STA414 assignment 3

Yubo Wang 1002138377

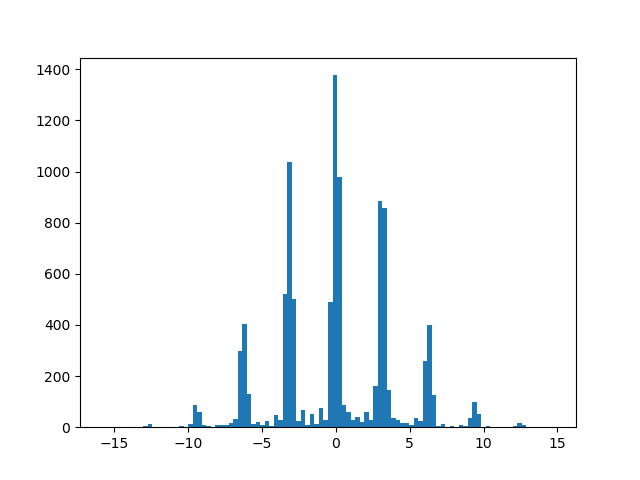


Code:

import numpy as np  
from scipy.stats import norm  
import matplotlib.pyplot as plt  
from numpy import sin, sqrt, exp, pi  
  
def d0(x):  
 return (1 - (sin(5 \* x)) \*\* 2 / (25 \* (sin(x) \*\* 2))) / sqrt(32 \* pi) \* exp(-x \*\* 2 / 32)  
  
def d1(x):  
 return (sin(5 \* x)) \*\* 2 / (25 \* (sin(x) \*\* 2)) / sqrt(32 \* pi) \* exp(-x \*\* 2 / 32)  
  
def d2(x):  
 return (sin(5 \* x))\*\* 2 / (25 \* (sin(x) \*\* 2))  
  
  
def q1a():  
 x = np.linspace(-20, 20, 10000)  
 y = (1 - sin(5 \* x) \*\* 2 / (25 \* sin(x) \*\* 2 )) / sqrt(32 \* pi) \* exp(-x\*\*2 / 32)  
 fig = plt.figure()  
 plt.scatter(x, y, s=1)  
 plt.show()  
 estimation = np.sum(y)  
 return estimation \* 40 / 10000  
  
  
def q1b():  
 accept = []  
 count = 0  
 total = 0  
 while count <= 10000:  
 s = np.random.uniform(0, 1)  
 t = np.random.uniform(-20, 20)  
 y = (sin(5\*t)\*\*2) / (25 \* sin(t) \*\* 2) / sqrt(32 \* pi) \* exp(-t\*\*2/32)  
 if s <= y:  
 accept.append(t)  
 count += 1  
 total += 1  
 plt.hist(accept, 100)  
 plt.show()  
 return count / total  
  
  
def q1c():  
 w = np.random.normal(0, scale=4, size=1000)  
 denom = 0  
 for i in range(1000):  
 x = w[i]  
 y1 = d0(x)  
 denom += y1 / norm.pdf(x, 0, scale=4)  
 result = 0  
 for i in range(1000):  
 x = w[i]  
 result += (1-d2(x))\*d0(x) / norm.pdf(x, 0, scale=4) / denom  
 return result  
  
  
def density(theta):  
 px = norm.pdf(1.7, loc=theta, scale=4)  
 pg = (np.sin(5 \* (1.7 - theta)) \*\* 2) / (25 \* sin(1.7 - theta) \*\* 2)  
 return px\*pg/((10\*pi)\*(1+(theta/10)\*\*2))  
  
  
def q1d():  
 theta = np.linspace(-20, 20, 10000)  
 d = density(theta)  
 plt.plot(theta, d)  
 plt.show()

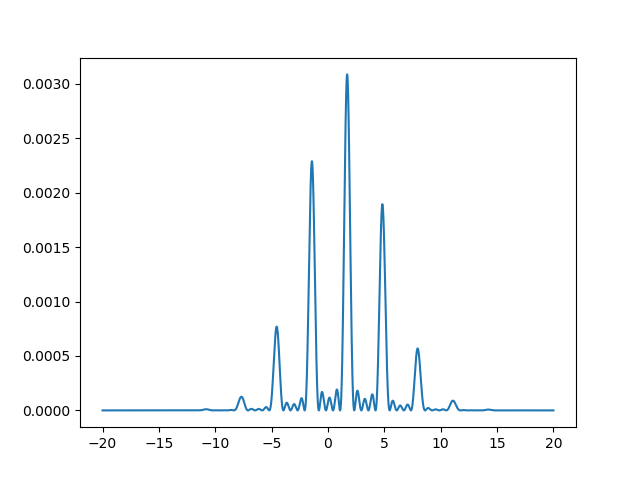
def metropolis(x, iter):  
 lst = []  
 for i in range(iter):  
 u = np.random.normal(x, scale=4)  
 accept = min(1, density(u)/density(x))  
 r = np.random.uniform(0, 1)  
 if r <= accept:  
 x = u  
 lst.append(x)  
 return lst  
  
def q1e():  
 lst = metropolis(0, 10000)  
 plt.hist(lst, 100)  
 plt.show()  
  
def q1f():  
 lst = metropolis(0, 10000)  
 count = 0  
 for i in lst:  
 if -3 < i < 3:  
 count += 1  
 print(count / 10000)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(q1a())  
 print(q1b())  
 print(q1c())  
 q1d()  
 q1e()  
 q1f()

1. 0.799919471407323

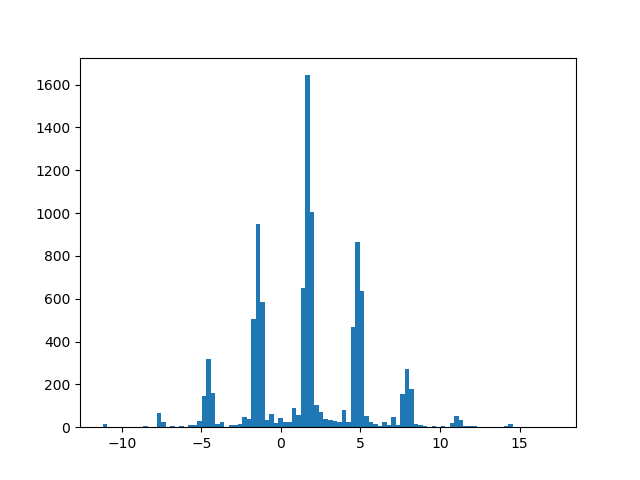


Accepted rate: 0.00499826327718346

1. 0.9253319948922147











0.6212



First apply the log derivation, then exchange the order of differentiation and integration.



Therefore, the reinforce is unbiased.



from part a and part b

Therefore, the reinforce with a fixed baseline is unbiased.



Let

Therefore, the reinforce is biased.







where

since



# since ,

# based on hint, for



*,* size (1xD)