Name: Deepika Kanade
Assignment 4- EE 511

Q 1)

Task: To estimate the value of pi by finding number of points lying in a quarter circle.

Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Apr  8 23:46:55 2018

@author: deepikakanade
"""
import numpy as np
import matplotlib.pyplot as plt

#####################################Question
1########################################
#pi-Estimation for 1 iteration
x_array = (np.random.uniform(0,1,100) **2)
y_array = (np.random.uniform(0,1,100) **2)
summation = x_array + y_array
print('Area of the inscribed quarter circle:', sum(summation<=1))


pi = []
for k in range (0,50):
   x_array = (np.random.uniform(0,1,100) **2)
   y_array = (np.random.uniform(0,1,100) **2)
   summation = x_array + y_array
   pi_value = sum(summation<=1)*4/100
   pi.append(pi_value)
variance=np.var(pi)
pi_value = sum(summation<=1)*4/100
print('Estimated value of pi: ', pi_value)
plt.figure()
plt.hist(pi)
```

```
plt.title('Samples(n)=100 and K=50')
plt.xlabel('Estimated Pi Value')
plt.ylabel('Count')

#pi-Estimation for k=50 iterations
n=[10, 50, 100, 500, 1000, 5000, 10000]
variance = []
mean=[]
for i in n:
   pi_1 = []
   for k in range (0,50):
      x_array = (np.random.uniform(0,1,i) **2)
      y_array = (np.random.uniform(0,1,i) **2)
      summation = x_array + y_array
      pi_value = sum(summation<=1)*4/i
      pi_1.append(pi_value)
   variance.append(np.var(pi_1))
   mean.append(np.mean(pi_1))

plt.figure()
plt.plot(n,variance)
plt.xlabel('n')
plt.ylabel('Variance')
plt.title('Plot of variance values for different n')


print('\nMean for all different values of n: ')
print(mean)

print('\nVariance for all different values of n: ')
print(variance)
```

Outputs:

Figure 1: Histogram for 100 samples and 50 iterations

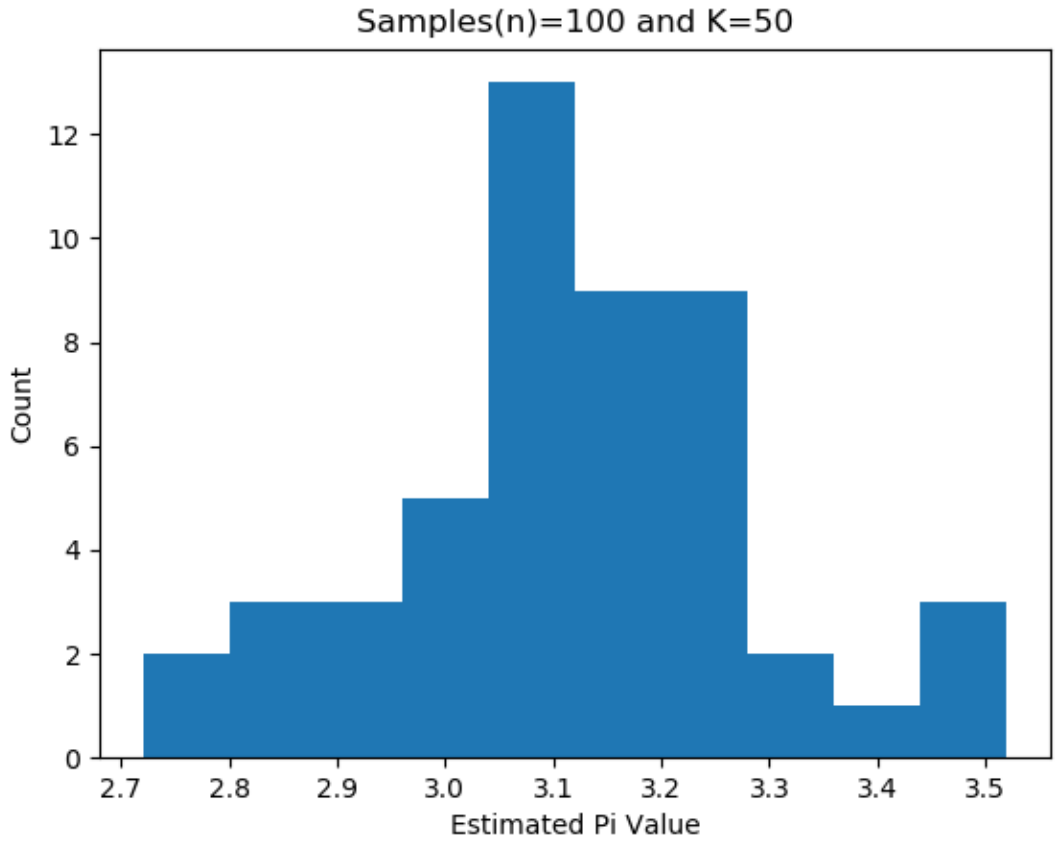


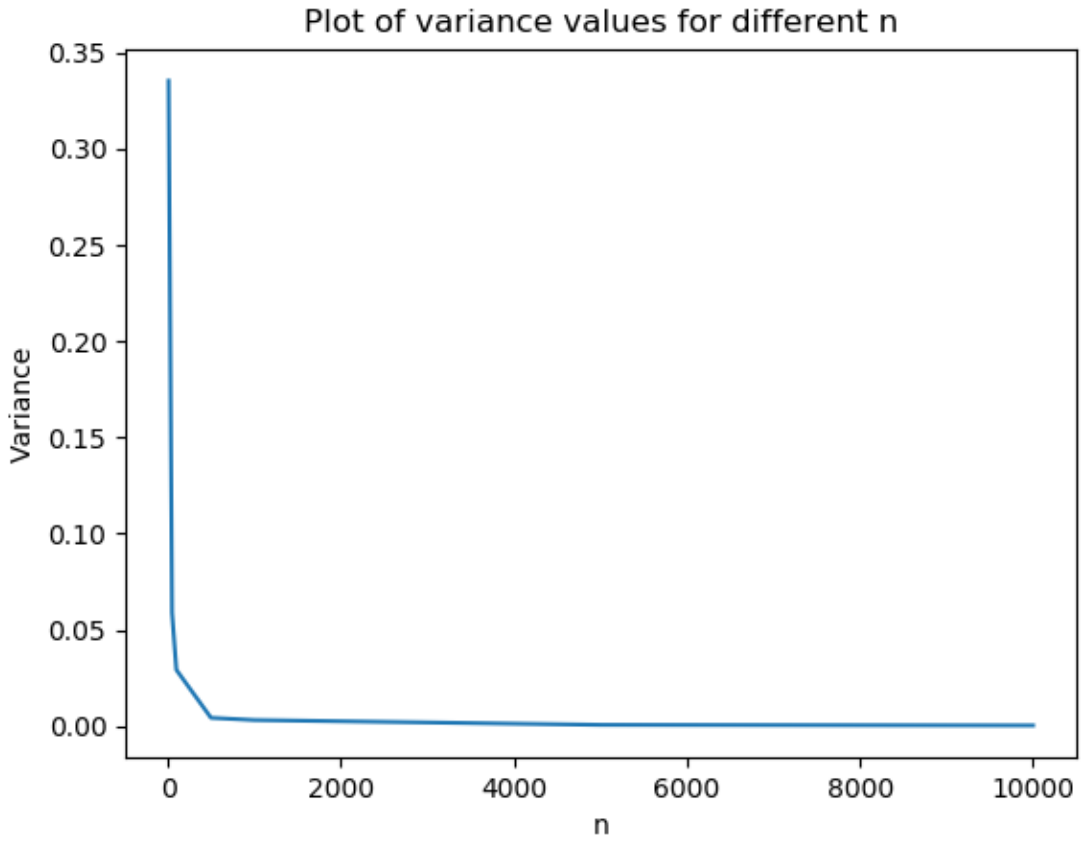
Figure 2: Final Output Generated

```
Area of the inscribed quarter circle: 80
Estimated value of pi:  3.2

Mean for all different values of n:
[3.1600000000000001, 3.1760000000000002, 3.1096000000000004, 3.15632,
3.1426400000000005, 3.142608000000001, 3.1398160000000006]

Variance for all different values of n:
[0.2960000000000001, 0.059711999999999987, 0.028467840000000005,
0.0068178176000000023, 0.0026992704000000029, 0.00062635033599999922,
0.00028801414399999938]
```

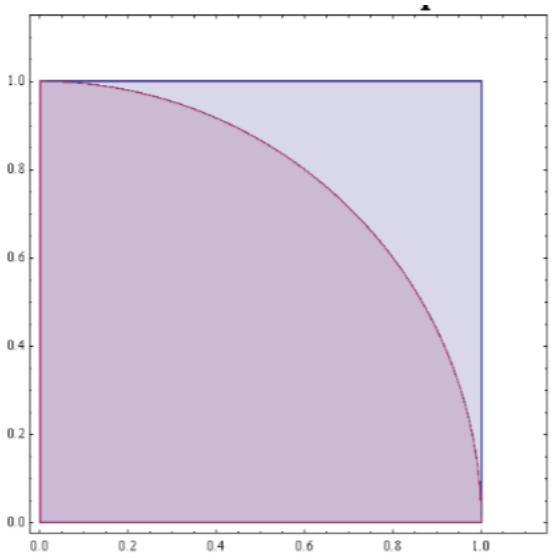
Figure 3: Plot of sample variance for different values of n



Procedure to estimate the value of pi:

Step 1: Generate random uniform 2D samples and substitute in the formula x^2+y^2 to calculate all the values.
Step 2: Discard the values which are x^2+y^2>=1 and multiply all the values by 4 to get the area of the entire circle.
Step 3: Run these for number of iterations and find the mean value of all iterations. This mean value will be the estimate of pi.
Step 4: Find the standard deviation with respect to all the values to get the variance.
Step 5: Repeat these steps for different values of number of samples and plot the histograms.

Observations:



i) To estimate the value of pi, the area of quarter circle should be estimated. As seen from Figure 1, a lot values of pi are in the range 3.1 to 3.2. This proves that in 50 iterations, the estimated value of pi is between 3.1 to 3.2 which coincides with our assumption. As seen from Figure 2, after 50 iterations, the final estimated value of pi is 3.2 which is pretty close to the actual value of pi. Also, the area of inscribed quarter circle is 80 units square. To get the actual value of pi, the area of inscribed quarter circle is multiplied by 4.

ii) Different values of n are used like 10,50,100,500,1000,5000,10000. It can be seen from Figure 2 that as n increases, the estimated value of pi which is the mean for 50 iterations becomes closer to 3.142. This is because as number of samples increase, the estimation becomes better. Also, as number of samples increase, the variance reduces which can be seen from Figure 3. Hence, to get a good estimate of the integral, the number of samples can be increased or the standard deviation between all the estimates can be reduced.

Q. 2) Monte Carlo Integration and Variation Reduction Strategies

Task: To estimate the value of integrals using simple Monte Carlo method, Stratification and Importance Sampling

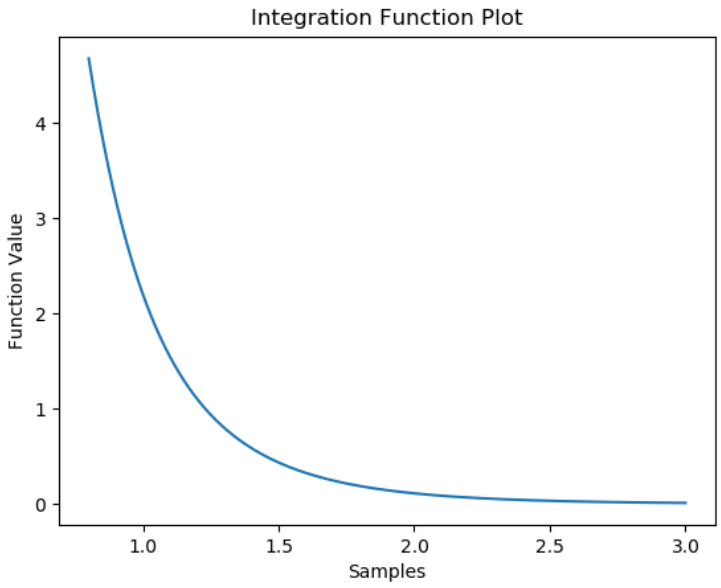
The integrals given are as follows:

(a) $[1+\sinh(2x)\ln(x)]^{-1}$, x in [0.8,3]

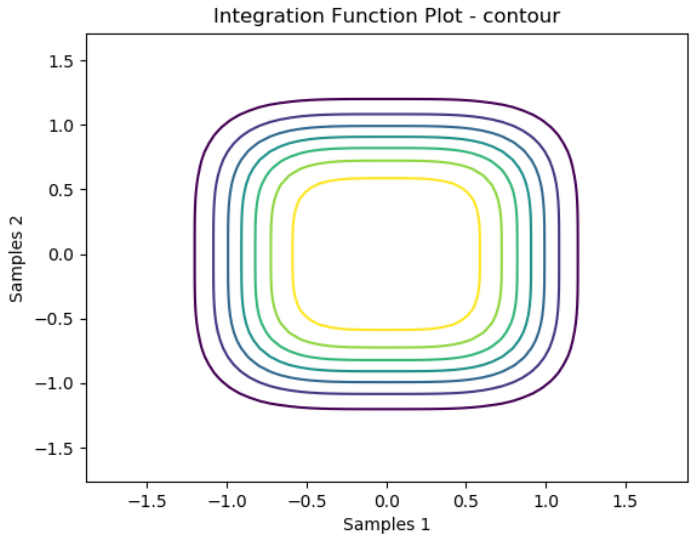
(b) $\text{Exp}[-x^4 - y^4]$, (x, y) in [-pi, pi]

The plots for the two functions are as shows below:

For Function 1:



For Function 2:



Monte Carlo estimation is performed to estimate the value of an integral by discretizing the function by taking different samples. To improve the estimation, either the variance can be reduced, or number of samples can be increased. There are two ways In which the variance can be reduced which are stratification and importance sampling.

Stratification: In Stratification, the integral is estimated by dividing the integral values into separate bins depending upon how the integral is changing in each bin.

Importance sampling: In importance sampling the estimate of the integral is found out by finding a dummy integral which estimates the given integral in the best way. By finding the dummy integral, the estimation of integral becomes closer to the actual value.

The procedure for each method is given below:

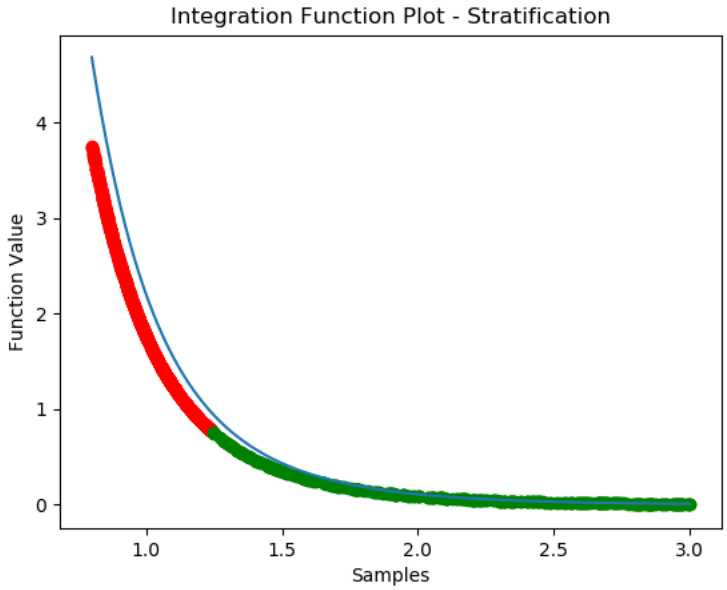
Procedure to do simple Monte Carlo Integral Estimation:

- Generate Samples from Uniform Distribution between given integral boundary values.
- Find the Function value for the generated samples.
- Get the mean of the function values (i.e.) Integral value.
- Iterate it max_iterations number of times.
- Every time append the mean integral value to a list. Multiply the integral value by the range of uniformly distributed samples.
- At the end get the mean of mean integral values.

- Also calculate the variance of mean integral values to find the quality of our integration.

Procedure to do simple Monte Carlo Integral Estimation using Stratification:

- Plot the given function between the given boundary values.
- Based on the function divide boundary values into various bins.
- Also decide allocation of number of samples to each bin based on the function plot. For example, in the given plot below for function 1, the number of samples are highly concentrated when the curve has direction changes. Very low number of samples are assigned to the places where information is less for integration



- Generate Samples from Uniform Distribution between given integral boundary values based on our bins.
- Find the Function value for the generated samples between different divided integral locations.
- Get the mean of the function values (i.e.) Integral value at different bins
- Get the summation of them to get final integral value
- Iterate it max_iterations number of times
- Every time append the final integral value to a list
- At the end get the mean of final integral values
- Also calculate the variance of final integral values to find the quality of our integration using stratification.

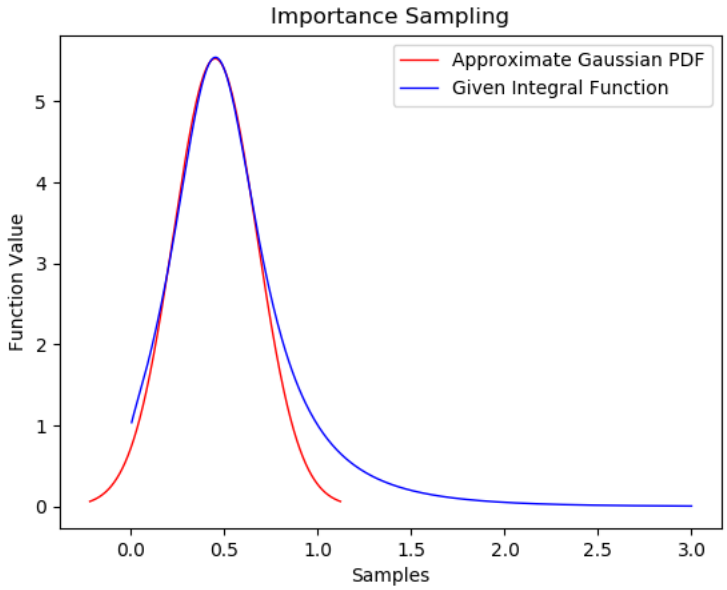
Procedure to do simple Monte Carlo Integral Estimation using Importance Sampling:

- Plot the given function between the given boundary values
- Find a pdf, that resembles similar to the plot function we have. This is called as Importance pdf. For instance, in both the functions given above, Gaussian pdfs were used as the importance pdf. The Gaussian pdf generated had different sigma and mean values.

$$\int h(x)p(x)dx = \int h(x)\frac{p(x)}{g(x)}g(x)dx = \int h(x)w(x)g(x)dx$$

- In the above equation, h(x) is the Integral function that we have to estimate, p(x) is the function that we assume to be Uniform Distribution (from theory behind Monte Carlo) and g(x) is our approximated Importance pdf.
- To proceed, generate samples from Importance pdf g(x)
- Based on the generated samples, estimate h(x), p(x) and g(x)
- Calculate the above expression h(x)*p(x) / g(x) within the integral
- Get the average of those values to get Mean Integral Value
- Iterate it max_iterations number of times
- Every time append the mean integral value to a list
- At the end get the mean of mean integral values
- Also calculate the variance of mean integral values to find the quality of our integration

The importance sampling for function 1 can be as sown below:



Here, the Gaussian pdf with mean=0.45 and standard deviation= 0.045 is used as the importance pdf.

Similarly, the importance sampling pdf for Function 2 is also Gaussian with mean=0 and standard deviation=0.5.

Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Apr  8 00:33:47 2018

@author: deepikakanade
"""

import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.stats import norm, uniform
from scipy.stats import multivariate_normal

########## STRATIFICATION
fcn=[]
stratified_function=[]
## Function 1
for i in range (0,50):
    x = np.random.uniform(0.8,3,1000)
    function=pow((1+(np.sinh(2*x) * np.log(x))),-1)
    fcn.append(((np.sum(function))/1000) * (3-0.8))

var1=np.var(fcn)
mean1=np.mean(fcn)
print('\nMean value for function 1 with simple Monte Carlo: ',mean1)
print('Variance value for function 1 with simple Monte Carlo',var1)

n1=750
n2=250
for i in range(0,50):
    x_1 = np.random.uniform(0.8,1.5,n1)
    function_1=pow((1+(np.sinh(2*x_1) * np.log(x_1))),-1)
    fcn_1=((np.sum(function_1))/n1 *(1.5 - 0.8))

    x_2 = np.random.uniform(1.5,3,n2)
    function_2=pow((1+(np.sinh(2*x_2) * np.log(x_2))),-1)
    fcn_2=((np.sum(function_2))/n2 * (3 - 1.5))

    stratified_function.append(fcn_1+fcn_2)

var2=np.var(stratified_function)
mean2=np.mean(stratified_function)
print('\nMean value for function 1 with stratification: ',mean2)
print('Variance value for function 1 with stratification',var2)

fcn2=[]
stratified_function2=[]
## Function 2
for i in range (0,50):
```

```
    x2= np.random.uniform(-3.142,3.142,(1000, 2))
    function2=np.exp(-pow(x2[:,0],4) - pow(x2[:,1],4))
    fcn2.append((np.sum(function2))/1000 * (3.142 -(-3.142)) * (3.142 -(-3.142)))

var1_f2=np.var(fcn2)
mean1_f2=np.mean(fcn2)


for i in range (0,50):
    x2_2= np.random.uniform(-1.7,1.7,(1000, 2))
    function2_2=np.exp(-pow(x2_2[:,0],4) - pow(x2_2[:,1],4))
    fcn2_2=(np.sum(function2_2))/1000 * 4 * 1.7 * 1.7

    stratified_function2.append(fcn2_2)

var2_f2=np.var(stratified_function2)
mean2_f2=np.mean(stratified_function2)


# Importance Sampling for Function 1
Sampling_Fnc1 = []
for i in range (0,50):
    mu = 0.45
    std = 0.045
    sigma = pow(std,2)
    X1 = np.random.normal(loc=mu, scale=sigma, size=10000)

    Fnc1 = pow((1 + (np.sinh(2*X1)*np.log(X1))),-1)*2.2

    Fnc1_dummy = norm.pdf(X1, loc=mu, scale=sigma)
    W1_x = uniform.pdf(X1, loc=0.8, scale=2.2)/(Fnc1_dummy)

    mul = Fnc1*W1_x
    mul = mul[~np.isnan(mul)]
    Sampling_Fnc1.append(np.mean(mul))

variance_impsampling_Fnc1=np.var(Sampling_Fnc1)
Mean_impsampling_Fnc1=np.mean(Sampling_Fnc1)

print('\nMean value for function 1 with important sampling', Mean_impsampling_Fnc1)
print('Variance value for function 1 with  important sampling: ', variance_impsampling_Fnc1)


print('\nMean value for function 2 with simple Monte Carlo: ',mean1_f2)
print('Variance value for function 2 with simple Monte Carlo',var1_f2)

print('\nMean value for function 2 with stratification: ',mean2_f2)
print('Variance value for function 2 with stratification',var2_f2)

# Importance Sampling for Function 2
def function_f(x,y):
    z = math.pow((math.pi*2),2) * (np.exp((-1 * math.pow(x,4)) + (-1 * math.pow(y,4))))
    return z
```

```
def function_p(x):
   z1 = uniform.pdf(x[0], loc=-math.pi, scale=math.pi)
   z2 = uniform.pdf(x[1], loc=-math.pi, scale=math.pi)
   return z1*z2

def function_g(x, mu, std):
   z = multivariate_normal.pdf(x, mean=mu, cov=std)
   return z

importance_sampling_estimates = []
mu = 0
std = 0.5

mu_mat = np.array([mu, mu])

for i in range(0,50):
   test_x3 = np.random.normal(loc=mu,size=1000)
   test_y3 = np.random.normal(loc=mu,size=1000)

   xx, yy = np.meshgrid(test_x3, test_y3)
   test_xy  =  np.hstack((xx.reshape(xx.shape[0]*xx.shape[1],  1,  order='F'),  yy.reshape(yy.shape[0]*yy.shape[1],  1,
order='F')))

   fx3 = math.pow((math.pi*2),2) * (np.exp((-1 * pow(test_xy[:,0],4)) + (-1 * pow(test_xy[:,1],4))))
   z1 = uniform.pdf(test_xy[:,0], loc=-math.pi, scale=math.pi+math.pi)
   z2 = uniform.pdf(test_xy[:,1], loc=-math.pi, scale=math.pi+math.pi)
   px3 = z1*z2
   gx3 = multivariate_normal.pdf(test_xy, mean=mu_mat)


   z3 = (np.array(fx3)) * (np.array(px3) / np.array(gx3))
   z3 = z3[~np.isnan(z3)]
   importance_sampling_estimates.append(np.mean(z3))

print('\nMean value for function 2 with important sampling', np.mean(importance_sampling_estimates))
print('Variance value for function 2 with  important sampling: ', np.var(importance_sampling_estimates))


last_fnc=[]
stratified_lastfunction=[]
## Function 3
for i in range (0,50):
   x3= np.random.uniform(-5,5,(1000,2))
   function3=20 + pow(x3[:,0],2) + pow(x3[:,1],2) - 10 * (np.cos(2 * math.pi * x3[:,0] ) + np.cos(2 * math.pi * x3[:,1]
))
   last_fnc.append((np.sum(function3))/1000 * (5 -(-5)) * (5 -(-5)))

var_lastFcn=np.var(last_fnc)
mean_lastFcn=np.mean(last_fnc)

print('\nMean value for function 3 with simple Monte Carlo: ',mean_lastFcn)
print('Variance value for function 3 with simple Monte Carlo',var_lastFcn)
```

```
n1=50
n2=50
n3=800
n4=50
n5=50
for i in range (0,50):

    x3_1= np.random.uniform(-5,5,n1)
    y3_1= np.random.uniform(-5,-2.5,n1)
    last_fnc_1=20 + pow(x3_1,2) + pow(y3_1,2) - 10 * (np.cos(2 * math.pi * x3_1 ) + np.cos(2 * math.pi * y3_1 ))
    fcn3_1=(np.sum(last_fnc_1))/n1 * 10 * 2.5

    x3_2= np.random.uniform(-5,5,n2)
    y3_2= np.random.uniform(-2.5,2.5,n2)
    last_fnc_2=20 + pow(x3_2,2) + pow(y3_2,2) - 10 * (np.cos(2 * math.pi * x3_2 ) + np.cos(2 * math.pi * y3_2 ))
    fcn3_2=(np.sum(last_fnc_2))/n2 * 10 * 5

    x3_3= np.random.uniform(-5,5,n3)
    y3_3= np.random.uniform(2.5,5,n3)
    last_fnc_3=20 + pow(x3_3,2) + pow(y3_3,2) - 10 * (np.cos(2 * math.pi * x3_3 ) + np.cos(2 * math.pi * y3_3 ))
    fcn3_3=(np.sum(last_fnc_3))/n3 * 10 * 2.5

    y3_4= np.random.uniform(-5,5,n4)
    x3_4= np.random.uniform(-5,-2.5,n4)
    last_fnc_4=20 + pow(x3_4,2) + pow(y3_4,2) - 10 * (np.cos(2 * math.pi * x3_4 ) + np.cos(2 * math.pi * y3_4 ))
    fcn3_4=(np.sum(last_fnc_4))/n4 * 10 * 2.5

    y3_5= np.random.uniform(-5,5,n5)
    x3_5= np.random.uniform(2.5,5,n5)
    last_fnc_5=20 + pow(x3_5,2) + pow(y3_5,2) - 10 * (np.cos(2 * math.pi * x3_5 ) + np.cos(2 * math.pi * y3_5 ))
    fcn3_5=(np.sum(last_fnc_5))/n5 * 10 * 2.5

    stratified_lastfunction.append(fcn3_1+fcn3_2+fcn3_3+fcn3_4+fcn3_5)

var_stratlastfunc=np.var(stratified_lastfunction)
mean2_stratlastfunc=np.mean(stratified_lastfunction)


print('\nMean value for function 3 with stratification: ',mean2_stratlastfunc)
print('Variance value for function 3 with stratification',var_stratlastfunc)
```

Outputs:

```
Mean value for function 1 with simple Monte Carlo:  0.604680539375
Variance value for function 1 with simple Monte Carlo 0.00123699968074

Mean value for function 1 with stratification:  0.609690615926
Variance value for function 1 with stratification 0.000222662475746

Mean value for function 1 with important sampling 0.721456174758
Variance value for function 1 with  important sampling:
0.000558295825768
```

```
Mean value for function 2 with simple Monte Carlo:  3.23741406523
Variance value for function 2 with simple Monte Carlo 0.0666637578691

Mean value for function 2 with stratification:  3.3029598645
Variance value for function 2 with stratification 0.0159838011978

Mean value for function 2 with important sampling 3.28882640939
Variance value for function 2 with  important sampling:
0.00785388926328
```

Type	**Mean (Estimate of Integral)**	**Variance (Variance in Estimates)**
Simple Monte Carlo (Function 1)	0.6046	0.0012
Simple Monte Carlo with Stratification (Function 1)	0.6096	0.00022
Simple Monte Carlo with Importance Sampling (Function 1)	0.7214	0.00055
Simple Monte Carlo (Function 2)	3.2374	0.667
Simple Monte Carlo with Stratification (Function 2)	3.3029	0.0159
Simple Monte Carlo with Importance Sampling (Function 2)	3.2888	0.0078

Observations:

If we observe the outputs from all the three methods, it can be seen that Monte Carlo estimation with stratification gives a better estimate to the integral value obtained from normal Monte Carlo estimation compared to the importance sampling method. However, the variance is the least when importance sampling is used, as the importance pdf helps to get accurate values for all the iterations. The variance of Importance Sampling should be less than any other methods. However, there is a trade-off for estimation when importance Sampling is used. Also, when the number of samples are increased, the estimation quality increases.

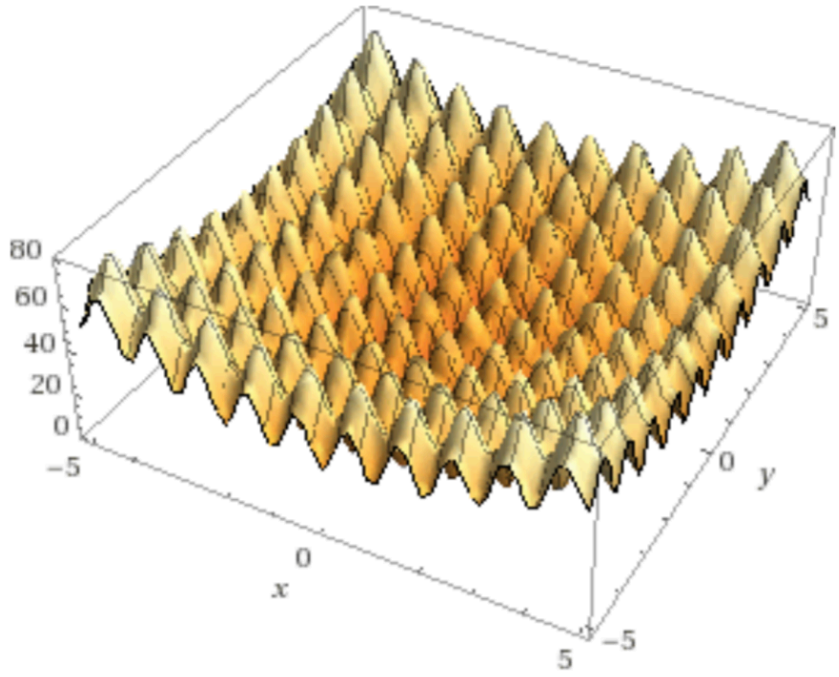
Type of sampling	Pros	Cons
Stratification	Accurate results can be obtained as it ensures that each of the sub group receives proper attention. The data can be divided in as many strata as we want.	It should accurately sort each member of the group and hence the degree of freedom is less.
Importance Sampling	This reduces the bias a lot as an importance pdf is used which approximates the given pdf in the best way.	The accuracy of estimation is not very proper as it all depends on the importance pdf chosen for sampling.

Part 3: To estimate the integral given below

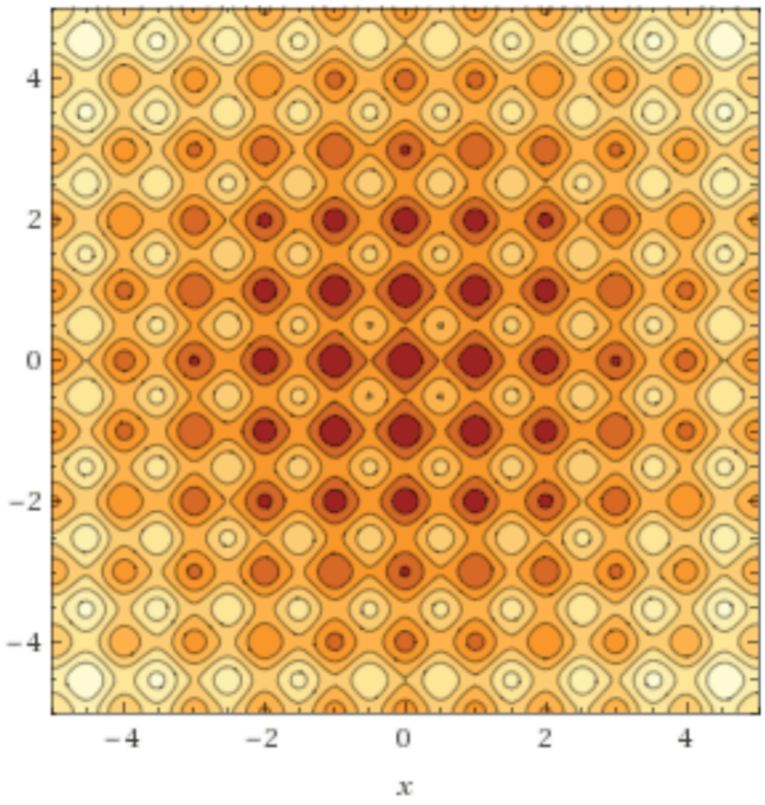
$$f(x,y) = 20 + x^2 + y^2 - 10(cos[2\pi \times x] + cos[2\pi \times y])$$

(x, y) in [-5, 5] for f(x,y)

To obtain the estimate of this integral, normal Monte Carlo estimation is done as well as Monte Carlo estimation using Stratification is done. The plot of the function is as given below:



The Contour plot is as shown below:



As seen above, the integral can be divided into five strata and different samples can be assigned to each strata.

Output:

```
Mean value for function 3 with simple Monte Carlo:  3667.80237881
Variance value for function 3 with simple Monte Carlo 2049.71678231

Mean value for function 3 with stratification: 3739.65702013
Variance value for function 3 with stratification: 540.989402
```

Observations:

As we can see, the estimate of the integral using stratification is pretty close to the actual value. Also, the variance has reduced considerably by using Stratification. As stratification is used for variance reduction, the values obtained prove that stratification is a better method to estimate value of any integral.