Lab 6
March 10, 2017

Waves
(Based on chapter 6 of "Computational Physics" by Giordano and Nakanishi)

Here we will consider a few examples associated with wave motion on a string and will use numerical methods to estimate the shape of the wave at various time intervals.

The central equation of wave motion is:

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2}$$

The string runs along the horizontal (x) axis and y represents the vertical displacement of the string as a function of time t.

To construct a numerical algorithm to solve the wave equation, we will write it in finite difference form. We treat both x and t as discrete variables with $x = i\Delta x$ and $t = j\,\Delta t$, and will write the wave equation as follows:

$$y(i, j+1) = 2[1 - r^2]\, y(i, j) - y(i, j-1) + r^2[y(i+1, j) + y(i-1, j)]$$

This equation assumes that the shape of the wave, i.e. the vertical displacement y at each point, is known at time steps j and j-1 and uses that information to calculates the displacement at time step j+1.

The dimensionless variable r is defined to be:

$$r \equiv c\frac{\Delta t}{\Delta x}$$

By knowing the initial condition at time t=0 and the boundary conditions at the two end points of the string, we can calculate the shape of the wave at any point in time.

Gaussian pluck: A script is provided that calculates the shape of a wave in a string with end point. In that script, we have assumed a simple "Gaussian pluck" of the string. That is, we have taken the initial string profile to be

$$y_0 = exp[-k(x - x_0)^2]$$

where the displacement is centered at x_0 and k determines the width of the Gaussian envelope. In that example, we have picked x_0 = 0.3 m and k=1000 m^{-2} and the length of the string to be l=1 m (runs from x=0 to x=1m).

The script Waves-fixed-end-points2.py (attached to this instruction) will do the following:

- Defines the following constants:
    - c = 300 # propagation speed of the wave in m/s
    - l = 1 # length of the string in meters
    - dx = 0.01 # grid spacing on the string in meters
    - r = 1 # dimensionless variable r to relate dt and dx to c
    - dt = r*dx/c
    - r = c*dt/dx

- Defines a variable called n as:
    - n= int(l/dx)

  where, the number of grid points on the string is n+1.

- Defines an array called Y to store the vertical displacement of the string at each time.

  To use the numerical algorithm described above, one needs to store the displacement of the string at each time step j and the previous time step j-1 to calculate the displacement at the next time step j+1.

  As a result, we will define the variable Y to have three rows. The first row will store the displacement of the string at each grid point (i=0 to n) at time step j-1. The second row will store the displacement at time step j, and the third row will store the displacement at the next time step j+1 as they are being calculated.

  So, the array Y will be a (n+1) by 3 matrix.

  We will initialize this array to zero.

  Y = np.zeros([n+1,3])

  The values of Y will be updated as time progresses. At each time, we will use the first and second row values to calculate the values of the third row.

step (j-1)	Y[0,0]	Y[1,0]	Y[2,0]	Y[3,0]	…	Y[n,0]
step j	Y[0,1]	Y[1,1]	Y[2,1]	Y[3,1]		Y[n,1]
step (j+1)						

- Define the shape of the string at time zero and set the first row of Y (Y[:,0]) to be equal to the initial displacement (in this case Gaussian pluck).

- Define a loop over j (time step), and calculate Y as a function of time.

    - At each time step, we need to calculate the position along the sting. The grid points along the string run from 0 to n, so we have to define (1) a loop to calculate Y at every position inside the string (excluding the end points) and calculate the third row of the matrix Y based on its first tow rows.

    We will use the following equation for this calculation:

$$Y(i,2) = 2[1 - r^2]\,Y(i,1) - Y(i,0) + r^2[y(i+1,1) + y(i-1,1)]$$

    and calculate Y for i=1.. n-1. To simplify the script, the above equation is written as follows for index (i+1):

$$Y(i+1,2) = 2[1 - r^2]\,Y(i+1,1) - Y(i+1,0) + r^2[y(i+2,1) + y(i,1)]$$

    - Now, we need to update the displacement at the end points. For a string with fixed end points, the displacement at the end points is always zero.

      Y[0,2]=0
      Y[n,2]=0

      For a string with free end points, it can be shown that he displacement at the end points is the same displacements the points that are one spatial unit in from the ends.

      Y[0,2]=Y[1,2]
      Y[n,2]=Y[n-1,2]

      However, the above lines WILL NOT be executed the way you expect them in python. You should NEVER assign to array elements to be the same, or updating one of them will automatically update the other element, too. To achieve what we want, i.e. assigning the value of

Y[1,2] to Y[0,2], we need to define an auxiliary variable, and use it for this assignment as follows:

temp1 = Y[1,2]
Y[0,2] = temp1

temp2 = Y[n-1,2]
Y[n,2] = Y[n,2]

One can also show that if the end points are free the wave equations at the end points can be written as:

$$\frac{d^2y_0}{dt^2} \approx c^2 \frac{y_1 - y_0}{(\Delta x)^2}$$

and

$$\frac{d^2y_n}{dt^2} \approx c^2 \frac{y_{n-1} - y_n}{(\Delta x)^2}$$

Once you have your script working, you might want to implement these equations for end point displacement and see that it will result in the same wave. To implement these numerically:

One can also show that if the end points are free the wave equations at the end points can be written as:

Y[0,2] = 2*(1-0.5*r**2)*Y[0,1] – Y[0,0] + r**2*Y[1,1]
Y[n,2] = 2*(1-0.5*r**2)*Y[n,1] – Y[n,0]+r**2*Y[n-1,1]

- Now, that all the rows in the matrix Y is filled, before moving to the next step, we need to update each row of Y from:

step (j-1)	Y[0,0]	Y[1,0]	Y[2,0]	Y[3,0]	…	Y[n,0]
step j	Y[0,1]	Y[1,1]	Y[2,1]	Y[3,1]		Y[n,1]
step (j+1)	Y[0,2]	Y[1,2]	Y[2,2]	Y[3,2]		Y[n,2]

to

step j	Y[0,1]	Y[1,1]	Y[2,1]	Y[3,1]	…	Y[n,1]
step (j+1)	Y[0,2]	Y[1,2]	Y[2,2]	Y[3,2]		Y[n,2]
step (j+2)						

And use the value of Y at time steps and j+1 to calculate Y at a future time step j+2. This is done by shifting the rows of Y as follows:

```
for in in range(n+1):
        Y0 = Y[i,1]
        Y1 = Y[i,2]
        Y[i,0] = Y0
        Y[i,1] = Y1
```

You can see again that we have used the auxiliary variables Y0 and Y1 to avoid setting two array elements equal.

- Finally, you can plot the wave along the string at every time step as you wish.

For this example, the string with free end points, please plot the wave at several time steps (on multiple plots or subplots and clearly labeled for timestep) and show the reflection of the wave from the end points. Plotting the waves on the same plot will not get any credit.

For the next example, please modify your script to numerically simulate the motion of a string for which one end is held fixed and the other end is made to oscillate. Do this by letting the string element at one end move according to

$$y(i = 0) = Asin(\omega t)$$

You should find that this generates a wave that propagates toward the opposite end of the string. This wave is then reflected, and it interferes with the initial wave.

Choose A = 1m, and vary the value of ω. Start from a value close to 500 and increase it to 2000. You can see that as you increase the frequency, the wavelength decreases. At some point, the wavelength will be equal to the length of the string l=1m. Can you identify this point and plot the wave at this frequency? Confirm that this frequency and wavelength are consistent with the parameter c in the wave equation.

At this frequency, what happens if you calculate the wave at t=1000*dt, t=2000*dt, and t=3000*dt. What is the difference between these plots?

Identify a few frequencies that result in standing waves and plot the waves. To show the standing waves, at each frequency, plot the wave at multiple time steps in the same plot and clearly indicate why it is a standing wave. Make sure that you have waited long enough for the wave to propagate along the string and reflect from the end points.