ENGR 695 Advanced Topics in Engineering Mathematics Fall 2024

Lab #9: Explicit Finite Difference Solution of Wave Equation

Introduction

In this lab session, you will continue to examine the solution of partial differential equations (PDEs) using explicit finite difference methods, this time applied to the wave equation. You will have the opportunity to observe solutions for both closed and open boundaries, to investigate the effects of solution parameter values on the solution's accuracy and stability, and to examine the effects of grid dispersion.

Theoretical Background

Recall that a one-dimensional (in space) wave equation problem can be defined as

$$v_p^2 \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}, \qquad a \le x \le b \quad \text{and} \quad t \ge 0,$$

$$u(a,t) = u_a \quad u(b,t) = u_b \quad u(x,0) = f(x) \quad \left.\frac{\partial u(x,t)}{\partial t}\right|_{t=0} = g(x),$$

where u_a and u_b comprise a set of Dirichlet boundary conditions and where functions $f(x)$ and $g(x)$ describe the initial conditions. The constant v_p is the propagation speed of waves within the material spanned by the solution space. It is implicitly assumed here that the speed is the same for all frequency components of the wave; in some materials, that is not the case. Note that the boundary conditions are not homogeneous if u_a and/or u_b are nonzero. Many numerical methods can easily handle nonhomogeneous problems.

A finite difference (FD) solution can be obtained by dividing the solution space into N_x points (counting the boundary locations $x = a$ and $x = b$) with $N_x - 1$ intervals between them. An explicit finite difference solution to the wave equation yields an update equation given by

$$u_{i,j+1} = C^2 u_{i+1,j} + 2\left(1 - C^2\right) u_{i,j} + C^2 u_{i-1,j} - u_{i,j-1},$$

where

$$C = \frac{v_p \Delta t}{\Delta x},$$

and where $u_{i,j}$ is the value of the dependent variable at location $x = a + (i - 1)\Delta x$ for $i = 1, 2, 3, \ldots, N_x$. Index j specifies the discrete moment in time $t = j\,\Delta t$. A special update equation must be applied at the initial time step ($j = 0$) of the algorithm to account for the $-u_{i,j-1}$ term, which corresponds to the solution two time steps before $j + 1 = 1$ (i.e., at $j = -1$). The special update equation is

$$u_{i,1} = \frac{C^2}{2} u_{i+1,0} + \left(1 - C^2\right) u_{i,0} + \frac{C^2}{2} u_{i-1,0} + \Delta t\, g(x_i),$$

where $g(x_i)$ is the value of the time derivative at $t = 0$ evaluated at location i (i.e., the initial condition associated with the time derivative of the dependent variable u).

The value of the constant C that appears in the update equation coefficients directly affects the stability of the algorithm. Moreover, its value sets an upper limit on the length of the time step Δt. Specifically, the constants C and Δt must satisfy the Courant-Friedrichs-Lewy (CFL) condition

$$C = \frac{v_p \Delta t}{\Delta x} \leq 1 \quad \rightarrow \quad \Delta t \leq \frac{\Delta x}{v_p}$$

to obtain a stable solution. The proof of the stability condition is beyond the scope of this course.

As with the FD solution to the heat equation, Dirichlet boundary conditions such as

$$u(a,t) = u_a \quad u(b,t) = u_b$$

are accommodated in the FD solution to the wave equation by setting the first and last elements of the solution vector u to the values

$$u_{1,j} = u_a \quad \text{and} \quad u_{N_x,j} = u_b$$

at all time steps. Also like the heat equation solution, the initial condition

$$u(x,0) = f(x)$$

is implemented by filling the solution vector at time $t = 0$ (corresponding to $j = 0$) with the values of $f(x)$ evaluated at each location within the solution space.

We have also seen that one-dimensional wave equation problems for the unbounded case, described by

$$v_p^2 \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}, \qquad -\infty < x < \infty \quad \text{and} \quad t \geq 0$$
$$u(x,0) = f(x) \quad \left.\frac{\partial u(x,t)}{\partial t}\right|_{t=0} = g(x),$$

can be solved by introducing open boundary conditions based on the one-way wave equation. The FD update equations applied to the interior points ($i = 2$ through $N_x - 1$) of the space are the same as those given above for Dirichlet problems, but the points at the boundaries must be updated using

$$u_{1,j+1} = \frac{2C^2}{1+C} u_{2,j} + 2(1-C) u_{1,j} - \frac{1-C}{1+C} u_{1,j-1} \quad \text{at the boundary } x = a$$

and

$$u_{N_x,j+1} = 2(1-C) u_{N_x,j} + \frac{2C^2}{1+C} u_{N_x-1,j} - \frac{1-C}{1+C} u_{N_x,j-1} \quad \text{at the boundary } x = b.$$

The special update equations used at the boundaries only at time step $j = 0$ are

$$u_{1,1} = C^2 u_{2,0} + \left(1 - C^2\right) u_{1,0} + \left(1 - C\right) \Delta t \, g\left(a\right) \quad \text{at the boundary } x = a$$

and

$$u_{N_x,1} = \left(1 - C^2\right) u_{N_x,0} + C^2 u_{N_x-1,0} + \left(1 - C\right) \Delta t \, g\left(b\right) \quad \text{at the boundary } x = b.$$

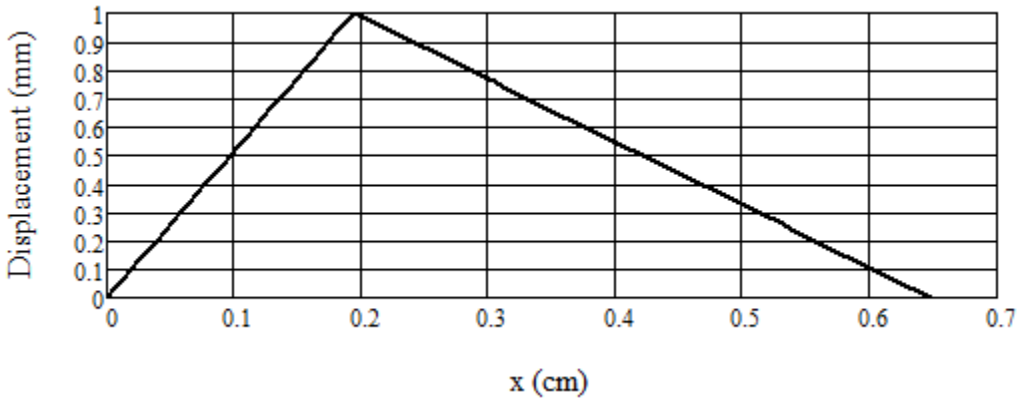
Procedure

- Download the following *Matlab* m-file, which is available at the course Moodle site. You should set up a separate folder to contain your work.

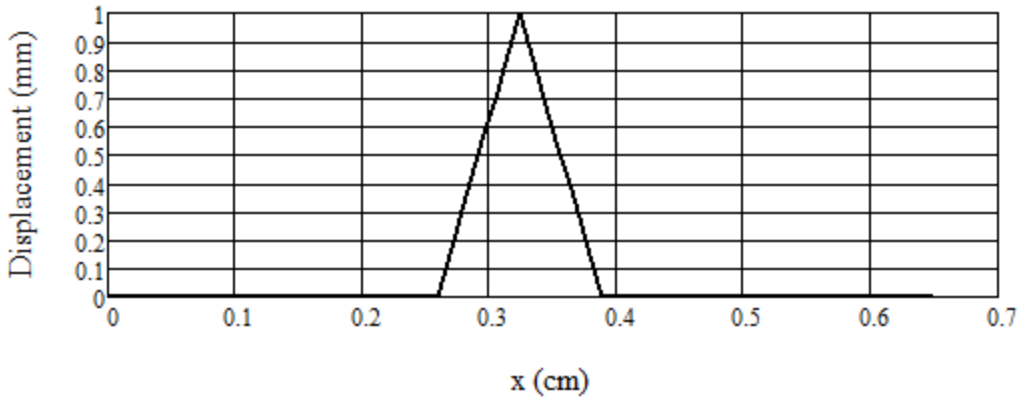
  `Lab9start.m` – script that contains the main algorithm and the two functions that define the initial string displacement $f(x)$ and the initial vertical velocity of the string $g(x)$

  The m-file is heavily commented. Look over the source code and try to understand in general how it works. The important problem parameters, such as the boundary locations a and b and the numbers of spatial and time steps, are coded near the top of the m-file. The problems that you will simulate in this lab exercise are mostly the same as the ones in a previous lab exercise that examined the solution of the 1-D wave equation using the separation of variables (SoV) method.

- Run the script to confirm that everything is working properly. The default initial displacement of the string should have the shape shown below. The string should change shape though one complete cycle and return to its initial position at the last time step. This problem is identical to the vibrating string problem to which we applied the separation-of-variables solution.



- After you are confident that the *Matlab* script is working, modify the *Matlab* function that defines the initial displacement $f(x)$ so that is has the "tent" shape shown at the top of the next page. (The required code is already there.) Also, at the top of the main part of the code, change the variable `iBC`, which determines the type of boundary condition, to the value 3, which corresponds to open boundary conditions based on the one-way wave equation.



The locations that correspond to the start, peak, and stop points of the triangular section are $x = 0.4L$, $x = 0.5L$, and $x = 0.6L$, respectively, where L is the length of the string.

- Execute your modified code. The "tent" function should split into two "tents" of half the original amplitude that propagate outward in opposite directions and then exit the space.

- After a successful execution, set the CFL (Courant-Friedrichs-Lewy) number near the top of the main section of code to 1.01, which will set the time step to a value that it is slightly above the stability limit. Run the script and observe the results.

- Now set the CFL number to 0.5, which is only half of the limit, and observe the results. You will probably see an example of grid dispersion exacerbated by the sharp discontinuity in $f(x)$. The discontinuity causes high frequency modes to be generated (remember the SoV solution?) that are not well resolved by the grid (i.e., the set of discrete x locations). Even if the CFL number is relatively high, like 0.9, the effect is evident. Executing the algorithm at the CFL limit often masks or at least reduces grid dispersion. Try increasing the density of the grid (i.e., increase the value of N_x) by, say, a factor of two or four and increasing the number of time steps by a proportionate amount (since Δt is reduced if Δx is reduced). Then run the script again. The waveform should be a little less distorted this time. Also try running the script at the original grid size but with even smaller CFL numbers (such as 0.1).

- Modify the *Matlab* function that defines the initial displacement $f(x)$ again so that the displacement function describes a piecewise parabolic initial condition. (Again, the required code is already there; it is the third section of code in the function.) Keep the `iBC` flag set to the value 3, which corresponds to open boundary conditions. Run the code with the CFL number set to 1 and then to 0.5, and compare the results to those obtained with the "tent" function. Note that the piecewise parabolic function is smoothly varying and continuous and that it has a continuous first derivative as well.

- There are no submission requirements. To obtain full credit for the lab exercise, by Friday, December 6, briefly discuss your observations related to the "tent" and piecewise parabolic functions with the instructor, emphasizing the effects of grid dispersion. Offer brief evidence that you understand at least partially what is happening and why. If you cannot complete this part in lab, you may do so during a Zoom or in-person help session or by appointment.