# spectral1

February 2, 2018

#### 0.0.1 MWR Pseudospectral 1

Using Pseudospectral methods to numerically approximate differential equation solutions.

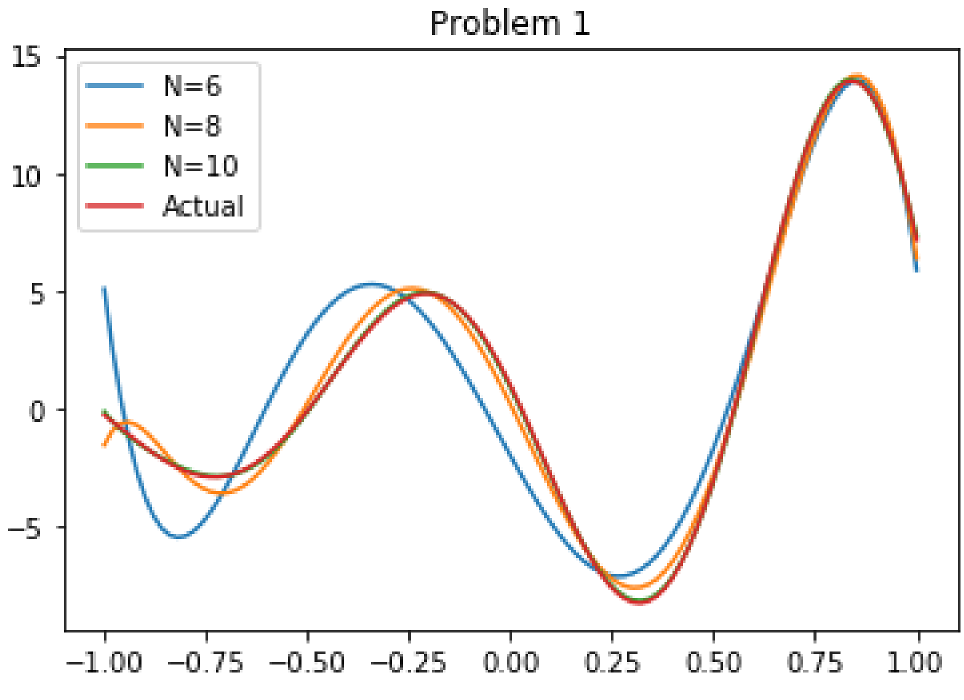
```
In [1]: import numpy as np
        from scipy.interpolate import barycentric_interpolate
        from scipy.optimize import root
        from matplotlib import pyplot as plt
        from scipy import linalg as la
        from mpl_toolkits.mplot3d import Axes3D
```

# 1 Problem 1

```
In [2]: def cheb(N):
            x = np.cos((np.pi/N)*np.linspace(0,N,N+1))
            x.shape = (N+1,1)
            lin = np.linspace(0,N,N+1)
            lin.shape = (N+1,1)
            c = np.ones((N+1,1))
            c[0], c[-1] = 2., 2.
            c = c*(-1.)**lin
            X = x*np.ones(N+1) # broadcast along 2nd dimension (columns)
            dX = X - X.T
            D = (c*(1./c).T)/(dX + np.eye(N+1))
            D = D - np.diag(np.sum(D.T,axis=0))
            x.shape = (N+1,)
            # Here we return the differentiation matrix and the Chebyshev points,
            # numbered from x_0 = 1 to x_N = -1
            return D, x
```

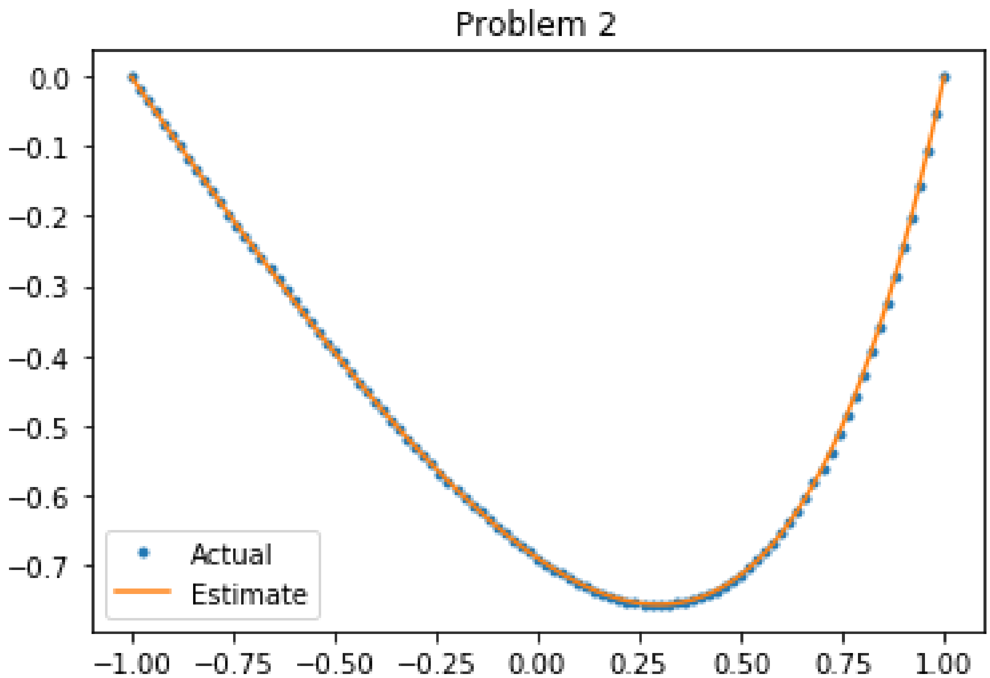
```
In [3]: u=lambda x: np.e**x*np.cos(6*x)
        du=lambda x: np.e**x*np.cos(6*x)-6*np.e**x*np.sin(6*x)
        domain=np.linspace(-1,1,101)

        for N in [6,8,10]:
            D,x=cheb(N)
            f=barycentric_interpolate(x,D.dot(u(x)),domain)
```

```
    plt.plot(domain,f,label="N={}".format(N))
plt.plot(domain,du(domain), label="Actual")
plt.title("Problem 1")
plt.legend()
plt.show()
```



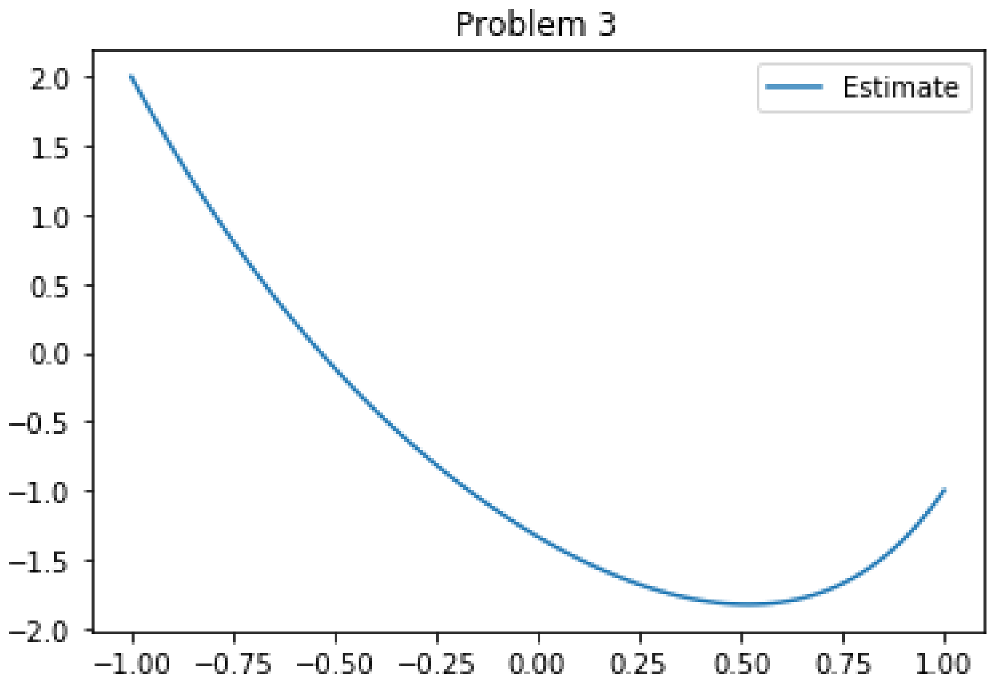

## 2 Problem 2

```
In [4]: D,x=cheb(8)
        u=lambda x: (-np.cosh(2)-np.sinh(2)*x+np.e**(2*x))/4
        d2u=lambda x: np.e**(2*x)
        domain=np.linspace(-1,1,101)
        D2=np.dot(D,D)
        D2[0,:], D2[-1,:] = 0, 0
        D2[0,0], D2[-1,-1] = 1, 1
        F=d2u(x)
        F[0], F[-1] = 0, 0
        sol=la.solve(D2,F)
        f=barycentric_interpolate(x,sol,domain)
        plt.plot(domain, u(domain),'.',label='Actual')
        plt.plot(domain, f,label='Estimate')
        plt.legend()
        plt.title("Problem 2")
        plt.show()
```



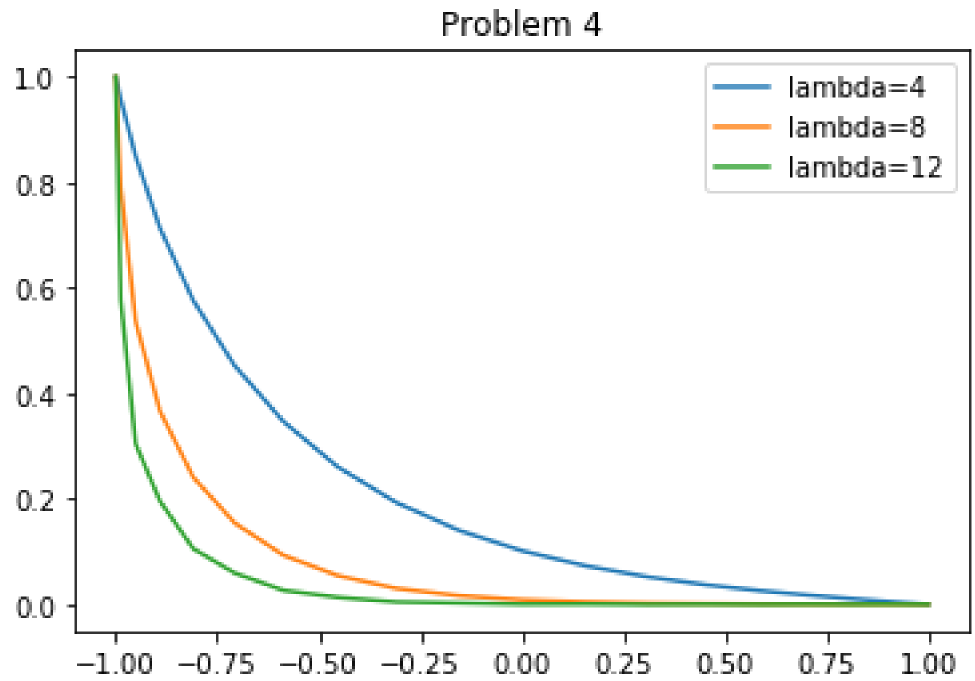

## 3 Problem 3

In [5]:
```
D,x=cheb(8)
u=lambda x: np.e**(3*x)
real=lambda x: (np.e**(-(x-2))-36*np.e**(1-x)+np.e**(38*x)-\
                np.e**(3*x+2)+24-1/np.e**3+12*np.e**2+np.e**5)/\
(12-12*np.e**2)
domain=np.linspace(-1,1,101)
D2=np.dot(D,D)+D
D2[0,:], D2[-1,:] = 0, 0
D2[0,0], D2[-1,-1] = 1, 1
F=u(x)
F[0], F[-1] = -1,2
sol=la.solve(D2,F)
f=barycentric_interpolate(x,sol,domain)
plt.plot(domain, f,label='Estimate')
#plt.plot(domain, real(domain))
plt.title("Problem 3")
plt.legend()
plt.show()
```



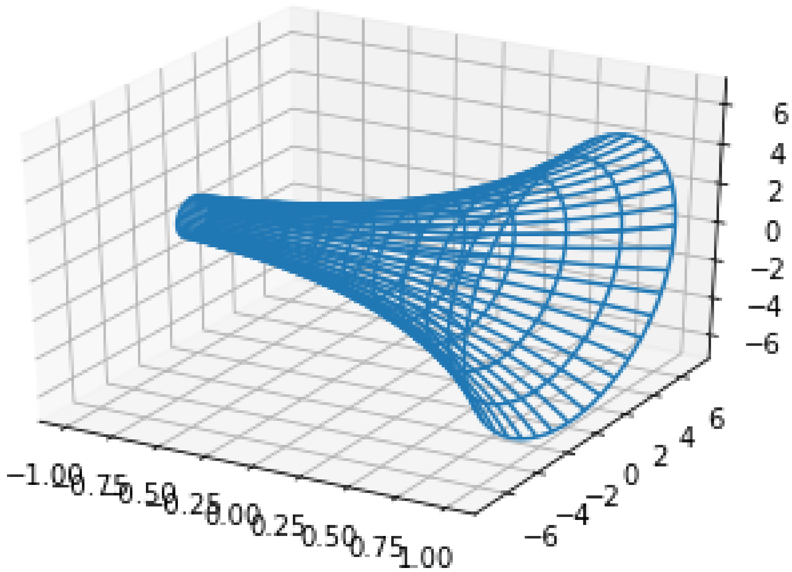

## 4 Problem 4

In [16]:
```
N=20
D,x=cheb(N)
u=lambda x: lam*np.sinh(lam*x)

def F(U):
    out=4*D.dot(D).dot(U)-lam*np.sinh(lam*U)
    out[0]=U[-1]-1
    out[-1]=U[0]
    return out

guess=np.ones_like(x)
domain=np.linspace(-1,1,101)

for lam in [4,8,12]:
    solution=root(F,guess).x
    plt.plot(x,solution,label="lambda={}".format(lam))
plt.legend()
plt.title("Problem 4")
plt.show()
```




## 5 Problem 5

In [12]:
```
barycentric=None

def F(U):
    out=U*D.dot(D).dot(U)-(D.dot(U)*D.dot(U))-np.ones_like(U)
    out[0]=U[0]-7
    out[-1]=U[-1]-1
    return out

D,x=cheb(50)
guess=2*np.ones_like(x)
solution=root(F,guess).x
lin=np.linspace(-1,1,100)
barycentric=barycentric_interpolate(x,solution,lin)

theta = np.linspace(0,2*np.pi,401)
X, T = np.meshgrid(lin, theta)
Y, Z = barycentric*np.cos(T), barycentric*np.sin(T)

fig = plt.figure()
ax = fig.gca(projection="3d")
ax.plot_wireframe(X,Y,Z, rstride=10, cstride=10)
plt.show()
```




In [ ]: