MATH 517 PROJECT 3: WAVE EQUATION, PERIODICITY

Turn in your Matlab/Scilab script on Blackboard by the end of Sunday 10/12.

Goal: obtain a numeric approximation to the solution of the PDE $u_{tt} = c^2 u_{xx}$ with Dirichlet boundary condition at the left end of the interval $[0, 3]$, and Neumann boundary condition at the right end.

Answer the following questions, based on output:

(1) What happens when a wave hits Dirichlet boundary (fixed end)?
(2) What happens when a wave hits Neumann boundary (free end)?
(3) What are the highest and lowest values attained by the solution over the period of computation?
(4) How similar is the last computed frame to the first one?

You can include the answers as comments when submitting the file on Blackboard, or as comments within the script.

Data (posted on Blackboard among the grades): propagation speed c.

Method: Use space step $\Delta x = 0.02$ and time step $\Delta x = 0.01$. With the value of c you are given, This will make sure that the stability condition $c\Delta t < \Delta x$ is met.

Set up the matrix of u values (so far, filled with zero), computing its size using the time and space steps, similar to previous projects.

The solution should be on space interval $[a, b]$ with $a = 0$, $b = 3$. Use time interval $0 \leq t \leq T$ where $T = 4(b - a)/c$.

Use the initial position $u(x, 0) = \max(0, (2x - 1)(3 - 2x))$. This formula creates a solitary parabola-shaped wave that is initially located within $[1/2, 3/2]$. The initial velocity $u_t(x, 0)$ will be set to 0.

The Dirichlet condition and the Neumann condition are enforced as in Project 2: see below.

For most of the matrix, the difference scheme

$$U(i+1, j) = 2 * U(i, j) - U(i-1, j) + \left(\frac{c\Delta t}{\Delta x}\right)^2 (U(i, j-1) - 2U(i, j) + U(i, j+1))$$

is used to calculate the solution. Exceptions: this formula should not be applied when j is at the right edge of the matrix, since $j + 1$ would go out of bounds. Instead, replace $j + 1$ by $j - 1$ in this case: this expresses the Neumann boundary condition:

$$U(i+1, j) = 2 * U(i, j) - U(i-1, j) + \left(\frac{c\Delta t}{\Delta x}\right)^2 (U(i, j-1) - 2U(i, j) + U(i, j-1))$$

Sample elements of program. The set up of parameters such as timeStep, numSpaceSteps, etc, is as in Project 2, except of course the numbers are different. You will also need to define $c =$ (the propagation speed) taken from Blackboard. The initial conditions are more involved than for diffusion because the initial velocity must be taken into account to compute the second row of the matrix.

```
initialVel = 0;
for j=1:numSpaceSteps
  x = a+(j-1)*spaceStep;
  u(1,j) = max(0, (2*x-1)*(3-2*x));
  u(2,j) = u(1,j) + initialVel*timeStep;
end
```

Then run the double loop that calculates the solution using a difference scheme:

```
for i = 2 : numTimeSteps-1
  for j = 2 : numSpaceSteps-1
    u(i+1,j) =2*u(i,j)-u(i-1,j)+(c*timeStep/spaceStep)^2*(u(i,j-1)-2*u(i,j)+u(i,j+1));
  end
  j = numSpaceSteps;
  u(i+1,j)=2*u(i,j)-u(i-1,j)+(c*timeStep/spaceStep)^2*(u(i,j-1)-2*u(i,j)+u(i,j-1));
end
```

Finally, plot the solution. For this problem, it is best to plot as an animation, with each frame replacing the previous one. With Matlab this is easy, because this is the default behavior of Matlab plots:

```
for i = 1 : numTimeSteps
  plot(a:spaceStep:b, u(i,:));
  axis([a b -1 1]);    %  keeps window size fixed
  pause(timeStep);   % if your computer is slow, you may want to reduce the pause
end
```

Scilab version:

```
set(gca(),"auto_clear", "on")    //  so that plots replace each other
for i = 1 : numTimeSteps
  plot(a:spaceStep:b, u(i,:))
  xpause(timeStep*1e6)
end
```

To help with Question 4, plot the first row (initial) condition again, using a different color, for comparison with the last row. Matlab version:

```
hold on
plot(a:spaceStep:b, u(1,:), 'r');
hold off
```

Scilab version:

```
set(gca(),"auto_clear","off")
plot(a:spaceStep:b, u(1,:), 'r')
```