Four Arithmetic Operations

```
1: #include <math.h>
2: #include <stdio.h>
3:
4: void main(){
5:
6:    int i;
7:
8: /* 下記 4 つは同じ意味 */
9:
10:    i=1;
11:    i=i+1;
12:    printf("i=1; i=i+1;  --->  %5d\n",i);
13:
14:    i=1;
15:    i+=1;
16:    printf("i=1; i+=1;   --->  %5d\n",i);
17:
18:    i=1;
```

```
19:     i++;
20:     printf("i=1; i++;     --->  %5d\n",i);
21:
22:     i=1;
23:     ++i;
24:     printf("i=1; ++i;     --->  %5d\n",i);
25:
26: /* 下記 2 つは同じ意味 */
27:
28:     i=1;
29:     i=i+10;
30:     printf("i=1; i=i+10; --->  %5d\n",i);
31:
32:     i=1;
33:     i+=10;
34:     printf("i=1; i+=10;  --->  %5d\n",i);
35: }
```

Output:

```
---------From Here------------
i=1; i=i+1;  --->      2
i=1; i+=1;   --->      2
i=1; i++;    --->      2
i=1; ++i;    --->      2
i=1; i=i+10; --->     11
i=1; i+=10;  --->     11
-----------------------------
```

`x=x+y;` is equivalent to `x+=y;`.

`x=x-y;` is equivalent to `x-=y;`.

`x=x/y;` is equivalent to `x/=y;`.

`x=x*y;` is equivalent to `x*=y;`.

Random number generation is utilized to perform Monte-Carlo integration.

All the random draws are based on uniform random draws between zero and one.

Random Draws from U(0,1)

```
1:  #include <math.h>
2:  #include <stdio.h>
3:
4:     int ix=1,iy=1;
5:
6:  void main()
7:  {
8:     int    i,n;
9:     double urnd(void);
10:
11:    for(i=1;i<=10000;i++) urnd();
12:
13:    scanf("%d",&n);
14:
```

```
15:    for(i=1;i<=n;i++) printf("%5d %7.5lf\n",i,urnd());
16:
17: }
18: /* ============================================== */
19: double urnd(void)
20: {
21:    int    kx,ky;
22:    double rn;
23: /*
24:    Input:
25:      ix, iy:  Seeds
26:    Output:
27:      rn: Uniform Random Draw U(0,1)
28: */
29:    kx=ix/53668;
30:    ix=40014*(ix-kx*53668)-kx*12211;
31:
32:    ky=iy/52774;
33:    iy=40692*(iy-ky*52774)-ky*3791;
34:
```

```
35:     rn=(float)(ix-iy)/2147483563.;
36:     rn-=(int)rn;
37:     if( rn<0.) rn++;
38:
39:     return rn;
40: }
```

In Line 11, the first `10000` uniform random draws are diacarded, because `urnd()` depends on the initial values `ix=1` and `iy=1`.

`rn` in Line 39 is returned to the function value `urnd()` in Line 19, i.e., `urnd() = rn`.

Monte Carlo Integration of N(0,1)

```
1: #include <math.h>
2: #include <stdio.h>
3:
4:    int ix=1,iy=1;
5:
6: void main(){
7:
8:    int i,n;
9:    double area=0.0;
10:   double w,f,g,x,rl,ru;
11:   double pi=3.14159265358979323846264338327950288419 7;
12:   double urnd(void);
13:
14:   for(i=1;i<=10000;i++) urnd();
15:
16:   scanf("%lf%lf%d",&rl,&ru,&n);
17:
18:   for(i=1;i<=n;++i){
```

```
19:       x=(ru-rl)*urnd()+rl;
20:       f=exp(-0.5*x*x)/sqrt(2.0*pi);
21:       g=1.0/(ru-rl);
22:       w=f/g;
23:       area+=w/((double)n);
24:     }
25:     printf("%10.8lf\n",area);
26: }
27: /* ============================================== */
28: double urnd(void)
29: {
30:   int    kx,ky;
31:     double rn;
32: /*
33:     Input:
34:       ix, iy:  Seeds
35:     Output:
36:       rn: Uniform Random Draw U(0,1)
37: */
38:     kx=ix/53668;
```

```
39:     ix=40014*(ix-kx*53668)-kx*12211;
40:
41:     ky=iy/52774;
42:     iy=40692*(iy-ky*52774)-ky*3791;
43:
44:     rn=(float)(ix-iy)/2147483563.;
45:     rn-=(int)rn;
46:     if( rn<0.) rn++;
47:
48:     return rn;
49: }
```

`x` in Line 19 is a uniform random draw between `rl` and `ru`.

Lines 20 – 22 correspond to $f(x)$, $g(x)$ and $w(x)$, respectively.

That is,

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, \qquad g(x) = \frac{1}{b-a}, \qquad w(x) = \frac{f(x)}{g(x)}$$

Monte Carlo Integration of Function $f(x)$:

$$\int_a^b f(x)\mathrm{d}\ x = \int_a^b \frac{f(x)}{g(x)} g(x)\mathrm{d}\ x = \int_a^b w(x)g(x)\mathrm{d}\ x \approx \frac{1}{n}\sum_{i=1}^{n} w(x_i)$$

where x_i is the ith random draw from distribution $g(x)$.

```
1: #include<stdio.h>
2: #include<math.h>
3: #include<time.h>
4:
5:    int ix=1,iy=1;
6:
7: void main()
8: {
9:    int     i,j,k,n;
10:   double  u1,u2,sn1,sn2;
11:   double  urnd(void);
12:   double  pi=3.14159265358979323846264338327950288419 7;
13:   double  x1=0.0,x2=0.0,x3=0.0,x4=0.0;
14:   clock_t t0,t1;
15:   double  dt;
16:
17:   for(i=1;i<=10000;i++) urnd();
18:
```

```
19:     scanf("%d",&n);
20:
21:     t0=clock();
22:     for(i=1;i<=n;i++){
23:       for(j=1;j<=n;j++){
24:         for(k=1;k<=n;k++){
25:           u1=urnd();
26:           u2=urnd();
27:           sn1=sqrt(-2.0*log(u1))*sin(2.0*pi*u2);
28:           sn2=sqrt(-2.0*log(u1))*cos(2.0*pi*u2);
29:           x1+=sn1;
30:           x2+=sn1*sn1;
31:           x3+=sn1*sn1*sn1;
32:           x4+=sn1*sn1*sn1*sn1;
33:         }
34:       }
35:     }
36:     x1/=((double)n)*((double)n)*((double)n);
37:     x2/=((double)n)*((double)n)*((double)n);
38:     x3/=((double)n)*((double)n)*((double)n);
```

```
39:    x4/=((double)n)*((double)n)*((double)n);
40:
41:    t1=clock();
42:    dt=(t1-t0)/((double)CLOCKS_PER_SEC);
43:
44: /*
45:    printf(" E(X)  =%10.7lf\n E(X^2)=%10.7lf\n E(X^3)=%10.7lf\n E(X^4)
46: */
47:    printf(" E(X)  =%10.7lf\n",x1);
48:    printf(" E(X^2)=%10.7lf\n",x2);
49:    printf(" E(X^3)=%10.7lf\n",x3);
50:    printf(" E(X^4)=%10.7lf\n",x4);
51:    printf("%4d^3 random draw generation time = %lf seconds\n",n,dt);
52: }
53: /* ============================================== */
54: double urnd(void)
55: {
56:    int    kx,ky;
57:    double rn;
58: /*
```

```
59:     Input:
60:       ix, iy:  Seeds
61:     Output:
62:       rn: Uniform Random Draw U(0,1)
63: */
64:     kx=ix/53668;
65:     ix=40014*(ix-kx*53668)-kx*12211;
66:
67:     ky=iy/52774;
68:     iy=40692*(iy-ky*52774)-ky*3791;
69:
70:     rn=(float)(ix-iy)/2147483563.;
71:     rn-=(int)rn;
72:     if( rn<0.) rn++;
73:
74:     return rn;
75: }
```

Lines 3, 14, 21, 41 and 42 are important to obtain computational time.

Matrix ⟶ Example of Inverse Matrix:

Inverse of Matrix

```
#include<stdio.h>
#include<math.h>

double x[100][100];

void main()
{
    int i,j,k;
    void inverse(int k);

    k=2;
    x[1][1]=1.;
    x[1][2]=1.;
    x[2][1]=x[1][2];
    x[2][2]=5.;

    printf("Input:\n");
```

```
18:     for(i=1;i<=k;++i){
19:       for(j=1;j<=k;++j){
20:       printf("%15.7lf",x[i][j]);
21:       }
22:     printf("\n");
23:     }
24:
25:     inverse(k);
26:
27:     printf("Output:\n");
28:     for(i=1;i<=k;++i){
29:       for(j=1;j<=k;++j){
30:       printf("%15.7lf",x[i][j]);
31:       }
32:     printf("\n");
33:     }
34:
35: }
36: /* ------------------------------------------------------------ */
37: void inverse(int k)
```

```
38: {
39:    double a1,a2;
40:    int    i,j,m;
41:
42:    for(i=1;i<=k;i++){
43:       a1=x[i][i];
44:       x[i][i]=1.0;
45:       for(j=1;j<=k;j++) x[i][j]=x[i][j]/a1;
46:       for(m=1;m<=k;m++){
47:          if(m != i){
48:             a2=x[m][i];
49:             x[m][i]=0.0;
50:             for(j=1;j<=k;j++) x[m][j]-=a2*x[i][j];
51:          }
52:       }
53:    }
54: }
```