

数值計算法

2010/6/30

林田 清

大阪大学大学院理学研究科

擬似乱数の発生

- モンテカルロ法のための準備
- 一様乱数($0 \leq x < 1$ の間、一様な確率)
- 計算機で与えられるのはあくまでも擬似乱数
- Cでの関数 `srandom`と`random`
 - 注) `srandom`と`random`は、`srand`と`rand`の改良版。ただし、最近のLinux OSでは、`srand`と`rand`も`srandom`と`random`と同じアルゴリズムを使用している。
 - `#include <stdlib.h>`
 - `srandom(seed)`で初期化。`seed`が同じ値だと同じ擬似乱数列になる。
 - `x=(double)(random()/(RAND_MAX+1.0))`が一様擬似乱数を与える。呼ぶたびに乱数列の次の値が出てくる。

乱数ルーチンの選び方

- `srandom`, `random`あるいは`srand`, `rand`は学習用
- 本格的な研究にはより”性能のいい”擬似乱数発生ルーチンを使うべき
 - 偏りがない、周期性がない、計算速度が速いなど
- 例えば、Numerical Recipes in C (W.Press他著、技術評論者) や計算物理(早野龍五、高橋忠幸著、共立出版)を参考のこと
- 新しい試みの一例 Mersenne Twister法(<http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/mt.html>)

乱数発生回路、ボード

■ 一例：東芝製ランダムマスター

□ http://www3.toshiba.co.jp/power/techno/secret/random/index_j.htm

□ http://www.t-rs.co.jp/products/p-5/index_j.htm

□ 半導体の熱雑音

■ 最近の記事

□ http://www.toshiba.co.jp/rdc/rd/detail_j/0603_01.htm

■ 亂数は、情報セキュリティ、暗号化にはなくてはならない技術。

一様乱数の応用例(簡単なモンテカルロ計算)

- 一様擬似乱数を与える
- 円の面積(円周率の計算)
 - (擬似)一様乱数の組(x,y)をN個生成
 - 半径1の円の中($x^2+y^2<1$)に含まれる点の個数m
 - $(m/N) \times 4$ が π の近似値になるはず
- 面積の計算、従って積分にも応用できる(モンテカルロ積分)。

プログラム例

- ここではseedは任意の数(できれば素数)を入力する。
 - `random(time(NULL));`として時間情報を種とするともできる。この場合は `time.h`をincludeする必要あり。
- 右の例でpi1は整数の割り算が最初に行われる所以値がゼロになる。pi2, pi3のような記法にせよ。
- `RAND_MAX`は `/usr/include/stdlib.h`の中で定義(`#define`)されている。

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

int main()
{
    int seed, n, i, m;
    double x,y,pi1,pi2,pi3;

    printf("input seed number\n");
    scanf("%d",&seed);

    printf("input number of points \n");
    scanf("%d",&n);

    random(seed);

    m=0;
    for(i=0;i<n;i++) {
        x=((double)random())/(RAND_MAX+1.0);
        y=((double)random())/(RAND_MAX+1.0);
        if((x*x+y*y)<1.0) m++;
        printf("%d x=%lf y=%lf\n",i,x,y);
    }
    pi1 = m/n * 4.0;
    pi2 = (m*4.0)/n;
    pi3 = ((double) m)/((double) n )*4.0;
    printf("pi=%lf %lf %lf\n",pi1,pi2,pi3);
}
```

プリプロセッサ

■ `#include <fname.h>`

- コンパイルされる際に、ソースコードのこの位置に `fname.h`(通常は`/usr/include`の下にある; 具体的には `stdio.h,stdlib.h,math.h`等) の内容が挿入される。

■ `#define ABC abc`

- ソースコードのこの位置以降にあるABCという文字列があれば、コンパイル時に自動的にabcにおきかえられて解釈される。
- `RAND_MAX`は `stdlib.h`の中で `define`されている。

与えられた分布関数に従う乱数1:逆変換法

分布関数 $f(x)$ に従う乱数 x を発生させたい

累積分布関数 $F(x)$

$$F(x) = \int_{x_{\min}}^x f(x') dx'$$

は区間(0, 1)に一様にランダムに分布する。

もし F の逆関数 F^{-1} を見つけることができれば $x = F^{-1}(r)$ により

擬似一様乱数 r から分布関数 $f(x)$ に従う乱数 x を求めることができる

例 : $f(x) = \lambda \exp(-\lambda x)$

$$F(x) = \int_0^x f(x') dx' = [-\exp(-\lambda x')]_0^x = 1 - \exp(-\lambda x)$$

$$x = -\frac{1}{\lambda} \ln(1 - F(x)) = -\frac{1}{\lambda} \ln(r)$$

逆変換法の考え方

r が $0 < r < 1$ で定義される一様乱数のとき

確率分布は $p_r(r)dr = dr \quad (0 < r < 1)$

この r がある関数 $x(r)$ で x に変換されるととき

x の確率分布 $p_x(x)$ は $p_x(x)dx = p_r(r)dr$ から

算出できる。

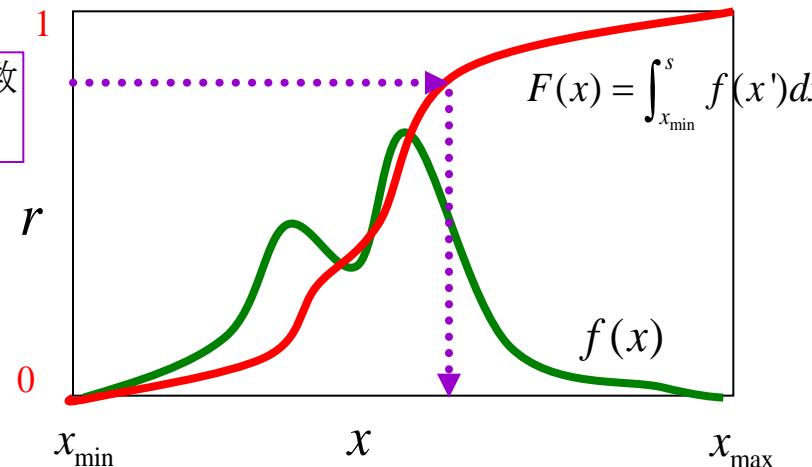
$$p_x(x) = \frac{dr}{dx} \quad (x(r=0) < x < x(r=1))$$

逆変換法では $r = F(x)$ 従って $x = F^{-1}(r)$ と
とる。すると

$$p_x(x) = \frac{dF(x)}{dx} = f(x) \quad (x_{\min} < x < x_{\max})$$

x の確率分布は $f(x)$ になる。

一様乱数
(入力)



$x = F^{-1}(r)$ と変換された乱数
(出力) を得る

$F^{-1}(r)$ が容易に計算できる
ことが必要

正規乱数の発生

- 積分は誤差関数、解析的に記述できない。 $x = \sum_{i=1}^{12} r_i - 6$
□ Box-Muller法か中心極限定理 → ここで r_i は $(0,1)$ の一様乱数

一般に確率密度関数 $q(x_1, x_2)$ に従う変数 x_1, x_2 の関数 y_1, y_2 の確率密度関数は

$$p(y_1, y_2) dy_1 dy_2 = q(x_1, x_2) \left| \frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} \right| dy_1 dy_2$$

$$p(y_1, y_2) dy_1 dy_2 = \left[\frac{1}{\sqrt{2\pi}} \exp(-y_1^2/2) \right] \left[\frac{1}{\sqrt{2\pi}} \exp(-y_2^2/2) \right] dy_1 dy_2$$

となるような変数 y_1, y_2 を生成したい ⇒ Box-Muller 法

$$y_1 = \sqrt{-2 \ln x_1} \cos 2\pi x_2, y_2 = \sqrt{-2 \ln x_1} \sin 2\pi x_2$$

x_1, x_2 が区間 $(0, 1)$ の独立な一様乱数であれば $q(x_1, x_2) dx_1 dx_2 = dx_1 dx_2$

$$x_1 = \exp \left[-\frac{1}{2} (y_1^2 + y_2^2) \right], x_2 = \frac{1}{2\pi} \arctan \frac{y_2}{y_1}$$

$$\frac{\partial(x_1, x_2)}{\partial(y_1, y_2)} = - \left[\frac{1}{\sqrt{2\pi}} \exp(-y_1^2/2) \right] \left[\frac{1}{\sqrt{2\pi}} \exp(-y_2^2/2) \right]$$

与えられた分布関数に従う乱数2:棄却法

発生が容易な分布関数（例えば一様分布） $g(x)$ の定数倍 $Cg(x)$ によって $f(x)$ を完全に包み込む。

1. g の確率分布に従う乱数を発生させる
2. 点 x_0 で $f(x_0)$ と $Cg(x_0)$ を計算する。 $(0,1)$ の乱数 r を発生させ、
 - (a) $f(x_0) > rCg(x_0)$ ならば $rCg(x_0)$ を棄ててステップ1からやり直す
 - (b) $f(x_0) \geq rCg(x_0)$ ならば、 x_0 が求める乱数

