シミュレーション
～宿題 3: Monte Calro Simulation～

e055717 金城佑典

2008/12/25

目次

1　Monte Carlo method を用いた積分

モンテカルロ法 (Monte Carlo method, MC) は乱数を用いて行なうシミュレーションの総称で、ジョン・フォン・ノイマンが中性子が物質中を動き回る様子を探るために考案した手法。

モンテカルロ法を用いた積分を下図をもちいて説明する。

$$Iy = \int_1^2 -x^2 + 4dx \tag{1}$$

1. 領域 Iy を含む領域 Ir を定義する。（ここでは、$y = 0, y = 2, x = 1, x = 2$ で囲まれた領域）
2. 領域 Ir 内にランダムに点を打つ。
3. 2 の点のうち領域 Iy に含まれる点をカウントする。

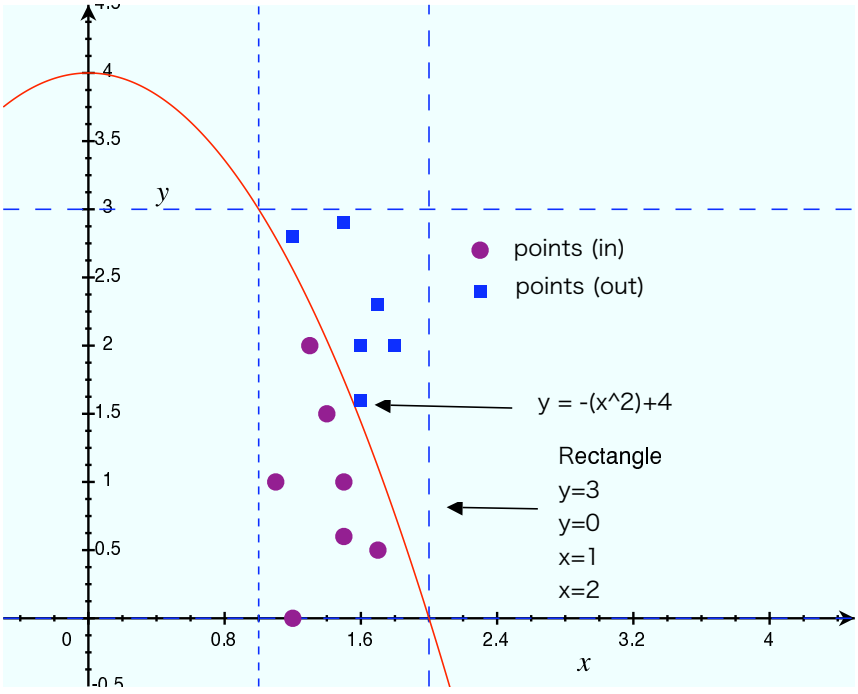

図 1　Monte Carlo Integral method

ランダムに打った点が領域 Iy 内に含まれる確率は領域 Iy の面積に比例するはずなので。

$$\frac{\text{領域 } Iy \text{ の面積}}{\text{領域 } Ir \text{ の面積}} \simeq \frac{\text{領域 } Iy \text{ 中の点の数}}{\text{点の総数}}$$
$$\longrightarrow \text{領域 } Iy \text{ の面積} \simeq \frac{\text{領域 } Iy \text{ 中の点の数}}{\text{点の総数}} (\text{領域 } Ir \text{ の面積})$$

よって積分の値を推定する事ができる。またランダムに打つ点の数が多いほど精度は高くなる。

2 シミュレーションプログラム

このレポートでは以下の式を計算する。

$$Iy = \int_1^2 \cos(\log x) dx \tag{2}$$

よって ly を含む長方形は $x = 1, x = 2, y = 0, y = 1$ になるので、長方形の面積 ir は 1 またランダムな点 (Px,Py) の範囲は以下のとおりである。

$$1 \leq Px \leq 2$$
$$0 \leq Py \leq 1$$

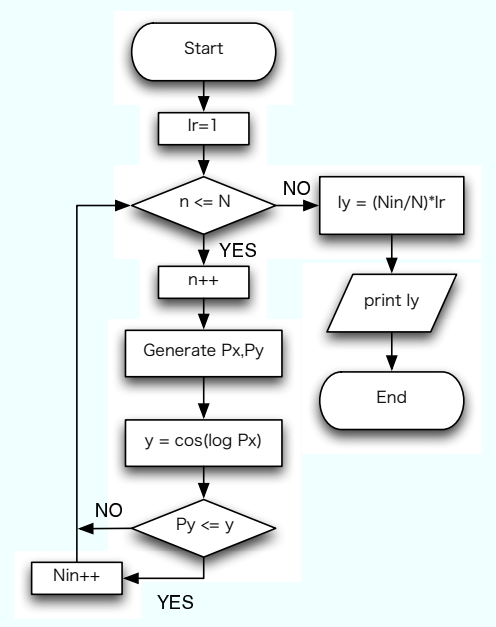

図 2 Simulation flow

終了条件 ($n < N$) に達するまで の間、以下を繰り返す。

1. ランダムな点 (Px,Py) を生成する
2. $y = \cos(\log Px)$ を計算
3. $Py \leq y$ なら ly に含まれる点の数 (Nin) をインクリメント
4. 1 にもどる

終わったら以下の式で積分を計算し、結果を出力

$$Iy = \int_1^2 \sin(\log x) dx \simeq \frac{Nin}{N} (Ir) \tag{3}$$

2.1 ソースコード

2.1.1 ヘッダファイル（mci.h）

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

#include <getopt.h>//実行時オプションの判定
#include <time.h>//rand 用 seed

#define DEFAULT_NUM 1000

void usage(char *myname);
```

2.1.2 シミュレーション本体 (mci.c)

```
#include "mci.h"

int main(int argc, char **argv){

  srand((unsigned)time(NULL));

  /*Initialize*/
  double
    Ir=1.0f*1.0f,//Rectangle : y=0,y=1,x=1,x=2
    Iy,//Integral y : integral_{1}^{2} cos(loge(x)) dx
    //random points
    Px,Py,
    y;//y = cos(loge(Px))

  int
    N = DEFAULT_NUM,//MAX Num
    n = 0,//counter
    Nin = 0;// number of internal dot

  //Read Exec options (実行オプションの読み込み)
  int option;
  while( (option = getopt(argc, argv,"n:h"))!=-1 ){
    switch(option){
```

```
case 'n':
N = atoi(optarg);
break;
case 'h':
default:
usage(argv[0]);
return 0;
break;
}
}
```

```
  printf("# N = %d\n",N);
  printf("# Px Py\n");
  for(n=1;n<=N;n++){
    //Generate random points
    //1 <= Px <= 2
    Px = ((double)rand()/RAND_MAX)+1;

    //0 <= Py <= 1
    Py = ((double)rand()/RAND_MAX);

    //Point
    printf("%f %f\n",Px,Py);

    //calculate y
    y=cos(log(Px));

    //if Py <= y, increment Nin
    if(Py <= y){
      printf("# y=%f so [in]\n",y);
      Nin++;
    }else printf("# y=%f so [out]\n",y);

  }

  // Iy/Ir = Nin/N so Iy = (Nin/N)Ir
  Iy = ((double)Nin/(double)N)*Ir;

  printf("\n## Result ##\n");
  printf("# Ir=%f,Nin=%d,N=%d\n",Ir,Nin,N);
```

```
  printf("# Iy = %f\n",Iy);


  return 0;
}

void usage(char *myname){
    printf("%s [nh]\n",myname);
    printf("\t -n <Loop Number>:Loop Number (default:10000)\n");
    printf("\t -h :show this message\n");
}
```

2.2 実行結果

```
# N = 1000
# Px Py
1.072960 0.245092
# y=0.997521 so [in]
1.253911 0.475616
# y=0.974511 so [in]
1.681577 0.266536
# y=0.867952 so [in]
1.663052 0.909962
# y=0.873401 so [out]
(中略)
1.866323 0.283331
# y=0.811565 so [in]

## Result ##
# Ir=1.000000,Nin=906,N=1000
# Iy = 0.906000
```

N=1000 で実行した結果、下図のような結果が出た。

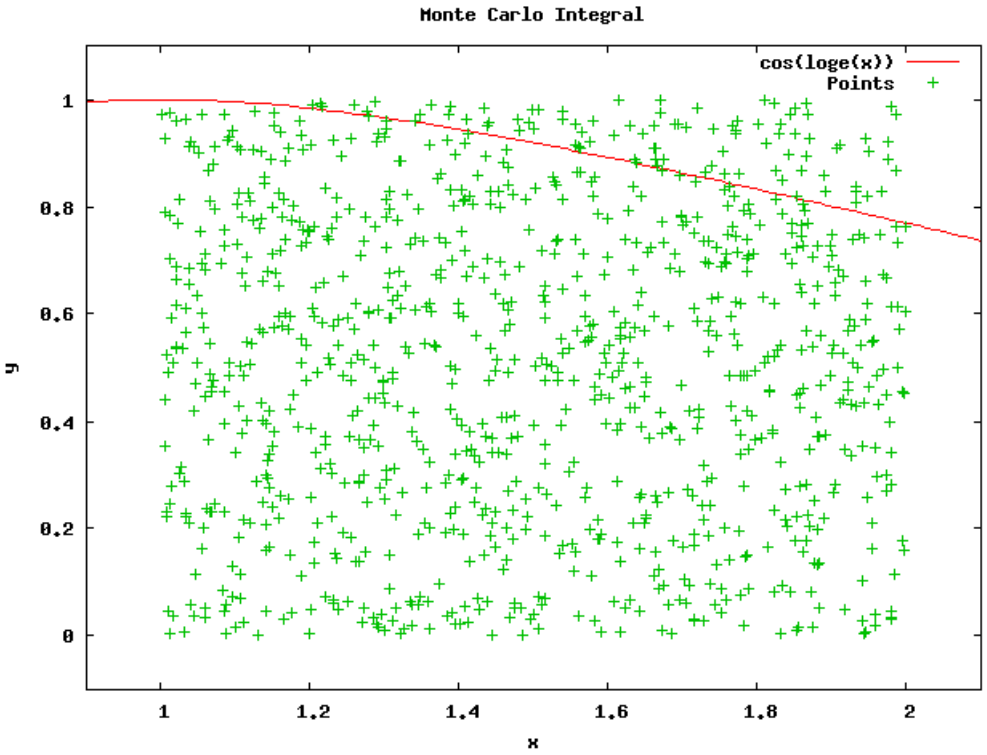

図 3　$y = \cos(\log x)$ とランダムな点

N=1000 で 100 回実行した際の平均値は 0.907980 であった、よって $\int_1^2 \cos(\log x)dx = 0.907980$ と推定される。

参考文献

[1] モンテカルロ法
http://ja.wikipedia.org/wiki/**モンテカルロ法**

[2] モンテカルロ積分法
http://www.akita-nct.ac.jp/yamamoto/lecture/2005/5E/integrate/integral_html/node