1 台形公式により 4 分円の面積を求めるプログラム

```
0001: /*
0002:   trape.c
0003:   数値積分プログラム
0004:   台形公式を使って数値積分を行います。
0005:   使い方  ./trape (区間分割数 N)
0006:   区間は 0〜10 に固定してあります。
0007: */
0008:
0009: #include <stdio.h>
0010: #include <stdlib.h>
0011: #include <math.h>
0012:
0013: double fx(double x);
0014:
0015: int main(int argc, char *argv[])
0016: {
0017:   double h;
0018:   double integral;
0019:   int    n;
0020:   int    i;
0021:
0022:   if(argc < 2){
0023:     fprintf(stderr, "使い方  ./trape (区間分割数N) \n");
0024:     exit(1);
0025:   }
0026:
0027:   if((n = atoi(argv[1])) <= 0){
0028:     fprintf(stderr, "区間分割数Nが不正です(%d)\n", n);
0029:     exit(1);
0030:   }
0031:
0032:   h = 1.0 / n;
0033:
0034:   integral = fx(0.0) / 2.0;
0035:   for(i = 1; i < n; ++i){
0036:     integral += fx((double)i / n);
0037:   }
0038:   integral += fx(1.0) / 2.0;
0039:   integral *= h;
0040:
0041:   printf("積分値I %.9lf    4I   %.9lf\n", integral, integral*4.0);
0042:
0043:   return 0;
0044: }
0045:
0046: double fx(double x)
```

```
0047: {
0048:   return sqrt(1.0 - x * x);
0049: }
```

2 「trape」の操作方法

「trape」の操作方法

./trape [区間分割数 N]「Enter」

上記の「./」は、Linux でのプログラムを実行する時の例です。
ウィンドウズでは、trape.exe [区間分割数 N] 「Enter」と入力してください。

3 実行結果

```
./trape 1000
積分値 I 0.785388867    4I   3.141555467

./trape 10000
積分値 I 0.785397869    4I   3.141591478

./trape 100000
積分値 I 0.785398154    4I   3.141592616

./trape 1000000
積分値 I 0.785398163    4I   3.141592652
```

4 擬似乱数により 4 分円の面積を求めるプログラム

```
0001: /*
0002:   ri.c
0003:   擬似乱数を使って数値積分を行います。
0004:   使い方  ./ri（試行回数 n）
0005:   区間は 0〜1 に固定してあります
0006: */
0007:
0008: #include <stdio.h>
0009: #include <stdlib.h>
0010:
```

```
0011: #define SEED RAND_MAX-1
0012: #define R 10
0013:
0014: double frand(void);
0015:
0016: int main(int argc, char *argv[])
0017: {
0018:   int    integral;
0019:   int    n;
0020:   int    i,r;
0021:   double x,y;
0022:
0023:   if(argc < 2){
0024:     fprintf(stderr, "使い方  ri  (試行回数 n) \n");
0025:     exit(1);
0026:   }
0027:   if((n = atoi(argv[1])) <= 0){
0028:     fprintf(stderr, "試行回数 n が不正です (%d)\n", n);
0029:     exit(1);
0030:   }
0031:
0032:   srand(SEED);
0033:
0034:   for(r = 0; r < R; ++r){
0035:     integral = 0;
0036:     for(i = 0; i < n; ++i){
0037:       x = frand();
0038:       y = frand();
0039:       if((x * x + y * y) <= 1){
0040:         ++integral;
0041:       }
0042:
0043:     }
0044:     printf("積分値 I  %.9lf    4I  %.9lf\n",
0045:       (double)integral / n, (double)integral / n * 4.0);
0046:   }
0047:
0048:   return 0;
0049: }
0050:
0051: double frand(void)
0052: {
0053:   return (double)rand() / RAND_MAX;
0054: }
```

5 「ri」の操作方法

「ri」の操作方法

./ri [試行回数 n]「Enter」

上記の「./」は、Linux でのプログラムを実行する時の例です。
ウィンドウズでは、ri.exe [試行回数 n] 「Enter」と入力してください。

6 実行結果

```
./ri 1000000
積分値 I  0.785204000    4I  3.140816000
積分値 I  0.784983000    4I  3.139932000
積分値 I  0.784631000    4I  3.138524000
積分値 I  0.785324000    4I  3.141296000
積分値 I  0.785548000    4I  3.142192000
積分値 I  0.784921000    4I  3.139684000
積分値 I  0.785234000    4I  3.140936000
積分値 I  0.784834000    4I  3.139336000
積分値 I  0.785533000    4I  3.142132000
積分値 I  0.785354000    4I  3.141416000
積分値 I  0.785388867    4I   3.141555467
```

同じ関数の数値積分を、台形公式と乱数に基づく方法で実行しました。台形公式は単純な公式です。しかし、被積分関数 $f(x)$ が連続でなめらかな関数であれば、分割数を増やすことでそれなりの精度を持った計算が可能になります。これに対して乱数を用いた数値積分は、計算量が大きいにもかかわらず精度は良くありません。乱数よる数値積分は、被積分関数 $f(x)$ の性質が悪く、ほかの数値積分では計算が困難な場合に用いる、特殊な方法であると考えるべきでしょう。

C による数値計算とシュミレーション　131 頁　小高知宏著　オーム社より