Problem set 6
Non-reflecting Boundary Conditions
and
Perfectly Matched Layers

Daniel Appelö

April 26, 2005

1 Introduction

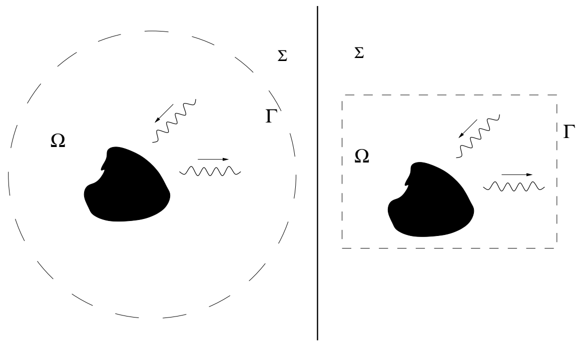


Figure 1: Two possible truncations Ω of an unbounded domain at an artificial boundary Γ. Here Σ is the tail or residual, i.e the union of Σ and Ω is the original unbounded domain.

Many interesting problems appearing in physics, biology and other natural sciences have solutions consisting of waves. An important part of these problems are posed on unbounded domains. To compute a numerical solution to such problems, it is necessary to truncate the unbounded domain, due to finite computational resources. This is done by introducing an non-reflecting boundary Γ, see Figure 1, defining a new domain Ω, which we will refer to as the computational domain. For the problem to be well-posed, it must be closed with a suitable boundary condition on Γ.

To obtain a solution (on Ω) close to the solution on the unbounded domain, the boundary condition on Γ has to be correctly imposed. Usually the non-reflecting boundary Γ is placed in the far field where the solution is composed of waves traveling out of Ω. The fundamental observation is therefore that all reflections caused by the boundary condition on Γ will contaminate the solution in the interior. Hence, an exact boundary condition should prevent all reflections at the boundary as indicated by the name non-reflecting boundary conditions.

The construction of non-reflecting boundary conditions for wave propagation problems has been a matter of ongoing research for over thirty years. The level of difficulty of constructing a particular boundary condition is determined by the underlying problem. It is possible to divide the boundary conditions into four different categories based on the underlying problem. In increasing order of difficulty they are

- Linear time-harmonic wave propagation problems,
- Linear constant coefficient time-dependent wave propagation problems,
- Linear variable coefficient time-dependent wave propagation problems,
- Nonlinear time-dependent wave propagation problems.

Here we will only consider Linear constant coefficient time-dependent wave propagation problems.

2 A hyperbolic system

In this exercise we consider the construction of non-reflecting boundary procedures for the symmetric hyperbolic system

$$\frac{\partial u}{\partial t} + A\frac{\partial u}{\partial x} + B\frac{\partial u}{\partial y} = 0. \tag{1}$$

Many important physical problems can be described by the system (1), e.g. Maxwells equations, the wave equation, the linearized Euler equations and the linearized shallow water equations.

2.1 The wave equation as a first order system

Here we will consider the wave equation in a media with wave speed 1,

$$\frac{\partial^2 v}{\partial t^2} = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}, \tag{2}$$

$$v(x, y, 0) = f(x, y), \quad \frac{\partial}{\partial t}v(x, y, 0) = 0. \tag{3}$$

Assignment

Show that (2) can be written on the form (1) with

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \tag{4}$$

and $u_1 = v$. What relation does u_2 and u_3 have to v?

Hint: take the derivative w.r.t time (x and y) for the first (second and third) row of (1).

2.2 The program bclab.m

You will use the matlab program `bclab.m` to experiment with non-reflecting boundary conditions for the wave equation. The file `bclab.m` can be downloaded from the course directory `/info/ndiff2-05/matlabfiles/`.

The program `bclab.m` solves the system (1) with the A and B as in (4) on the domain $(x, y) \in [-L_x, L_x] \times [-L_y, L_y]$ with the initial data (3). In the downloaded version of the program only periodic boundary conditions are implemented. One of the tasks in this assignment will be to implement non-reflecting boundary conditions. Another task will be to implement a perfectly matched absorbing layer.

Assignment

To get some feeling for the solutions to the initial value problem (2-3) we begin with some simple simulations. Find the lines

```
% Initial Data
f = 0.1*sin(X*pi).*sin(Y*pi);
u1 = reshape(f,N,1);
u2 = zeros(N,1);
u3 = zeros(N,1);
```

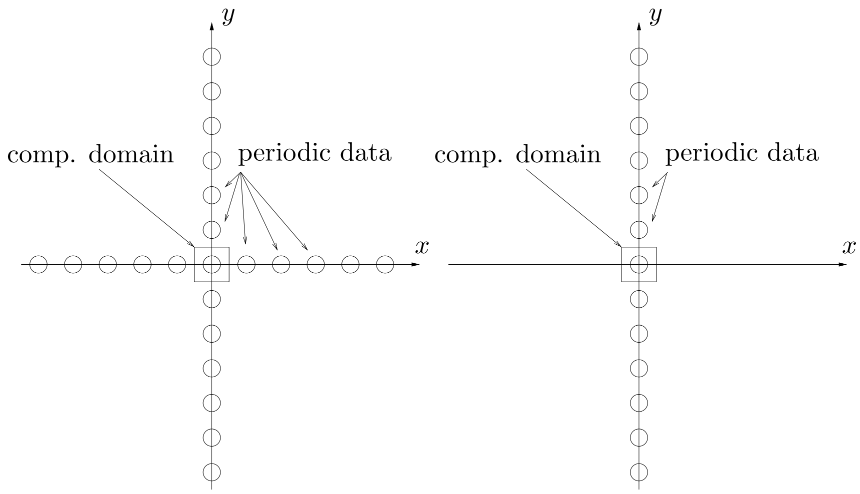
in `bclab.m` and change $f(x, y)$ to a pulse centered in origo

$$f(x, y) = e^{-\frac{x^2+y^2}{\delta^2}}, \ \delta = 0.1. \tag{5}$$

Run the program by typing `bclab` in the matlab command window. You should now be seeing contour plots of the components of the solution. By changing the value of `t_end` you can control the duration of the simulation time. Choose some larger value for `t_end` to see what happens when the solution reaches the boundary. If you don't want to plot every time-step you can change the value of `t_mod`. If you prefer `mesh` instead of `contour` you can change `p_type` to 1.

2.3 Periodic boundary conditions

If we solve (2) with periodic boundary conditions we force the solution to be periodic in space at all times. Specifically it will be periodic at $t = 0$. Therefore if we run `bclab.m` with periodic boundary conditions in both directions the problem we actually solve is (2) with initial pulses (5) not only centered in $(x, y) = (0, 0)$ but centered in $(x, y) = (k\,L_x, 0)$, $k = 0, \pm 1, \ldots$ and $(x, y) = (0, l\,L_y)$, $l = 0, \pm 1, \ldots$, i.e. periodically along the x and y axis (see figure 2(a)).



(a) Periodic initial data in both directions. (b) Periodic initial data along the y axis.

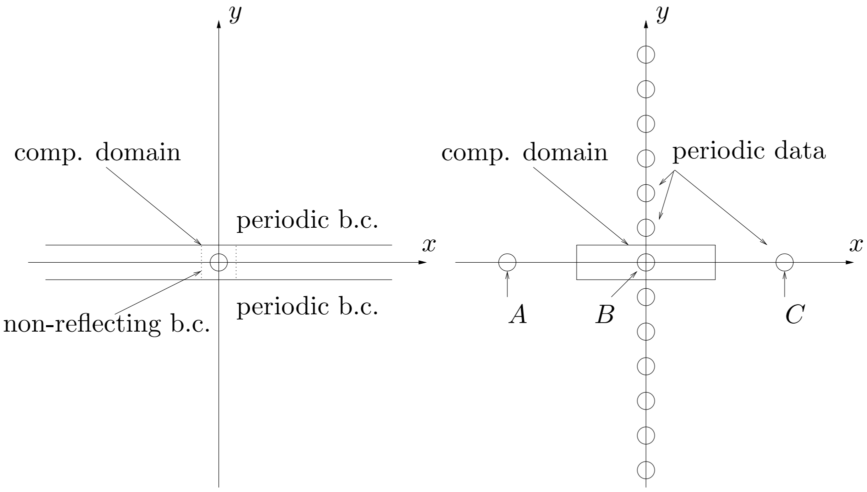
Figure 2: Initial data, periodic in two and one direction.

Assume now that the problem we want to solve is (2) with initial pulses (5) centered in $(x, y) = (0, l\,L_y)$, $l = 0, \pm 1, \ldots$, i.e. periodically along the y axis only (see figure 2(b)).

We can still use periodic boundary conditions in the y-direction for the numerical simulation of this problem. However, since the problem is infinite in the x-direction, we will need to devise some special procedure for the treatment of the boundary $x = \pm L_x$. One option is to impose non-reflecting boundary conditions, see figure 3(a).

Before we discuss non-reflecting boundary conditions we should first understand the main properties that we want such b.c. to possess. Therefore consider a wavefront, caused by the initial data, radiating in the positive x direction towards the line $(x, y) \in L_x \times (-\infty, \infty)$. Since there is no physical boundary at that line, the wavefront will just travel across the line without being reflected, hence the name *non-reflecting* boundary condition.

The most important property is that the b.c. should be non-reflecting. However, seen from a computational point of view, it is also important that it is easy



(a) Problem equivalent with the problem in figure 2(b).

(b) Periodic initial data in both directions with the computational domain extended ion the x direction.

Figure 3: Initial data, periodic in two and one direction.

to implement and efficient in terms of cpu and memory usage.

The first procedure we consider for the computation of an accurate solution to the y-periodic problem is not really a boundary condition, neither is it efficient. It is however very easy to implement. It is also exact, at least for some time.

3 Boundary procedure 1: Making the computational domain bigger

Assume that we are interested of the solution in the domain $(x, y) \in [-1, 1] \times [-1, 1]$. If we compute the solution on the domain $(x, y) \in [-L_x, L_x] \times [-1, 1]$ with periodic b.c. in both direction and take $L_x > 1$ the numerical solution will be equivalent with the solution obtained from initial data positioned as pictured in figure 3(b).

Assignment

From figure 3(b) we can see that waves originating from the initial data A will not influence the solution for $x > -1$ for $t \leq t_c$.

- Estimate the value of t_c if $L_x = 2$ (change the parameter `x_bnd = 2`) by running `bclab` until `t_end = 2`.

If you want to see solution on the full computational domain you can comment the extra `x_bnd = 1` in the `% plotting parameters` section.

- Discuss how the required amount of memory and cpu-time grows as the simulation time grows.
- Extend the discussion to a problem in three dimensions using this method (making the computational domain bigger) instead of a non-reflecting b.c. in all three directions.

It should now be clear that a more sofisticated b.c. than just making the computational domain bigger is required for long simulations. For our first non-reflecting b.c. we will make use of one dimensional theory of characteristics.

4 Boundary procedure 2: Characteristic boundary conditions and diagonalization

Consider the scalar transport equation in one dimension

$$\frac{\partial q}{\partial t} = a\frac{\partial q}{\partial x}, \quad q(x,0) = \begin{cases} g(x) & |x| < l, \\ 0 & |x| \geq l, \end{cases} \tag{6}$$
$$a \neq 0,\ -L < x < L,\ t \geq 0,\ L > l.$$

The solution of (6) is $q(x,t) = g(x + at)$ i.e. the solution is a translation of the initial data in the positive (negative) direction of the x axis if $a < 0$ ($a > 0$), see figure 4. We say that the equation is left-going (right-going) if $a < 0$ ($a > 0$).

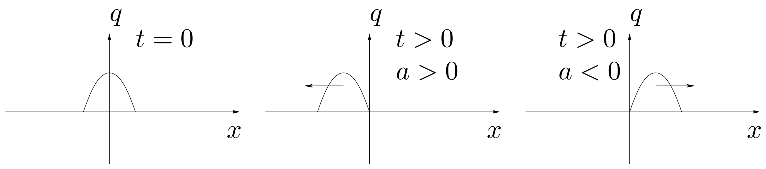


Figure 4: Solution of (6) for $a > 0$ and $a < 0$

The theory of characteristics `REFERENCE` tells us that we must specify waves that are incoming at a boundary for the problem to be well-posed. Hence if the solution is left-going it will be ingoing at $x = L$ and outgoing at $x = -L$ and we need to specify the boundary condition at $x = L$. Since we do not want waves coming in through the boundary we must choose $u(L,t) = 0$, which is compatible with the initial data.

For the numerical simulation of (6) we will need to specify boundary conditions at the outflow boundary to close the numerical scheme. If we for example approximate (6) with the fourth order differential scheme

$$\frac{\partial p_j}{\partial t} = a\big(\frac{4}{3}D_0(h) - \frac{1}{3}D_0(2h)\big)p_j,\ j = 1,2,\ldots,n-2,n-1, \tag{7}$$

we will need two boundary conditions at $x = -L$ and two at $x = L$.

If $a > 0$ the boundary conditions

$$D_+^4 q_0 = D_+^4 q_{-1} = 0, \tag{8}$$

at the outflow (left) boundary and

$$q_n = 0,\ D_+ D_- q_n = 0, \tag{9}$$

at the inflow (right) will be stable and yield a forth order approximation. Here D_+^4 is the one-sided fourth order difference stencil

$$D_+^4 v_j = (-25 v_j + 48 v_{j+1} - 36 v_{j+2} + 16 v_{j+3} - 3 v_{j+4})/12h.$$

If $a < 0$ the boundary conditions

$$D_-^4 q_n = D_-^4 q_{n+1} = 0, \tag{10}$$

at the outflow (right) boundary and

$$q_0 = 0,\ D_+ D_- q_0 = 0, \tag{11}$$

at the inflow (left) can be used. If $a = 0$ the boundary conditions

$$\begin{aligned} q_{-1} &= q_0 = 0, \\ q_n &= q_{n+1} = 0, \end{aligned}$$

can be used.

4.1 Characteristic variables for a system: Diagonalization

If we have a $n \times n$ symmetric hyperbolic system in one dimension

$$\frac{\partial q}{\partial t} = A \frac{\partial q}{\partial x}, \tag{12}$$

we can introduce the new variable $\tilde{q}$ by

$$q = S\tilde{q},$$

where S is a matrix composed by the eigenvectors of A

$$S = [\psi^{(1)}, \ldots, \psi^{(n)}].$$

S transforms A to diagonal form

$$S^{-1} A S = \Lambda \equiv \mathrm{diag}(\lambda_1, \cdots, \lambda_n).$$

In the new variable $\tilde{q}$ the system (12) becomes

$$\frac{\partial \tilde{q}}{\partial t} = \Lambda \frac{\partial \tilde{q}}{\partial x},$$

that is, n decoupled scalar equations

$$\frac{\partial \tilde{q}^{(i)}}{\partial t} = \lambda_i \frac{\partial \tilde{q}^{(i)}}{\partial x}, \quad i = 1, \ldots, n, \tag{13}$$

for which we can apply the boundary conditions discussed in the previous section. The transformation from (12) to (13) is usually referred to as diagonalization of the system (12). The variables $\tilde{q}$ are the characteristic variables of (12).

4.2 Characteristic b.c for the system (1)

Since the characteristic boundary conditions are perfectly non-reflecting in one dimension it is reasonable to assume that they are useful in two dimensions if the wavefront is almost one dimensional.

Assignment

To use characteristic b.c. for our problem we need to diagonalize the system (1).

- Find the matrix S diagonalizing A (from (4)).
- Compute Λ and the matrix $\tilde{B} = S^{-1}BS$.
- Save a copy of the file `bclab.m` in a new file, say `bclab2.m`.
- In the file `bclab2.m` find the lines

```
A=[0 1 0
   1 0 0
   0  0 0];
B=[0 0 1
   0 0 0
   1 0 0];

%%% Assignment X insert values for Lambda and B~.
% A=[
%
%        ];
% B=[
%
%        ];
```

  and insert Λ and $\tilde{B}$.

- Use $\tilde{u}(x, 0) = S^{-1}u(x, 0)$ to specify the original initial data in the new variables. I.e. modify the lines

```
% Initial Data
f = 0.1*sin(X*pi).*sin(Y*pi);
u1 = reshape(f,N,1);
u2 = zeros(N,1);
u3 = zeros(N,1);
w(1:3*N) = [u1;u2;u3];
```

- Change the plotting so you plot the original variables u. Check by running the `bclab.m` and `bclab2.m`, if you have the correct initial data and have done the plotting right the plots should be identical.

Assignment, changing the periodic b.c. to ch. b.c.

The program `bclab2.m` uses the fourth order approximation (7) to approximate $\partial/\partial x$ and $\partial/\partial y$ so you can use the boundary conditions (8-9)or (10-11) depending on the sign of λ_i (For $\lambda_i = 0$ you do not have to change anything). Find the lines

```
% $$$   TO BE MODIFIED FOR LEFTGOING CH. B.C.
% $$$   Dxn(1,??) = ??
% $$$   Dxn(nx,??:??) = [??:??];
% $$$   Dxn(nx-1,??:??) = [??:??];
```

and modify the `??` using (8-9) and then use (10-11) for the lines

```
% $$$   TO BE MODIFIED FOR RIGHTGOING CH. B.C.
% $$$   Dxp(nx,??) = ??;
% $$$   Dxp(1,??:??) = [??:??];
% $$$   Dxp(2,??:??) = [??:??];
```

Depending on how you have defined S you may have to change the lines

```
Dx=[Dxch Z Z
     Z Dxn Z
     Z Z Dxp];
```

which are correct for $\Lambda = \mathrm{diag}(0, \lambda < 0, \lambda > 0)$.

To test your characteristic b.c. change the parameter `char` in `bclab2.2m` to 1. If you have succeeded with the assignment most of the wave from the initial data should disappear from the computational domain thru the x boundaries.

- Compare the solution obtained with the characteristic b.c. on the domain $(x, y) \in [-1, 1] \times [-1, 1]$ at time $t = 2.5$ with the (exact) solution on $(x, y) \in [-1, 1] \times [-1, 1]$ obtained by making computations with `x_bnd=3`.
- Discuss how the required amount of memory and cpu-time grows as the simulation time grows when ch. b.c. are used.
- Extend the discussion to a problem in three dimensions using ch. b.c. as non-reflecting b.c.

5 Boundary procedure 3: Absorbing layers, PML

An alternative to non-reflecting boundary conditions is to surround the computational domain with a finite width, absorbing layer. Typical setups are pictured in figure 5. Ideally, all waves traveling into the layer, independent of frequency or angle of incidence, should be absorbed to such extent that reflections at the outermost boundary are of no importance. Absorbing layers with this ideal properties do exits, and are referred to as Perfectly Matched Layers (PML). They were first introduced in the context of computational electromagnetics, where they are considered as state of the art and are widely used in practical computations.

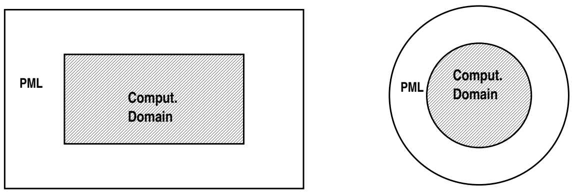


Figure 5: The computational domain is surrounded by an absorbing layer (PML). Different geometries are possible.

5.1 Construction of a PML for the system (1)

In this section we will discuss the construction of a perfectly matched layer for the symmetric hyperbolic system (1) in the strip $(x, y) \in (0, \delta_x) \times (-\infty, \infty)$. By performing a Laplace transform in time and a Fourier transform in y (1) we can define the modal solutions in the computational domain

$$\hat{u} = e^{\lambda x}\hat{\psi}, \quad (-sI + \lambda A + ik_y B)\,\hat{\psi} = 0.$$

Here s is the Laplace transform variable and ik_y.

The layer is constructed by modifying the solutions in the layer so that $\Re\lambda \neq 0$ for $\Re s \geq 0$. If the eigenfunctions $\hat{\psi}$ remain unchanged the layer will be perfectly matched. Taking the modal solutions in the layer to be

$$\hat{u} = e^{\lambda x + \frac{\lambda}{s}\int_0^x \sigma(z)dz}\hat{\psi}(s, ik_y). \tag{14}$$

It can be shown that the modal solution (14) correspond to the system

$$\begin{aligned} \frac{\partial u}{\partial t} &= A(\frac{\partial u}{\partial x} + \sigma w) + B\frac{\partial u}{\partial y}, \\ \frac{\partial w}{\partial t} &= -\frac{\partial u}{\partial x} - \sigma w. \end{aligned} \tag{15}$$

where we have introduced the auxiliary variable w. Formally $\sigma(x)$ can be taken as a constant function (which would give the largest damping). However due to numerical effects the reflection will be significant if σ is large at the interface between the layer and the computational domain. Therefore it is often taken as a monotone positive polynomial taking the value zero at the interface.

Assignment

To implement the PML (15) modify the lines

```
% The right hand side of the system
% To be modified for PML
P=[A*Dx Z
   Z Z]+...
  [B*Dy Z
   Z Z]+...
  [Z Z
   Z Z];
```

in `bclab2.m` ($\sigma(x)$ is denoted by the variable `SIGMA`). To turn on the PML put `sigma_max` to some nonzero value.

- When the PML is implemented, run `bclab2.m` with a 0.3 thick PML (`pml_width = 0.3`, `x_bnd = 1.3`) of strength `sigma_max = 10` until $t = 2.5$ and compare it with the (exact) solution on $(x, y) \in [-1, 1] \times [-1, 1]$ obtained by making computations with `x_bnd=3`. By default $\sigma(x) = \text{const.}$ so you should experience a substantial amount of numerical (spurious) reflections.
- Change $\sigma(x)$ to be a 4th order polynomial by changing the parameter `deg` to 4. Is the solution improved?
- Compare this solution with the solution computed with characteristic b.c.
- Compute the solution at time $t = 10$ for using the PML and haracteristic b.c. How would you expect the exact solution to look? Which of the solutions is likely to be moore accurate?
- Make some other experiments with the PML and report your findings.