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:

$$\partial_{tt}E(x,t) - \partial_{xx}E(x,t) = (\partial_t - \partial_x)(\partial_t + \partial_x)E(x,t) = 0 \ . \tag{1}$$

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:

$$(\partial_t + \partial_x)E(x_{boundary},t) = 0 \quad , \quad (\partial_t - \partial_x)E(x_{boundary},t) = 0 \tag{2}$$

for the right-hand side and left-hand side computational domain boundary, respectively. These boundary conditions in effect say that the wave propagation if 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

$$E_B(t+\Delta t) = E_I(t) + \frac{c\Delta t - \Delta x}{c\Delta t + \Delta x}\left[E_I(t+\Delta t) - E_B(t)\right] \ , \tag{3}$$

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 Δx and Δt 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 (2), derive the finite-difference approximation (3).

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 effectivness 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

$$\partial_x f(x) \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \tag{4}$$

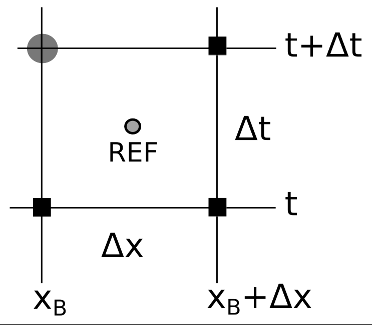
is more accurate than the one-sided

$$\partial_x f(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x} \ . \tag{5}$$

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 the 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 (3) for the left-hand domain edge is the point centered between the nearest field samples:

$$x_{REF} = x_B + \frac{1}{2}\Delta x \qquad t_{REF} = t + \frac{1}{2}\Delta t \tag{6}$$



Derivation of the finite-difference approximation for the Mur's first-order transparent boundary condition. Electric field samples at nodes of the computational grid marked by squares are assumed to be known from the previous step (at integration time t) and from the inner-domain update of the current step (at time $t + \Delta t$). The boundary condition will determine the new boundary value indicated by the gray circle (upper-left corner). This value is adjusted such that the boundary condition constraint is fulfilled with the best accuracy at the reference point in the center of the grid square.

The boundary condition we aim to enforce is

$$(\partial_t - \partial_x)\, E(x_{REF}, t_{REF}) = 0 \ . \tag{7}$$

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

$$\frac{\partial}{\partial x} E(x_{REF}, t_{REF}) \approx \frac{1}{2}\frac{\partial}{\partial x} E(x_{REF}, t+\Delta t) + \frac{1}{2}\frac{\partial}{\partial x} E(x_{REF}, t) \ . \tag{8}$$

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

$$\frac{\partial}{\partial x} E(x_{REF}, t_{REF}) \approx \frac{1}{2\Delta x}\left(E(x_B+\Delta x, t+\Delta t) - E(x_B, t+\Delta t)\right) + \frac{1}{2\Delta x}\left(E(x_B+\Delta x, t) - E(x_B, t)\right) \tag{9}$$

Similar averaging procedure is applied to the temporal derivative

$$\frac{\partial}{\partial t} E(x_{REF}, t_{REF}) \approx \frac{1}{2}\frac{\partial}{\partial t} E(x_B+\Delta x, t_{REF}) + \frac{1}{2}\frac{\partial}{\partial t} E(x_B, t_{REF}) \ . \tag{10}$$

After inserting two-point estimates of temporal derivatives one has

$$\frac{\partial}{\partial t} E(x_{REF}, t_{REF}) \approx \frac{1}{2\Delta t}\left(E(x_B+\Delta x, t+\Delta t) - E(x_B+\Delta x, t)\right) + \frac{1}{2\Delta t}\left(E(x_B, t+\Delta t) - E(x_B, t)\right) \tag{11}$$

With (9) and (11) we approximate (7) and collection of like terms gives us the expression we sought:

$$E(x_B, t + \Delta t) = E(x_I, t) + \frac{\Delta t - \Delta x}{\Delta t + \Delta x} \left[E(x_I, t + \Delta t) - E(x_B, t) \right] \ , \tag{12}$$

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 Δt 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, Δx 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 $dtoverdx = \Delta t/\Delta x$. 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: 1D-Maxwell update modification for Mur ABC

```
void OneStep(double *E, double *H, double dtoverdx, int n) {
  int i;

  // since we do the update in-place, save these auxiliaries
  double aux_L_I = E[1];     // this is E_I at left domain end
  double aux_L_B = E[0];     // this is E_B at left domain end
  double aux_R_I = E[n-2];   // this is E_I at right domain end
  double aux_R_B = E[n-1];   // this is E_B at right domain end

  // update inside of the domain as before
  for(i=1;i<n-1;i++) E[i] = E[i] + dtoverdx*(H[i+1] - H[i  ]);

  // ABCs:
  E[0  ]=aux_L_I+(dtoverdx-1.0)/(dtoverdx+1.0)*(E[1]  -aux_L_B);
  E[n-1]=aux_R_I+(dtoverdx-1.0)/(dtoverdx+1.0)*(E[n-2]-aux_R_B);

  // update inside of the domain as before
  for(i=1;i<n  ;i++) H[i] = H[i] + dtoverdx*(E[i  ] - E[i-1]);

  // this grid point is not used - it is outside
  H[0]   = 0.0;
}
```

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.

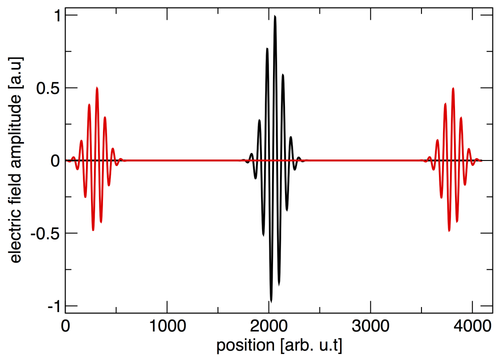
The following figures show snapshots of the electric field at three different times. The first panel illustrates how the initial condition gives rise to two identical wavepackets, and shows them at the moment just before they hit the computational box boundaries.

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.

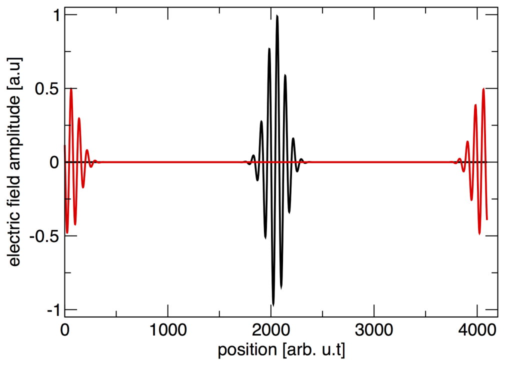
Task D)

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.

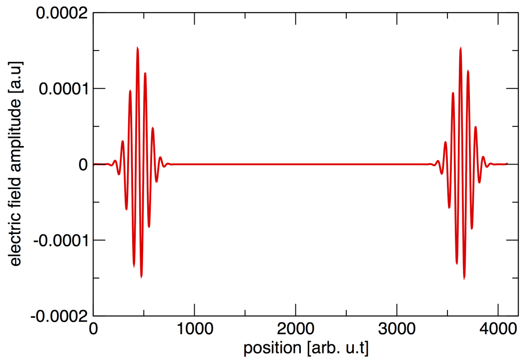
However, their amplitude is now significantly lower. The reflection coefficient can be roughly estimated as $r \approx 1.5 \times 10^{-4}/0.5$ (with 0.5 standing for the amplitude of the incident pulse).



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...

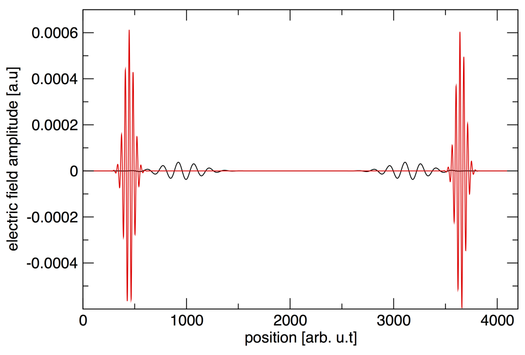


Simulated electric waveform: remnant pulses reflected off the boundaries. Note that the vertical scale is much smaller than the initial one. The reflection coefficient can be estimated roughly as the amplitude of the reflected wavepacket divided by one half, the amplitude of the initial pulse split-off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