Fortran プログラミング 応用編

１．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。

1.1 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。

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

ここで，k, M, C は素数である。

$$k = 65539 \tag{2}$$

$$M = 2^{31} - 1 = 2147483647 \tag{3}$$

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。

$$r^{n+1} = R^{n+1} / M \tag{4}$$

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。

1.2 擬似乱数による円周率の計算

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。

ここでは乱数を利用して円の面積を計算し，円周率πの値を求める。いま，2×2 の辺を持つ正方形の中に接する半径 1 の円を考える。円の面積はπであり，正方形の面積は

4 である。これを的と思ってランダムに石を投げたとすると，正方形にぶつかった石の中で円の中に入る確率は$\pi/4$ である。これは積分

$$I = \int_0^1 \sqrt{1-x^2}\, dx \tag{5}$$

に等しい。この考察から，２つの０から１の値を持つ一様乱数(r, s)を発生させたとき，

$$s \leq \sqrt{1-r^2} \tag{6}$$

となる確率は$\pi/4$ であることが分かる。つまり，２組の一様乱数を発生させて(6)のようになる確率を計算することによって積分ができるのである。このような確率論的に計算を行う方法を**モンテカルロ法**(Monte-Carlo method)とよぶ。

問題 18

(1) プログラムを入力しπの値を計算せよ。

(2) 乱数の組を増やしたときに，結果がπに近づいていくか調べよ。

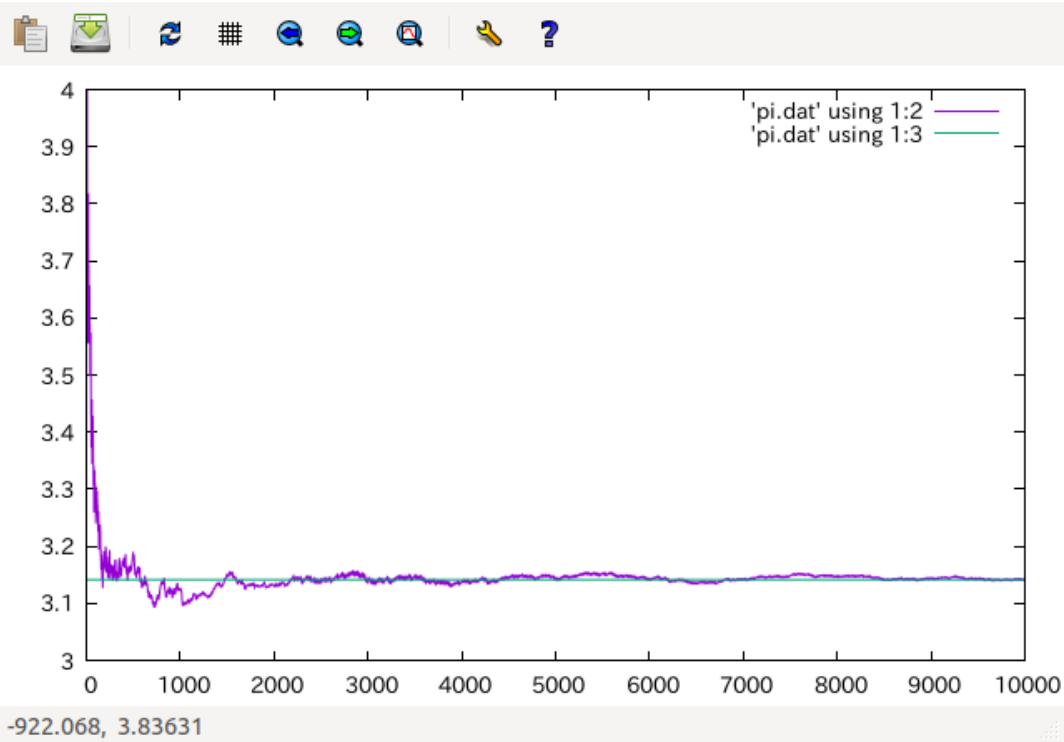

図１　モンテカルロ法による円周率の計算。

問題 18　プログラム

積分をモンテカルロ法により解いてπの値を求めるプログラムである。

メインプログラム　pi_mon_main.f90

```
!
!  ****  Search value of pi by monte-carlo method  ****
!
   program pi_monte

      implicit none

      real(8):: pi, pitrue, diffpi
      real(8):: x, y, y_x
      integer:: nmax, i, np, ip

      pitrue = 4.0d0 * atan( 1.0d0 )

      write ( 6,* ) 'Input number of trials'
      read ( 5,* ) nmax

!  ****  open file  ****

      open ( 10, file='xy.dat' )
      open ( 11, file='pi.dat' )

      call ranseed( )

      np = 0

!  ****  Loop of trials  ************************************************************

      do i = 1, nmax
```

```
        ip = 0

        call random_number( x )
        call random_number( y )

        y_x = sqrt( 1.0d0 - x**2 )

!  ***  case when y < sqrt( 1.0d0 - x**2 )  ****

        if ( y < y_x ) then
          np = np + 1
          ip = 1
        end if

!  **** Calculate pi  ****

        pi = 4.0d0 * dfloat( np ) / dfloat( i )
        diffpi = pi - pitrue

        write ( 10,* ) x,y,ip
        write ( 11,* ) i, pi, pitrue

      end do

      close ( 10 )
      close ( 11 )

!  ****  End of loop  ***************************************************************

      stop

    end program pi_monte
```

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(x)：乱数を計算するサブルーチン random_number を呼び出す。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。

サブルーチンプログラム　pi_mon_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。