```python
import numpy as np
import random
import time

# the pdf
def pdf(x):
    a = 0.5535
    b = 1.0347
    c = 1.6214
    return a * np.exp(-x / b) * np.sinh(np.sqrt(c * x))

# the line pdf
def line_pdf(x, x1, y1, x2, y2):
    m = (y1 - y2) / (x1 - x2)
    c = y1 - x1 * (y1 - y2) / (x1 - x2)
    h_x =  m * x + c
    return h_x

# the line cdf
# def line_cdf(x, x1, y1, x2, y2):
#     m = (y1 - y2) / (x1 - x2)
#     c = y1 - x1 * (y1 - y2) / (x1 - x2)
#     F_x =  m * x**2 / 2 + c * x
#     return F_x

# inverse of the line cdf
def inv_line_cdf(x, x1, y1, x2, y2):
    m = (y1 - y2) / (x1 - x2)
    c = y1 - x1 * (y1 - y2) / (x1 - x2)
    Finv_x = - c / m + (np.sqrt(c**2 + 2 * m * x))/(m)
    return Finv_x

# Acceptance rejection method using triangle approach
def triangle_approach():
    uniform_rn = np.random.uniform(0, 1, 100000)

    prob_scaled_rn_1 = inv_line_cdf(uniform_rn, 0, 0.2, 10, 0)

    prob_scaled_rn_2_list = []

    for i in range(0, len(prob_scaled_rn_1)):
        c = 2
        h = line_pdf(prob_scaled_rn_1[i], 0, 0.2, 10, 0)
        u = np.random.rand()
        f = pdf(prob_scaled_rn_1[i])

        if u * c * h <= f:
            prob_scaled_rn_2_list.append(prob_scaled_rn_1[i])

        if len(prob_scaled_rn_2_list) >= 10000:
            break

    prob_scaled_rn_2 = np.array(prob_scaled_rn_2_list)

    mean_rn = np.average(prob_scaled_rn_2)
    var_rn = np.var(prob_scaled_rn_2)
    sd_rn = np.std(prob_scaled_rn_2)

    return prob_scaled_rn_2, mean_rn, var_rn, sd_rn
```

```
60
61
62  # PART 1
63  def run(seed):
64      np.random.seed(seed)
65
66      rns, mean_rns, var_rns, sd_rns = triangle_approach()
67      '''
68      print(f'mean of random numbers = {mean_rns}')
69      print(f'variance of random numbers = {var_rns}')
70      print(f'SD of random numbers = {sd_rns}')
71      '''
72      var_mean = var_rns / len(rns)
73      sd_mean = np.sqrt(var_mean)
74      '''
75      print(f'\nvariance of mean = {var_mean}')
76      print(f'SD of mean = {sd_mean}')
77      '''
78      interval_1_left = mean_rns - sd_mean
79      interval_1_right = mean_rns + sd_mean
80      interval_2_left = mean_rns - 2 * sd_mean
81      interval_2_right = mean_rns + 2 * sd_mean
82      interval_3_left = mean_rns - 3 * sd_mean
83      interval_3_right = mean_rns + 3 * sd_mean
84      '''
85      print(f'\nConfidence interval 1 = ({interval_1_left}, {interval_1_right})')
86      print(f'Confidence interval 2 = ({interval_2_left}, {interval_2_right})')
87      print(f'Confidence interval 3 = ({interval_3_left}, {interval_3_right})')
88      '''
89      return interval_1_left, interval_1_right, interval_2_left, interval_2_right,
    interval_3_left, interval_3_right
90
91
92  # PART 2
93  def mean_var_sd_for_means_1(seed, m): # m = number of means we want
94      np.random.seed(seed)
95
96      means = np.zeros(m)
97
98      start = time.process_time()
99
100     for i in range(0, m):
101         means[i] = triangle_approach()[1]
102
103     end = time.process_time()
104     print(f'time taken to generate {m} means = {end - start} seconds')
105
106     mean_means = np.average(means)
107     var_means = np.var(means)
108     sd_means = np.std(means)
109
110     print(f'mean of means = {mean_means}')
111     print(f'variance of means = {var_means}')
112     print(f'SD of means = {sd_means}')
113
114     dev_means = abs(means - mean_means)
115     acc = 0
116     rej = 0
117     for k in dev_means:
118         if abs(k) <= sd_means:
```

```
            acc += 1
        else:
            rej += 1

    print(f'Ratio of random numbers within 1 SD = {acc / (acc + rej)}')


def check(actual, n):
    one = 0
    two = 0
    three = 0

    for i in range(987654321, 987654321 + n):
        interval_1_left, interval_1_right, interval_2_left, interval_2_right,
interval_3_left, interval_3_right = run(i)

        if actual >= interval_1_left and actual <= interval_1_right:
            one += 1

        if actual >= interval_2_left and actual <= interval_2_right:
            two += 1

        if actual >= interval_3_left and actual <= interval_3_right:
            three += 1

        #print(i - 987654320)

    print(f'Chance to fall in interval 1 = {one / n}')
    print(f'Chance to fall in interval 2 = {two / n}')
    print(f'Chance to fall in interval 3 = {three / n}')


#run(987654321)
check(2, 10000)
#mean_var_sd_for_means_1(987654321, 10000)
triangle_approach()

```