DG FEM of Square Wave Equation in 1D and 2D

Amin Nadimy

Contents

1 Mathematical model 1D

$$\frac{\partial U}{\partial t} + \nabla.(cU) = 0 \tag{1}$$

Boundary Conditions:
$U(0,t) = U(l,t) = 0$, $0 \leq$ x $\leq$ l for $t \geq 0$
Initial conditions:
$U(x, t = 0) =$ Known (square wave)
Estimation of $U(t,x) \approx \bar{U}(t,x)$:

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

Now I multiply both sides by a weighting function w and integrate over the domain Ω.

$$\int_\Omega [\frac{\partial \bar{U}}{\partial t} w] d\Omega + \int_\Omega [\nabla.(c\bar{U})w] d\Omega = 0 \tag{3}$$

$$\int_\Omega [\frac{\partial \bar{U}}{\partial t} w - c\bar{U}\nabla.w] d\Omega + \int_\Gamma \hat{n}.c\bar{U} w d\Gamma = 0 \tag{4}$$

For and element e the equation becomes:

$$\int_{\Omega_e} \frac{\partial \bar{U}}{\partial t} w d\Omega_e - \int_{\Omega_e} c\bar{U}\nabla.w d\Omega_e + \int_{\Gamma_e} \hat{n}.c\bar{U} w d\Gamma_e = 0 \tag{5}$$

Taking $w = \phi_i$, where i is the i-th node in the element,which here i is 2 because of linear interpolation and Equation (5) becomes:

$$\int_{\Omega_e} \frac{\partial \bar{U}}{\partial t} \phi_i d\Omega_e - \int_{\Omega_e} [c\bar{U}\frac{d\phi_i}{dx}] dx = -\oint_{\Gamma_e} \hat{n}.c\bar{U}\phi_i d\Gamma_e \tag{6}$$

Here, j is the j-th function of U. Also let's take the length of each element as h.

$$\int_{x_0}^{x_1} \frac{\sum_{j=1}^{2}[U_j^t \phi_j(x)]}{\partial t} \sum_{i=1}^{2} \phi_i dx - \int_{x_0}^{x_1} [\sum_{j=1}^{2} [cU_j^t \phi_j(x)] \frac{d[\sum_{i=1}^{2} \phi_i]}{dx}] dx = - (\oint_{\Gamma_e} \hat{n}.c\bar{U}_{in}\phi_i d\Gamma_e + - \oint_{\Gamma_e} \hat{n}.c\bar{U}_{out}\phi_i d\Gamma_e) \tag{7}$$

$$\frac{\partial \sum_{j=1}^{2}[U_j^t]}{\partial t} \underbrace{\int_0^h [\sum_{i=1}^{2} \phi_i(x) \sum_{j=1}^{2} \phi_j(x)] dx}_{M} = \sum_{j=1}^{2}[U_j^t] \underbrace{\int_{x_0}^{x_1} [c \sum_{j=1}^{2} \phi_j(x)] \frac{d[\sum_{i=1}^{2} \phi_i]}{dx}] dx}_{K} - \underbrace{\left[c(x_1)\bar{U}^t(x_1^-)\phi_j(x_1)\phi_i(x_1) - c(x_0)\bar{U}^t(x_0^-)\phi_j(x_0)\phi_i(x_0)\right]}_{F} \quad (8)$$

$$M \frac{\partial \sum_{j=1}^{2}[U_j^t]}{\partial t} = K \sum_{j=1}^{2}[U_j^t] - F \quad (9)$$

Let's apply forward Euler's method to $\frac{\partial U}{\partial t}$:

$$\frac{\partial U_j^t}{\partial x} = \frac{U_j^{t+1} - U_j^t}{\Delta t} \quad (10)$$

$$M \frac{\sum_{j=1}^{2}[U_j^{t+1}] - \sum_{j=1}^{2}[U_j^t]}{\Delta t} = K \sum_{j=1}^{2}[U_j^t] - F \quad (11)$$

$$M \frac{\sum_{j=1}^{2}[U_j^{t+1}]}{\Delta t} = M \frac{\sum_{j=1}^{2}[U_j^t]}{\Delta t} + K \sum_{j=1}^{2}[U_j^t] - F \quad (12)$$

$$M \sum_{j=1}^{2}[U_j^{t+1}] = (M + \Delta t K) \sum_{j=1}^{2} [U_j^t] - \Delta t F \quad (13)$$

If we consider Upwind flux method with $c > 0$, x_0 and x_1 are the element boundaries, F becomes:

$$F = c(x_1)\bar{U}^t(x_1^-)\phi_j(x_1)\phi_i(x_1) - c(x_0)\bar{U}^t(x_0^-)\phi_j(x_0)\phi_i(x_0) \quad (14)$$

At x_0, $\phi_1 = 1$ and $\phi_2 = 0$ and at x_1, $\phi_1 = 0$ and $\phi_2 = 1$:

$$\phi_{1(x_1)}^2 - \phi_{1(x_0)} = -\phi_{1(x_0)}^2 = -1 \quad (15)$$

$$\phi_{2(x_1)}^2 - \phi_{2(x_0)}^2 = 1 \quad (16)$$

Therefore the D matrix for element k becomes:

$$F = c \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U_{(x_0)}^- \\ U_{(x_1)}^- \end{bmatrix} \quad (17)$$

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 $M = \int_0^h \phi_i \phi_j d\bar{x}$ and if c is constant then $K = c \int_0^h \phi_j \frac{d\phi_i}{d\bar{x}} d\bar{x}$:
For $i = 1$ and $j = 1$:

$$M = \frac{h}{3} \tag{18}$$

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

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

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

$$K = \frac{-1}{2} \tag{21}$$

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

$$M = \frac{h}{6} \tag{22}$$

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

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

$$M = \frac{h}{3} \tag{24}$$

$$K = \frac{1}{2} \tag{25}$$

$K = \begin{bmatrix} \frac{-1}{2} & \frac{-1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$, $M = \begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix}$

$$\begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \begin{bmatrix} U_1^{t+1} \\ U_2^{t+1} \end{bmatrix} = \begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \begin{bmatrix} U_1^t \\ U_2^t \end{bmatrix} + \begin{bmatrix} \frac{-c\Delta t}{2} & \frac{-c\Delta t}{2} \\ \frac{c\Delta t}{2} & \frac{c\Delta t}{2} \end{bmatrix} \begin{bmatrix} U_1^t \\ U_2^t \end{bmatrix} - \begin{bmatrix} -c\Delta t & 0 \\ 0 & c\Delta t \end{bmatrix} \begin{bmatrix} U_{(x_0)}^- \\ U_{(x_1)}^- \end{bmatrix} \tag{26}$$

$$\begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} \begin{bmatrix} U_1^{t+1} \\ U_2^{t+1} \end{bmatrix} = \left(\begin{bmatrix} \frac{h}{3} & \frac{h}{6} \\ \frac{h}{6} & \frac{h}{3} \end{bmatrix} + \begin{bmatrix} \frac{-c\Delta t}{2} & \frac{-c\Delta t}{2} \\ \frac{c\Delta t}{2} & \frac{c\Delta t}{2} \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & c\Delta t \end{bmatrix} \right) \begin{bmatrix} U_1^t \\ U_2^t \end{bmatrix} \tag{27}$$

1.1 Consistent mass matrix

For two elements:

$$\underbrace{\begin{bmatrix} \frac{h}{3} & \frac{h}{6} & 0 & 0 \\ \frac{h}{6} & \frac{h}{3} & 0 & 0 \\ 0 & 0 & \frac{h}{3} & \frac{h}{6} \\ 0 & 0 & \frac{h}{6} & \frac{h}{3} \end{bmatrix}}_{M} \begin{bmatrix} U_1^{t+1} \\ U_2^{t+1} \\ U_3^{t+1} \\ U_4^{t+1} \end{bmatrix} =$$

$$\left(\underbrace{\begin{bmatrix} \frac{h}{3} & \frac{h}{6} & 0 & 0 \\ \frac{h}{6} & \frac{h}{3} & 0 & 0 \\ 0 & 0 & \frac{h}{3} & \frac{h}{6} \\ 0 & 0 & \frac{h}{6} & \frac{h}{3} \end{bmatrix}}_{M} + \underbrace{\begin{bmatrix} \frac{-c\Delta t}{2} & \frac{-c\Delta t}{2} & 0 & 0 \\ \frac{c\Delta t}{2} & \frac{c\Delta t}{2} & 0 & 0 \\ 0 & 0 & \frac{-c\Delta t}{2} & \frac{-c\Delta t}{2} \\ 0 & 0 & \frac{c\Delta t}{2} & \frac{c\Delta t}{2} \end{bmatrix}}_{K} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -c\Delta t & 0 & 0 \\ 0 & c\Delta t & 0 & 0 \\ 0 & 0 & 0 & -c\Delta t \end{bmatrix}}_{F} \right) \begin{bmatrix} U_1^t \\ U_2^t \\ U_3^t \\ U_4^t \end{bmatrix} \tag{28}$$

$$MU_i^{t+1} = (M + K + F)U_i^t \tag{29}$$

$$U_i^{t+1} = M^{-1}(M + K + F)U_i^t \tag{30}$$

1.2 Lumped matrix

Diagonalising mas matrix using row sum method gives:

$$\underbrace{\begin{bmatrix} \frac{h}{2} & 0 & 0 & 0 \\ 0 & \frac{h}{2} & 0 & 0 \\ 0 & 0 & \frac{h}{2} & 0 \\ 0 & 0 & 0 & \frac{h}{2} \end{bmatrix}}_{M} \begin{bmatrix} U_1^{t+1} \\ U_2^{t+1} \\ U_3^{t+1} \\ U_4^{t+1} \end{bmatrix} =$$

$$\left(\underbrace{\begin{bmatrix} \frac{h}{2} & 0 & 0 & 0 \\ 0 & \frac{h}{2} & 0 & 0 \\ 0 & 0 & \frac{h}{2} & 0 \\ 0 & 0 & 0 & \frac{h}{2} \end{bmatrix}}_{M} + \underbrace{\begin{bmatrix} \frac{-c\Delta t}{2} & \frac{-c\Delta t}{2} & 0 & 0 \\ \frac{c\Delta t}{2} & \frac{c\Delta t}{2} & 0 & 0 \\ 0 & 0 & \frac{-c\Delta t}{2} & \frac{-c\Delta t}{2} \\ 0 & 0 & \frac{c\Delta t}{2} & \frac{c\Delta t}{2} \end{bmatrix}}_{K} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -c\Delta t & 0 & 0 \\ 0 & c\Delta t & 0 & 0 \\ 0 & 0 & 0 & -c\Delta t \end{bmatrix}}_{F} \right) \begin{bmatrix} U_1^t \\ U_2^t \\ U_3^t \\ U_4^t \end{bmatrix} \tag{31}$$

Explicit equations for local U_i^{t+1} :

$$U_1^{t+1} = U_1^t - \frac{c\Delta t}{h}(U_1^t + U_2^t) \tag{32}$$

$$U_2^{t+1} = U_2^t + \frac{c\Delta t}{h}(U_1^t + U_2^t) - c\Delta t U_2^t \tag{33}$$

$$U_3^{t+1} = U_3^t - \frac{c\Delta t}{h}(U_3^t + U_4^t) + c\Delta t U_2^t \tag{34}$$

$$U_4^{t+1} = U_4^t + \frac{c\Delta t}{h}(U_3^t + U_4^t) - c\Delta t U_4^t \tag{35}$$

2 Results

2.1 Explicit method with lumped mass matrix

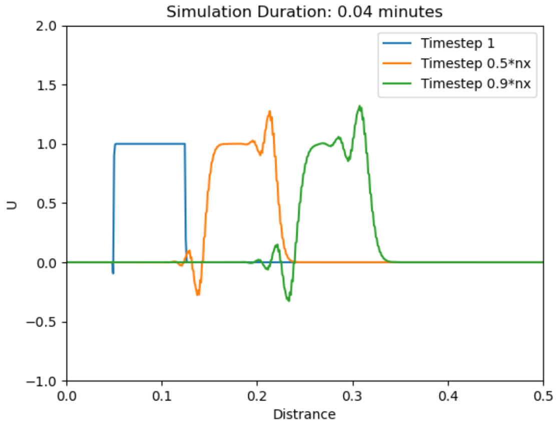


Figure 1: Number of time steps=2000, number of nx=500, CFL=0.05, velocity=0.1.

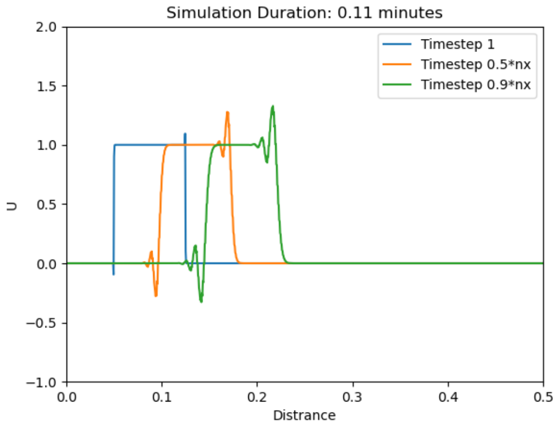


Figure 2: Number of time steps=2000, nx=1000, CFL=0.05, velocity=0.1.

2.2 Explicit method with consistent mass matrix

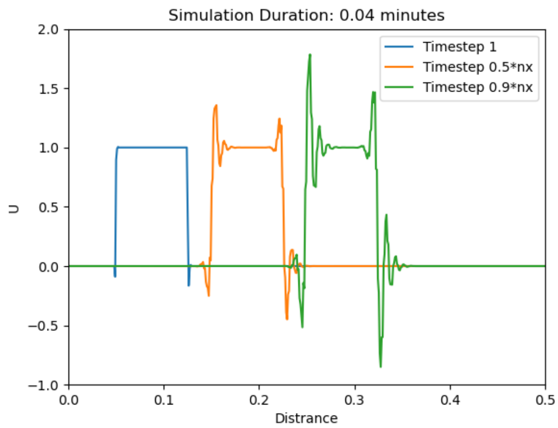


Figure 3: Number of time steps=2000, number of nx=500, CFL=0.05, velocity=0.1.

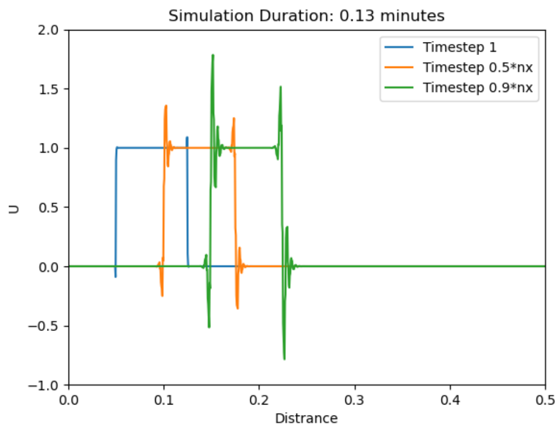


Figure 4: Number of time steps=2000, nx=1000, CFL=0.05, velocity=0.1.

2.3 Implicit method with consistent mass matrix

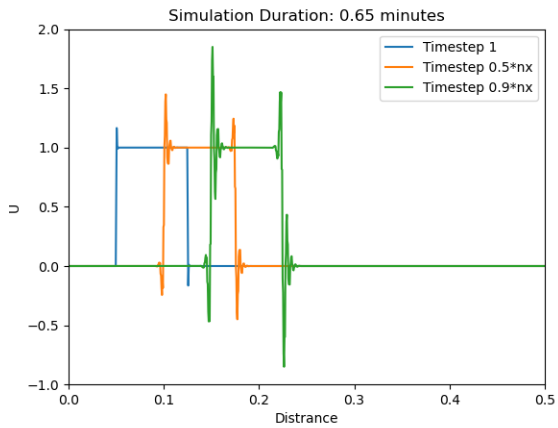


Figure 5: Number of time steps=2000, number of nx=1000, CFL=0.05, velocity=0.1.

3 Python codes

3.1 Explicit method with lumped mass matrix

```python
#-------------------------------------------------1D-DG FEM---------------------------------------------
import numpy as np
import matplotlib.pyplot as plt
import time

nx = 1000                    # total number of nodes(degree of freedom)
nt = 2000                    # total number of time steps
L =  0.5                     # Totla length
C = .05                      # Courant number
c = .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_plot = np.ones((3,nx))     # A matrix to save 3 time steps used for plotting the results

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

#---------------------------------------------------------------------------------------------------------
# Initial conditions
U[int(L*nx*0.2):int(L*nx*0.5)]=1     # defining square wave shape

#--------------------------------------------Explicit method-------------------------------------------
Un=np.zeros(nx)                     # dummy vbl to save current values of U (U^t)
for n in range(nt):                 # Marching in time
    Un = U.copy()                   # saving U^t to be used in the next time step calculation
    U[1] = Un[1] - c* dt/dx *(Un[1]+Un[2])
    i=2
    while i<nx-1:
        U[i] = Un[i] + (-1)**i*c*dt/dx*(Un[i] + Un[i-1]) + (-1)**(i+1)*c*Un[i]
        i +=1
        if i==nx-1:
            StopIteration
        else:
            U[i] = Un[i] + (-1)**i*c*dt/dx*(Un[i] + Un[i+1]) + (-1)**(i+1)*c*Un[i-1]
        i +=1

    if n==1:
        U_plot[0,:] = U.copy()      # saving U(t=1)
    if n==int(nt/2):
        U_plot[1,:] = U.copy()      # saving U(t=nt/2)
    if n==int(nt*0.99):
```

```python
        U_plot[2,:] = U.copy()      # saving U(t= almost the end to time steps)

#--------------------------------plot initiation ------------------------------------------
plt.figure(1)
plt.axis([0,L, -1,2])
plt.plot(x, U_plot[0,:], label='Timestep 1')
plt.plot(x, U_plot[1,:], label='Timestep 0.5*nx')
plt.plot(x, U_plot[2,:], label='Timestep 0.9*nx')
plt.xlabel('Distrance')
plt.ylabel('U')
plt.legend()
```

3.2 Implicit method with consistent mass matrix

```
#---------------------------------------------------------1D-DG FEM---------------------------------------------------
import numpy as np
import matplotlib.pyplot as plt
import time

nx = 1000                          # total number of nodes(degree of freedom)
nt = 2000                          # total number of time steps
L =  0.5                           # Totla length
C = 0.05                           # 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_plot = np.ones((3,nx))           # A matrix to save 3 time steps used for plotting the
    results

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

#-------------------------------------------------------------------------------------------------------------------
# Initial conditions
U[int(L*nx*0.2):int(L*nx*0.5)]=1      # defining square wave shape

#---------------------------------------Mass Matrix 'M' in Equation 29 -------------------------------------------
t1 = time.time()                          # starting for timing the M_diag_inv calculation
sub_M = np.array([[dx/3,dx/6],[dx/6,dx/3]]) # local mass matrix
M=np.zeros((nx-2,nx-2))                              # generating global mass matrix
i=0
while i<nx-2:
    M[i:i+2, i:i+2]= sub_M[0:2,0:2]
    i+=2
M_inv=np.linalg.inv(M)
t2 = time.time()                               # end point of M_diag_inv generation
print(str(t2-t1))
#---------------------------------------Stifness Matrix 'K' in Equation 29-------------------------------------------
sub_K=np.array([[-c*dt/2,-c*dt/2],[c*dt/2,c*dt/2]]) # local stifness matrix
K = np.zeros((nx-2,nx-2))                                    # generating global stifness
    matix
i=0
while i<nx-2:
    K[i:i+2, i:i+2]= sub_K[0:2,0:2]
    i+=2

# #----------------------------------------Flux matrix in Equation 29------------------------------------------------
F = np.zeros((nx-2,nx-1))
i=0
```

```
j=1
while i<=nx-3:
     F[i,i]= c*dt
     F[j,j+1] = -c*dt
     i+=2
     j+=2
F=F[:,1:]                              # excludig left boundary to get nx by nx matrix

#-----------------------------------RHS in equation 29-----------------------------------
RHS_cst = M_inv.dot((M + K + F))

#-------------------------------Explicit using consistance mass matrix-------------------
Un=np.zeros(nx)                        # dummy vbl to save current values of U (U^t)
for n in range(nt):                    # Marching in time
     Un = U.copy()                     # saving U^t to be used in the next time step calculation
     U[1] = RHS_cst[0,0]*Un[1] + RHS_cst[0,1]*Un[2]
     U[2] = RHS_cst[1,0]*Un[1] + RHS_cst[1,1]*Un[2]
     i=3
     j=1
     while i<nx-2:
          U[i] = RHS_cst[i-1,j]*Un[i-1] + RHS_cst[i-1,j+1]*Un[i] + RHS_cst[i-1,j+2]*Un[i+1]
          i+=1
          U[i] = RHS_cst[i-1,j]*Un[i-2] + RHS_cst[i-1,j+1]*Un[i-1] + RHS_cst[i-1,j+2]*Un[i]
          i+=1
          j+=2
     if n==1:
          U_plot[0,:] = U.copy()       # saving U(t=1)
     if n==int(nt/2):
          U_plot[1,:] = U.copy()       # saving U(t=nt/2)
     if n==int(nt*0.99):
          U_plot[2,:] = U.copy()       # saving U(t= almost the end to time steps)

#---------------------------------plot initiation ---------------------------------------
plt.figure(1)
plt.axis([0,L, -1,2])
plt.plot(x, U_plot[0,:], label='Timestep 1')
plt.plot(x, U_plot[1,:], label='Timestep 0.5*nx')
plt.plot(x, U_plot[2,:], label='Timestep 0.9*nx')
plt.xlabel('Distrance')
plt.ylabel('U')
plt.legend()
```

3.3 Implicit method with consistent mass matrix

```
#--------------------------------------------------------1D-DG FEM--------------------------------------------
import numpy as np
import matplotlib.pyplot as plt
import time

nx = 1000                        # total number of nodes(degree of freedom)
nt = 1000                        # total number of time steps
L =  0.5                         # Totla length
C = .05                          # Courant number
c = .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_plot = np.ones((3,nx))         # A matrix to save 3 time steps used for plotting the results

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

#---------------------------------------------------------------------------------------------------------------
# Initial conditions
U[int(L*nx*0.2):int(L*nx*0.5)]=1      # defining square wave shape

#------------------------------------Mass Matrix 'M' in Equation 29 ---------------------------------------------
t1 = time.time()                          # starting for timing the M_diag_inv calculation
sub_M = np.array([[dx/3,dx/6],[dx/6,dx/3]]) # local mass matrix
M=np.zeros((nx,nx))                          # generating global mass matrix
i=0
while i<nx:
    M[i:i+2, i:i+2]= sub_M[0:2,0:2]
    i+=2

t2 = time.time()                             # end point of M_diag_inv generation
print(str(t2-t1))
#-----------------------------------Stifness Matrix 'K' in Equation 29-------------------------------------------
sub_K=np.array([[-c*dt/2,-c*dt/2],[c*dt/2,c*dt/2]]) # local stifness matrix
K = np.zeros((nx,nx))                                  # generating global stifness matix
i=0
while i<nx:
    K[i:i+2, i:i+2]= sub_K[0:2,0:2]
    i+=2

# #-----------------------------------------Flux matrix in Equation 29--------------------------------------------
F = np.zeros((nx,nx+1))
i=0
```

```
j=1
while i<=nx-1:
    F[i,i]= c*dt
    F[j,j+1] = -c*dt
    i+=2
    j+=2
F=F[:,1:]                               # excludig left boundary to get nx by nx matrix

#-----------------------------------RHS in equation 29------------------------------------
RHS_cst = (M + K + F)

#-------------------------------Implicit using consistance mass matrix------------------------
Un=np.zeros(nx)                         # dummy vbl to save current values of U (U^t)
for n in range(nt):                     # Marching in time
    Un = U.copy()                       # saving U^t to be used in the next time step
    calculation
    RHS = RHS_cst.dot(Un)
    U[1:nx-1] = np.linalg.solve(M[1:nx-1, 1:nx-1],RHS[1:nx-1])
    if n==1:
        U_plot[0,:] = U.copy()          # saving U(t=1)
    if n==int(nt/2):
        U_plot[1,:] = U.copy()          # saving U(t=nt/2)
    if n==int(nt*0.99):
        U_plot[2,:] = U.copy()          # saving U(t= almost the end to time steps)

#---------------------------------plot initiation ------------------------------------------
plt.figure(1)
plt.axis([0,L, -1,2])
plt.plot(x, U_plot[0,:], label='Timestep 1')
plt.plot(x, U_plot[1,:], label='Timestep 0.5*nx')
plt.plot(x, U_plot[2,:], label='Timestep 0.9*nx')
plt.xlabel('Distrance')
plt.ylabel('U')
plt.legend()
```

article [utf8]inputenc amsmath graphicx caption color [a4paper, total=6in, 8in]geometry tikz listings amsthm chemformula tikz

4 Mathematical model 2D

$$\frac{\partial U}{\partial t} + \nabla .cU = 0 \tag{36}$$

Boundary Conditions:
$U(0, y, t) = 1$ and $U(a, y, t) = 0$, where $0 \leq x \leq a$ for $t \geq 0$
$U(x, 0, t) = 1$ and $U(x, b, t) = 0$, where $0 \leq y \leq b$ for $t \geq 0$
Initial Conditions:
$U(x, y, t = 0) =$Known square wave.
Estimation of $U(t, x, y) \approx \bar{U}(t, x, y)$:

$$\bar{U}(t, x, y) = \sum_{j=1}^{4} U_j^t \phi_j(x, y) \tag{37}$$

Now I multiply both sides by a weighting function w and integrate over the domain Ω.

$$\int_\Omega [\frac{\partial \bar{U}}{\partial t} w] d\Omega + \int_\Omega [\nabla .c\bar{U}w] d\Omega = 0 \tag{38}$$

Integration by part:

$$\int_\Omega [\frac{\partial \bar{U}}{\partial t} w] d\Omega + \oint_\Gamma \hat{n}.c\bar{U}wd\Gamma - \int_\Omega [c\bar{U}.\nabla w] d\Omega = 0 \tag{39}$$

The above equation at the elemental level becomes:

$$\int_{\Omega_e} [\frac{\partial \bar{U}}{\partial t} w] d\Omega_e + \oint_{\Gamma_e} \hat{n}.c\bar{U}wd\Gamma_e - \int_{\Omega_e} [c\bar{U}.\nabla w] d\Omega_e = 0 \tag{40}$$

Estimating the weighting function as:

$$w = \sum_{i=1}^{4} \phi_i(x, y) \tag{41}$$

$$\sum_{i=1}^{4} \int_{\Omega_e} [\frac{\partial \bar{U}}{\partial t} \phi_i] d\Omega_e = \sum_{i=1}^{4} \int_{\Omega_e} [c\bar{U}.\nabla \phi_i] d\Omega_e - \sum_{i=1}^{4} \oint_{\Gamma_e} [\hat{n}_x.c\bar{U}\phi_i + \hat{n}_y.c\bar{U}\phi_i] d\Gamma_e \tag{42}$$

Implementing (37) into (42):

$$\sum_{i=1}^{4} \sum_{j=1}^{4} \frac{\partial U_j^t}{\partial t} \int_{\Omega_e} [\phi_j \phi_i] d\Omega_e = \sum_{i=1}^{4} \sum_{j=1}^{4} cU_j^t \int_{\Omega_e} [\phi_j.\nabla \phi_i] d\Omega_e - \sum_{i=1}^{4} \sum_{j=1}^{4} cU_j^t \oint_{\Gamma_e} [\hat{n}_x.\phi_j \phi_i + \hat{n}_y.\phi_j \phi_i] d\Gamma_e \tag{43}$$

Let's apply forward Euler's method to $\frac{\partial U}{\partial t}$:

$$\frac{\partial U_j^t}{\partial x} = \frac{U^{t+1} - U^t}{\Delta t} \tag{44}$$

$$\sum_{i=1}^{4}\sum_{j=1}^{4} \frac{U_j^{t+1} - U_j^t}{\Delta t} \int_{\Omega_e} [\phi_j \phi_i] d\Omega_e = \sum_{i=1}^{4}\sum_{j=1}^{4} cU_j^t \int_{\Omega_e} [\phi_j . \nabla \phi_i] d\Omega_e - \sum_{i=1}^{4}\sum_{j=1}^{4} cU_j^t \oint_{\Gamma_e} [\hat{n}_x . \phi_j \phi_i + \hat{n}_y . \phi_j \phi_i] d\Gamma_e \tag{45}$$

$$\sum_{i=1}^{4}\sum_{j=1}^{4} \frac{U_j^{t+1}}{\Delta t} \int_{\Omega_e} [\phi_j \phi_i] d\Omega_e = \sum_{i=1}^{4}\sum_{j=1}^{4} \frac{U_j^t}{\Delta t} \int_{\Omega_e} [\phi_j \phi_i] d\Omega_e + \sum_{i=1}^{4}\sum_{j=1}^{4} cU_j^t \int_{\Omega_e} [\phi_j . \nabla \phi_i] d\Omega_e - \sum_{i=1}^{4}\sum_{j=1}^{4} cU_j^t \oint_{\Gamma_e} [\hat{n}_x . \phi_j \phi_i + \hat{n}_y . \phi_j \phi_i] d\Gamma_e \tag{46}$$

The matrix form becomes:

$$[M_{i,j}] \left\{U_j^{t+1}\right\} = [M_{i,j}] \left\{U_j^t\right\} + [K_{i,j}] \left\{U_j^t\right\} - [F_{i,j}] \left\{U_j^t\right\} \tag{47}$$

$$[M_{i,j}] \left\{U_j^{t+1}\right\} = [[M_{i,j}] + [K_{i,j}] - [F_{i,j}]] \left\{U_j^t\right\} \tag{48}$$

$$[M_{i,j}] = \sum_{i=1}^{4}\sum_{j=1}^{4} \int_{\Omega_e} [\phi_j \phi_i] d\Omega_e \tag{49}$$

$$[K_{i,j}] = \sum_{i=1}^{4}\sum_{j=1}^{4} c \int_{\Omega_e} [\phi_j . \nabla \phi_i] d\Omega_e \tag{50}$$

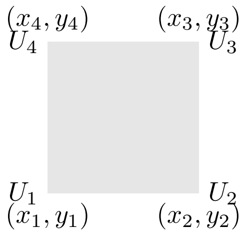
$$[F_{i,j}] = \sum_{i=1}^{4}\sum_{j=1}^{4} c\Delta t \oint_{\Gamma_e} [\hat{n}_x . \phi_j \phi_i + \hat{n}_y . \phi_j \phi_i] d\Gamma_e \tag{51}$$

4.1 Interpolation functions ϕ_i at the local level:

For rectangular element, there are four nodes with the local coordinates:

$$\begin{aligned} \text{node } 1 &= (x_1, y_1) = (0, 0) \\ \text{node } 2 &= (x_2, y_2) = (a, 0) \\ \text{node } 3 &= (x_3, y_3) = (a, b) \\ \text{node } 4 &= (x_4, y_4) = (0, b) \end{aligned}$$

Where a and b are the length of the element in x and y directions, respectively.



$$U^e(\bar{x}, \bar{y}) = C_1 + C_2\bar{x} + C_3\bar{y} + C_4\bar{x}\bar{y} = \sum_{j=1}^{4} U_j \phi_j(\bar{x}\bar{y}) \tag{52}$$

Where $\bar{x}$ and $\bar{y}$ are local coordinates.

at node 1: $U^e(0, 0) = U_1 = C_1 \rightarrow C_1 = U_1$

at node 2: $U^e(a, 0) = U_2 = C_1 + C_2 a \rightarrow C_2 = \frac{U_2 - U_1}{a}$

at node 3: $U^e(0, b) = U_3 = C_1 + C_3 b \rightarrow C_3 = \frac{U_4 - U_1}{b}$

at node 4: $U^e(a, b) = U_3 = C_1 + C_2 a + C_3 b + C_4 ab \rightarrow C_4 = \frac{(U_3 - U_4(+(U_1 - U_3)}{ab}$

Putting C_1, C_2, C_3 and C_4 into (52):

$$U^e(\bar{x}, \bar{y}) = U_1 \underbrace{(1 - \frac{\bar{x}}{a})(1 - \frac{\bar{y}}{b})}_{\phi_1} + U_2 \underbrace{\frac{\bar{x}}{a}(1 - \frac{\bar{y}}{b})}_{\phi_2} + U_3 \underbrace{\frac{\bar{x}\bar{y}}{ab}}_{\phi_3} + U_4 \underbrace{\frac{\bar{y}}{b}(1 - \frac{\bar{x}}{a})}_{\phi_4} \tag{53}$$

$$\phi_i^e(\bar{x}, \bar{y}) = (-1)^{i+1}[1 - \frac{\bar{x} + x_i}{a}][1 - \frac{\bar{y} + y_i}{b}] \tag{54}$$

$$\begin{cases} \frac{\partial \phi_1}{\partial \bar{x}} = \frac{\bar{y} - b}{ab} \\ \frac{\partial \phi_1}{\partial \bar{y}} = \frac{\bar{x} - b}{ab} \end{cases}, \begin{cases} \frac{\partial \phi_2}{\partial \bar{x}} = \frac{\bar{y} - b}{ab} \\ \frac{\partial \phi_2}{\partial \bar{y}} = \frac{-\bar{x}}{ab} \end{cases}, \begin{cases} \frac{\partial \phi_3}{\partial \bar{x}} = \frac{\bar{y}}{ab} \\ \frac{\partial \phi_3}{\partial \bar{y}} = \frac{\bar{x}}{ab} \end{cases} \text{and} \begin{cases} \frac{\partial \phi_4}{\partial \bar{x}} = \frac{-\bar{y}}{ab} \\ \frac{\partial \phi_4}{\partial \bar{y}} = \frac{\bar{x} - a}{ab} \end{cases}.$$

4.2 Calculating $[M_{i,j}]$ components:

Coordinate transformation:

If x_1 and y_1 are coordinates of node 1 of the element in the global system: $\begin{cases} x = \bar{x} + x_1^e \\ y = \bar{y} + y_1^e \\ dx = d\bar{x} \\ dy = d\bar{y} \end{cases}$

$$M_{i,j} = \int_0^a \int_0^b \phi_1 \phi_j dxdy \tag{55}$$

Calculating for all i and j:

$$M = \frac{ab}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 2 & 4 & 2 \\ 2 & 1 & 2 & 4 \end{bmatrix} \tag{56}$$

4.3 Calculating $[K_{i,j}]$ components:

$$[K_{i,j}] = \sum_{i=1}^{4} \sum_{j=1}^{4} c \int_{\Omega_e} [\phi_j . \nabla \phi_i] d\Omega_e = \sum_{i=1}^{4} \sum_{j=1}^{4} c \int_0^a \int_0^b [\phi_j \frac{\partial \phi_i}{\partial x} + \phi_j \frac{\partial \phi_i}{\partial y}] dxdy \tag{57}$$

Calculating for all i and j:

$$K = \begin{bmatrix} -\frac{b}{6} + \frac{a-3b}{12} & \frac{-b}{6} - \frac{a}{12} & \frac{ab}{12} + \frac{a}{12} & \frac{-b}{12} - \frac{a}{6} \\ \frac{-b}{6} + \frac{2a-3b}{12} & \frac{-b}{6} - \frac{1}{6} & \frac{b}{12} - \frac{a}{6} & \frac{-b}{12} - \frac{a}{12} \\ \frac{-b}{12} + \frac{2a-3b}{12} & \frac{-b}{12} - \frac{a}{6} & \frac{b}{6} + \frac{a}{6} & \frac{-b}{6} - \frac{a}{12} \\ \frac{-b}{12} + \frac{a-3b}{12} & \frac{-b}{12} - \frac{a}{12} & \frac{b}{6} + \frac{a}{12} & \frac{-b}{6} - \frac{a}{6} \end{bmatrix} \tag{58}$$

4.4 Calculating $[F_{i,j}]$*components*:

$$\begin{gathered} [F_{i,j}] = \\ \left[c(x_2)\bar{U}^t(x_2^-)\phi_j(x_2)\phi_i(x_2) - c(x_1)\bar{U}^t(x_1^-)\phi_j(x_1)\phi_i(x_1)\right] + \\ \left[c(x_3)\bar{U}^t(x_3^-)\phi_j(x_3)\phi_i(x_3) - c(x_4)\bar{U}^t(x_4^-)\phi_j(x_4)\phi_i(x_4)\right] + \\ \left[c(y_4)\bar{U}^t(y_4^-)\phi_j(y_4)\phi_i(y_4) - c(y_1)\bar{U}^t(y_1^-)\phi_j(y_1)\phi_i(y_1)\right] + \\ \left[c(y_3)\bar{U}^t(y_3^-)\phi_j(y_3)\phi_i(y_3) - c(y_2)\bar{U}^t(y_2^-)\phi_j(y_2)\phi_i(y_2)\right] \end{gathered} \tag{59}$$

$$F = \begin{bmatrix} 1|1 & 0|0 & 0|0 & 0|0 \\ 0|0 & -1|1 & 0|0 & 0|0 \\ 0|0 & 0|0 & -1|-1 & 0|0 \\ 0|0 & 0|0 & 0|0 & 1|-1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{60}$$