II. 数値計算の基礎

4．級数による関数の表現

関数を数値的に計算する方法として級数に展開する方法がある。ここで紹介するテイラー級数のほか, 周期関数に対するフーリエ級数や球面上の関数を表す球関数が地球物理学ではしばしば用いられる。

4.1 テイラー級数展開

テイラー級数展開 (Taylor Series expansion)とは，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,

$$f\left(x_0+\delta x\right)=\sum_{n=0}^{\infty}a_n\delta x^n \tag{4.1}$$

$$a_n=\frac{f^{(n)}\left(x_0\right)}{n!} \tag{4.2}$$

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。

4.2 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

$$f\left(x\right)=\sin x \tag{4.3}$$

を $x=0$ の周りでテイラー(マクローリン)展開すると,

$$\sin x=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+-... \tag{4.4}$$

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 II-1 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 II-1

(1) テイラー展開を使う意味は何か。

(2) プログラムのなかで,

(3) プログラムを入力し，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。

(4) 正弦関数の２周期目の値まで計算できるようにプログラムを改良せよ。２周期目が近似としてよく表されるには，何次まで級数を計算する必要があるか。

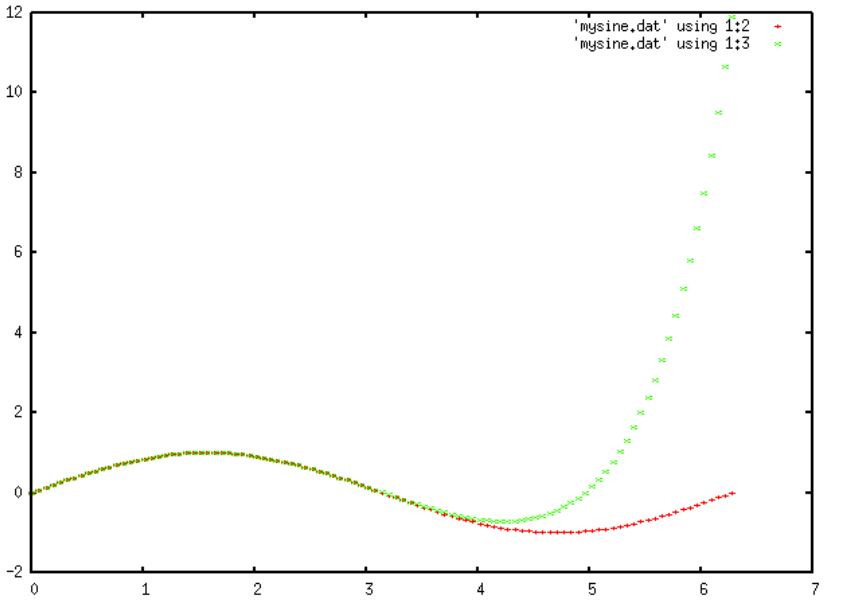

図 4.1 ９次まで計算した結果

4.2 フーリエ級数展開

フーリエ級数 (Fourier Series)は周期関数

$$f\left(t+T\right)=f\left(t\right) \tag{4.11}$$

を三角関数の無限級数に展開した級数である。周期(T, $T/2$, $T/3$, ...)をもつ sin, cos の重ね合わせにより表す。すなわち，

$$f\left(t\right)=\frac{a_0}{2}+a_1\cos\frac{2\pi t}{T}+a_2\cos\frac{4\pi t}{T}+a_3\cos\frac{6\pi t}{T}+\ldots$$

$$+b_1\sin\frac{2\pi t}{T}+b_2\sin\frac{4\pi t}{T}+b_3\sin\frac{6\pi t}{T}+\ldots \tag{4.12}$$

$$f\left(t\right)=\frac{a_0}{2}+\sum_{n=1}^{\infty}\left(a_n\cos\frac{2\pi nt}{T}+b_n\sin\frac{2\pi nt}{T}\right) \tag{4.13}$$

である。　$f(t)$が偶関数の場合，cos だけの級数で表され，余弦級数と呼ばれる。

また，奇関数の場合，sin だけの級数で表され，正弦級数と呼ばれる。

フーリエ級数の係数はフーリエ係数と呼ばれ，積分

$$a_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cos\frac{2\pi nt}{T}dt \tag{4.14}$$

$$b_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\sin\frac{2\pi nx}{T}dt \tag{4.15}$$

で表される。

4.2 鋸波のフーリエ級数

ここでは，次のように周期 $2L$ をもつ，ノコギリの刃状の関数を考える。

$$f(x) = \frac{L-x}{L} \tag{4.16}$$

ただし，

$$0 \le x < 2L \tag{4.17}$$

ある。この関数は奇関数であるので，フーリエ級数は正弦級数

$$f(x) = \sum_{n=1}^{\infty} b_n \sin\frac{\pi nx}{L} \tag{4.18}$$

となる。さらに，(4.14)より，

$$b_n = \frac{1}{L}\int_0^{2L} \frac{L-x}{L}\sin\frac{\pi nx}{L}dx = \frac{2}{n\pi} \tag{4.19}$$

である。したがって，

$$f(x) = \sum_{n=1}^{\infty} \frac{2}{n\pi}\sin\frac{\pi nx}{L} \tag{4.20}$$

である。このフーリエ級数は両端の温度を固定した 1 次元熱伝導問題に応用できる。

問題 II-2

(1) 問題 II-2 プログラムを実行し，フーリエ級数が鋸波と一致することを確かめよ。

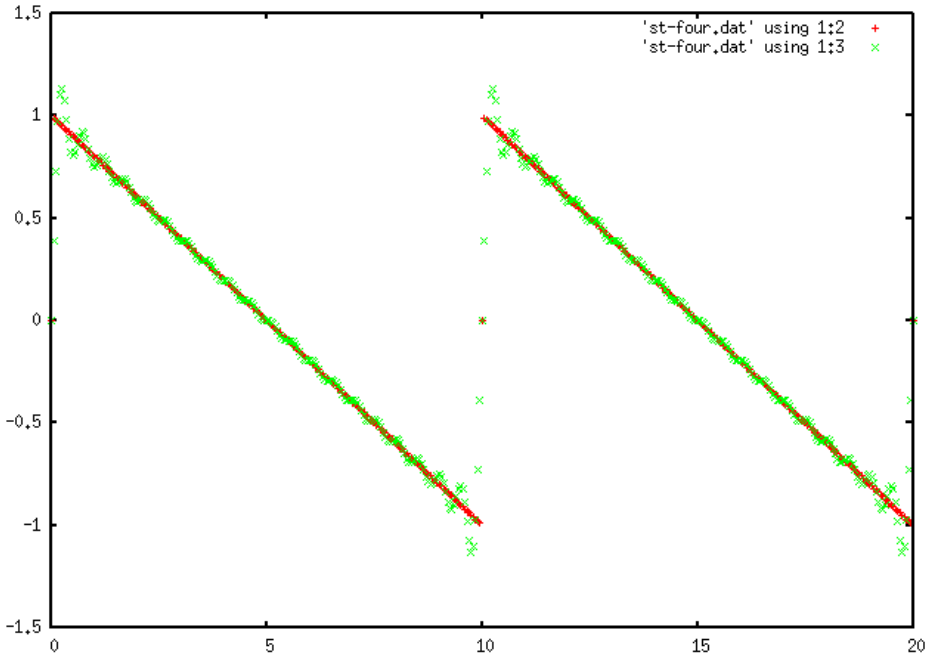

図 4.2 フーリエ級数による鋸波の再現。$L = 5, n = 20$ で 2 周期分計算

5．数値積分

数値積分とは数値的な計算により，定積分

$$I = \int_a^b f(x)dx \tag{5.1}$$

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。

5.1 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

$$\Delta S = f(x)h \tag{5.2}$$

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。

$$I = \left[f(a) + f(a+h) + ... + f(a+ih) + ... + f(b-h) \right] h \tag{5.3}$$

式をまとめると，

$$I = h \sum_{i=1}^{N-1} f(a+ih) \tag{5.4}$$

である。この方法は，**長方形法** (rectangular method)とよばれ，1次の打ち切り誤差を持つ。

5.2 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

$$\Delta S = \frac{1}{2} \left[f(x) + f(x+h) \right] h \tag{5.5}$$

で表される。積分値は次のように表される。

$$I = \frac{1}{2} \left[f(a) + 2f(a+h) + ... + 2f(a+ih) + ... + 2f(b-h) + f(b) \right] h \tag{5.6}$$

端以外の係数が2になっている。まとめると，

$$I = \frac{1}{2} \left[f(a) + \sum_{i=1}^{N-1} 2f(a+ih) + f(b) \right] h \tag{5.7}$$

である。この方法は**台形法** (trapezoid method)とよばれ，2次の打ち切り誤差を持つ。

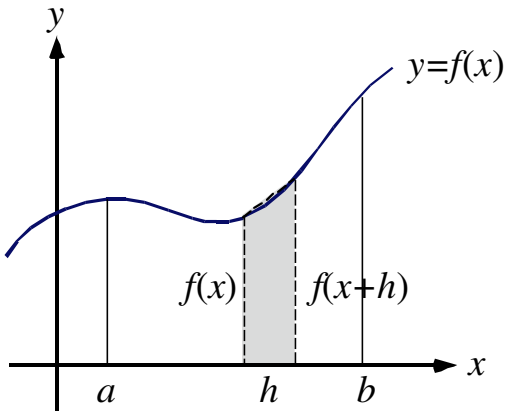

図 5.1 台形法

5.3 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を**シンプソン法** (Simpson method)という。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，

$$\Delta S = \frac{1}{3}\left[f(x) + 4f(x+h) + f(x+2h) \right] h \tag{5.8}$$

である。このとき，区間$[a, b]$全体では，

$$I = \frac{1}{3}[f(a) + 4f(a+h) + 2f(a+2h) \ldots$$

$$+ 2f(a+2ih) + 4f(a+\{2i+1\}h) + \ldots$$

$$+ 2f(b-2h) + 4f(b-h) + f(b)]h \tag{5.9}$$

となる。係数が，$1 \to 4 \to 2 \to 4 \to 2 \cdots \to 2 \to 4 \to 1$ 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，

$$I = \frac{1}{3}\left[f(a) + \sum_{i=1}^{N/2-1} 4f(a+\{2i+1\}h) + \sum_{i=1}^{N/2} 2f(a+2ih) + f(b) \right] h \tag{5.10}$$

となる。２次式で近似する方法は４次の打ち切り誤差を持つ。

5.4 誤差関数の数値積分

正規分布の形状を表す関数*1

$$f(x) = e^{-x^2} = \exp\left(-x^2\right) \tag{5.11}$$

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，すなわち，

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\left(-\zeta^2\right) d\zeta \tag{5.12}$$

を**誤差関数** (error function)とよぶ。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。

*1 平均μ，標準偏差σの正規分布は次のように表される。

$$f(x)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{5.13}$$

この式は，確率密度を表すので，-∞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 II-3

(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。

６．非線型方程式の反復解法

非線型方程式の解は直接求めることが出来ない。また，元数の大きい連立一次方程式を直接法で解くと時間がかかりすぎることがある。このような場合には，計算を繰り返して解を十分精度の良いものに近づけていく方法が用いられる。このような方法は**反復法** (iterative method)と呼ばれる。反復法は連立方程式 (この場合，解がベクトルになる)であっても使うことができる。ここでは，２つの代表的な方法を採り上げる。

6.1 線型反復法

方程式

$$f(x)=0 \tag{6.1}$$

を変形して

$$x=g(x) \tag{6.1}$$

のように表す。 xを反復計算により求めることにする。つまり，

$$x_{n+1}=g(x_n) \tag{6.3}$$

である。ここで, n は繰り返しの回数を表す。すなわち，解は次のような方法で求める。

(1) 適当な x_0 を仮定する。

(2) $g(x)$に代入して， $x_1=g(x_0)$ を求める。

(3) (2)を繰り返す。

(4) 必要な精度で $x_{n+1}=x_n$ となったら繰り返しをやめて， $x=x_{n+1}$ を解とする。

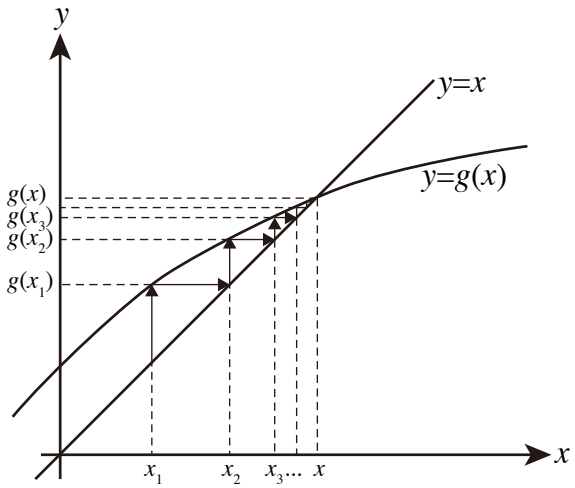

図 6.1 線型反復法

6.2 ニュートン法

$$\varphi(x) \neq 0 \tag{6.4}$$

のとき,

$$g(x) = x - \varphi(x) f(x) \tag{6.5}$$

とする。このとき，(6.1)の解と,

$$x = g(x) \tag{6.6}$$

の解は同じ解 x を持つ。ここで,

$$\varphi(x) = \frac{1}{f'(x)} \tag{6.7}$$

を選ぶ，すなわち,

$$g(x) = x - \frac{f(x)}{f'(x)} \tag{6.8}$$

である。ここで， $x = g(x)$ に反復法を適用する。つまり,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{6.9}$$

のように繰り返し計算を行う。この方法を**ニュートン法** (Newton method)という。ニュートン法は線型反復法よりも少ない回数で収束する。

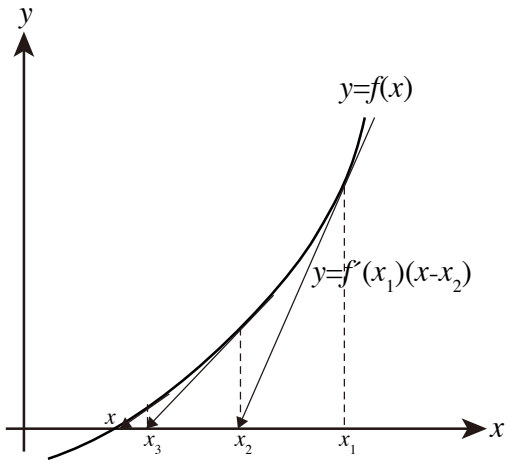

図 6.2 ニュートン法

問題 II-4

(1) ニュートン法による２次方程式求解のプログラムを入力し，実行せよ。

(2) 惑星の楕円軌道上の時刻 t での位置は，ケプラー方程式

$$f(x) = x - e\sin x - \frac{2\pi t}{T} = 0 \tag{6.10}$$

により求めることができる。ここで，e は軌道離心率であり，x は離心近点角，t は近日点(惑星などが太陽に一番近づく点)からの時間，T は軌道周期(地球の場合は 1 恒星年)である。t を与えたときに，x をニュートン法で求めるプログラムを作成せよ。

(3) (2)で作成したプログラムを用いて現在のハレー彗星の離心近点角 x を求めよ。

補足

軌道離心率

$$e = \sqrt{\frac{a^2 - b^2}{a^2}} \tag{6.11}$$

ここで，a:長半径，b:短半径である。

離心近点角

軌道楕円の外側に接する円を考える。惑星の位置 E を長径に垂直な線で円に投影したとき位置を E'とする。円の中心周りで，E'の近日点から角度が離心近点角 x である。

ケプラー方程式

ケプラー方程式は面積速度一定の法則から得られる。円の場合は離心率 e が 0 であるので，x は時刻に t に比例する式になる。

問題 II-1　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.f t_sine.f
```

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
      real( 8 ):: T_sine
      real( 8 ):: dx
      real( 8 ), allocatable:: x( : ), s1( : ), s2( : )
      real( 8 ), parameter:: pi = 4.0d0 * atan( 1.0d0 )
      integer:: Nmax, ms, mx
      integer:: i
      character( 20 ):: fmt1

!  ****  Input parameters  ****

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

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

```
!  ****  Total number of x-points  ****

      mx = ms * 1
      allocate ( x( 0:mx ),s1( 0:mx ),s2( 0:mx ) )

!  ****  Interval of x and maximum number of x points ****

      dx = 2.0d0 * pi / dfloat( ms )

!  ****  For output  ****

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

      open ( 10, file = 'mysine.dat' )
      fmt1 = '(3f12.7)'

      do i = 0, mx
        x( i )  = dx * dble( i )
        s1( i ) = dsin( x( i ) )
        s2( i ) = T_sine( x( i ), Nmax )
      end do

!  ****  Output results  ****

      do i = 0, ms
        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, Nmax )

      implicit none
      real( 8 ):: pi
      real( 8 ):: x, xp, fn
      real( 8 ):: Tser_n
      real( 8 ):: T_sine
      real( 8 ), allocatable:: fac_n( : ), a( : )
      integer:: Nmax, kmax
      integer:: n,    k
      integer:: sig

      allocate( fac_n( 0:Nmax ), a( 1:Nmax ) )

!  ****  kmax: max. of aux. index k  ( n = 2 * k - 1 )  ****

      kmax = ( Nmax + 1 ) / 2

!   ****  calculate factrial n  ****

      fac_n( 0 ) = 1.0d0

      do n = 1, Nmax
        fac_n( n ) = fac_n( n - 1 ) * dble( n )
      end do

!  ****  calculate taylor coefficeints: a( n ) ****
```

```
      do k = 1, kmax
        n = 2 * k - 1
        sig = ( -1 )**( k - 1 )
        a( n ) = sig / fac_n( n )
      end do

!  ****  Calculate Taylor series: summation of an * x^n  ****

      Tser_n = 0.0d0

      do n = 1, Nmax, 2
        Tser_n = Tser_n + a( n ) * x**n
      end do

      T_sine = Tser_n

      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。

問題 II-2　プログラム　フーリエ級数による鋸波関数

プログラムは１つの主プログラムのみからなっている。

主プログラム four-saw.f90

```
!
!  *******  Fourier series of saw-tooth function  **********
!
   program f_saw

      implicit none

      real( 8 ), allocatable:: fs( : ), ff( : )
      real( 8 ), allocatable:: b( : )

      real( 8 ):: L
      real( 8 ):: dx,  x,   xp
      integer::   np,  m,   Mp,  Nmax
      integer::   j,   n
      integer::   k,   jp

      real( 8 ), parameter:: pi = 4.0d0 * atan( 1.0d0 )

!   ****  input parameters  ****

      write ( 6, * ) 'Number of period'
      read  ( 5, * ) np
      write ( 6, * ) 'Sampling number for one period'
      read  ( 5, * ) m
      write ( 6, * ) 'Half length of period'
      read  ( 5, * ) L
      write ( 6, * ) 'Order of Fourier Series'
```

```
      read  ( 5, * ) Nmax

!  ****  Total number of x-points  ****

      Mp = np * m

!  ****  Interval of x-point  ****

      dx = 2.0d0 * L / dble( m )

!  ****  Allocate dimension variables  ****

      allocate ( fs( 0: Mp ), ff( 0: Mp ) )
      allocate ( b( 1: Nmax ) )

!  ****  Calculate Function  ****

      do n = 1, Nmax
        b( n ) = 2.0d0 / ( dble( n ) * pi )
      end do

!  ****  Saw-tooth function  ****

      fs( 0 ) = 0.0d0
      do k = 1, np
        do j = 1, m - 1
          jp = m * ( k - 1 ) + j
          xp = dx * dble( j )
          fs( jp ) = ( L - xp ) / L
        end do
        jp = m * ( k - 1 ) + m
        fs( jp ) = 0.0d0
      end do
```

```
!  ****  Fourier Series  *****

      do j = 0, Mp
        x = dx * dble( j )
        ff( j ) = 0.0d0
        do n = 1, Nmax
          ff( j ) = ff( j ) + b( n ) * sin( dble( n ) * pi * x / L )
        end do
      end do

!  ****  Open output file  ****

      open ( 10, file='st-four.dat' )

!  ****  Write result to file  ****

      do j = 0, Mp
        x = dx * dble( j )
        write ( 10, * ) x, fs( j ), ff( j )
      end do

      close ( 10 )

      stop
   end program f_saw
```

フーリエ級数によって鋸波関数を計算するプログラムである。フーリエ級数による近侍値と直接計算による関数値とを比較する。

`b( n ) = 2.0d0 /…`：フーリエ係数を計算

`do k = 1, np`：鋸波関数を直接計算するための do 文。

`fs( 0 ) = 0.0d0`：関数が不連続になる点には 0 を代入している。

`do n = 1, Nmax`：次数で総和をとるための do 文。

`ff( j ) = ff( j ) +`…：総和をとり，x_j でのフーリエ級数の値を計算。

問題 II-3　プログラム　台形法

```
!
!  **  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
```

台形法により数値積分を計算す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。

問題 II-4　プログラム　ニュートン法

```
!
!    Solve 2nd-order algebraic equation ax^2+bx+c=0 by a Newton method
!
      program newton
      implicit none
      real(4), parameter:: eps=1.0e-6
      real(4):: a,b,c,x,fx,dfdx,xini,corr
      integer:: itr,nitr

      write (6,*) 'input a,b,c (use space between numbers)'
      read  (5,*) a,b,c
      write (6,*) 'input inital guess'
      read  (5,*) xini
      write (6,*) 'input maximum iteration'
      read  (5,*) nitr

      x =  xini
!
!   Iteration
!
      do itr = 1,nitr
        fx  = a*x**2  + b*x + c
        dfdx = 2.0 * a * x + b
        corr = fx / dfdx
        write (6,*) itr,x,fx,dfdx,corr
        x = x - corr
        If ( abs(corr/x) .lt. eps ) Exit  ! Exit loop if convergent
     end do

     write (6,*) 'solution x = ',x
     stop
```

```
end program newton
```

ニュートン法により２次方程式の解を求めるプログラムである。２つの解のうち，極値よりも初期値に近い方の解１つだけが求められる。このプログラムでは，配列変数は使用されていない。

`dfdx = 2.0 * a * x + b`：代入文。２次関数の微分係数値を計算して，実数変数 `dfdx` に代入している。変数名に割り算の記号/は使えないのに注意。
`x = x - corr`：代入文。式(2)の計算である。x に新たに計算した値を上書きで代入している。
`If ( abs(corr/x) .lt. eps ) Exit`：条件判定(if)文。解の修正値 `corr` が絶対値で `x` と比べて `eps` より小さいとき，do ループを脱出し end do の下の行に飛ぶ。`abs(v)`は変数 `v` の絶対値$|v|$を求める関数である。`eps` は解の必要精度である。