Fortran プログラミング入門

Fortran プログラムの作成と実行　その３

１．正弦波をプロットする: 繰り返しと do 文

正弦波すなわち

$$y = A\sin\left(\frac{2\pi t}{T}\right)$$

のグラフをプロットすることを考える。計算機の中では、時空を連続的に扱うことは出来ないので、離散的に扱う。つまり、時間 t は

$$t = t_i$$

ただし、i は 1 から n+1 までとする。i と i+1 の間の時間間隔(time interval)を Δt とすると、時間 t は

$$t_i = \Delta t\left(i-1\right)$$

で表される。このとき、

$$y_i = A\sin\left(\frac{2\pi t_i}{T}\right)$$

である。プログラムでは、時間 t や sine の値を繰り返し計算する。この繰り返しには do 文に夜繰り返し(do ループ)が用いられている。(繰り返しを行うには他に go to 文を使う方法が考えられるが、余り使われない。go to 文を使う際には、無限ループを作ってしまわないように注意が必要である。回数が決まっている場合は do 文を使用した方がよい。) 計算に並列性がある場合、do 文にはベクトル化や並列化などのメリットがある場合もある。

問題 F8

(1) プログラムを入力、実行せよ。

(2) GNUplot 等を用いてグラフを図に表せ。

(3) 周期の異なる２つの正弦波を重ね合わせるプログラムを作成せよ。２つの波の周期がわずかに異なるとき、どのような現象が起きるか、シミュレーションせよ。結果を図に表示せよ。

２．正弦波をプロットする: 配列変数

前のプログラムは do 文で正弦波の変位を計算し、すぐに変位をファイルに書き込むようになっている。ここで紹介するプログラムでは、計算をすべて行って変数に書き込んでから、最後にまとめてファイルに出力するようになっている。時系列に対するたくさんの数値を変数に記憶するため、配列変数を使用している。配列変数には添え字が付くので、その添え字を変化させながら、変数を利用することが出来る。このため、プログラムの繰り返し(do 文)と組み合わせて１つの式を利用することができる。

問題 F9

プログラムを入力、実行せよ。

問題 F10

プログラムを入力、実行せよ。

問題 F11

点の２次元運動の座標(x, y)が時間 t を媒介変数として

$$x = A\sin(2\pi mt)$$
$$y = A\sin(2\pi nt + \delta)$$

と表されるとする。ただし、m, n は整数、δ は初期位相(ラジアン)である。得られる図形をリサジュー図形という。

(1) 点の座標(x, y)を計算し１つのファイルに保存するプログラムを作成せよ。

(2) 何通りかの m, n 組み合わせで計算を実行し、図を作成せよ。

問題 F12

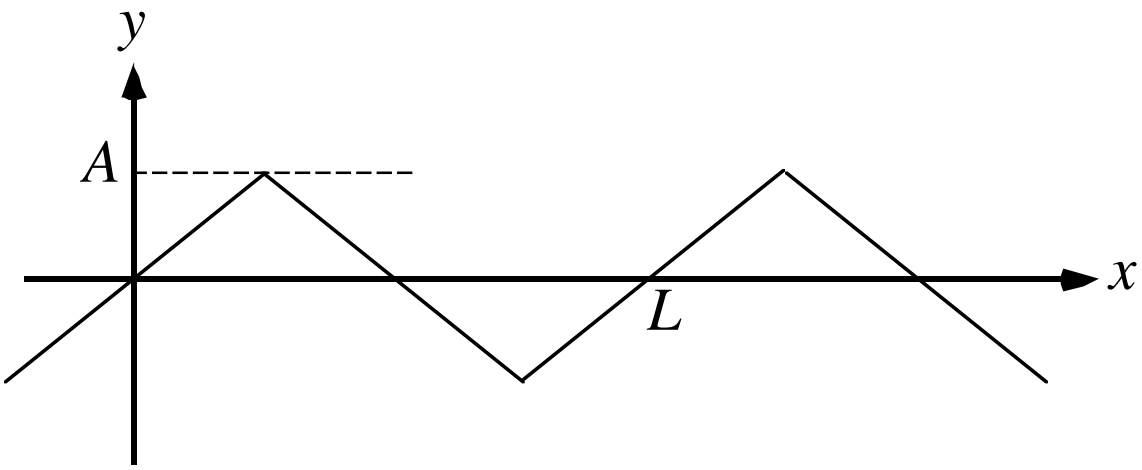
三角波(triangle wave)を数周期分計算し、作図する。

(1) 波長 L、振幅 A の三角波を x の関数として式に表せ。y が増加する区間と減少する区間で異なる関数となることに注意せよ。

(2) 関数を計算し、結果をファイルに出力するプログラムを入力せよ。

(3) プログラムを解説せよ。

(4) 結果をグラフに表せ。



GNUplot によるデータの可視化

sine.dat をグラフに表示する作業を行う。

１．グラフの画面への表示

sine.dat が出来ていることを確認する。

```
ls -l
```

と入力。

gnuplot を起動する

```
gnuplot
```

と入力。プロンプトが“gnuplot>”に変わるのを確認する。以下はほんのさわりであるので、gnuplot でどんな作図が出来るのかをインターネットで検索して調べてみよう。

“sine.dat”のグラフを表示させる

```
plot "sine.dat"
```

これで画面にグラフが表示される。3 カラム以上ある場合はどのカラムを x 軸、y 軸とするか選択する (付録参照)。レポートに図を貼り付ける場合は、ウィンドウをキャプチャーする。

以下は、グラフのみを印刷する場合の手順である。ここではする必要がない。

２．グラフのポストスクリプトファイルへの保存

画像の出力先をファイルに指定する。GNUplot では直接ポストスクリプトへ出力できる。

```
set output "graph1.ps"
set terminal postscript
```

と入力すると出力先が画面からポストスクリプトファイルに変更される。この後、上の plot コマンドを実行する。

gnuplot を終了する。

```
quit
```

と入力。

３．ポストスクリプトファイルの PDF への変換

出力のファイルができているか確認する。

```
ls -l
```

出力のポストスクリプトファイル graph1.ps が出来ているか確かめる。

```
ps2pdf graph1.ps
```

で PDF に変換したファイルが graph1.pdf に保存される。

４．出力されたファイルをプリントする

```
lpr graph1.ps
```

と入力する。lpr はポストスクリプトファイルをプリンターに送るコマンド。

プログラムなどのテキストファイルの出力には a2ps を使う。例えば、

```
a2ps sine.f
```

とする。

問題 F8 プログラム

```
!
!    plot sine curve
!
      program sine

      implicit none
      real(4):: Tp,a,pi
      real(4):: y,t,dt
      integer:: ns,np,i,n

      write (6,*) 'I will draw sine curve'
      write (6,*) 'Input period (s)'
      read  (5,*) Tp
      write (6,*) 'Input Amplitude'
      read  (5,*) A
      write (6,*) 'How many sample points for one period'
      read  (5,*) ns
      write (6,*) 'How many period to be plotted'
      read  (5,*) np

      n = ns*np+1
      pi = 4.0e0 * atan(1.0e0)
      dt = Tp / float(ns)

      open ( 10,file='sine.dat' )

      t = 0.0e0
      do i = 1,n
        t = dt * float(i-1)
        y = A * sin( 2.0e0*pi*t/Tp )
        write (10,*) t,y
```

```
      end do

      stop
      end program sine
```

read (5,*) Tp：入力文。正弦波の周期 Tp を読み込む。

read (5,*) A：正弦波の振幅 A を入力。

read (5,*) ns：時間軸をディジタル的に表すため、１周期の時間を区切る個数(サンプリング数)を入力。

read (5,*) np：プロットしたい正弦波の周期数を入力。

n = ns*np+1：代入文。正弦波を表す点の数を計算している。１をプラスしているのは t=0 の点を含めるため。

dt = Tp / float(ns)：代入文。２つの点の時間差を計算して dt とする。

open (10,file='sine.dat')：補助入出力文。出力するファイルの名前を sine.dat に指定し、装置番号 10 を割り振っている。

do i = 1,n：do 文と end do 文の間にある行を繰り返し実行する。i はカウンターの役割をしており、i を１から n まで１ずつカウントアップする。つまり、ここでは n 回実行される。

t = dt * float(i-1)：代入文。i 番目の点の時間を計算している。ここでは t=0 の点を

y = A * sin(2.0e0*pi*t/Tp)：代入文。i 番目の点、すなわち時間 t における変位を計算している。

write (10,*) t,y：出力分。時間 t と変位 y の値を装置番号 10 に書き込む。

end do：do 文の終わりであることを示す。必ず、do 文と end do 文はペアで使用する。

問題 F9 プログラム

```
c2345&78901234567890223456789032345678904234567890523456789062345678907 2

c
c    plot sine curve
c
      program sine2

      implicit none
      real(4):: Tp,a,pi,dt

      integer:: i

      integer, parameter:: ns = 100, np = 3
      integer, parameter:: n = ns * np +1

      real(4):: y(n),t(n)

      write (6,*) 'I will draw sine curve'
      write (6,*) 'Input period (s)'
      read  (5,*) Tp
      write (6,*) 'Input Amplitude'
      read  (5,*) A

      pi = 4.0e0 * atan(1.0e0)
      dt = Tp / float(ns)

      open ( 10,file='sine.dat' )

      do i = 1,n
        t(i) = dt * float(i-1)
        y(i) = A * sin( 2.0e0*pi*t(i)/Tp )
```

```
end do

do i = 1,n
  write (10,*) t(i),y(i)
end do

stop
end program sine2
```

integer, parameter:: ns = 100, np = 3：パラメータ文。( )の中でパラメータの値を指定する。パラメータは変数ではなく、数値を記号で置き換えたものである。このため、プログラム実行中に値が変化しない。パラメータ文以外の場所で、数値が代入されることはない。代入文の左辺にパラメータがあると、コンパイル時にエラーになる。

Real(4):: y(n),t(n)：配列宣言。配列変数の大きさを指定する。同時に y と t が実数型であることを宣言している。( )の中は数値またはパラメータ*1 でなければならない。このため、プログラミングの時、実行時使用する配列の大きさを考えておかなければならない。配列の添え字は１から始まる整数である。０や負の値、１以外の正の値から始めたいときは

```
real(4):: a(0:n),b(-5:n),c(3:100)
```

と書く。２次元以上の配列の時には、配列の大きさを“,”で区切って

```
real(4):: a(m,0:n),c(k,l:m,50)
```

のように書く。Fortran では、左側の添え字が変化する方がメモリ上に連続的なアドレスで領域が割り当てられる。コンピュータのメモリはアドレスを連続して読み出す方が高速なので、２重以上の do 文では、左側の添え字になる方を内側のループにすべきである。

*1 サブルーチンでのみ、変数を( )に入れることが出来る(整合配列)。

問題 F10 プログラム

```
c2345&7890123456789022345678903234567890423456789052345678906234567890 72

c
c    plot sine curve
c
      program sine
      implicit none
      real(4):: Tp,a,pi,dt
      integer:: ns,np,i,n

      real(4),allocatable:: y(:),t(:)

      write (6,*) 'I will draw sine curve'
      write (6,*) 'Input period (s)'
      read  (5,*) Tp
      write (6,*) 'Input Amplitude'
      read  (5,*) A
      write (6,*) 'How many sample points for one period'
      read  (5,*) ns
      write (6,*) 'How many period to be plotted'
      read  (5,*) np

      n = ns*np+1
      allocate ( y(n),t(n) )

      pi = 4.0e0 * atan(1.0e0)
      dt = Tp / float(ns)

      open ( 10,file='sine.dat' )

      do i = 1,n
```

```
  t(i) = dt * float(i-1)
  y(i) = A * sin( 2.0e0*pi*t(i)/Tp )
end do

do i = 1,n
  write (10,*) t(i),y(i)
end do

stop
end
```

配列変数の動的割り付けを使用したバージョンである。allocatable という宣言文と allocate という割り付けの実行文の組み合わせを使用する。実行後に配列の大きさを決めることが出来る。ここでは、全部の点数 n が決まった後に割り付けを実行している。

動的割り付けは Fortran 77 では対応していないので、コンパイルには Fortran 95 のコンパイラ f95 を使用する。

`real(4),allocatable:: y(:),t(:)`：動的割り付けをする変数の配列宣言。数値の代わりに“:”を使用する。２次元配列の時は

```
real(4),allocatable:: a(:,:),b(:,:)
```

のように書く。実数宣言と動的割り付け変数の宣言は同時に書くことが出来る。ここでは、Fortran90 の新しい書き方を使用している。この書き方では、コロン２つの左側に属性、右側に変数のリストを書く。

`allocate (y(n),t(n))`：配列変数に大きさを割り付ける。ここで、メモリ上の領域が確保される。n は real 文などの時と異なり、変数を入れることが出来る。

問題 F12 プログラム

```
! ------------------------------------------------------------------------
! **  plot triangle wave                                                **
! ------------------------------------------------------------------------
      program triangle

      implicit none

      integer:: i,    j,    is
      integer:: mcyc, msam, mtime
      real(8):: Tperi, a
      real(8):: t,     t2,   dt
      real(8),allocatable:: g(:)
!
!  **** Input control parameters  ****
!

!
!   ****  period  ****
!
      write (6,*) 'Input period'
      read (5,*) Tperi
      write (6,*) 'Tperi = ',Tperi
!
!   ****  cycles  ****
!
      write (6,*) '# of cycles for plot'
      read (5,*) mcyc
      write (6,*) 'mcyc = ',mcyc
!
!   ****  sampling freqency for one period  ****
!
```

```
      write (6,*) '# of sampling for one period'
      read (5,*) msam
      write (6,*) 'msam = ',msam
!
!   ****  amplitude  ****
!
      write (6,*) 'amplitude'
      read (5,*) a
      write (6,*) 'a = ',a
!
!    ****  time interval and # of time steps  ****
!
      dt = Tperi / dfloat( msam )
      write (6,*) 'dt = ',dt
      mtime = mcyc * msam
      write (6,*) 'mtime = ',mtime
!
!  ****  allocate dimension variables  ****
!
      allocate( g(mtime) )
!
!  ****   open file  ****
!
      open ( 11, file='tw.dat' )
!
!  ****   triangle wave  ****
!
      do is = 1,mcyc
        do i = 1,msam/4
          j = (is-1)*msam+i
          t = dt * 0.5d0 + dt * dfloat(j-1)
          t2 = dt * 0.5d0 + dt * dfloat(i-1)
          g(j) = a * t2 / ( 0.25d0 * Tperi )
```

```
!          write (6,*) t,t2,g(j)
           write (11,*) t,g(j)
         end do
         do i = msam/4+1,3*msam/4
           j = (is-1)*msam+i
           t = dt * 0.5d0 + dt * dfloat(j-1)
           t2 = dt * 0.5d0 + dt * dfloat(i-1)
           g(j) = a * ( 0.5d0 * Tperi - t2 ) / ( 0.25d0 * Tperi )
!          write (6,*) t,t2,g(j)
           write (11,*) t,g(j)
         end do
         do i = 3*msam/4+1,msam
           j = (is-1)*msam+i
           t = dt * 0.5d0 + dt * dfloat(j-1)
           t2 = dt * 0.5d0 + dt * dfloat(i-1)
           g(j) = a * ( t2 -  Tperi ) / ( 0.25d0 * Tperi )
!          write (6,*) t,t2,g(j)
           write (11,*) t,g(j)
         end do
       end do

       close (11)

       stop
       end program triangle
```

まとめ

繰り返し

```
do i=1,n
  …
end do
```

…を i=1 から n になるまで繰り返す。i は 1 ずつカウントアップされる。

```
do i=0,7,2
```

と書くと i=0,2,4,6 の 4 回繰り返す。

配列変数

`real*4 a(100)`：1 次元配列変数の領域を宣言。大きさは数値またはパラメータで指定する。

`real*4 a(100,200)`

`real*4 a(0:1000)`

`real*4, allocatable:: b(:)`：動的割り付けをする変数の配列宣言

`allocate (b(n))`：allocatable に指定された配列変数に使用領域を割り付ける。

Gnuplot のコマンド

`plot "datafile"` ：点でグラフのプロット

`plot "datafile" with line` ：線でグラフをプロット

`plot "datafile" using 1:2` ：データの 1 コラム目を x 軸、2 コラム目を y 軸としてグラフをプロット

`plot "datafile" using 1:3 title "example"` ：データの 1 コラム目を x 軸、3 コラム目を y 軸としてグラフをプロット、右上の注釈に example と名前を付ける。

`replot "datafile2"` ：別のファイルのグラフを重ねてプロットする

`set title "Graph1"`：グラフに Graph1 とタイトルを付ける

`xlabel "axis1"`： x 軸に axis1 とラベルを付ける

`set yrange [-1.0:2.0]`： y 軸の範囲を-1.0 から 2.0 までに指定する

`set xtics 1.0`： x 軸に 1.0 ごとに目盛りを付け、値を入れる。

`set mxtics 5`：目盛りの間を 5 等分して数値のない小目盛りを入れる。

`set terminal postscript`：グラフの出力先をポストスクリプトファイルに指定する

`set output "graph1.ps"`：グラフの出力先のファイル名を graph1.ps に指定する

`save "graph1.plt"`：Gnuplot で作成したグラフを保存する

付録

自分のパソコンに Fortran と gnuplot をインストールする

Windows PC の場合

Fortran

以下のホームページを参考にする。

http://d.hatena.ne.jp/arakik10/20120214/1329167074

実際にやることは、

1. TDM-GCC のホームページ(http://tdm-gcc.tdragon.net/download)から tgm-gcc をダウンロード(32bit 版または 64bit 版を OS に応じて選ぶ)
2. インストーラで、Fortran (gfortran)を選択して、gcc と gfortran をインストール
3. Fortran プログラムを置くフォルダを作る
4. Fortran プログラムを置くフォルダにコマンドプロンプトのエリアスを作る。
5. Gfortran と入力して gfortran が動くか確認
6. プログラムはメモ帳で作成する。保存するときに、最初でも必ず、名前をつけて保存を選ぶ(そうしないと、拡張子 txt が付いたファイルになってしまう)
7. programname.f としてファイルを保存

Gnuplot

1. Sorceforge (http://sourceforge.net/projects/gnuplot/files/gnuplot/4.6.0/)から gp460win32.zip をダウンロード
2. ZIP ファイルを解凍して、インストーラを起動
3. コマンドプロンプトに gnuplot と入力して gnuplot が動くか確認

Mac の場合

Fortran

1. App Store から Xcode をインストール
2. Xcode を起動して、Preference, Downloads から Command Line Tools をインストール(これで、gcc がインストールされる。
3. G95 プロジェクトのホームページから Downloads, Binaries をクリック
4. x86 OS X の HTTP をクリックしてダウンロード
5. g95-x86-osx.tgz アイコンをダブルクリックして、解凍
6. Macintosh HD の直下に opt というフォルダを作る
7. g95-install フォルダを opt フォルダに移動

8. アプリケーション、ユーティリティフォルダにある X11(あるいは XQuartz)を起動
9. ターミナルが開いたら、以下のコマンドを打つ
   cd /opt/g95-install/bin
   ln –s i686-apple-darwin10.3.0-g95 /usr/bin/g95
10. g95 と入力して動くか確認する

Gnuplot

1. このページ(http://www.muskmelon.jp/?page_id=2)にある、gnuplot のページを開く
2. gnuplot 4.7 のどちらか(OS による)をクリックしてダウンロード
3. アプリケーション、ユーティリティフォルダにある X11(あるいは XQuartz)を起動
4. ターミナルが開いたら、以下のコマンドを打つ
   cd　/Applications/gnuplot.app/bin/
   ln –s gnuplot /usr/bin/