物理学情報処理演習
11. モンテカルロ法

2016年6月28日
ver20160628_2

本日の推奨作業directory

lesson11

11.1 モンテカルロ法(棄却法)

11.2 モンテカルロ法(逆変換法)

11.3 モンテカルロ積分

参考文献

・ やさしいC++ 第4版 高橋 麻奈 (著)
 ソフトバンククリエイティブ

・プログラミング言語C++第4版
 ビャーネ・ストラウストラップ, Bjarne Stroustrup, 柴田 望洋

・ Numerical Recipes: The Art of Scientific Computing, Third Edition in C++

身内賢太朗

レポート提出:fsci-phys-jouhou@edu.kobe-u.ac.jp

11.1　乱数

- 乱数はシミュレーションなどを行う際に有用である。
- defaultでは再現する乱数が発生する。
  →挙動の確認のため。
- 「真の」乱数を得るためには適切な初期値(乱数の種)を与える必要がある。通常time(NULL)を用いて、現在時刻を種とする。

- 算術ライブラリ math.h

  int rand()　　0からRAND_MAXまでの整数を発生する。

  int RAND_MAX

  int srand();

rand_1.cxx

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <math.h>
using namespace std;
#define MAX 10
int main(int argc, char *argv[] )
{
 int j;
  for(j=0;j<MAX;j++){
   cout << j << "¥t" << rand() << "¥t" << RAND_MAX<< endl;
  }
 return 0;
}
```

rand_3.cxx

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <math.h>
using namespace std;
#define MAX 10
int main(int argc, char *argv[] )
{
 int j;
 srand(time(NULL));
  for(j=0;j<MAX;j++){
   cout << j << "¥t" << rand() << "¥t" << RAND_MAX<< endl;
  }
 return 0;
}
```

演習9.1(提出不要) rand_1を何回か走らせてみて、乱数発生が再現していることを確認してみよう。rand2.cxxを変更してsrandで与える種を変更して実行してみよう。rand_1では乱数の種に何が与えられているだろうか。

rand_3を何回か実行してみよう。再現性はあるだろうか。rand_3を変更して、rand_4として1列目にj、2列目に0～1までの間の乱数を表示するプログラムとしておこう。(double)rand()としてint型をdouble型に変更する必要がある。

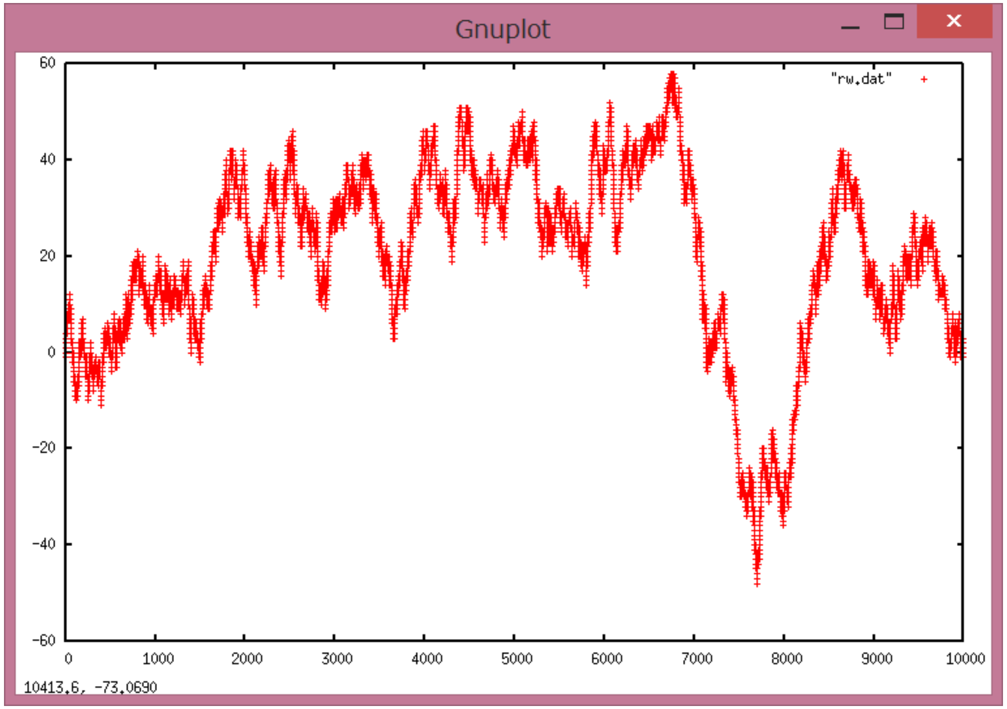
11.2 モンテカルロ法：棄却法

- 確率的に起こる物理現象を乱数を用いて計算機上で再現することができる。
  → モンテカルロ法（簡単な例：課題9 演習11.1a）
- 乱数を発生させて、不要な点を排除するということで、必要な乱数を得ることができる。（演習11.1b）

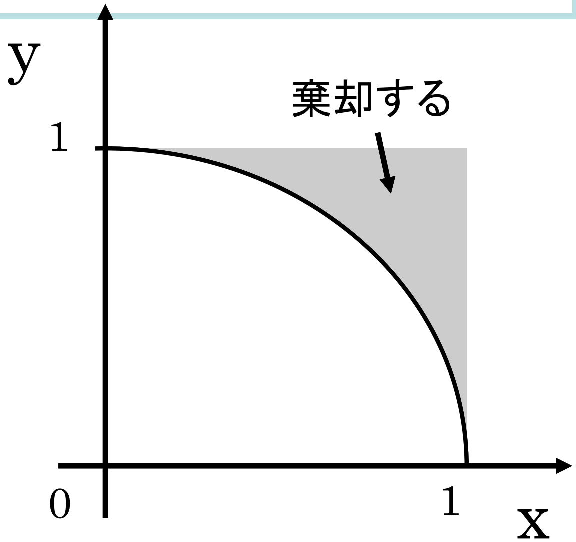
- 同様にして、変数xに対しての分布関数f(x)に従う乱数を発生させることも可能である。(演習11.1c)

ランダムウォークの実行例。

演習11.2.1 (提出不要) 0から1までの乱数を10000回発生させる。初期値でx=0として、0.5より大きい場合はxに1を加え、それ以外の場合にはxから1を引く。横軸に乱数の発生回数N、縦軸にxをとったグラフを描画してみよう。（ランダムウォークと呼ばれる。）



演習11.2.2 (提出不要) circle_1.cxx は半径1の円の第一象限内に均等に点を発生させるプログラムである。棄却が行われていることを確認してみよう。点を10000点発生させ、図示してみよう。



circle_1.cxx

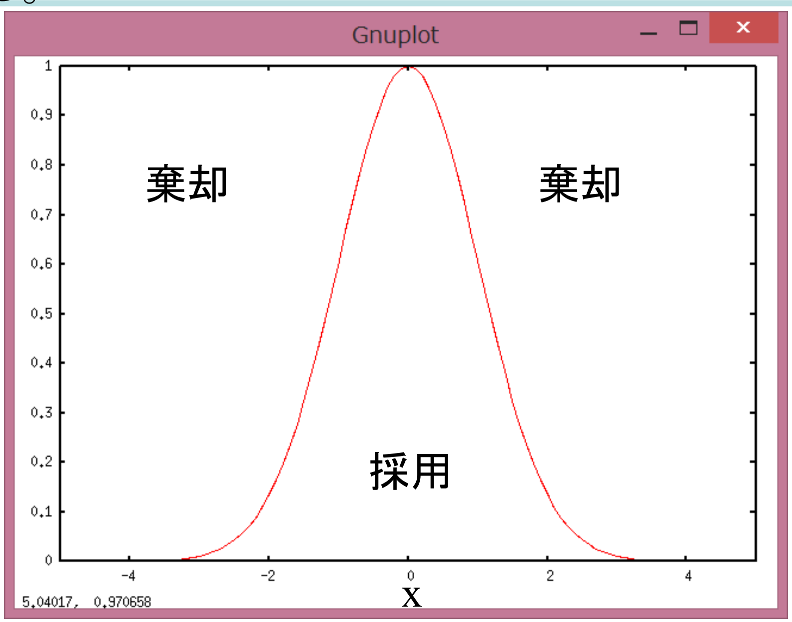
```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <math.h>
using namespace std;
#define MAX_DEFAULT 10
//random points in a circle
int main(int argc, char *argv[] )
{
  int j,max;
  double x,y,r;
  r=1;
  if(argc>1)max=(int)(atof(argv[1]));
  else max=MAX_DEFAULT;
  srand(time(NULL));
   for(j=0;j<max;j++){
    //     cout << j << "¥t" << (double)rand()/RAND_MAX<< endl;

    x=r*(double)rand()/RAND_MAX;
    y=r*(double)rand()/RAND_MAX;
    if(y<sqrt(1-x*x)){
      cout << x<< "¥t" <<y<<endl;
    }
   }
  return 0;
}
```

演習11.2.3 (提出不要)　rand_gaus_1.cxxはガウシアンに従う乱数を発生させるコードである。

内容を理解しよう。

横軸に試行回数、縦軸に出力されたxの値をplotしてみよう。

(発展)hist_1などを参考、に採用されたxをヒストグラムにしてみよう。rand_gaus_1.cxxの出力をgaus.datとしたときに
hist_1 ＜ gaus.dat ＞ gaus_hist.dat
としてヒストグラム化が可能。これを
plot ‘gaus_hist.dat’ using1:3　プロットできる。



rand_gaus_1.cxx

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <math.h>
using namespace std;

#define X_MIN -5
#define X_MAX 5
#define RAND_NUM 10000

double gaus(double x){
  // return exp(-x*x/2.)/sqrt(2*M_PI);
  return exp(-x*x/2.);//normalize  gaus(0)=1
}

int main(int argc, char *argv[] )
{
  int i,bin;
  double x,r;

  srand(time(NULL));
  i=0;
  while(RAND_NUM>i){
    x=(X_MAX-X_MIN)*((double)rand()/RAND_MAX-0.5);
    if((double)rand()/RAND_MAX<gaus(x)){
      cout<<x<<endl;
      i++;
    }
  }


  return 0;
```

11.3 モンテカルロ法:逆変換法

- 棄却法では条件によっては無駄になる乱数の発生が多数発生する。効率よく乱数を発生させる方法として、逆関数を用いる方法がある。

  - 一般にある確率密度が$p(x)$に従う変数xをある関数で$y(x)$に変換したとする。この時、yの確率密度関数を$q(y)$とすると、確率の保存より $p(x)dx=q(y)dy$となる。 ①
  - ここで、yを0から1までの一様な乱数であると考える。すなわち$0<y<1$で$q(y)dy=dy$ が成り立つ。②

  - 今、確率密度関数が$f(x)$に従う乱数xを、一様な乱数yを用いて発生させたい(関数$x(y)$を知りたい)とする。①と②より、$f(x)=dy/dx$ という微分方程式が成り立つ。これは$f(x)$の不定積分$F(x)$を用いて$F(x)=y$ とかける。$F(x)$の逆関数を用いて$x=F^{-1}(y)$ が求める関数である。

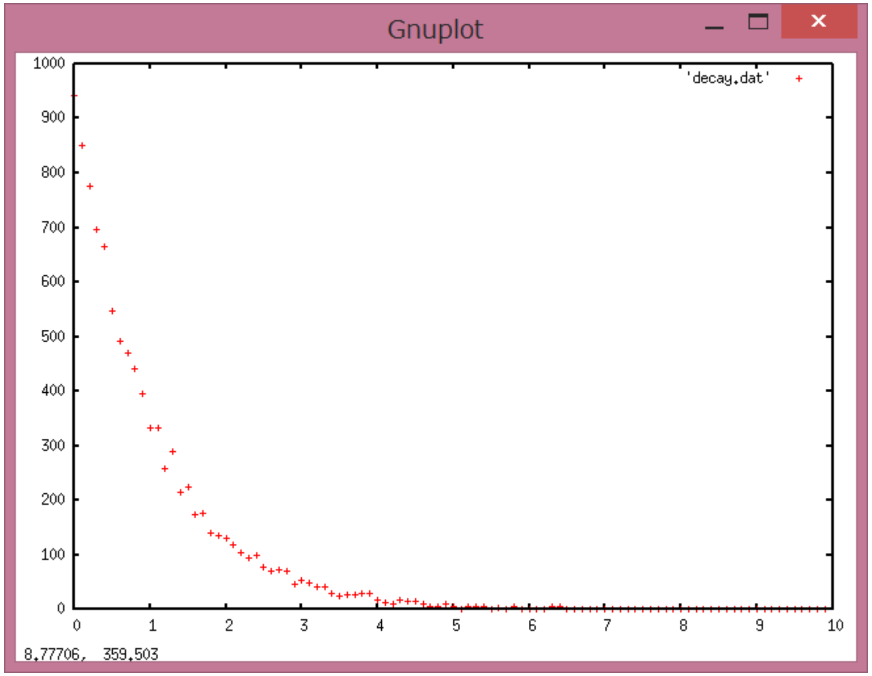
- 以上より、確率密度関数$f(x)$に従う乱数xを発生させるためには、0から1までの一様な乱数yを用いて、$f(x)$の不定積分$F(x)$の逆関数$x=F^{-1}(y)$とすればよい。

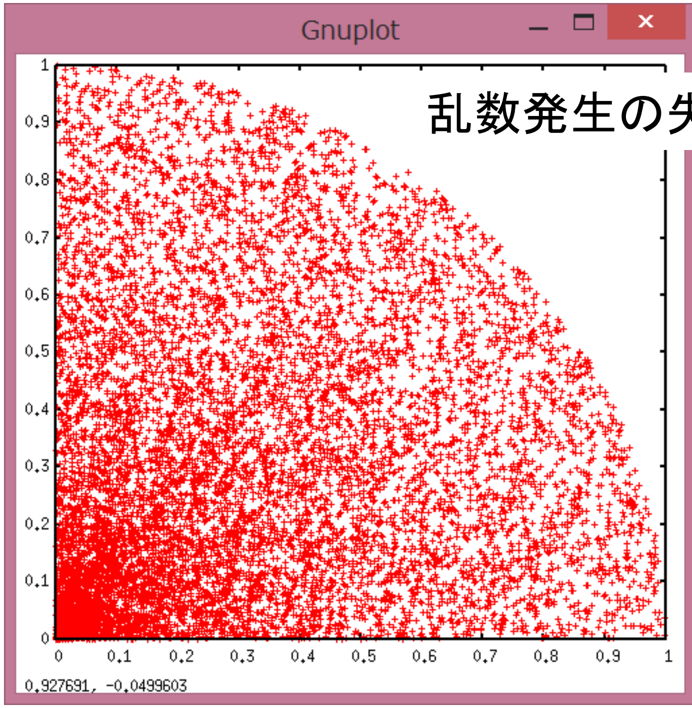
演習11.3.1 (提出不要)　原子核の崩壊は$f(x)=\lambda \exp(-\lambda x)$とかける。ここで$\lambda$は壊変定数、xは時刻である。ある時刻での崩壊数は指数関数的に減少してゆく。崩壊の観測される時刻xを乱数として発生させたい。

前のページの議論より、逆関数　$x = -\frac{1}{\lambda}\ln y$　を用いることで乱数xを発生させることができるはずである。decay_1.cxxはこの考え方で崩壊時刻xを発生させたものである。xについてヒストグラムを描くと右下の図のように指数関数的に発生していることを確認せよ。

decay_1.cxx

```
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <math.h>
using namespace std;
#define LAMBDA 1
#define RAND_NUM 10000
int main(int argc, char *argv[] )
{
 int i;
 double x,y;

 srand(time(NULL));
 i=0;
 while(RAND_NUM>i){
  x=-log((double)rand()/RAND_MAX)/LAMBDA;
  cout<<x<<endl;
  i++;
 }
 return 0;
}
```

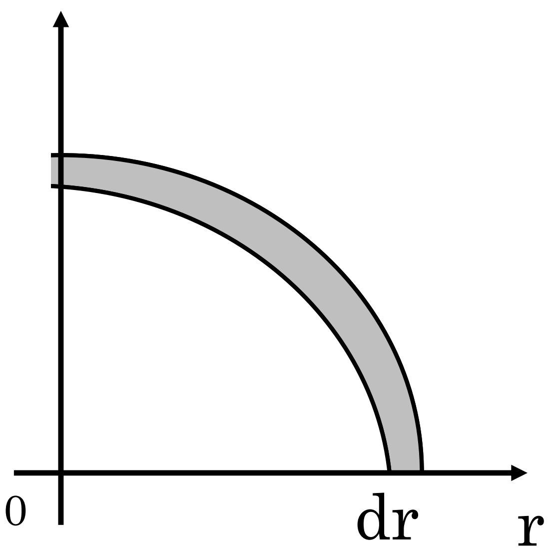




乱数発生の失敗例

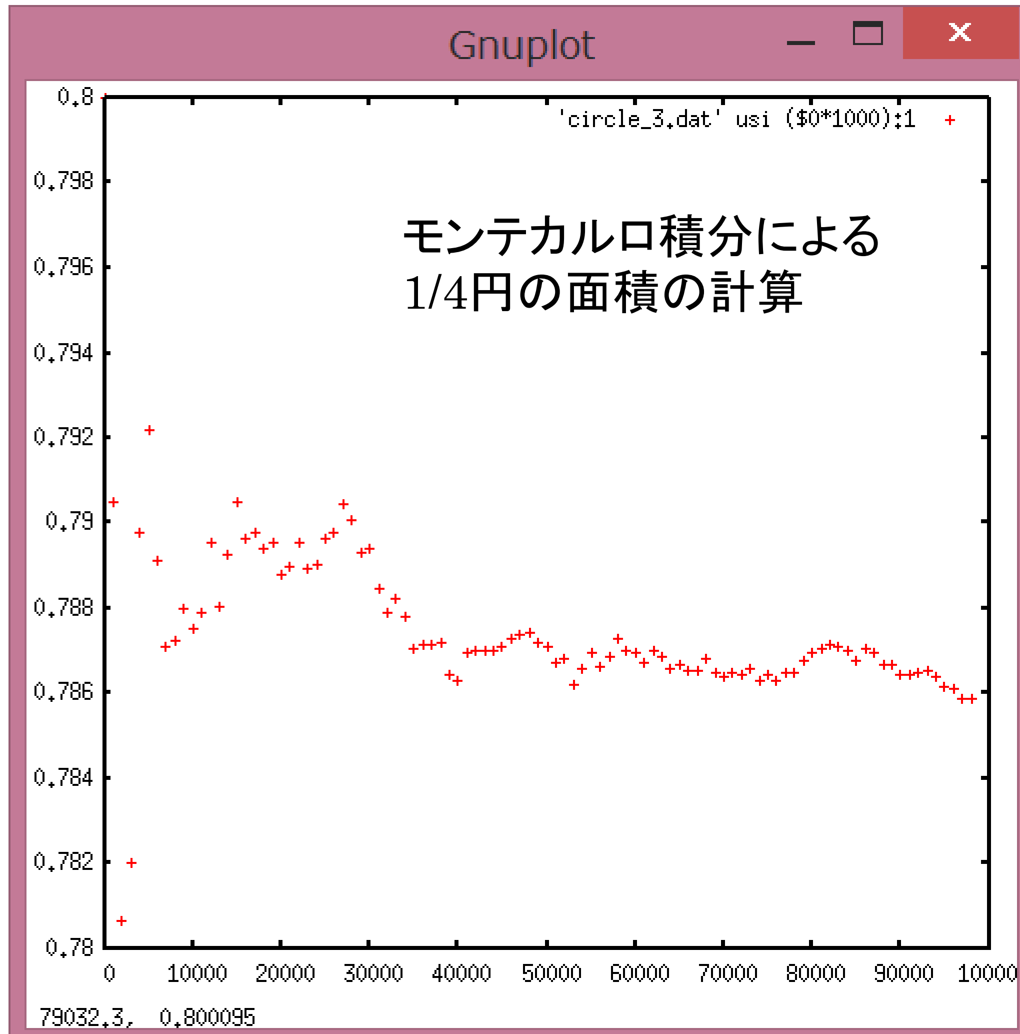
演習11.3.2 (提出不要) circle_1.cxxではx,yを乱数として発生させていた。極座標r, θ を乱数として、同様のことを行おうとした失敗例をcircle_2e.cxxに示す。図示して、失敗の理由を考え、修正プログラムを考えよう。

図の様に半径r～r+drの弧の部分の微小面積はrdrに比例する。従って乱数の発生確率をrに対して一定ではなく∝rとする必要がある。棄却法、逆変換法どちらも使用可能。最低限棄却法では自力で作ってみよう。



11.4 モンテカルロ積分

- 式で表すことはできるが、解析的に積分が難しい複雑な形状の積分を乱数を発生させることで計算可能。



演習11.4.1(提出不要) circe_3.cxxは1/4円の面積をモンテカルロ積分で求めるものである。積分の回数を重ねると面積が右図の様に収束してゆくことを確認せよ。

レポート提出：fsci-phys-jouhou@edu.kobe-u.ac.jp
件名　2016-report11_学籍番号の下4桁
提出期限　2016年7月3日13:00

ソースコードファイル名：2016_jouhou_11_学籍番号の下4桁.cxx
出力画像ファイル名：2016_jouhou_11_学籍番号の下4桁.pdf

課題11：モンテカルロ法を用いて、以下の試行実験を行なえ。
① 確率の低い現象(p)を多数回(N)繰り返し期待値(μ)となるように行い、発現回数Xを求める。
② 上記試行をμ=1について100回くり返し、横軸X、縦軸を100回中の確率としてplotする。
③ 期待値=1のポアソン分布

$$\boxed{\frac{\mu^x}{x!}e^{-\mu} \equiv P(x,\mu)}$$

を同じグラフに線(with lines)で重ね書きし、比較する。