104C_Final

June 1, 2022

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

Question 3a.

Here our finite difference scheme can be represented by,

$$\frac{p_j^{n+1} - 2p_j^n + p_j^{n-1}}{(\Delta t)^2} = a^2 \frac{p_{j+1}^n - 2p_j^n + p_{j-1}^n}{(\Delta x)^2}$$

To initialize this multistep scheme we can use $p_j^0 = p(0, x)$ and to obtain p_j^1 we can use our second initial condition, $p_t(0, x)$, as

$$p_t(0, x_j) \approx \frac{p_j^1 - p_j^0}{\Delta t}$$

Since we are given $p_t(0, x_j) = 0$, we have $p_j^1 = p_j^0$

We can now conduct Von Nuemann analysis to look at stability. Substituting $p_j^n = \xi^n e^{ikj\Delta x}$ and cancelling the common terms we find that

$$\xi - 2 + \frac{1}{\xi} = -4\lambda^2 \sin^2 \frac{1}{2}\theta$$

setting $\lambda = a\Delta t/\Delta x$ and $\theta = k\Delta x$, we can rearrange our expression

$$\left(\sqrt{\xi} - \frac{1}{\sqrt{\xi}}\right)^2 = \left(\pm 2i\lambda \sin \frac{1}{2}\theta\right)^2$$

thus,

$$\sqrt{\xi} - \frac{1}{\sqrt{\xi}} = \pm 2i\lambda \sin \frac{1}{2}\theta$$

Multiplying by $\sqrt{\xi}$ we get

$$\xi \pm 2i\sqrt{\xi}\lambda \sin \frac{1}{2}\theta - 1 = 0.$$

Here we have a quadratic equation for $\sqrt{\xi}$ with roots

$$\xi_{\pm}^{1/2} = \pm i\lambda \sin \frac{1}{2}\theta \pm \sqrt{1 - \lambda^2 \sin^2 \frac{1}{2}\theta}$$

$$= \xi_{\pm} = \left(\sqrt{1 - \lambda^2 \sin^2 \frac{1}{2}\theta} \pm i\lambda \sin \frac{1}{2}\theta\right)^2$$

Thus, $|\xi_\pm| \leq 1$ if and only if $|\lambda| \leq 1$. Also $\xi_+ = \xi_-$for $\theta = 0$ or if $|\lambda| = 1$ and $\theta = \pi$. With equal roots, $n\xi_+^{n-1} e^{ikj\Delta x}$ is also a solution of the scheme. Since the wave equation is second order in time, it allows linearly growing solutions like Ct so the mode $n\xi_+^{n-1} e^{ikj\Delta x}$ with $|\xi_+| = 1$ is a solution. We can now conclude that our scheme is stable $\iff$ $|\lambda| \leq 1$

Since we took $\lambda = a\Delta t/\Delta x$ and are given $a = 1$, we see that our scheme is stable for $\Delta t/\Delta x \leq 1$.

Our scheme is second order, both in space and time which is consistent with the wave equation we are given.

Looking at the truncation error

$$\tau_j^{n+1}(\Delta t, \Delta x) = \frac{p(t_{n+1}, x_j) - 2p(t_n, x_j) + p(t_{n-1}, x_j)}{(\Delta t)^2} - a^2 \frac{p(t_n, x_{j+1}) - 2p(t_n, x_j) + p(t_n, x_{j-1})}{(\Delta x)^2}$$

Taylor expanding around $p(t_n, x_j)$,

$$p(t_{n+1}, x_j) = p(t_n, x_j) + \Delta t p_t + \frac{\Delta t^2}{2} p_{tt} + \frac{\Delta t^3}{6} p_{ttt} + \mathcal{O}(\Delta t^4)$$

$$p(t_{n-1}, x_j) = p(t_n, x_j) - \Delta t p_t + \frac{\Delta t^2}{2} p_{tt} - \frac{\Delta t^3}{6} p_{ttt} + \mathcal{O}(\Delta t^4)$$

$$p(t_n, x_{j+1}) = p(t_n, x_j) + \Delta x p_x + \frac{\Delta x^2}{2} p_{xx} + \frac{\Delta x^3}{6} p_{xxx} + \mathcal{O}(\Delta x^4)$$

$$p(t_n, x_{j-1}) = p(t_n, x_j) - \Delta x p_x + \frac{\Delta x^2}{2} p_{xx} - \frac{\Delta x^3}{6} p_{xxx} + \mathcal{O}(\Delta x^4)$$

We are given that $a = 1$. Thus,

$$\begin{aligned}\tau_j^{n+1}(\Delta t, \Delta x) &= \frac{\Delta t^2 p_{tt} + \mathcal{O}(\Delta t^4)}{\Delta t^2} - \frac{\Delta x^2 p_{xx} + \mathcal{O}(\Delta x^4)}{\Delta x^2} \\ &= p_{tt} + \mathcal{O}(\Delta t^2) - p_{xx} + \mathcal{O}(\Delta x^2) \\ &= \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2)\end{aligned}$$

Taking the limit we see

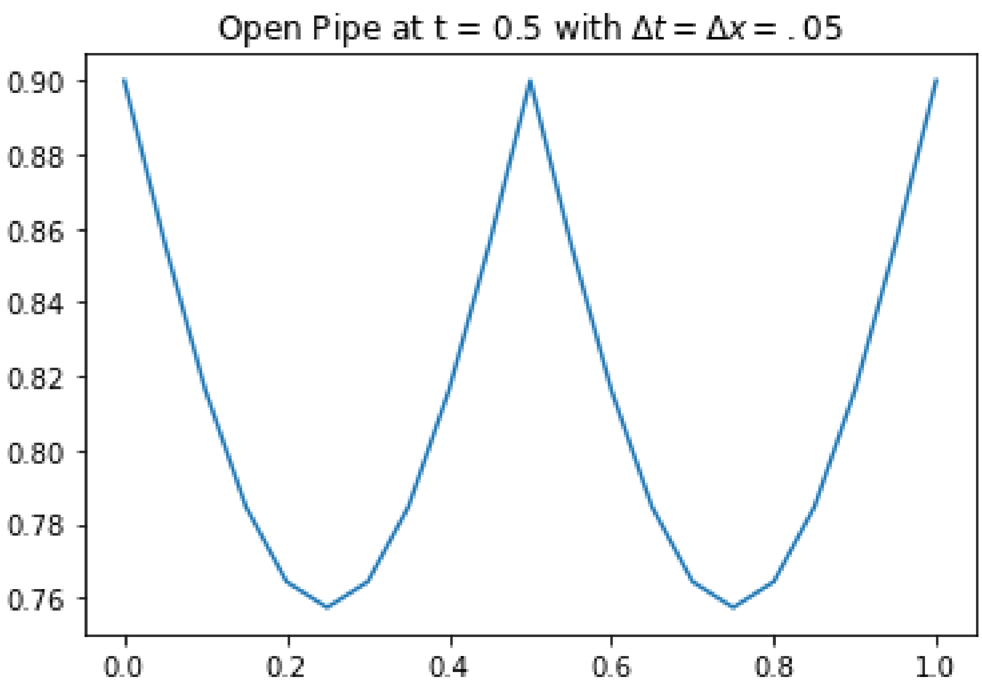
$$\lim_{\Delta t, \Delta x \to 0} \tau_j^{n+1}(\Delta t, \Delta x) = \lim_{\Delta t, \Delta x \to 0} \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2) = 0$$

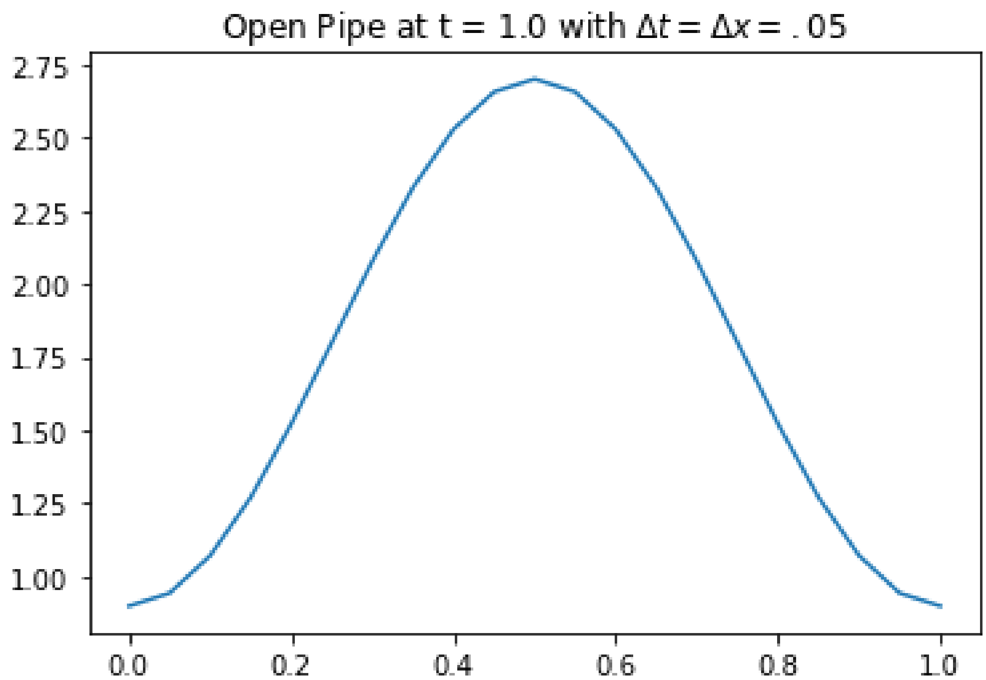
Question 3b.

[2]:
```
def open_pipe(p_init,delta_t,M):
    mat = np.empty([M+1,M+1])

    for i in range(M+1):
        mat[0][i] = p_init
        mat[M][i] = p_init

    for j in range(1,M):
        mat[j][0] = p_init * np.cos(2*np.pi*(delta_t *(j)))
```

```
        mat[j][1] = mat[j][0]

    for k in range(2,M+1):
        for j in range(1,M):
            mat[j][k] = mat[j-1][k-1]+mat[j+1][k-1]-mat[j][k-2]

    return mat
```

[3]:
```
M = 20
p_init = .9
delta_t = .05
x_values = np.linspace(0,1,21)
p_approx = open_pipe(p_init,delta_t, M)
plt.plot(x_values,p_approx[:,int(M/2)])
plt.title('Open Pipe at t = 0.5 with $\Delta t = \Delta x = .05$')
plt.show()

M = 20
p_init = .9
delta_t = .05
x_values = np.linspace(0,1,21)
p_approx = open_pipe(p_init,delta_t,M)
plt.plot(x_values,p_approx[:,int(M)])
plt.title('Open Pipe at t = 1.0 with $\Delta t = \Delta x = .05$')
plt.show()
```



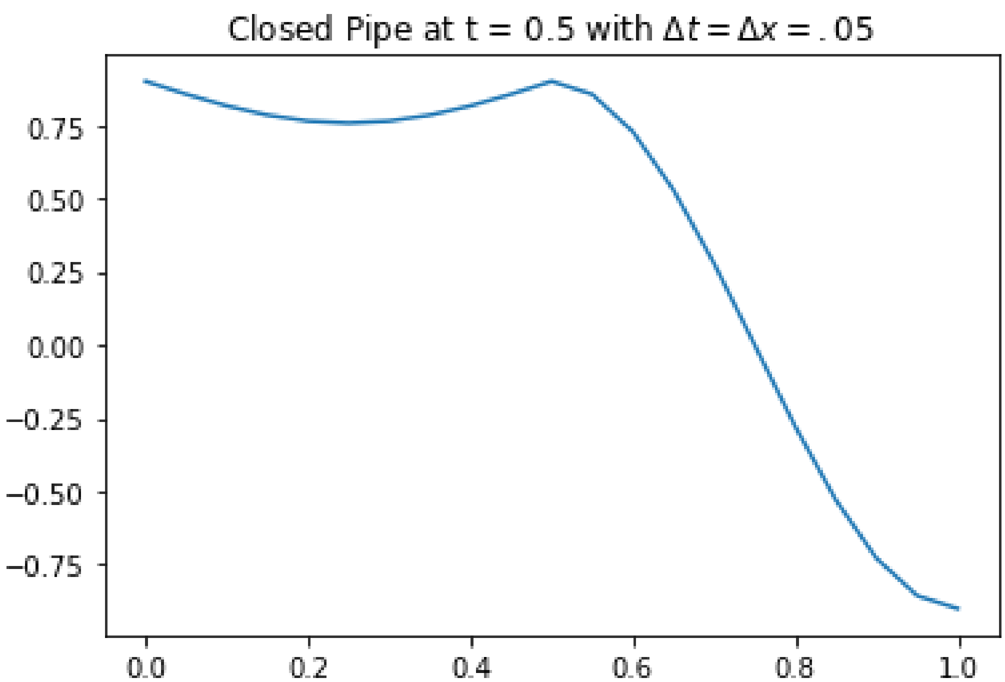


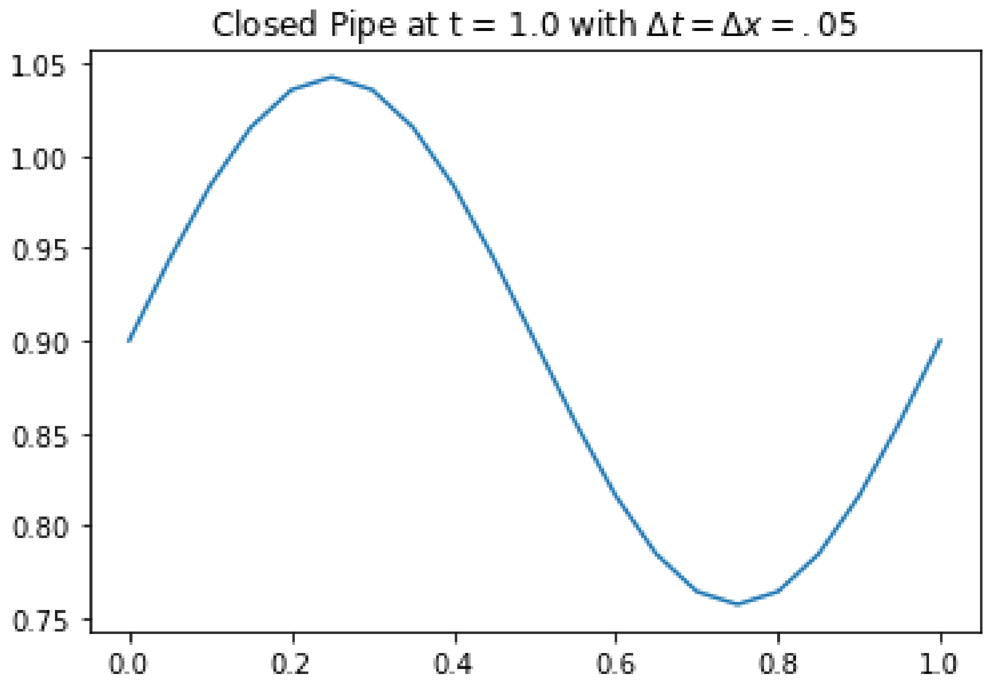
Question 3c.

[4]:
```
def closed_pipe(p_init,delta_t,M):
    mat2 = np.empty([M+1,M+1])

    for i in range(M+1):
        mat2[0][i] = p_init

    for j in range(1,M+1):
        mat2[j][0] = p_init * np.cos(2*np.pi*(delta_t*(j)))
        mat2[j][1] = mat2[j][0]

    mat2[M][0] = p_init
    mat2[M][1] = p_init

    for k in range(2,M+1):
        for j in range(1,M+1):
            if (j >= M):
                mat2[j][k] = 2 * mat2[j-1][k-1] - mat2[j][k-2]
            else:
                mat2[j][k] = mat2[j-1][k-1] + mat2[j+1][k-1]-mat2[j][k-2]

    return mat2
```

[5]:

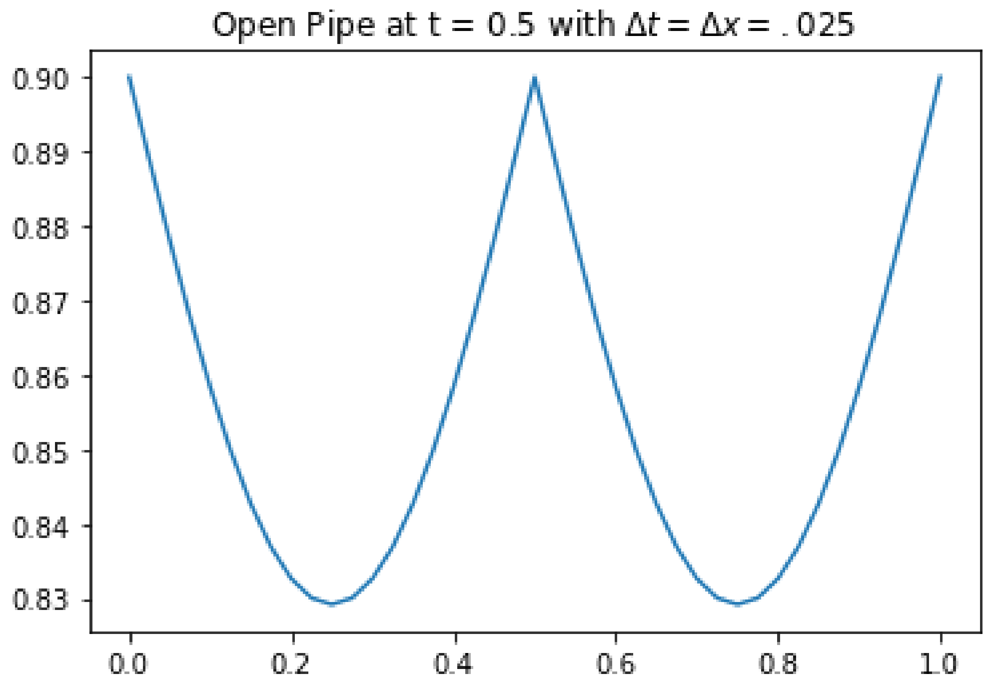
```
p_init = .9
delta_t = .05
M = int(1/delta_t)
x_values = np.linspace(0,1,M+1)
p_approx = closed_pipe(p_init,delta_t,M)
plt.plot(x_values,p_approx[:,int(M/2)])
plt.title('Closed Pipe at t = 0.5 with $\Delta t = \Delta x = .05$')
plt.show()

p_init = .9
delta_t = .05
M = int(1/delta_t)
x_values = np.linspace(0,1,M+1)
p_approx = closed_pipe(p_init,delta_t,M)
plt.plot(x_values,p_approx[:,int(M)])
plt.title('Closed Pipe at t = 1.0 with $\Delta t = \Delta x = .05$')
plt.show()
```

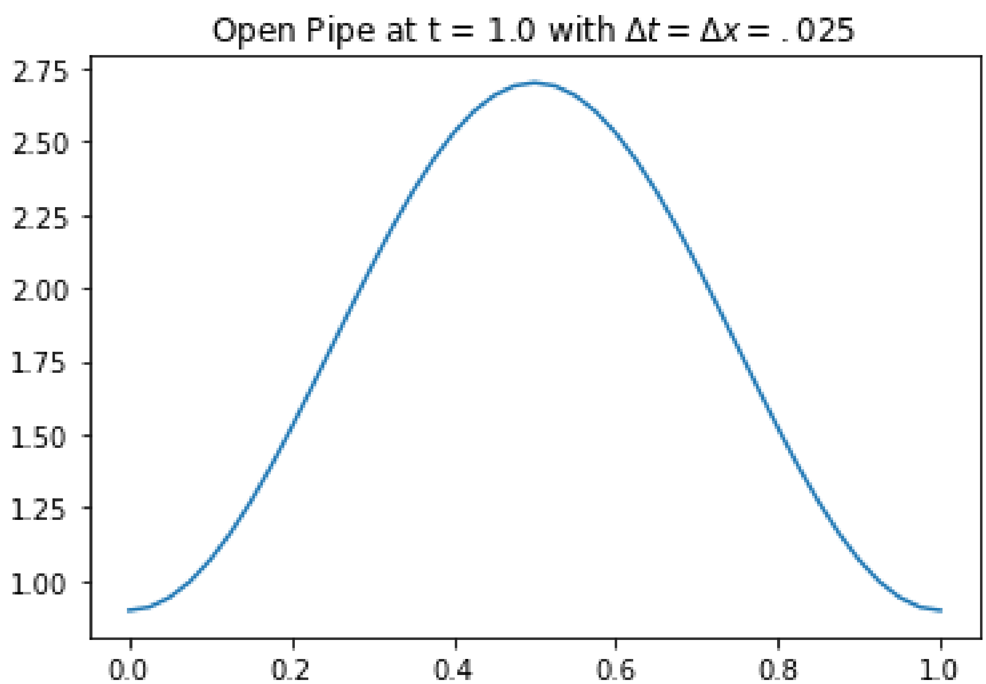




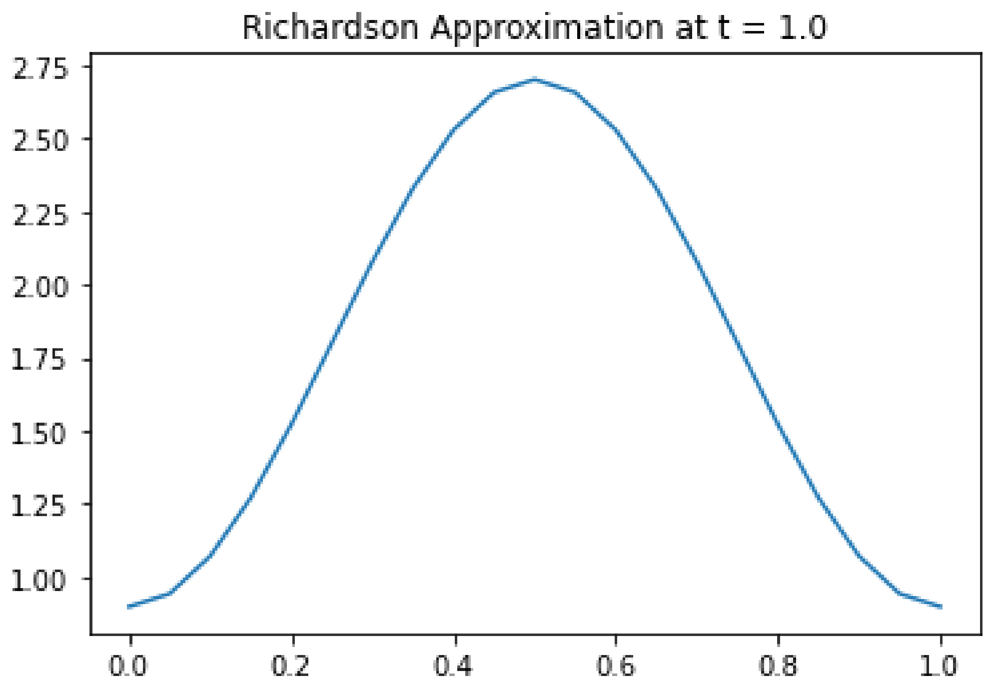
Question 3d.

[6]:
```
## Delta_t = Delta_x = .025

p_init = .9
delta_t = .025
M = int(1/delta_t)
x_values = np.linspace(0,1,M+1)
p_approx = open_pipe(p_init,delta_t,M)
plt.plot(x_values,p_approx[:,int(M/2)])
plt.title('Open Pipe at t = 0.5 with $\Delta t = \Delta x = .025$')
plt.show()


p_init = .9
delta_t = .025
M = int(1/delta_t)
x_values = np.linspace(0,1,M+1)
p_approx = open_pipe(p_init,delta_t,M)
plt.plot(x_values,p_approx[:,int(M)])
plt.title('Open Pipe at t = 1.0 with $\Delta t = \Delta x = .025$')
plt.show()
```


Open Pipe at t = 0.5 with $\Delta t = \Delta x = .025$
0.90
0.89
0.88
0.87
0.86
0.85
0.84
0.83
0.0
0.2
0.4
0.6
0.8
1.0


Open Pipe at t = 1.0 with $\Delta t = \Delta x = .025$
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.0
0.2
0.4
0.6
0.8
1.0

[7]:
```
## Richardsons Approximation

p_approx1 = open_pipe(.9 , .05 , 20)
p_approx2 = open_pipe(.9,.025, 40)

richardson = ((4*p_approx2[::2,40] - p_approx1[:,20])/3)
x_values = np.linspace(0,1,21)

plt.plot(x_values,richardson)
plt.title('Richardson Approximation at t = 1.0')
plt.show()
```



continuing our Taylor expansion from part a to get the order of the richardson approximation,

$$p(t_{n+1},x_j) = p(t_n,x_j) + \Delta t p_t + \frac{\Delta t^2}{2}p_{tt} + \frac{\Delta t^3}{6}p_{ttt} + \frac{\Delta t^4}{24}p_{tttt} + \frac{\Delta t^5}{120}p_{ttttt} + \mathcal{O}(\Delta t^6)$$

$$p(t_{n-1},x_j) = p(t_n,x_j) - \Delta t p_t + \frac{\Delta t^2}{2}p_{tt} - \frac{\Delta t^3}{6}p_{ttt} + \frac{\Delta t^4}{24}p_{tttt} - \frac{\Delta t^5}{120}p_{ttttt} + \mathcal{O}(\Delta t^6)$$

$$p(t_n,x_{j+1}) = p(t_n,x_j) + \Delta x p_x + \frac{\Delta x^2}{2}p_{xx} + \frac{\Delta x^3}{6}p_{xxx} + \frac{\Delta x^4}{24}p_{xxxx} + \frac{\Delta x^5}{120}p_{xxxx} + \mathcal{O}(\Delta x^6)$$

$$p(t_n,x_{j-1}) = p(t_n,x_j) - \Delta x p_x + \frac{\Delta x^2}{2}p_{xx} - \frac{\Delta x^3}{6}p_{xxx} + \frac{\Delta x^4}{24}p_{xxxx} - \frac{\Delta x^5}{120}p_{xxxx} + \mathcal{O}(\Delta x^6)$$

Now

$$\tau_j^{n+1}(\Delta t, \Delta x) = \frac{\Delta t^2 p_{tt} + \frac{\Delta t^4}{12} p_{tttt} + \mathcal{O}(\Delta t^6)}{\Delta t^2} - \frac{\Delta x^2 p_{xx} + \frac{\Delta x^4}{12} p_{xxxx} + \mathcal{O}(\Delta x^6)}{\Delta x^2}$$

$$\tau_j^{n+1}(\Delta t, \Delta x) = \frac{\Delta t^4}{12} p_{tttt} + \mathcal{O}(\Delta t^4) - \frac{\Delta x^4}{12} p_{xxxx} + \mathcal{O}(\Delta x^4)$$

Let $u(t,x) = p_{tt}(t,x) - p_{xx}(t,x)$ and $h = \Delta t = \Delta x$.

$$u(t,x) = U(t,x,h) + \frac{h^2}{12} p_{tttt}(t,x) + \mathcal{O}(h^4) - \frac{h^2}{12} p_{xxxx}(t,x) + \mathcal{O}(h^4)$$

where

$$U(t,x,h) = \frac{p(t+h,x) - 2p(t,x) + p(t-h,x)}{h^2} - \frac{p(t,x+h) - 2p(t,x) + p(t,x-h)}{h^2}$$

Now substituting h with $\frac{h}{2}$,

$$u(t,x) = U(t,x,\frac{h}{2}) + \frac{h^2}{48} p_{tttt}(t,x) + \mathcal{O}(\frac{h^4}{16}) - \frac{h^2}{48} p_{xxxx}(t,x) + \mathcal{O}(\frac{h^4}{16})$$

with

$$U(t,x,\frac{h}{2}) = \frac{p(t+\frac{h}{2},x) - 2p(t,x) + p(t-\frac{h}{2},x)}{\frac{h^2}{4}} - \frac{p(t,x+\frac{h}{2}) - 2p(t,x) + p(t,x-\frac{h}{2})}{\frac{h^2}{4}}$$

Using the formula for richardson extrapolation we see

$$\frac{4(U(t,x,\frac{h}{2}) + \frac{h^2}{48} p_{tttt}(t,x) + \mathcal{O}(\frac{h^4}{16}) - \frac{h^2}{48} p_{xxxx}(t,x) + \mathcal{O}(\frac{h^4}{16})) - (U(t,x,h) + \frac{h^2}{12} p_{tttt}(t,x) - \frac{h^2}{12} p_{xxxx}(t,x) + \mathcal{O}(h^4))}{3}$$

after combining like terms we are left with

$$\frac{4}{3} U(t,x,\frac{h}{2}) - \frac{1}{3} U(t,x,h) + \mathcal{O}(h^4)$$

Therefore our method is 4^{th} order.