1

Boundary conditions, Part I

Summary:

- Implementation of Hadley's absorbing boundary conditions (ABCs).
- Appreciation of both pros and cons of this particular way to handle the edge in a computational domain:
  — it is extremely efficient computationally
  — it works very well in "typical" situations
  — but it is not difficult to cook up regimes that break this ABC-method

1.0.1 Implementation of absorbing boundary using method by Hadley

Task 1: ABC implementation, take one
For this exercise, we can re-use program(s) we wrote for the basic Crank-Nicolson method using a tri-diagonal Thomas algorithm solver. The first task consist in a modification of the treatment of the boundary values as described in the Hadley's paper. As a first step, we shall assume that the division required to estimate the ghost point electric field is always possible, i.e. we hope that during the simulation, our program will not encounter any divisions by zero. Of course, such an approach is nothing to rely on in practice. Here we do this as a part of a numerical experiment to illustrate certain aspects of Hadley's absorbing boundary conditions.

The program section below shows the modification implementing the ABC. This is a sloppy realization which does not guard against division by zeros, and it does not check that the wave is outgoing.

Listing 1.1. 1D Crank-Nicolson BPM with a "shortcut" ABC

```
idelta = 1i*zstep/(4.0*k0*dr*dr);
for row=1:n
  A(row) = -idelta*A(row);
  B(row) = +1.0 - idelta*B(row);
  C(row) = -idelta*C(row);
end

% LHS matrix elements need a modification:
B(n) = B(n) + C(n)*Eold(n)/Eold(n-1);
B(1) = B(1) + A(1)*Eold(1)/Eold(2);
```

```
11
12 % RHS vector needs a modification:
13 LBNDRY = Eold(1)/Eold(2   )*Eold(1);
14 RBNDRY = Eold(n)/Eold(n-1)*Eold(n);
15
16 row = 1;
17 R(row)=
18 Eold(row)+idelta*(LBNDRY-2.0*Eold(row)+Eold(row+1));
19
20 for row=2:n-1
21 R(row)=
22 Eold(row)+idelta*(Eold(row-1)-2.0*Eold(row)+Eold(row+1));
23 end
24
25 row = n;
26 R(row)=Eold(row)+idelta*(Eold(row-1)-2.0*Eold(row)+RBNDRY);
27
28 Enew = TDMAsolver(A, B, C, R);
```

There are two modifications on top of the plain C-N algorithm. Both assume that a "ghost" amplitude sample is added to a virtual grid node just outside of the boundary. This value is represented by the ratio(s) that appear in lines 13, and 14.

The first modification is to the left-hand-side of the linear system solved at each step. In lines 9,10 the off-diagonal matrix elements collect the ghost value from beyond the domain edge, and add them to the diagonal elements. After eventual multiplication of the diagonal element with the corresponding amplitude value, the effect is the same as if we extended the array with the ghost values defined in lines 13,14. Note that this extension is locally explicit, because we guess the "new" amplitude sample with the help of the "old" ones.

The second modification occurs in the right-hand-side evaluation. This is done with the help of the ghost boundary values estimated in lines 13,14. They are used in the very first and very last rows 1 and n.

Instructor's solution: CrankNicolson.m, Main.m

Task 2: Test

Check that the ABC implementation works. One way to do this is to set-up a simulation with an initial beam propagating under angle with respect to the axis, and observe how its amplitude decreases after each bounce from the domain edge. One should be able to verify that the boundary conditions work well as long as the field close to the boundary has a profile that submits well to extrapolation. Figure 1.1 illustrates such a numerical experiment.

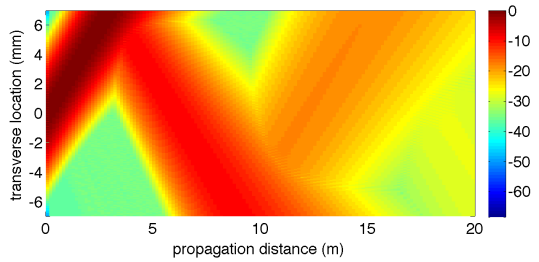

Fig. 1.1. This map shows the logarithmic intensity of a beam that initially propagated at an angle until it reached the boundary of the computational domain, where it suffered residual reflection. This repeats multiple times until all energy from the computational domain dissipates through the absorbing boundary.

In an ideal case, the beam would simply disappear through the edge. In reality, the transparent boundary has a small but nonzero reflectivity. This is what gives rise to the reflected beam seen to emanate from the place where the primary beam hit the boundary. Note that its intensity is orders of magnitude smaller than that of the incident wave, meaning that the absorbing boundary condition works as it should. The color scale in the above figure gives a rough idea about the (intensity) reflection coefficient of the transparent boundary. Taken how little computational effort its implementation adds, and how simple it is to implement, the result is quite impressive.

However, it is possible to create regimes in which ABC will cause instability. In many cases increasing the length of the integration step is sufficient to cause problems. In the case used for this illustration, shown in Fig. 1.2, already a step twice longer leads to an abrupt deterioration of the solution with an exponential growth of artifacts seeded from the computational boundary.

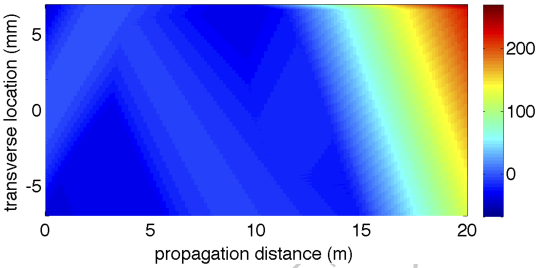

Fig. 1.2. The same simulation executed with a longer (2.5×) integration step. The logarithmic-scale of the beam intensity shows barely visible traces of the beam going through first two reflections from the domain edge, together with the bright region representing exceedingly large values caused by instability.

The figure above shows very clearly that the corruption of the numerical solution occurs at the top boundary of the grid, when the magnitude of the solution becomes small. This is when the solution starts to be dominated by numerical noise, and then it is impossible to tell what is the local wavenumber of the beam incident on the boundary. As a consequence, the algorithm fails and the update scheme becomes unstable.

Instructor‘s solution: CrankNicolson.m, Main.m, including Parameters.m and/or Parameters1.m

Task 3: ABC implementation, take two

Having seen that the simplest implementation of ABCs works most of the time, but occasionally fails in a spectacular fashion, it is clear that the method needs refinement. To handle ABC-unfriendly situations, we will add to the code a test to ensure that divisions are with non-zeros,

and that the estimated local wavenumber has a real part that corresponds to propagation in the direction of the domain edge (i.e. outwards). Note that this addition was in fact contained in the original description given by Hadley.

The listing below shows how the diagonal of the system matrix and the right-hand side vector are both modified on both sides of the domain. The rest of the code remains unchanged. First of all, one needs to make sure that divisions by zero do not happen. If the amplitude in the next-to-out-most grid point is too small, we decide to do nothing, as there is in fact not enough information to establish that the wave behaves as outgoing. This may very well be the case in the initial stages of the simulation, especially with initial conditions that are very small in the vicinity of the computational domain boundary. Interference could also potentially result in a (complex-valued) zero that this implementation must avoid.

The second addition to the code consists in the calculation of the local wavenumber k_x. Only if this has a sign that corresponds to the outgoing wave, we proceed to apply absorbing boundary condition.

Listing 1.2. 1D Crank-Nicolson BPM with Hadley's ABC

```
if( abs(Eold(n-1)) > 1.0e-16 )

  kx = -1i*log(Eold(n)/Eold(n-1))
  if(real(kx)>0)
    B(n) = B(n) + C(n)*Eold(n)/Eold(n-1);
    RBNDRY = Eold(n)/Eold(n-1)*Eold(n);
  end;
end;

if( abs(Eold(2)) > 1.0e-16 )

  kx = -1i*log(Eold(1)/Eold(2))

  if(real(kx)>0)
    B(1) = B(1) + A(1)*Eold(1)/Eold(2);
    LBNDRY = Eold(1)/Eold(2)*Eold(1);
  end;
end;
```

It is left to the reader to verify that this implementation of ABC can execute the same simulation that was failing with the "short-cut" method. With this test for the outgoing wave, the probability that this absorbing boundary condition fails greatly diminishes.

Instructor's solution: CrankNicolsonABC.m, Main.m, including Parameters.m and/or Parameters1.m

1.0.2 Boundary-induced artifacts: Multimode beam propagation

Task 4: Breaking it once more

It is important to understand how a numerical method works and also how it fails. Here we want to visualize an interesting mode of failure in Hadley's boundary conditions. These conditions are designed under an assumption of a single wave incident on the domain edge. The method calculates an auxiliary ghost point field value in a way that is non-linear; as such it does not

respect the superposition principle of the original beam propagation equation, which sounds like a serious violation of the original equation. This suggest that if it happens that two waves with different transverse wavenumbers hit the domain boundary at the same time, the algorithm becomes "confused" because the waveform does not have one dominant transverse wavenumber. Unlike in the previous cases, where the field incident on the boundary was either very weak, or it had a well-defined local wavenumber, the method can fail easily.

To explore this kind of behavior, we can set up an initial condition with two beams propagating at and angle, and position them in such a way that they will reach the domain boundary at the same propagation distance. Figure 1.3 depicts an example of such a beam superpostion.

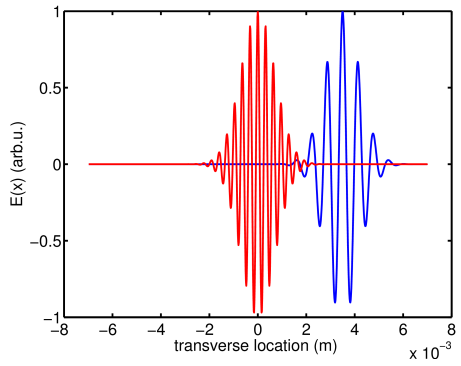

Fig. 1.3. Two-beam initial condition designed to expose the computational boundary to a superposition of waves which does not have a single well-defined transverse wavenumber. Only the real part of the electric field envelope is shown here. The frequency of oscillations in each profile correlates with the propagation angle of the given beam; the second beam shown in read propagates at a steeper angle. It transverse location was adjusted such that both beams reach the boundary at the same time.

This initial condition results in a situation in which there is a superposition of waves with sufficiently different transverse wavenumbers approaching the domain boundary. Such a regime will manifest the nonlinear nature of the implemented transparent boundary — one should see that the reflection is significantly stronger, and that new "frequencies" (i.e. transverse wavenumbers) appear that propagate under angles different from those of the two beams. This is illustrated in the following figure 1.4.

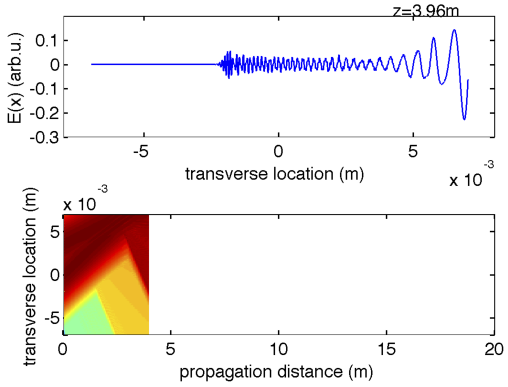

Fig. 1.4. This simulation was stopped shortly after the two beam reached the computational boundary. The top panel shows the snapshot of the (real part) electric field envelope as it propagates to the left. The same instant is shown in the lower panel depicting the logarithmic intensity map. Clearly evident is the fast propagating artifact consisting of waves with higher spatial frequencies. Note that its intensity is still rather high, and that ABC fails in this particular case.

The main reason Hadley's boundary conditions do not work in this particular situation is the superposition of waves that have comparable intensities, and it is therefore impossible to assign one wavenumber to them, even locally at the boundary. This example also illustrates the nonlinearity of this absorbing boundary; it does make a difference if there is one or more waves incident on the computational domain edge.

The take-away lesson here is that while Hadley's transparent boundary method is extremely efficient and applicable in many situations, there are regimes, in particular those characterized by multi-mode propagation, in which this method should not be used. Nevertheless, the ease of implementation together with the negligible computational cost makes this ABC a good candidate to try before taking on a more sophisticated transparent boundary algorithm.

Instructor‘s solution: CrankNicolsonABC.m, Test.m, including Parameters.m