Computer simulation # Hw3

r09942082 楊雅婷

4.

Part 1 After trunction, mean should be the same, and variance $< \sigma^2$

- Notation (mean μ, variance σ^2), $x \in [a,b]$

  $\xi = \frac{x-\mu}{\sigma}$, $\alpha = \frac{a-\mu}{\sigma}$, $\beta = \frac{b-\mu}{\sigma}$, $Z = \Phi(\beta) - \Phi(\alpha)$

- PDF $f(x;\mu,\sigma,a,b) = \frac{\phi(\xi)}{\sigma Z}$

- CDF $F(x;\mu,\sigma,a,b) = \frac{\Phi(\xi) - \Phi(\alpha)}{Z}$

- mean: $\mu + \frac{\phi(\alpha) - \phi(\beta)}{Z}\sigma$

- Variance: $\sigma^2\left[1 + \frac{\alpha\phi(\alpha) - \beta\phi(\beta)}{Z} - \left(\frac{\phi(\alpha) - \phi(\beta)}{Z}\right)^2\right]$ #

$$\phi(\xi) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\xi^2\right)$$

$$\Phi(x) = \frac{1}{2}\left(1 + \operatorname{erf}(x/\sqrt{2})\right)$$

$$\operatorname{erf} z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt$$

Part 2

Desired mean = 10

Desired variance = 16 ($\sigma^2 = 16$, $\sigma = 4$)

$[a,b] = [0, 20]$

$$\Rightarrow f(x) = \frac{1}{\sigma'\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-10}{\sigma'}\right)^2\right]$$

$$F(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-10}{\sigma'\sqrt{2}}\right)\right]$$

* mean should be the same

* var: $(\sigma')^2\left[1 + \frac{\alpha\phi(\alpha) - \beta\phi(\beta)}{Z} - \left(\frac{\phi(\alpha) - \phi(\beta)}{Z}\right)^2\right] = 16$

where $\alpha = \frac{a-\mu}{\sigma'}$, $\beta = \frac{b-\mu}{\sigma'}$, $Z = \Phi(\beta) - \Phi(\alpha)$ 找到 σ' 之後帶回

[用二分逼近找]

```
// calculate actual variance that would make the truncated normal variance desired
// approximation using bisection method
float desired2actual_variance(float var){
    float alpha, beta, phi_alpha, phi_beta, Z, temp, actual_var;
    float low = 1.0;
    float high = desired_var * 5.0;
    while ( high-low > 1e-5){
        actual_var = (high + low) / 2.0;
        float actual_std = sqrt(actual_var);
        alpha = (a - desired_mean) / actual_std;
        beta = (b - desired_mean) / actual_std;
        phi_alpha = normal_pdf(alpha);
        phi_beta = normal_pdf(beta);
        Z = normal_cdf(beta) - normal_cdf(alpha);
        // formula from wiki
        temp = actual_var * (1.0 + (alpha * phi_alpha - beta * phi_beta) / (Z) - ((phi_alpha - phi_beta)
        if (abs(desired_var - temp) < 1e-5){
            return actual_var;
        }else if (desired_var > temp){
            low = actual_var;
        }else if (desired_var < temp){
            high = actual_var;
        }
    }
    return actual_var;
}
```

Part 3.

Pseudo code

For generating truncated normal: (ref: https://link.springer.com/content/pdf/10.1007/BF00143942.pdf)

The accept–reject algorithm based on $\mathcal{U}_{[\mu^-,\mu^+]}$ is:

1. Generate $z \sim \mathcal{U}_{[\mu^-,\mu^+]}$;
2. Compute

$$\rho(z) = \begin{cases} \exp(-z^2/2) & \text{if } 0 \in [\mu^-, \mu^+] \\ \exp(\{(\mu^+)^2 - z^2\}/2) & \text{if } \mu^+ < 0 \\ \exp(\{(\mu^-)^2 - z^2\}/2) & \text{if } 0 < \mu^- \end{cases}$$

3. Generate $u \sim \mathcal{U}_{[0,1]}$ and take $x = z$ if $u \leq \rho(z)$; otherwise, go back to Step 1.

Slightly modify from above, I use:

```
// reference from: https://link.springer.com/content/pdf/10.1007/BF00143942.pdf
for (int i = 0; i < sample_size; i ++){
    float x, z, phi_z, u;
    x = uniform(a, b, STREAM_Z);
    z = (x - desired_mean) / actual_std;
    phi_z = phi(z);
    u = lcgrand(STREAM_U);
    while(u > phi_z){
        x = uniform(a, b, STREAM_Z);
        z = (x - desired_mean) / actual_std;
        phi_z = phi(z);
        u = lcgrand(STREAM_U);
    }
    result[i] = x;
    sum += x;
}
```

All C code

```
#include "simlib.h"             /* Required for use of simlib.c. */


#define PI 3.14159265358979323846
#define STREAM_Z 3
#define STREAM_U 4

float normal_pdf(float epsilon);
float normal_cdf(float x);
float phi(float z);
float sample_mean(float z[]);
float sample_variance(float z[]);
float desired2actual_variance(float var);

float desired_mean = 10.0;
float desired_var = 16.0;
float desired_std = 4.0;
float a = 0.0;
float b = 20.0;
int sample_size = 10000;
float result[30000];
```

```
23  int main(){
24      float sum = 0;
25      float v_sum = 0;
26      float sample_mean, sample_variance;
27      float actual_var = desired2actual_variance(desired_var);
28      float actual_std = sqrt(actual_var);
29
30      // reference from: https://link.springer.com/content/pdf/10.1007/BF00143942.pdf
31      for (int i = 0; i < sample_size; i ++){
32          float x, z, phi_z, u;
33          x = uniform(a, b, STREAM_Z);
34          z = (x - desired_mean) / actual_std;
35          phi_z = phi(z);
36          u = lcgrand(STREAM_U);
37          while(u > phi_z){
38              x = uniform(a, b, STREAM_Z);
39              z = (x - desired_mean) / actual_std;
40              phi_z = phi(z);
41              u = lcgrand(STREAM_U);
42          }
43          result[i] = x;
44          sum += x;
45      }
46      sample_mean = sum/sample_size;
47      for (int i = 0; i < sample_size; i ++){
48          v_sum += (result[i] - sample_mean) * (result[i] - sample_mean);
49      }
```

```
50      sample_variance = v_sum/(sample_size-1);
51      printf("actual variance = %f\n", actual_var);
52      printf("actual std = %f\n", actual_std);
53      printf("sample mean = %f\n", sample_mean);
54      printf("sample variance = %f\n", sample_variance);
55      printf("sample std = %f\n", sqrt(sample_variance));
56
57      FILE  *outfile;
58      outfile = fopen("truncated_normal.txt", "w");
59      for (int i = 0; i < sample_size; i ++){
60          fprintf(outfile, "%f\n", result[i]);
61      }
62
63      return 0;
64  }
```

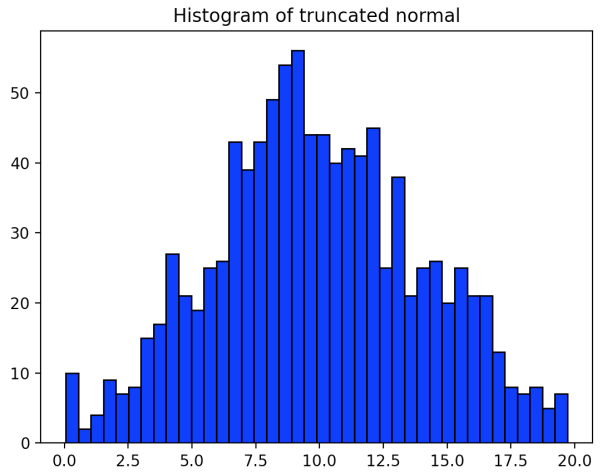
```
66  float phi(float z){
67      return exp(-z*z/2);
68  }
69
70  float normal_pdf(float epsilon){
71      return 1/(sqrt(2 * PI)) * exp(-0.5 * epsilon * epsilon);
72  }
73
74  float normal_cdf(float x){
75      return 0.5 * (1 + erff(x / (sqrt(2.0))));
76  }
```

```
77
78  // calculate actual variance that would make the truncated normal variance desired
79  // approximation using bisection method
80  float desired2actual_variance(float var){
81      float alpha, beta, phi_alpha, phi_beta, Z, temp, actual_var;
82      float low = 1.0;
83      float high = desired_var * 5.0;
84      while ( high-low > 1e-5){
85          actual_var = (high + low) / 2.0;
86          float actual_std = sqrt(actual_var);
87          alpha = (a - desired_mean) / actual_std;
88          beta = (b - desired_mean) / actual_std;
89          phi_alpha = normal_pdf(alpha);
90          phi_beta = normal_pdf(beta);
91          Z = normal_cdf(beta) - normal_cdf(alpha);
92          // formula from wiki
93          temp = actual_var * (1.0 + (alpha * phi_alpha - beta * phi_beta) / (Z) - ((phi_alpha - phi_beta)
94          if (abs(desired_var - temp) < 1e-5){
95              return actual_var;
96          }else if (desired_var > temp){
97              low = actual_var;
98          }else if (desired_var < temp){
99              high = actual_var;
100         }
101     }
102     return actual_var;
103 }
```

Part 4.

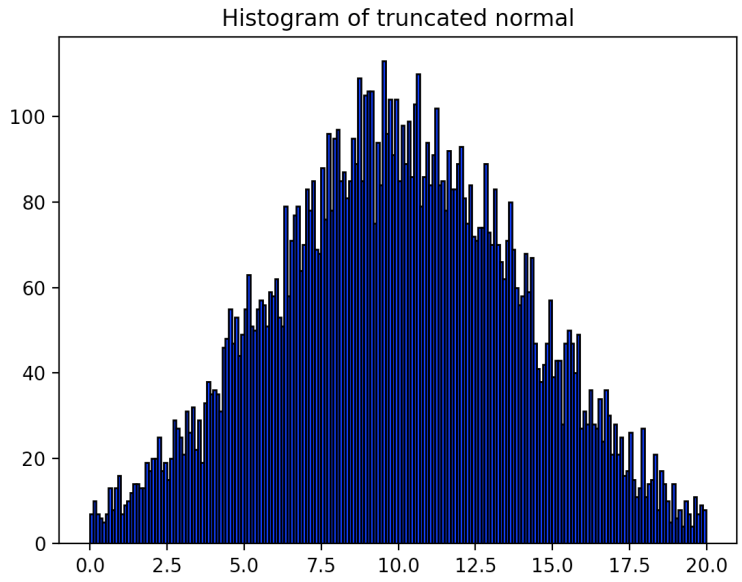
For sample size = 1000:

```
actual variance for normal = 18.281250
actual std for normal = 4.275658
sample mean for 1000 samples = 9.906714
sample variance for 1000 samples = 16.501314
sample std for 1000 samples = 4.062181
```



For sample size = 10000:

```
actual variance for normal = 18.281250
actual std for normal = 4.275658
sample mean for 10000 samples = 9.927570
sample variance for 10000 samples = 15.927991
sample std for 10000 samples = 3.990989
```



For sample size = 20000

```
actual variance for normal = 18.281250
actual std for normal = 4.275658
sample mean for 20000 samples = 9.942147
sample variance for 20000 samples = 15.939733
sample std for 20000 samples = 3.992459
```