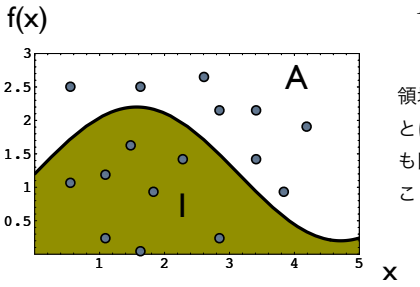
モンテカルロ積分

- 乱数を用いた f(x) の積分の計算

領域 A (0 ≤ x ≤ 5, 0 ≤ y ≤ 3) でランダムな点をとる。
積分 I は、ランダムな点が曲線 f の下側に落ちる割合と領域 A の面積の積で与えられる（と予想される）

$$I = \int_3^5 f(x)dx$$



領域 A でランダムな点をとるとは、領域 A 内のどの場所にも同じ確からしさで点を打つこと。

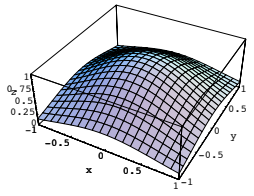
モンテカルロ積分の用途

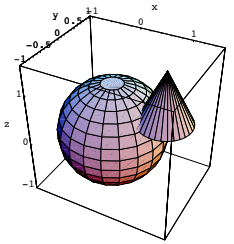
- 重積分など被積分関数 f の評価や積分範囲の数式表現が困難な場合に用いられるモンテカルロ積分
- 疑似乱数を用いて積分値の近似値を求める（高い精度は望めない）

$$V = \iint_\Omega f(x,y)dxdy$$

$$\Omega : -1.5 \le x, y \le 1.5$$

被積分関数と積分領域の解析表示は困難



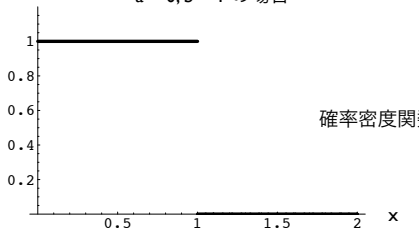


一様乱数

- 区間 [a, b] で一様な乱数 X（a から b の値を同じ確からしさで取る）

$$\mathrm{Prob}[x < X < x + dx] = \begin{cases} \frac{1}{b-a}dx & a < x < b \\ 0 & \text{Otherwise} \end{cases}$$

a = 0, b = 1 の場合



確率密度関数はステップ関数

疑似乱数の生成

- 疑似乱数をソフトウェア的に生成するアルゴリズムには幾つかある

線形合同法

$$X_{i+1} = (A \times X_i + B) \mod M$$

適当な定数 A, B, M (M > A, M > B, A > 0, B > 0) を用いて、
X_0 を乱数の種 seed として逐次乱数列 X_i を生成する方法

周期は最大で M

線形合同法

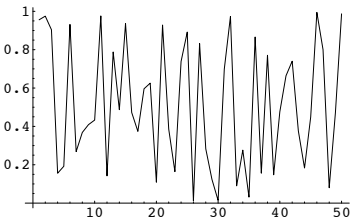
- 初期値 X0 を与える漸化式に他ならない
- A, B, M をうまく選ぶとそれなりの質の疑似乱数が生成される？

```
#define IA   3877
#define IB   29573
#define IM   139968 // 周期！

int main (int argc, const char * argv[]) {

   int i, randomInt;
   double randomDouble;

   randomInt = 10;

   for(i=0; i<1000; i++){
       randomInt = (randomInt*IA + IB) % IM;
       randomDouble = randomInt/(IM + 1.0);
   }

   return 0;
}
```

疑似乱数の質

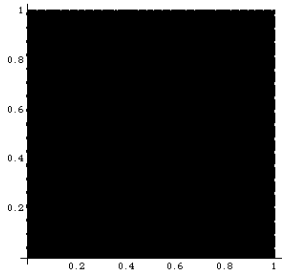
- 線形合同法で生成された乱数 [0, 1) はどの程度良質か？
- 疑似乱数列 {x0, x1, x2, ... } を、1次元、2次元、3次元上の点として描画

```
data = ReadList["data-myrand", Real];
ListPlot[Take[data, -50], PlotJoined→True]
```

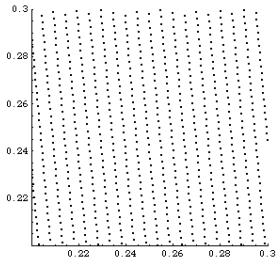


- Graphics -

```
data = ReadList["data-myrand", {Real, Real}];
ListPlot[data, AspectRatio -> 1]
```

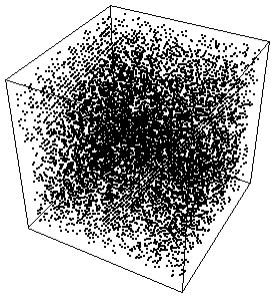



```
data = ReadList["data-myrand", {Real, Real}];
ListPlot[data, AspectRatio -> 1, PlotRange->{{0.2,0.3},{0.2,0.3}}]
```



拡大すると点 (Xn, Xn+1) が
規則的に配置

点 (Xn, Xn+1, Xn+2) が規則的に配置？



```
data = ReadList["data-myrand", {Real, Real, Real}];
seqPoint = Map[Point, data];
g = Show[Graphics3D[Take[seqPoint, -10000]]]
```

rand 関数

- 多くの C 言語処理系で実装される rand 関数は線形合同法により疑似乱数を生成

void srand(unsigned)
乱数の種（初期値）を指定

int rand(void)
疑似乱数を生成

```
#include <stdlib.h>
#include <time.h>

unsigned seed;
int randomInt;
double randomDouble;

seed = (unsigned)time(NULL);

srand(seed);
printf("RAND_MAX is %d\n", RAND_MAX);
printf("Seed is %d\n", seed);

for(i=0; i<100; i++){
  randomInt = rand();
  randomDouble = randomInt/(RAND_MAX + 1.0);
  printf("%d, %f\n", randomInt, randomDouble);
}
```

メルセンヌ・ツイスタ

- より高品質な疑似乱数を生成する方法に Mersenne Twister がある

void init_genrand(long)
乱数の種（初期値）を指定

double genrand_real2(void)
疑似乱数 [0, 1) を生成

両関数とも外部ファイルで
定義されている

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

extern void init_genrand(long);
extern double genrand_real2(void);

long seed;
int i;
double rand;

seed = (long)time(NULL);

init_genrand(seed);

for(j=0; j<100; j++){
  rand = genrand_real2();
  printf("%.20f\n", rand);
}
```

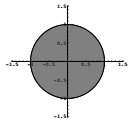
問題 1

- メルセンヌ・ツイスタを用いて、区間 [0, 1) の一様疑似乱数を N 個生成せよ。
- [0, 1) を区間幅 0.1 で 10 個の区間に区切り、各区間に落ちた疑似乱数の数を数えてファイルに書き出せ。
- 乱数が一様であれば、各区間に落ちる数の平均は N/10 と予想される。生成した疑似乱数が統計的に一様であるか、どのようにして判定したら良いか考えよ。
- 乱数列 {x1, x2, ..., } を2次元もしくは3次元上に描いて、視覚的に乱数の質を調べよ。

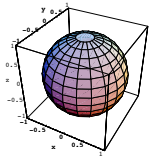
問題 2

- モンテカルロ法により以下の積分の近似値を求め、真の解と比較せよ

$$2\int_{-1}^{1}\sqrt{1-x^2}dx = \iint_{x^2+y^2\leq 1} dxdy$$

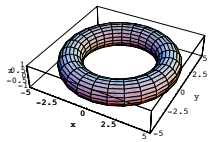


$$\iint_{x^2+y^2\leq 1}\sqrt{1-x^2-y^2}dxdy = \iiint_{x^2+y^2+z^2\leq 1} dxdydz$$



半径 1 の円を z 軸の周りで半径 4 で回転させて出来る
トーラスの体積

$$\iiint_{(\sqrt{x^2+y^2}-4)^2+z^2\leq 1} dxdydz$$



問題 3

- モンテカルロ法により以下の積分の近似値を求めよ

原点を中心とする半径 2 の球と、
(1, 0, 1) を中心とする半径 1 の球の
合成部分の体積