness of the lattice the wave experiences is different along the grid axis and along the diagonals, for example. This is illustrated in the following figure:



Finite grid-resolution manifests as an artificial anisotropic medium. Wave-packet propagation velocity depends on the direction of propagation and in particular on its orientation with respect to the axes of the computational box. The variation in the group velocity depends on both the integration step (in the case shown  $\Delta t/\Delta x = 1/2$ ) and on the wavelength of the wave (here labeled as  $N$ , the number of grid points spanning one wavelength).

The numerical anisotropy adds to the issue of numerical dispersion. In general, grid resolution refinement is a straightforward way to control both accuracy and dispersion. However, the above

examples should make it clear that the price to pay for a sufficient accuracy in a phase matching regime may be too high for practical applications. One example of a situation where this is the case is nonlinear interactions in ultra-short laser pulses. It is left to the Reader as an exercise to obtain a qualitative estimate of the computational time and memory resources required for such simulations.

### 1.3 **Practice track:** Numerical properties of FD Maxwell solvers

#### 1.3.1 Initial conditions, stability, and numerical dispersion

**Purpose:** This exercise illustrates numerical issues relevant for construction of initial conditions of Maxwell equations. Utilizing a simple Maxwell solver, dispersion and stability properties of numerical waves are examined and compared with theoretical predictions.

**Take-away message:**

- Unlike in BPM, initial conditions in Maxwell solvers require that the magnetic and electric fields are properly “orchestrated.”
- Proper relation between the initial values of the electric and magnetic fields is determined by the direction of propagation of the initial waveform, and is influenced by the dispersion properties of the algorithm.
- The integration step of the numerical algorithm must be sufficiently small in order to avoid instability.
- Carefully designed numerical experiment is required to verify the theoretical numerical dispersion relation.

**Tasks:**

**A)** Implement one-dimensional Maxwell solver based on the Yee scheme. Code the discretized equation derived in the class. To keep the program as concise as possible, assume periodic boundary conditions (PBC). PBC mean that a point at the right-most end of the computational domain can be identified with the very first point at the left end of the domain:

$$E[N - 1] = E[0] \quad , \quad H[N - 1] = H[0] \quad .$$

Here,  $N$  stands for the number of points in the one-dimensional computational “box,” and c-like array indexing is assumed (i.e. the very first array index is zero). Choose the unit of length equal to the grid spacing, and the speed of light equal to one. With such a choice of units, the only free parameter characterizing the method is the Courant ratio  $\Delta t / \Delta x$ . For your initial runs, set the Courant ratio less or equal to 0.5.

To further simplify the programming, you can take advantage of the fact that the update of both,  $H$  and  $E$  field can be done in-place, utilizing the same array holders for the current and new field snapshots. However, keep in mind that this is not possible in higher dimensions and when simulated waves interact with the propagation medium.

**B)** Construct the initial condition in a form of a pulse characterized by a given wavelength. Give the pulse a Gaussian-shaped envelope. To ensure that the spectrum of the waveform is narrow, the envelope must encompass at least several wavelengths. Assume that the electric field component is given by the formula which you will implement. The question is how one needs to choose the magnetic field in order to initialize the solution such that the pulse propagates to the right?

**C)** Execute a short simulation and inspect the outcome carefully to see if the solution indeed propagates in the desired direction. Zoom in so that small features are not missed. Most likely, you will find that while the bulk of the pulse does propagate in one direction, there is a low-intensity “ghost” propagating in the opposite direction.

a) Explain the origin of the ghost

b) Discuss various options to achieve truly clean, one-way propagating initial conditions.

**D)** Verify that the simulation method becomes unstable for the Courant ratio larger or equal to one.

**E)** Design a method, and collect simulation data to verify the theoretic numerical dispersion relation derived in the class. Briefly describe the method of your choice, and discuss specifically what potential numerical artifacts could affect the results. Make sure that your measured numerical dispersion data span the whole region of wavelengths admissible for the discrete grid.

The following figure illustrates two cases of dispersion, and the accuracy you should achieve:

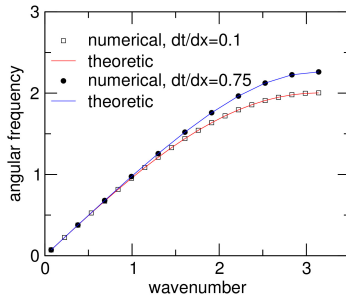


Figure 1. When an initial condition has a sharply-defined spatial wavenumber, the numerical integration algorithm produces a solution with a specific temporal angular frequency. This figure illustrates the dispersion relation which ties the spatial and temporal frequencies. Curves show the expected relation derived from the update scheme, and symbols are results of numerical simulations. Data for different Courant ratios indicate that the dispersion relation depends on  $\Delta t/\Delta x$ . The continuum limit corresponds to a straight line with unit slope.

## Solution

### Task A)

The implementation of the one-dimensional Maxwell integration step is straightforward. Here we use C-language to illustrate that the core of the procedure is nothing but a direct realization of the scheme derived in the class. A complete instructor's solution can be viewed in *OneDMaxwell.c*

**Listing 1.1.** 1D-Maxwell integration step implementation

```
1 void OneStep(double *E, double *H, double dtverdx, int n) {
2     int i;
3
4     for (i=0; i<n-1; i++) E[i] = E[i] + dtverdx*(H[i+1] - H[i]);
5     E[n-1] = E[0];
6
7     for (i=1; i<n; i++) H[i] = H[i] + dtverdx*(E[i] - E[i-1]);
8     H[0] = H[n-1];
9 }
```

Periodic boundary conditions are realized in lines 5 and 8 for electric and magnetic fields, respectively. The way they are implemented here requires one additional grid point where the field sample is a copy “slaved” to the value on the other side of the grid. An alternative method would utilize indexing which wraps the indices that “reach” beyond the array end like this:

$$H[i+1] \rightarrow H[(i+1)\%N]$$

This is computationally more expensive than the addition of an additional boundary grid point.

Note that there seems to be a slight asymmetry in the way electric and magnetic field arrays are indexed. This depends on the chosen correspondence between the index and the location in space it represents. In the present case  $H[i]$  is located on the left from  $E[i]$ .

### Task B)

In the beam propagation method, the initial condition is simply a specification of the electric (vector) field for  $z = 0$ . It is assumed tacitly that the corresponding values of the magnetic field are orchestrated such that the beam solution propagates in the forward direction. In most BPM versions, magnetic fields are never calculated.

In the case of direct Maxwell solver, the issue of the initial condition is slightly more subtle. It is up to the programmer to ensure that the whole (initial) electromagnetic field has desired properties. Often it is the requirement that the initial field represents a pulsed waveform propagating in the direction of positive  $z$ -axis.

The principle that guides the construction of the initial condition is that the relation between the vector amplitudes of the electric and magnetic fields in the numerical solution should mimic that in real electromagnetic plane waves. Because only one-dimensional propagation is considered in this section, the relation simplifies. If one chooses the computational units such that the numerical field samples represent  $E \rightarrow E_y$  and  $H \rightarrow cB_z$  for propagation along  $x$ , then  $H = \pm E$  corresponds to the amplitude relation in a left- and right propagating harmonic wave. The choice of the sign selects the propagation direction along the positive or negative  $x$ -axis direction.

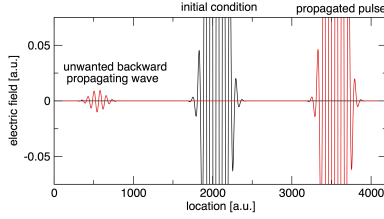
So it seems that if one chooses  $H(x, t = 0) = \pm E(x, t = 0)$  to define the magnetic field in the initial condition the resulting wavepacket should propagate either forward or backward as a whole. However, one must keep in mind that the numerical grid points for the two fields are staggered. This is reflected in the following example:

**Listing 1.2.** Initial condition, take one

```

1  for (i=0; i<N; i++) {
2    /*
3     We aim to orchestrate the magnetic field with the electric
4     such that the initial waveform will propagate to the right
5     Plane-wave properties dictate the opposite signs between
6     the electric field and magnetic field amplitudes
7     */
8
9     double x0 = 0.5*((double) N);
10    double xe = (double) i;
11
12    // same-index magnetic location shifted by half of lattice spacing:
13    double xh = xe - 0.5;
14
15    EF[i] = +InitialElectricField(xe, x0);
16    HF[i] = -InitialElectricField(xh, x0);
17
18    // ... which is almost working ... but not accurately enough ...
19  }
```

Here,  $x0$  only marks the location where we want to place the center of the wavepacket.  $xe$  is a holder for the coordinate (derived from the array index  $i$ ) corresponding to the spatial location of an electric field sample.  $xh$  plays the same role for the magnetic field sample. In line 10, it is shifted left by one half of the spatial grid spacing, and subsequently the same function is used to generate both electric and magnetic field values. The result is that the spatial profile of the electric and magnetic field is the same, as is expected for a one-way propagating electromagnetic pulse (in vacuum).



Inaccurate implementation of the one-way propagating initial condition in a one-dimensional Maxwell solver. The scale of the figure cuts the peaks of the waveform in the initial (black) and propagated (red) pulses, in order to emphasize the weak pulse in the left portion of this figure. This is a wavepacket that propagates in the opposite direction than the bulk of the pulse. It is an unwanted artifact that can not be removed by refining either grid resolution or the integration step.

### Task C)

The figure above illustrates that this implementation of the initial does not work very well. It appears that the initial pulse splits into two identically shaped waveforms that propagate in the opposite directions. An undesired weak pulse is “ejected” from the initial condition. It is left to the reader to verify that it can not be eliminated by better grid resolution or by refining the integration step, because its amplitude decreases with smaller  $\Delta x$  and/or  $\Delta t$  but remains significant for all practically usable parameters.

What is wrong with the above-described realization of the initial condition? The implementation did take into account that the electric and magnetic field grids are staggered in space, but did not account for the fact that they are also staggered in time. The temporal stagger means that the waveform of the magnetic field, which represents time earlier by  $\Delta t/2$ , must be shifted by the distance that the radiation travels during that time. This shift is accounted for in the following modified code:

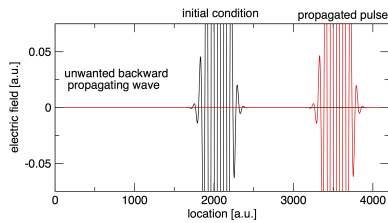
**Listing 1.3.** Initial condition, take two

```

1  for(i=0;i<N;i++) {
2      // here we aim to orchestrate the magnetic field with the electric
3      // such that the initial waveform will propagate to the right
4      // this part is the same as before ...
5
6      double x0 = 0.5*((double) N);
7      double xe = (double) i;
8      double xh = xe - 0.5;
9
10     // ... but one has to account for different time:
11     xh -= 0.5*dtoverdx;
12
13     EF[i] = +InitialElectricField(xe, x0);
14     HF[i] = -InitialElectricField(xh, x0);
15 }

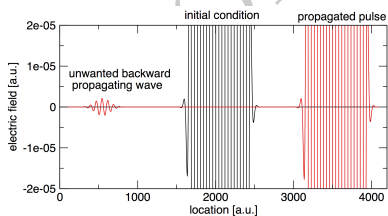
```

The addition is in line 11 where the the coordinate where the magnetic field is evaluated from the function that specifies the electric field initial condition is shifted by  $1/2c\Delta t$  (with  $c = 1$ ).



Better implementation of the one-way propagating initial condition in a one-dimensional Maxwell solver. The scale of the figure is the same as in the previous one. In this case the backward propagating pulse is not visible.

So it seems that the initial condition is properly constructed and does not produce any pulse propagation in the unwanted direction. However, as soon as one zooms into the figure (shown below) the artifact can be readily identified. We thus see that while the initial condition is much better, and the artifact is about three orders of magnitude weaker, it is still not perfect. At this point the origin of this undesirable feature may seem mysterious, but we shall soon understand that it originates in the numerical dispersion; Numerical dispersion is the reason why the propagation speed of the initial pulse is not exactly equal to  $c = 1$ . Because we have assumed just that in our correction of the initial condition, the latter is not completely one-way propagating.

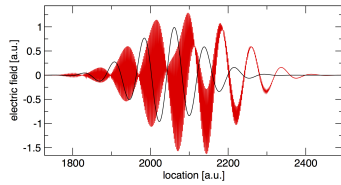


The same data shown at a finer vertical resolution reveals that a very weak pulse propagating in the wrong direction still exists...

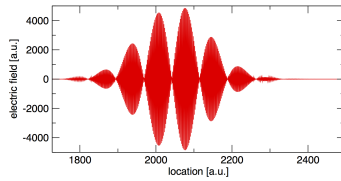
It should be emphasized that the initial condition as constructed in this exercise is sufficiently accurate for practical purposes. After all, in terms of pulse energy, only about one part in  $10^{10}$  propagates in the wrong direction. Nevertheless, the appearance of the artifact, however minute, illustrates that the numerical dispersion, discussed in what follows, affects all aspects of the numerical algorithm.

#### Task D)

For this part of the exercise, we set the Courant ratio equal to, say, 1.01. It is most instructive to keep it quite close to unity, so that one has a chance to observe that instability develops gradually, albeit extremely fast. This is illustrated in the following figures.



Instability onset in Maxwell simulation. One hundred and twenty integration steps were executed with the Courant ratio chosen equal to 1.01, starting from the initial condition shown in solid black. This simulation was interrupted at a time chosen to reveal the very onset of numerical artifacts. As it is often the case, even-odd “oscillations” suddenly appear in the solution (red line) and continue to grow.



Only thirty additional integration steps are sufficient to amplify numerical artifacts by several orders of magnitude. Exponential growth of numerical noise is usually fast enough to make the solution deteriorate so quickly that no gradual transition from a well-behaved, smooth solution can be noticed.

In practice, numerical instability is manifested through solutions that appear noisy and exhibit very large positive and negative values. It may also happen that an unstable solution appears smooth, but grows exponentially.

#### Task E)

There are many valid ways to solve this problem. We will illustrate two approaches, and invite the reader to explore their variations.

The first issue one has to deal with is the setup of a solution with a well-defined wavenumber. The initial condition must ensure that the wavenumber is preserved during the simulation and the evolution will exhibit a single angular frequency — this will only happen if the wavenumber is chosen from the discrete set of values implied by the grid resolution and the computational domain size. Of course this relies on the periodic boundary conditions.

The second issue is how to deal with the fact that waves propagate in both directions. One must either create truly one-way propagating initial condition (as we did in the first part of this exercise) or, alternatively, one can take advantage of a symmetric situation in which the intensity of waves in opposite direction is strictly equal. The second option is simpler, as it only requires that the initial magnetic field is set to zero. Left-right symmetry of the update scheme then ensures that forward and backward component of the waveform is exactly the same. We will

therefore initialize the simulation run with a standing wave with a chosen spatial wavenumber, and follow its evolution to see what angular frequency it will “produce.”

The third problem is a method to extract the numerical value of the angular frequency that the algorithm produces. This can be done in many ways. For example, a direct though not very elegant method is to let the simulation running, and simply count zero-crossings at an arbitrary but fixed location in space. Together with the elapsed time this gives the information needed to calculate the value of  $\omega$ . Another method is to compare solutions after a single step — for this approach one has to derive a formula that relates the solution at two times, and isolate  $\omega$  from it.

Two versions of instructor’s solutions are hidden out of sight in the sub-directory named *Solution*. We describe their main features next. It is strongly recommended that the reader tries to come up with his/her own approach before reading further.

#### *Initialization*

**Listing 1.4.** Initialization for dispersion measurement

```

1
2 // set the Courant ratio , dispersion will depend on this value
3 double dtoverdx = 0.1;
4
5 int el; // scaled spatial wavenumber
6 double k0; // spatial wavenumber – we measure omega for each
7
8 // this starts the main measurement loop
9 for (el=N/2-1; el>0; el-=100) {
10
11 // spatial wavenumber must have one of the grid-supported values
12 k0 = el*2.0*M_PI/((double) N - 1.0);
13
14 // the initial condition has zero magnetic field
15 for (int i=0; i<N; i++) {
16     double xe = (double) i;
17
18     EF[i] = +cos(xe*k0);
19     HF[i] = 0.0;
20 }
```

There are a couple of points to note in the above code snippet. First,  $\Delta t/\Delta x$  is fixed such that the integration scheme is stable. The dispersion curve depends on this value, of course. Then the main loop, controlled by *el*, scans values of spatial wavenumbers for which the angular frequencies are to be measured. The wavenumber value *k0* is carefully chosen in line 12 such that the corresponding plane wave smoothly wraps around the periodic boundary condition of the computational domain. The initial electric field is specified as a harmonic wave, and because the initial magnetic field vanishes, this simulation will evolve a standing wave with the spatial wavenumber *k0*.

#### *Counting zero crossings*

The next code section runs the simulation for a chosen value of *k0*, and counts zero crossing of the electric field at the grid point  $i = 0$ :

**Listing 1.5.** Counting zero-crossings

```

1
2 // this is the spatial point, i=0, where we follow temporal evolution
3 double EFold = EF[0];
4
5 int start = 0; // this variable to indicate the first zero-crossing was detected
6 int istart = 0; // step at which the first crossing was detected
7 int count = 0; // zero-crossing counter
8
9 for(i=0;count<50;i++) { // go until fifty crossings
10     OneStep(EF,HF,dtoverdx,N); // integration step
11
12     // this is where we test for zero-crossing
13     if( EF[0]*EFold <= 0.0 ) {
14         if(start == 0) { start = 1; istart = i;count=0;}
15         else count++;
16     }
17
18     // keep the previous field value to compare with the new one
19     EFold = EF[0];
20 }
21
22 // here, the angular frequency is approximated based on the number of detected
23 // zero crossings
24 i--;
25 double T = 2.0*dtoverdx*(i - istart)/((double) count); // estimated period
26 double omega = 2.0*M.PI/T; // estimated omega
27
28 // theoretical value to show in the output
29 double YeeValue = 2.0/dtoverdx*asin( dtoverdx*sin(k0/2.0));
30
31 printf("%E %E %E\n",k0,omega,YeeValue);
32 } // end of wave-number-scan loop

```

This program can measure the numerical relation between the spatial wavenumber and angular frequency with a good accuracy. There are a few obvious drawbacks. The accuracy of the temporal period estimate,  $T$ , is limited by the (hard-coded in line 9) number of periods to simulate. Thus, to increase the accuracy, length of the simulation run must be increased accordingly. Moreover, the duration of the measurement at low wavenumbers (and low frequencies) increases as  $1/\omega$ . Nevertheless, these are simulation parameters that can be easily controlled and chosen to reflect the accuracy one aims for. Because the one-dimensional Maxwell simulation does not require significant numerical effort, increase in simulation time is not a really serious issue.

It is an easy fix to increase the accuracy of this measurement without increasing the numerical effort. With only fifty period long simulation the the accuracy is limited to a few percent. This can be easily improved by better localization of the first and last zero crossings. For example, one can use simple linear interpolation to estimate the true location of the zero by linear interpolation between the old and new field samples  $EFold$  and  $EF[0]$ . This allows to improve the measurement of the time between zero crossing of the electric field to an accuracy better than  $\Delta t$ . As a result, the the quality of the measurement increases significantly. It is left to the reader to implement this improvement.

*Extracting angular frequency from subsequent field snapshots*

The previous method can be characterized as straightforward. It has a big advantage that the numerical measurement is direct, and does not require additions to the simulation code that would be based on “further development or considerations.” If designed as a test of the algorithm implementation, this is exactly what one wants. On the other hand, the method can be hardly considered elegant. Next we show an alternative measurement of chromatic properties of the Yee algorithm. It only requires a couple of simulation step to execute, and does not suffer any accuracy issues. However, it requires to derive a formula to relate the simulated field after one step to the initial one in order to extract the numerical dispersion relation. Let us derive this formula.

Consider the electric field value, as evolved by the Yee algorithm, at a fixed location. The observation spot can be chosen arbitrarily with one condition that the initial field at this point is not zero. When set up as in the method described before, i.e. with the initial magnetic field equal to zero, the initial value of the electric field, denoted by  $E_0$  corresponds to time  $t = -\Delta t/2$ . The next value at time  $t = \Delta t/2$  is denoted by  $E_1$ , but it turns out to be equal to  $E_1$  since the magnetic field vanishes during the very first update of the electric field. The subsequent step creates the field value denoted  $E_2$ , and it corresponds to time  $t = 3/2\Delta t$ . All three values are related, because they represent harmonic oscillation with an “unknown” amplitude  $A$ :

$$E_0 = A \cos\left(-\frac{1}{2}\omega\Delta t\right) \quad E_1 = A \cos\left(+\frac{1}{2}\omega\Delta t\right) = E_0 \quad E_2 = A \cos\left(+\frac{3}{2}\omega\Delta t\right) \quad (1.43)$$

Thus, it is sufficient to execute two steps to generate  $E_2$ , and  $\omega$  can be calculated from these equations through elimination of  $A$ . Probably the simplest way to do this is to use

$$\cos(a+b) = \cos a \cos b - \sin a \sin b \quad \text{and} \quad \sin(a+b) = \sin a \cos b + \cos a \sin b$$

to obtain

$$E_2 = A \left[ 4 \cos^3\left(\frac{1}{2}\omega\Delta t\right) - 3 \cos\left(\frac{1}{2}\omega\Delta t\right) \right] = E_0 \left[ 4 \cos^2\left(\frac{1}{2}\omega\Delta t\right) - 3 \right] \quad (1.44)$$

Angular frequency  $\omega$  can be expressed from the last relation as

$$\omega = \frac{2}{\Delta t} \arccos \left[ \frac{1}{2} \sqrt{\frac{E_2 + 3E_0}{E_0}} \right] \quad (1.45)$$

Numerical measurement based on this formula can be implemented as shown in the following program listing

**Listing 1.6.** Angular frequency extracted from subsequent simulated field samples

```

1 // listing starts within the main wave-number-scan loop
2
3 // save the initial field
4 EFold = EF[0];
5
6 // execute two steps:
7 // this step does not change electric field due to zero H
8 OneStep(EF,HF,dtoverdx,N);
9 // and this step does...
10 OneStep(EF,HF,dtoverdx,N);
```

```

11
12 // formula XX
13 double omega=2.0/dtoverdx*acos(sqrt((EF[0]+3.0*EFold)/EFold)/2.0);
14
15 // theoretical formula value
16 double YeeValue = 2.0/dtoverdx*asin(dtoverdx*sin(k0/2.0));
17
18 // show results
19 printf("%E %E %E\n",k0,omega,YeeValue);
20 } // end of the wave-number-scan

```

Application of either of the two methods will produce data similar to that shown in Figure 1. Readers should experiment with different parameters for these simulations. In particular, it should become obvious that numerical dispersion becomes more and more pronounced for high-frequency waves. The deviation from the ideal, continuum-limit dispersion decreases with decreasing integration step, but numerical waves with the spatial frequency in the vicinity of the Nquist frequency never propagate as their real counterparts.

### 1.3.2 Absorbing boundary conditions

Boundary conditions for computational domains are an important and difficult numerical and modeling issue. Throughout this course we will revisit this topic a few times, discussing different methods with increasing complexity. The purpose of this exercise is to explore a simple version of the so-called absorbing boundary conditions (ABC) in the one-dimensional Maxwell solver which we have implemented in the previous exercise. While the method examined here does not apply directly to the beam propagation, it will help us to identify important issues related to ABC implementation in the simplest possible setting.

The role of the ABCs is to truncate the computational domain in such a way that waves which reach the edge of the computational box will disappear as if they propagated freely into open space. Needless to say this is not an easy problem to solve. It should also be obvious that there is no ideal implementation of transparent boundary conditions (TBC) as they are also often called. The quality of any particular boundary condition is often characterized in terms of its reflection coefficient. Any numerical wave will partly reflect from the domain boundary, and the reflection coefficient says what is the amplitude of the reflected wave provided the incident has a unit amplitude. Reflectance, or fraction of the wave energy reflected, is also often used. Reflection coefficients are in general functions of the wave frequency (temporal or spatial) and also of the angle of incidence at which the wave approaches the domain boundary.

The method illustrated here is called Mur ABCs. [*Gerrit Mur, IEEE Trans. on Electromagnetic Compatibility, EMC-23 (1981) 377.*] The idea is very simple, at least when applied in one spatial dimension as we do it here. It is based on a local modification of the propagation equation in the region of the computational grid adjacent to the boundary.

The one-dimensional wave equation, which is what the 1D-Maxwell solver effectively simulates, can be exactly factored into two one-way propagation equations:

$$\partial_{tt}E(x,t) - \partial_{xx}E(x,t) = (\partial_t - \partial_x)(\partial_t + \partial_x)E(x,t) = 0. \quad (1.46)$$

Each of the operator factors represents waves propagating in a distinct direction. For example  $(\partial_t - \partial_x)A(x,t) = 0$  is an equation satisfied by waves propagating to the right. If we assume that our boundary condition at the left edge of the computational box is such that it absorbs

outgoing waveforms completely, the wave equation can be replaced by this one-way equation in the vicinity of the boundary. In fact, this “replacement” is only done at a single point, right at the end, where we require:

$$(\partial_t + \partial_x)E(x_{\text{boundary}}, t) = 0 \quad , \quad (\partial_t - \partial_x)E(x_{\text{boundary}}, t) = 0 \quad (1.47)$$

for the right-hand side and left-hand side computational domain boundary, respectively. These boundary conditions in effect say that the wave propagation is forced in the outward direction at both boundaries.

The above equations must be discretized and realized on the grid. This can be accomplished in many different ways. The simplest discretization formula is

$$E_B(t + \Delta t) = E_I(t) + \frac{c\Delta t - \Delta x}{c\Delta t + \Delta x} [E_I(t + \Delta t) - E_B(t)] \quad , \quad (1.48)$$

where subscript  $B$  denotes a field sample located right at the domain boundary, and subscript  $I$  marks its nearest neighbor inside the computational box. Obviously  $\Delta x$  and  $\Delta t$  stand for the spatial grid spacing and integration time step, respectively. To apply this prescription, the inside value  $E_I(t + \Delta t)$  must be evaluated first. So the whole update proceeds as before for all grid points *inside* the domain. The inner update is then followed by calculating the  $E_B$  values on both ends.

Similar boundary conditions could be required for the magnetic field. However, magnetic boundary conditions can be by-passed altogether. We make sure that the boundary domain edges are placed exactly at grid positions that carry electric field samples. With such an arrangement, all magnetic fields have both left-hand and right-hand electric-grid neighbors available for the update step. It is assumed in the following that the computational grid has been chosen accordingly.

This absorbing, or transparent as they are also called, boundary conditions belong to the family of algorithms that estimate the wave-form close to the boundary and assume that the solution is outgoing. This results in a fast, inexpensive algorithm with fairly good properties (low reflectivity coefficient). However, things become more complicated in higher dimensions which bring into play different angles of incidence. More accurate ABCs are possible, for example by using higher-order approximations of the outgoing solution, or by adding so-called perfectly matched layer to the outside of the computational domain proper. We will examine both types of ABCs designed specifically for the BPM methods.

### Exercise: One-Dimensional Maxwell Solver and Absorbing Boundary Conditions

**A)** Starting from the one-way continuum conditions (1.47), derive the finite-difference approximation (1.48).

**B)** Use the Maxwell solver implementation from the previous exercise, and modify its periodic boundary conditions to absorbing boundary conditions.

**C)** Execute a short simulation to demonstrate that the absorbing boundaries work. Test the boundary condition function on both sides of the computational domain. One way to do this simply is to prepare the initial condition such that it will induce two identical pulses propagating in the opposite directions. After a sufficiently long time these pulses reach domain boundaries, and will give rise to weak reflected waveforms.

**D)** There will be small reflection from any numerical absorbing boundary implementation. Identify the reflected pulses in your simulation and estimate the effective reflection coefficient.

**E)** Demonstrate that the effectiveness of the absorbing boundary depends on the wavelength of the incident wave.

### Solution

#### Task A)

When developing a finite-difference approximation for an expression which contains derivatives, the first thing to do is to choose a reference point. It is a location (in space and/or time) at which the finite-difference expression attains its best accuracy. It is important that all terms that contribute to the approximated quantity share the same reference point.

The second important issue is that symmetric approximations to derivatives are superior to one-sided ones. In other words, the symmetric scheme

$$\partial_x f(x) \approx \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \quad (1.49)$$

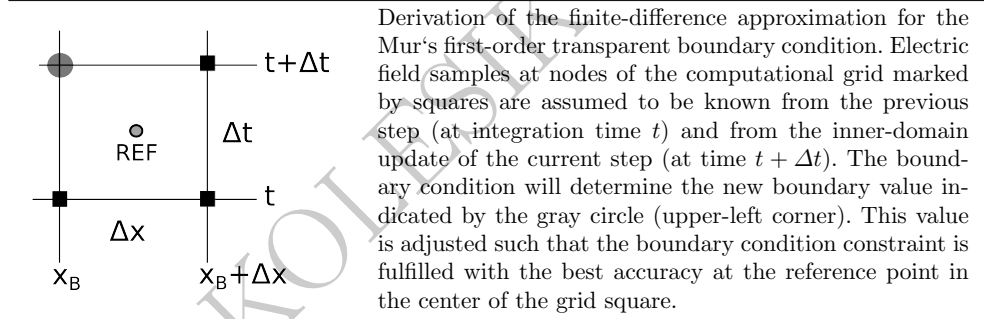
is more accurate than the one-sided

$$\partial_x f(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x} . \quad (1.50)$$

We will re-visit this and similar expressions in the section on finite-difference approximated propagation equations where we will prove that symmetric approximations are always more accurate.

Let us assume that we have calculated all field samples for time  $t$ , together with all inner-domain field samples for time  $t + \Delta t$ , and consider the left-hand side domain boundary. Keeping in mind the “symmetry issue,” the natural candidate for the reference point to construct our finite-difference approximation (1.48) for the left-hand domain edge is the point centered between the nearest field samples:

$$x_{REF} = x_B + \frac{1}{2}\Delta x \quad t_{REF} = t + \frac{1}{2}\Delta t \quad (1.51)$$



The boundary condition we aim to enforce is

$$(\partial_t - \partial_x) E(x_{REF}, t_{REF}) = 0 . \quad (1.52)$$

Of course, we have no field samples available for either  $x_{REF}$  or  $t_{REF}$ . What is often done in such a situation is averaging the derivatives on both sides of the reference point. For the spatial derivative we average its values at the two time-levels

$$\frac{\partial}{\partial x} E(x_{REF}, t_{REF}) \approx \frac{1}{2} \frac{\partial}{\partial x} E(x_{REF}, t + \Delta t) + \frac{1}{2} \frac{\partial}{\partial x} E(x_{REF}, t) . \quad (1.53)$$

Now one can insert symmetric two-point estimates for each spatial derivative to obtain

$$\frac{\partial}{\partial x} E(x_{REF}, t_{REF}) \approx \frac{1}{2\Delta x} (E(x_B + \Delta x, t + \Delta t) - E(x_B, t + \Delta t)) + \frac{1}{2\Delta x} (E(x_B + \Delta x, t) - E(x_B, t)) \quad (1.54)$$

Similar averaging procedure is applied to the temporal derivative

$$\frac{\partial}{\partial t} E(x_{REF}, t_{REF}) \approx \frac{1}{2} \frac{\partial}{\partial t} E(x_B + \Delta x, t_{REF}) + \frac{1}{2} \frac{\partial}{\partial t} E(x_B, t_{REF}) . \quad (1.55)$$

After inserting two-point estimates of temporal derivatives one has

$$\frac{\partial}{\partial t} E(x_{REF}, t_{REF}) \approx \frac{1}{2\Delta t} (E(x_B + \Delta x, t + \Delta t) - E(x_B + \Delta x, t)) + \frac{1}{2\Delta t} (E(x_B, t + \Delta t) - E(x_B, t)) \quad (1.56)$$

With (1.54) and (1.56) we approximate (1.52) and collection of like terms gives us the expression we sought:

$$E(x_B, t + \Delta t) = E(x_I, t) + \frac{\Delta t - \Delta x}{\Delta t + \Delta x} [E(x_I, t + \Delta t) - E(x_B, t)] , \quad (1.57)$$

One missing detail is that no propagation speed ( $c$ ) appears in this formula. This is because we have utilized scaled units in which  $c = 1$ . To recover the the expression for normal units is simple, it suffices to replace  $\Delta t$  with  $c\Delta t$ .

One could repeat the whole procedure for the right-hand-side boundary of the computational domain. It is however not really necessary. Suffice to realize that only two things would change in the whole calculation. First,  $\Delta x$  would change its sign and, second, the sign of the spatial derivative in the boundary condition would also change. These changes would cancel each other and the result is the same.

### Task B)

The implementation in the code is simple. The following listing shows the update procedure, only slightly modified in comparison to its counterpart from the previous exercise. The first lines reflect that the whole update is done “in-place,” i.e. the same arrays holding the current magnetic and electric fields are used to calculate and store the result of the current integration step. This makes it necessary to store values close to the domain boundary.

The second step is the electric field update. Note that the loop leaves out the points  $i = 0$  and  $i = n - 1$  which are the boundary values, and have not all neighbors needed for their update.

Next lines (14,15) apply the absorbing boundary condition as discretized above. Here we use the equivalent formulation using the parameter  $d_{\text{tover}dx} = \Delta t / \Delta x$ . Previously stored auxiliaries are used in place of field samples at time  $t$ . Electric field array values are used for the just updated values for  $t + \Delta t$ .

The lines that follow constitute the update for the magnetic field. Note that the first value stored in the array is not used and is (unnecessarily) set to zero at the end of the procedure.

**Listing 1.7.** 1D-Maxwell update modification for Mur ABC

```

1 void OneStep(double *E, double *H, double dtoverdx, int n) {
2     int i;
3
4     // since we do the update in-place, save these auxiliaries
5     double aux_L_I = E[1];    // this is E_I at left domain end
6     double aux_L_B = E[0];    // this is E_B at left domain end
7     double aux_R_I = E[n-2];  // this is E_I at right domain end
8     double aux_R_B = E[n-1];  // this is E_B at right domain end
9
10    // update inside of the domain as before
11    for(i=1; i<n-1; i++) E[i] = E[i] + dtoverdx*(H[i+1] - H[i-1]);
12
13    // ABCs:
14    E[0] = aux_L_I + (dtoverdx - 1.0)/(dtoverdx + 1.0)*(E[1]
15    - aux_L_B);
16    E[n-1] = aux_R_I + (dtoverdx - 1.0)/(dtoverdx + 1.0)*(E[n-2]
17    - aux_R_B);
18
19    // update inside of the domain as before
20    for(i=1; i<n; i++) H[i] = H[i] + dtoverdx*(E[i] - E[i-1]);
21
22    // this grid point is not used - it is outside
23    H[0] = 0.0;
24 }

```

**Task C)**

In order to test performance of the boundary conditions around both edges of the computational box, we need to direct an incident waveform onto both. A convenient way to achieve this is to generate two identical pulses that propagate in opposite directions. The corresponding initial condition is one that defines electric field in an arbitrary way, coupled with zero magnetic field.

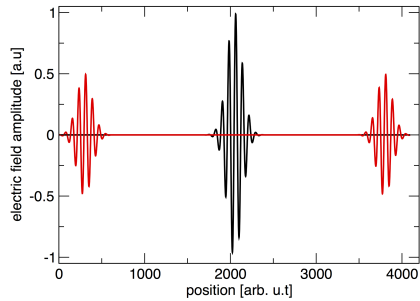
Instructor's solution is included in the working directory as *OneDMaxwell-WithBoundary.c*. Three runs were executed on a grid of 4096 points, integrating for 3500, 4000, and 5000 steps. With the Courant ratio  $\Delta t/\Delta x = 0.5$ , these time correspond to moments before, during and after the waveform hits domain edges.

The following figures show snapshots of the electric field at three different times. The first panel illustrates how the initial condition gives rise to two identical wavepackets, and shows them at the moment just before they hit the computational box boundaries.

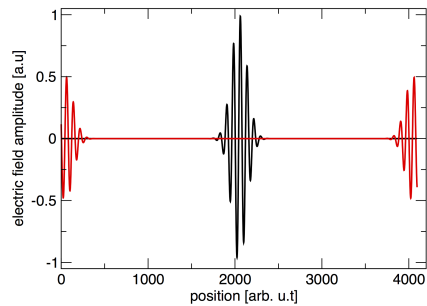
The second panel demonstrates that the boundary condition indeed approximates a half-infinite space in which the incident waveform disappears without deformation. Because the intensity is still significant around the boundary, the figure scale does not allow to appreciate weak reflected radiation.

**Task D)**

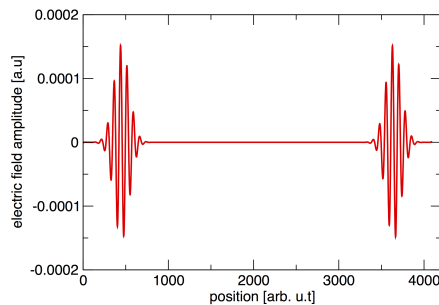
The latter is evident in panel three, which depicts the reflected fields at appropriate scale. Note that the waveforms preserved their shapes upon their (undesired) reflection from the domain edge. However, their amplitude is now significantly lower. The reflection coefficient can be roughly estimated as  $r \approx 1.5 \times 10^{-4}/0.5$  (with 0.5 standing for the amplitude of the incident pulse).



Simulated electric waveform. Black line shows the initial condition, with the magnetic field equal to zero. This initial condition creates two pulses, shown in red at a later time when they propagate toward the edges of the computational box. This picture shows the moment just before they reach the boundary.



Simulated electric waveform. The pulses are being “absorbed” in the transparent boundaries...

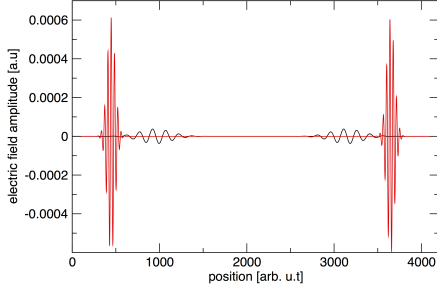


Simulated electric waveform: remnant pulses reflected off the boundaries. Note that the vertical scale is much smaller than the initial one. The reflection coefficient can be estimated roughly as the amplitude of the reflected wavepacket divided by one half, the amplitude of the initial pulse split-off from the symmetric initial condition.

In tests like this one it is often useful to take advantage of the symmetry in the solution. In this case we look for any differences in the behavior of the reflected pulses. The fact that the pulses appear to be mirror images of each other provides an indication that the absorbing boundary conditions work as they should at both ends of the computational box.

### Task E)

Reflection coefficients of numerical boundary conditions are in general dependent on the parameters of the incident waves. In this one-dimensional case, the sole relevant parameter is the wavelength. The following example demonstrates that the differences can be quite pronounced.



Wavepacket reflected from the absorbing boundaries, simulated for two different wavelengths. The amplitude of the long-wavelength (black line) solution is about an order of magnitude smaller than that of the short-wavelength pulse (red line). This is yet another manifestation of numerical dispersion. This time it shows up in the discretized version of the boundary condition which should exhibit no dispersion in the continuum limit. As the pulse wavelength increases, the reflection coefficient improves, because the discreteness of the lattice is less and less evident on the scale given by the wavelength of the waveform.

### 1.3.3 Numerical dispersion curve extracted from noise

We have explored two different ways to measure the resulting numerical dispersion properties of the one-dimensional Maxwell solver implementation. In both cases, we had to sample the dispersion curve point by point by setting up an initial condition(s) with a well-defined spatial frequency. In one case we have to rely on the periodic boundary conditions (in order to ensure that the numerical solution preserved its wave-number), and in the other we used our knowledge of the algorithm stepping procedure. Now imagine that we were given a “black-box” simulator, and did not know anything about its underlying algorithm and/or about the computational boundary conditions — we could specify initial conditions and read off the the result of the evolution after a specified simulated time. Can we still somehow measure the numerical dispersion?

**Purpose:** This exercise introduces an elegant method to obtain the complete dispersion relation of a numerical solver, pretty much independently of the algorithm it employs. In this case, we will recover the 1D Maxwell simulator dispersion curve, i.e.

$$\sin\left(\frac{\omega\Delta t}{2}\right)^2 = \left(\frac{\Delta t}{\Delta x}\right)^2 \sin\left(\frac{k\Delta x}{2}\right)^2,$$

in a single simulation run.

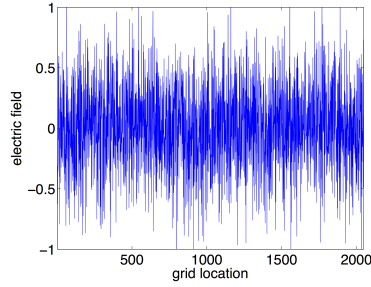
**Method:** The implementation of this method only calls for a very minor modification of the 1D Maxwell solver implemented in the previous assignment. The idea utilizes the fact that wave propagation is linear. It means, that one can initialize all possible types of waves and observe their behavior in a single simulation. Surely, such a simulation must contain information about the propagation of waves of arbitrary wavelengths. The following is the algorithm that allows to extract the numerical dispersion relation from such a simulation.

#### Step 1.

One way to imprint “all” wavelength in the single initial condition supplied to the solver is to create it as white noise (in both electric and magnetic field). Such a chaotic initial field contains short as well as long-wavelength waves that will propagate independently through the lattice.

Step 2.

Starting from the white-noise initial condition, the simulator is run for a fixed (large) number of steps. The following figure shows one snapshot of the evolving electric field:



It is not surprising that field snapshots that are result of evolution from a white noise initial condition all look the same, namely as white noise. Yet, as an ensemble they carry full information about the numerical dispersion relation governing all waves that the simulator can realize.

These electric field snapshots are stored after every integration step. The result is a two-dimensional array (matrix) of

$$M_{i,j} = E(x_i, t_j) .$$

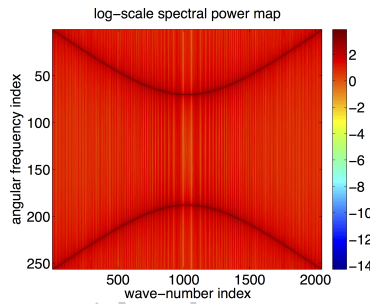
(Note that  $x_i$  and  $t_j$  here stand for discrete “index” values corresponding to the grid nodes and discrete time steps.)

Step 3.

The next step is to calculate two-dimensional Fourier transform of this matrix, to obtain the spectrum

$$\hat{M}_{u,v} = \text{FFT}(M)_{u,v} = \hat{E}(k_u, \omega_v) ,$$

where  $u$  and  $v$  label discrete values of wave-numbers and angular frequency. This picture shows the logarithmic-scale map of  $\hat{M}$ . The lines reveal the numerical dispersion relation.



Two dimensional Fourier spectrum of of the field  $E(x, t)$  amplitude that was initiated by white noise. The dark lines, marking loci where the spectral energy concentrates, show the dispersion relation of the numerical method.

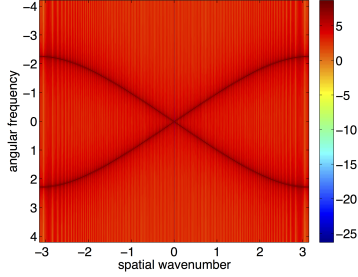
The vertical axis of the above map spans the array of temporal frequencies defined through the Fourier transform of a temporal evolution of the field at each spatial point. With  $NT$  steps executed by the solver, the duration of the temporal domain becomes  $NT\Delta t$ . Consequently the discrete frequencies (sampled by the FFT) are

$$\omega_v = v \frac{2\pi}{NT\Delta t} \quad v = -\frac{NT}{2}, \dots, +\frac{NT}{2}$$

The horizontal axis index labels the allowed wave-vectors

$$k_u = u \frac{2\pi}{NX\Delta x} \quad u = -\frac{NX}{2}, \dots, +\frac{NX}{2}$$

In this physical coordinate system, the spectral power map looks like this (note that this coordinate system transformation can be achieved easily by the Matlab *fftshift* function):



Two dimensional Fourier spectrum of the field  $E(x, t)$  amplitude that was initiated by white noise. Physical coordinates (with  $c = 1$  and  $\Delta x = 1$ ) are used in this map. Waves with approximately realistic propagation properties are now in the central portion of the figure.

It is now straightforward to verify that the loci of spectral power concentration indeed correspond to the numerical wave dispersion relation of the 1D Maxwell solver. The four nearly linear segments emanating from the center represent numerical waves that approximate their physical counterparts accurately. There are four “branches,” because the simulated field is real-valued (and both positive and negative frequencies must be present), and because there are two possible propagation directions. Because the slope of the curve,  $\partial\omega/\partial k$  is the group velocity, we can see that numerical waves with the shortest wavelengths (close to edges of the figure) do not propagate at all!

With the method and expected results described, the exercise has three parts:

#### Task A

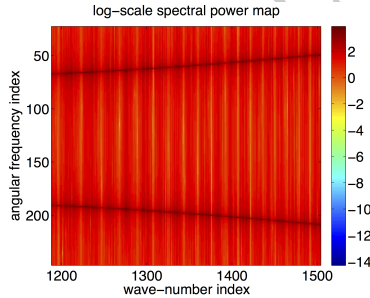
Explain why and how the above-described method works.

#### Task B

Explain how the choice of the simulation parameters  $\Delta x$ ,  $\Delta t$ , and their ratio affect where in the map does the dispersion relation locus appear. Convert the theoretical dispersion relation into the “coordinate system” of the spectrum-matrix  $M$ , and verify the simulation result.

#### Task C

A zoom into the above picture reveals a texture, consisting of semi-regular lines:



A zoom into the power spectrum shows a system of “lines.” Their spacing seems to exhibit definite periodicity, which indicates that this is no random noise. This should raise a red flag...

In the numerical simulation practice it is important not to overlook artifacts — every unexpected feature must be understood: Explain where the lines originate from. Is this effect “mostly harmless,” and the dispersion curve obtained is correct, or is it necessary to remove what must be an artifact?

**Hint:** When trying to decide if something in numerical simulation results is an artifact, it is often useful to list all “un-physical” parameters that control the simulation. In this case, we have  $\Delta x$ ,  $\Delta t$ , their ratio, number of spatial grid points  $NX$ , the number of executed steps  $NT$ , and the random number generator seed and algorithm. Having identified what quantities have the potential to introduce artificial effects, one can study the response of the system to changes of these parameters. More often than not one can uncover clues that help to explain what is going on...

**Hint:** With the above in mind, consider how a given computational domain box affects the trajectories of a wave-packet with a reasonably well defined wave-number.

M.KOLESIK OPT1547/583