In [1]:
```
%matplotlib notebook
%pylab
from scipy.fft import fft, ifft
```

```
Using matplotlib backend: nbAgg
Populating the interactive namespace from numpy and matplotlib
```

1. (Spectral differentiation)

a)

In [2]:
```
#function u(t)
def u(t):
        return (exp(cos(5*(t-0.1))))

def uprime(uj,N):
    '''
    Arguments
    ---------
    N: Total number of iterations
    uj: Samples of u(t) at j=0,1,...,N

    Return
    ------
    uprim: real part of the inverse fast fourier tansform of dk
    '''
    #apply fft
    ucap=fft(uj)
    n=int((N-1)/2)
    #computing dk, j is complex
    dk1=[]
    dk2=[]
    for k in range(0,n+1):
        dk_1=j*k*ucap[k]
        dk1.append(dk_1)
    for k in range(-n,0):
        dk_2=j*k*ucap[k]
        dk2.append(dk_2)

    dk=dk1 + dk2
    uprim=real(ifft(dk))
    return uprim
```

b)

Using N=63

In [3]:
```
N=63
tj6=zeros(N)
for j in range(N): #for N odd
    tj6[j]=(2*pi*j)/N

uj6=u(tj6)
up63=uprime(uj6,N)
print('Derivatives of u for N=63; \n',up63)
```

```
Derivatives of u for N=63;
 [ 0.00000000e+00 -3.60866539e-15  6.31593543e-14 -3.60910596e-15
  2.70683129e-14 -1.44363285e-14 -2.34591887e-14 -2.88733064e-14
 -2.52638371e-14 -8.43104723e-15 -9.11854707e-15  1.04925271e-14
  5.78055075e-15 -2.58122714e-15  8.06537512e-15  4.02579294e-14
  1.49068716e-14 -4.16863997e-14  1.33009658e-14  4.26662212e-15
 -5.61338824e-14 -2.34037662e-14 -3.91931414e-14 -3.92958928e-17
  2.03578888e-14 -1.87652639e-14  3.62120408e-14  1.44187154e-14
  8.98496269e-15  1.50157260e-14  1.44863539e-14  2.93853640e-14
  4.29063870e-15 -6.83658983e-14  1.72336611e-14  2.29386014e-14
 -1.44187154e-14 -2.48069734e-15  5.70312639e-14  1.64437694e-14
  4.36828371e-15 -2.22923740e-14  2.75041507e-14 -4.38495255e-15
 -6.54460285e-14 -1.33009658e-14 -2.65953773e-15 -3.81162005e-14
 -1.90054935e-14 -9.58220386e-15  2.60530235e-14  3.53665648e-14
 -1.72395057e-14  1.42086390e-14  8.43104723e-15  4.61894668e-15
 -5.14487884e-14  9.30769158e-15 -9.46526093e-15 -1.64037132e-15
 -3.16614468e-14  3.71192847e-14  5.90363166e-14]
```

Using N=127

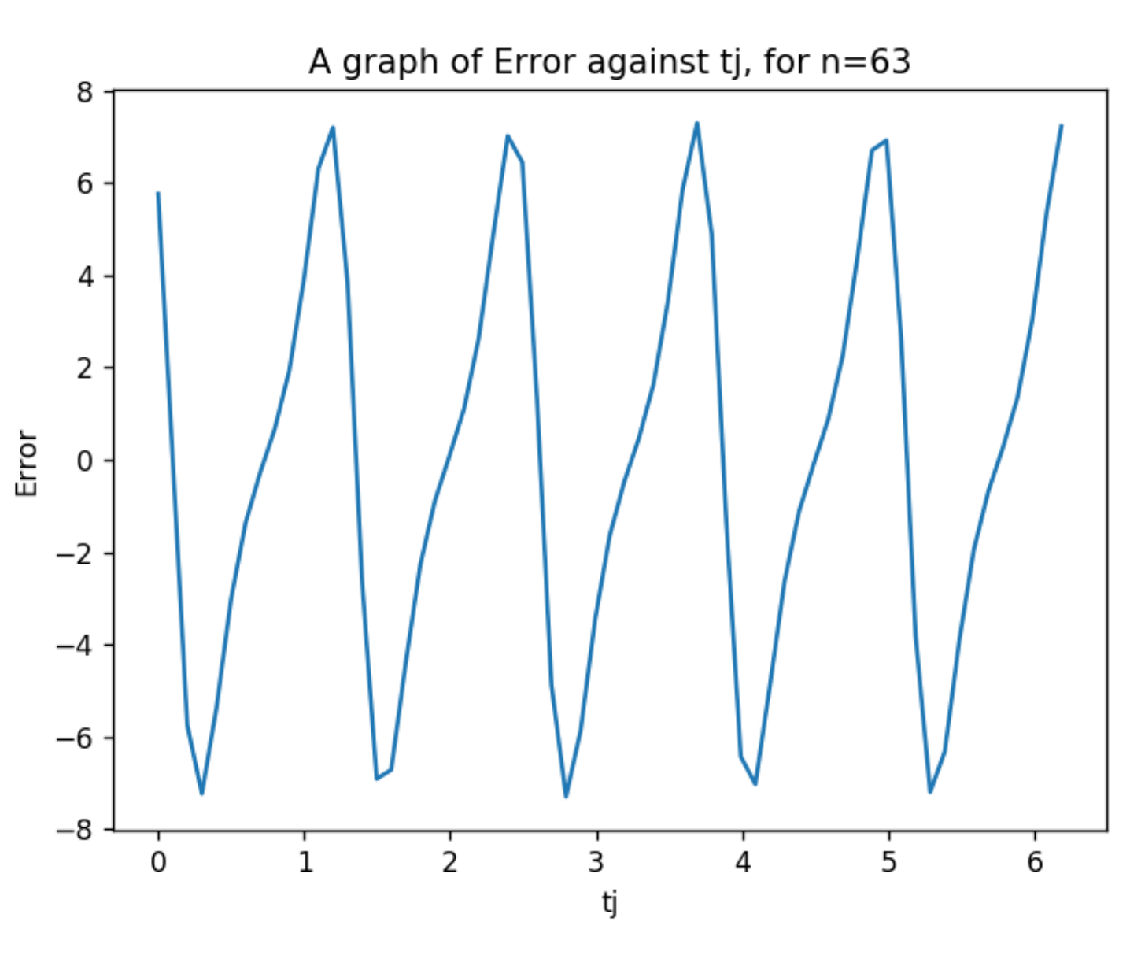
In [4]:
```
N=127
tj1=zeros(N)
for j in range(N): #for N odd
    tj1[j]=(2*pi*j)/N

uj1=u(tj1)
up127=uprime(uj1,N)
print('Derivatives of u for N=127; \n',up127)
```

```
Derivatives of u for N=127;
 [-2.36325395e-13 -1.89776454e-13 -2.80524510e-13 -1.28904761e-13
  8.59365073e-14 -1.71873015e-13 -1.71873015e-13  0.00000000e+00
  8.41461634e-14 -1.14582010e-13 -8.59365073e-14 -9.30978829e-14
 -5.72910048e-14 -5.01296292e-14  1.79034390e-15 -1.60235779e-13
 -4.78916994e-14  4.29682536e-14 -1.43227512e-14  1.14582010e-13
 -5.72910048e-14  2.00518517e-13  0.00000000e+00  1.71873015e-13
 -4.29682536e-14 -8.59365073e-14 -8.59365073e-14 -2.43486771e-13
  0.00000000e+00 -1.14582010e-13  2.86455024e-13 -2.86455024e-14
  2.00518517e-13  3.15100527e-13 -5.72910048e-14 -2.86455024e-14
  0.00000000e+00  1.00259258e-13 -2.93616400e-13 -1.64711639e-13
  9.26502969e-14 -2.04099205e-13 -1.00259258e-13  2.22002644e-13
 -1.43227512e-13 -5.72910048e-14  0.00000000e+00  2.00518517e-13
  0.00000000e+00  5.72910048e-14 -5.72910048e-14 -4.29682536e-14
  1.71873015e-13  8.59365073e-14  1.93357141e-13 -5.72910048e-14
  2.29164019e-13  1.71873015e-13 -2.72132273e-13  8.59365073e-14
  0.00000000e+00  2.86455024e-14 -1.00259258e-13 -1.00259258e-13
 -2.86455024e-14 -1.59340607e-13  1.12791666e-13  7.87751316e-14
 -1.00259258e-13 -4.47585975e-14  1.00259258e-13 -8.59365073e-14
  0.00000000e+00  7.16137560e-14 -1.14582010e-13 -2.29164019e-13
  1.07420634e-13  2.29164019e-13  2.50648146e-14  8.59365073e-14
  1.14582010e-13  5.72910048e-14  5.72910048e-14 -1.71873015e-13
  2.00518517e-13  2.86455024e-14 -2.29164019e-13  5.72910048e-14
 -1.89776454e-13 -1.71873015e-13 -8.59365073e-14 -2.42591599e-13
 -8.95171951e-14 -7.16137560e-14  1.21743385e-13  3.15100527e-13
  1.14582010e-13  0.00000000e+00  5.72910048e-14 -1.14582010e-13
  1.14582010e-13  5.72910048e-14  2.86455024e-14 -2.14841268e-14
  2.86455024e-14  2.57809522e-13 -2.14841268e-13  2.29164019e-13
  5.72910048e-14  1.43227512e-14 -5.72910048e-14  2.86455024e-14
 -2.86455024e-14 -2.50648146e-14 -1.57550263e-13 -7.87751316e-14
 -1.23981315e-13 -1.50388888e-13  3.58068780e-14 -1.03392360e-13
  4.29682536e-14  2.00518517e-13  0.00000000e+00  5.72910048e-14
  1.14582010e-13  2.29164019e-13  1.71873015e-13]
```

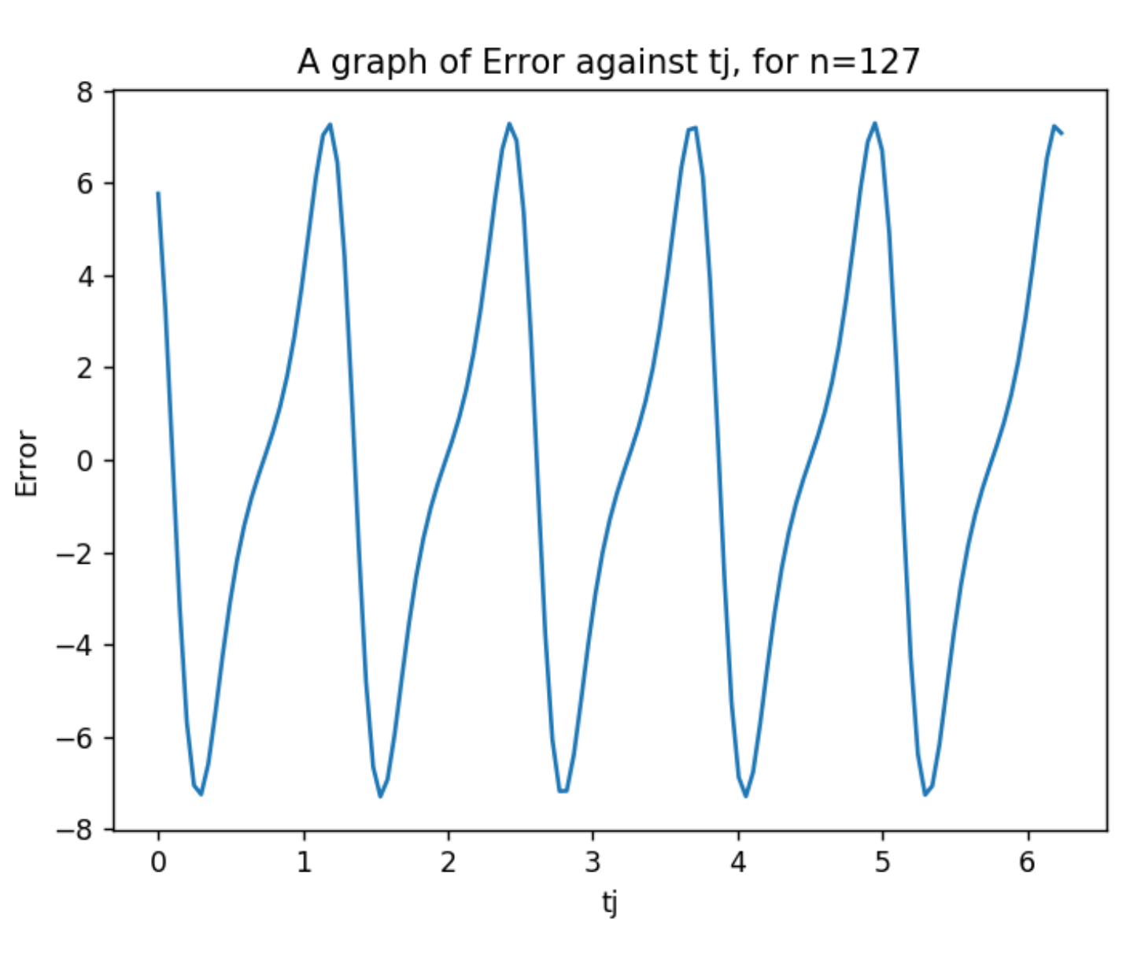
Correct Derivatives

In [5]:
```
def up(t):
    return (-5*sin(5*(t-0.1))*u(t))

#for N=63
uexact=up(tj6)
Error63=uexact-up63
#for N=127
uexact=up(tj1)
Error127=uexact-up127
```

In [6]:
```
figure(1)
plot(tj6,Error63)
title('A graph of Error against tj, for n=63')
ylabel('Error')
xlabel('tj')
show()
```



In [7]:
```
figure(2)
plot(tj1,Error127)
title('A graph of Error against tj, for n=127')
ylabel('Error')
xlabel('tj')
show()
```



c)

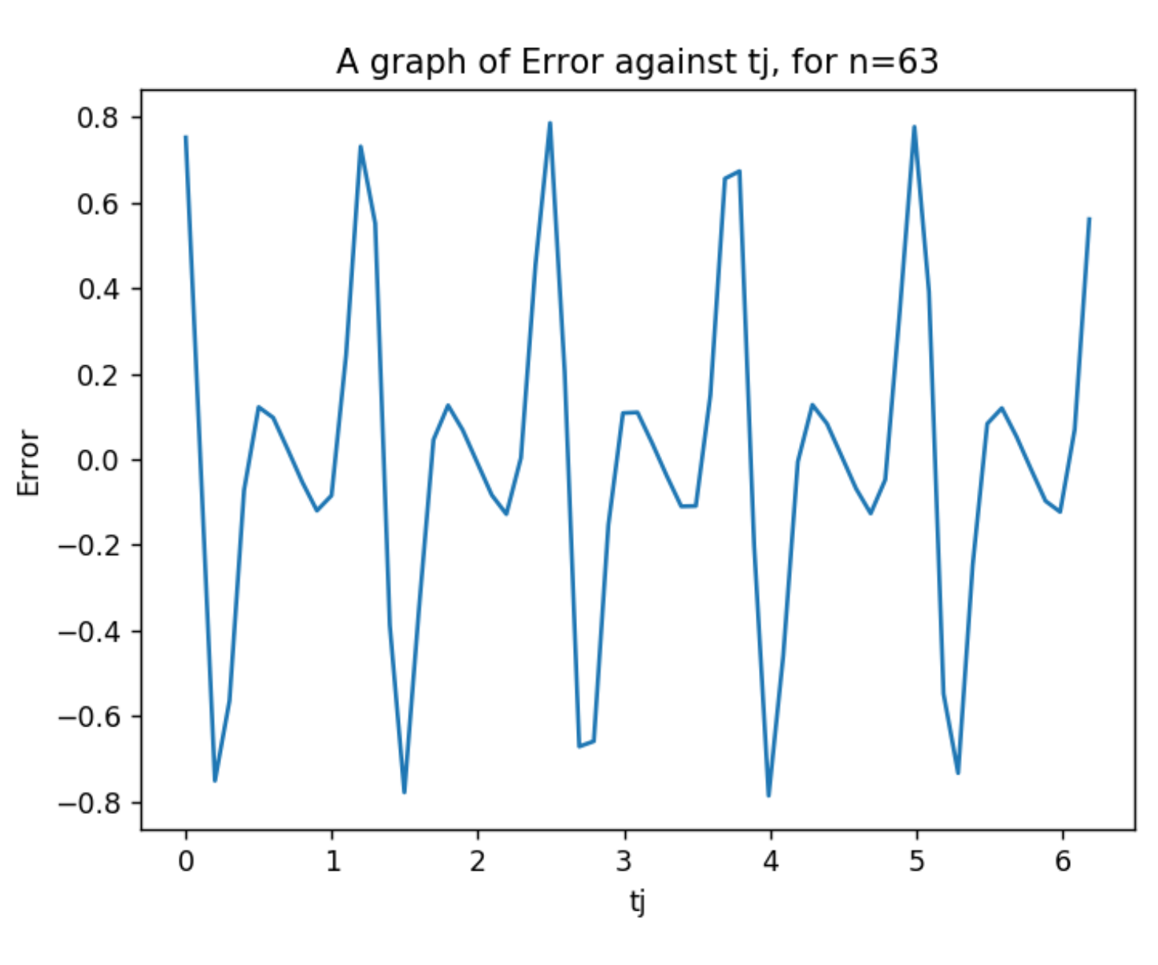
Computing the approximte Derivative

In [8]:
```
def uapp(uj,N):
    h=(2*pi)/N
    #computing N by N matrix DN
    DN=zeros((N,N))
    DN[N-1,N-2]=-1
    DN[N-1,0]=1
    for j in range(0,N-1):
        DN[j,j]=0
        DN[j,j-1]=-1
        DN[j,j+1]=1
    DN1=(1/(2*h))*DN
    #computing DN*uj'
    up=matmul(DN1,transpose(uj))
    return(up)
```

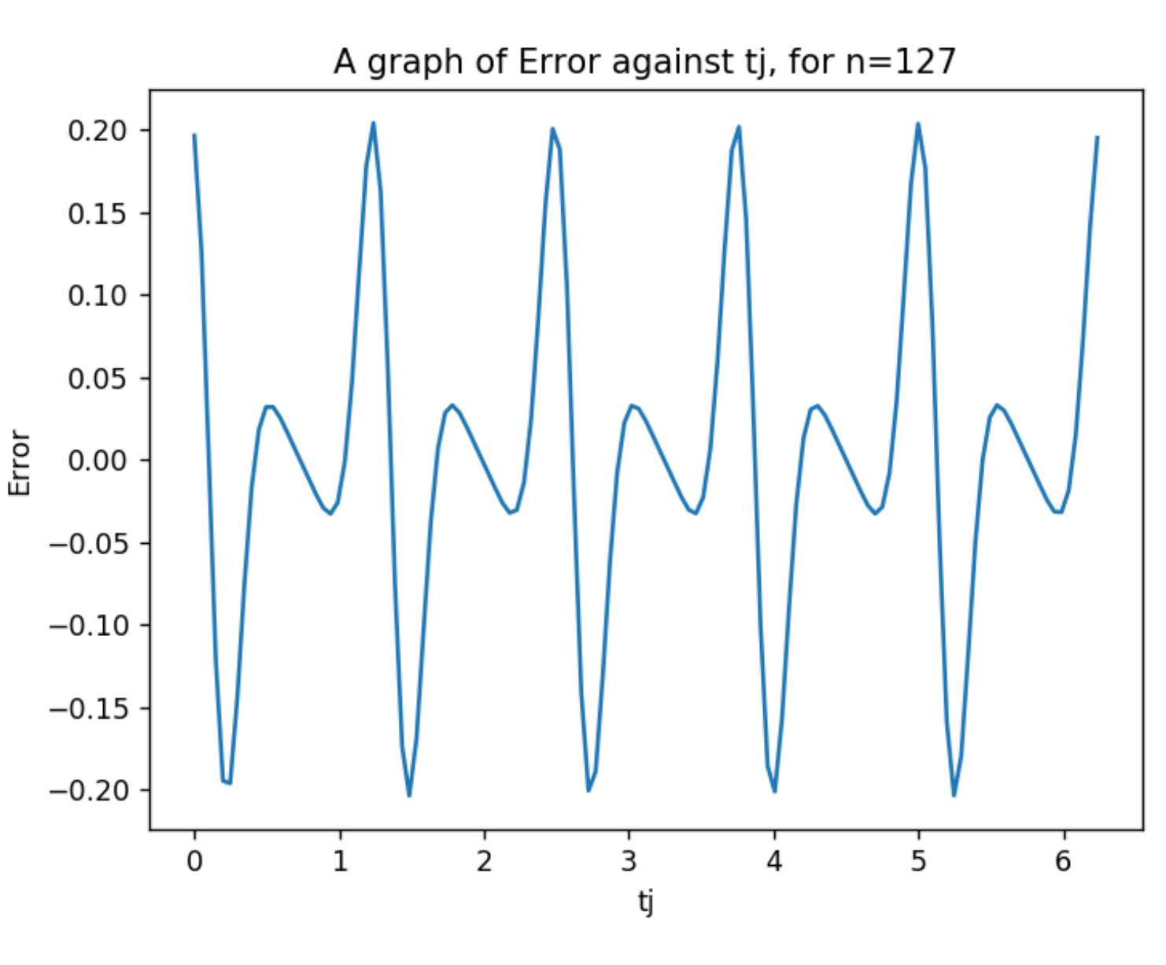
In [9]:
```
#for N=63
N=63
uapp6=uapp(uj6,N=63)
print('Approximate derivative of uj, for N=63; \n',uapp6,'\n')

#for N=127
N=127
uapp1=uapp(uj1,N=127)
print('Approximate derivative of uj, for N=127; \n',uapp1)

#error
#for N=63
uexact6=up(tj6)
Error6=uexact6-uapp6

#for N=127
uexact1=up(tj1)
Error1=uexact1-uapp1
```

```
Approximate derivative of uj, for N=63;
 [ 5.01250784  0.01540083 -4.99317052 -6.66339618 -5.30770166 -3.14863042
 -1.46149004 -0.30287277  0.71881508  2.0499095   3.98021027  6.06030942
  6.46601939  3.29546465 -2.23621873 -6.13047339 -6.36116917 -4.43375793
 -2.3955685  -0.95303551  0.096774    1.19076568  2.74915002  4.86487851
  6.5620818   5.65627971  1.1596331  -4.19656497 -6.63318923 -5.71126031
 -3.56112532 -1.74772102 -0.50920268  0.50357729  1.73974945  3.54982155
  5.70096962  6.63580421  4.21968362 -1.12935048 -5.64109289 -6.56625587
 -4.87664041 -2.75922485 -1.19754851 -0.10210447  0.94672966  2.38620215
  4.42186932  6.35439095  6.14135661  2.26497653 -3.26914372 -6.45938719
 -6.06912208 -3.99192094 -2.05856394 -0.72473361  0.29744382  1.45414808
  3.13789248  5.2964482   6.662374  ]

Approximate derivative of uj, for N=127;
 [ 5.56900399  3.16495835  0.06865415 -3.04322287 -5.48747285 -6.84974493
 -7.09855327 -6.50149364 -5.44013237 -4.24873205 -3.13945557 -2.204089
 -1.45146035 -0.846809   -0.33838753  0.12961866  0.61323502  1.16850764
  1.84991198  2.70212055  3.7398339   4.91327319  6.06686103  6.91649777
  7.08629984  6.23437267  4.24035002  1.35047642 -1.843853   -4.62476585
 -6.44507176 -7.12019962 -6.81772676 -5.89663385 -4.72433886 -3.56511019
 -2.55544658 -1.73221231 -1.0740489  -0.53376336 -0.05632186  0.41432879
  0.93422655  1.55896793  2.33879011  3.30374027  4.43475676  5.62315762
  6.63615156  7.12467825  6.71263767  5.16821612  2.58203844 -0.57624972
 -3.60396953 -5.85299204 -6.98326578 -7.03500951 -6.31423461 -5.2035429
 -4.01582527 -2.93726793 -2.03964252 -1.32022076 -0.73919751 -0.2433479
  0.22330076  0.71676614  1.29304141  2.00561002  2.89522458  3.96687061
  5.15281647  6.272331    7.0173758   7.00592563  5.92482764  3.71921778
  0.71291225 -2.45483192 -5.07735527 -6.67068409 -7.12791316 -6.67037408
 -5.67194525 -4.48532319 -3.34886441 -2.37597814 -1.5886673  -0.95827886
 -0.43504293  0.03656032  0.51257241  1.04904533  1.70113533  2.51663884
  3.51857126  4.67337266  5.84956842  6.78835     7.12508037  6.49698332
  4.72444817  1.97541901 -1.21573953 -4.1324941  -6.1724564  -7.07276362
 -6.94034408 -6.11164467 -4.96430832 -3.78766155 -2.74255347 -1.88243621
 -1.19455434 -0.63499842 -0.14947538  0.31805463  0.82361444  1.42306203
  2.1684977   3.09583975  4.19885069  5.39013951  6.46310158  7.08779943
  6.8819105 ]
```

In [10]:
```
figure(3)
plot(tj6,Error6)
title('A graph of Error against tj, for n=63')
ylabel('Error')
xlabel('tj')
show()
```



In [11]:
```
figure(4)
plot(tj1,Error1)
title('A graph of Error against tj, for n=127')
ylabel('Error')
xlabel('tj')
show()
```



d)

According to both results, Fourier is more accurate than the centered second order accurate difference formula, because Fourier depicts almost the actual figure for the error between the first correct derivatives and the approximated unlike the centered one. In fourier the amplitudes of the oscillations are almost the same, which nearly depicts the sunisoidal nature of the error, nearing to simple hamonic motion unlike the centered one. The amplitude of Oscillation in fourier are large to almost dividing it by 10 to obatin that in the centered approximation, meaning that the oscillations amplitude in fourier are uniform, while in centered they increase and suddenly decrease and vice versa, making predictions difficult.

For both methods, when N is increased from 63 to 127, the curves smoothen, meaning the bigger the N the better the results. However, in fourier the amplitudes of the oscillations remain constant even if N changes, but in Centered, the amplitude Drops with increasing N, and my a big magnitude.

2. (Discrete convolution)

Computing b using fft

In [12]:
```
N=64;sig=0.1

#functions
def a(t):
    return((1/(sig*sqrt(2*pi)))*(exp((-0.5*(t**2))/sig**2)))
def x(t):
    return(1+0.05*cos(16*t))*cos(2*t)

#computing tl
tl=zeros(N)
for l in range(N):
    tl[l]=-pi + ((2*pi*l)/N)

#compute the DFTs of a and x
acap=fft(a(tl))
xcap=fft(x(tl))

#Obtain bcap
bcap=[]
for m in range(N):
    bcap.append(acap[m]*xcap[m])

#Compute the inverse
bf=real(ifft(bcap))
print('b=',bf)
```

```
b= [ 1.01301958e+01  9.80119038e+00  9.08935564e+00  8.27648067e+00
  7.16313018e+00  5.58449569e+00  3.76493438e+00  1.90352083e+00
  2.34734160e-15 -1.90352083e+00 -3.76493438e+00 -5.58449569e+00
 -7.16313018e+00 -8.27648067e+00 -9.08935564e+00 -9.80119038e+00
 -1.01301958e+01 -9.80119038e+00 -9.08935564e+00 -8.27648067e+00
 -7.16313018e+00 -5.58449569e+00 -3.76493438e+00 -1.90352083e+00
 -1.63218254e-15  1.90352083e+00  3.76493438e+00  5.58449569e+00
  7.16313018e+00  8.27648067e+00  9.08935564e+00  9.80119038e+00
  1.01301958e+01  9.80119038e+00  9.08935564e+00  8.27648067e+00
  7.16313018e+00  5.58449569e+00  3.76493438e+00  1.90352083e+00
 -1.97631958e-15 -1.90352083e+00 -3.76493438e+00 -5.58449569e+00
 -7.16313018e+00 -8.27648067e+00 -9.08935564e+00 -9.80119038e+00
 -1.01301958e+01 -9.80119038e+00 -9.08935564e+00 -8.27648067e+00
 -7.16313018e+00 -5.58449569e+00 -3.76493438e+00 -1.90352083e+00
  7.75279768e-16  1.90352083e+00  3.76493438e+00  5.58449569e+00
  7.16313018e+00  8.27648067e+00  9.08935564e+00  9.80119038e+00]
```

Computing b using (6)

In [13]:
```
from scipy.linalg import circulant
#computing the first column of A
an=a(tl)
#computing A
A=circulant(an)

N=64;tj=tl ;xj=x(tl)

#computing x
xt=transpose(xj)
#computing b
bc=matmul(A,xt)
print('b=',bc)
```

```
b= [ 1.01301958e+01  9.80119038e+00  9.08935564e+00  8.27648067e+00
  7.16313018e+00  5.58449569e+00  3.76493438e+00  1.90352083e+00
  2.42737153e-15 -1.90352083e+00 -3.76493438e+00 -5.58449569e+00
 -7.16313018e+00 -8.27648067e+00 -9.08935564e+00 -9.80119038e+00
 -1.01301958e+01 -9.80119038e+00 -9.08935564e+00 -8.27648067e+00
 -7.16313018e+00 -5.58449569e+00 -3.76493438e+00 -1.90352083e+00
 -1.87963086e-15  1.90352083e+00  3.76493438e+00  5.58449569e+00
  7.16313018e+00  8.27648067e+00  9.08935564e+00  9.80119038e+00
  1.01301958e+01  9.80119038e+00  9.08935564e+00  8.27648067e+00
  7.16313018e+00  5.58449569e+00  3.76493438e+00  1.90352083e+00
 -1.76670808e-15 -1.90352083e+00 -3.76493438e+00 -5.58449569e+00
 -7.16313018e+00 -8.27648067e+00 -9.08935564e+00 -9.80119038e+00
 -1.01301958e+01 -9.80119038e+00 -9.08935564e+00 -8.27648067e+00
 -7.16313018e+00 -5.58449569e+00 -3.76493438e+00 -1.90352083e+00
  5.78684034e-16  1.90352083e+00  3.76493438e+00  5.58449569e+00
  7.16313018e+00  8.27648067e+00  9.08935564e+00  9.80119038e+00]
```

In [14]:
```
#Difference between the computed values of b from the two methods
error=abs(bc-bf)
#computing the max-norm
max1=max(error)
print('Max-norm =', max1)
```

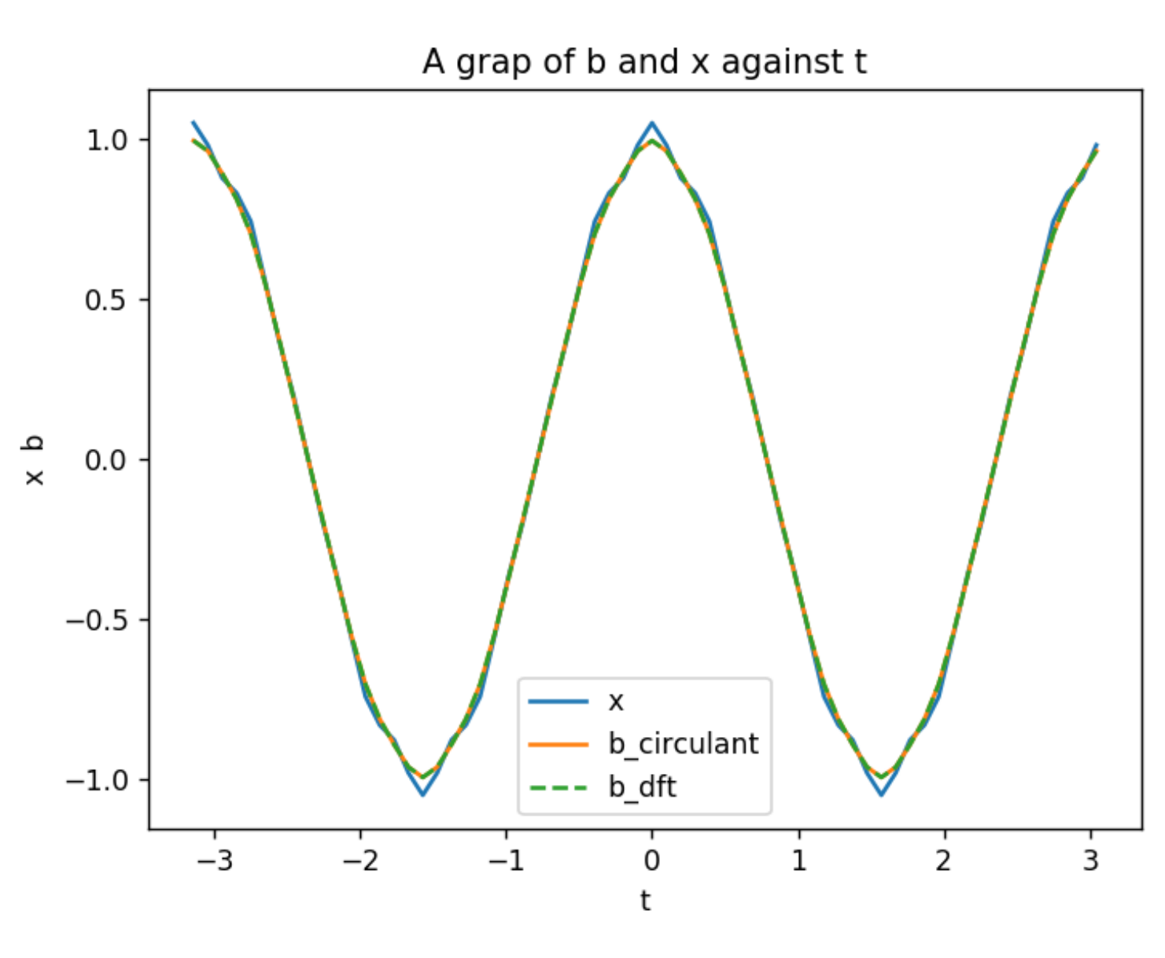
```
Max-norm = 3.552713678800501e-15
```

The max-norm is very small which implies there is almost no big difference between the two values of b, produced by the two different formulas.

Scaling b

In [15]:
```
#Using fft
scale=2*pi/N
bnf=scale*bf

#Using (6)
scale=2*pi/N
bnc=scale*bc
```

In [16]:
```
#ploting
figure(5)
plot(tj,xj,label='x')
plot(tj,bnc,label='b_circulant')
plot(tj,bnf,'--',label='b_dft')
title('A grap of b and x against t')
xlabel('t')
ylabel('x '' b')
legend()

show()
```



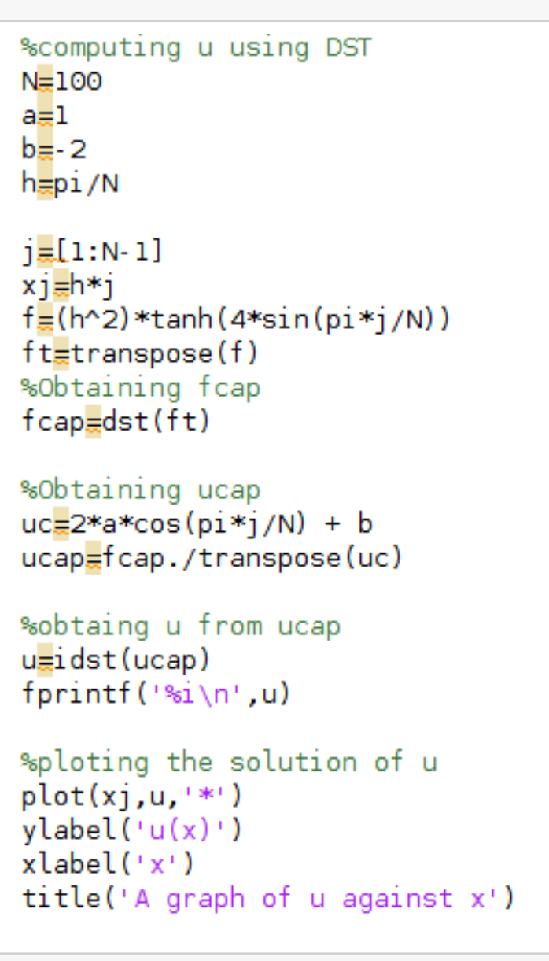
As from the graph above, its concluded that the scaled b is actually a smoothed version of x

3. (Fast solutions of trigonal linear systems)

d)

In [17]:
```
from PIL import Image, ImageDraw, ImageFilter

im = Image.open('/home/brian/Documents/Ph.D./fall_2020/Math_567/homework/hmk2/no.3.png')
im
```

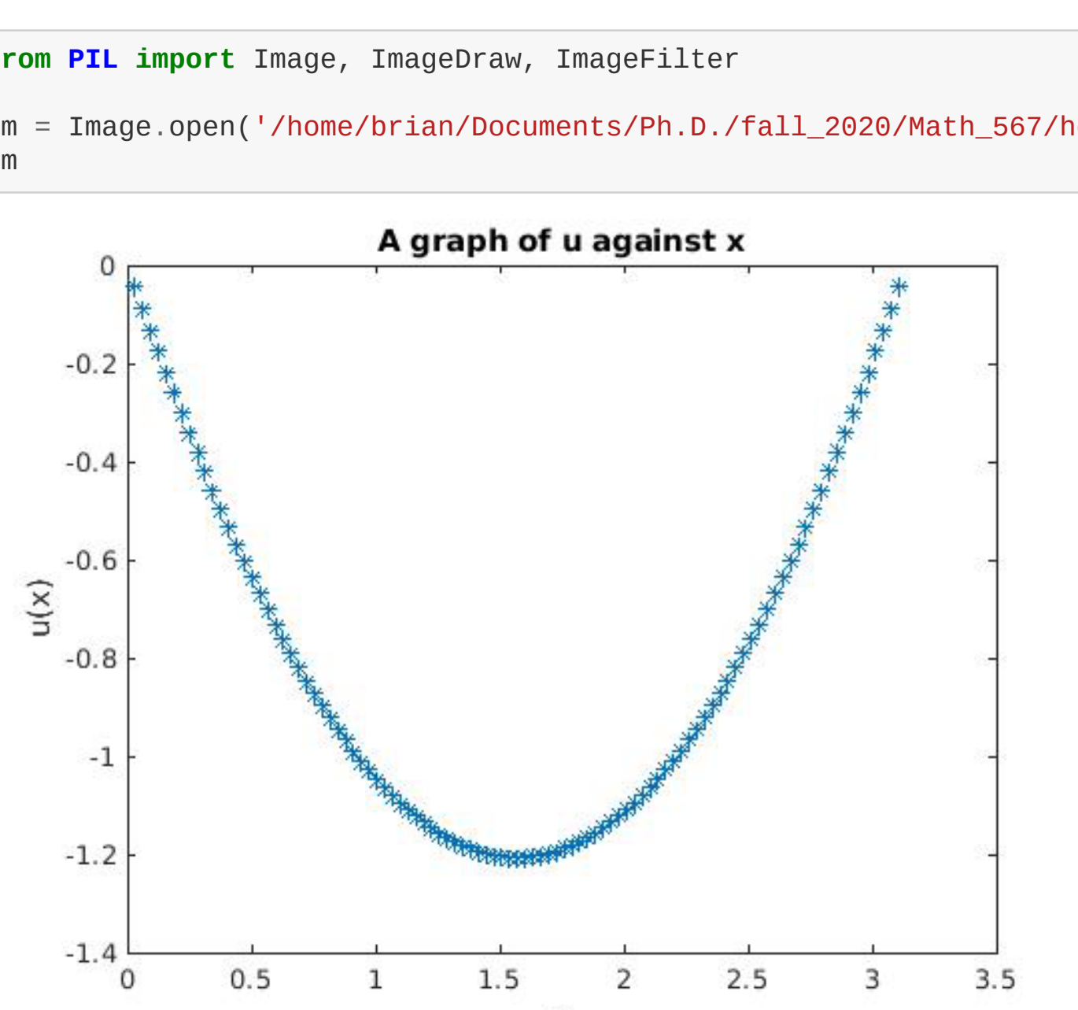
Out[17]:



```
%computing u using DST
N=100
a=1
b=-2
h=pi/N

j=[1:N-1]
xj=h*j
f=(h^2)*tanh(4*sin(pi*j/N))
ft=transpose(f)
%Obtaining fcap
fcap=dst(ft)

%Obtaining ucap
uc=2*a*cos(pi*j/N) + b
ucap=fcap./transpose(uc)

%obtaing u from ucap
u=idst(ucap)
fprintf('%i\n',u)

%ploting the solution of u
plot(xj,u,'*')
ylabel('u(x)')
xlabel('x')
title('A graph of u against x')
```

In [18]:
```
from PIL import Image, ImageDraw, ImageFilter

im = Image.open('/home/brian/Documents/Ph.D./fall_2020/Math_567/homework/hmk2/q3.jpg')
im
```

Out[18]:

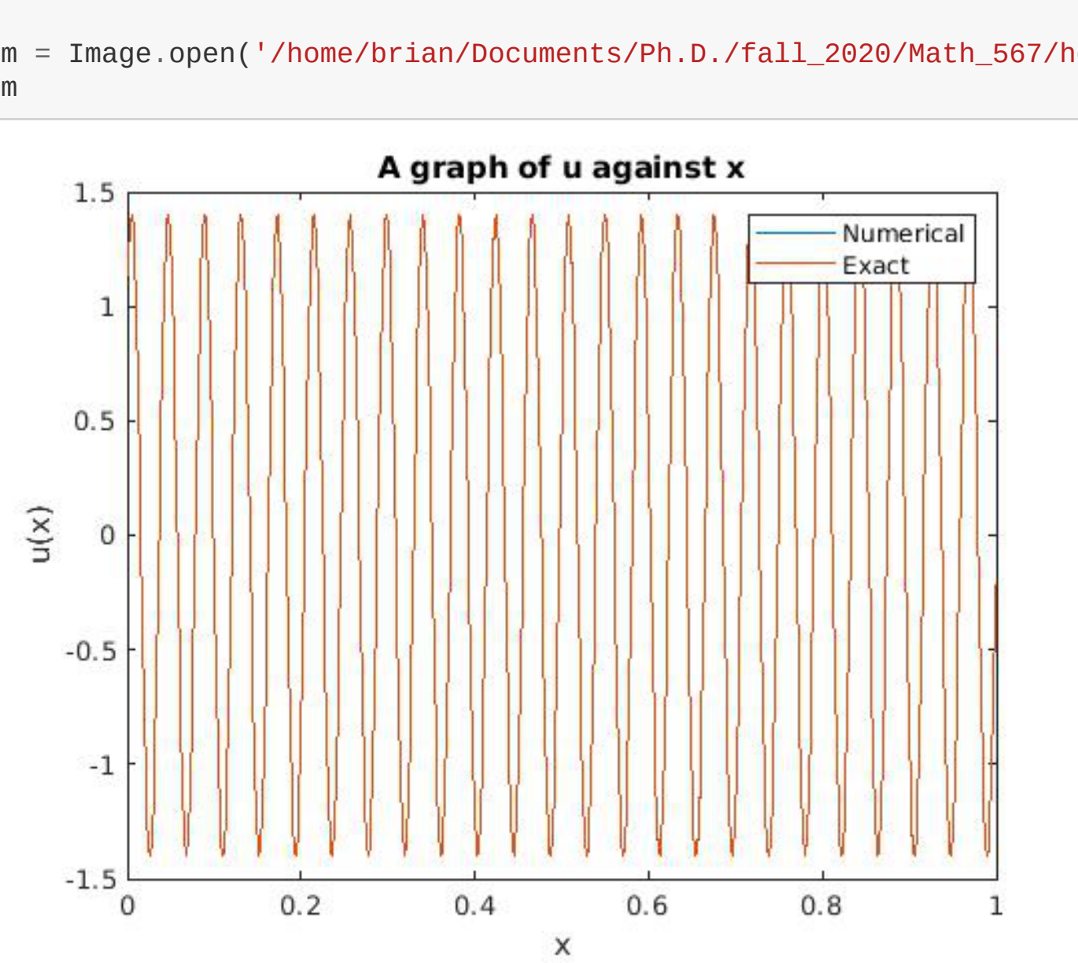


4. (Implicit FD methods)

b) Finest grid

In [20]:
```
from PIL import Image, ImageDraw, ImageFilter

im = Image.open('/home/brian/Documents/Ph.D./fall_2020/Math_567/homework/hmk2/4b.jpg')
im
```

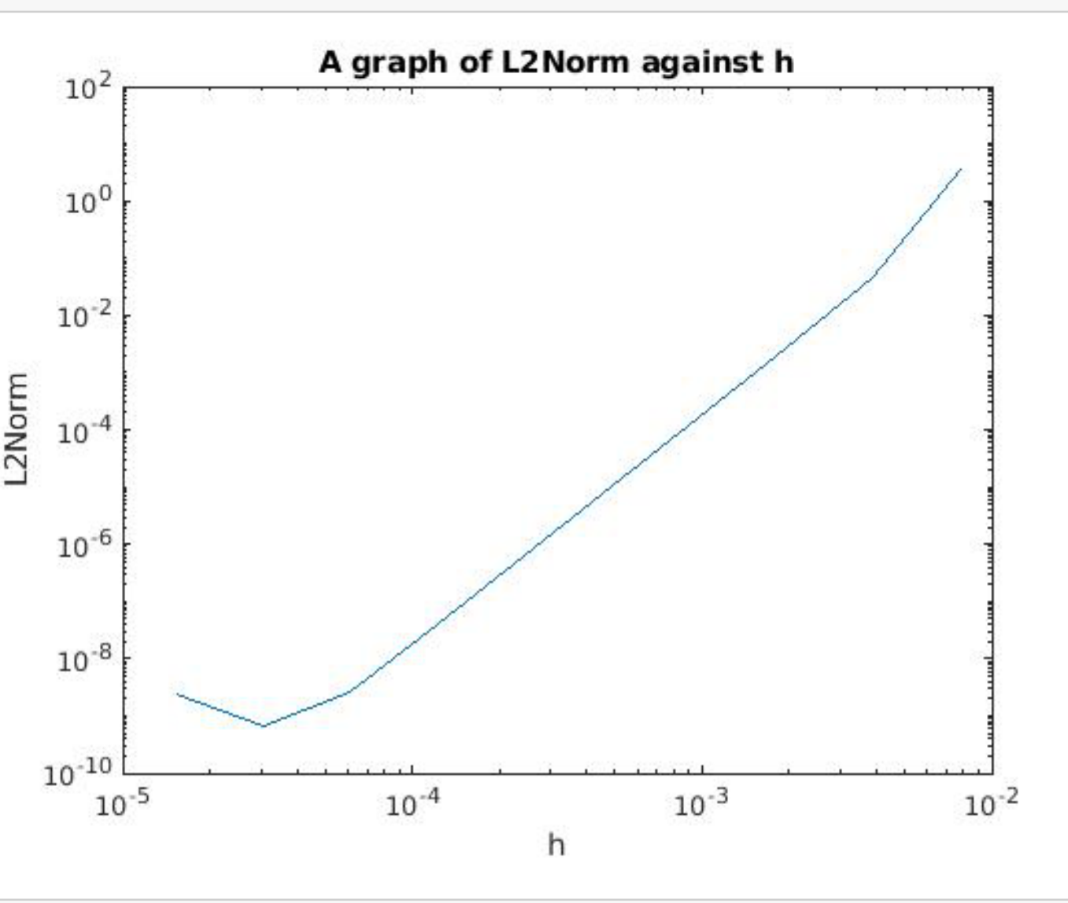
Out[20]:



Relative two-norm

In [22]:
```
from PIL import Image, ImageDraw, ImageFilter

im = Image.open('/home/brian/Documents/Ph.D./fall_2020/Math_567/homework/hmk2/L2N.jpg')
im
```

Out[22]:



In [ ]:

3. (Fast Solutions of tridiagonal linear systems) 5.

a) Show that row j of this systems simplifies to

$$\sum_{k=1}^{N-1}\hat{u}_k\left(2a\cos\left(\frac{\pi k}{N}\right)+b\right)\sin\left(\frac{\pi j k}{N}\right)=\sum_{k=1}^{N-1}\hat{f}_k\sin\left(\frac{\pi j k}{N}\right)$$

~~for the~~ Solution

from (40), at the j^{th} row

$a u_{j-1} + b u_j + a u_{j+1} = f_j$, for $f_0 = f_N = u_0 = u_N = 0$

for $j=1$

$a u_0 + b u_1 + a u_2 = f_1$, $u_0 = 0$

but $u_j = 2\sum_{k=1}^{N-1}\hat{u}_k\sin\left(\frac{\pi j k}{N}\right)$

for $j=1 \Rightarrow u_1 = 2\sum_{k=1}^{N-1}\hat{u}_k\sin\left(\frac{\pi k}{N}\right)$

$$u_2 = 2\sum_{k=1}^{N-1}\hat{u}_k\sin\left(\frac{2\pi k}{N}\right)$$

So then putting u_1 and u_2 into $a u_0 + b u_1 + a u_2 = f_1$

$$2b\sum_{k=1}^{N-1}\hat{u}_k\sin\left(\frac{\pi k}{N}\right)+2a\sum_{k=1}^{N-1}\hat{u}_k\sin\left(\frac{2\pi k}{N}\right)=2\sum_{k=1}^{N-1}\hat{f}_k\sin\left(\frac{\pi k}{N}\right)$$

$$\sum_{k=1}^{N-1}\hat{u}_k\left(b\sin\left(\frac{\pi k}{N}\right)+a\sin\left(\frac{2\pi k}{N}\right)\right)=\sum_{k=1}^{N-1}\hat{f}_k\sin\left(\frac{\pi k}{N}\right)$$

from Trigonometry $\sin 2\theta = 2\sin\theta\cos\theta$

$$\sin 2\frac{\pi k}{N} = 2\sin\frac{\pi k}{N}\cos\frac{\pi k}{N}$$

$$\sum_{k=1}^{N-1}\hat{u}_k\left(b\sin\left(\frac{\pi k}{N}\right)+2a\cos\left(\frac{\pi k}{N}\right)\sin\left(\frac{\pi k}{N}\right)\right)=\sum_{k=1}^{N-1}\hat{f}_k\sin\left(\frac{\pi k}{N}\right)$$

$$\sum_{k=1}^{N-1} \hat{U}_k \left(2a\cos\left(\frac{\pi k}{N}\right)+b\right)\sin\left(\frac{\pi k}{N}\right) = \sum_{k=1}^{N-1}\hat{f}_k \sin\left(\frac{\pi k}{N}\right)$$

for $j = N-1$

$aU_{j-1} + bU_j + aU_{j+1} = f_j$, $f_0 = f_N = U_0 = U_N =$

then

$$aU_{N-2} + bU_{N-1} + aU_N = f_{N-1} \quad — ①$$

but $U_N \equiv 0$

$$aU_{N-2} + bU_{N-1} = f_{N-1}$$

$$U_{N-2} = 2\sum_{k=1}^{N-1}\hat{U}_k \sin\left(\frac{\pi(N-2)k}{N}\right) \quad — ②$$

$$U_{N-1} = 2\sum_{k=1}^{N-1}\hat{U}_k \sin\left(\frac{\pi(N-1)k}{N}\right) \quad — ③$$

$$f_{N-1} = 2\sum_{k=1}^{N-1}\hat{f}_k \sin\left(\frac{\pi(N-1)}{N}k\right) \quad — ④$$

Putting ②, ③, ④ and into ①

$$a\sum_{k=1}^{N-1}\hat{U}_k \sin\left(\frac{\pi(N-2)k}{N}\right) + b\sum_{k=1}^{N-1}\hat{U}_k \sin\left(\frac{\pi(N-1)k}{N}\right) = \sum_{k=1}^{N-1}\hat{f}_k \sin\left(\frac{\pi(N-1)}{N}k\right)$$

$$\sum_{k=1}^{N-1}\hat{U}_k \left[a\sin\left(\frac{\pi(N-2)}{N}k\right) + b\sin\left(\frac{\pi(N-1)k}{N}\right)\right] = \sum_{k=1}^{N-1}\hat{f}_k \sin\left(\frac{\pi(N-1)k}{N}\right)$$

but $\sin\left(\frac{\pi(N-2)}{N}k\right) = 2\cos\left(\frac{\pi k}{N}\right)\sin\left(\frac{\pi(N-1)}{N}k\right)$

$$\sum_{k=1}^{N-1}\hat{U}_k \left[2a\cos\left(\frac{\pi k}{N}\right) + b\right]\sin\left(\frac{\pi(N-1)}{N}k\right) = \sum_{k=1}^{N-1}\hat{f}_k \sin\left(\frac{\pi(N-1)}{N}\right)$$

$$\sum_{k=1}^{N-1} \hat{u}_k \left[2a\cos\left(\frac{\pi k}{N}\right)+b\right] \sin\left(\frac{\pi(N-1)}{N}k\right) = \sum_{k=1}^{N-1} \hat{f}_k \sin\left(\frac{\pi(N-1)}{N}k\right)$$

b) from (a)

$$\sum_{k=1}^{N-1} \hat{u}_k \left(2a\cos\left(\frac{\pi k}{N}\right)+b\right) \sin\left(\frac{\pi j k}{N}\right) = \sum_{k=1}^{N-1} \hat{f}_k \sin\left(\frac{\pi j k}{N}\right),$$

$$\sum_{k=1}^{N-1} \left[\left(2a\cos\left(\frac{\pi k}{N}\right)+b\right) \sin\frac{\pi j k}{N}\right] \hat{u}_k = \sum_{k=1}^{N-1} \left[\sin\left(\frac{\pi j k}{N}\right)\right] \hat{f}_k$$

$$\sum_{k=1}^{N-1} \left[\sin\left(\frac{\pi j k}{N}\right)\right] \left(2a\cos\left(\frac{\pi k}{N}\right)+b\right) \hat{u}_k = \sum_{k=1}^{N-1} \left[\sin\left(\frac{\pi j k}{N}\right)\right] \hat{f}_k$$

hence we can conclude that

$\left(2a\cos\left(\frac{\pi k}{N}\right)+b\right)\hat{u}_k = \hat{f}_k$,, since we have the term $\sum_{k=1}^{N-1} \sin\frac{\pi j k}{N}$ on both sides of the equation,

therefore
$$\hat{u}_k = \frac{\hat{f}_k}{2a\cos\left(\frac{\pi k}{N}\right)+b}, \quad k=1,\dots,N-1$$

c) To obtain the solution to (10) we need to obtain the values of u,

But $U = \{U_j\}_{j=0}^{j=N-1}$, and to get u_j we need $\hat{u}_k$ and ~~f_k~~, since $U = A^{-1}f$ ~~from (b)~~ so from (b)

$$\hat{u}_k = \frac{\hat{f}_k}{2a\cos(\pi k/N)+b}$$

from (a) $\hat{f}_k = \frac{1}{N}\sum_{j=1}^{N-1} f_j \sin\left(\frac{\pi j k}{N}\right)$

then

$$\hat{u}_k = \cancel{\hat{f}_k}\ \frac{\frac{1}{N}\sum_{j=1}^{N-1} f_j \sin\left(\frac{\pi j k}{N}\right)}{2a\cos\left(\frac{\pi k}{N}\right) + b}$$

but in part (a) have have the value of u_j,

$$u_j = 2\sum_{k=1}^{N-1} \hat{u}_k \sin\left(\frac{\pi j k}{N}\right)$$

$$u_j = 2\sum_{k=1}^{N-1}\left[\frac{\frac{1}{N}\sum_{j=1}^{N-1} f_j \sin\left(\frac{\pi j k}{N}\right)}{2a\cos\left(\frac{\pi k}{N}\right) + b}\right]\sin\left(\frac{\pi j k}{N}\right)$$

Which is the solution for (10), so with that we can solve $u = A^{-1}f$

where A^{-1} is the inverse of $A = \begin{bmatrix} b & a & 0 & \cdots & 0 \\ a & b & a & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & a & b & a \\ 0 & \cdots & 0 & a & b \end{bmatrix}$

and $f = \{f_j\}_{j=1}^{j=N-1}$

4. Implicit FD methods.

a) Generalize the technique above to derive an Implicit, three point, fourth-order accurate FD formula for the following BVP:

$$u''(x) + k^2 u(x) = f(x), \quad a \le x \le b \quad - ①$$

Using Fornberg [1, P.67], Equation ① can be written as

$$\frac{1}{h^2}\left[-\tfrac{1}{12} \;\; \tfrac{4}{3} \;\; -\tfrac{5}{2} \;\; \tfrac{4}{3} \;\; -\tfrac{1}{12}\right]u + k^2 u = f + O(h^4) \quad --- ②$$

Differentiating ② both sides with respect to x twice, we obtain:

$$u''''(x) + k^2 u''(x) = f''(x), \quad a \le x \le b$$

Based on (13), we obtain

$$\frac{1}{h^4}\left[1 \;\; -4 \;\; 6 \;\; -4 \;\; 1\right]u + \frac{k^2}{h^2}\left[1 \;\; -2 \;\; 1\right]u = \frac{1}{h^2}\left[1 \;\; -2 \;\; 1\right]f + O(h^2) \quad ---- ③$$

Combining Equations ② and ③ using linear Combination, we obtain

$$(2) + \frac{h^2}{12}(3)$$

This is the Same as,

$$\frac{1}{h^2}\left[0 \;\; 1 \;\; -2 \;\; 1 \;\; 0\right]u + k^2\left[\tfrac{1}{12} \;\; \tfrac{5}{6} \;\; \tfrac{1}{12}\right]u = \left[\tfrac{1}{12} \;\; \tfrac{5}{6} \;\; \tfrac{1}{12}\right]f + O(h^4)$$

Factoring out $1/h^2$ on the left hand side, we obtain
,

$$\frac{1}{h^2}\left[[1 \quad -2 \quad 1]u + \frac{k^2h^2}{12}[1 \quad 10 \quad 1]u\right] = \frac{1}{12}[1 \quad 10 \quad 1]f + O(h^4)$$

which simplifies to

$$\frac{1}{h^2}\left[1+\frac{(kh)^2}{12} \quad -2+\frac{5}{6}(kh)^2 \quad 1+\frac{(kh)^2}{12}\right]u = \frac{1}{12}[1 \quad 10 \quad 1]f + O(h^4)$$

Which is an implicit, three point, fourth-order accurate FD formula, similar to equation (11)

b)

```
%computing u using DST
N=100;
a=1;
b=-2;
h=pi/N;

j=[1:N-1];
xj=h*j;
f=(h^2)*tanh(4*sin(pi*j/N));
ft=transpose(f);
%Obtaining fcap
fcap=dst(ft);

%Obtaining ucap
uc=2*a*cos(pi*j/N) + b;
ucap=fcap./transpose(uc);

%obtaing u from ucap
u=idst(ucap);
fprintf('%10s %16.8e\n',u);

%ploting the solution of u
plot(xj,u,'*');
ylabel('u(x)');
xlabel('x');
title('A graph of u against x');
```

```
-4.375795e-02  -8.73925476e-02
-1.307843e-01  -1.73821216e-01
-2.164010e-01  -2.58432845e-01
-2.998381e-01  -3.40549912e-01
-3.805122e-01  -4.19678848e-01
-4.580122e-01  -4.95481690e-01
-5.320629e-01  -5.67736156e-01
-6.024858e-01  -6.36299427e-01
-6.691670e-01  -7.01080751e-01
-7.320343e-01  -7.62022613e-01
-7.910417e-01  -8.19088438e-01
-8.461601e-01  -8.72254798e-01
-8.973707e-01  -9.21506537e-01
-9.446612e-01  -9.66833728e-01
-9.880235e-01  -1.00822978e+00
-1.027452e+00  -1.04569027e+00
-1.062944e+00  -1.07921216e+00
-1.094495e+00  -1.10879337e+00
-1.122106e+00  -1.13443243e+00
-1.145773e+00  -1.15612831e+00
-1.165497e+00  -1.17388028e+00
-1.181277e+00  -1.18768782e+00
-1.193112e+00  -1.19755057e+00
-1.201003e+00  -1.20346831e+00
-1.204948e+00  -1.20544091e+00
-1.204948e+00  -1.20346831e+00
-1.201003e+00  -1.19755057e+00
-1.193112e+00  -1.18768782e+00
-1.181277e+00  -1.17388028e+00
-1.165497e+00  -1.15612831e+00
```

```
  -1.145773e+00  -1.13443243e+00
  -1.122106e+00  -1.10879337e+00
  -1.094495e+00  -1.07921216e+00
  -1.062944e+00  -1.04569027e+00
  -1.027452e+00  -1.00822978e+00
  -9.880235e-01  -9.66833728e-01
  -9.446612e-01  -9.21506537e-01
  -8.973707e-01  -8.72254798e-01
  -8.461601e-01  -8.19088438e-01
  -7.910417e-01  -7.62022613e-01
  -7.320343e-01  -7.01080751e-01
  -6.691670e-01  -6.36299427e-01
  -6.024858e-01  -5.67736156e-01
  -5.320629e-01  -4.95481690e-01
  -4.580122e-01  -4.19678848e-01
  -3.805122e-01  -3.40549912e-01
  -2.998381e-01  -2.58432845e-01
  -2.164010e-01  -1.73821216e-01
  -1.307843e-01  -8.73925476e-02
  -4.375795e-02
```

Published with MATLAB® R2020a

```
%Calculating relative two norm of the error in the approximate solution

l=[7:16];
h=1./(2.^l);
k=150;
u0=1; u1=0;

L2Norm=zeros(10,1);
for ii=1:10
    N=1/h(ii);
    j=[1:N-1]';
    x=h(ii)*j;
    uaprox=numerical(k,h(ii),u0);
    uexact=u_ex(x,k);
    L2Norm(ii)=relat(uaprox,uexact);
end

loglog(h,L2Norm)
polyfit(log(h),log(L2Norm),1)

ylabel('L2Norm')
xlabel('h')
title('A graph of L2Norm against h')
fprintf('since the slope of the graph is 3.60697 which is approximately 4, hence the graph converges as O(h^4)')
%exact solution
function uexact=u_ex(xj,k)
c=1/k^2;
uexact=c+(1-c)*cos(k*xj)-(c+(1-c)*cos(k))*(csc(k))*sin(k*xj);
end

%Numerical solution
function uaprox=numerical(k,h,u0)
%N=1000;
N=1/h;
j=[1:N-1]';
a=(1+(1/12)*(k*h)^2)/(h^2);
b=(-2+(5/6)*(k*h)^2)/(h^2);

xj=j/N;
f=zeros(N-1,1);
f(j)=1;

%boundary condition
f(1)=f(1) - a*u0;

%obtaining fcap
fcap=dst(f);

%Obtaining ucap
uc=2*a*cos(pi*j/N) + b;
ucap=fcap./uc;

%obtaing u from ucap
uaprox=idst(ucap);
end

%Relative two_norm
function Re=relat(uaprox,uexact)
error = (uaprox - uexact).^2;
Re=sqrt(sum(error)/sum(uexact.^2));
end
```

```
ans =

    3.6097   16.7657
```

```
since the slope of the graph is 3.60697 which is approximately 4, hence the graph converges as O(h^4)
```

Published with MATLAB® R2020a