

```

1 #####
2 # IMPORTS
3 #####
4 import numpy as np
5 import os
6 import matplotlib.pyplot as plt
7 import matplotlib as mpl
8 #####
9 # Variable declaration
10 #####
11 isPlotReqd = True
12 isPlotPdf = True
13 #####
14 # Settings for plot
15 #####
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)
32 #####
33 # CODE STARTS HERE
34 #####
35 n_samples = 10000
36 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)
43 #####
44 # Plot of histogram of y_hat: UNIFORM Phi
45 #####
46 if (True == isPlotReqd):
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()
56     ax.grid(which='major', linestyle='-', linewidth='0.5')
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')
61     ax.plot(y_linspace, y_pdf, 'r', label='Analytical PDF')

```

```

62
63     ax.set_xlabel(r'$\hat{y}$', fontsize=8)
64     ax.set_ylabel(r'$\mathbf{Sf}_{\hat{Y}(\hat{y})}$', 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         else:
72             plt.show()
73     #####
74     # 1.d Gaussian phi
75     #####
76     phi_norm = np.random.normal(0, 1, n_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)
82     #####
83     # Plot of histogram of y_hat: NORMAL Phi
84     #####
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'$\mathbf{Sf}_{\hat{Y}(\hat{y})}$', fontsize=8)
102
103        plt.legend()
104        if (True == isPlotPdf):
105            if not os.path.exists('./generatedPlots'):
106                os.makedirs('generatedPlots')
107                fig.savefig('./generatedPlots/q1_norm_phi_mc.pdf')
108            else:
109                plt.show()

```