[1]:
```python
import numpy as np
from matplotlib import pyplot as plt
```

Course: Computational Electrodynamics and Applications

Tutor: Theodoros Samaras

Hand-in date: April 6, 2020

Student: Konstantinos Foutzopoulos < kfoutzop@auth.gr >

0.1 Numerical study of 1D wave equation

- The code required for the study is given below.

[2]:
```python
def weq_es(u0, um1, c, dt, dx, Nt, Nx):
    '''
    Explicit scheme of second-order (using central differences)
    Given in [GedneyEE624]; solved in vector form
    '''
    cx2 = (c*dt/dx)**2

    u = np.empty((Nt, Nx))
    u[0, :] = u0

    u[1, 1:-1] = -um1[1:-1] + 2*u[0, 1:-1] + \
            cx2*(u[0, 2:] - 2*u[0, 1:-1] + u[0, :-2])
    u[:, 0] = u[:, -1] = 0

    for n in range(1, Nt-1):
        u[n+1, 1:-1] = -u[n-1, 1:-1] + 2*u[n, 1:-1] + \
            cx2*(u[n, 2:] - 2*u[n, 1:-1] + u[n, :-2])

    return u
```

[3]:
```python
def weq_is(u0, um1, c, dt, dx, Nt, Nx):
    '''
    Implicit scheme of second-order (using Newmark scheme)
    Given in [GedneyEE624]; solved in matrix form
    '''
    cx2 = (c*dt/dx)**2

    L = 2*np.identity(Nx) \
            - np.diag(np.ones(Nx-1), -1) \
            - np.diag(np.ones(Nx-1), 1)        #tridiag([-1,2,-1])

    beta = 1/4

    I = np.identity(Nx)
```

```
    A = beta*L + (1/cx2)*I
    B = (2*beta-1)/2*L + (1/cx2)*I
    Ainv = np.linalg.inv(A)

    u = np.empty((Nt, Nx))
    u[0, :] = u0

    u[1, :] = 2*Ainv @ B @ u[0, :] - um1
    u[1, 0] = u[1, -1] = 0

    for n in range(1, Nt-1):
        u[n+1, :] = 2*Ainv @ B @ u[n, :] - u[n-1, :]
        u[n+1, 0] = u[n+1, -1] = 0

    return u
```

- Assuming that

[4]:
```
c = 1
```

- Asssuming a rectangular pulse as the initial considition.

Not explicitly mentioned, so we consider that $i \in [0, Nx]$, or else an index-0 numbering is used. Under this, pulse starts in the third (i=2) space step, where the first (i=0) space step should be the boundary. Also, required are the space length. As not an exact number was given we consider that sufficiently big.

[5]:
```
L = 10
Nx = 100
```

[6]:
```
def rec_pulse(lo, hi, Nx, ex):
    v = np.zeros(Nx)
    v[ex[0]:ex[1]+1] = 1
    return v

def rec_pulse_mv(v, t):
    hidx = v.argmax()
    ex = [hidx+t, hidx+v[hidx:].argmin()-1+t]
    return rec_pulse(v.min(), v[hidx], np.size(v), ex)

um1 = rec_pulse(0, 1, Nx, [2, 11])
u0 = rec_pulse_mv(um1, 1)
```

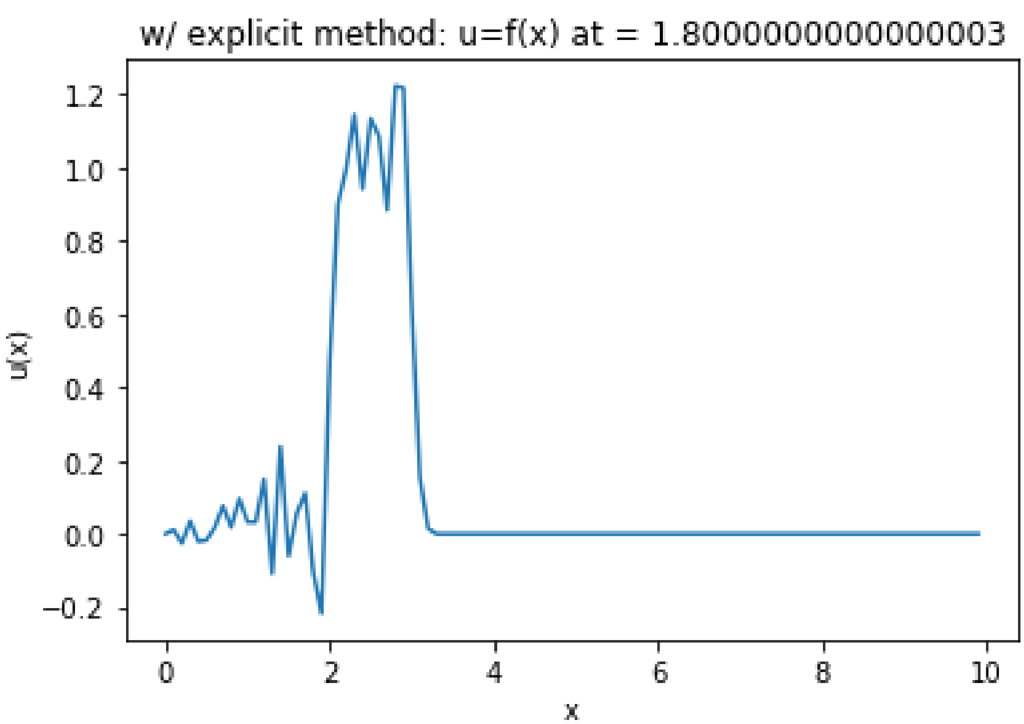
- Recording and showing snapshots of the wave.

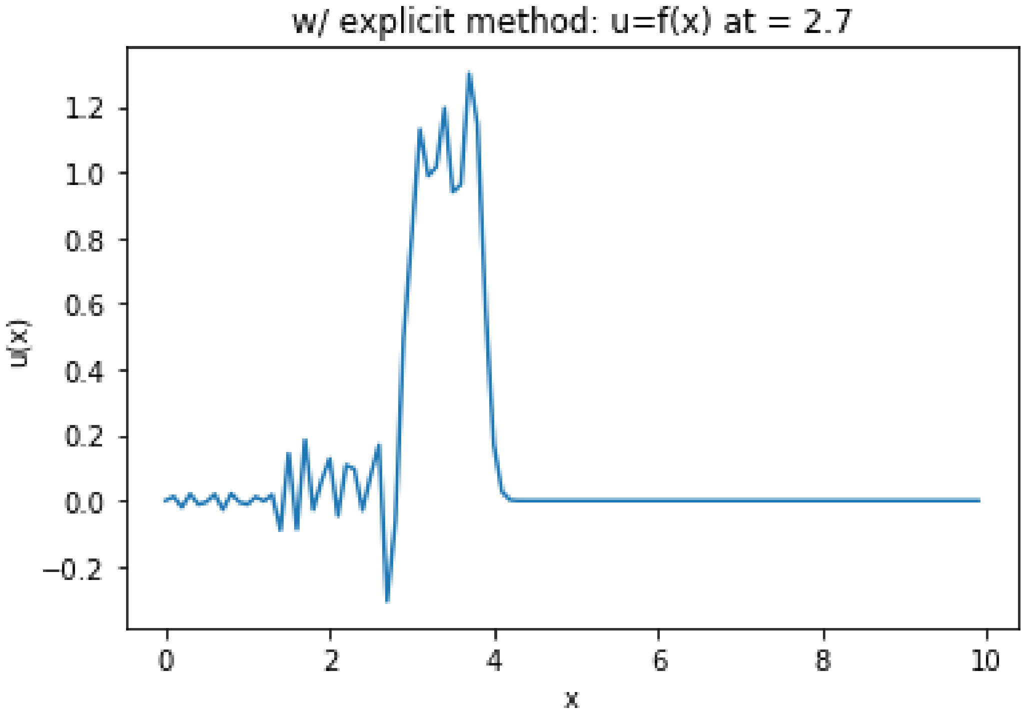
[7]:
```
dx = L/Nx
n_range = range(20, 60+1, 10)
Nt = n_range[-1]+1
```

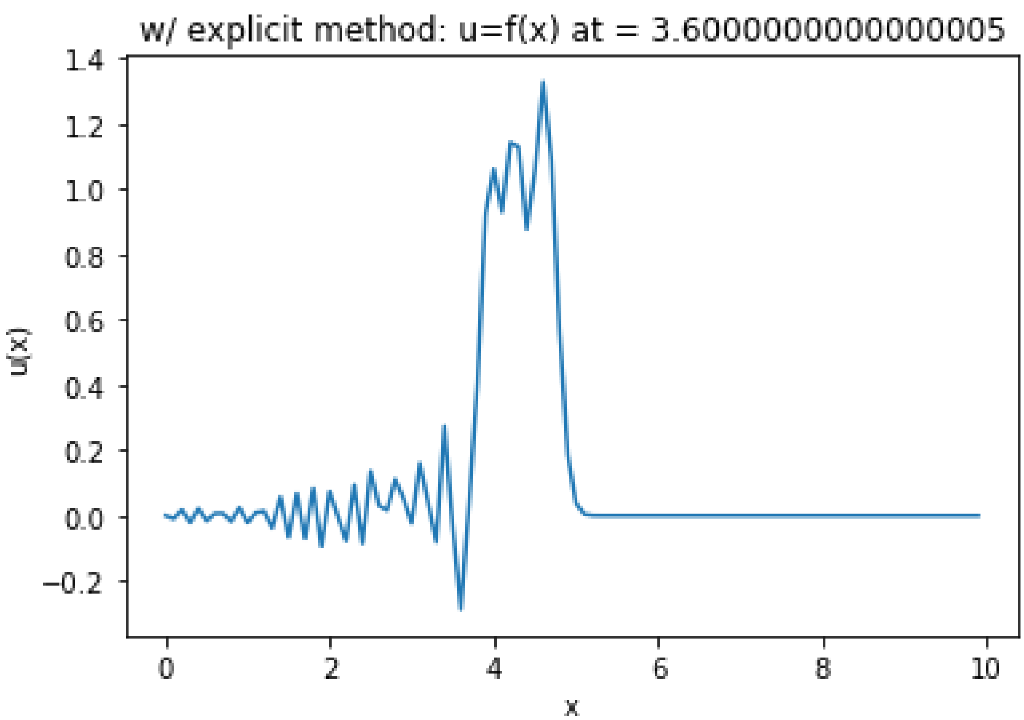
[8]:
```
def plot_wrapper(n_range, u0, um1, c, dt, dx, Nt, Nx):
    x = [i*dx for i in range(Nx)]

    u = weq_es(u0, um1, c, dt, dx, Nt, Nx)
    for n in n_range:
        plt.figure()
        plt.plot(x, u[n,:])
        plt.title("w/ explicit method: u=f(x) at = {}".format(n*dt))
        plt.xlabel('x')
        plt.ylabel('u(x)')
        plt.show()

    u = weq_is(u0, um1, c, dt, dx, Nt, Nx)
    for n in n_range:
        plt.figure()
        plt.plot(x, u[n, :])
        plt.title("w/ implicit method: u=f(x) at t = {}".format(n*dt))
        plt.xlabel('x')
        plt.ylabel('u(x)')
        plt.show()
```

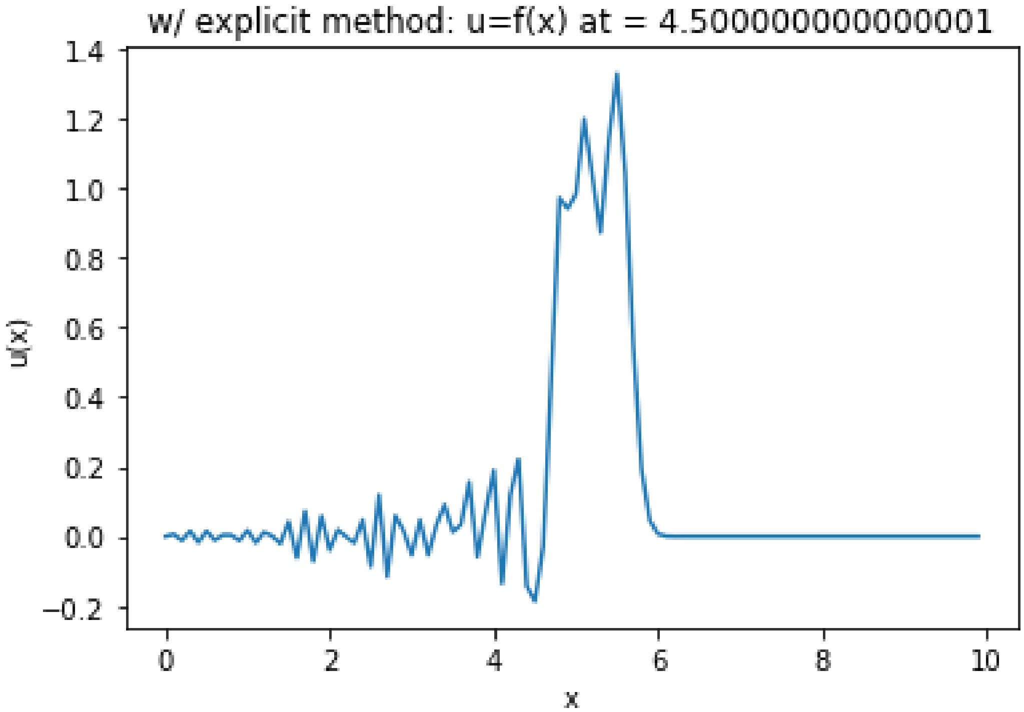
For $\Delta t = 0.9\Delta x/c$

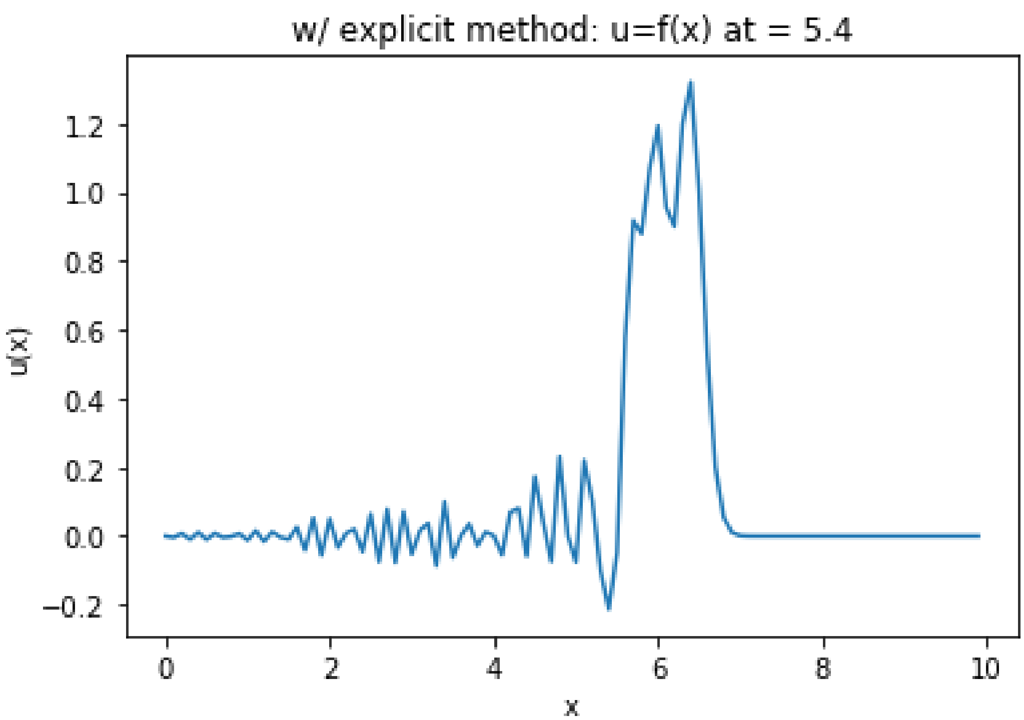
[9]:
```
dt = 0.9*dx/c
plot_wrapper(n_range, u0, um1, c, dt, dx, Nt, Nx)
```

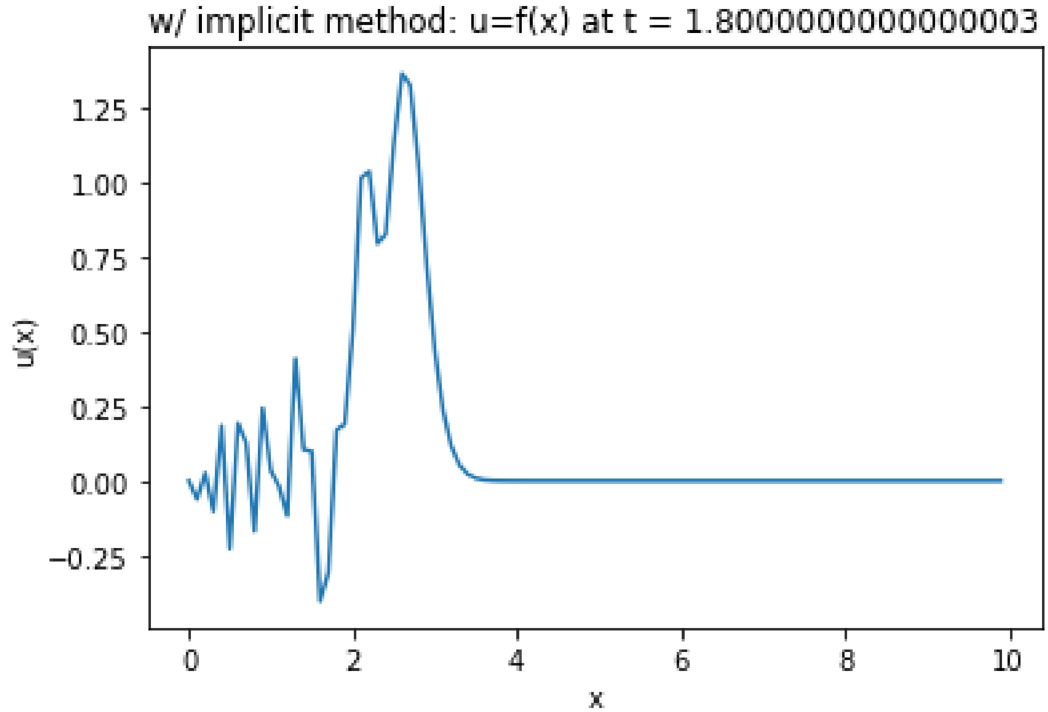


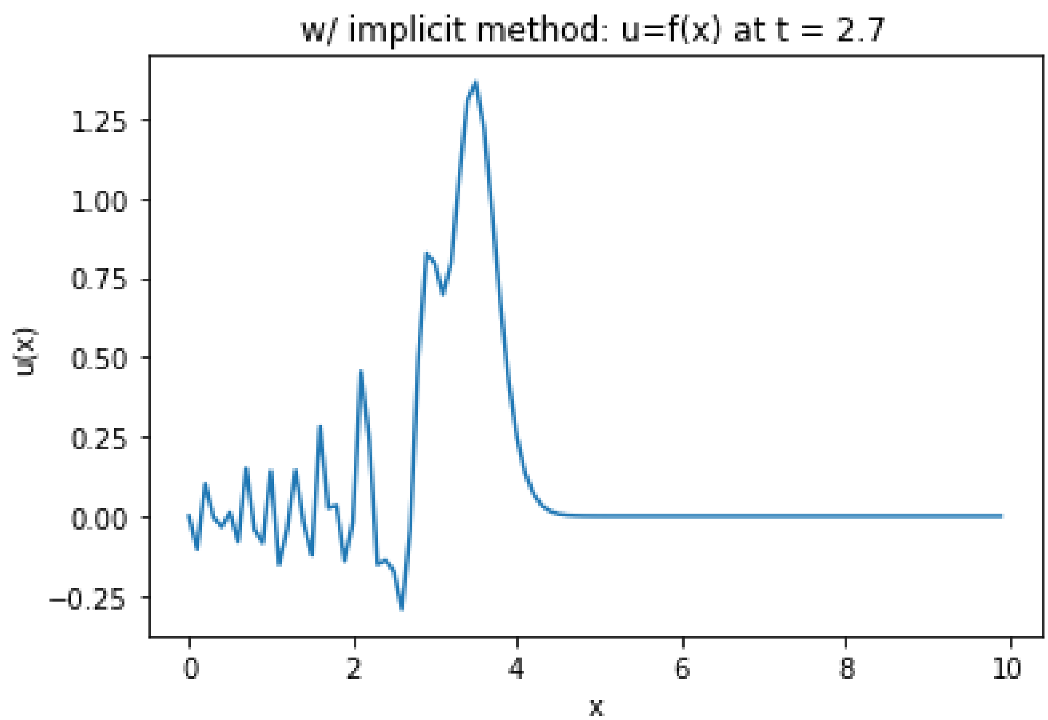

w/ explicit method: u=f(x) at = 2.7
u(x)
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2
4
6
8
10

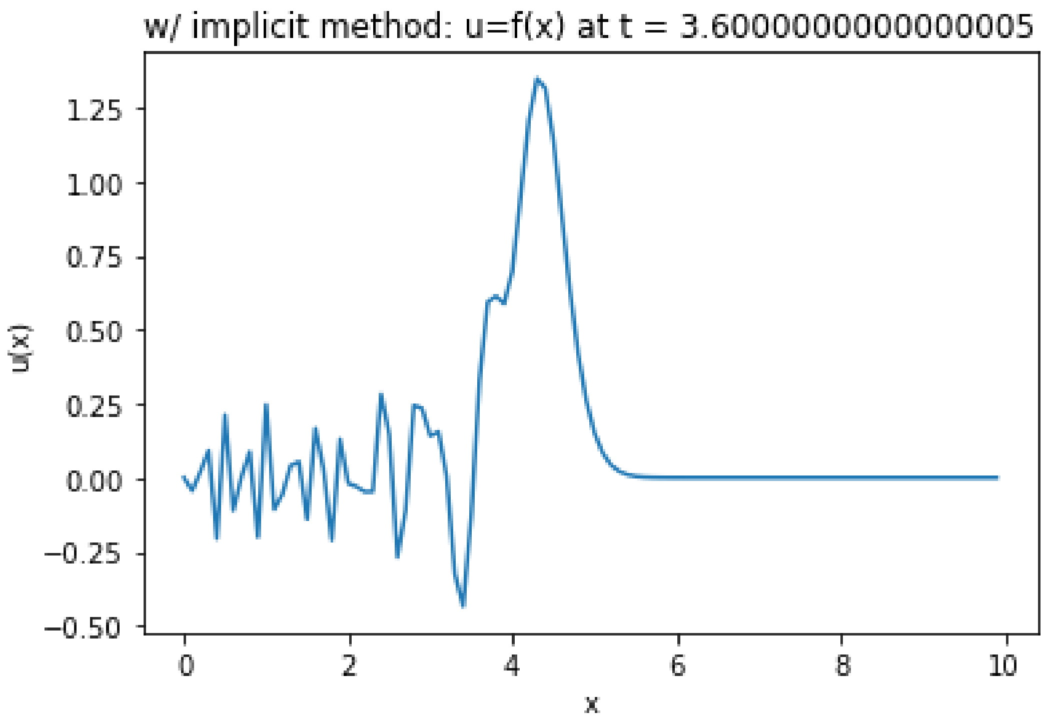

w/ explicit method: u=f(x) at = 3.6000000000000005
u(x)
x
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2
4
6
8
10

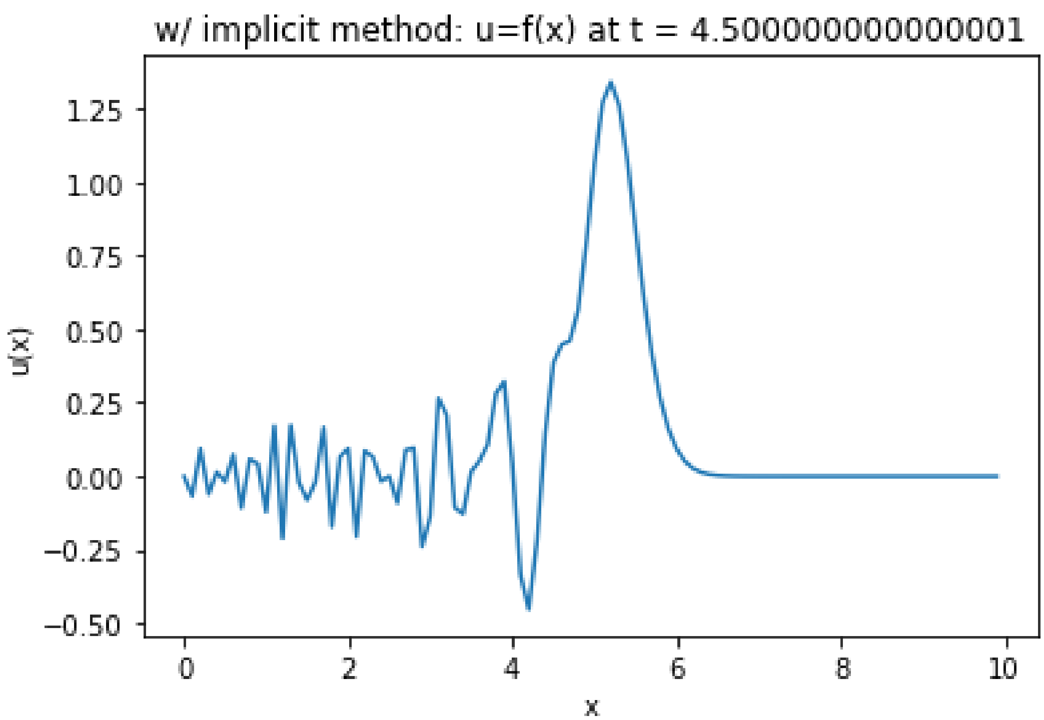

w/ explicit method: u=f(x) at = 4.500000000000001
u(x)
x
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2
4
6
8
10

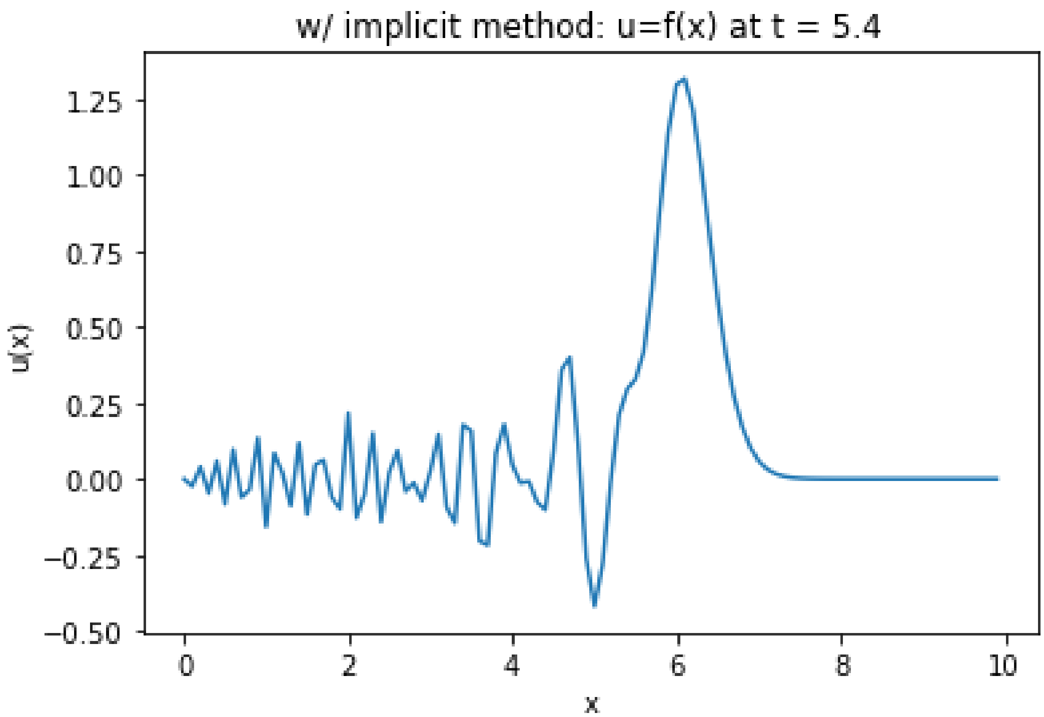

w/ explicit method: u=f(x) at = 5.4
u(x)
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2
4
6
8
10


w/ implicit method: u=f(x) at t = 1.8000000000000003
u(x)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0
2
4
6
8
10
x


w/ implicit method: u=f(x) at t = 2.7
u(x)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0
2
4
6
8
10
x

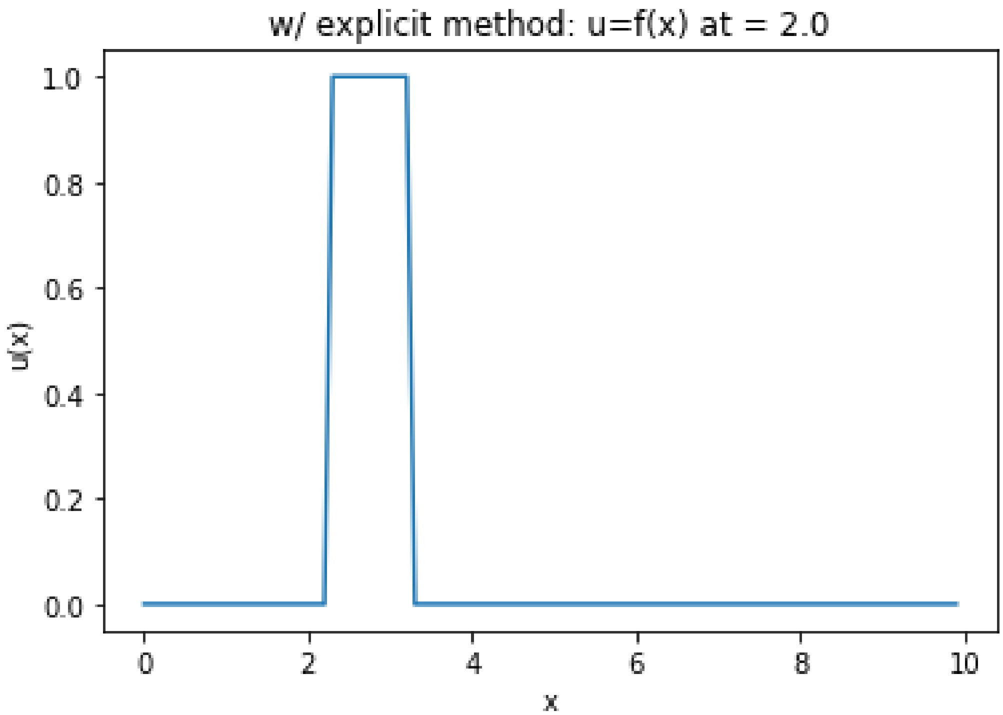

w/ implicit method: u=f(x) at t = 3.6000000000000005
u(x)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
2
4
6
8
10
x

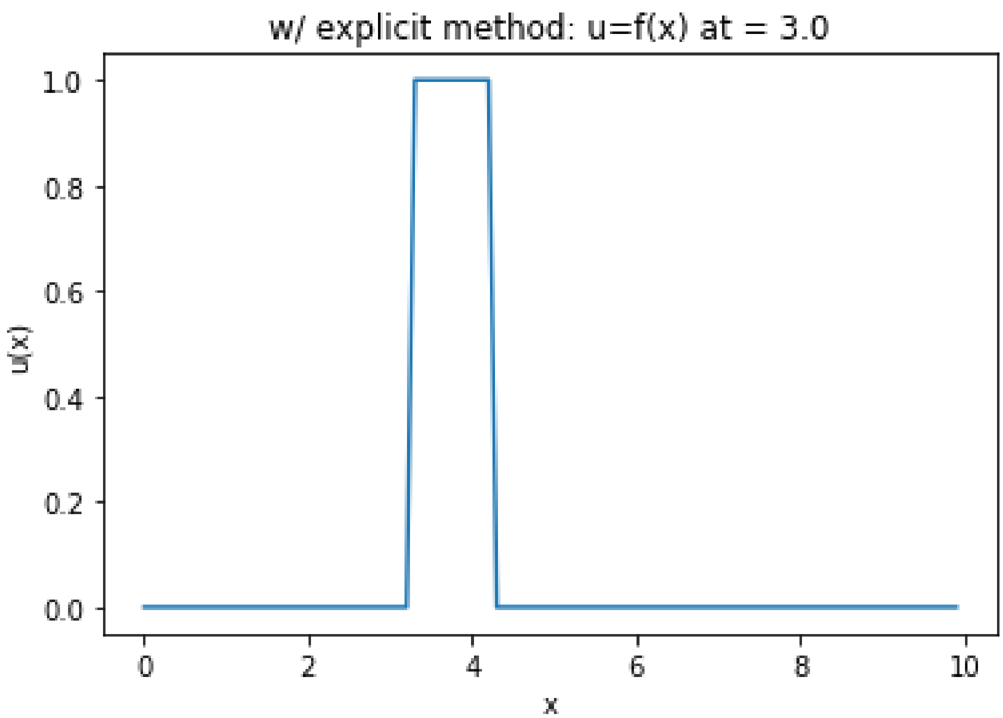

w/ implicit method: u=f(x) at t = 4.500000000000001
u(x)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
2
4
6
8
10
x

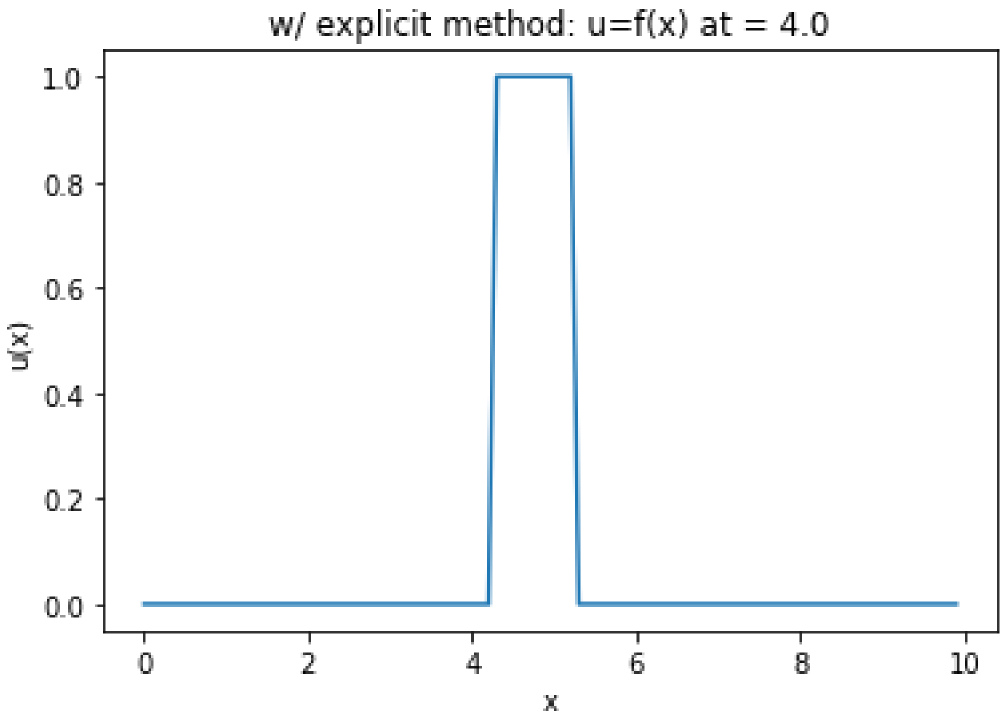


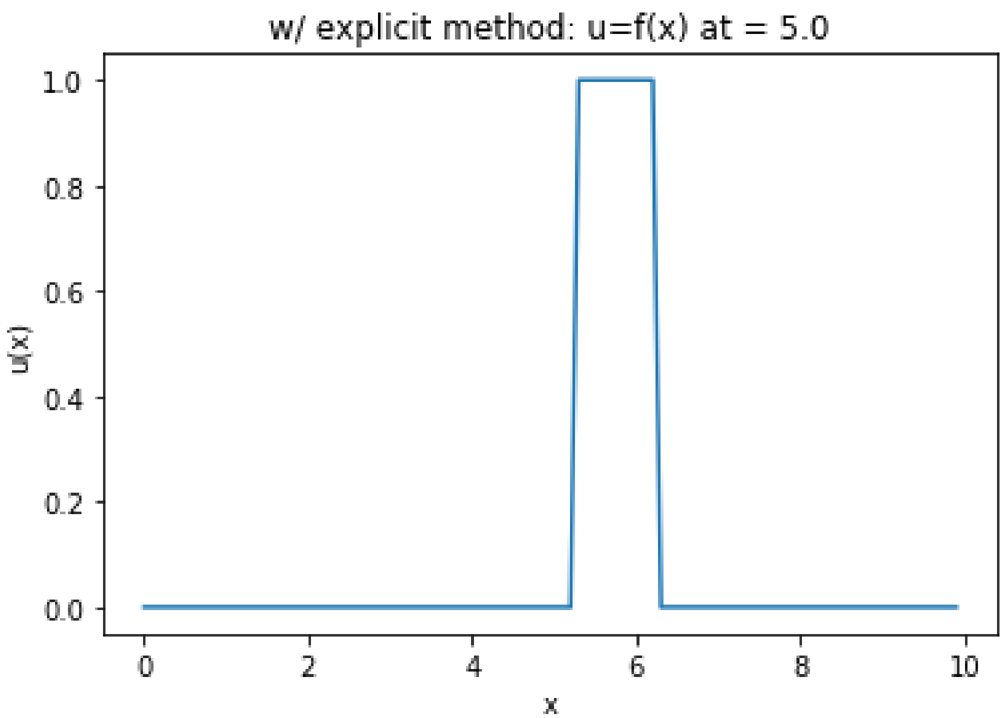
For $\Delta t = \Delta x/c$

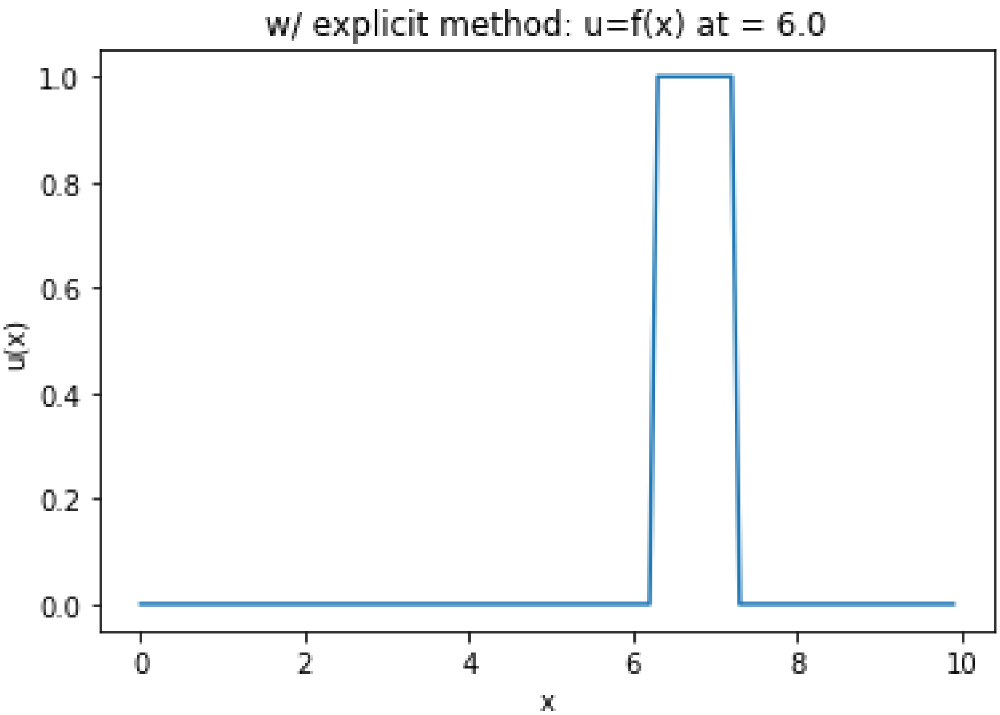
[10]:
```
dt = 1.0*dx/c
plot_wrapper(n_range, u0, um1, c, dt, dx, Nt, Nx)
```

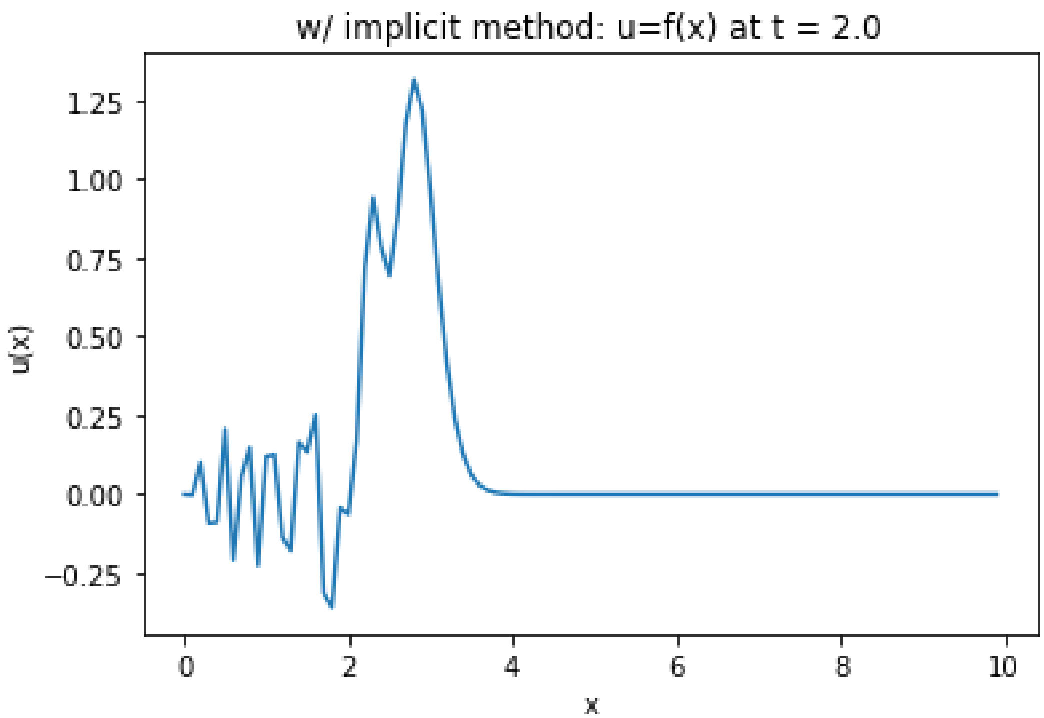

w/ explicit method: u=f(x) at = 2.0
1.0
0.8
0.6
0.4
0.2
0.0
u(x)
0
2
4
6
8
10
x

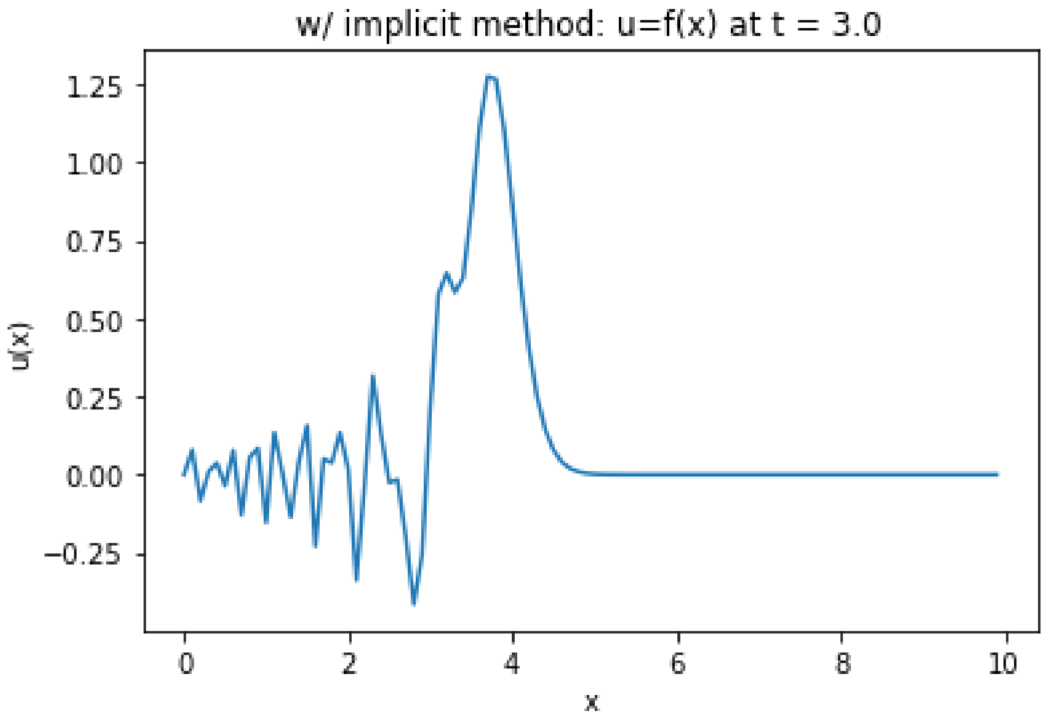

w/ explicit method: u=f(x) at = 3.0
1.0
0.8
0.6
0.4
0.2
0.0
u(x)
0
2
4
6
8
10
x

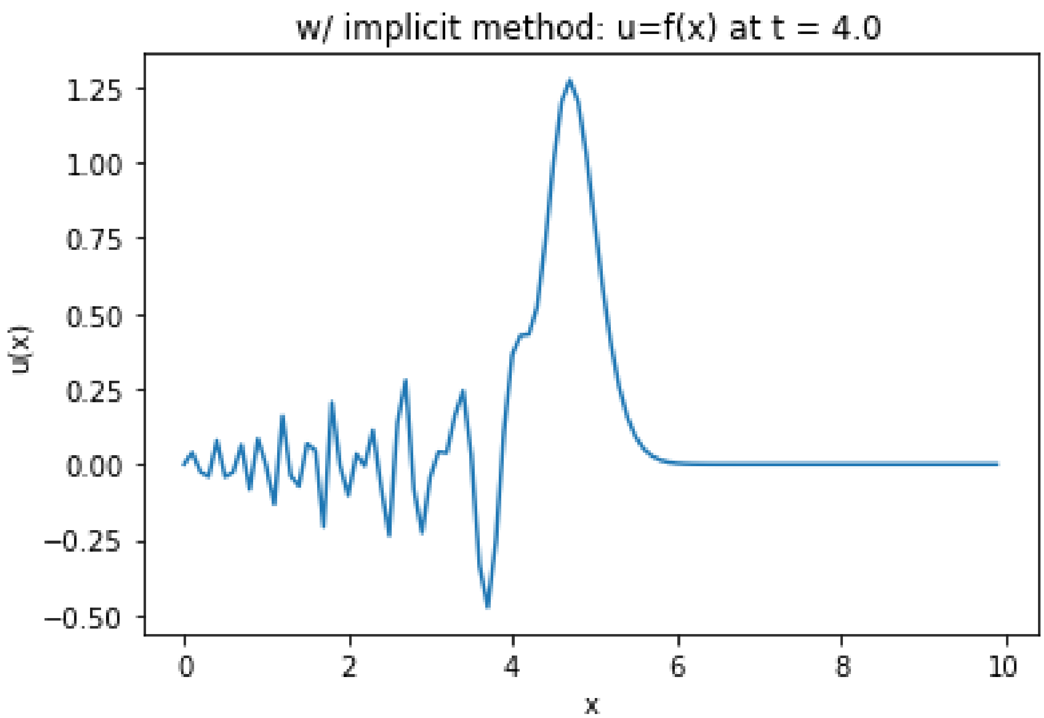

w/ explicit method: u=f(x) at = 4.0
u(x)
x
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10

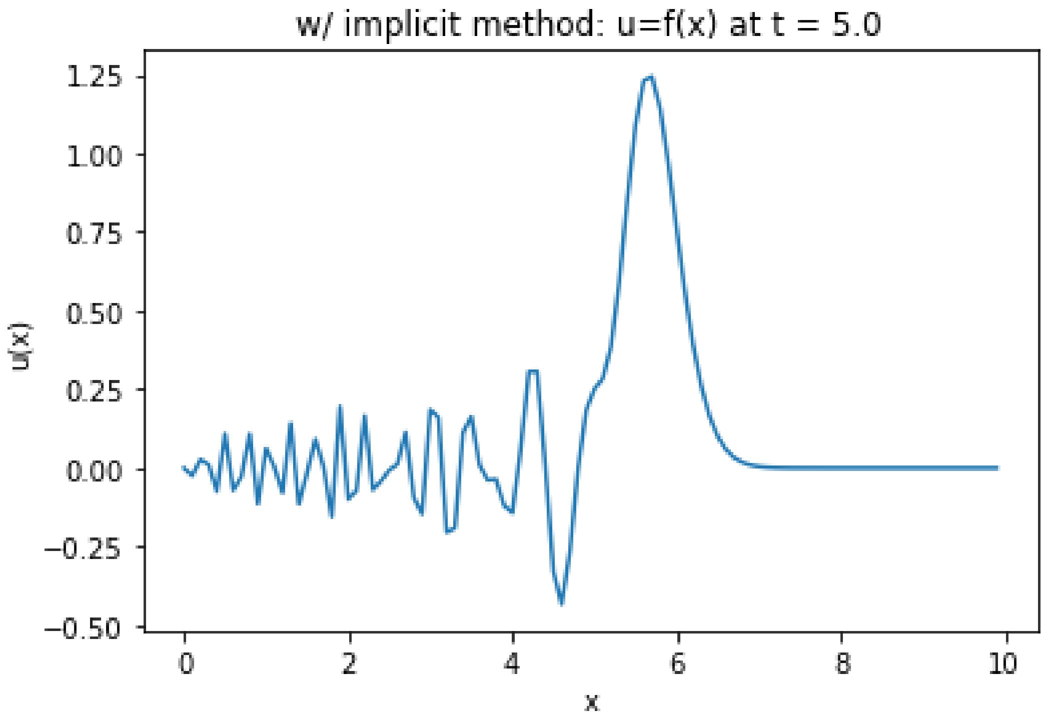

w/ explicit method: u=f(x) at = 5.0
u(x)
x
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10


w/ explicit method: u=f(x) at = 6.0
1.0
0.8
0.6
0.4
0.2
0.0
u(x)
0
2
4
6
8
10
x

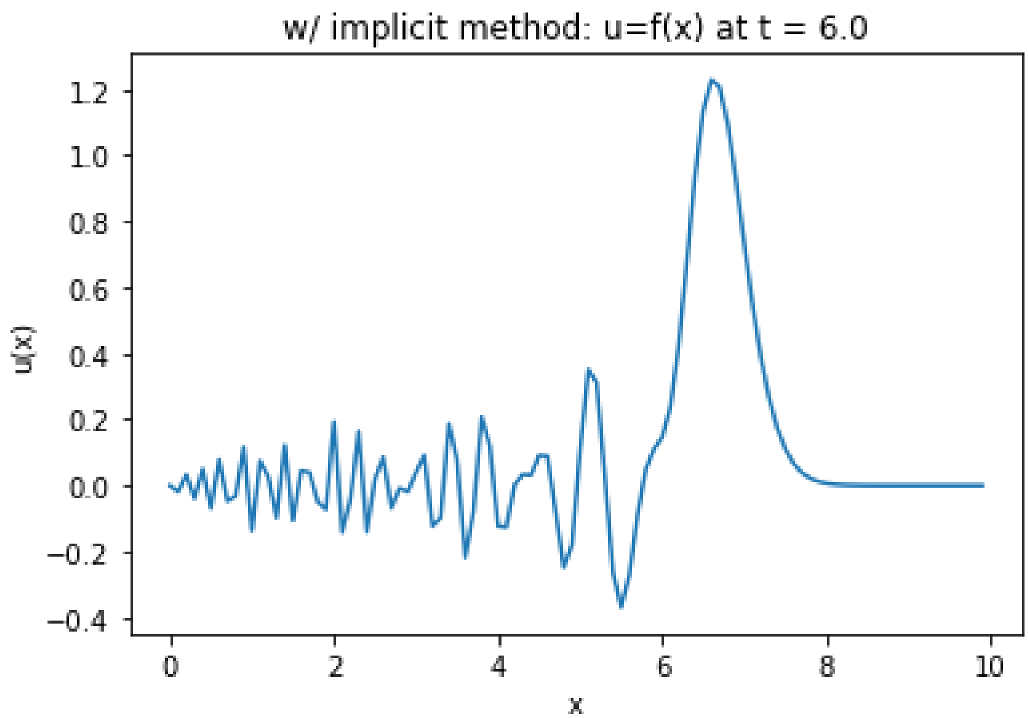

w/ implicit method: u=f(x) at t = 2.0
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
u(x)
0
2
4
6
8
10
x


w/ implicit method: u=f(x) at t = 3.0
u(x)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
0
2
4
6
8
10
x


w/ implicit method: u=f(x) at t = 4.0
u(x)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
2
4
6
8
10
x



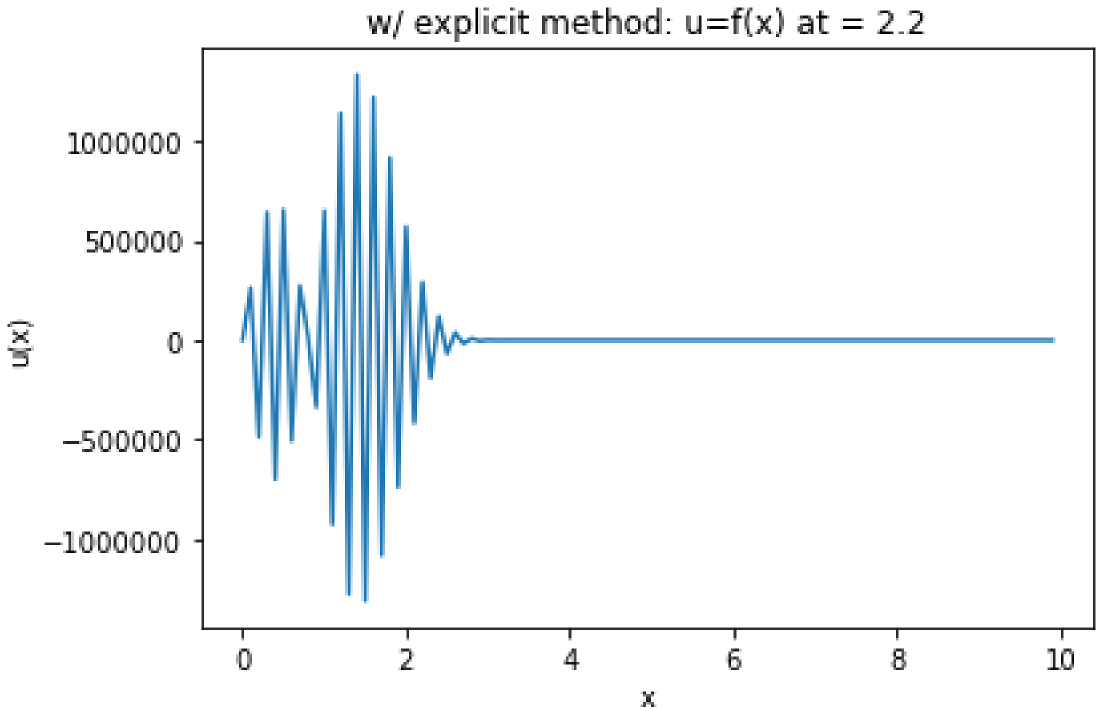
w/ implicit method: u=f(x) at t = 5.0
u(x)
x

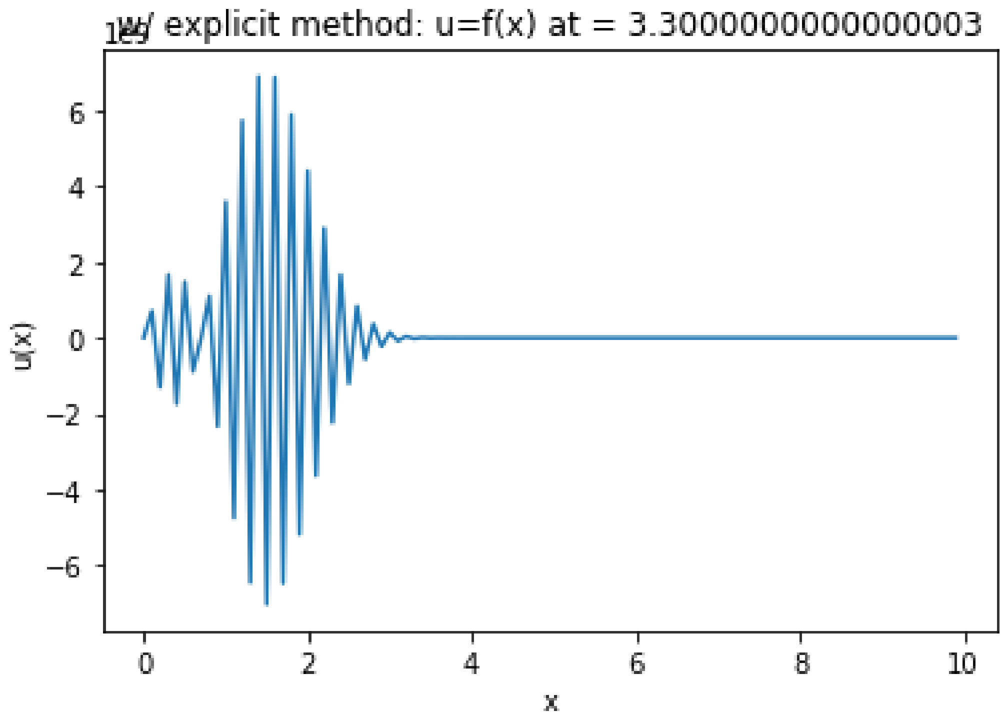


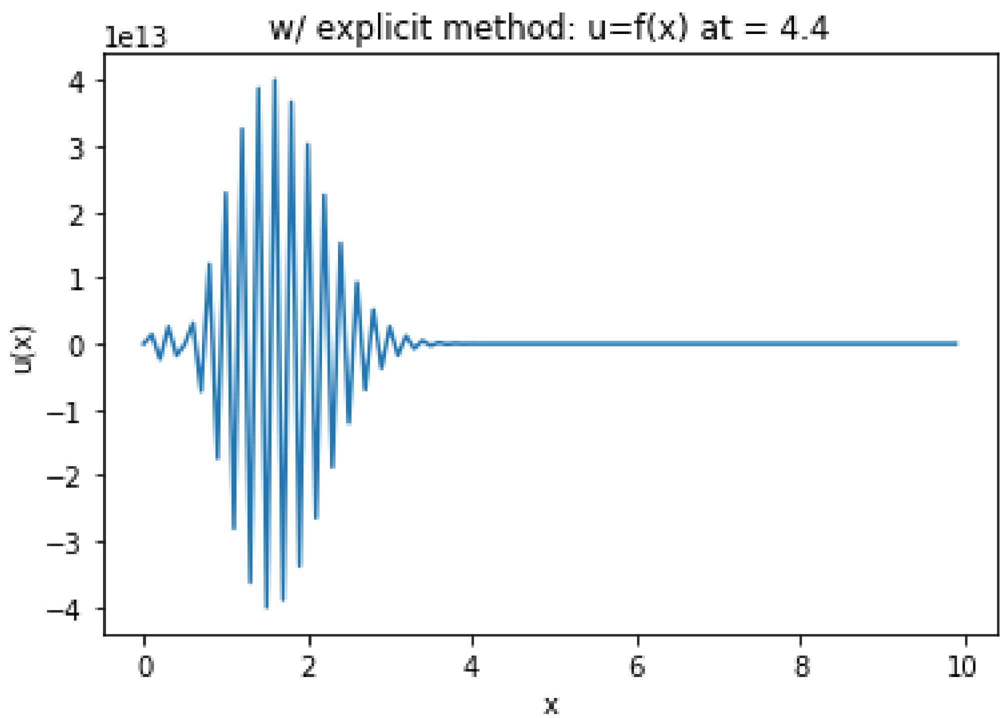
w/ implicit method: u=f(x) at t = 6.0
u(x)
x

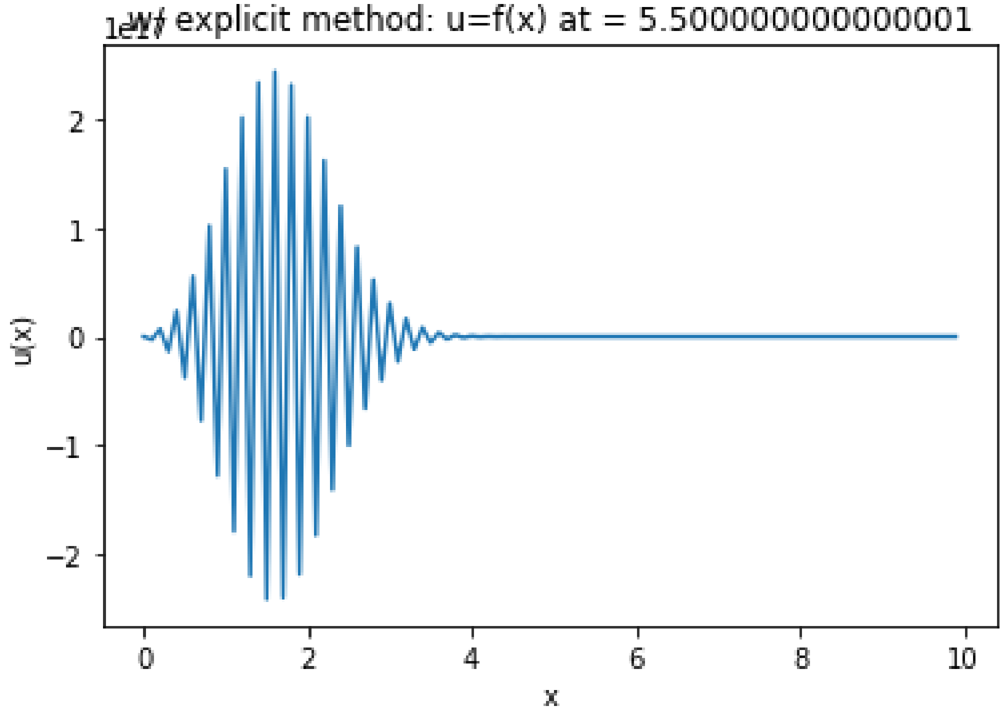
For $\Delta t = 1.1\Delta x/c$

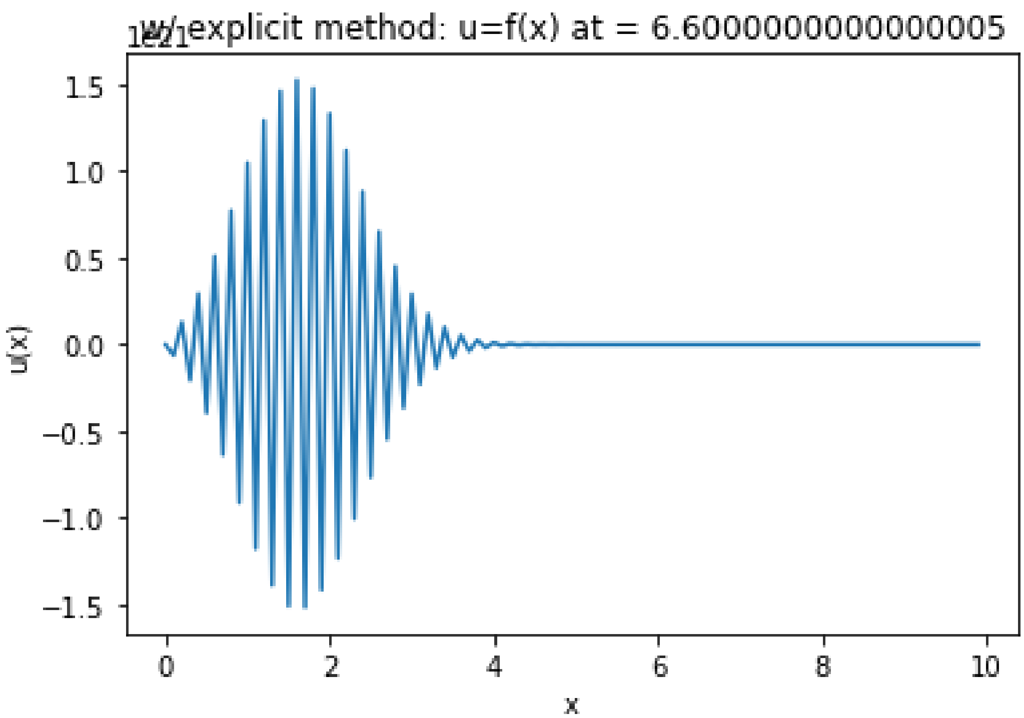
[11]:
```
dt = 1.1*dx/c
plot_wrapper(n_range, u0, um1, c, dt, dx, Nt, Nx)
```

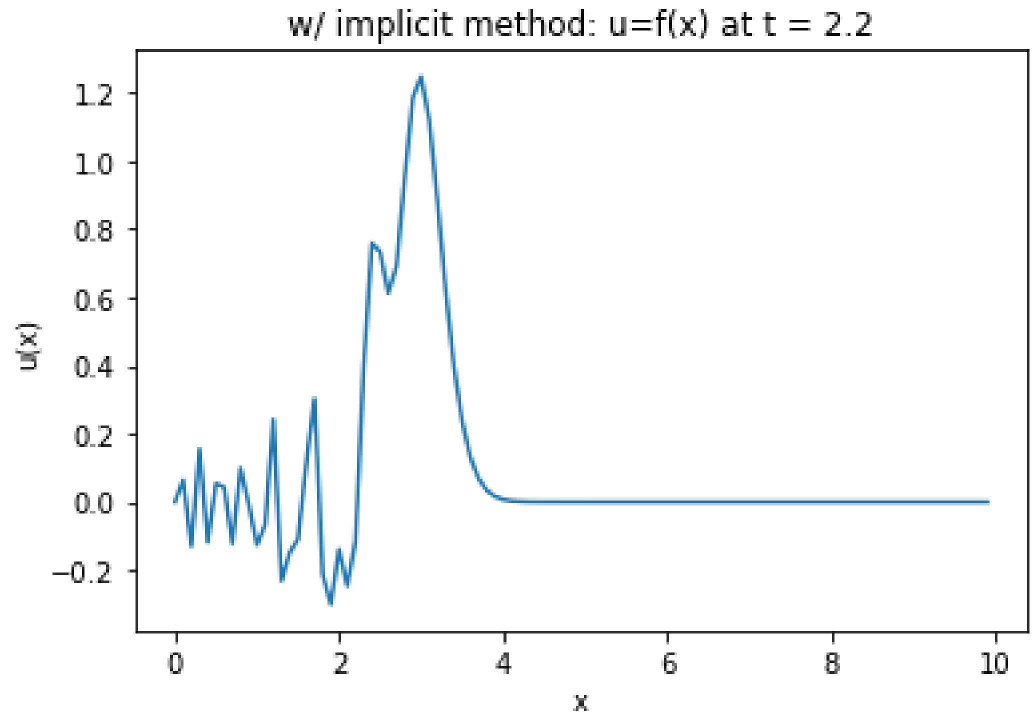


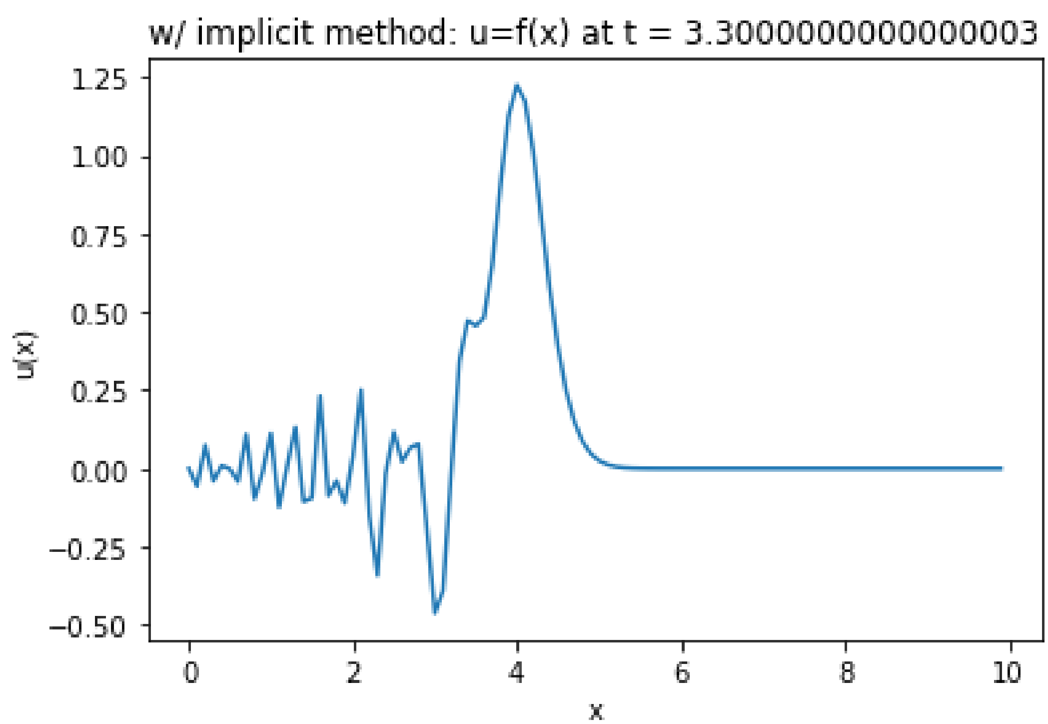

w/ explicit method: u=f(x) at = 3.3000000000000003
1e9
u(x)
6
4
2
0
−2
−4
−6
0
2
4
6
8
10
x

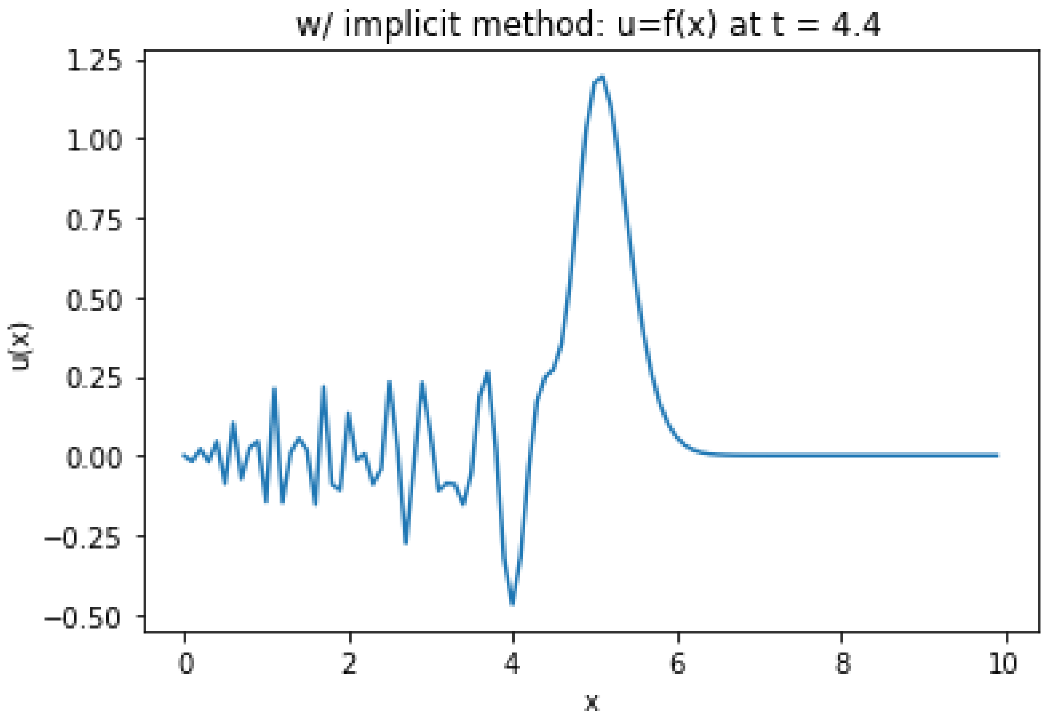

w/ explicit method: u=f(x) at = 4.4
1e13
u(x)
4
3
2
1
0
−1
−2
−3
−4
0
2
4
6
8
10
x

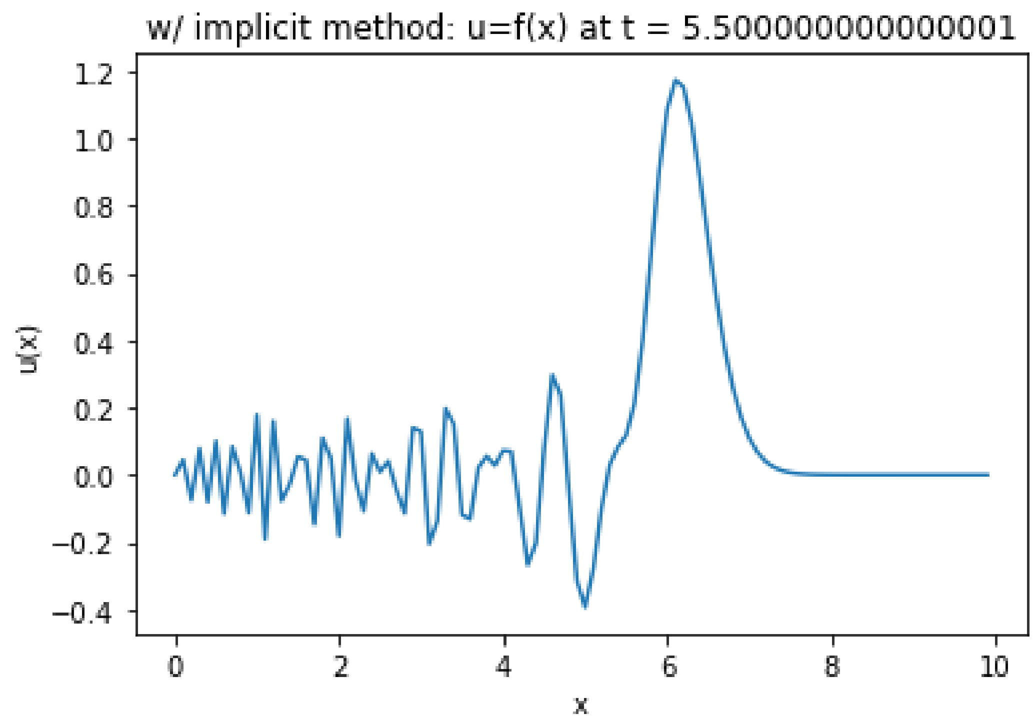

w/ explicit method: u=f(x) at = 5.500000000000001
1e17
2
1
0
−1
−2
u(x)
0
2
4
6
8
10
x

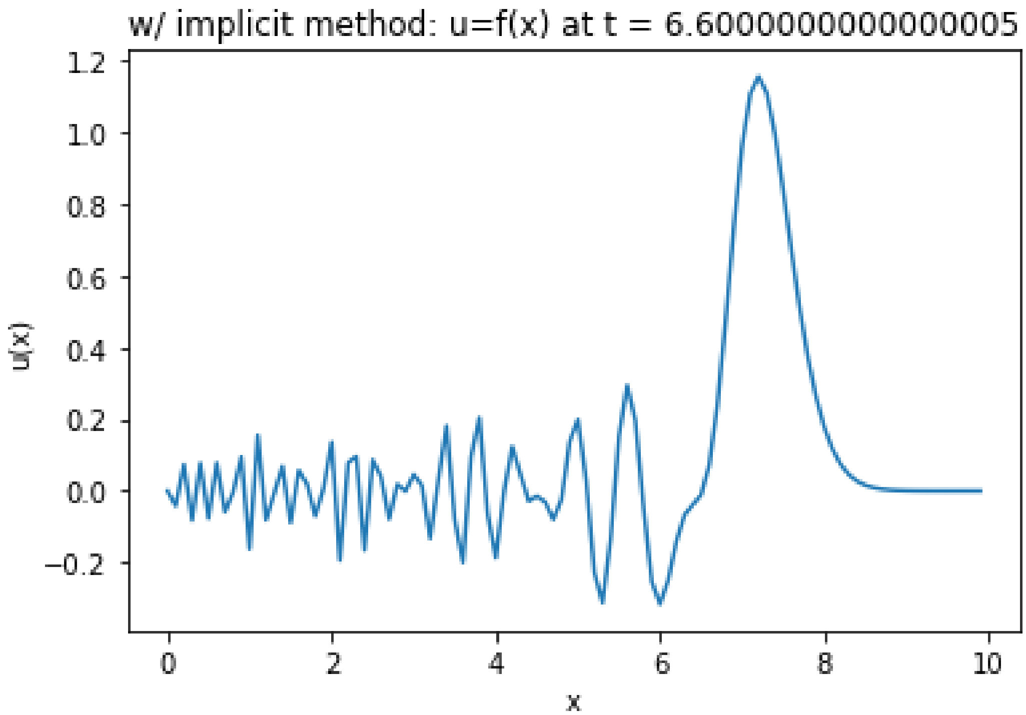

w/ explicit method: u=f(x) at = 6.6000000000000005
1e21
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
u(x)
0
2
4
6
8
10
x


w/ implicit method: u=f(x) at t = 2.2
u(x)
x
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
2
4
6
8
10


w/ implicit method: u=f(x) at t = 3.3000000000000003
u(x)
x
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
2
4
6
8
10


w/ implicit method: u=f(x) at t = 4.4
u(x)
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
2
4
6
8
10
x


w/ implicit method: u=f(x) at t = 5.500000000000001
u(x)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
2
4
6
8
10
x



- Comments

We tried two 2nd-order methods, an explicit and an implicit, for numerically solving the one-dimensional wave equation, using a time step $dt = a \cdot dx/c$ for various values of a. As the initial conditions represent a rectangular pulse, the integration should result in a moving pulse.

For the explicit scheme, in the first case where $a < 1$ we're seeing ringing due to numerical dispersion. In the second case where $a = 1$ we get an exact rectangular pulse. Dispersion has disappeared as we've taken a time step $= dx/c$, the so-called magic time step.

This magic time step doesn't apply to the implicit scheme. Instead the way to improve the results is taking $dx, dt \to 0$ (utilizing fine sampling).

In the third and final case where $a > 1$, we notice that the explicit scheme becomes unstable, whereas the implicit scheme continues working. Theoretically this was expected as the stability condition for the explicit scheme is $a = c \cdot dt/dx \leq 1$. In contrast the implicit scheme is absolute stable.

Bibliography

- [GedneyEE624] Gedney Stephen, University of Kentucky, EE624 Notes, Fall 2005, "1D Solution of the Wave Equation".