Fortran プログラミング 応用編

１．テイラー展開

テイラー展開は、連続関数をある点の周りで、増分 δx の冪級数として表すことである。すなわち、

$$f\left(x_0+\delta x\right)=\sum_{n=0}^{\infty}a_n\delta x^n$$

$$a_n=\frac{f^{(n)}\left(x_0\right)}{n!}$$

である。このことは、連続関数ならば冪級数として表すことができることを示している。テイラー級数は無限級数であるが、δx があまり大きくない場合、n を十分大きくとれば、元の関数を十分な精度で近似することができる。

正弦関数のテイラー展開による計算と近似の精度

正弦関数

$$f\left(x\right)=\sin x$$

を x=0 の周りでテイラー(マクローリン)展開すると、

$$\sin x=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+-...$$

となる。ただし、この式では増分を x で表している。この式を計算するプログラムを問題 16 プログラムに示す。プログラムでは級数の次数は有限であることに注意すべし。

問題 16

(1) プログラムのなかで、

```
ak = (-mod(k,2))**(k/2) / fact_k
```

という行があるが、これはどのように動作するのか手計算により確認せよ。

(2) プログラムを入力して実行し、テイラー級数の次数が大きくなると近似が良くなることを確認せよ。

(3) 正弦関数の 1 周期分が精度良く計算できる(10^{-6} 程度まで)のに必要な次数を求めよ。

２．数値積分

数値積分とは数値的な計算により、定積分

$$I = \int_a^b f(x)dx$$

の値を求める方法である。基本的に、下記のような考えで積分する。

・定積分の区間を細かく分割する。

・細かく分割したそれぞれの領域で関数を積分しやすい関数(直線、２次曲線、平面など)で補間する。

・補間した関数を積分して分割領域の面積(体積)を求める。

・分割領域の値を足し上げて定積分の値を求める。

補間関数の取り方によっていろいろな方法が考えられる。

2.1 長方形法

区間[a,b]を N 等分し、それぞれの区間の関数値を端のどちらか 1 つの点の値で関数の値を近似する。 1 つの区間の面積は

$$\Delta S = f(x)h$$

で表される。関数値を左側に取る場合、積分値は次のように表される。

$$I = \left[f(a) + f(a+h) + ... + f(a+ih) + ... + f(b-h) \right] h$$

式をまとめると、

$$I = h\sum_{i=1}^{N-1} f(a+ih)$$

である。この方法は 1 次の打ち切り誤差を持つ。

2.2 台形法

区間[a,b]を N 等分し、近接する２点間を結ぶ直線で関数を近似する。つまり、細長い台形とし、小区間の面積を求める。すなわち、その面積は

$$\Delta S = \frac{1}{2}\left[f(x) + f(x+h) \right] h$$

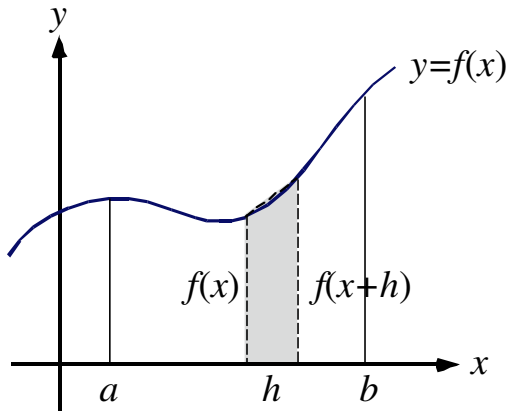
で表される。積分値は次のように表される。

$$I = \frac{1}{2}\left[f(a) + 2f(a+h) + ... + 2f(a+ih) + ... + 2f(b-h) + f(b) \right] h$$

端以外の係数が２になっている。まとめると、

$$I=\frac{1}{2}\left[f(a)+\sum_{i=1}^{N-1}2f(a+ih)+f(b)\right]h$$

である。台形法は２次の打ち切り誤差を持つ。



2.3 シンプソン法

近接するいくつかの区間を１まとめにして、より高次の次数をもつ多項式で関数を近似する方法である。精度は補間する関数の次数で決まる。例えば、区間[a,b]を偶数個 n に等分し、２つまとめにすると、３点を用いて２次多項式を当てはめることができる。このとき、小区間を２つ合わせた区間の面積は、

$$\Delta S=\frac{1}{3}\left[f(x)+4f(x+h)+f(x+2h)\right]h$$

である。このとき、区間[a,b]全体では、

$$I=\frac{1}{3}[f(a)+4f(a+h)+2f(a+2h)\ldots$$
$$+2f(a+2ih)+4f(a+\{2i+1\}h)+\ldots$$
$$+2f(b-2h)+4f(b-h)+f(b)]h$$

となる。係数が、$1\to4\to2\to4\to2\cdots\to2\to4\to1$ となっている。２は小区間を２つにまとめた ΔS のつなぎ目である。まとめると、

$$I=\frac{1}{3}\left[f(a)+\sum_{i=1}^{N/2-1}4f(a+\{2i+1\}h)+\sum_{i=1}^{N/2}2f(a+2ih)+f(b)\right]h$$

となる。２次式で近似する方法は３次の打ち切り誤差を持つ。

2.4 誤差関数の数値積分

正規分布の形状を表す関数*1

$$f(x) = e^{-x^2} = \exp\left(-x^2\right)$$

を 0 から x まで積分したものに係数 $2/\sqrt{\pi}$ を掛けた関数

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-\zeta^2\right) d\zeta$$

を誤差関数と呼ぶ。この積分は $x=\infty$ の時を除き、解析的に解くことが出来ない。このため、数値計算によりこの積分を解いてみよう。なお、係数 $2/\sqrt{\pi}$ は $x=\infty$ としたときに誤差関数の値が 1 になるように決めている。

誤差関数を台形法あるいはシンプソン法により計算するプログラムを別ページに示す。このプログラムでは、新しい Fortran の文法は使っていない。

*1 平均 μ、標準偏差 σ の正規分布は次のように表される。

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

この式は、確率を表すので、-∞から+∞まで積分すると 1 となるように係数を決めている。

問題 17

(1) 台形法による誤差関数計算のプログラムを入力し、コンパイルせよ。

(2) x_2 の値を変化させながら実行し、誤差関数のグラフを作成せよ。

問題 16 プログラム　テイラー級数による正弦関数

プログラムは２つのプログラム単位からなる。それは、主プログラムと関数副プログラムである。プログラム単位ごとに１つのファイルに保存する。コンパイルするときには、

```
f95 taylor.f90 t_sine.f90
```

のようにプログラムのファイルを羅列する。

主プログラム taylor.f90

```
!
!   ******   Calculate sine by Talor-series expansion   ******
!
      program taylor

      implicit none
      integer i, n, ns
      real(8):: dx, xmax, pi
      real(8), allocatable:: x(:),s1(:),s2(:)
      real(8):: T_sine
      character(20):: fmt1

      pi = 4.0d0 * atan(1.0d0)
      fmt1 = '(3f12.7)'

      write (6,*) 'Input number of samples for 1 period'
      read  (5,*) ns
      write (6,*) 'Your input for number of samples = ',ns

      dx = 2.0d0 * pi / dfloat(ns)

      allocate ( x(0:ns),s1(0:ns),s2(0:ns) )

      write (6,*) 'Input maximum order of Taylor series'
```

```
      read  (5,*) n
      write (6,*) 'Your input maximum order of Taylor series = ',n

      write (6,*)
      write (6,*) '       x        sin(x)    T_sine(x)'

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

      do i = 0,ns
        x(i)  = dx * dble(i)
        s1(i) = dsin(x(i))
        s2(i) = T_sine(x(i),n)
        write ( 6,fmt1) x(i), s1(i), s2(i)
        write (10,fmt1) x(i), s1(i), s2(i)
      end do

      stop
      end program taylor
```

関数副プログラム　t_sine.f90

```
!
!   ******   function T_sine   ****************************************
!
     function T_sine(x,n)

      implicit none
      integer:: k, n
      real(8):: pi
      real(8):: x, xp, fact_k, ak, fk
      real(8):: T_sine_k
      real(8):: T_sine
      real(8),parameter:: eps = 1.0d-10
```

```
      pi = 4.0d0 * atan(1.0d0)
!
!   initalize: f(x=0) = sin(0), 0! = 1, x**0 = 1.0
!
      T_sine_k = 0.0d0
      fact_k = 1.0d0
      xp = 1.0d0
!
!   take summation sigma_( a(i)*x**i ) for i = 1 to n
!
      do k = 1,n
        fact_k = fact_k * dfloat(k)  ! calculate factorial real(k!)
        ak = (-mod(k,2))**(k/2) / fact_k  ! coefficient of the series

        xp = xp * x     ! calculate xp = x**k
        fk = ak * xp    ! k-th order term

        T_sine_k = T_sine_k + fk  ! calculate summation
      end do

      T_sine = T_sine_k

      return
      end function T_sine
```

このプログラムでは、sine の値を一周期分、テイラー級数によって計算し、その値を sine の実際の値(Fortran の組み込み関数から計算した値)と比較する。1 周期のサンプル数と、テイラー級数の最高次数 (highest order)を指定する。このプログラムでは、プログラムのうち、テイラー級数を計算する部分を関数副プログラムとしてプログラム単位を２つに分けている。プログラム単位は、主プログラム(メインルーチン)と関数副プログラム(関数サブルーチン)である。

主プログラム(main routine)

`real(8), allocatable:: x(:),s1(:),s2(:)`：配列宣言文。倍精度であることと、動的割り付けを行うことを同時に宣言している。

`allocate ( x(0:ns),s1(0:ns),s2(0:ns) )`：allocate 文。配列変数を割り付けている。(0:ns) というのは、配列の範囲が 0 から ns までであることを表す。Fortran では、“0:”を指定しないと、1 からになる。

`real(8):: T_sine`：ユーザー定義関数 `T_sine` が倍精度型であることを宣言する。

`T_sine s2(i) = T_sine(x(i),n)`：関数副プログラムの呼び出し。( )の中にある変数(またはパラメータ)を引数(arguments)と呼ぶ。( )の中にある変数の値が関数副プログラムに引き渡される。つまり、関数の変数である。その個数や変数の型などが、function 文の中にある引数ときちんと 1 対 1 に対応していなければならない。関数副プログラムで計算した値は、`T_sine` の中に返って来る。

ところで、配列宣言をしていない括弧( )の付いている変数は関数と見なされる。配列変数のつもりで配列宣言をし忘れた場合には、関数の定義がない(関数サブルーチンがない)というコンパイル時エラーが出るので、戸惑わないように。

`write (10,500) x(i), s1(i), s2(i)`：出力文。フォーマット指定付きである。3 カラムで出力される。gnuplot でグラフを書くときに注意が必要である。

関数副プログラム(function subroutine)

`real*8 function T_sine(x,n)`：function 文。ここから、end 文までが、関数副プログラムであることを宣言する。同時に関数が、倍精度実数であることを宣言している。( )中は引数である。関数を呼び出したプログラムでの値が引き渡される。変数の名前は必ずしも、呼び出し側と同じでなくて良い。ただし、変数の型や個数が合っていなければならない (例外もあるが)。ここでは、メインルーチン側は倍精度(8 bytes = 64 bits)の配列変数 1 つと、単精度整数 (4 bytes = 32 bits) 1 つである。関数副プログラム側は、倍精度の配列変数でない変数 1 つと単精度整数 1 つである。配列変数をまとめて渡す時には、( )を付けずに名前だけで渡す。このとき、メイン側とサブ側で配列の個数が同じでないとエラーが起きる。

`T_sine = T_sine_k`：代入文。T_sine_k の値を関数 T_sine へ入れている。この式がないと関数に値を入らない。

`return`：return 文。呼び出したプログラム単位へ処理が返される。

図の書き方

３カラム以上あるファイルの場合は using オプションで x 軸、 y 軸がどのカラムであるのか指定する。

```
plot 'mysine.dat' using 1:2
```

１カラム目を x 軸、２カラム目を y 軸にしてプロットする。

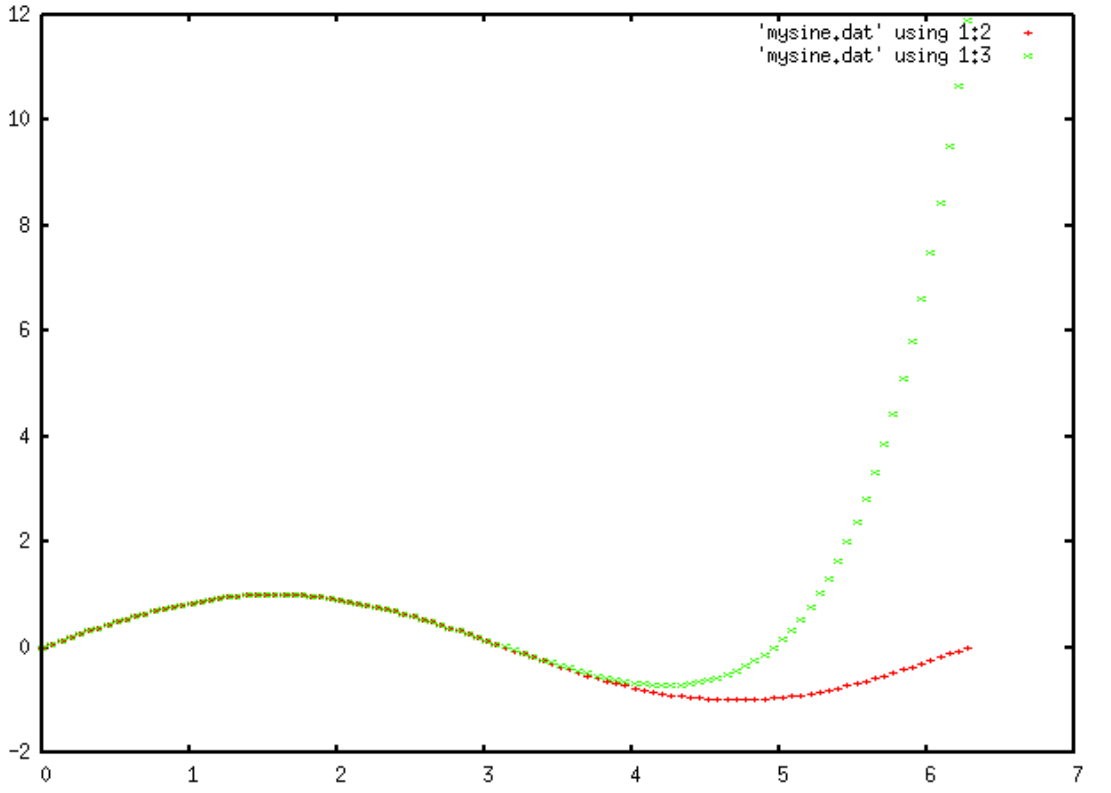
```
plot 'mysine.dat' using 1:2, 'mysine.dat' using 1:3
```

１カラム目を x 軸、２カラム目と３カラム目の２つを y 軸にして１つのグラフにプロットする。

```
plot 'mysine.dat' using 1:2
replot 'mysine.dat' using 1:3
```

replot コマンドを使うとグラフを上書きできる。

図：９次まで計算した結果



問題 17 プログラム　台形法による誤差関数の計算

```
!
!  **  Error function  **
!  Numerical integration by trapezoid method
!
      program trapezoid_i
      implicit none
      integer, parameter:: n = 1000
      integer:: i
      real(8)::  f(0:n)
      real(8)::  x,x1,x2,dx
      real(8)::  s1,sall
      real(8)::  erf
      real(8)::  pi,rootpi

!      write (6,*) 'x1 ='
!      read  (5,*) x1
      x1 = 0.0d0
      write (6,*) 'x ='
      read  (5,*) x2

      dx = ( x2-x1 ) / dfloat( n )

      do i = 0,n
        x = x1 + dx * dfloat(i)
        f(i) = exp( - x**2 )
      end do

      s1 = 0.0d0
      do i = 1,n-1
        s1 = s1 + 2.0d0*f(i)
      end do
```

```
sall = dx * 0.5d0  * ( f(0) + s1 + f(n) )

pi = 4.0d0 * atan( 1.0d0 )
rootpi = sqrt( pi )
erf = 2.0d0 / rootpi * sall

write (6,*) 'Integral F = ',sall
write (6,*) 'erf(',x2,')= ',erf

stop
end program trapezoid_i
```

台形法により誤差関数を計算するプログラム errfun.f90 である。また、errfun_2.f は自動的に引数 x を変化させて誤差関数の数値表を作るプログラムである。

`integer, parameter:: n = 1000`：n を整数パラメータとして宣言。積分区間の分割数を決めている。

`real(8):: f(0:n)`：固定配列宣言。f は被積分関数である。

`f(i) = exp( - x**2 )`：代入文。被積分関数の値を計算している。右辺を書き換えると様々な関数の定積分を計算できる。積分値は変数 `sall` に代入される。

`s1 = s1 + 2.0d0*f(i)`：代入文。両端を除く、関数の値の総和

$$s_1 = \sum_{i=1}^{n-1} f\left(x_i\right)$$

である。右辺の計算が終わった後、`s1` に上書きする。総和を計算するときの常套文である。