In [1]:
```
%matplotlib notebook
%pylab
```
```
Using matplotlib backend: nbAgg
Populating the interactive namespace from numpy and matplotlib
```

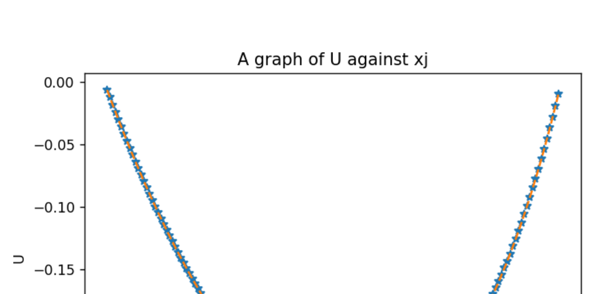
1. Linear two-point boundary value problems

a)

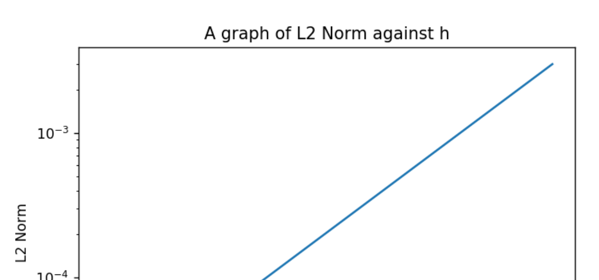
In [2]:
```
def fd2tpbvp(p,q,r,alpha,beta,m):
    '''
    Arguments
    ---------
    p,q,r: are function coefficients in the equation u''=p(x)u' + q(x)u +r(x)
    alpha and beta: are boundary function points:u(0) and u(1) respectively
    m: number of interior discretization points

    Parameters
    ----------
    A: is m by m interior matrix
    e1 and em: are first and last columns of and m by m identity matrix respectively
    rh: is a column vector of r(x)

    Returns
    -------
    U_h: second derivative of fuction u(x)
    '''
    a=0;b=1
    h=(b-a)/(m+1)
    x=zeros(m)
    A=zeros((m,m))
    F=zeros(m)
    rh=zeros(m)
    rh[m-1]=r(b)
    e1=zeros(m);em=zeros(m)
    e1[0]=1;em[m-1]=1
    for j in range(m):
        x[j]=(1+j)*h
        f1=(1/(h**2))+(p(x[j]))/(2*h)
        f2=(1/(h**2))-(p(x[j]))/(2*h)
        rh[j]=r(x[j])

        A[j,j]=(-2/(h**2))-q(x[j])
        if  j == 0:
            A[j,j+1]=(1/(h**2))-(p(x[j]))/(2*h)
            F=r(x[j])-(f1*alpha*e1)

        elif 0<j<=m-2:
            A[j,j+1]=(1/(h**2))-p(x[j])/(2*h)
            A[j,j-1] = (1/h**2) + p(x[j])/(2*h)
            F=r(x[j])
        else:
            A[j,j-1] = (1/h**2) + p(x[j])/(2*h)
            F=r(x[j])-(f2*beta*em)
    U_h=linalg.solve(A,F) #inverse(A)*F
    return U_h
```

b)

In [3]:
```
#numerically
p=lambda x: 2*(tan(x))
q=lambda x: 0
r=lambda x: 2
#exact solution
uexact=lambda x: (x-1)*(tan(x))

alpha=0;beta=0
Uexact=[]
h=[]
uapprox=[]
m=[10,20,40,80,160]
for j in m:
    uapp=fd2tpbvp(p,q,r,alpha,beta,j)
    uapprox.append(uapp)
    xj=linspace(0,1,j+2)
    hi=1/(j+1)
    xj=xj[1:-1]

    h.append(hi)
    u_exact=uexact(xj)
    Uexact.append(u_exact)

figure(1)
plot(xj,uapprox[4],'*',label="Numerical")
plot(xj,Uexact[4],'--',label="Exact")

title("A graph of U against xj")
xlabel("j")
ylabel("U")
legend()
show()
```



In [4]:
```
#Relative two-norm
def r2norm(V,A):
    '''
    Parameters
    ----------
    A: Is a vector containing Approximated values
    V: Contains exact values of the derivatives

    Return
    ------
    L2: L2 norm of the error
    '''
    error= V-A
    L2=sqrt(sum(error**2)/sum(V**2))
    return L2
V=array(Uexact)
A=array(uapprox)
Error=zeros(5)
for i in range(5):
    Error[i]=r2norm(V[i],A[i])
#plots
figure(2)
loglog(h,Error)
title("A graph of L2 Norm against h")
xlabel("h")
ylabel("L2 Norm")

show()
```



In [5]:
```
pol=polyfit(log(h),log(Error),1) #polyfit
p=pol[0]
print("Order of accuracy, p = ",p)
```
```
Order of accuracy, p =  1.9997796951609452
```

Since the order of accuracy p = 1.9997796951609452 which is approximately 2 then the approximate solution is second order accurate.

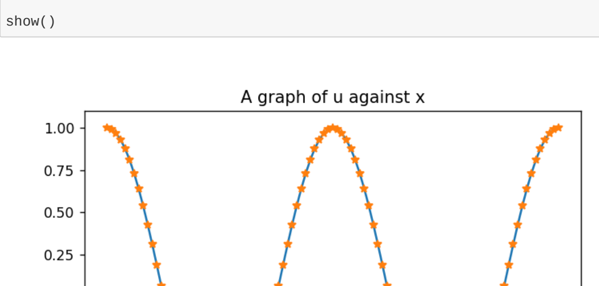
3. Neumann-Neumann boundary conditions

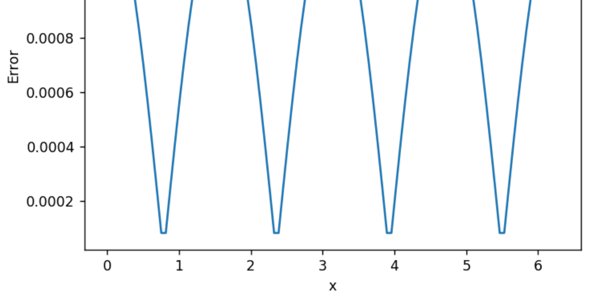
c) Solve th BVP (6) numerically

In [6]:
```
a=0 ;b=2*pi

m=99

sig0=0; sig1= 0

def f(x):
    return (-4*cos(2*x))

def numerical(f,sig0,sig1,m,a,b):
    '''
    Return: returns the numerical approximation of a function f
    '''
    h=(b-a)/(m+1)
    c=1/(h**2)
    x=zeros(m+2)
    A=zeros((m+3,m+3)) #(m+3)-by-(m+3) matrix
    F=zeros(m+3)

    for j in range(m+2):
        x[j]=a+j*h
        A[j,j]=-2*c

        if j == 0:
            A[j,j+1]=2*c
            A[j,-1]=1/2
            A[-1,j]=1/2
            F[j]=f(a)+(2/h)*sig0

        elif 0<j<m+1:
            A[j,j+1]=c
            A[j,j-1]=c
            A[j,-1]=1
            A[-1,j]=1
            F[j]=f(x[j])

        else:
            A[j,j+1]=1/2
            A[j,j-1]=2*c
            A[-1,j]=1/2
            F[j]=f(b)-(2/h)*sig1

    U=solve(A,F)
    return U
```

In [7]:
```
#Numerical solution at U=0
U=numerical(f,sig0,sig1,m,a,b)
Uapp=U[:-1]
```

In [13]:
```
x=zeros(m+2)
h=(b-a)/(m+1)
for j in range(m+2):
    x[j]=a+j*h

#True solution
def u(x):
    return cos(2*x)
u=u(x)

#check the numerical solution against the exact solution
figure(3)
plot(x,u,label="true solution")
plot(x,Uapp,'*',label="numerical")
title("A graph of u against x")
legend()
xlabel("x")
ylabel("u")
show()


#error
Error=abs(Uapp-u)

figure(5)
plot(x,Error)
title("A graph of Error against x")
xlabel("x")
ylabel("Error")

show()
```





The numerical solution well approximatees the exact solution

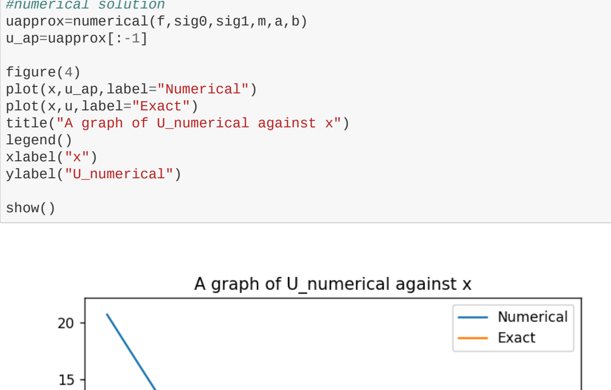
In [27]:
```
#Relative two-norm of the error
L2_norm=r2norm(u,Uapp)
print("The relative two-norm of the error:",L2_norm)
```
```
The relative two-norm of the error: 0.0013169869352425603
```

In [28]:
```
#Report the value of lambda
# the value of lambda is equivalent to U[m+1]=U[-1]
print("The value of lambda:",U[-1])
```
```
The value of lambda: -2.0084436626384742e-16
```

Since the value of λ is negative very close to zero, and also that the relative error is small, then it implies that the solution u approximatly solves the original solution. And also λ is an eigen value, so since its negative, it implies that we have a stable saddle point at the fixed point were we want to obtain the solution.

d)

In [29]:
```
def f(x):
    return x

sig0=-pi**2; sig1=pi**2

#numerical solution
uapprox=numerical(f,sig0,sig1,m,a,b)
u_ap=uapprox[:-1]

figure(4)
plot(x,u_ap,label="Numerical")
plot(x,u,label="Exact")
title("A graph of U_numerical against x")
legend()
xlabel("x")
ylabel("U_numerical")

show()
```



In [30]:
```
#Report the value of lambda
# the value of lambda is equivalent to U[m+1]=U[-1]
print("The value of lambda:",uapprox[-1])
```
```
The value of lambda: 2.85645320908583e-16
```

In [31]:
```
#Relative two-norm of the error
L2_norm=r2norm(u,u_ap)
print("The relative two-norm of the error:",L2_norm)
```
```
The relative two-norm of the error: 12.058211579440641
```

Since λ is a postive and very close to zero doesnot gurantee approximation of the solution to (2). Since the relative two norm is too big, this implies that is a significantly big difference between the exact and numerical solution. Hence this solution canot approximate the solution to the original system (2). And also according to the graph there is a huge difference between the nature and bahaveiour of the exact compared to the approximated. Hence concluding that the solutions are not even near to each other. Also the value of the eigen value λ is positive which implies that we have unstable saddle point at fixed point x where we want to obtain the solution which doesn't seem reasonable.

In [ ]:

2. Fictitious Point method for Robin Boundary Conditions

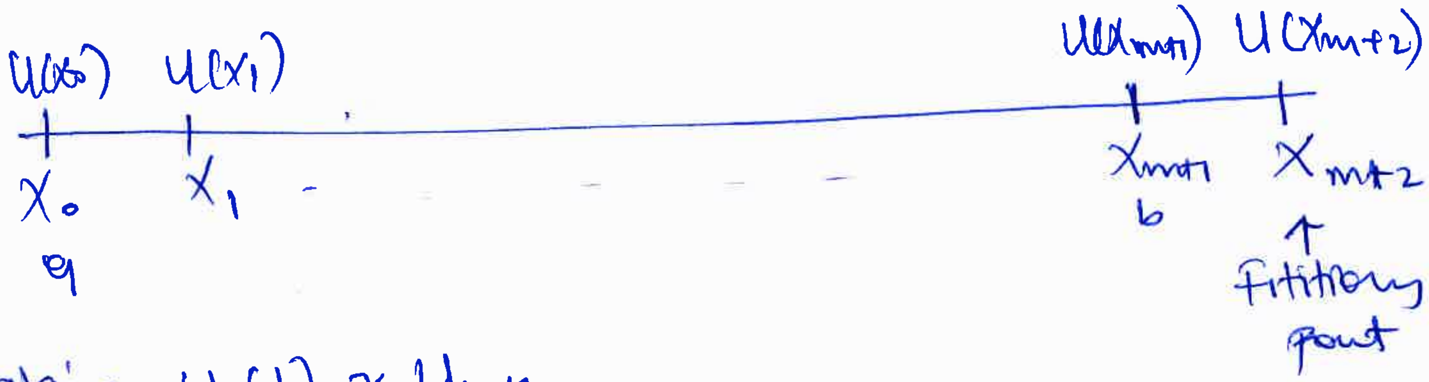
Consider

$$U'' = P(x)U' + q(x)U + r(x) \quad \text{--- ①}, \; x \in [a, b]$$

with mixed Boundary Conditions.

$$U(a) = \alpha \quad \text{and} \quad \beta_1 U(b) + \beta_2 U'(b) = \beta_3$$

Discretize equation ① with m+1 equally spaced Sub intervals.



taking $U(b) \approx U_{m+1}$

$U(b-h) \approx U_m$

$P_{m+1} = P(b)$

$q_{m+1} = q(b)$

$r_{m+1} = r(b)$

After discretization at x_{m+1}, equation ① becomes;

$$U''_{m+1} = P_{m+1}U'_{m+1} + q_{m+1}U_{m+1} + r_{m+1} \quad \text{--- ②}$$

From the Centeral diffence formula;

$$U''_{m+1} = \frac{U_{m+2} - 2U_{m+1} + U_m}{h^2}$$

where U_{m+2} is $U(x_{m+2})$, where for x_{m+2} a Fititious point.

from the Boundary conditions; $\beta_1 U_{m+1} + \beta_2 U'_{m+1} = \beta_3$

we have

$$U'_{m+1} = \frac{\beta_3 - \beta_1 U_{m+1}}{\beta_2}$$

So

$$P_{m+1} U'_{m+1} = \frac{P_{m+1}}{\beta_2}\left(\beta_3 - \beta_1 U_{m+1}\right) \quad \text{—} \; ③$$

but U'_{m+1} can be discretised to),

$$U'_{m+1} = \frac{U_{m+2} - U_m}{2h} = \frac{\beta_3 - \beta_1 U_{m+1}}{\beta_2}$$

$$U_{m+2} = \frac{2h}{\beta_2}\left(\beta_3 - \beta_1 U_{m+1}\right) + U_m$$

from

$$U''_{m+1} = \frac{U_{m+2} - 2U_{m+1} + U_m}{h^2}$$

Substituting U_{m+2}, we obtain

$$U''_{m+1} = \frac{\frac{2h}{\beta_2}\left(\beta_3 - \beta_1 U_{m+1}\right) + U_m - 2U_{m+1} + U_m}{h^2}$$

Substituting U''_{m+1} and equation ③ into equation ②

we have

$$\frac{\frac{2h}{\beta_2}\left(\beta_3 - \beta_1 U_{m+1}\right) + 2U_m - 2U_{m+1}}{h^2} = \frac{P_{m+1}}{\beta_2}\left(\beta_3 - \beta_1 U_{m+1}\right) + q_{m+1} U_{m+1} + r_{m+1}$$

$$\frac{2h}{\beta_2}(\beta_3 - \beta_1 U_{m+1}) + 2U_m - 2U_{m+1} - q_{m+1} h^2 U_{m+1} + \frac{p_{m+1}}{\beta_2} h^2 \beta_1 U_{m+1} = \frac{p_{m+1}}{\beta_2} h^2 \beta_3 + h^2 r_{m+1}$$

which simplifies to

$$2U_m + \left[-2 - h^2 q_{m+1} + \beta_1/\beta_2 \, h^2 p_{m+1} - 2h \, \beta_1/\beta_2\right] U_{m+1} = -2h\frac{\beta_3}{\beta_2} + \frac{\beta_3}{\beta_2} h^2 p_{m+1} + h^2 r_{m+1}$$

$$-2U_m + \left[2 + h^2 q_{m+1} + (2 - h p_{m+1}) h \frac{\beta_1}{\beta_2}\right] U_{m+1} = -h^2 r_{m+1} + (2 - h p_{m+1}) h \, \beta_3/\beta_2$$

$\boxed{\cdot}$

3 9)

Show that the linear system (5) has a unique solution regardless of b.

from (1) $Au + \lambda w = 0$

multiplying through by w^T from the left hand side.

$$w^T(Au + \lambda w) = 0$$

$$w^T A u + w^T \lambda w = 0$$

Since w is an non zerovector with nt2 entries ~~a~~ and an eigen value of A then $w^T A = 0^T$, then,

$$w^T A u = 0 \Rightarrow w^T \lambda w = 0$$

This becomes

$\lambda w^T w = 0$, Since λ is a constant

Since w is non zerovector then $w^T w \neq 0$, therefore for $\lambda w^T w = 0$ then λ must be zero.

hence $\lambda = 0$

from (1): $Au + \lambda w = 0$

if $\lambda = 0$

$$Au = 0$$

(ii) $w^T u = 0$

If $Au = 0$, This means $u = \alpha e$ for some α

Using $w^T u = 0$, then substituting in $u = \alpha e$

$w^T \alpha e = 0$, Since α is a constant then

$$\alpha w^T e = 0.$$

but $w^T = \begin{bmatrix} \frac{1}{2} & 1 & \cdots & 1 & \frac{1}{2} \end{bmatrix}$

So,

$\cancel{w^T e_1}$ $e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$

So $w^T e = \begin{bmatrix} \frac{1}{2} & 1 & \cdots & 1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$

$$w^T e = \frac{1}{2} + 1 + 1 \cdots 1 + 1 + \frac{1}{2}$$

Since w and e are vectors of length $m+2$ ~~in~~

then $w^T e = 1 + 1 + 1 \cdots 1 + 1$

we are suming 1 $m+1$ terms which reduces to

$$w^T e = \sum_{i=1}^{m+1} 1 = m+1 \neq 0$$

but $\alpha w^T e = 0$, hence for $\alpha w^T e$ to be zero then α must be zero because $m+1 \neq 0$.

therefore $\underline{\alpha = 0}$

b)

Show that if $w^T b = 0$ in (5) then $\lambda = 0$.

from (5)

$$Au + \lambda w = b$$

multiplying through by w^T from the left hand side we have

$$w^T A u + w^T \lambda w = w^T b$$

If $w^T b = 0$ ~~than~~ ~~A~~ and $Au = 0$ then

$$w^T \lambda w = 0 \Rightarrow \lambda w^T w = 0$$

Since w is an non zero vector then $w^T w \neq 0$, therefore for $\lambda w^T w$ to be ~~z~~ zero then $\underline{\underline{\lambda = 0}}$

4. Neuman-Neuman Boundary Conditions and DST

a) Show that row j of ~~system~~ (2) simplifies to

$$\sum_{k=0}^{m+1}{}'' \hat{u}_k \left(2\cos\left(\frac{\pi k}{m+1}\right)-2\right)\cos\left(\frac{\pi j k}{m+1}\right) = h^2 \sum_{k=0}^{m+1}{}'' \hat{u}_k \cos\left(\frac{\pi j k}{m+1}\right)$$

from (2), we can conclude that the j^{th} row is given by

$$\frac{1}{h^2}\left(u_{j-1}-2u_j+u_{j+1}\right) = f_j$$

Starting for the case $1 \le j \le m$, we have

$$u_{j-1}-2u_j+u_{j+1} = 2\sum_{k=0}^{m+1}{}'' \hat{u}_k \cos\left(\frac{\pi(j-1)}{m+1}k\right) - 2\left(2\sum_{k=0}^{m+1}{}'' \hat{u}_k \cos\left(\frac{\pi j k}{m+1}\right)\right)$$
$$+ 2\sum_{k=0}^{m+1}{}'' \hat{u}_k \cos\left(\frac{\pi(j+1)}{m+1}k\right)$$

$$= 2\sum_{k=0}^{m+1}{}'' \hat{u}_k \cos\left(\frac{\pi(j-1)}{m+1}k\right) - 4\sum_{k=0}^{m+1}{}'' \hat{u}_k \cos\left(\frac{\pi j k}{m+1}\right)$$
$$+ 2\sum_{k=0}^{m+1}{}'' \hat{u}_k \cos\left(\frac{\pi(j+1)}{m+1}k\right)$$

$$= 2\sum_{k=0}^{m+1}{}'' \hat{u}_k \left[\cos\left(\frac{\pi(j-1)}{m+1}k\right) - 2\cos\left(\frac{\pi j k}{m+1}\right) + \cos\left(\frac{\pi(j+1)}{m+1}k\right)\right]$$

but $\cos\left(\frac{\pi(j-1)}{m+1}k\right) + \cos\left(\frac{\pi(j+1)}{m+1}k\right) = 2\cos\left(\frac{\pi j k}{m+1}\right)\cos\left(\frac{\pi k}{m+1}\right)$

$$= 2\sum_{k=0}^{m+1}{}'' \hat{u}_k \left[2\cos\left(\frac{\pi j k}{m+1}\right)\cos\left(\frac{\pi k}{m+1}\right) - 2\cos\left(\frac{\pi j k}{m+1}\right)\right]$$

therefore;

$$U_{j-1} - 2U_j + U_{j+1} = 2\sum_{k=0}^{m+1}{}'' \hat{U}_k \left[2\cos\left(\frac{\pi k}{m+1}\right) - 2\right] \cos\left(\frac{\pi j k}{m+1}\right)$$

So

$$U_{j-1} - 2U_j + U_{j+1} = h^2 f_j \;, \; f_j = \sum_{k=0}^{m+1}{}'' \hat{f}_k \cos\left(\frac{\pi j k}{m+1}\right), \text{ becomes}$$

$$\sum_{k=0}^{m+1}{}'' \hat{U}_k \left(2\cos\left(\frac{\pi k}{m+1}\right) - 2\right) \cos\left(\frac{\pi j k}{m+1}\right) = h^2 \sum_{k=0}^{m+1}{}'' \hat{f}_k \cos\left(\frac{\pi j k}{m+1}\right)$$

Checking for $j=0$.

$\cancel{U_{j-1}} - 2U_j + 2U_{j+1} = h^2 \hat{f}_j$ becomes;

$\cancel{U_{0-1}} - 2U_0 + 2U_1 = h^2 \hat{f}_0$

$$-2U_0 + 2U_1 = -4\sum_{k=0}^{m+1}{}'' \hat{U}_k \cos\left(\frac{\pi k}{m+1}(0)\right) + 4\sum_{k=0}^{m+1}{}'' \hat{U}_k \cos\left(\frac{\pi k}{m+1}\right)$$

$$= 2\sum_{k=0}^{m+1}{}'' \hat{U}_k \left(2\cos\left(\frac{\pi k}{m+1}\right) - 2\cos\left(\frac{\pi k}{m+1}(0)\right)\right)$$

$$= 2\sum_{k=0}^{m+1}{}'' \hat{U}_k \left(2\cos\left(\frac{\pi k}{m+1}\right) - 2\right) \cos\left(\frac{\pi k}{m+1}(0)\right)$$

So $-2U_0 + 2U_1 = h^2 f_0$, becomes;

$$2\sum_{k=0}^{m+1}{}'' \hat{U}_k \left(2\cos\left(\frac{\pi k}{m+1}\right) - 2\right) \cos\left(\frac{\pi k}{m+1}(0)\right) = h^2 \sum_{k=0}^{m+1}{}'' \hat{f}_k \cos\left(\frac{\pi 0 k}{m+1}\right)$$

In case $j = m+1$

from (2), we have $\quad 2U_m - 2U_{m+1} = h^2 f_{m+1}$

but $U_m = U_{m+2}$, from (7)

then

$$2U_{m+2} - 2U_{m+1} = h^2 f_{m+1}$$

$$4\sum_{k=0}^{m+1}{}'' \hat{U}_k \cos\left(\frac{\pi(m+2)}{m+1}k\right) - 4\sum_{k=0}^{m+1}{}'' \hat{U}_k \cos\left(\frac{(m+1)k}{m+1}\right) \cancel{= h^2 \sum_{k=0}^{m+1}{}'' \hat{f}_k \cos}$$

$$= h^2 \sum_{k=0}^{m+1}{}'' \hat{f}_k \left(\frac{\pi(m+1)k}{m+1}\right)$$

This reduces to

$$2\sum_{k=0}^{m+1}{}'' \hat{U}_k \left[2\cos\left(\frac{\pi(m+2)k}{m+1}\right) - 2\cos\left(\frac{\pi(m+1)}{m+1}k\right)\right] = h^2 \sum_{k=0}^{m+1}{}'' \hat{f}_k \left(\frac{\pi(m+1)k}{m+1}\right)$$

but

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

Since $\sin \pi k = 0$, for k, integer, then

$$\cos\left(\frac{\pi(m+2)}{m+1}k\right) = \cos\left(\frac{\pi(m+1)}{(m+1)}k\right)\cos\left(\frac{\pi k}{m+1}\right)$$

then;

$$2\sum_{k=0}^{m+1}{}'' \hat{U}_k \left[2\cos\left(\frac{\pi(m+1)}{m+1}k\right)\cos\left(\frac{\pi k}{m+1}\right) - 2\cos\left(\frac{\pi(m+1)}{m+1}k\right)\right] = h^2 \sum_{k=0}^{m+1}{}'' \hat{f}_k \left(\frac{\pi(m+1)k}{m+1}\right)$$

$$2\sum_{k=0}^{m+1}{}'' \hat{U}_k \left[2\cos\left(\frac{\pi k}{m+1}\right) - 2\right]\cos\left(\frac{\pi(m+1)}{m+1}k\right) = h^2 \sum_{k=0}^{m+1}{}'' \hat{f}_k \left(\frac{\pi(m+1)}{m+1}k\right).$$

b) from

$$\sum_{k=0}^{m+1}{}'' \hat{U}_k \left(2\cos\left(\frac{\pi k}{m+1}\right)-2\right)\cos\left(\frac{\pi j k}{m+1}\right) = h^2 \sum_{k=0}^{m+1}{}'' \hat{f}_k \cos\left(\frac{\pi j k}{m+1}\right)$$

for $k=0$:

$$\sum_{\cancel{k \neq 0}}^{m+1} \hat{U}_0 (2-2)\cos\left(\frac{\pi j (0)}{m+1}\right) = h^2 \sum \hat{f}_0$$

$$0 \neq \hat{f}_0$$, hence undefined, therefore we start from $k=1$

$$\sum_{k=1}^{m+1}{}'' \hat{U}_k \left(2\cos\left(\frac{\pi k}{m+1}\right)-2\right)\cos\left(\frac{\pi j k}{m+1}\right) = h^2 \sum_{k=1}^{m+1}{}'' \hat{f}_k \cos\left(\frac{\pi j k}{m+1}\right)$$

which reduces to

$$\hat{U}_k \left(2\cos\left(\frac{\pi k}{m+1}\right)-2\right) = h^2 \hat{f}_k$$

hence

$$\hat{U}_k = \frac{h^2 \hat{f}_k}{2\cos\left(\frac{\pi k}{m+1}\right)-2}$$

---

if $\hat{f}_0 = 0$, gives

$$\frac{1}{m+1}\left[\frac{1}{2}f_0 + \sum_{j=1}^{m} f_j \cos\left(\frac{\pi (0) j}{m+1}\right) + \frac{1}{2} f_{m+1}\right] = 0$$

$$\left[\frac{1}{2}f_0 + \sum_{j=1}^{m} f_j + \frac{1}{2} f_{m+1}\right] = 0 \qquad \text{——— } 1$$

~~thus~~ for the discrete Compatibility Condition

$$w^T b = w^T f = 0.$$

$$w^T = \left[\tfrac{1}{2}, 1, \ldots, 1, \tfrac{1}{2}\right]$$

therefore;

$$W^T f = [\tfrac{1}{2}, 1, \cdots, 1, \tfrac{1}{2}] \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{m+1} \end{bmatrix}$$

$$W^T f = \tfrac{1}{2} f_0 + f_1 + f_2 + \cdots f_{m-1} + f_m + \tfrac{1}{2} f_{m+1}$$

but from ①, $\tfrac{1}{2} f_0 + \sum_{j=1}^{m} f_j + \tfrac{1}{2} f_{m+1} = 0$ for

$\hat{f}_0 = 0$, therefore;

$$W^T f = \tfrac{1}{2} f_0 + \sum_{j=1}^{m} f_j + \tfrac{1}{2} f_{m+1} = 0$$

Hence $\hat{f}_0 = 0$ corresponds to $W^T b = W^T f = 0$

---

c) - To obtain the solution for (2), the discrete compatibility condition $W^T f = 0$ must be satisfied.

- So since its satisfied on the right hand side, hence we can obtain the solution to (2)

- So for $W^T f = 0$ to be satisfied on the right hand means the non ~~zero~~ ~~eigen~~ vector ~~value~~ ~~eigen~~ is orthogonal to f, hence their dot product is zero.

Also explain how one makes the solution unique by fixing the arbitrary constant to U

let $\hat{U}_0$ be fixed to an arbitrary constant U i.e. $\hat{U}_0 = U$

but
$$\hat{U}_0 = \frac{1}{m+1}\left[\frac{1}{2}U_0 + \sum_{j=1}^{m} U_j + \frac{1}{2}U_{m+1}\right] = U$$

$$\frac{1}{2}U_0 + \sum_{j=1}^{m} U_j + \frac{1}{2}U_{m+1} = (m+1)U.$$

but
$$\frac{1}{2}U_0 + \sum_{j=1}^{m} U_j + \frac{1}{2}U_{m+1} = \begin{bmatrix} \frac{1}{2} & 1 & \cdots & 1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ \vdots \\ U_{m+1} \end{bmatrix}$$

$$= w^T U$$

then
$$\frac{1}{2}U_0 + \sum_{j=1}^{m} U_j + \frac{1}{2}U_{m+1} = w^T U = (m+1)U$$

So $w^T U = (m+1)U$ implies that the solution to (2) is unique.

---

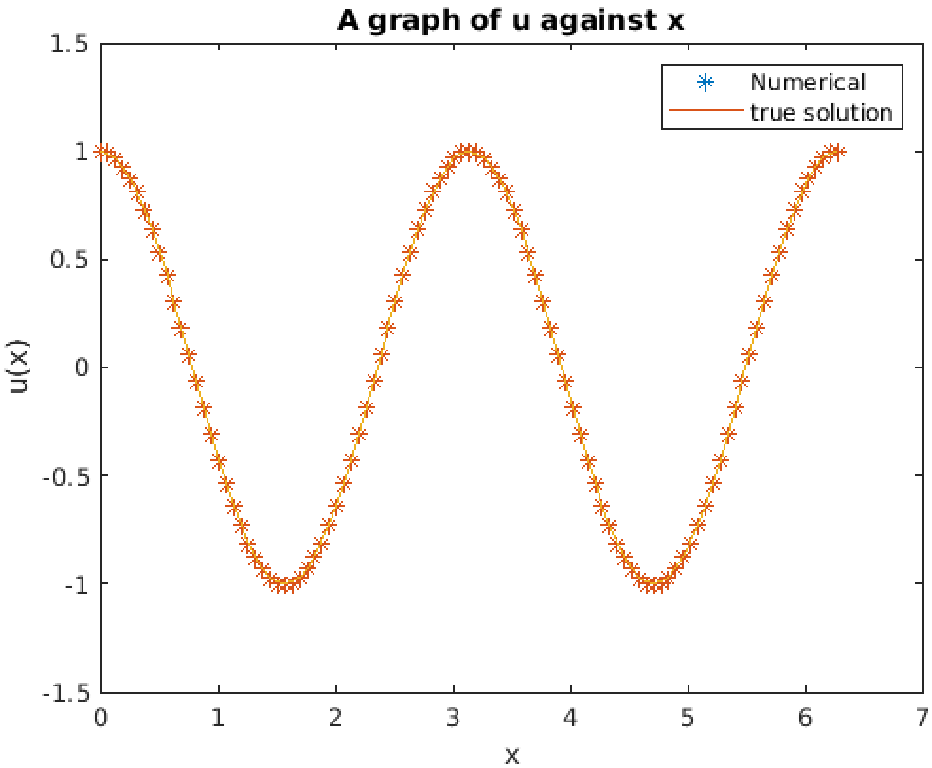
e) First of all they are mathematically equivalent since in both we are solving the same equation, and the conditions in both methods almost draw to the same conclusion.
- In problem (3), we are interested more in the value of lambda, λ. if its zero ($\lambda = 0$) then the solution exists, and also the error gives some information

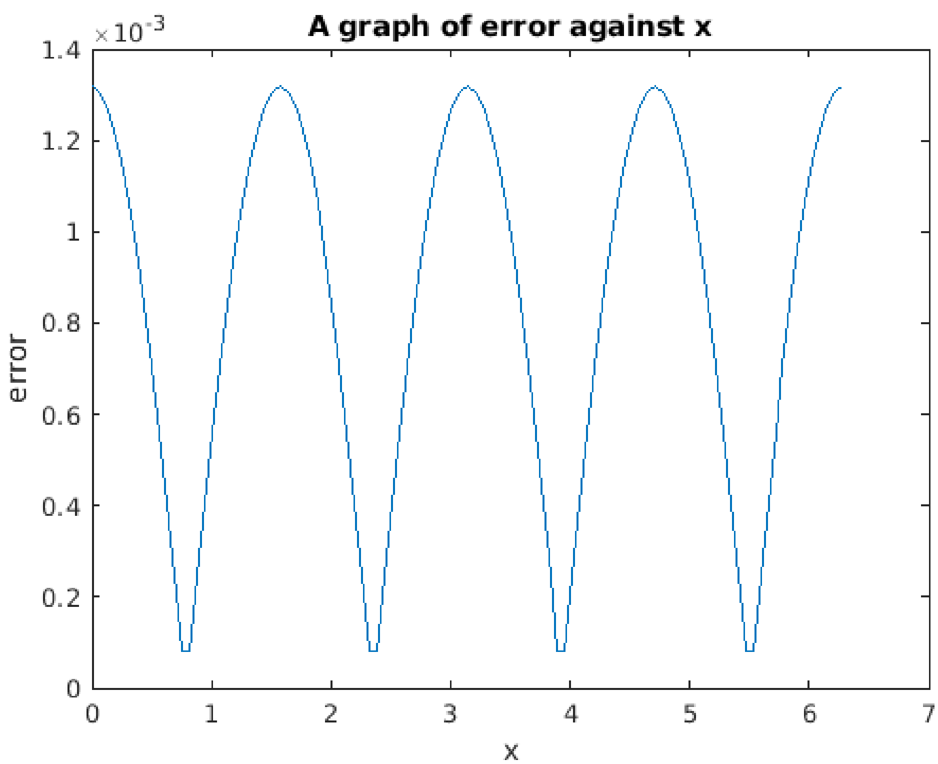
- As in problem (4) we see that the Poisson equation doesn't have a solution unless the discrete compatibility condition is satisfied, and that the solution is unique if u_0 is fixed to u.
- So all these methods will draw to same equivalent solutions.

```
%The program uses idct and dct ad procedures (a)-(c) to solve problem from
%4(c)

a=0; b=2*pi;
m=99;

h=(b-a)/(m+1);
j=[0:m+1]';
xj=a+j*h;
k=[1:m+1]';

%take v(0) to be 0.000002 since at k=0, ucap is undefined, so ucap(0) can be
%choosen arbitrary.
v=[0.000002;(2*cos((pi*k)/(m+1)))-2];
f=-4*cos(2*xj);

%obtaining fcap
fcap=dct(f);

%obtaining ucap
ucap=(h^2)*fcap./v;

%obtaining u
uap=idct(ucap);


%relative two norm
L2norm=RelL2Norm(uex,uap);
fprintf('%10s %16.8e\n','Relative two norm =',L2norm);
fprintf('According to the results from the two graphs, we can conclude that the results are the same.');

%ploting the solution of u
figure(1);
plot(xj,uap,'*');
hold on;
uex=u_ex(xj);
plot(xj,uex);
legend( 'Numerical','true solution')
ylabel('u(x)');
xlabel('x');
title('A graph of u against x');

figure(2);
err=er(uex,uap);
plot(xj,err);
ylabel('error');
xlabel('x');
title('A graph of error against x');


%exact solution
function uexact=u_ex(xj)
uexact=cos(2*xj);
end

%error
function error=er(uex,uap)
error=abs(uex - uap);
end

%relative two norm of the error
function L2 = RelL2Norm(uex,uap)
```

```
R = (uex - uap).^2;
L2 = sqrt(sum(R)/sum(uap.^2));
end
```

```
Relative two norm =   1.31525476e-03
According to the results from the two graphs, we can conclude that the results are the same.
```





Published with MATLAB® R2020a