Suman Patra, Assistant Professor, Department of Physics,

Netaji Nagar Day College

Topic for

Semester – 4, Paper – PHSA CC8

SOLVING WAVE EQUATION NUMERICALLY

Let us now develop the numerical solution techniques for the second order wave equation.

Consider the wave equation:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \;\; ; \;\; 0 \le x \le L, t \ge 0$$

Let us now specify the boundary conditions and initials conditions.

Boundary Conditions

u (0, t) = 0 = u (L, t) [The string is clamped at both ends x=0 and x=L]

Initial Conditions

u (x, 0) = f(x), $\frac{\partial u}{\partial t}$ (x, 0) = g(x) [Initially the element of string at x was at position f(x) and this element was given a blow with velocity g(x)]

Let us first try to discretize the wave equation.

We can write, $\frac{\partial u}{\partial x} = \frac{u(x+\Delta x)-u(x)}{\Delta x}$

Therefore, $\frac{\partial^2 u}{\partial x^2} = \frac{\frac{\partial u}{\partial x}(at\ x) - \frac{\partial u}{\partial x}(at\ x-\Delta x)}{\Delta x}$

$$= \frac{\frac{u(x+\Delta x)-u(x)}{\Delta x} - \frac{u(x)-u(x-\Delta x)}{\Delta x}}{\Delta x} = \frac{u(x+\Delta x) - 2u(x) + u(x-\Delta x)}{(\Delta x)^2}$$

Similarly, $\frac{\partial^2 u}{\partial t^2} = \frac{\frac{\partial u}{\partial t}(at\ t) - \frac{\partial u}{\partial t}(at\ t-\Delta t)}{\Delta t}$

$$= \frac{\frac{u(t+\Delta t)-u(t)}{\Delta t} - \frac{u(t)-u(t-\Delta t)}{\Delta t}}{\Delta t} = \frac{u(t+\Delta t) - 2u(t) + u(t-\Delta t)}{(\Delta t)^2}$$

$$\therefore \frac{u(x+\Delta x) - 2u(x) + u(x-\Delta x)}{(\Delta x)^2} = c^2\ \frac{u(t+\Delta t) - 2u(t) + u(t-\Delta t)}{(\Delta t)^2} \qquad ....(1)$$

We have discretized the wave equation.

Let us now constitute a finite difference mesh.

First divide the interval $0 \le x \le L$ into (N+1) equally spaced points.

$\therefore$ dx $= \frac{L}{N} = \Delta x$

Now x=0 is denoted by x_0, $x=0+\Delta x$ is denoted by x_1, ………, $x_n = n\Delta x$, …... x=L is denoted by x_{N+1}

Then divide the interval $0 \le t \le 1$ into (K+1) equally spaced points.

$\therefore$ dt $= \frac{1}{K} = \Delta t$ [We have assumed the final value of time t is 1 sec]

Now t=0 is denoted by t_0, $t=0+\Delta t$ is denoted by t_1, ………, $t_k = k\Delta t$, …... t=1 is denoted by t_{K+1}

So, the mesh point at x= x_n and t= t_k is denoted by (n,k).

And the value of u at x= x_n and t= t_k is denoted by $u(x_n, t_k) = u_n^k$

Therefore, equation (1) takes the form –

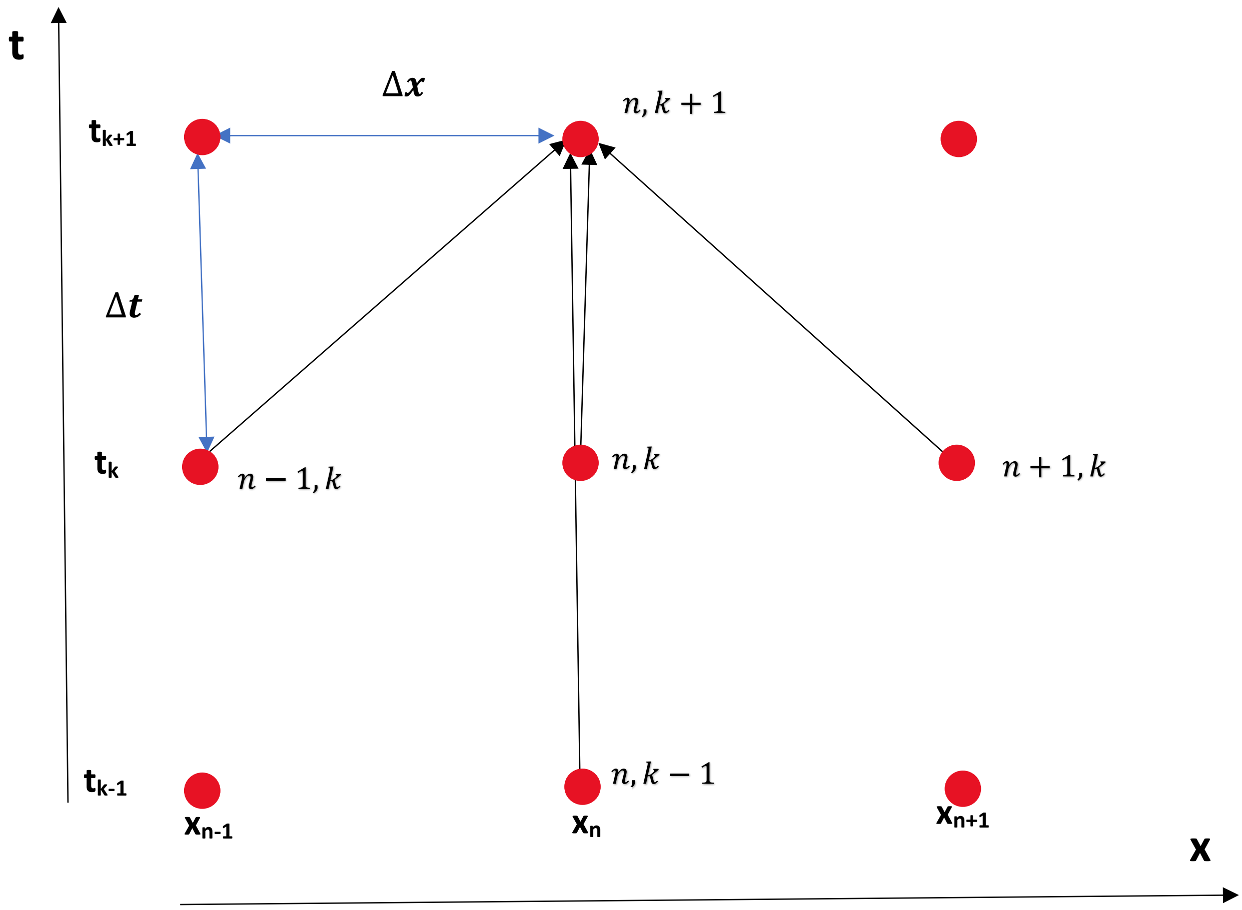
$$\frac{u_{n+1}^k - 2u_n^k + u_{n-1}^k}{(\Delta x)^2} = c^2\ \frac{u_n^{k+1} - 2u_n^k + u_n^{k-1}}{(\Delta t)^2}$$

$$\therefore u_n^{k+1} = 2u_n^k - u_n^{k-1} + \left(\frac{c\Delta t}{\Delta x}\right)^2 (u_{n+1}^k - 2u_n^k + u_{n-1}^k)$$

$$\therefore \underbrace{u_n^{k+1}}_{\text{time level } k+1} = \underbrace{r^2 u_{n+1}^k + 2(1-r^2)\, u_n^k + r^2 u_{n-1}^k}_{\text{time level } k} - \underbrace{u_n^{k-1}}_{\text{time level } k-1}$$

*r= $(c\Delta t/\Delta x)$ is known as courant number.

……. (2)



We can calculate u-values at all mesh points in (k+1)th row using the

u-values at mesh points in k-th row and (k-1)th row.

So, if we know u-values for all the mesh points in row k=0 and row k= -1 then we can calculate u-values for all the mesh points at row k=1.

Similarly, we can calculate u-values for all the mesh points in row k=1 using the u-values at row k=0 and k=1.

In this manner we can calculate u-values at all mesh points in the entire region.

From the initial conditions we know u-values at all the mesh points in row k=0 (row k=0 corresponds to time t=0).

u (x, 0) = f(x)

So, we can write u (x_n, 0) = f(x_n)

$$u_n^0 = f_n$$

$u_0^0 = f_0$, $u_1^0 = f_1$, $u_2^0 = f_2$, ………., $u_n^0 = f_n$, ………………., $u_N^0 = f_N$

But what are the u-values at row **k= -1?** (k= -1 corresponds to time t<0!)

Let us write the other initial condition which is,

$$\frac{\partial \mathrm{u}}{\partial \mathrm{t}}(\mathrm{x}, 0) = \mathrm{g}(\mathrm{x})$$

$$\therefore \frac{u(x_n,t+\Delta t)-u(x_n,t-\Delta t)}{2\Delta t} = \mathrm{g}(x_n)$$

Instead of **Forward difference method we are using here **Central difference method**

$$\therefore \frac{u(n,k+1)-u(n,k-1)}{2\Delta t} = \mathrm{g}(x_n)$$

Let us imagine a row of false mesh points at time $t = 0 - \Delta t = -\Delta t = t_{-1}$, then this initial velocity condition can be written using the central difference method as,

$$\frac{u(n,1)-u(n,-1)}{2\Delta t} = \mathrm{g}(x_n) \quad [\textit{as}\ \mathrm{k=0}\ \textit{for}\ \mathrm{t=0}]$$

$$\therefore \frac{u_n^1-u_n^{-1}}{2\Delta t} = \mathrm{g}(x_n)$$

$$\therefore u_n^{-1} = u_n^1 - \mathrm{g}(x_n)*\ 2\Delta t \quad \textbf{........ (3)}$$

The discretized wave equation (2) holds true at t=0.

$$\therefore u_n^{0+1} = \mathrm{r}^2 u_{n+1}^0 + 2(1-\mathrm{r}^2)\, u_n^0 + \mathrm{r}^2 u_{n-1}^0 - u_n^{0-1}$$

$$\therefore u_n^1 = \mathrm{r}^2 u_{n+1}^0 + 2(1-\mathrm{r}^2)\, u_n^0 + \mathrm{r}^2 u_{n-1}^0 - u_n^{-1}$$

$$\therefore u_n^1 = \mathrm{r}^2 u_{n+1}^0 + 2(1-\mathrm{r}^2)\, u_n^0 + \mathrm{r}^2 u_{n-1}^0 - \{u_n^1 - \mathrm{g}(x_n)*\ 2\Delta t\} \quad [\text{from eqn(3)}]$$

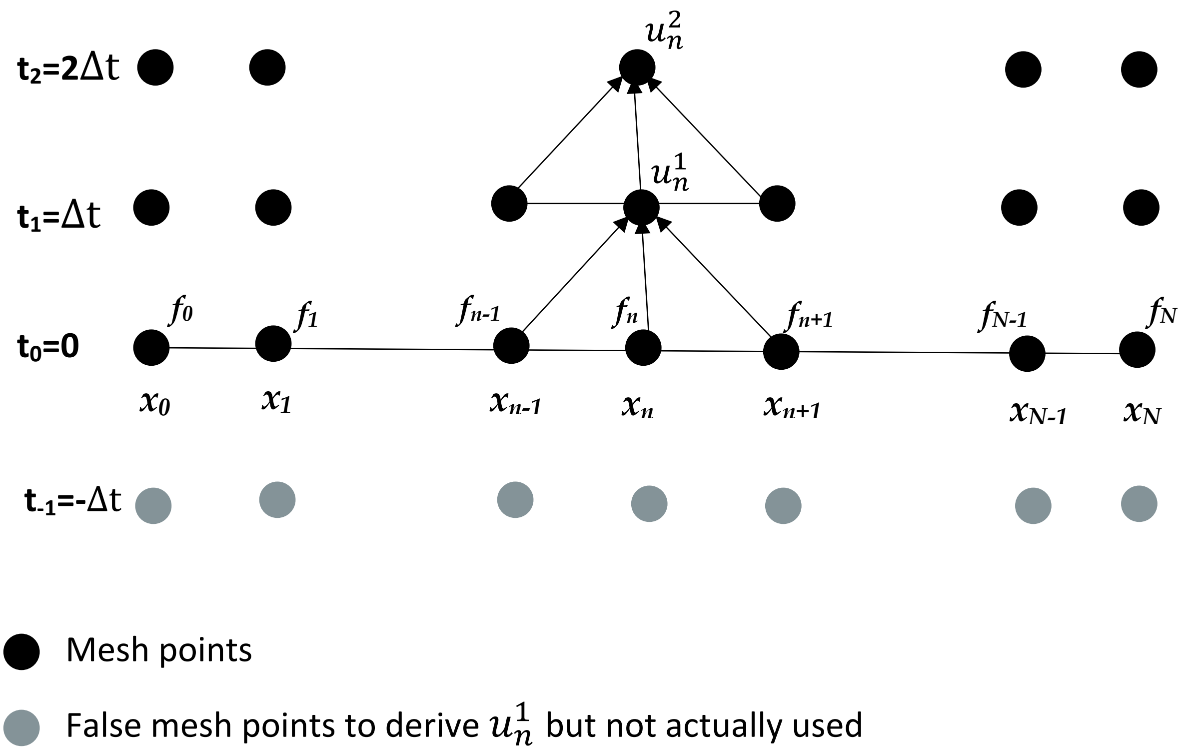
$$\therefore 2u_n^1 = \mathrm{r}^2 u_{n+1}^0 + 2(1-\mathrm{r}^2)\, u_n^0 + \mathrm{r}^2 u_{n-1}^0 + \mathrm{g}(x_n)*\ 2\Delta t$$

$$\therefore u_n^1 = \frac{1}{2}\{\mathrm{r}^2 u_{n+1}^0 + 2(1-\mathrm{r}^2)\, u_n^0 + \mathrm{r}^2 u_{n-1}^0\} + \Delta t\, \mathrm{g}(x_n) \quad \textbf{........ (4)}$$

$u_n^0 = \mathrm{f}(x_n)$ and $\mathrm{g}(x_n)$ are known from initial conditions.

So, using equation (4) we can determine u-values at all the mesh points in row k=1.

Now we are in a position to find out u-values at all the subsequent rows.



Let us have a look at the python code of this numerical technique.

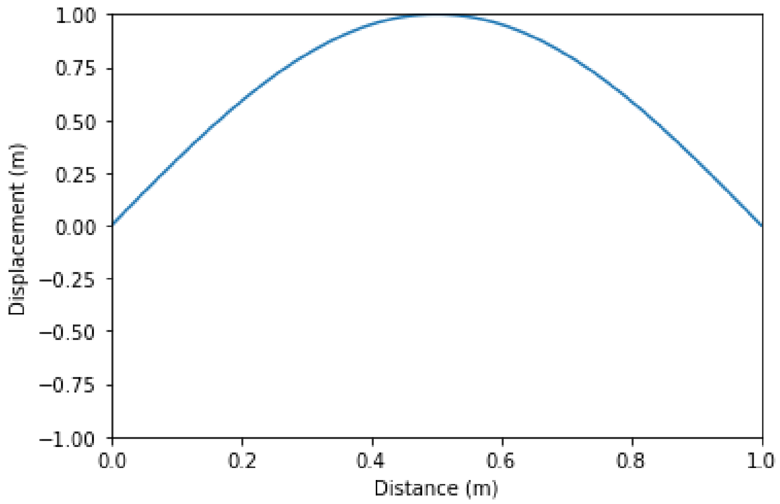
Python Code

```
#1D Wave Equation
import numpy as np
import matplotlib.pyplot as plt

L=1.0 #Length of String
n=100 #no of elements in string
c=300 #Wave Velocity
dx=L/n #Element size
t_final=0.1 #Final time at which you want to see wave
dt=dx/(2*c) #Time interval
r=0.5
vel=0.0 #Initial Velocity

x=np.linspace(0,L,n) #Constructing position matrix

t=np.arange(0,t_final,dt) #Constructing time matrix

row=len(t)
col=len(x)
Dis=np.empty([row,col])  #Constructiong Displacement matrix

for i in range(0,col):     #Calculating un0
        Dis[0][i]=np.sin((np.pi*x[i])/L)

for i in range(1,col-1):  #Calculating un1
        Dis[1][i]=((r**2*(Dis[0][i+1]+Dis[0][i-1])+2*(1-r**2)*Dis[0][i])/2)+(dt*vel)
```

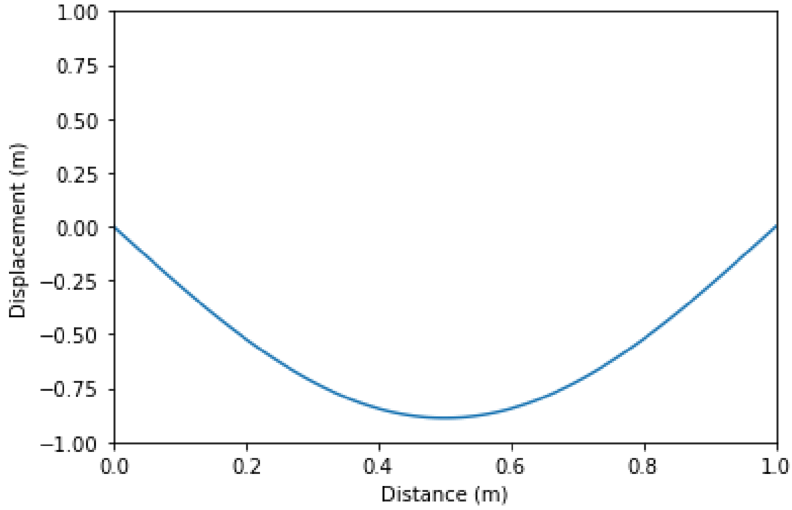
```
for j in range(1,row):
        Dis[j][0]=0
        Dis[j][col-1]=0

for j in range(1,row-1):
        for i in range(1,col-1):
                Dis[j+1][i]=(r**2*(Dis[j][i+1]+Dis[j][i-1])+2*(1-r**2)*Dis[j][i])-Dis[j-1][i]

for j in range(0,row,500):
        print('Distance vs Displacement Curve at t = ',round(t[j],3),'sec')
        plt.figure(1)
        plt.plot(x,Dis[j,:])
        plt.axis([0,L,-1.0,1.0])
        plt.xlabel('Distance (m)')
        plt.ylabel('Displacement (m)')
        plt.show()
print('Thanks')

dt1=dt
t1=np.arange(0,t_final,dt1)
row1=len(t1)
tamp=np.empty([row1,col])
for j in range(0,row1):
        for i in range(0,col):
                tamp[j][i]=Dis[j][i]
for i in range(0,col):
        print('Time vs Displacement Curve at x = ',round(x[i],3),'cm')
        plt.figure(2)
        plt.plot(t1,tamp[:,i])
        plt.axis([0,1.2*t_final,-1.2,1.2])
        plt.xlabel('Time (Sec)')
        plt.ylabel('Displacement (m)')
        plt.show()
print('Thanks Again')
```

Output

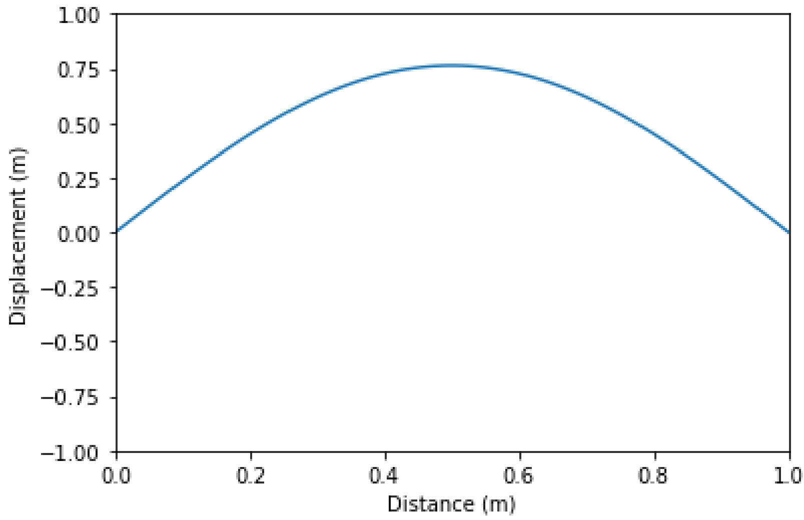
Distance vs Displacement Curve at t = 0.0 sec



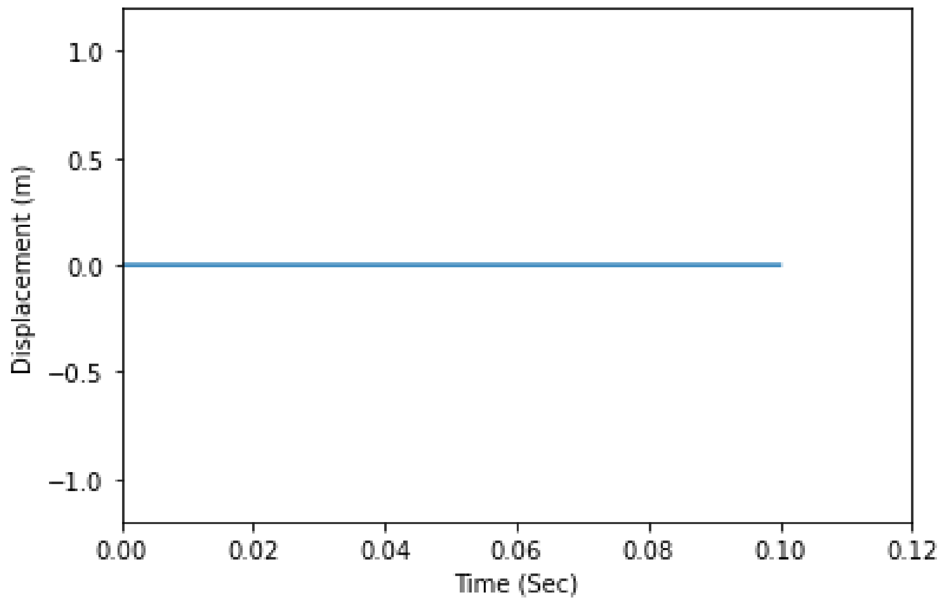
Distance vs Displacement Curve at t = 0.05 sec



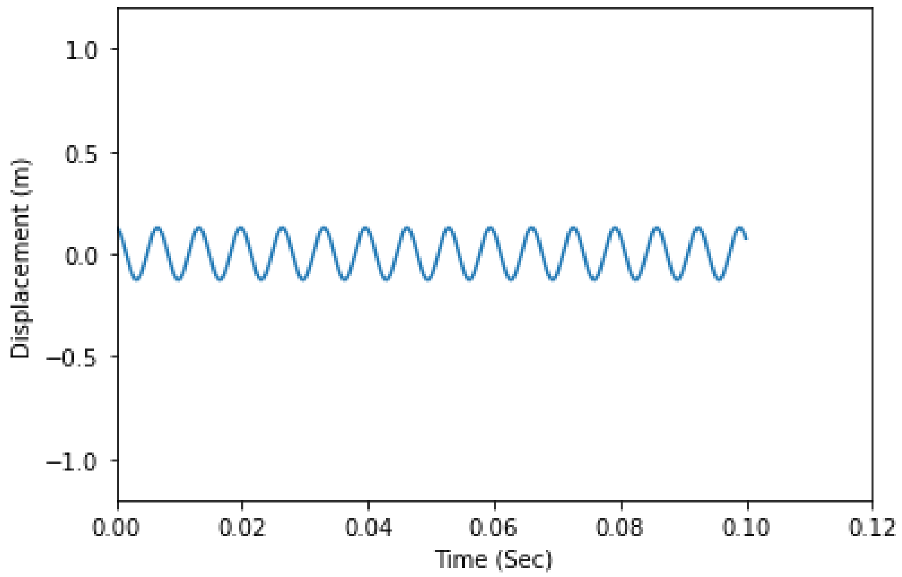
Distance vs Displacement Curve at t = 0.092 sec



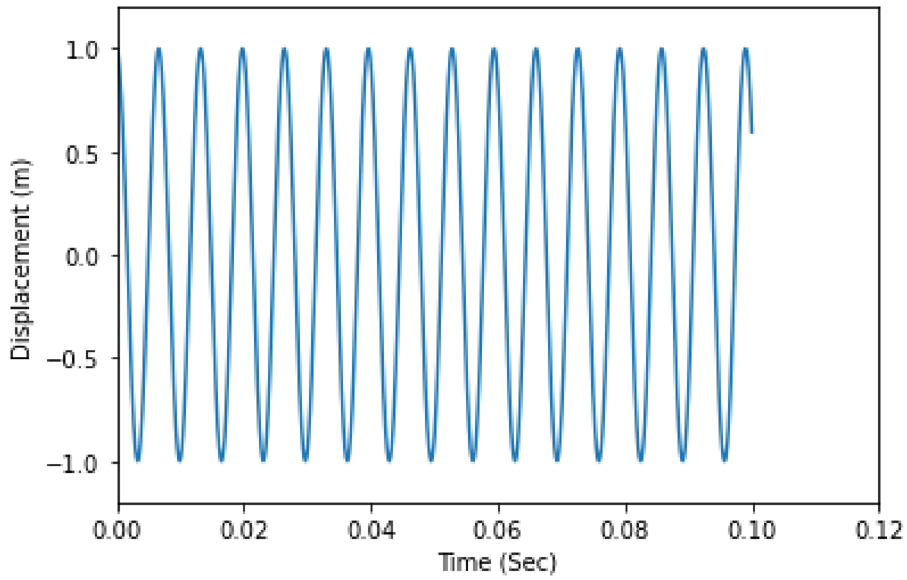
Time vs Displacement Curve at x = 0.0 cm



Time vs Displacement Curve at x = 0.04 cm



Time vs Displacement Curve at x = 0.495 cm



Time vs Displacement Curve at x = 0.99 cm