```
2 # IMPORTS
4 import numpy as np
5 import os
6 import matplotlib.pyplot as plt
7 import matplotlib as mpl
9 # Variable declaration
11 isPlotReqd = True
12 isPlotPdf = True
14 # Settings for plot
16 if (True == isPlotReqd):
17
   if (True == isPlotPdf):
18
     mpl.use('pdf')
19
     fig width = 3.487
20
     fig height = fig width / 1.618
21
     rcParams = {
22
      'font.family': 'serif',
23
      'font.serif': 'Times New Roman',
24
      'text.usetex': True,
25
      'xtick.labelsize': 8,
26
      'ytick.labelsize': 8,
27
      'axes.labelsize': 8,
28
      'legend.fontsize': 8,
29
      'figure.figsize': [fig width, fig height]
30
31
     plt.rcParams.update(rcParams)
33 # CODE STARTS HERE
34
35 n samples = 10000
36 \text{ n bins} = 100
37 phi = np.random.uniform(0, np.pi, n samples)
38 y hat = np.sin(phi)
39 y hat mean = np.mean(y hat)
40 y hat var = np.var(y hat)
41 print('UNIFORM Phi: Mean of y_hat = ', y_hat_mean)
42 print('UNIFORM Phi: Variance of y_hat = ', y hat var)
44 # Plot of histogram of y_hat: UNIFORM Phi
46 if (True == isPlotRegd):
47
   48
   # Configure axis and grid
49
   50
   fig = plt.figure()
51
   ax = fig.add subplot(111)
52
   fig.subplots adjust(left=.15, bottom=.16, right=.99, top=.97)
53
54
   ax.set axisbelow(True)
55
   ax.minorticks on()
   ax.grid(which='major', linestyle='-', linewidth='0.5')
56
57
   ax.grid(which='minor', linestyle="-.", linewidth='0.5')
58
   y linspace = np.linspace(0, 1, n samples, endpoint=False)
59
   y pdf = 2./(np.pi*np.sqrt(1-y linspace**2))
60
   ax.hist(y hat, bins=n bins, density=True, label='Monte Carlo')
   ax.plot(y_linspace, y_pdf, 'r', label='Analytical PDF')
61
```

```
62
63
     ax.set xlabel(r'\$\hat{y}\$', fontsize=8)
64
     ax.set ylabel(r'f_{\hat{Y}}(\hat{y})), fontsize=8)
65
66
     plt.legend()
67
     if (True == isPlotPdf):
68
       if not os.path.exists('./generatedPlots'):
69
        os.makedirs('generatedPlots')
70
       fig.savefig('./generatedPlots/q1_unif_phi_mc.pdf')
71
72
       plt.show()
74 # 1.d Gaussian phi
76 phi norm = np.random.normal(0, 1, n \text{ samples})
77 y hat norm = np.sin(phi norm)
78 y hat mean norm = np.mean(y hat norm)
79 y hat var norm = np.var(y hat norm)
80 print('NORMAL Phi: Mean of y hat = ', y hat mean norm)
81 print('NORMAL Phi: Variance of y_hat = ', y_hat_var_norm)
83 # Plot of histogram of y hat: NORMAL Phi
85 if (True == isPlotReqd):
     86
87
     # Configure axis and grid
     88
89
     fig = plt.figure()
90
     ax = fig.add subplot(111)
91
     fig.subplots adjust(left=.15, bottom=.16, right=.99, top=.97)
92
93
     ax.set axisbelow(True)
94
     ax.minorticks on()
95
     ax.grid(which='major', linestyle='-', linewidth='0.5')
96
     ax.grid(which='minor', linestyle=''-.", linewidth='0.5')
97
98
     ax.hist(y hat norm, bins=n bins, density=True, label='Monte Carlo: Normal Phi')
99
100
     ax.set xlabel(r'\$\hat{y}\$', fontsize=8)
101
     ax.set ylabel(r'\f {\hat{y}}(\hat{y})\', fontsize=8)
102
103
     plt.legend()
     if (True == isPlotPdf):
104
105
       if not os.path.exists('./generatedPlots'):
106
        os.makedirs('generatedPlots')
       fig.savefig('./generatedPlots/q1 norm phi mc.pdf')
107
108
     else:
109
       plt.show()
```