```python
# -*- coding: utf-8 -*-
"""
Created on Wed Feb  3 01:40:44 2021

@author: 11327
"""

import numpy as  np
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy.special import eval_legendre
from scipy.optimize import linprog


# 1a

x = np.linspace(-3,3)
fX = 1/np.sqrt(2*np.pi) * np.exp(-(x*x) / 2)

plt.plot(x,fX)
plt.xlabel('x')
plt.ylabel('fX(x)')
plt.savefig('D:/phd2/ece595/1a.png')
plt.show()

# 1b
x = np.random.normal(0, 1, 1000)

plt.figure(1)
hp1 = plt.hist(x,4)

plt.figure(2)
hp2 = plt.hist(x,1000)

m,sd = norm.fit(x)
plt.figure(3)
plt.plot(hp1[1],norm.pdf(hp1[1]))
plt.figure(4)
plt.plot(hp2[1],norm.pdf(hp2[1]))

# 1c
x = np.random.normal(0, 1, 1000)
n = 1000
maxx = np.max(x)
minx = np.min(x)
diff = maxx - minx
# pj = x[0:200:1]
# sumpj = np.sum(pj**2)
J = []
M = np.linspace(1,200,200)
for m in M:
    m = int(m)
    pj = (np.histogram(x,m))[0] / np.sum((np.histogram(x,m))[0])
    sumpj = np.sum(pj**2)
    Jh = 2/((n-1)*(diff/m)) - (n+1)/((diff/m)*(n-1))*sumpj
    J = np.append(J,Jh)

plt.figure(1)
plt.plot(M,J)

Jmin = np.min(J)
```

```
mmin = np.where(J==Jmin)

plt.figure(2)
hp1 = plt.hist(x,int(mmin[0]))
plt.figure(3)
plt.plot(hp1[1],norm.pdf(hp1[1]))

# 2a
x1 = np.linspace(-1,5,100)
x2 = np.linspace(0,10,100)
X1,X2 = np.meshgrid(x1,x2)
fX = (1/np.sqrt(3*(2*np.pi)**2))*np.exp((2*X1**2 + 4*X1 - 2*X1*X2 + 2*X2**2+56-20*X2)/(-6))

plt.figure(1)
plt.contour(X1,X2,fX)

# 2c
x = np.random.multivariate_normal([0, 0], [[1, 0], [0, 1]], 5000)
plt.figure(1)
plt.scatter(x[:, 0], x[:, 1])

X = np.random.multivariate_normal([2, 6], [[2, 1], [1, 2]], 5000)
plt.figure(2)
plt.scatter(X[:, 0], X[:, 1])

S = [[2, 1], [1, 2]]
b = [2,6]
L,U = np.linalg.eig(S)
A = np.dot(np.dot(U,np.diag(L**(0.5))),U.T)
y = np.dot(A,x.T)
y1 = y[0,:] + 2
y2 = y[1,:] + 6
plt.figure(3)
plt.scatter(y1, y2)
y = [y1,y2]

# 3a
b0 = -0.001
b1 = 0.01
b2 = 0.55
b3 = 1.5
b4 = 1.2
x = np.linspace(-1,1,50) # 50 points in the interval [-1,1]
L1 = eval_legendre(1,x)
L2 = eval_legendre(2,x)
L3 = eval_legendre(3,x)
L4 = eval_legendre(4,x)
error = np.random.normal(0,0.2,50)
y = b0 + b1*L1 + b2*L2 + b3*L3 + b4*L4 + error


X = np.column_stack((eval_legendre(0,x), eval_legendre(1,x), \
                     eval_legendre(2,x), eval_legendre(3,x), \
                     eval_legendre(4,x)))
theta = np.linalg.lstsq(X, y, rcond=None)[0]
t     = np.linspace(-1, 1, 200);
yhat  = theta[0]*eval_legendre(0,t) + theta[1]*eval_legendre(1,t) + \
        theta[2]*eval_legendre(2,t) + theta[3]*eval_legendre(3,t) + \
        theta[4]*eval_legendre(4,t)

plt.figure(1)
```

```
plt.scatter(x, y)
plt.plot(t,yhat,'g', linewidth=2)
plt.show()

idx = [10,16,23,37,45] # these are the locations of the outliers
y[idx] = 5 # set the outliers to have a value 5
plt.figure(2)
plt.scatter(x, y)
plt.plot(t,yhat,'g', linewidth=2)
plt.show()


N = 50
for p in range(5):
  X[:,p] = eval_legendre(p,x)

Au = np.hstack((X,-np.eye(N)))
Al = np.hstack((-X,-np.eye(N)))
A = np.vstack((Au,Al))
b = np.hstack((y,-y))
c = np.hstack((np.zeros(5),np.ones(N)))

sol = linprog(c,A,b,bounds=(None,None))
beta = sol.x[0:5]

t2     = np.linspace(-1, 1, 50);
yhat2  = beta[0]*eval_legendre(0,t2) + beta[1]*eval_legendre(1,t2) + \
         beta[2]*eval_legendre(2,t2) + beta[3]*eval_legendre(3,t2) + \
         beta[4]*eval_legendre(4,t2)
plt.plot(t2,yhat2,'r', linewidth=2)
plt.show()
```