In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
import scipy
import cmath
import timeit
import scipy.integrate as integrate
```

Question 1

Write a function which creates the Fourier transform matrix Fn (for any input n, not necessarily a power of 2).1 For n = 10, use suitable norms to verify that Fn is symmetric and that FnF n = nI.

In [2]:
```
def F(n):
    F = np.zeros((n,n),dtype = complex)
    w = cmath.exp(-2*cmath.pi*(0+1j)/n)
    for i in range(n):
        for k in range(n):
            F[i,k] = w**(i*k)
    return F
```

In [3]:
```
# n=10
print(F(10))
```

```
[[ 1.         +0.00000000e+00j  1.         +0.00000000e+00j
   1.         +0.00000000e+00j  1.         +0.00000000e+00j
   1.         +0.00000000e+00j  1.         +0.00000000e+00j
   1.         +0.00000000e+00j  1.         +0.00000000e+00j
   1.         +0.00000000e+00j  1.         +0.00000000e+00j]
 [ 1.         +0.00000000e+00j  0.80901699-5.87785252e-01j
   0.30901699-9.51056516e-01j -0.30901699-9.51056516e-01j
  -0.80901699-5.87785252e-01j -1.         +0.00000000e+00j
  -0.80901699+5.87785252e-01j -0.30901699+9.51056516e-01j
   0.30901699+9.51056516e-01j  0.80901699+5.87785252e-01j]
 [ 1.         +0.00000000e+00j  0.30901699-9.51056516e-01j
  -0.80901699-5.87785252e-01j -0.80901699+5.87785252e-01j
   0.30901699+9.51056516e-01j  1.         +5.55111512e-17j
```

```
 0.30901699-9.51056516e-01j -0.80901699-5.87785252e-01j
-0.80901699+5.87785252e-01j  0.30901699+9.51056516e-01j]
[ 1.         +0.00000000e+00j -0.30901699-9.51056516e-01j
-0.80901699+5.87785252e-01j  0.80901699+5.87785252e-01j
 0.30901699-9.51056516e-01j -1.         -1.66533454e-16j
 0.30901699+9.51056516e-01j  0.80901699-5.87785252e-01j
-0.80901699-5.87785252e-01j -0.30901699+9.51056516e-01j]
[ 1.         +0.00000000e+00j -0.80901699-5.87785252e-01j
 0.30901699+9.51056516e-01j  0.30901699-9.51056516e-01j
-0.80901699+5.87785252e-01j  1.         +5.55111512e-17j
-0.80901699-5.87785252e-01j  0.30901699+9.51056516e-01j
 0.30901699-9.51056516e-01j -0.80901699+5.87785252e-01j]
[ 1.         +0.00000000e+00j -1.         +0.00000000e+00j
 1.         +5.55111512e-17j -1.         -1.66533454e-16j
 1.         +5.55111512e-17j -1.         -1.66533454e-16j
 1.         +1.11022302e-16j -1.         -1.11022302e-16j
 1.         +1.11022302e-16j -1.         -1.11022302e-16j]
[ 1.         +0.00000000e+00j -0.80901699+5.87785252e-01j
 0.30901699-9.51056516e-01j  0.30901699+9.51056516e-01j
-0.80901699-5.87785252e-01j  1.         +1.11022302e-16j
-0.80901699+5.87785252e-01j  0.30901699-9.51056516e-01j
 0.30901699+9.51056516e-01j -0.80901699-5.87785252e-01j]
[ 1.         +0.00000000e+00j -0.30901699+9.51056516e-01j
-0.80901699-5.87785252e-01j  0.80901699-5.87785252e-01j
 0.30901699+9.51056516e-01j -1.         -1.11022302e-16j
 0.30901699-9.51056516e-01j  0.80901699+5.87785252e-01j
-0.80901699+5.87785252e-01j -0.30901699-9.51056516e-01j]
[ 1.         +0.00000000e+00j  0.30901699+9.51056516e-01j
-0.80901699+5.87785252e-01j -0.80901699-5.87785252e-01j
 0.30901699-9.51056516e-01j  1.         +1.11022302e-16j
 0.30901699+9.51056516e-01j -0.80901699+5.87785252e-01j
-0.80901699-5.87785252e-01j  0.30901699-9.51056516e-01j]
[ 1.         +0.00000000e+00j  0.80901699+5.87785252e-01j
 0.30901699+9.51056516e-01j -0.30901699+9.51056516e-01j
-0.80901699+5.87785252e-01j -1.         -1.11022302e-16j
-0.80901699-5.87785252e-01j -0.30901699-9.51056516e-01j
 0.30901699-9.51056516e-01j  0.80901699-5.87785252e-01j]]
```

In [4]:

```
#show F(n) is symmetric
F(10) == F(10).T
```

Out[4]:

```
array([[ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
```

```
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True],
       [ True,  True,  True,  True,  True,  True,  True,  True,  True,
         True]])
```

As Fn = FnT (Fn transpose), Fn is symmetric

In [5]:

```
# calculate Fn conjugate
F10_conj = np.conjugate(F(10))
print(F10_conj)
```

```
[[ 1.         -0.00000000e+00j  1.         -0.00000000e+00j
   1.         -0.00000000e+00j  1.         -0.00000000e+00j
   1.         -0.00000000e+00j  1.         -0.00000000e+00j
   1.         -0.00000000e+00j  1.         -0.00000000e+00j]
 [ 1.         -0.00000000e+00j  0.80901699+5.87785252e-01j
   0.30901699+9.51056516e-01j -0.30901699+9.51056516e-01j
  -0.80901699+5.87785252e-01j -1.         -0.00000000e+00j
  -0.80901699-5.87785252e-01j -0.30901699-9.51056516e-01j
   0.30901699-9.51056516e-01j  0.80901699-5.87785252e-01j]
 [ 1.         -0.00000000e+00j  0.30901699+9.51056516e-01j
  -0.80901699+5.87785252e-01j -0.80901699-5.87785252e-01j
   0.30901699-9.51056516e-01j  1.         -5.55111512e-17j
   0.30901699+9.51056516e-01j -0.80901699+5.87785252e-01j
  -0.80901699-5.87785252e-01j  0.30901699-9.51056516e-01j]
 [ 1.         -0.00000000e+00j -0.30901699+9.51056516e-01j
  -0.80901699-5.87785252e-01j  0.80901699-5.87785252e-01j
   0.30901699+9.51056516e-01j -1.         +1.66533454e-16j
```

```
  0.30901699-9.51056516e-01j  0.80901699+5.87785252e-01j
 -0.80901699+5.87785252e-01j -0.30901699-9.51056516e-01j]
[ 1.         -0.00000000e+00j -0.80901699+5.87785252e-01j
  0.30901699-9.51056516e-01j  0.30901699+9.51056516e-01j
 -0.80901699-5.87785252e-01j  1.         -5.55111512e-17j
 -0.80901699+5.87785252e-01j  0.30901699-9.51056516e-01j
  0.30901699+9.51056516e-01j -0.80901699-5.87785252e-01j]
[ 1.         -0.00000000e+00j -1.         -0.00000000e+00j
  1.         -5.55111512e-17j -1.         +1.66533454e-16j
  1.         -5.55111512e-17j -1.         +1.66533454e-16j
  1.         -1.11022302e-16j -1.         +1.11022302e-16j
  1.         -1.11022302e-16j -1.         +1.11022302e-16j]
[ 1.         -0.00000000e+00j -0.80901699-5.87785252e-01j
  0.30901699+9.51056516e-01j  0.30901699-9.51056516e-01j
 -0.80901699+5.87785252e-01j  1.         -1.11022302e-16j
 -0.80901699-5.87785252e-01j  0.30901699+9.51056516e-01j
  0.30901699-9.51056516e-01j -0.80901699+5.87785252e-01j]
[ 1.         -0.00000000e+00j -0.30901699-9.51056516e-01j
 -0.80901699+5.87785252e-01j  0.80901699+5.87785252e-01j
  0.30901699-9.51056516e-01j -1.         +1.11022302e-16j
  0.30901699+9.51056516e-01j  0.80901699-5.87785252e-01j
 -0.80901699-5.87785252e-01j -0.30901699+9.51056516e-01j]
[ 1.         -0.00000000e+00j  0.30901699-9.51056516e-01j
 -0.80901699-5.87785252e-01j -0.80901699+5.87785252e-01j
  0.30901699+9.51056516e-01j  1.         -1.11022302e-16j
  0.30901699-9.51056516e-01j -0.80901699-5.87785252e-01j
 -0.80901699+5.87785252e-01j  0.30901699+9.51056516e-01j]
[ 1.         -0.00000000e+00j  0.80901699-5.87785252e-01j
  0.30901699-9.51056516e-01j -0.30901699-9.51056516e-01j
 -0.80901699-5.87785252e-01j -1.         +1.11022302e-16j
 -0.80901699+5.87785252e-01j -0.30901699+9.51056516e-01j
  0.30901699+9.51056516e-01j  0.80901699+5.87785252e-01j]]
```

In [6]:

```
#FnFn_conj
F10F10conj = F(10)@F10_conj
print(F10F10conj)
```

```
[[ 1.00000000e+01+0.00000000e+00j -4.44089210e-16-4.44089210e-16j
  -4.44089210e-16-5.55111512e-16j -7.21644966e-16-2.22044605e-16j
  -1.11022302e-15-1.11022302e-16j -1.33226763e-15+2.22044605e-16j
  -1.55431223e-15+6.66133815e-16j -1.55431223e-15+1.55431223e-15j
  -2.22044605e-15+2.99760217e-15j -1.88737914e-15+7.66053887e-15j]
 [-4.44089210e-16+4.44089210e-16j  1.00000000e+01+0.00000000e+00j
```

```
 -4.25032467e-16-2.49452174e-15j -1.26084788e-15-1.25925379e-15j
 -1.49414117e-15-9.86400549e-16j -1.05728493e-15-4.96184264e-16j
 -1.09113991e-15+1.25074273e-17j -2.01654481e-15+4.49401988e-16j
 -2.55108953e-15+1.87653346e-15j -1.77635684e-15+3.74017009e-15j]
[-4.44089210e-16+5.55111512e-16j -4.25032467e-16+2.49452174e-15j
  1.00000000e+01+0.00000000e+00j -1.15581907e-15-4.02113701e-15j
 -1.56534632e-15-2.29956230e-15j -2.45436168e-15-1.16403063e-15j
 -2.44023185e-15-2.40733845e-16j -1.80800351e-15+4.30335081e-16j
 -3.55271368e-15+7.09477650e-16j -3.01624987e-15+2.17367313e-15j]
[-7.21644966e-16+2.22044605e-16j -1.26084788e-15+1.25925379e-15j
 -1.15581907e-15+4.02113701e-15j  1.00000000e+01+0.00000000e+00j
 -1.62402434e-15-5.77147424e-15j -1.81887545e-15-3.00399759e-15j
 -2.14414650e-15-1.45178036e-15j -2.66453526e-15-3.00176359e-16j
 -2.74892008e-15+1.87906410e-16j -2.83417542e-15+1.25483355e-15j]
[-1.11022302e-15+1.11022302e-16j -1.49414117e-15+9.86400549e-16j
 -1.56534632e-15+2.29956230e-15j -1.62402434e-15+5.77147424e-15j
  1.00000000e+01+0.00000000e+00j -1.74319870e-15-6.49948757e-15j
 -2.66453526e-15-3.58674104e-15j -2.68343690e-15-1.99774605e-15j
 -3.43882588e-15-1.09951728e-15j -2.78914391e-15+3.33663885e-16j]
[-1.33226763e-15-2.22044605e-16j -1.05728493e-15+4.96184264e-16j
 -2.45436168e-15+1.16403063e-15j -1.81887545e-15+3.00399759e-15j
 -1.74319870e-15+6.49948757e-15j  1.00000000e+01+0.00000000e+00j
 -2.32458300e-15-8.87836810e-15j -3.19259581e-15-4.43918855e-15j
 -3.46986385e-15-2.61541953e-15j -3.20160607e-15-6.70816686e-16j]
[-1.55431223e-15-6.66133815e-16j -1.09113991e-15-1.25074273e-17j
 -2.44023185e-15+2.40733845e-16j -2.14414650e-15+1.45178036e-15j
 -2.66453526e-15+3.58674104e-15j -2.32458300e-15+8.87836810e-15j
  1.00000000e+01+0.00000000e+00j -3.16730780e-15-1.02415339e-14j
 -3.53632295e-15-4.56706184e-15j -3.57932850e-15-2.78779691e-15j]
[-1.55431223e-15-1.55431223e-15j -2.01654481e-15-4.49401988e-16j
 -1.80800351e-15-4.30335081e-16j -2.66453526e-15+3.00176359e-16j
 -2.68343690e-15+1.99774605e-15j -3.19259581e-15+4.43918855e-15j
 -3.16730780e-15+1.02415339e-14j  1.00000000e+01+0.00000000e+00j
 -3.18212722e-15-1.15324386e-14j -3.92046388e-15-5.86865977e-15j]
[-2.22044605e-15-2.99760217e-15j -2.55108953e-15-1.87653346e-15j
 -3.55271368e-15-7.09477650e-16j -2.74892008e-15-1.87906410e-16j
 -3.43882588e-15+1.09951728e-15j -3.46986385e-15+2.61541953e-15j
 -3.53632295e-15+4.56706184e-15j -3.18212722e-15+1.15324386e-14j
  1.00000000e+01+0.00000000e+00j -4.45420550e-15-1.30696885e-14j]
[-1.88737914e-15-7.66053887e-15j -1.77635684e-15-3.74017009e-15j
 -3.01624987e-15-2.17367313e-15j -2.83417542e-15-1.25483355e-15j
 -2.78914391e-15-3.33663885e-16j -3.20160607e-15+6.70816686e-16j
 -3.57932850e-15+2.78779691e-15j -3.92046388e-15+5.86865977e-15j
 -4.45420550e-15+1.30696885e-14j  1.00000000e+01+0.00000000e+00j]]
```

In [7]:
```
#nI
I_10 = 10*np.eye(10)
print(I_10)
```

```
[[10.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 10.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. 10.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 10.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 10.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. 10.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. 10.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 10.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. 10.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. 10.]]
```

In [8]:
```
#norm of FnFn_conj
normFnFnconj = np.linalg.norm(F10F10conj)
print('The norm of FnFnconj is',normFnFnconj)
#norm of nI when n=10
normI = np.linalg.norm(I_10)
print('The norm of 10I is',normI)

print('The norm of FnFn_conj is approximately equal to the norm of 10I, shown as:')
np.isclose(normFnFnconj,normI)
print('The norms are not exactly equal to each other due to rounding errors')
```

```
The norm of FnFnconj is 31.622776601683853
The norm of 10I is 31.622776601683793
The norm of FnFn_conj is approximately equal to the norm of 10I, shown as:
The norms are not exactly equal to each other due to rounding errors
```

Question2

For n = 1; 2; 4; 8; 16; : : : ; 8192 (or less if it takes too long), plot the runtime of (a) building the matrix Fn using your code above; (b) calculating the DFT of x = (1; 2; 3; : : : ; n)T using matrix-vector multiplication (not including creating the matrix Fn); and (c) calculating the same DFT using NumPy's FFT function. See lab 4 for code which can measure runtime. Discuss whether your results match the expected results from theory.

In [9]:
```
 n = np.zeros(14,dtype=int)
for i in range(14):
```

```
    n[i] = (2**i)
print(n)
```

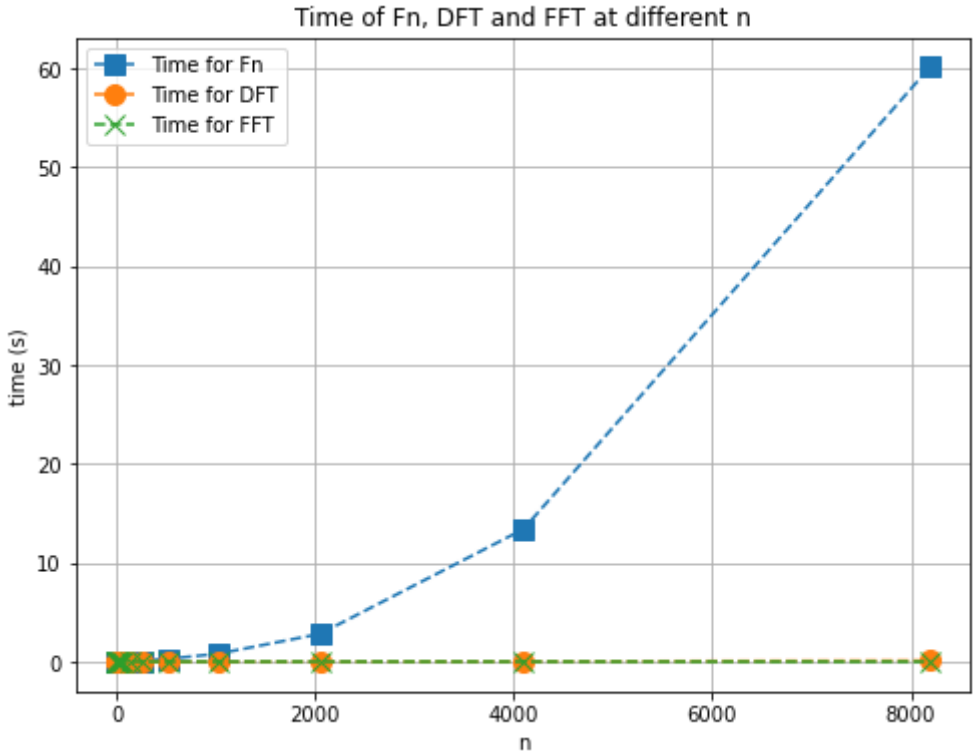
```
[   1    2    4    8   16   32   64  128  256  512 1024 2048 4096 8192]
```

In [10]:

```
#(a) the runtime of building the matrix Fn
time1 = []
time2 = []
time3 = []

for value in n:
    ts_Fn = timeit.default_timer()
    Fn = F(value)
    te_Fn = timeit.default_timer()
    time_Fn = te_Fn - ts_Fn
    time1.append(time_Fn)

#(b) the runtime of DFT
    x = np.arange(value).T
    ts_DFT = timeit.default_timer()
    xhat = Fn@x
    te_DFT = timeit.default_timer()
    time_DFT = te_DFT - ts_DFT
    time2.append(time_DFT)

#(c) the runtime of FFT
    ts_FFT = timeit.default_timer()
    xFFT = np.fft.fft(x)
    te_FFT = timeit.default_timer()
    time_FFT = te_FFT - ts_FFT
    time3.append(time_FFT)

print(time1,time2,time3)
```

```
[7.650000000047896e-05, 3.530000000040445e-05, 5.149999999964905e-05, 9.040000000037907e-05, 0.0002729999999999677, 0.0008571000000001661, 0.00420290000000012, 0.01172430000000002, 0.08139330000000022, 0.27644170000000035, 0.8047890000000004, 2.8070478000000003, 13.392626299999998, 60.126888] [2.0399999999476393e-05, 0.0008634999999994619, 2.949999999923847e-05, 1.71000000008803e-05, 1.3599999999946988e-05, 4.109999999979408e-05, 7.120000000071514e-05, 0.0015423000000005516, 0.0001932000000000448, 0.0004031000000006557, 0.002237199999999717, 0.005895399999999995, 0.02433680000000038, 0.10118900000000508] [0.000436299999999612, 0.0002941999999999112, 0.00011880000000000855, 2.3999999999801958e-05, 0.0002176000000000003998, 3.9300000000075386e-05, 3.8100000000262924e-05, 7.080000000048159e-05, 6.039999999973844e-05, 6.4000000000064e-05, 5.330000000025592e-05, 0.00012839999999947338, 0.0001983000000009838, 0.003128500000002532]
```

In [11]:
```
plt.figure(figsize=(8,6))
plt.plot(n,time1,'s--',markersize=10,label='Time for Fn')
plt.plot(n,time2,'o--',markersize=10,label='Time for DFT')
plt.plot(n,time3,'x--',markersize=10,label='Time for FFT')
plt.title('Time of Fn, DFT and FFT at different n')
plt.xlabel('n')
plt.ylabel('time (s)')
plt.legend()
plt.grid()
```



Observing from the plot above, the calculating Fn using our own code takes much longer time than calculating DFT of x and using FFT for x. The time taken for DFT and FFT are similar. This is because that the time complexity of Fn is O(n^2) as two 'for' loops are used, while FFT has a smaller time complexity of O(nlogn). Therefore, FFT calculation is much faster.

Question 3

Looking at your plots, pick a magnitude which is slightly larger than the typical 'noise' contribution, but much smaller than the size of any spikes. Modify the DFT of noisy_data by setting to zero all entries with magnitude less than your chosen level, and calculate the inverse DFT to produce a 'denoised' signal (you will have to look up how to calculate an inverse DFT in NumPy). Produce a plot the original signal, noisy signal and your denoised signal and check that you have removed a reasonable amount of the noise. Quantitatively compare the error (versus data) of the noisy and your denoised signal using suitable norms, to check that your denoised signal is closer to the true data than noisy_data.

In [12]:
```
data = np.loadtxt('lab5_piano_data.csv', delimiter=',')
time = np.linspace(0.0, 1.0, len(data)) # data represents 1 second of audio
print("CSV has a vector of size =", data.shape)
```

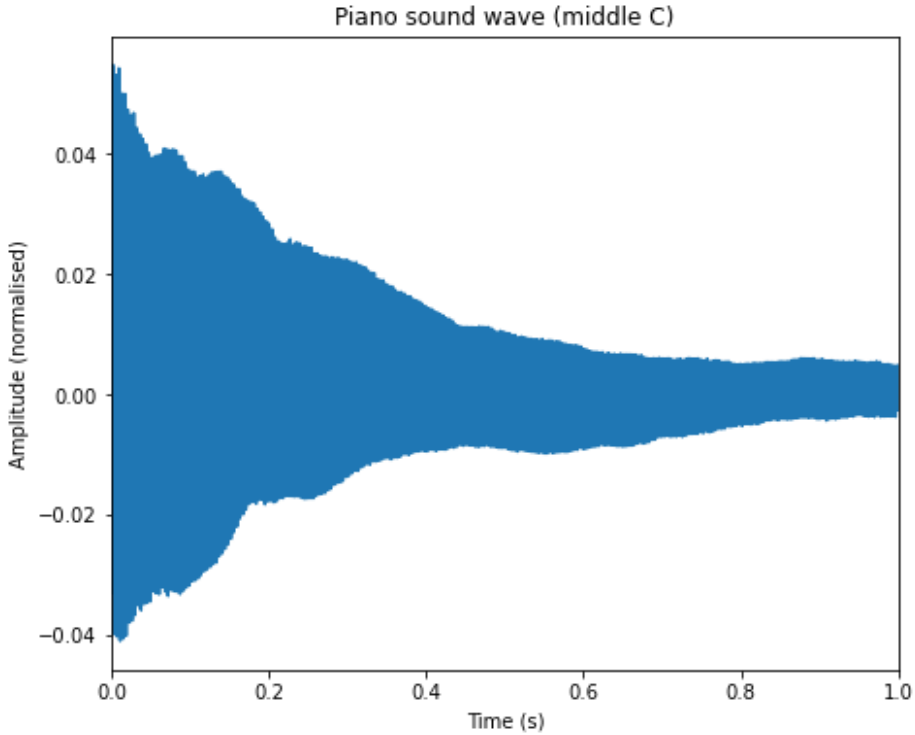
```
CSV has a vector of size = (44100,)
```

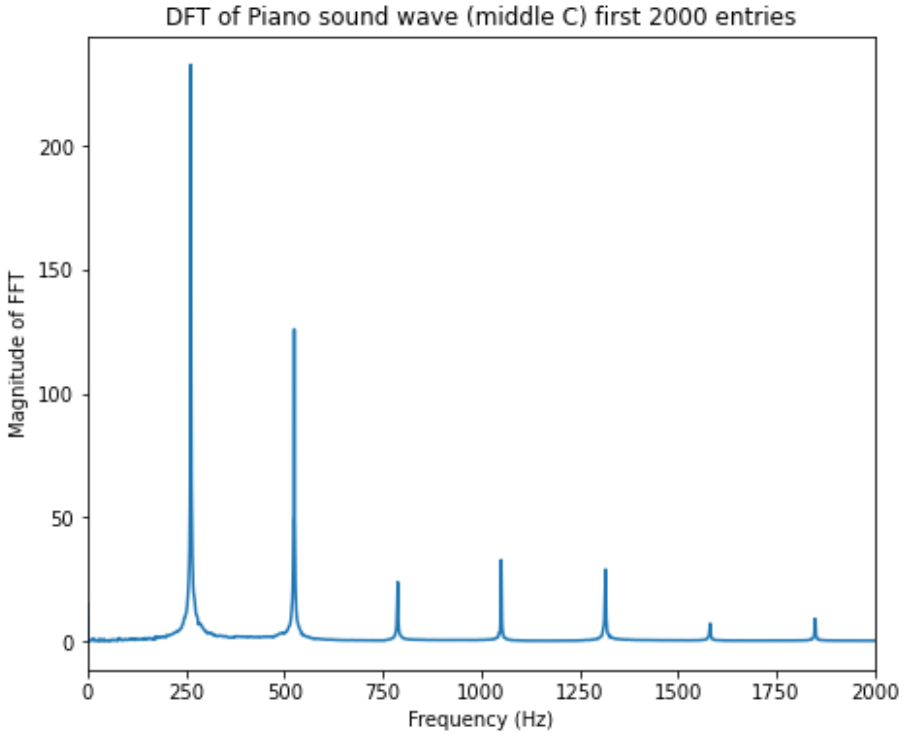
In [13]:
```
#from scipy.fft import fft, fftfreq

# Number of samples in normalized_tone
N = 44100
xf = np.fft.fftfreq(N, 1 / N)
print(xf)
```

```
[ 0.   1.   2. ... -3. -2. -1.]
```

In [14]:
```
plt.figure(figsize=(16,6))

plt.subplot(1,2,1)
plt.plot(time,data)
plt.title('Piano sound wave (middle C)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude (normalised)')
plt.xlim(0.0,1.0)

plt.subplot(1,2,2)
plt.plot(np.arange(len(data)),abs(np.fft.fft(data[:])))
plt.title('DFT of Piano sound wave (middle C) first 2000 entries')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude of FFT')
plt.xlim(0.0,2000)
```

Out[14]:
```
(0.0, 2000.0)
```



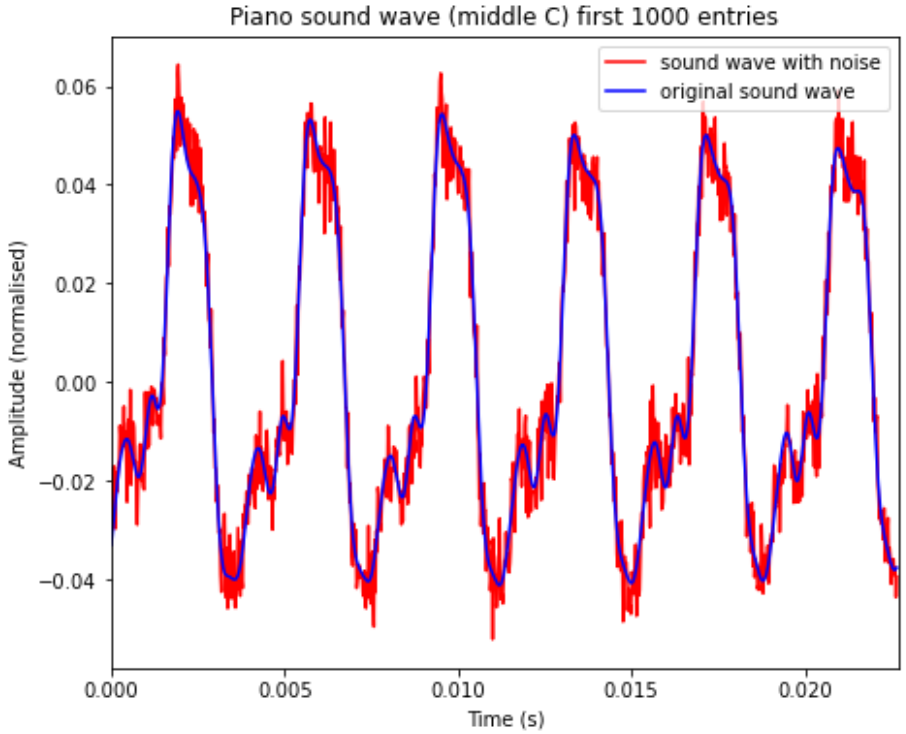


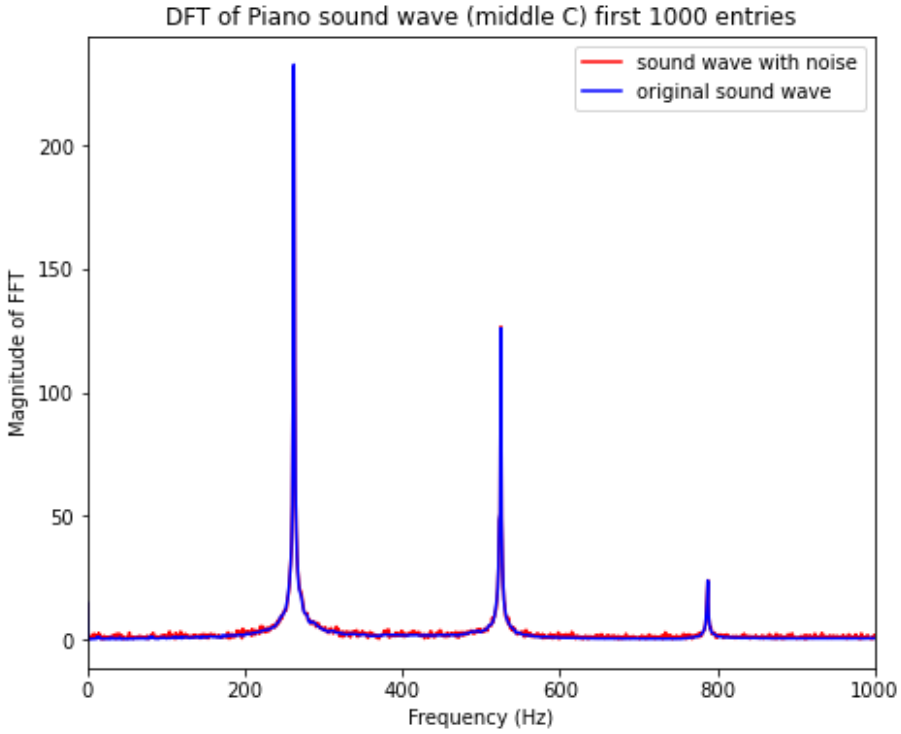
In [15]:
```
np.random.seed(0) # produce the same random numbers every time the code is run (optional)
noisy_data = data + 0.005 * np.random.randn(len(data)) # perturb data with random noise
```

In [16]:
```
plt.figure(figsize=(16,6))

plt.subplot(1,2,1)
plt.plot(time[:1000],noisy_data[:1000],'r',label='sound wave with noise')
plt.plot(time[:1000],data[:1000],'b',label='original sound wave')
plt.title('Piano sound wave (middle C) first 1000 entries')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude (normalised)')
plt.xlim(0.0,time[1001])
plt.legend()

plt.subplot(1,2,2)
noise_FFT = abs(np.fft.fft(noisy_data[:]))
```

```
plt.plot(np.arange(len(noisy_data)),noise_FFT,'r',label='sound wave with noise')
plt.plot(np.arange(len(data)),abs(np.fft.fft(data)),'b',label='original sound wave')
plt.title('DFT of Piano sound wave (middle C) first 1000 entries')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude of FFT')
plt.xlim(0.0,1000)
plt.legend()
```

Out[16]: <matplotlib.legend.Legend at 0x2439d9d5070>





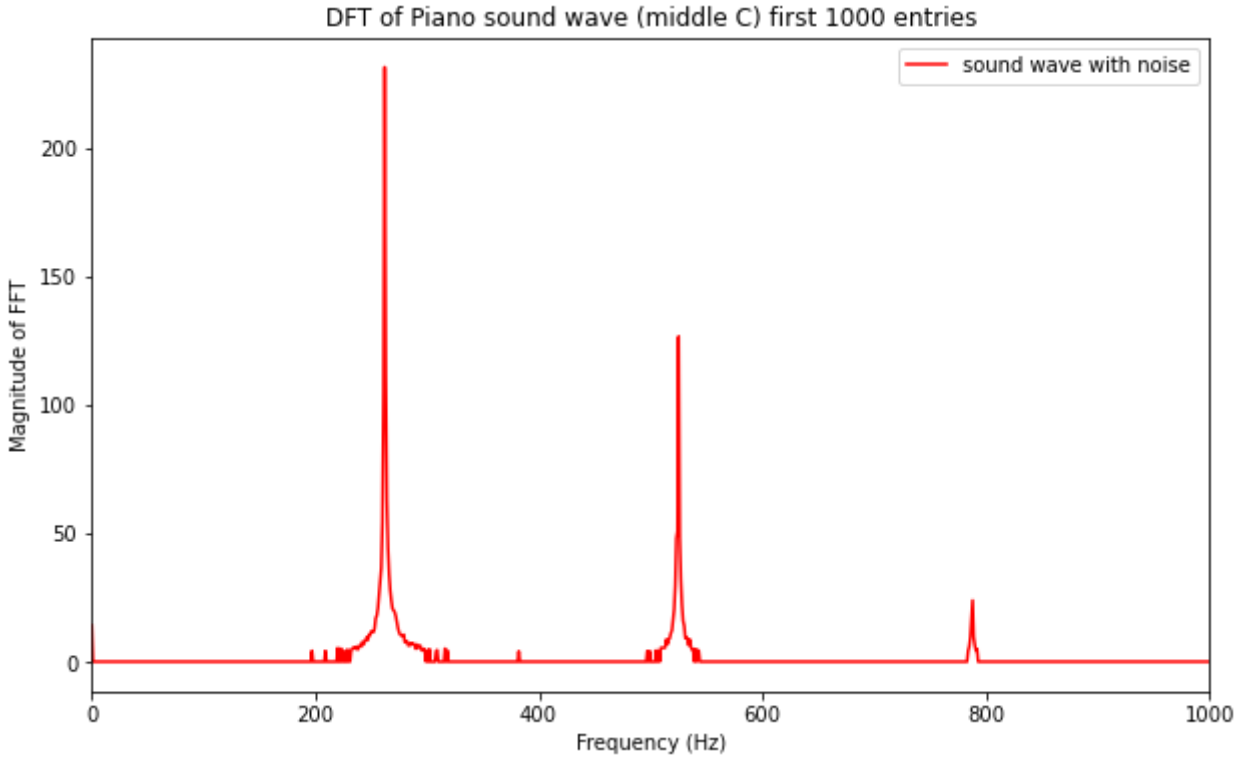
In [17]:

```
#choose the denoised value = 4
mask = (abs(np.fft.fft(noisy_data))>4) #find an array with magnitude>4 is 1 and magnitude =<4 is 0
print(mask)
denoised_FFT = mask*np.fft.fft(noisy_data[:])
print(denoised_FFT)

plt.figure(figsize=(10,6))
plt.plot(np.arange(len(noisy_data)),abs(denoised_FFT),'r',label='sound wave with noise')
plt.title('DFT of Piano sound wave (middle C) first 1000 entries')
```

```
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude of FFT')
plt.xlim(0.0,1000)
plt.legend()
```

```
[ True False False ... False False False]
[14.06344878+0.j  0.        -0.j  0.        +0.j ...  0.        +0.j
  0.        +0.j -0.        +0.j]
```

Out[17]: <matplotlib.legend.Legend at 0x2439d9191f0>



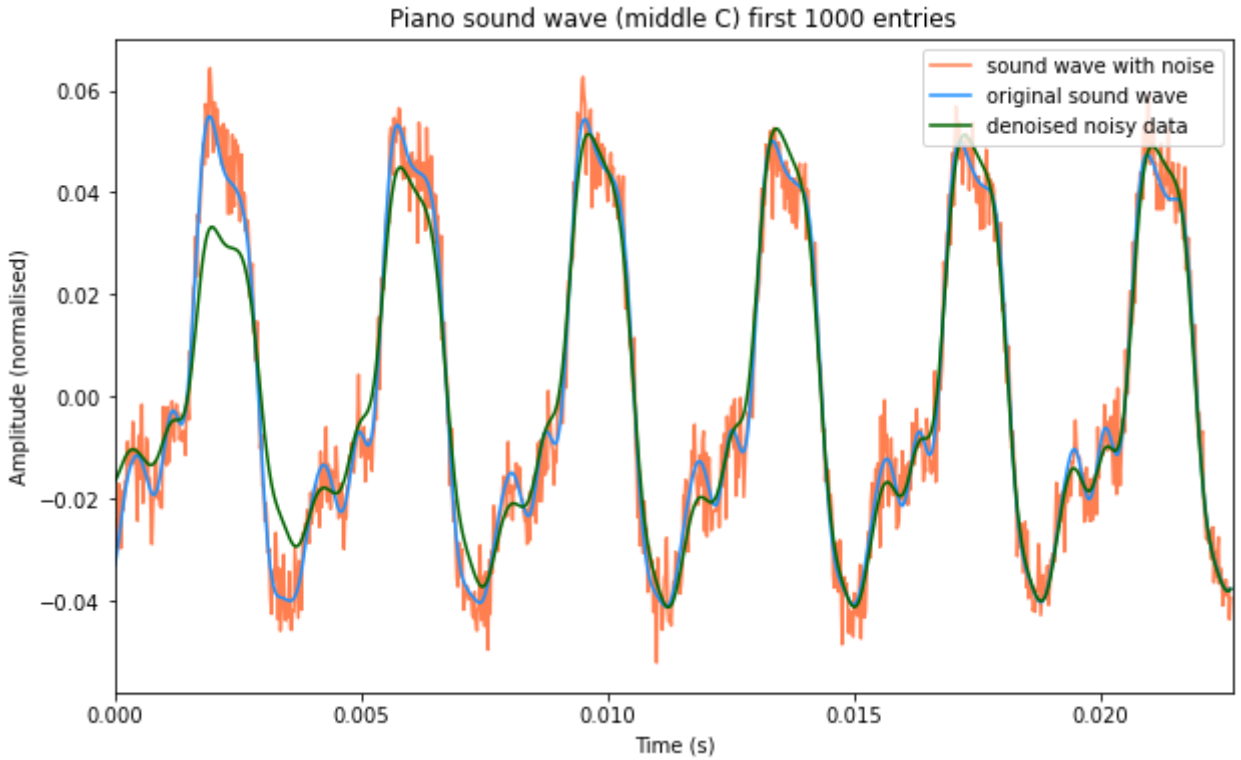
In [18]:

```
# inverse FFT of the denoised FFT
denoised_data = np.fft.ifft(denoised_FFT)

plt.figure(figsize=(10,6))
plt.plot(time[:1000],noisy_data[:1000],color='coral',label='sound wave with noise')
plt.plot(time[:1000],data[:1000],color='dodgerblue',label='original sound wave')
plt.plot(time[:1000],denoised_data[:1000],color='darkgreen',label='denoised noisy data')
plt.title('Piano sound wave (middle C) first 1000 entries')
```

```
plt.xlabel('Time (s)')
plt.ylabel('Amplitude (normalised)')
plt.xlim(0.0,time[1001])
plt.legend()
```

```
C:\Users\huang\anaconda3\lib\site-packages\numpy\core\_asarray.py:102: ComplexWarning: Casting complex values to real discards the im
aginary part
  return array(a, dtype, copy=False, order=order)
```

Out[18]:
```
<matplotlib.legend.Legend at 0x2439d9128e0>
```



In [19]:
```
# error between denoised data and original data
err_deno = denoised_data - data
err_noise = noisy_data - data
print('The norm of error between denoised data and original data is',np.linalg.norm(err_deno))
print('The norm of error between noisy data and original data is',np.linalg.norm(err_noise))
```

```
The norm of error between denoised data and original data is 0.3119844711582622
The norm of error between noisy data and original data is 1.0451029075185467
```

Observing from above plot and errors, the denoised data is much better than the noisy data as the norm of error between denoised data and original data is much smaller.

Question 4

Modify the above code to implement our ODE (f and u0) for time horizon T = 100 days. Then, implement the one-step methods: explicit Euler, Heun's method and the classical 4th order Runge-Kutta scheme (RK4). For n = 200, plot your three computed solutions against the true solution u(t). Interpret the solution to this ODE in terms of the application setting (infectious disease).

In [224...
```
c = 0.2
P = 10000.0
T = 100.0 # end time
```

In [225...
```
#True solution
n = 200
ts = np.linspace(0, T, n+1)
u_true = 1/(1+(P-1)*np.exp(-c*ts))
```

In [226...
```
def f(t, u):
    dudt = c*u*(1-u)
    return dudt
```

In [227...
```
#Explicit Euler
def Euler(n):

    def phi(t, u):
        phi = f(t,u)
        return phi

    u0 = 1/P # initial condition

    ts = np.linspace(0, T, n+1) # vector of timesteps, tk = ts[k]
    h = T / n # gap between timesteps

    u_euler = np.zeros((n+1,)) # create an empty vector for our solution
    u_euler[0] = u0 # set initial condition
```

```
    # Run the one-step method
    for k in range(n):
        u_euler[k+1] = u_euler[k] + h * phi(ts[k], u_euler[k])

    return u_euler
```

In [228…

```
#Heun's method
def Heun(n):


    def phi(t, u, h):
        phi = f(t,u)/2+f(t+h,u+h*f(t,u))/2
        return phi

    u0 = 1/P # initial condition

    ts = np.linspace(0, T, n+1) # vector of timesteps, tk = ts[k]
    h = T / n # gap between timesteps

    u_heun = np.zeros((n+1,)) # create an empty vector for our solution
    u_heun[0] = u0 # set initial condition
    # Run the one-step method
    for k in range(n):
        u_heun[k+1] = u_heun[k] + h * phi(ts[k], u_heun[k],h)

    return u_heun
```

In [229…

```
#RK4
def RK4(t0, u0, tmin, tmax, h):
    ts = []
    us = []
    t = t0
    ts.append(t)
    u = u0
    us.append(u)

    while ts[-1]+h <= tmax:
        k1 = f(t, u)
        k2 = f(t + h/2.0, u + k1*h/2.0)
        k3 = f(t + h/2.0, u + k2*h/2.0)
        k4 = f(t + h, u + k3*h)
```

```
            # Update next value of x
        u = u + (h / 6.0)*(k1 + 2.0*k2 + 2.0*k3 + k4)
            # Update next value of t
        t = t + h

        ts.append(t)
        us.append(u)
    return ts,us
```

In [230…

```
u0 = 1/P
t0 = 0.0
tmin = 0.0
tmax = 100.0
h = T/n

ts,u_RK4 = RK4(t0, u0, tmin, tmax, h)
```

In [231…

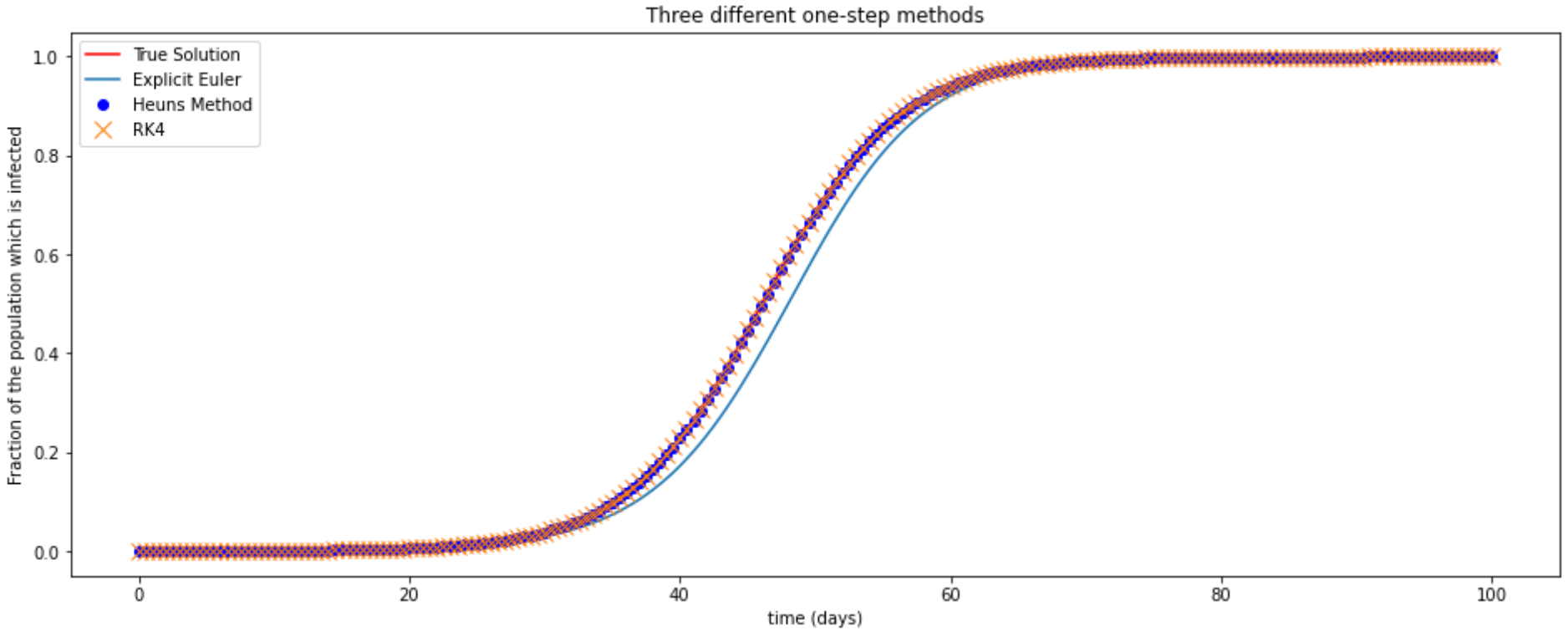
```
u_euler = Euler(200)
u_heun = Heun(200)
```

In [232…

```
plt.figure(figsize=(16,6))
plt.plot(ts,u_true,'r-',label='True Solution')
plt.plot(ts,u_euler,'-',markersize=5,label='Explicit Euler')
plt.plot(ts,u_heun,'bo',label='Heuns Method')
plt.plot(ts,u_RK4,'x',markersize=10,label='RK4')
plt.xlabel('time (days)')
plt.ylabel('Fraction of the population which is infected')
plt.title('Three different one-step methods')
plt.legend()
```

Out[232…

```
<matplotlib.legend.Legend at 0x2441cd48640>
```



Comparing the three methods, the Heun's method and RK4 return more accurate results than the Explicit Euler method. In terms of the application setting, in the explicit euler method, the increasing rate of the infected people is slower at the first 50 days and increase faster in the later 50 days.

Question 5

For n = 50; 100; 200; 400; 800; 1600 and each of the three one-step methods, produce a table similar to slide 41 of the ODE lectures, computing the maximum absolute error juk u(tk)j at any time step tk for each method. From your table, determine what order of convergence you are seeing in each method and compare this to the theory in lectures.

In [233…
```
from tabulate import tabulate
```

In [236…
```
nn = np.array([50,100,200,400,800,1600])
data = []
for n in nn:
```

```
    ts = np.linspace(0, T, n+1)
    u_true = 1/(1+(P-1)*np.exp(-c*ts))
    u_euler = Euler(n)
    u_heun = Heun(n)
    t_RK4,u_RK4 = RK4(t0, u0, tmin, tmax, T/n)

    h = T/n
    err_euler = max(abs(u_euler - u_true))
    err_heun = max(abs(u_heun - u_true))
    err_RK4 = max(abs(u_RK4 - u_true))

    data.append([h,err_euler,err_heun,err_RK4])

print(tabulate(data,headers=["h","Euler","Heun","RK4"]))
```

```
     h       Euler          Heun           RK4
------  ---------  -----------  -----------
2       0.351179   0.0392733    0.000278612
1       0.187613   0.0112002    2.03236e-05
0.5     0.0961273  0.00299973   1.37305e-06
0.25    0.0485262  0.00077616   8.92409e-08
0.125   0.0243631  0.000197484  5.68831e-09
0.0625  0.0122043  4.98083e-05  3.59014e-10
```

In [269…

```
h = np.zeros(6)
E = np.zeros(6)
H = np.zeros(6)
RK = np.zeros(6)
for i in range(6):
    h[i] = data[i][0]
    E[i] = data[i][1]
    H[i] = data[i][2]
    RK[i] = data[i][3]
```

In [272…

```
slope_E = np.polyfit(np.log(h),np.log(E),1)[0]
slope_H = np.polyfit(np.log(h),np.log(H),1)[0]
slope_RK = np.polyfit(np.log(h),np.log(RK),1)[0]
print('The rate of convergence of Euler method is approximately',slope_E)
print('The rate of convergence of Heuns method is approximately',slope_H)
print('The rate of convergence of RK4 method is approximately',slope_RK)
```

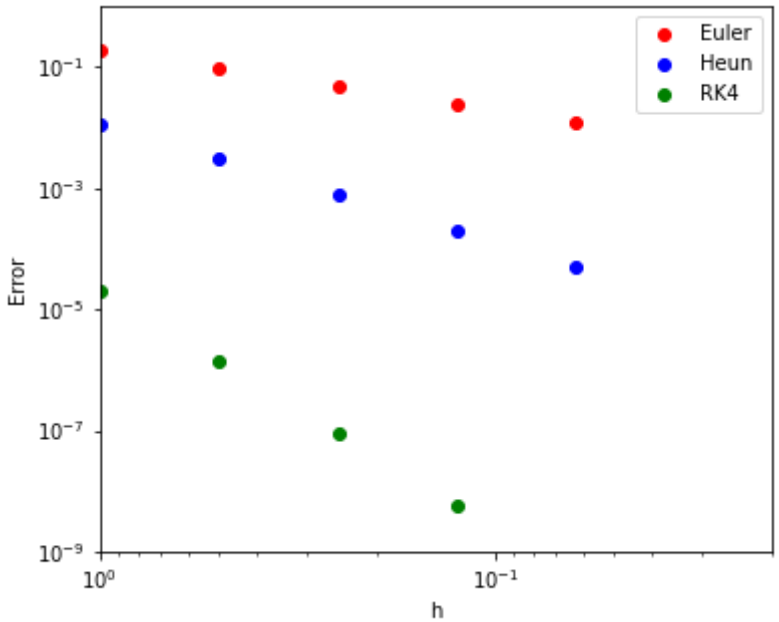
```
The rate of convergence of Euler method is approximately 0.9729965612487067
The rate of convergence of Heuns method is approximately 1.9297731741060287
The rate of convergence of RK4 method is approximately 3.919458729847732
```

In [268…

```
plt.figure(figsize=(6,5))
for i in range(6):
    plt.plot(data[i][0],data[i][1],'ro')
    plt.plot(data[i][0],data[i][2],'bo')
    plt.plot(data[i][0],data[i][3],'go')
plt.xscale('log')
plt.yscale('log')
plt.xlim((1.0,0.02))
plt.ylim((1e-9,1.0))
plt.xlabel('h')
plt.ylabel('Error')
plt.legend(['Euler','Heun','RK4'])
```

Out[268…

```
<matplotlib.legend.Legend at 0x24423452a00>
```



From the table above, Euler method is having a rate of convergence O(h) as the error halved for each halved step size (h); Heun's method is having a rate of convergence O(h^2) as the number of accurate digit increase approximate by one for each halved step size (h), RK4 is having a rate of

convergence O(h^4) as the slope of error in relation with h is approximately 4. Observe from the calculated slope, table, and plot, the rate of convergence of all three methods are same as documented in the lecture notes.

Question 6

Use solve_ivp with RK45 to solve the original and our new ODE and plot the two solutions. How many people are healthy after 100 days under the two scenarios? How many extra people would be healthy at day 100 if we started vaccinating on day t = 0 rather than day t = 5?

In [33]:
```
#no vaccination
def f(t, u):
    dudt = c*u*(1-u)
    return dudt

#vaccination starting from day 5
def immu(t,u):
    dudt = c*u*max(1-u-0.01*max(t-5,0),0)
    return dudt
```

In [34]:
```
#vaccination starting from day 0
def immu0(t,u):
    dudt = c*u*max(1-u-0.01*max(t,0),0)
    return dudt
```

In [35]:
```
c = 0.2
P = 10000.0
T = 100.0 # end time
u0 = np.array([1/P]) # initial condition (must be a NumPy array)

sol_u = integrate.solve_ivp(f, [0.0, T], u0, method='RK45')
sol_immu = integrate.solve_ivp(immu, [0.0, T], u0, method='RK45')
sol_immu0 = integrate.solve_ivp(immu0, [0.0, T], u0, method='RK45')
print(sol_u.y[0,:])
print(sol_immu.y[0,:])
print(sol_immu0.y[0,:])
```

```
[1.00000000e-04 1.04558076e-04 1.63278161e-04 7.90486247e-04
 4.51457260e-03 2.25428218e-02 9.69788815e-02 3.38789064e-01
 7.43196951e-01 8.88468325e-01 9.56391351e-01 9.90418131e-01
```

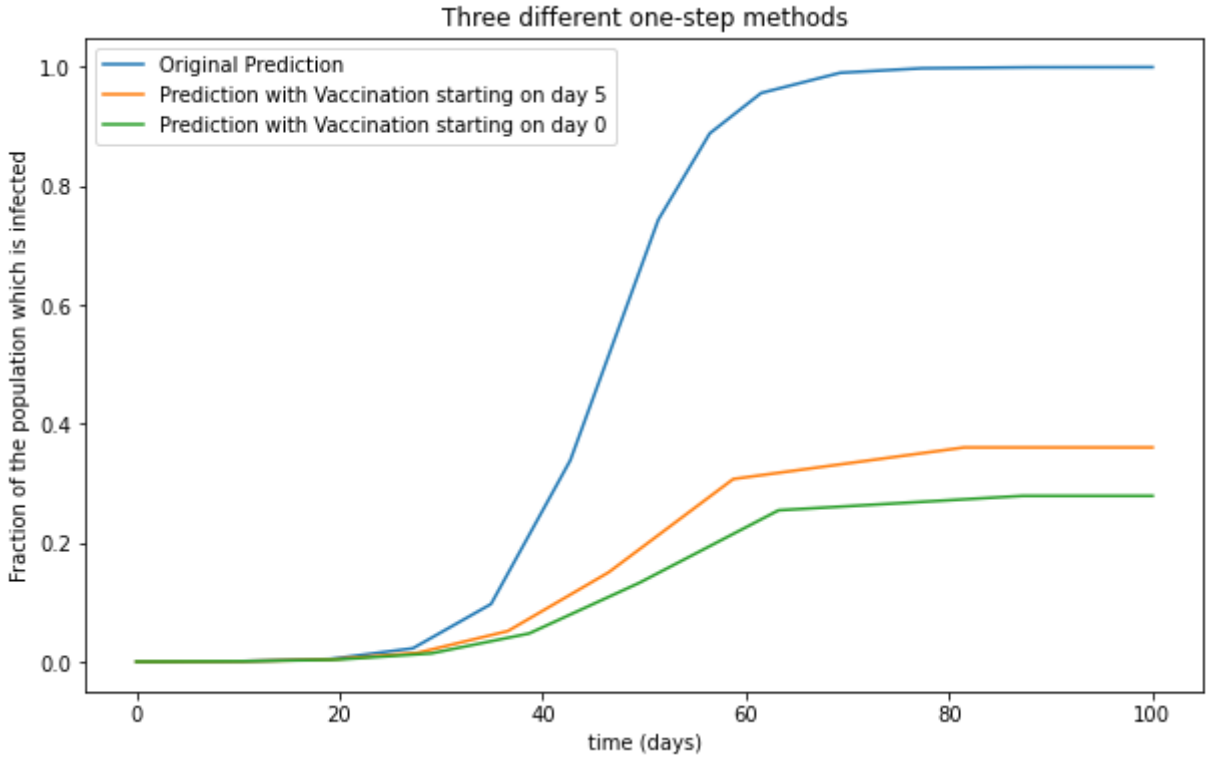
```
 9.97999512e-01 9.99652244e-01 9.99925117e-01]
[1.00000000e-04 1.04558076e-04 1.63278161e-04 7.69446527e-04
 3.65870409e-03 1.48210204e-02 5.15684380e-02 1.50456584e-01
 3.07277154e-01 3.60410221e-01 3.60410221e-01]
[1.00000000e-04 1.04552882e-04 1.62299646e-04 7.30204257e-04
 3.51265917e-03 1.40424026e-02 4.74876782e-02 1.32780007e-01
 2.54695575e-01 2.78914116e-01 2.78914116e-01]
```

In [36]:

```
plt.figure(figsize=(10,6))
plt.plot(sol_u.t,sol_u.y[0,:],'-',label='Original Prediction')
plt.plot(sol_immu.t,sol_immu.y[0,:],'-',label='Prediction with Vaccination starting on day 5')
plt.plot(sol_immu0.t,sol_immu0.y[0,:],'-',label='Prediction with Vaccination starting on day 0')
plt.xlabel('time (days)')
plt.ylabel('Fraction of the population which is infected')
plt.title('Three different one-step methods')
plt.legend()
```

Out[36]:

```
<matplotlib.legend.Legend at 0x2439fe45910>
```



In [37]:
```
print('In normal case, the number of healthy people after 100 days is',P - P*sol_u.y[0,-1],',which is no one.')
print('In case with vaccination starting from day 5, the number of healthy people after 100 days is',P - P*sol_immu.y[0,-1],',which
```

```
In normal case, the number of healthy people after 100 days is 0.7488286951411283 ,which is no one.
In case with vaccination starting from day 5, the number of healthy people after 100 days is 6395.897785341493 ,which is 6395 people.
```

In [38]:
```
# Compare vaccination starting from day 0 and day 5
print((P - P*sol_immu0.y[0,-1])-(P - P*sol_immu.y[0,-1]),',which is 814 extra people are healthy if vaccination start from day 0 ra
```

```
814.9610528256426 ,which is 814 extra people are healthy if vaccination start from day 0 rather than day 5.
```

Question 7

Solve this system of ODEs using the explicit Euler method, uk+1 = uk + ┌t f(t;uk) for some time discretisation ┌t. Using data a = b = 0, c = 1 and f(x) = sin(├x), plus spatial discretisation n = 10 and time step ┌t = 0:5=n2, plot the solution u(t; x) versus x for t = 0; 0:1; 0:2; : : : ; 0:5.

In [163…
```
a = 0
b = 0
c = 1

def u_int(x):
    f = np.sin(np.pi*x)
    return f
```

In [165…
```
x,ans = ExplicitEuler(10)
print('The shape of answer is',ans.shape,',which corresponds to 11 position points xi and 101 time steps')
```

```
The shape of answer is (11, 101) ,which corresponds to 11 position points xi and 101 time steps
```

In [166…
```
for i in range(101):
    print('The values of u at t=',dt*i,'is',ans[:,i])
```

```
The values of u at t= 0.0 is [0.00000000e+00 3.09016994e-01 5.87785252e-01 8.09016994e-01
 9.51056516e-01 1.00000000e+00 9.51056516e-01 8.09016994e-01
 5.87785252e-01 3.09016994e-01 1.22464680e-16]
The values of u at t= 0.005 is [0.00000000e+00 2.93892626e-01 5.59016994e-01 7.69420884e-01
 9.04508497e-01 9.51056516e-01 9.04508497e-01 7.69420884e-01
```

```
 5.59016994e-01 2.93892626e-01 1.22464680e-16]
The values of u at t= 0.01 is [0.00000000e+00 2.79508497e-01 5.31656755e-01 7.31762746e-01
 8.60238700e-01 9.04508497e-01 8.60238700e-01 7.31762746e-01
 5.31656755e-01 2.79508497e-01 1.22464680e-16]
The values of u at t= 0.015 is [0.00000000e+00 2.65828378e-01 5.05635621e-01 6.95947728e-01
 8.18135621e-01 8.60238700e-01 8.18135621e-01 6.95947728e-01
 5.05635621e-01 2.65828378e-01 1.22464680e-16]
The values of u at t= 0.02 is [0.00000000e+00 2.52817811e-01 4.80888053e-01 6.61885621e-01
 7.78093214e-01 8.18135621e-01 7.78093214e-01 6.61885621e-01
 4.80888053e-01 2.52817811e-01 1.22464680e-16]
The values of u at t= 0.025 is [0.00000000e+00 2.40444026e-01 4.57351716e-01 6.29490633e-01
 7.40010621e-01 7.78093214e-01 7.40010621e-01 6.29490633e-01
 4.57351716e-01 2.40444026e-01 1.22464680e-16]
The values of u at t= 0.03 is [0.00000000e+00 2.28675858e-01 4.34967330e-01 5.98681169e-01
 7.03791924e-01 7.40010621e-01 7.03791924e-01 5.98681169e-01
 4.34967330e-01 2.28675858e-01 1.22464680e-16]
The values of u at t= 0.035 is [0.00000000e+00 2.17483665e-01 4.13678513e-01 5.69379627e-01
 6.69345895e-01 7.03791924e-01 6.69345895e-01 5.69379627e-01
 4.13678513e-01 2.17483665e-01 1.22464680e-16]
The values of u at t= 0.04 is [0.00000000e+00 2.06839257e-01 3.93431646e-01 5.41512204e-01
 6.36585775e-01 6.69345895e-01 6.36585775e-01 5.41512204e-01
 3.93431646e-01 2.06839257e-01 1.22464680e-16]
The values of u at t= 0.045 is [0.00000000e+00 1.96715823e-01 3.74175730e-01 5.15008711e-01
 6.05429050e-01 6.36585775e-01 6.05429050e-01 5.15008711e-01
 3.74175730e-01 1.96715823e-01 1.22464680e-16]
The values of u at t= 0.05 is [0.00000000e+00 1.87087865e-01 3.55862267e-01 4.89802390e-01
 5.75797243e-01 6.05429050e-01 5.75797243e-01 4.89802390e-01
 3.55862267e-01 1.87087865e-01 1.22464680e-16]
The values of u at t= 0.055 is [0.00000000e+00 1.77931133e-01 3.38445128e-01 4.65829755e-01
 5.47615720e-01 5.75797243e-01 5.47615720e-01 4.65829755e-01
 3.38445128e-01 1.77931133e-01 1.22464680e-16]
The values of u at t= 0.06 is [0.00000000e+00 1.69222564e-01 3.21880444e-01 4.43030424e-01
 5.20813499e-01 5.47615720e-01 5.20813499e-01 4.43030424e-01
 3.21880444e-01 1.69222564e-01 1.22464680e-16]
The values of u at t= 0.065 is [0.00000000e+00 1.60940222e-01 3.06126494e-01 4.21346971e-01
 4.95323072e-01 5.20813499e-01 4.95323072e-01 4.21346971e-01
 3.06126494e-01 1.60940222e-01 1.22464680e-16]
The values of u at t= 0.07 is [0.00000000e+00 1.53063247e-01 2.91143597e-01 4.00724783e-01
 4.71080235e-01 4.95323072e-01 4.71080235e-01 4.00724783e-01
 2.91143597e-01 1.53063247e-01 1.22464680e-16]
The values of u at t= 0.075 is [0.00000000e+00 1.45571798e-01 2.76894015e-01 3.81111916e-01
 4.48023927e-01 4.71080235e-01 4.48023927e-01 3.81111916e-01
 2.76894015e-01 1.45571798e-01 1.22464680e-16]
The values of u at t= 0.08 is [0.00000000e+00 1.38447007e-01 2.63341857e-01 3.62458971e-01
```

```
 4.26096076e-01 4.48023927e-01 4.26096076e-01 3.62458971e-01
 2.63341857e-01 1.38447007e-01 1.22464680e-16]
The values of u at t= 0.085 is [0.00000000e+00 1.31670929e-01 2.50452989e-01 3.44718966e-01
 4.05241449e-01 4.26096076e-01 4.05241449e-01 3.44718966e-01
 2.50452989e-01 1.31670929e-01 1.22464680e-16]
The values of u at t= 0.09 is [0.00000000e+00 1.25226495e-01 2.38194947e-01 3.27847219e-01
 3.85407521e-01 4.05241449e-01 3.85407521e-01 3.27847219e-01
 2.38194947e-01 1.25226495e-01 1.22464680e-16]
The values of u at t= 0.095 is [0.00000000e+00 1.19097474e-01 2.26536857e-01 3.11801234e-01
 3.66544334e-01 3.85407521e-01 3.66544334e-01 3.11801234e-01
 2.26536857e-01 1.19097474e-01 1.22464680e-16]
The values of u at t= 0.1 is [0.00000000e+00 1.13268428e-01 2.15449354e-01 2.96540596e-01
 3.48604378e-01 3.66544334e-01 3.48604378e-01 2.96540596e-01
 2.15449354e-01 1.13268428e-01 1.22464680e-16]
The values of u at t= 0.105 is [0.00000000e+00 1.07724677e-01 2.04904512e-01 2.82026866e-01
 3.31542465e-01 3.48604378e-01 3.31542465e-01 2.82026866e-01
 2.04904512e-01 1.07724677e-01 1.22464680e-16]
The values of u at t= 0.11 is [0.00000000e+00 1.02452256e-01 1.94875771e-01 2.68223488e-01
 3.15315622e-01 3.31542465e-01 3.15315622e-01 2.68223488e-01
 1.94875771e-01 1.02452256e-01 1.22464680e-16]
The values of u at t= 0.115 is [0.00000000e+00 9.74378857e-02 1.85337872e-01 2.55095697e-01
 2.99882977e-01 3.15315622e-01 2.99882977e-01 2.55095697e-01
 1.85337872e-01 9.74378857e-02 1.22464680e-16]
The values of u at t= 0.12 is [0.00000000e+00 9.26689361e-02 1.76266791e-01 2.42610424e-01
 2.85205659e-01 2.99882977e-01 2.85205659e-01 2.42610424e-01
 1.76266791e-01 9.26689361e-02 1.22464680e-16]
The values of u at t= 0.125 is [0.00000000e+00 8.81333956e-02 1.67639680e-01 2.30736225e-01
 2.71246701e-01 2.85205659e-01 2.71246701e-01 2.30736225e-01
 1.67639680e-01 8.81333956e-02 1.22464680e-16]
The values of u at t= 0.13 is [0.00000000e+00 8.38198401e-02 1.59434810e-01 2.19443190e-01
 2.57970942e-01 2.71246701e-01 2.57970942e-01 2.19443190e-01
 1.59434810e-01 8.38198401e-02 1.22464680e-16]
The values of u at t= 0.135 is [0.00000000e+00 7.97174052e-02 1.51631515e-01 2.08702876e-01
 2.45344946e-01 2.57970942e-01 2.45344946e-01 2.08702876e-01
 1.51631515e-01 7.97174052e-02 1.22464680e-16]
The values of u at t= 0.14 is [0.00000000e+00 7.58157576e-02 1.44210141e-01 1.98488230e-01
 2.33336909e-01 2.45344946e-01 2.33336909e-01 1.98488230e-01
 1.44210141e-01 7.58157576e-02 1.22464680e-16]
The values of u at t= 0.145 is [0.00000000e+00 7.21050703e-02 1.37151994e-01 1.88773525e-01
 2.21916588e-01 2.33336909e-01 2.21916588e-01 1.88773525e-01
 1.37151994e-01 7.21050703e-02 1.22464680e-16]
The values of u at t= 0.15 is [0.00000000e+00 6.85759970e-02 1.30439298e-01 1.79534291e-01
 2.11055217e-01 2.21916588e-01 2.11055217e-01 1.79534291e-01
 1.30439298e-01 6.85759970e-02 1.22464680e-16]
```

```
The values of u at t= 0.155 is [0.00000000e+00 6.52196488e-02 1.24055144e-01 1.70747257e-01
 2.00725439e-01 2.11055217e-01 2.00725439e-01 1.70747257e-01
 1.24055144e-01 6.52196488e-02 1.22464680e-16]
The values of u at t= 0.16 is [0.00000000e+00 6.20275720e-02 1.17983453e-01 1.62390292e-01
 1.90901237e-01 2.00725439e-01 1.90901237e-01 1.62390292e-01
 1.17983453e-01 6.20275720e-02 1.22464680e-16]
The values of u at t= 0.165 is [0.00000000e+00 5.89917265e-02 1.12208932e-01 1.54442345e-01
 1.81557866e-01 1.90901237e-01 1.81557866e-01 1.54442345e-01
 1.12208932e-01 5.89917265e-02 1.22464680e-16]
The values of u at t= 0.17 is [0.00000000e+00 5.61044659e-02 1.06717036e-01 1.46883399e-01
 1.72671791e-01 1.81557866e-01 1.72671791e-01 1.46883399e-01
 1.06717036e-01 5.61044659e-02 1.22464680e-16]
The values of u at t= 0.17500000000000002 is [0.00000000e+00 5.33585179e-02 1.01493932e-01 1.39694414e-01
 1.64220632e-01 1.72671791e-01 1.64220632e-01 1.39694414e-01
 1.01493932e-01 5.33585179e-02 1.22464680e-16]
The values of u at t= 0.18 is [0.00000000e+00 5.07469662e-02 9.65264657e-02 1.32857282e-01
 1.56183102e-01 1.64220632e-01 1.56183102e-01 1.32857282e-01
 9.65264657e-02 5.07469662e-02 1.22464680e-16]
The values of u at t= 0.185 is [0.00000000e+00 4.82632329e-02 9.18021242e-02 1.26354784e-01
 1.48538957e-01 1.56183102e-01 1.48538957e-01 1.26354784e-01
 9.18021242e-02 4.82632329e-02 1.22464680e-16]
The values of u at t= 0.19 is [0.00000000e+00 4.59010621e-02 8.73090084e-02 1.20170541e-01
 1.41268943e-01 1.48538957e-01 1.41268943e-01 1.20170541e-01
 8.73090084e-02 4.59010621e-02 1.22464680e-16]
The values of u at t= 0.195 is [0.00000000e+00 4.36545042e-02 8.30358014e-02 1.14288976e-01
 1.34354749e-01 1.41268943e-01 1.34354749e-01 1.14288976e-01
 8.30358014e-02 4.36545042e-02 1.22464680e-16]
The values of u at t= 0.2 is [0.00000000e+00 4.15179007e-02 7.89717400e-02 1.08695275e-01
 1.27778959e-01 1.34354749e-01 1.27778959e-01 1.08695275e-01
 7.89717400e-02 4.15179007e-02 1.22464680e-16]
The values of u at t= 0.20500000000000002 is [0.00000000e+00 3.94858700e-02 7.51065879e-02 1.03375350e-01
 1.21525012e-01 1.27778959e-01 1.21525012e-01 1.03375350e-01
 7.51065879e-02 3.94858700e-02 1.22464680e-16]
The values of u at t= 0.21 is [0.00000000e+00 3.75532940e-02 7.14306099e-02 9.83158000e-02
 1.15577155e-01 1.21525012e-01 1.15577155e-01 9.83158000e-02
 7.14306099e-02 3.75532940e-02 1.22464680e-16]
The values of u at t= 0.215 is [0.00000000e+00 3.57153049e-02 6.79345470e-02 9.35038823e-02
 1.09920406e-01 1.15577155e-01 1.09920406e-01 9.35038823e-02
 6.79345470e-02 3.57153049e-02 1.22464680e-16]
The values of u at t= 0.22 is [0.00000000e+00 3.39672735e-02 6.46095936e-02 8.89274765e-02
 1.04540518e-01 1.09920406e-01 1.04540518e-01 8.89274765e-02
 6.46095936e-02 3.39672735e-02 1.22464680e-16]
The values of u at t= 0.225 is [0.00000000e+00 3.23047968e-02 6.14473750e-02 8.45750560e-02
 9.94239413e-02 1.04540518e-01 9.94239413e-02 8.45750560e-02
```

```
 6.14473750e-02 3.23047968e-02 1.22464680e-16]
The values of u at t= 0.23 is [0.00000000e+00 3.07236875e-02 5.84399264e-02 8.04356581e-02
 9.45577872e-02 9.94239413e-02 9.45577872e-02 8.04356581e-02
 5.84399264e-02 3.07236875e-02 1.22464680e-16]
The values of u at t= 0.23500000000000001 is [0.00000000e+00 2.92199632e-02 5.55796728e-02 7.64988568e-02
 8.99297997e-02 9.45577872e-02 8.99297997e-02 7.64988568e-02
 5.55796728e-02 2.92199632e-02 1.22464680e-16]
The values of u at t= 0.24 is [0.00000000e+00 2.77898364e-02 5.28594100e-02 7.27547363e-02
 8.55283220e-02 8.99297997e-02 8.55283220e-02 7.27547363e-02
 5.28594100e-02 2.77898364e-02 1.22464680e-16]
The values of u at t= 0.245 is [0.00000000e+00 2.64297050e-02 5.02722863e-02 6.91938660e-02
 8.13422680e-02 8.55283220e-02 8.13422680e-02 6.91938660e-02
 5.02722863e-02 2.64297050e-02 1.22464680e-16]
The values of u at t= 0.25 is [0.00000000e+00 2.51361432e-02 4.78117855e-02 6.58072772e-02
 7.73610940e-02 8.13422680e-02 7.73610940e-02 6.58072772e-02
 4.78117855e-02 2.51361432e-02 1.22464680e-16]
The values of u at t= 0.255 is [0.00000000e+00 2.39058928e-02 4.54717102e-02 6.25864398e-02
 7.35747726e-02 7.73610940e-02 7.35747726e-02 6.25864398e-02
 4.54717102e-02 2.39058928e-02 1.22464680e-16]
The values of u at t= 0.26 is [0.00000000e+00 2.27358551e-02 4.32461663e-02 5.95232414e-02
 6.99737669e-02 7.35747726e-02 6.99737669e-02 5.95232414e-02
 4.32461663e-02 2.27358551e-02 1.22464680e-16]
The values of u at t= 0.265 is [0.00000000e+00 2.16230831e-02 4.11295482e-02 5.66099666e-02
 6.65490070e-02 6.99737669e-02 6.65490070e-02 5.66099666e-02
 4.11295482e-02 2.16230831e-02 1.22464680e-16]
The values of u at t= 0.27 is [0.00000000e+00 2.05647741e-02 3.91165249e-02 5.38392776e-02
 6.32918667e-02 6.65490070e-02 6.32918667e-02 5.38392776e-02
 3.91165249e-02 2.05647741e-02 1.22464680e-16]
The values of u at t= 0.275 is [0.00000000e+00 1.95582624e-02 3.72020259e-02 5.12041958e-02
 6.01941423e-02 6.32918667e-02 6.01941423e-02 5.12041958e-02
 3.72020259e-02 1.95582624e-02 1.22464680e-16]
The values of u at t= 0.28 is [0.00000000e+00 1.86010129e-02 3.53812291e-02 4.86980841e-02
 5.72480313e-02 6.01941423e-02 5.72480313e-02 4.86980841e-02
 3.53812291e-02 1.86010129e-02 1.22464680e-16]
The values of u at t= 0.28500000000000003 is [0.00000000e+00 1.76906146e-02 3.36495485e-02 4.63146302e-02
 5.44461132e-02 5.72480313e-02 5.44461132e-02 4.63146302e-02
 3.36495485e-02 1.76906146e-02 1.22464680e-16]
The values of u at t= 0.29 is [0.00000000e+00 1.68247743e-02 3.20026224e-02 4.40478308e-02
 5.17813307e-02 5.44461132e-02 5.17813307e-02 4.40478308e-02
 3.20026224e-02 1.68247743e-02 1.22464680e-16]
The values of u at t= 0.295 is [0.00000000e+00 1.60013112e-02 3.04363025e-02 4.18919765e-02
 4.92469720e-02 5.17813307e-02 4.92469720e-02 4.18919765e-02
 3.04363025e-02 1.60013112e-02 1.22464680e-16]
The values of u at t= 0.3 is [0.00000000e+00 1.52181513e-02 2.89466439e-02 3.98416373e-02
```

```
 4.68366536e-02 4.92469720e-02 4.68366536e-02 3.98416373e-02
 2.89466439e-02 1.52181513e-02 1.22464680e-16]
The values of u at t= 0.305 is [0.00000000e+00 1.44733219e-02 2.75298943e-02 3.78916488e-02
 4.45443046e-02 4.68366536e-02 4.45443046e-02 3.78916488e-02
 2.75298943e-02 1.44733219e-02 1.22464680e-16]
The values of u at t= 0.31 is [0.00000000e+00 1.37649471e-02 2.61824853e-02 3.60370995e-02
 4.23641512e-02 4.45443046e-02 4.23641512e-02 3.60370995e-02
 2.61824853e-02 1.37649471e-02 1.22464680e-16]
The values of u at t= 0.315 is [0.00000000e+00 1.30912427e-02 2.49010233e-02 3.42733183e-02
 4.02907021e-02 4.23641512e-02 4.02907021e-02 3.42733183e-02
 2.49010233e-02 1.30912427e-02 1.22464680e-16]
The values of u at t= 0.32 is [0.00000000e+00 1.24505116e-02 2.36822805e-02 3.25958627e-02
 3.83187347e-02 4.02907021e-02 3.83187347e-02 3.25958627e-02
 2.36822805e-02 1.24505116e-02 1.22464680e-16]
The values of u at t= 0.325 is [0.00000000e+00 1.18411402e-02 2.25231872e-02 3.10005076e-02
 3.64432824e-02 3.83187347e-02 3.64432824e-02 3.10005076e-02
 2.25231872e-02 1.18411402e-02 1.22464680e-16]
The values of u at t= 0.33 is [0.00000000e+00 1.12615936e-02 2.14208239e-02 2.94832348e-02
 3.46596212e-02 3.64432824e-02 3.46596212e-02 2.94832348e-02
 2.14208239e-02 1.12615936e-02 1.22464680e-16]
The values of u at t= 0.335 is [0.00000000e+00 1.07104120e-02 2.03724142e-02 2.80402225e-02
 3.29632586e-02 3.46596212e-02 3.29632586e-02 2.80402225e-02
 2.03724142e-02 1.07104120e-02 1.22464680e-16]
The values of u at t= 0.34 is [0.00000000e+00 1.01862071e-02 1.93753173e-02 2.66678364e-02
 3.13499219e-02 3.29632586e-02 3.13499219e-02 2.66678364e-02
 1.93753173e-02 1.01862071e-02 1.22464680e-16]
The values of u at t= 0.34500000000000003 is [0.00000000e+00 9.68765863e-03 1.84270217e-02 2.53626196e-02
 2.98155475e-02 3.13499219e-02 2.98155475e-02 2.53626196e-02
 1.84270217e-02 9.68765863e-03 1.22464680e-16]
The values of u at t= 0.35000000000000003 is [0.00000000e+00 9.21351086e-03 1.75251391e-02 2.41212846e-02
 2.83562707e-02 2.98155475e-02 2.83562707e-02 2.41212846e-02
 1.75251391e-02 9.21351086e-03 1.22464680e-16]
The values of u at t= 0.355 is [0.00000000e+00 8.76256954e-03 1.66673977e-02 2.29407049e-02
 2.69684160e-02 2.83562707e-02 2.69684160e-02 2.29407049e-02
 1.66673977e-02 8.76256954e-03 1.22464680e-16]
The values of u at t= 0.36 is [0.00000000e+00 8.33369886e-03 1.58516372e-02 2.18179069e-02
 2.56484878e-02 2.69684160e-02 2.56484878e-02 2.18179069e-02
 1.58516372e-02 8.33369886e-03 1.22464680e-16]
The values of u at t= 0.365 is [0.00000000e+00 7.92581861e-03 1.50758029e-02 2.07500625e-02
 2.43931615e-02 2.56484878e-02 2.43931615e-02 2.07500625e-02
 1.50758029e-02 7.92581861e-03 1.22464680e-16]
The values of u at t= 0.37 is [0.00000000e+00 7.53790144e-03 1.43379406e-02 1.97344822e-02
 2.31992752e-02 2.43931615e-02 2.31992752e-02 1.97344822e-02
 1.43379406e-02 7.53790144e-03 1.22464680e-16]
```

```
The values of u at t= 0.375 is [0.00000000e+00 7.16897028e-03 1.36361918e-02 1.87686079e-02
 2.20638218e-02 2.31992752e-02 2.20638218e-02 1.87686079e-02
 1.36361918e-02 7.16897028e-03 1.22464680e-16]
The values of u at t= 0.38 is [0.00000000e+00 6.81809590e-03 1.29687891e-02 1.78500068e-02
 2.09839415e-02 2.20638218e-02 2.09839415e-02 1.78500068e-02
 1.29687891e-02 6.81809590e-03 1.22464680e-16]
The values of u at t= 0.385 is [0.00000000e+00 6.48439453e-03 1.23340514e-02 1.69763653e-02
 1.99569143e-02 2.09839415e-02 1.99569143e-02 1.69763653e-02
 1.23340514e-02 6.48439453e-03 1.22464680e-16]
The values of u at t= 0.39 is [0.00000000e+00 6.16702568e-03 1.17303799e-02 1.61454828e-02
 1.89801534e-02 1.99569143e-02 1.89801534e-02 1.61454828e-02
 1.17303799e-02 6.16702568e-03 1.22464680e-16]
The values of u at t= 0.395 is [0.00000000e+00 5.86518996e-03 1.11562543e-02 1.53552667e-02
 1.80511986e-02 1.89801534e-02 1.80511986e-02 1.53552667e-02
 1.11562543e-02 5.86518996e-03 1.22464680e-16]
The values of u at t= 0.4 is [0.00000000e+00 5.57812713e-03 1.06102283e-02 1.46037264e-02
 1.71677100e-02 1.80511986e-02 1.71677100e-02 1.46037264e-02
 1.06102283e-02 5.57812713e-03 1.22464680e-16]
The values of u at t= 0.405 is [0.00000000e+00 5.30511415e-03 1.00909268e-02 1.38889692e-02
 1.63274625e-02 1.71677100e-02 1.63274625e-02 1.38889692e-02
 1.00909268e-02 5.30511415e-03 1.22464680e-16]
The values of u at t= 0.41000000000000003 is [0.00000000e+00 5.04546338e-03 9.59704166e-03 1.32091946e-02
 1.55283396e-02 1.63274625e-02 1.55283396e-02 1.32091946e-02
 9.59704166e-03 5.04546338e-03 1.22464680e-16]
The values of u at t= 0.41500000000000004 is [0.00000000e+00 4.79852083e-03 9.12732901e-03 1.25626906e-02
 1.47683286e-02 1.55283396e-02 1.47683286e-02 1.25626906e-02
 9.12732901e-03 4.79852083e-03 1.22464680e-16]
The values of u at t= 0.42 is [0.00000000e+00 4.56366450e-03 8.68060573e-03 1.19478288e-02
 1.40455151e-02 1.47683286e-02 1.40455151e-02 1.19478288e-02
 8.68060573e-03 4.56366450e-03 1.22464680e-16]
The values of u at t= 0.425 is [0.00000000e+00 4.34030286e-03 8.25574664e-03 1.13630604e-02
 1.33580787e-02 1.40455151e-02 1.33580787e-02 1.13630604e-02
 8.25574664e-03 4.34030286e-03 1.22464680e-16]
The values of u at t= 0.43 is [0.00000000e+00 4.12787332e-03 7.85168164e-03 1.08069127e-02
 1.27042878e-02 1.33580787e-02 1.27042878e-02 1.08069127e-02
 7.85168164e-03 4.12787332e-03 1.22464680e-16]
The values of u at t= 0.435 is [0.00000000e+00 3.92584082e-03 7.46739299e-03 1.02779847e-02
 1.20824957e-02 1.27042878e-02 1.20824957e-02 1.02779847e-02
 7.46739299e-03 3.92584082e-03 1.22464680e-16]
The values of u at t= 0.44 is [0.00000000e+00 3.73369649e-03 7.10191276e-03 9.77494433e-03
 1.14911362e-02 1.20824957e-02 1.14911362e-02 9.77494433e-03
 7.10191276e-03 3.73369649e-03 1.22464680e-16]
The values of u at t= 0.445 is [0.00000000e+00 3.55095638e-03 6.75432041e-03 9.29652450e-03
 1.09287200e-02 1.14911362e-02 1.09287200e-02 9.29652450e-03
```

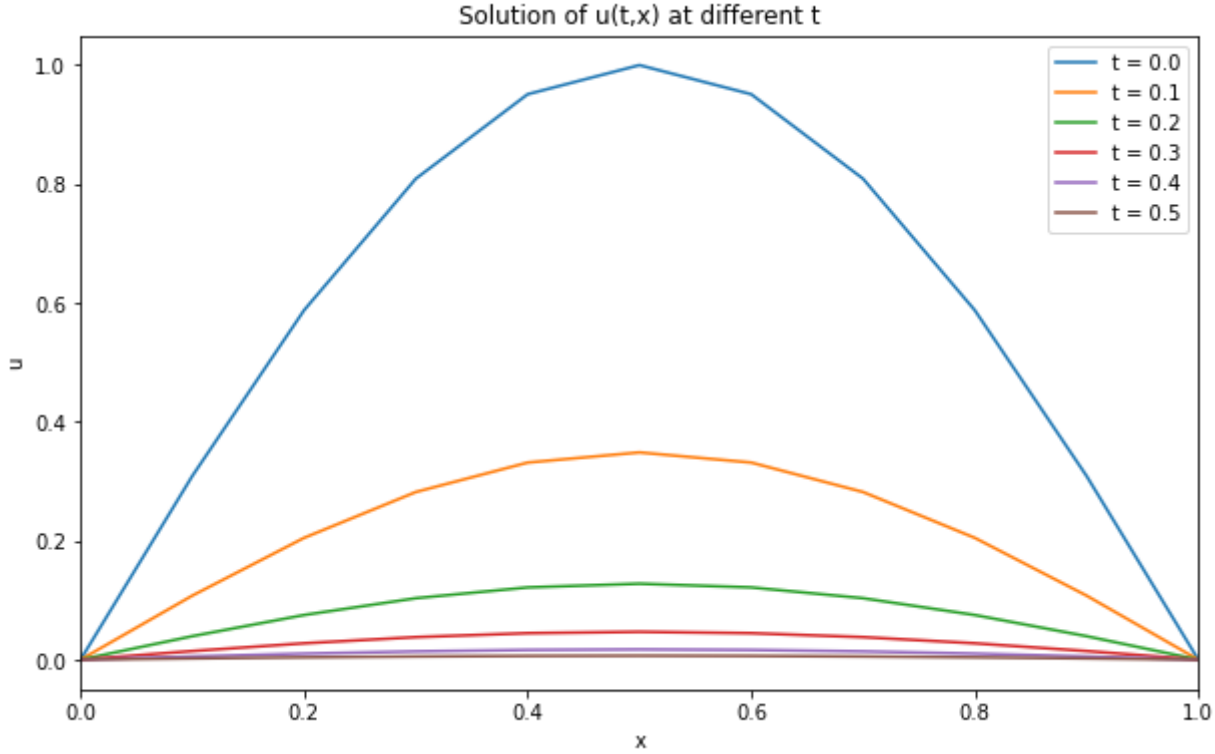
```
 6.75432041e-03 3.55095638e-03 1.22464680e-16]
The values of u at t= 0.45 is [0.00000000e+00 3.37716021e-03 6.42374044e-03 8.84152020e-03
 1.03938304e-02 1.09287200e-02 1.03938304e-02 8.84152020e-03
 6.42374044e-03 3.37716021e-03 1.22464680e-16]
The values of u at t= 0.455 is [0.00000000e+00 3.21187022e-03 6.10934020e-03 8.40878540e-03
 9.88512010e-03 1.03938304e-02 9.88512010e-03 8.40878540e-03
 6.10934020e-03 3.21187022e-03 1.22464680e-16]
The values of u at t= 0.46 is [0.00000000e+00 3.05467010e-03 5.81032781e-03 7.99723015e-03
 9.40130788e-03 9.88512010e-03 9.40130788e-03 7.99723015e-03
 5.81032781e-03 3.05467010e-03 1.22464680e-16]
The values of u at t= 0.465 is [0.00000000e+00 2.90516391e-03 5.52595013e-03 7.60581785e-03
 8.94117513e-03 9.40130788e-03 8.94117513e-03 7.60581785e-03
 5.52595013e-03 2.90516391e-03 1.22464680e-16]
The values of u at t= 0.47000000000000003 is [0.00000000e+00 2.76297506e-03 5.25549088e-03 7.23356263e-03
 8.50356287e-03 8.94117513e-03 8.50356287e-03 7.23356263e-03
 5.25549088e-03 2.76297506e-03 1.22464680e-16]
The values of u at t= 0.47500000000000003 is [0.00000000e+00 2.62774544e-03 4.99826884e-03 6.87952687e-03
 8.08736888e-03 8.50356287e-03 8.08736888e-03 6.87952687e-03
 4.99826884e-03 2.62774544e-03 1.22464680e-16]
The values of u at t= 0.48 is [0.00000000e+00 2.49913442e-03 4.75363616e-03 6.54281886e-03
 7.69154487e-03 8.08736888e-03 7.69154487e-03 6.54281886e-03
 4.75363616e-03 2.49913442e-03 1.22464680e-16]
The values of u at t= 0.485 is [0.00000000e+00 2.37681808e-03 4.52097664e-03 6.22259051e-03
 7.31509387e-03 7.69154487e-03 7.31509387e-03 6.22259051e-03
 4.52097664e-03 2.37681808e-03 1.22464680e-16]
The values of u at t= 0.49 is [0.00000000e+00 2.26048832e-03 4.29970429e-03 5.91803525e-03
 6.95706769e-03 7.31509387e-03 6.95706769e-03 5.91803525e-03
 4.29970429e-03 2.26048832e-03 1.22464680e-16]
The values of u at t= 0.495 is [0.00000000e+00 2.14985215e-03 4.08926179e-03 5.62838599e-03
 6.61656456e-03 6.95706769e-03 6.61656456e-03 5.62838599e-03
 4.08926179e-03 2.14985215e-03 1.22464680e-16]
The values of u at t= 0.5 is [0.00000000e+00 2.04463089e-03 3.88911907e-03 5.35291317e-03
 6.29272684e-03 6.61656456e-03 6.29272684e-03 5.35291317e-03
 3.88911907e-03 2.04463089e-03 1.22464680e-16]
```

In [172…

```
plt.figure(figsize=(10,6))
for t in [0,21,41,61,81,-1]:
    plt.plot(x,ans[:,t])
plt.xlim((0.0,1.0))
plt.title('Solution of u(t,x) at different t')
plt.xlabel('x')
plt.ylabel('u')
plt.legend(['t = 0.0','t = 0.1','t = 0.2','t = 0.3','t = 0.4','t = 0.5'])
```

Out[172…

```
<matplotlib.legend.Legend at 0x24387096550>
```



Question 8

With this choice of f(x), plot the solution after 500 iterations for ┌t = ♂=n2 where ♂ 2 f0:48; 0:5; 0:52g and n = 100. What do you observe?

In [208…

```
a = 0.5
b = -0.5
c = 1

def u_int(x):
    f = 0.5 - x
    return f
```

In [209…

```
#Explicit Euler
```

```
def ExplicitEuler(n):
    dx = 1/n
    x = np.linspace(0.0,n,n+1)*dx

    u0 = np.zeros(len(x))
    u0 = u_int(x) # initial condition

    u = np.zeros(len(x))

    def f(u):
        dudt = np.zeros(len(x))
        dudt[0] = 0.0
        dudt[-1] = 0.0
        for i in range(1,len(x)-1):
            dudt[i] = c*((u[i-1]-2*u[i]+u[i+1])/(dx**2))
        return dudt

    def phi(u):
        phi = f(u)
        return phi

    dt = alpha/(n**2) # alpha is maxmimum t
    ts = np.linspace(0, alpha, (n**2)+1) # alpha is maxmimum t

    u_euler = np.zeros(((n+1),((n**2)+1))) # create an empty vector for our solution
    #each column of u_euler is the position ui at every xi at a time step, each row represent the change of position ui at xi over ti

    u_euler[:,0] = u0 # set initial condition
    # Run the one-step method
    for k in range(n**2):
        u_euler[:,k+1] = u_euler[:,k] + dt * phi(u_euler[:,k])

    return x, u_euler
```

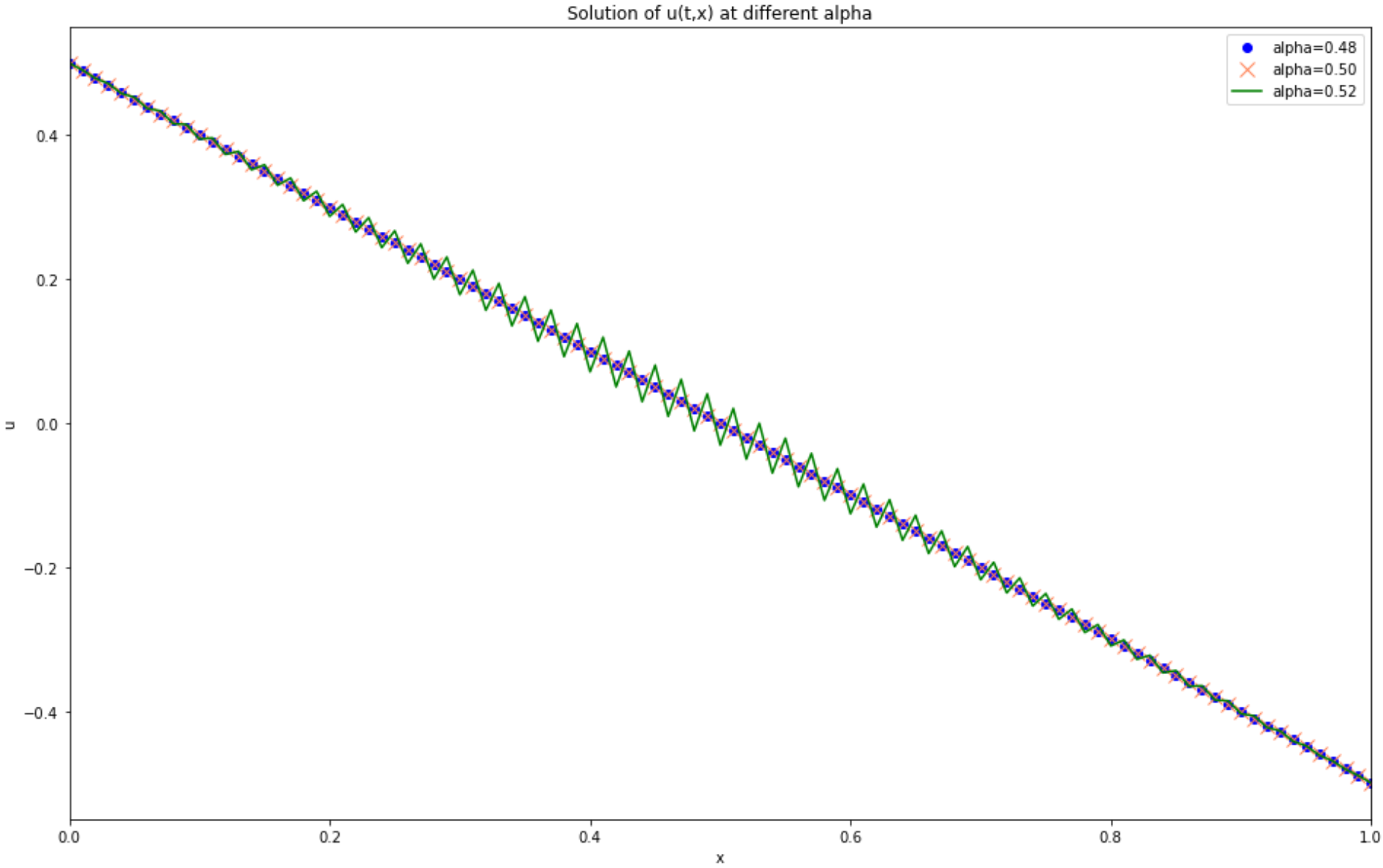
In [221…

```
alpha = 0.48
x1,ans1 = ExplicitEuler(100)
alpha = 0.5
x2,ans2 = ExplicitEuler(100)
alpha = 0.52
x3,ans3 = ExplicitEuler(100)

plt.figure(figsize=(16,10))
```

```
plt.plot(x1,ans1[:,501],'o',color='blue',label='alpha=0.48')
plt.plot(x2,ans2[:,501],'x',color='coral',markersize=10,label='alpha=0.50')
plt.plot(x3,ans3[:,501],'g-',label='alpha=0.52')
plt.xlim((0.0,1.0))
plt.title('Solution of u(t,x) at different alpha')
plt.xlabel('x')
plt.ylabel('u')
plt.legend()
```

```
C:\Users\huang\AppData\Local\Temp/ipykernel_46052/186763276.py:16: RuntimeWarning: overflow encountered in double_scalars
  dudt[i] = c*((u[i-1]-2*u[i]+u[i+1])/(dx**2))
C:\Users\huang\AppData\Local\Temp/ipykernel_46052/186763276.py:32: RuntimeWarning: invalid value encountered in add
  u_euler[:,k+1] = u_euler[:,k] + dt * phi(u_euler[:,k])
```

Out[221… <matplotlib.legend.Legend at 0x24419ba2130>



when f(x)=0.5-x, with initial condition of 0.5 and -0.5, the temperature derived from this differential equation does not change with time at any position. After 500 iterations, the alpha=0.48 and alpha=0.5 cases are returning correct values. However, alpha=0.52 case is having fluctuations of temperature (u) around the temperature linear line.

In [ ]:

In [ ]:

In [206…

```
plt.figure(figsize=(16,6))
plt.plot(x1,ans1[:,501],'o',label='alpha=0.48')
plt.plot(x2,ans2[:,501],'x',label='alpha=0.50')
plt.plot(x3,ans3[:,501],'-',label='alpha=0.52')
plt.xlim((0.0,1.0))
plt.title('Solution of u(t,x) at different alpha')
plt.xlabel('x')
plt.ylabel('u')
plt.legend()
```

Out[206…

```
<matplotlib.legend.Legend at 0x24395140b20>
```