Math 478 HW 6

Lucas Bouck

11/17/16

1 Problem 3.7.10

We must show that every column of the matrix F_n is an eigenvector of the matrix L. Let v_j be the j^{th} column of F_n. Then,

$$v_j = \begin{pmatrix} \omega_n^0 \\ \omega_n^{j-1} \\ \omega_n^{2(j-1)} \\ . \\ . \\ . \\ \omega_n^{(n-1)(j-1)} \end{pmatrix}$$

The k^{th} component of v_j is $\omega_n^{(k-1)(j-1)}$. If $0 < k < n-1$ the k^{th} component of the vector Lv_j is

$$\begin{aligned} v_{kj} &= \omega_n^{(k-2)(j-1)} - 2\omega_n^{(k-1)(j-1)} + \omega_n^{k(j-1)} \\ &= (-2 + \omega_n^{j-1} + \omega_n^{1-j})\omega_n^{(k-1)(j-1)} \end{aligned}$$

The first and last components of Lv_j are the following

$$\begin{aligned} v_{1j} &= -2\omega_n^0 + \omega_n^{j-1} + \omega_n^{(n-1)(j-1)} = \omega_n^0(-2 + \omega_n^{j-1} + \omega_n^{(n-1)(j-1)}) = \omega_n^0(-2 + \omega_n^{j-1} + \omega_n^{1-j}) \\ v_{nj} &= -2\omega_n^{(n-1)(j-1)} + \omega_n^{n(j-1)} + \omega_n^{(n-2)(j-1)} = \omega_n^{(n-1)(j-1)}(-2 + \omega_n^{j-1} + \omega_n^{(-1)(j-1)}) \\ &= \omega_n^{(n-1)(j-1)}(-2 + \omega_n^{j-1} + \omega_n^{1-j}) \end{aligned}$$

We have now established that given the j^{th} column of F_n, v_j, that

$$Lv_j = (-2 + \omega_n^{j-1} + \omega_n^{1-j})v_j$$

The above constant $-2 + \omega_n^{j-1} + \omega_n^{1-j}$ is just a constant depending on which column we're using, so v_j is an eigenvector of L. Therefore, the columns of F_n are eigenvectors for L.

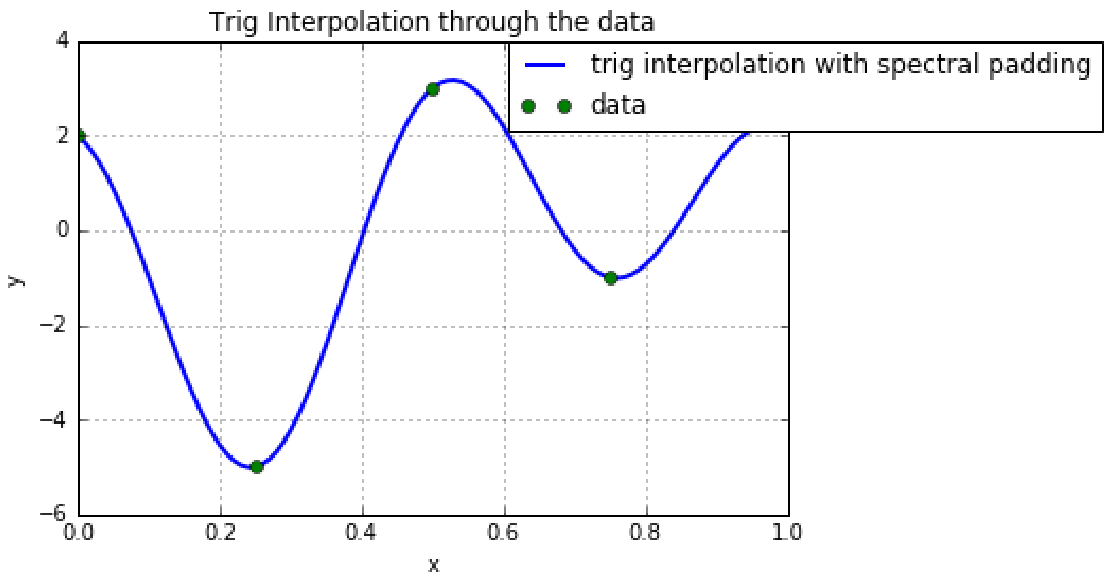
2 Problem 3.7.13

Below is the code and output of that code to solve 3.7.13.

In [1]:
```
#import the needed modules/libraries
import numpy as np
import scipy.fftpack as fftpack
import matplotlib.pyplot as plt


#data points that we have on 0,.25,.5,.75
xvals1=np.array([0,.25,.5,.75])
yvals1=np.array([2,-5,3,-1])


#fft on the yvals1
fftvals=fftpack.fft(yvals1)


#computing the coeffecents we need for the trig interpolation
a_0=.25*np.real(fftvals[0])
a_2=.25*np.real(fftvals[2])
a_1=.5*np.real(fftvals[1])
b_1=-.5*np.imag(fftvals[1])


#checking whether phi agrees with data
def phi(x):
    return a_0+a_1*np.cos(2*np.pi*x)\
+a_2*np.cos(4*np.pi*x)+b_1*np.sin(2*np.pi*x)
print('phi(x) agree with data? '
      +str(np.allclose(phi(xvals1),yvals1,atol=1e-16)))


#spectral padding part
#evenly spaced points on [0,1)
x=np.arange(0,100)/100


#creating Y
Y=np.zeros(100,dtype=np.complex128)
Y[2]=fftvals[2]/2
Y[-2]=fftvals[2]/2
for i in range(0,2):
    Y[i]=fftvals[i]
for i in range (-1,1):
    Y[i]=fftvals[i]
Y=25*Y
X=fftpack.ifft(Y)
```

```
#checking whether the spectral padding worked by comparing
#X to phi(x) at the mesh points
print('X agree with phi(x)? '
      +str(np.allclose(X,phi(x),atol=1e-16)))


#checking whether X agrees with original data points
print('X agree with data? '
      +str(np.allclose(np.array([X[0],X[25],X[50],X[75]]),
                       yvals1,atol=1e-16)))


#plotting the data and X
%matplotlib inline
plt.plot(x,np.real(X),label='trig interpolation with spectral padding',
         linewidth=2)
plt.plot(xvals1,yvals1,'o',label='data')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Trig Interpolation through the data')
plt.grid()
plt.legend(bbox_to_anchor=(1.46, 1.025))
plt.show()
```

```
phi(x) agree with data? True
X agree with phi(x)? True
X agree with data? True
```


Trig Interpolation through the data
trig interpolation with spectral padding
data
4
2
0
−2
−4
−6
y
0.0
0.2
0.4
0.6
0.8
1.0
x