FEM 1-D First order Square Wave Equation

Mathematical model

$$\frac{\partial U}{\partial t} + c\frac{\partial U}{\partial x} = 0 \tag{1}$$

Boundary Conditions:

U(0, t) = 0

U(L,t) = 0

Now I multiply both sides by a weighting function "w" and integrate over each element. Each element starting point is "a" and end point is "b":

$$\int_a^b [\frac{\delta U}{\delta t} w]dx + c\int_a^b [\frac{\delta U}{\delta x} w] = 0 \tag{2}$$

The weak form at the element level becomes:

$$\int_a^b [\frac{\delta U}{\delta t} w - c.U\frac{\delta w}{\delta x}]dx = -cUw]_a^b \tag{3}$$

Let's implement FDM for $\frac{\delta U}{\delta t}$:

$$\int_a^b [\frac{U^{t+1} - U^t}{\Delta t} w]dx - \int_a^b [c.U\frac{\delta w}{\delta x}]dx = -cUw]_a^b \tag{4}$$

$$\int_a^b [\frac{U^{t+1}}{\Delta t} w]dx - \int_a^b [\frac{U^t}{\Delta t} w]dx - \int_a^b [c.U\frac{\delta w}{\delta x}]dx = -cUw]_a^b \tag{5}$$

Considering Galerkin's method let's assume:

$$w = \phi_i \tag{6}$$

$$u(t,x) = \sum_{j=1}^{2} U_j^t \phi_j(x) \tag{7}$$

where i is the i-th node in the element,which here i is 2 because of linear interpolation and j is the j-th function of U. Also let's take the length of each element as h.

$$\int_0^h [\sum_{j=1}^{2}[U_j^{t+1}\phi_j(x)]\frac{\phi_i}{\Delta t}]dx - \int_0^h [\sum_{j=1}^{2}[U_j^{t}\phi_j(x)]\frac{\phi_i}{\Delta t}]dx - \int_0^h [c\sum_{j=1}^{2}[U_j^{t}\phi_j(x)]\frac{d\phi_i}{dx}]dx = -cUw]_a^b \tag{8}$$

$$\begin{aligned} \frac{\sum_{j=1}^{2}[U_j^{t+1}]}{\Delta t} \underbrace{\int_0^h [\phi_i(x)\sum_{j=1}^{2}\phi_j(x)]dx}_{A} &= \frac{\sum_{j=1}^{2}[U_j^{t}]}{\Delta t} \underbrace{\int_0^h [\phi_i(x)\sum_{j=1}^{2}[\phi_j(x)]dx}_{A} \\ &+ c\sum_{j=1}^{2}[U_j^t] \underbrace{\int_0^h [\sum_{j=1}^{2}[\phi_j(x)]\frac{d\phi_i}{dx}]dx}_{B} - \underbrace{cUw]_a^b}_{D} \end{aligned} \tag{9}$$

If we consider Upwind method with $c > 0$, a and b are element boundaries, D becomes:

$$D = cU_{k(b)}\phi_{i(b)} - (cU_{(k-1)(b)}\phi_{i(a)}) \tag{10}$$

Where k is the number of element.
At the left boundary of the element k, $\phi_{1(b)} = 0$ and $\phi_{1(a)} = 1$:

$$cU_{k(b)}\phi_{1(b)} - cU_{k-1(b)}\phi_{1(a)} = -cU_{k-1(b)} \tag{11}$$

At the right boundary of element k, $\phi_{2(a)} = 0$ and $\phi_{2(b)} = 1$:

$$cU_{k(b)}\phi_{2(b)} - cU_{k-1(b)}\phi_{2(a)} = cU_{k(b)} \tag{12}$$

Therefore the D matrix for one element becomes:

$$D = c\begin{bmatrix} -U_{k-1}(x_R) \\ U_k(x_R) \end{bmatrix} \tag{13}$$

Defining $\begin{cases} \bar{U}_i^{t+1} = \sum_{j=1}^{2}[U_j^{t+1}] \\ \bar{U}_i^{t} = \sum_{j=1}^{2}[U_j^{t}] \\ D = 0, \text{ boundary conditions} \end{cases}$:

$$\frac{A}{\Delta t}\bar{U}_i^{t+1} = \frac{A}{\Delta t}\bar{U}_i^{t} + cB\bar{U}_i^t \tag{14}$$

Without using matrix method:

$$\bar{U}_i^{t+1} = \bar{U}_i^t + c\Delta t A^{-1} B \bar{U}_i^t \tag{15}$$

From Lagrangian interpolation $\begin{cases} \phi_1 = 1 - \frac{\bar{x}}{h} \\ \phi_2 = \frac{\bar{x}}{h} \\ \frac{d\phi_1}{dx}\phi_1 = \frac{-1}{h} \\ \frac{d\phi_2}{dx}\phi_1 = \frac{1}{h} \end{cases}$:

Considering only one element and $A = \int_0^h \phi_i \phi_j d\bar{x}$ and $B = \int_0^h \phi_j \frac{d\phi_i}{d\bar{x}} d\bar{x}$:
For $i = 1$ and $j = 1$:

$$A = \frac{h}{3} \tag{16}$$

$$B = \frac{-1}{2} \tag{17}$$

For $i = 1$ and $j = 2$:

$$A = \frac{h}{6} \tag{18}$$

$$B = \frac{-1}{2} \tag{19}$$

For $i = 2$ and $j = 1$:

$$A = \frac{h}{6} \tag{20}$$

$$B = \frac{1}{2} \tag{21}$$

For $i = 2$ and $j = 2$:

$$A = \frac{h}{3} \tag{22}$$

$$B = \frac{1}{2} \tag{23}$$

$$B = \begin{bmatrix} \frac{-1}{2} & \frac{-1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}, A = \begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \longrightarrow A^{-1} = \begin{bmatrix} \frac{4}{h} & \frac{-2}{h} \\ \frac{-2}{h} & \frac{4}{h} \end{bmatrix} \xrightarrow{A^{-1} \times B} = \frac{1}{h} \begin{bmatrix} -3 & -3 \\ 3 & 3 \end{bmatrix}$$

So equation number 11 becomes:

$$\begin{bmatrix} \bar{U}_1^{t+1} \\ \bar{U}_2^{t+1} \end{bmatrix} = \begin{bmatrix} \bar{U}_1^t \\ \bar{U}_2^t \end{bmatrix} + \frac{c\Delta t}{h} \times \begin{bmatrix} -3 & -3 \\ 3 & 3 \end{bmatrix} \begin{bmatrix} \bar{U}_1^t \\ \bar{U}_2^t \end{bmatrix} \tag{24}$$

For point 1 and point 2 in an element:

$$\bar{U}_1^{t+1} = \bar{U}_1^t - \frac{c\Delta t}{h}(\bar{U}_1^t + \bar{U}_2^t) \tag{25}$$

$$\bar{U}_2^{t+1} = \bar{U}_2^t + \frac{c\Delta t}{h}(\bar{U}_1^t + \bar{U}_2^t) \tag{26}$$

With matrix method:

Continue from equation 10:

$$A\bar{U}_i^{t+1} = \underbrace{(A + c\Delta t B)}_{E}\bar{U}_i^t \tag{27}$$

Here for CG matrix D is zero.

$$\begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \begin{bmatrix} U_1^{t+1} \\ U_2^{t+1} \end{bmatrix} = \begin{bmatrix} \frac{h}{3} - \frac{c\Delta t}{2} & \frac{h}{6} - \frac{c\Delta t}{2} \\ \frac{h}{6} + \frac{c\Delta t}{2} & \frac{h}{3} + \frac{c\Delta t}{2} \end{bmatrix} \begin{bmatrix} U_1^t \\ U_2^t \end{bmatrix} \tag{28}$$

Global matrix of RHS:

$$LHS = \begin{bmatrix} \frac{h}{3} & \frac{h}{6} & 0 & 0 & 0 & 0 & ... & 0 \\ \frac{h}{6} & \frac{h}{3} + \frac{h}{3} & \frac{h}{6} & 0 & 0 & 0 & ... & 0 \\ 0 & \frac{h}{6} & \frac{h}{3} + \frac{h}{3} & \frac{h}{6} & 0 & 0 & ... & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & ... & \vdots \\ 0 & 0 & 0 & 0 & ... & 0 & \frac{h}{6} & \frac{h}{3} \end{bmatrix} \begin{bmatrix} U_1^{t+1} \\ U_2^{t+1} \\ U_3^{t+1} \\ \vdots \\ U_N^{t+1} \end{bmatrix} \tag{29}$$

$$B = \begin{bmatrix} \frac{-1}{2} & \frac{-1}{2} & 0 & 0 & 0 & 0 & ... & 0 \\ \frac{1}{2} & \frac{1}{2} + \frac{-1}{2} & \frac{-1}{2} & 0 & 0 & 0 & ... & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} + \frac{-1}{2} & \frac{-1}{2} & 0 & 0 & ... & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & ... & \vdots \\ 0 & 0 & 0 & 0 & ... & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{30}$$

Results and CFL sensitivity analysis

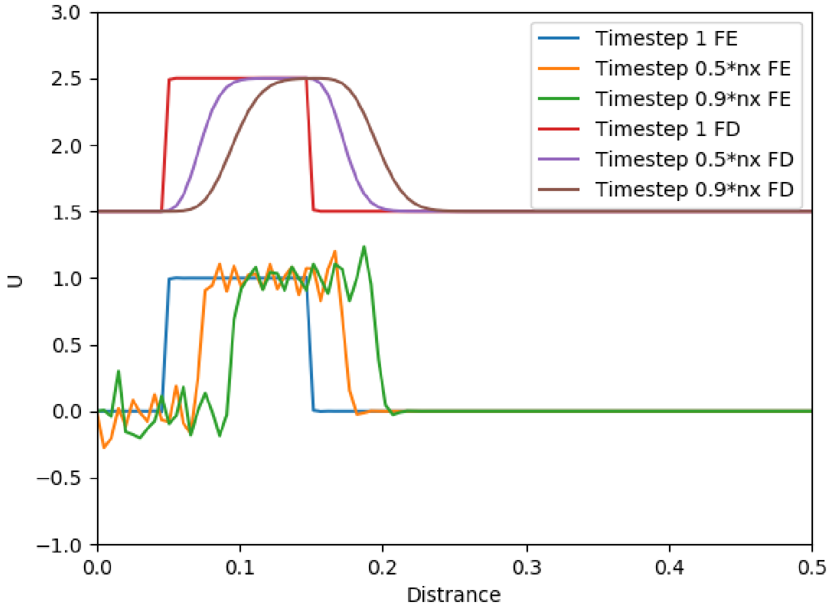

Figure 1: Number of time steps=1000, number of elements=100, Courant Number=0.01, velocity=0.1.

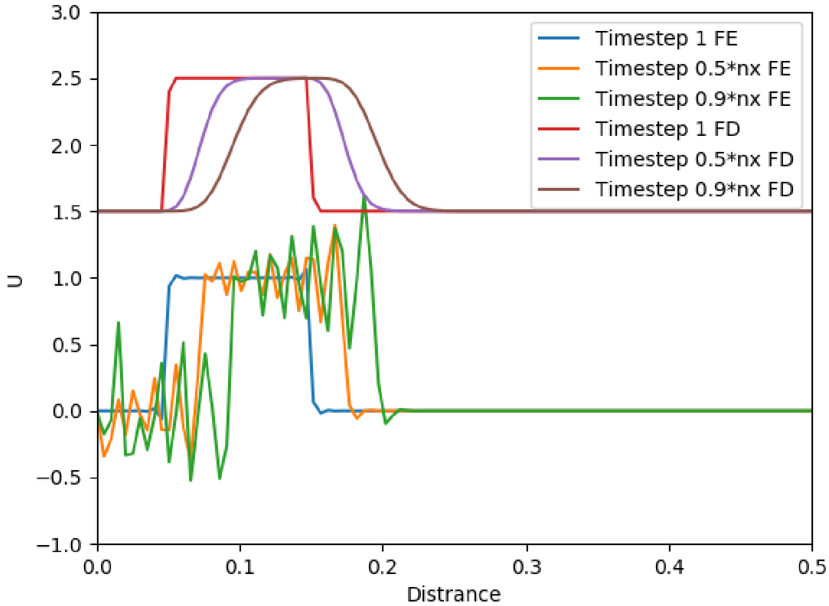

Figure 2: Number of time steps=100, number of elements=100, Courant Number=0.1, velocity=0.1.

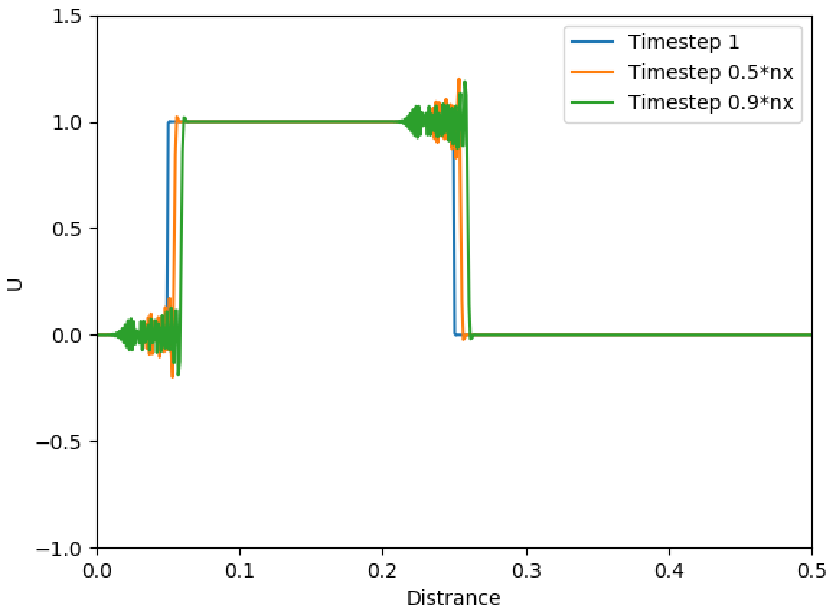

Figure 3: Number of time steps=1000, number of elements=500, Courant Number=0.01

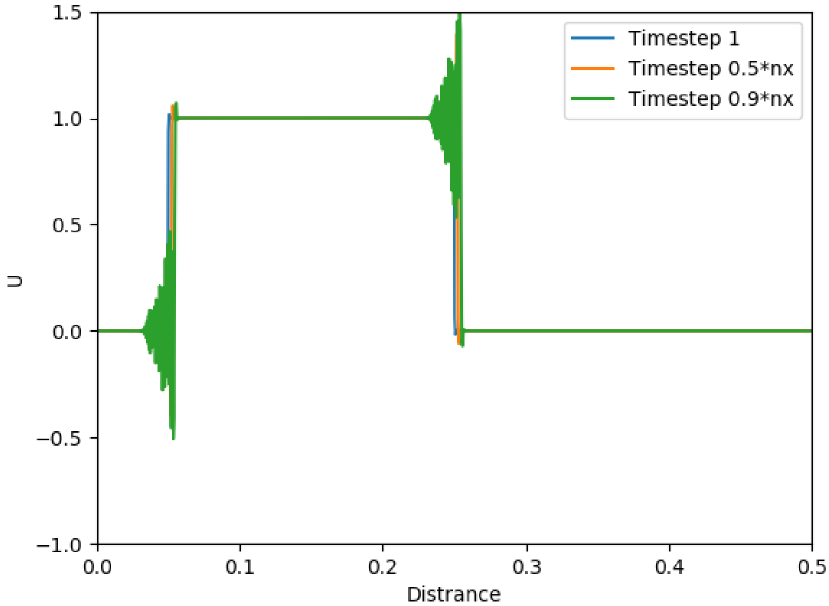

Figure 4: Number of time steps=100, number of elements= 1000, Courant Number=0.1

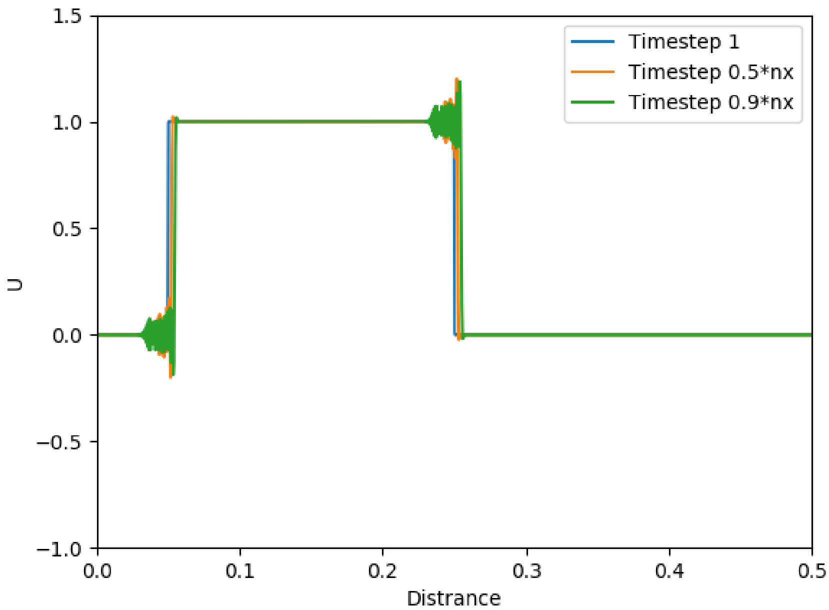

Figure 5: Number of time steps=1000, number of elements=1000, Courant Number=0.01

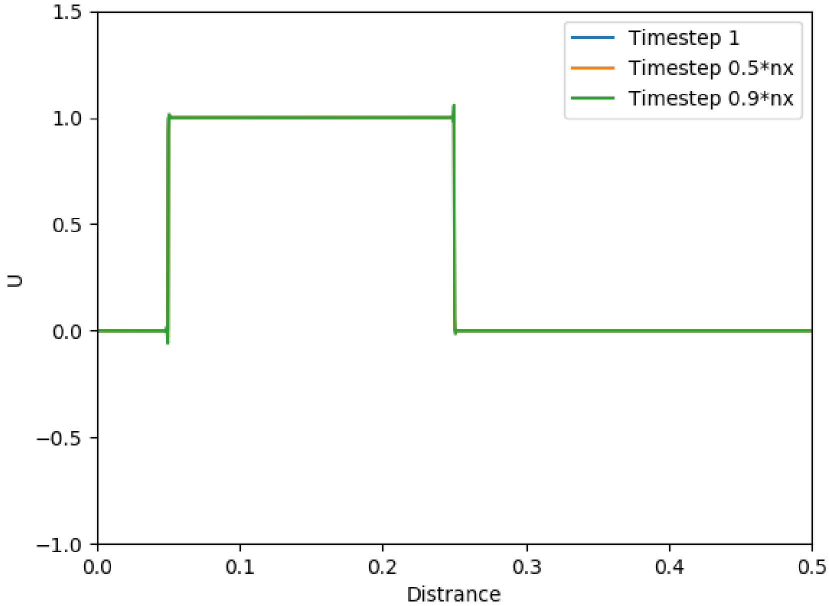

Figure 6: Number of time steps=100, number of elements=1000, Courant Number=0.001

Python Implementation

```python
import numpy as np
import matplotlib.pyplot as plt

nx = 200                           # distance which the wave travels
nt = 100                          # total number of time steps
L =  0.5                           # Totla length
C = 0.1                           # Courant number
c = 0.1                            # Wave velocity
dx = L/(nx-1)                      # Distace stepping size
dt = C*dx/c                        # Time stepping size
x  = np.arange(0, nx)*dx           # or x=np.linspace(0,2,nx)
U = np.zeros(nx)                   # U is a square wave between 0 <U< 1
U_FD = np.ones(nx)*1.5             # U is a square wave between 0 <U< 1
U_plot = np.ones((3,nx))           # A matrix to save 3 time steps used for plotting the results of FEM
U_FD_plot = np.ones((3,nx))        # An matrix to save 3 time steps used for plotting the results of FDM

#-----------------------------------------------------------------------------
#Boundary Conditions
U[0] = U[nx-1] = 0                 # Dirichlet BC for
```

```
U_FD[0] = U_FD[nx-1] = 1.5      # Dirichlet BC for FDM

#------------------------------------------------------------------------------
#Initial conditions
U[int(.1*L*nx):int(.5*L*nx)]=1
U_FD[int(.1*L*nx):int(.5*L*nx)]=2.5

#------------------------------------------------------------------------------
# Matrix A it is the LHS (mass matrix) shown in Equation 14
A = np.zeros((nx,nx))
for i in range(nx-1):
    for j in range (nx-1):
        if j==i:
            A[i,j] = dx/3*2
        elif j==i+1:
            A[i,j] = dx/6
        elif j==i-1:
            A[i,j] = dx/6
        else:
            A[i,j] = 0

#corner cells
A[0,0] = A[nx-1, nx-1] = dx/3
A[1,0] = A[0,1] = A[nx-1, nx-2] = A[nx-2, nx-1] = dx/6

 A_inv= np.linalg.inv(A)          # Inverse of matrix A used in Equation 15

#------------------------------------------------------------------------------
B = np.zeros((nx, nx))            # Matrix B in Equation 15
for i in range(nx):
    for j in range(nx):
        if  i==j+1:
            B[i,j] = 0.5
        elif i==j-1:
            B[i,j] = -0.5
B[0,0]=-0.5
B[nx-1,nx-1]=0.5

#------------------------------------------------------------------------------
# Without matrix: u(t+1)= u(t)+cdt A^(-1) B u(t)
dummy=A_inv.dot(B)                # A dummy vbl to make the RHS of Equation 15 clearer
                                  # Dummy is a nx by nx matrix
for n in range(1,nt):
    Un=U.copy()
    Un_FD=U_FD.copy()
    dummy2 = dummy.dot(Un)        # Dummy 2 creats a nx by 1 matrix
    for j in range(1,nx):
        U[j]=Un[j] + c*dt*dummy2[j]     # Overall equation
        U_FD[j] = U_FD[j]-c*(dt/dx)*(Un_FD[j]-Un_FD[j-1])

```

```
    if n==1:
        U_plot[0,:] = U.copy()
        U_FD_plot[0,:] = U_FD.copy()
    if n==int(nt/2):
        U_plot[1,:] = U.copy()
        U_FD_plot[1,:] = U_FD.copy()
    if n==int(nt*0.95):
        U_plot[2,:] = U.copy()
        U_FD_plot[2,:] = U_FD.copy()

plt.figure(1)
plt.axis([0,L, -1,3])
plt.xlabel('Distrance')
plt.ylabel('U')


# Plotting FEM---------------------------------------------------------------
plt.plot(x, U_plot[0,:], label='Timestep 1 FE')
plt.plot(x, U_plot[1,:], label='Timestep 0.5*nx FE')
plt.plot(x, U_plot[2,:], label='Timestep 0.9*nx FE')

# Plotting FDM---------------------------------------------------------------
plt.plot(x, U_FD_plot[0,:], label='Timestep 1 FD')
plt.plot(x, U_FD_plot[1,:], label='Timestep 0.5*nx FD')
plt.plot(x, U_FD_plot[2,:], label='Timestep 0.9*nx FD')

plt.legend()
```