VIII. 粒子を用いた拡散のモンテカルロシミュレーション

１８．乱数

乱数とは，規則性なく発生させた数値のことである。複数個の数値を発生させた場合，１つの数値から次に発生する数値を予測することができない。すべての数値の発生確率が等しいものを一様乱数，ガウス分布のものを正規乱数とよぶ。コンピュータにより確定的な計算によって発生させた乱数のように見える数値を擬似乱数とよぶ。擬似乱数は，シミュレーションにおいて確率論的に起こる現象を再現するときに用いられる。

18.1 擬似乱数の発生法

ここで，擬似乱数を発生する方法である合同乗積法を紹介する。合同乗積法では非常に長い周期を持つ数値列を発生させることで乱数に見せかける。

$$R^{n+1} = \left(k + R^n + C\right) \bmod M \tag{18.1}$$

ここで，k, M, C は素数である。

$$k = 65539 \tag{18.2}$$

$$M = 2^{31} - 1 = 2147483647 \tag{18.3}$$

などを利用する。M をメルセンヌ数とよぶ。この方法では 0 から $M-1$ までの擬似乱数が発生する。

$$r^{n+1} = R^{n+1} / M \tag{18.4}$$

とすると，0 から 1 までの一様乱数を発生させることができる。

正規乱数を擬似的に発生させるには 0 から 1 までの一様乱数を 12 個たして 6 引けばよい。これは 0 から 1 までの一様分布の分散が 1/12 であることによる。

Fortran において，一様な整数の乱数を発生させるサブルーチンとして `random_number( )` が組み込まれている。引数に実数変数を入れると，0 から 1 までの一様乱数がその変数に代入される。実数は単精度でも倍精度のどちらでも良い。以下ではこの組み込みサブルーチンを利用する。乱数を発生させるには乱数シード (random seed，乱数の種？) が必要になるが，それを発生させるプログラムの書き方は独特なので，実際のプログラムを見てほしい。

１９．酔歩運動と拡散

拡散とは，酔歩運動(ランダムウォーク)によって不均質が物質中に広がっていく過程である。ここでは乱数により，酔歩運動を再現し，拡散過程が再現できるか調べよう。

19.1 酔歩運動とマルコフ過程

粒子がδt の間にδx だけ進むとする。ここでδx はランダムに決まるとする。このような場合，演習Aの資料VIII で示したとおり，粒子の密度$f(x,t)$はフォッカー・プランク方程式(Fokker-Plank equation)

$$\frac{\partial f}{\partial t} = -\frac{\langle \delta x \rangle}{\delta t}\frac{\partial f}{\partial x} + \frac{\langle \delta x^2 \rangle}{2\delta t}\frac{\partial^2 f}{\partial x^2} \tag{19.1}$$

に従う。ここで,

$$v = \frac{\langle \delta x \rangle}{\delta t} \tag{18.2}$$

は流速,

$$D = \frac{\langle \delta x^2 \rangle}{2\delta t} \tag{18.3}$$

は拡散係数を表す。つまり，この式は移流拡散方程式

$$\frac{\partial f}{\partial t} = -v\frac{\partial f}{\partial x} + D\frac{\partial^2 f}{\partial x^2} \tag{18.4}$$

を表す。左右へ動く確率が等しい，つまり,

$$\langle \delta x \rangle = 0 \tag{18.5}$$

ときには

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2} \tag{18.6}$$

となり，拡散方程式を表す。つまり拡散は固体中などでランダムに粒子が運動することによって起きること，また，そのような場合には拡散方程式が成立することを示している。

2.2 拡散現象のモンテカルロ・シミュレーション

ここでは, 実際にシミュレーションを行って酔歩運動により拡散現象が再現できるか試してみよう。粒子は最初に原点x=0に密集しているとする。粒子の番号をiとすると,

$$x_i = 0 \tag{18.7}$$

ここで$<\delta x^2>=1$ となるような正規乱数を発生させ，

$$x_i^{n+1} = x_i^n + \delta x_i \tag{18.8}$$

を計算する。ただし$\delta t=1$ とする。このとき，粒子の座標を区間Δx に区切って区間ごとの粒子数をカウントする。カウントした粒子は拡散方程式

$$\frac{\partial f}{\partial t} = D\frac{\partial^2 f}{\partial x^2} \tag{18.9}$$

に従うはずである。この方程式の解は，初期条件をデルタ関数

$$f(x,0) = \delta(x) \tag{18.10}$$

としたときに

$$f(x,t) = \frac{1}{2\sqrt{\pi Dt}} e^{-x^2/4Dt} \tag{18.11}$$

である。ただし，デルタ関数$\delta(x)$は

$$\delta(x - x_0) = \lim_{\varepsilon \to 0} \frac{1}{\varepsilon} \qquad |x - x_0| \le \varepsilon/2 \tag{18.12}$$

$$\delta(x - x_0) = 0 \qquad |x - x_0| \ge \varepsilon/2 \tag{18.13}$$

となる関数である。いま，

$$\int_{-\infty}^{\infty} f(x,t)dx = N_0 \tag{18.14}$$

$$D = \frac{1}{2} \tag{18.15}$$

であるので，その解は

$$f(x,t^n) = \frac{N_o}{\sqrt{2\pi n}} \exp\left[\frac{-x^2}{2n}\right] \tag{18.16}$$

となる。ただし，n はタイムステップを表す。この式は平均 0 分散 n の正規分布を表す。

問題 VIII-1

(1) プログラムを実行せよ。

(2) 粒子密度の分布が正規分布と一致することを示せ。

(3) 熱伝導方程式のプログラムを変更して，別の数値計算法で解いた拡散方程式の解も

正規分布と一致することを確認せよ。

参考文献

矢部 孝・福田 昌宏・川田 重夫, シミュレーション物理入門―超粒子モデルの世界, 朝倉書店, 1989.

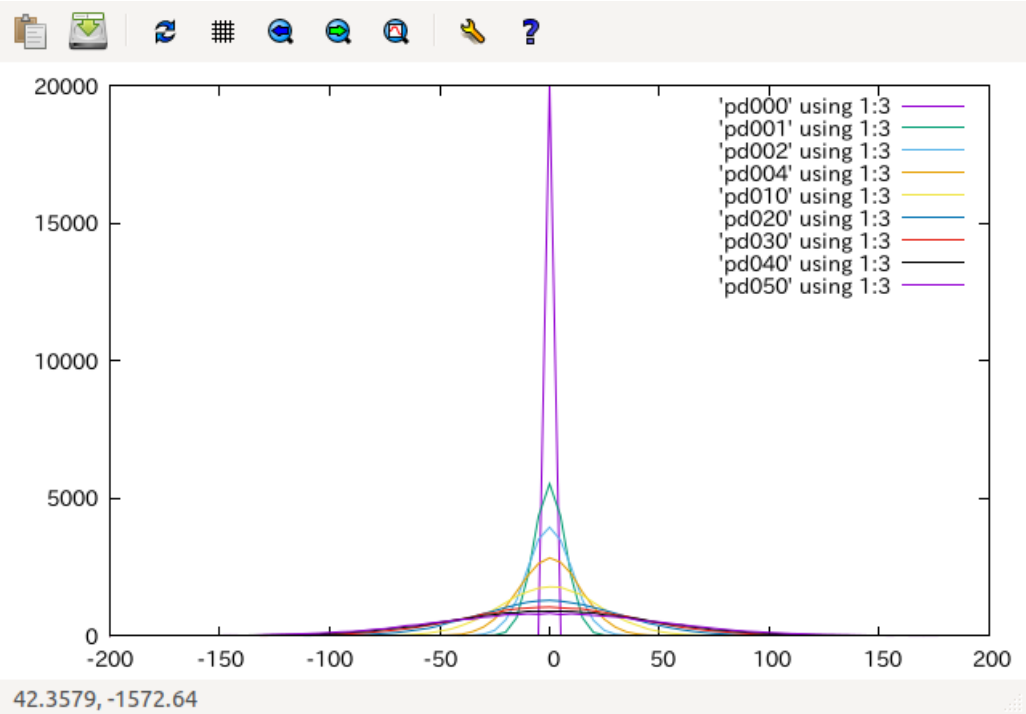

図２　拡散の粒子シミュレーション。図は単位長さ辺りの粒子数を表す。

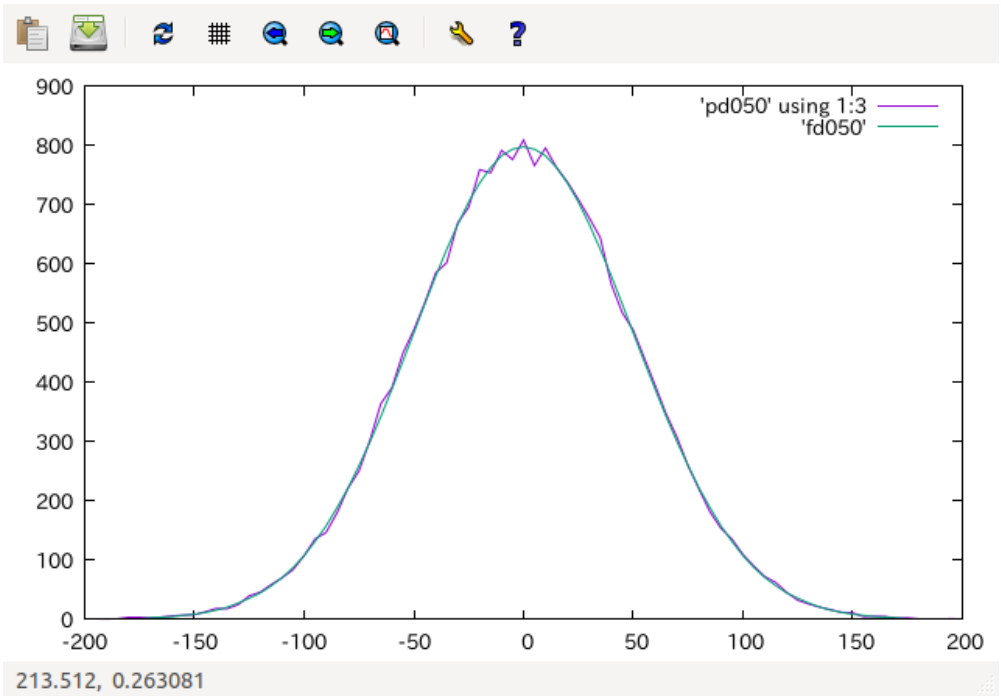

図３　粒子シミュレーション(紫)と解析解(緑)による粒子密度の比較。

問題 VIII-1　プログラム

拡散をモンテカルロ法により解くプログラムである。プログラムはメインルーチン，３つのサブルーチン，２つのモジュールからなる。コンパイルの途中でモジュールをバイナリ化した中間ファイル(拡張子.mod のファイル)が必要になるので，コンパイルは

```
f95 −c diff_mod_para.f90
f95 −c diff_mom_main.f90 diff_ medisp.f90 diff_distrib.f90 diff_ranseed.f90
f95 −o df.out diff_*.o
```

のように行うこと。-o と-c を間違えるとファイルが消えることもあるので，バックアップをするなど注意すること。

メインプログラム　diff_mon_main.f90

```
! ------------------------------------------------------------------------
! *   Montecallo simulation of diffusion                                 *
! ------------------------------------------------------------------------

   program diff_monte

      use mod_para

      implicit none

      real(8), allocatable:: dx(:)
      real(8), allocatable:: x(:), xL(:)
      real(8):: r1(12), rnorm
      real(8):: t
      integer:: it,  i,  iL,  j
      integer:: nt,  nf, nif
      character(60):: apara

!  ****  read parameters  ****
```

```
      open ( 10, file='diff_para.dat', status='old' )

      read ( 10,* )
      read ( 10,* ) apara
      read ( 10,* ) NP
      write ( 6,* ) apara
      write ( 6,* ) NP
      read ( 10,* ) apara
      read ( 10,* ) nt
      write ( 6,* ) apara
      write ( 6,* ) nt
      read ( 10,* ) apara
      read ( 10,* ) dt
      write ( 6,* ) apara
      write ( 6,* ) dt
      read ( 10,* ) apara
      read ( 10,* ) Lx
      write ( 6,* ) apara
      write ( 6,* ) Lx
      read ( 10,* ) apara
      read ( 10,* ) nx
      write ( 6,* ) apara
      write ( 6,* ) nx
      read ( 10,* ) apara
      read ( 10,* ) nif
      write ( 6,* ) apara
      write ( 6,* ) nif

!  ****  allocate dimension variables  ****

      allocate ( x(NP), dx(NP) )
      allocate ( xL(-nx:nx) )
```

```
!  ****  interval length for distribution function  ****

      dLx = Lx / dfloat( nx )

      write ( 6,* ) dLx

!  ****  coordinates of distribution function

      do iL = -nx,nx
        xL(iL) = dLx * dfloat( iL )
      end do

!  ****  set particles at inital position  ****

      x(:) = 0.0d0
      call ranseed( )

      it = 0
      nf = 0

!  ****  calculate mean, rms, and distribution function  ****

      open ( 20, file='t_sig_dis.dat' )

      call medisp ( x,  t,  it, nif )
      call distrib( x,  xL, it, nf )

!  ****  loop of time marching
*********************************************************

      do it = 1,nt

        t = t + dt
```

```
!  ****  calculate increment of particle movement  ****

        do i = 1,NP

!  ****  calculate normal random number  ****

          rnorm = 0.0d0
          call random_number( r1 )
          do j = 1,12
            rnorm = rnorm + r1( j )
          end do

          dx(i) = rnorm - 6.0d0

        end do

!  ****  advance particle  ****

        x(:) = x(:) + dx(:)


!  ****  output to file  ****

        call medisp ( x,  t,  it, nif )

        if ( mod( it,nif ) == 0 ) then
          nf = nf + 1
          call distrib( x,  xL, it, nf )
        end if

      end do
```

```
!      call thdist( xL, t,  it, nf )

!  ****  end of time marching
*****************************************************

      close ( 20 )
      stop

   end program diff_monte
```

use mod_para：モジュール副プログラム mod_para を呼び出す。
! **** calculate normal random number ****から dx(i) = rnorm - 6.0d0 まで：0 から 1 の一様乱数を 12 個足し合わせて 6 引くことにより，正規乱数を擬似的に作り出している。
call ranseed( )：乱数の計算に必要な乱数の種(random seed)を得るサブルーチン ranseed を呼び出す。引数がない場合はこのように空白で良い。
call random_number(r1)：組み込みサブルーチンを呼び出して一様乱数を得ている。random_number は組み込みサブルーチンと呼ばれる Fortran95 の機能である。乱数の値は引数の変数に出力される。r1 は配列変数で、配列変数を引数に入れると，その配列の数だけ乱数を作る。
!   call thdist(xL, t, it, nf) ：コメント文。ここで解析的な分布関数を計算するためのサブルーチンを呼び出すようにする。

サブルーチンプログラム　diff_medisp.f90

粒子位置全体の平均と分散を計算する。シミュレーションが精度良くできているか示す指標を示す。結果は t_me_dis.dat に出力される。

```
! ------------------------------------------------------------------
! *   calculate mean and dispersion of particle position             *
! ------------------------------------------------------------------
   subroutine medisp ( x, t, it, nif )

      use mod_para
```

```
      implicit none
      real(8):: t, xmean, xdisp
      real(8):: x(NP)
      integer:: it, i, nif

!  ****  calculate mean and dispersion  ****

      xmean = 0.0d0
      xdisp = 0.0d0

      do i = 1,NP
        xmean = xmean + X(i)
        xdisp = xdisp + X(i)**2
      end do

      xmean = xmean / dfloat( NP )
      xdisp = xdisp / dfloat( NP )

!  ****  write to file  ****

      if ( mod( it,nif ) == 0 ) write ( 6,* )  t, xmean, xdisp
      write ( 20,* ) t, xmean, xdisp

      return

   end subroutine medisp
```

サブルーチンプログラム　diff_distrib.f90

粒子の分布関数を計算する。粒子密度を単位長さ辺りの粒子数で表す。

```
! ---------------------------------------------------------------------
! *   calculate distribution function of particles                    *
! ---------------------------------------------------------------------
```

```
   subroutine distrib( x,  xL, it, nf )

      use mod_para

      implicit none

      integer, allocatable:: iLp(:), mLp(:)
      integer:: it, nf, i, iL

      real(8):: x(NP)
      real(8):: xL(-nx:nx)

      real(8), allocatable:: dLp(:)
      character(3):: cnf
      character(5):: fname1

!  ****  allocate dimension variables  ****

      allocate ( iLp(NP) )
      allocate ( mLp(-nx:nx), dLp(-nx:nx) )

!  ****  position of particle on L grid  ****

      iLp(:) = int( ( x(:) + Lx + 0.5d0 * dLx ) / dLx ) - nx

!  ****  count particles and distribution function  ****

      mLp(:) = 0
      do i = 1,NP
        if ( iLp(i) >= -nx .and.  iLp(i) <= nx )  mLp( iLp(i) ) = mLp( iLp(i) ) + 1
      end do

      dLp(:) = dfloat( mLp(:) ) / dLx
```

```
!  ****  write distribution function  ****

      write ( cnf,'(i3.3)' ) nf
      fname1 = 'pd'//cnf

      open ( 21,file=fname1 )

      do iL = -nx,nx
        write ( 21,* ) xL(iL),mLp(iL),dLp(iL)
      end do

      close ( 21 )
      return

   end subroutine distrib
```

use mod_para：モジュール副プログラム mod_para を呼び出す。
iLp(:) = int((x(:) +…：代入文。1 つ 1 つの粒子がどの区間に入るか計算している。iLp はその粒子が何番目(x=0 のまわりが iLp=0)かを示すインデックスである。
write (cnf,'(i3.3)') nf：数値を 3 文字の文字列に変換している。Fortran 独特の書き方。
write (cnf,'(i3.3)') nf：代入文。右辺は文字列の連結式である。ファイル名を作っている。
粒子数密度のファイルは pd001 のような名前で出力される。

サブルーチンプログラム　diff_ranseed.f90

random seed をクロック値から取得する。

```
! ---------------------------------------------------------------------
! *   get random seed                                                 *
! ---------------------------------------------------------------------
   subroutine ranseed( )

      implicit none
```

```
      integer, allocatable:: seed(:)
      integer:: nrand
      integer:: clock

!  ****  initialize  ****

      call system_clock( count=clock )
      call random_seed( size=nrand )
      allocate ( seed( nrand ) )

      seed(:) = clock

      call random_seed( put=seed )

      return

   end subroutine ranseed
```

call system_clock(count=clock)：乱数シードをリセットするためには，最初になんらかの値を入力しなければならない。この数を得るため，変化していくシステムクロックを用いることにする。そのため，システムクロックを調べる組み込みサブルーチン system_clock を呼び出し，クロックのカウント値を得る。クロックのカウント値を整数変数 clock に代入する。
call random_seed(size=nrand)：Fortran の仕様上，乱数シードをリセットするため数の入力には配列変数を使う必要がある。その変数の配列サイズは，システムやコンパイラによって異なる。このため，組み込みサブルーチン random_seed を呼び出して，配列サイズをまず調べている。配列サイズの値が nrand に代入される。
seed(:) = clock：配列機能を使った代入文。seed の配列すべてに同じ clock の値が代入される。
call random_seed(put=seed) ：組み込みサブルーチン random_seed を呼び出して乱数シードを計算させる。seed には乱数シードを計算するためのクロック値が入っている(出力ではない)。これによって乱数シードがリセットされる。この新しい乱数シードは，次に random_number を呼び出したときに渡されて使われる。

モジュール副プログラム　diff_mod_para.f90

複数のサブルーチンで共通に使われる変数の型を宣言する。

```
! ------------------------------------------------------------------------
! *   set common physical variables                                      *
! ------------------------------------------------------------------------
   module mod_para

      implicit none

      integer:: NP,  nx
      real(8):: dt,  dLx,  Lx

   end module mod_para
```

パラメータデータファイル　diff_para.dat

パラメータの値を与える。

```
#  parameters for monte-carlo diffision simulation
'# (1) number of particles '
100000
'# (2) number of time steps '
2500
'# (3) time-step increment '
1.0d0
'# (4) half length to calculate distribution function'
200.0d0
'# (5) section numbers to count particles '
40
'# (6) interval number of time steps for d.func. output '
50
```

(1) 粒子数，(2) タイムステップ数，(3) 1 タイムステップの時間，(4) 分布関数を計算する領域の半分(正の部分)の長さ，(5) 粒子密度を積算する区間の長さ，(6) 分布関数ファイルを出力するタイムステップ数の間隔