Due Monday, May 6, 2012

This project examines the numerical solution of the wave equation in a liquid in one and two dimensions, subject to various boundary conditions. Ultimately you will account for variable depth and see how it changes the propagation of the waves.

1 1-D Wave equation

If one defines $u(x,t)$ to be the vertical displacement of a wire (or a fluid surface), measured from an equilibrium position of zero, then the application of Newton's law (sum of the forces equals mass times acceleration) leads to the basic one dimensional wave equation

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}\,, \tag{1}$$

where c is the propagation speed of the wave, x is the horizontal distance measured from the left side of the wire, and t is time. To solve this we require two initial conditions and two boundary conditions. For initial conditions, we will assume an initial displacement of the wire specified by the function $f(x)$,

$$u(x,0) = f(x)\,, \tag{2}$$

and an initial speed of zero,

$$u_t(x,0) = 0\,. \tag{3}$$

For boundary conditions we will use a known but variable displacement for the left side of the wire,

$$u(0,t) = g(t)\,, \tag{4}$$

and a rather flexible form for the boundary condition on the right side of the wire

$$\alpha u(L,t) + \beta\frac{du(L,t)}{dx} = \gamma\,. \tag{5}$$

By selecting different values for α, β and γ, one can form several different physically realistic boundary conditions. What remains, is to consider the calculations and results that follow for specific functions $f(x)$ and $g(t)$, and particular values of α, β and γ.

1.1 Fixed boundary conditions at both ends

During some of the recent labs you wrote a Matlab script to numerically calculate $u(x,t)$ for a "plucked" wire by solving (1) with specific initial displacement functions $f(x)$, and zero displacement at the left and right sides of the wire, $g(t) = 0$. Using the shorthand notation $u(x_i,t_k) = u_i^k$ we used a simple finite difference approximation for each of the second derivative terms in (1) to arrive at

$$(\,u_i^{k+1} - 2u_i^k + u_i^{k-1}\,)/\Delta t^2 = c^2(\,u_{i-1}^k - 2u_i^k + u_{i+1}^k\,)/\Delta x^2\,, \tag{6}$$

from which one could solve for

$$u_i^{k+1} = \lambda u_{i-1}^k + 2(1-\lambda)u_i^k + \lambda u_{i+1}^k - u_i^{k-1}\,, \tag{7}$$

where $\lambda = (c\Delta t/\Delta x)^2$. In this expression, it was assumed that u values at times t_k and t_{k-1} are known. Hence (7) is valid for $k = 2,3,4,\ldots$. There is, however a problem when trying to calculate u_i^2 values using (7) since although $u_i^1 = f(x_i)$ from the initial condition, there are no u_i^0 values. To solve this problem, we invoked the initial condition (3) and created phantom values u_i^0. Then, by coupling the finite difference forms of (3) and (6) we arrived at the special case

$$u_i^2 = (\,\lambda u_{i-1}^1 + 2(1-\lambda)u_i^1 + \lambda u_{i+1}^1\,)/2\,. \tag{8}$$

Thus to summarize, we get u_i^1 values from the initial condition $u(x,0) = f(x)$, we get u_i^2 values from (8), and finally we get the remaining u_i^{k+1} values from (7) for $k = 2, 3, 4, \ldots$.

Included with this document is a file named `WaveOrig.m` that performs the above calculations. You should probably use this code to examine the vibrating wire for several initial displacements, such as $f(x) = x(1-x)^3$, and $f(x) = x(1-x^3)$, just to get a feel for how waves behave.

1.2 Left side boundary moves

In lab you further modified your wave equation script to examine the wave propagation resulting from a wiggle at the left end of an initially stationary wire at an equilibrium position. Specifically, with a zero initial displacement of the wire ($u(x,0) = f(x) = 0$), you wiggled the left end (a non-zero $g(t)$ in (4)) and watched the wave travel down the wire and reflect back from the right–hand side of the wire.

The file called `Wave1D.m` performs these calculations. Note that it calls a file named `leftBoundary.m` that contains the definition of $g(t)$.

1.3 Flexible boundary condition on right

Finally, in a recent lab, you modified your wave program to allow for the boundary condition $\alpha u + \beta \dfrac{du}{dx} = \gamma$ at the right–hand side of our wave region where $x = L$. By selecting different values of α, β, and γ you can change the right boundary from a fixed value to a free-floating condition. The finite difference version of our new right–hand side boundary condition can be used to solve for, and eliminate, any phantom values of u at the $x = L$ location.

Included with this document is a file called `Wave1DFlex.m` that performs these calculations. Note that it calls an additional file named `rightBoundary.m` that determines the value of the phantom node at the right side of the wire.

1.4 Variable depth model

Now we'll examine the effect of variable depth on the propagation of our 1-D wave. If one assumes the depth of the fluid measured down from the equilibrium surface of the fluid is $h(x)$, then the wave equation (1) must be modified. The result is

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(h(x)\frac{\partial u}{\partial x}\right) = \frac{dh}{dx}\frac{\partial u}{\partial x} + h(x)\frac{\partial^2 u}{\partial x^2}\,, \tag{9}$$

Clearly, if $h(x) = 1$, then (9) becomes (1). If one creates the finite difference version of (9), you will notice that it essentially has the same from as (7) and (8) but with different coefficients in front of the three u^k terms.

- Make a copy of `Wave1DFlex.m` and call it `Tsunami1D.m`.
- Modify this file to incorporate the finite difference version of (9). It should call a supporting file named `depth1D.m` that was included in the package of files. Notice that `depth1D.m` accepts a vector of x locations and returns two vectors of $h(x)$ and dh/dx values.
- Make a nice plot of u versus x, as time increases.

2 2-D Wave equation

Now we'll examine the propagation of a 2-D wave confined to a rectangular region of dimension L_x by L_y. Since the surface of the liquid now varies with both x and y location and with time t, we will refer to $u(x_i, y_j, t_k) = u_{i,j}^k$. If one assumes the depth of the fluid measured down from the equilibrium surface position is now $h(x,y)$, then equation (9) must be modified. The result is

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(h(x,y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(h(x,y)\frac{\partial u}{\partial y}\right)\,. \tag{10}$$

The two initial conditions will be a flat surface at equilibrium, $u(x, y, 0) = 0$, and zero velocity, $u_t(x, y, 0) = 0$. To simplify your code, assume free-floating conditions on three out of the four sides of the rectangular region in the x-y plane. Specifically, $u_x(L, y, t) = 0$, $u_y(x, 0, t) = 0$, $u_y(x, L_y, t) = 0$. The remaining boundary condition at $x = 0$ is, again, a short sinusoidal wiggle. Specifically $u(0, y, t) = \sin(2\pi t)$ for $0 \le t \le 0.5$, and $u(0, y, t) = 0$ for $0.5 \le t$. You may also assume that $\Delta x = \Delta y$.

2.1 2-D uniform depth

To start the 2-D analysis, take a look at the simple case of uniform depth, $h(x, y) = 1$. In this case, the general 2-D wave equation (10) becomes

$$\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\,. \tag{11}$$

- Convert (11) to finite difference form. The result should involve values of u at five spatial locations in the t_k plane. It should also involve one value of u in the t_{k-1} plane, one value in the t_k plane, and one value in the t_{k+1} plane, all at location (x_i, y_j).
- Make a copy of `Wave1DFlex.m` and call it `Wave2D.m`. Modify this file to incorporate the finite difference version of (11).
- You may simplify your `Wave2D.m` code by assuming free-floating conditions on the top, bottom and right boundaries of the rectangular region. In other words, $u_n = 0$ where n is x on the right side, and n is y on the top and bottom. This allows you to eliminate the "flexible" boundary condition (5) and replace it with the much simpler condition $u_n = 0$ on the top, bottom, and right sides.
- Make a nice illuminated surface plot of the fluid surface over the rectangular region. You might want to use a `'cool'` colormap.

2.2 2-D variable depth

We must finally address the most general problem of solving (10) on a rectangular region with the boundary and initial conditions from the previous section.

- Convert (10) to finite difference form. The result should still involve five values of u in the t_k plane. It should also still involve one value of u in the t_{k-1} plane, one value in the t_k plane, and one value in the t_{k+1} plane, all at location (x_i, y_j). However, the coefficients on the various u terms should now involve local values of h, dh/dx and dh/dy.
- Make a copy of `Wave2D.m` and name it `Tsunami2D.m`. Modify this file to incorporate the finite difference version of (10).
- Your `Tsunami2D.m` file should call a supporting file named `depth2D.m` that accepts two-dimensional arrays of x and y coordinates and returns three two-dimensional arrays of h, $\partial h/\partial x$ and $\partial h/\partial y$ values at specific (x, y) locations.

  Be careful to keep the bottom of your "ocean" below the water surface. In other words, don't let $h(x)$ become negative at any location in your L_x by L_y region.
- You may still simplify your `Tsunami2D.m` code by assuming free-floating boundary conditions on the top, bottom and right sides of the rectangular region. In other words, $u_n = 0$ where n is x on the right side, and n is y on the top and bottom.
- Make a nice illuminated surface plot of the surface of the fluid over the rectangular region.

3 The project write-up

- You must work in groups of size two or three. Even if more than three of you work together to write your programs, you must break up into smaller groups when it comes time to write the summary report.
- The document should be done in LaTeX. I have included the LateX file for this document to provide you with another sample document done in LaTeX.
- The style should be similar to that used in Calculus III or Differential Equations. In short, it should look like a real project report — something that you might submit to an employer or a journal to clearly explain what the problem was all about, how you approached solving it, what your results are, and what the solution and results mean.
- You should submit a hard–copy of the report by the due date listed at the top of this document. Hard–copies will be accepted until 5 pm. Electronic files will be accepted until 11:59 pm.
- You should also submit all of your Matlab code, any supporting Matlab files, and the LaTeX file for your report. Please put the Matlab code in one folder and the LaTeX file in another. Be sure to use the naming conventions at the end of this document.

Electronic work should:

- have all code and any output inside a directory (folder) named `YourLastNamePR02`. This directory should be "zipped" in an archive file named `YourLastNameHPR02.zip` and submitted to D2L by 11:59 PM on the due date.
- contain the following comment lines at the beginning of each file, (with the appropriate information filled in):

```
% A one-line description of the program
% Your name
% Today's date
% APPM 3050, Project #02
```

- contain sufficient comments to clearly explain what is being done at various points in the program.