神戸市立工業高等専門学校
電気工学科／電子工学科
専門科目「数値解析」

2017.7.14

演習3

山浦 剛（tyamaura@riken.jp）

講義資料ページ

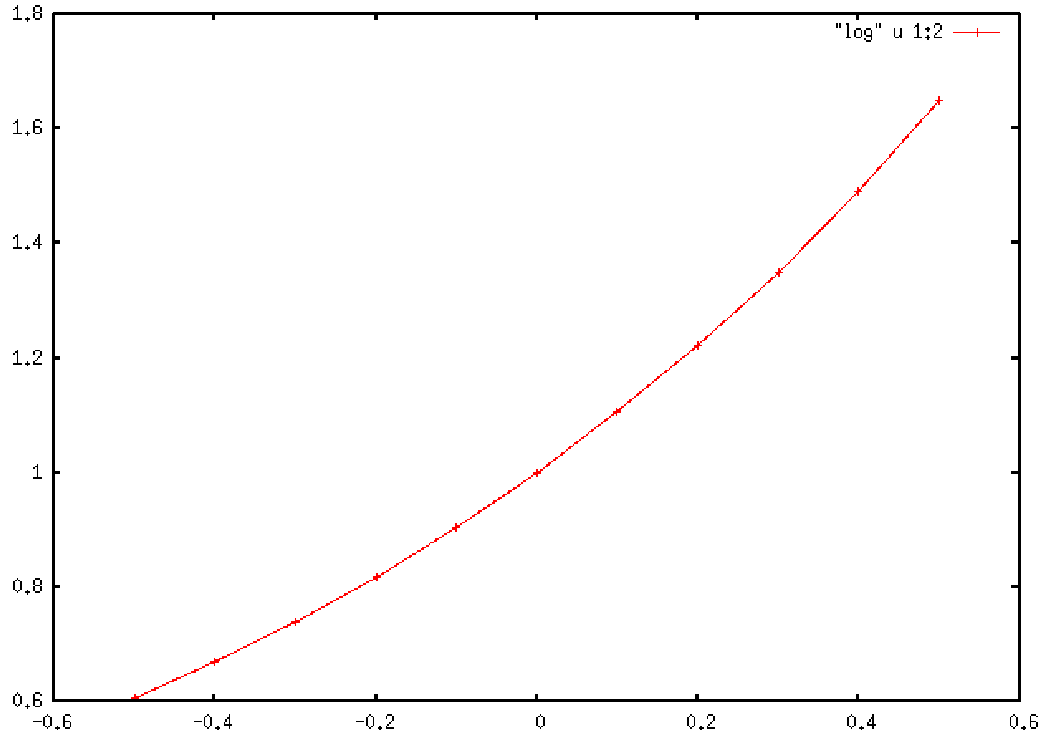
- http://climate.aics.riken.jp/members/yamaura/numerical_analysis.html

復習： テイラー展開

- テイラーの定理: 関数$f(x)$ が$a \leq x \leq b$ においててn 階微分可能であるとき、次式を満たす$c\ (a < c < b)$ が存在する
  - $f(b) = \sum_{k=0}^{n-1} f^{(k)}(a) \frac{(b-a)^k}{k!} + f^{(n)}(c) \frac{(b-a)^n}{n!}$
- テイラーの定理をもとに、a を定数、$b = x$ を変数とする
  - $f(x) = \sum_{k=0}^{n-1} f^{(k)}(a) \frac{(x-a)^k}{k!} + f^{(n)}(c) \frac{(x-a)^n}{n!}$
  - 第2項は剰余項
- $a = 0$ のときはマクローリン展開と呼ばれる
  - $f(x) = \sum_{k=0}^{n-1} f^{(k)}(0) \frac{x^k}{k!} + f^{(n)}(c) \frac{x^n}{n!}$

（サンプル３－１） テイラー展開

```
program taylor_expansion
 implicit none

 integer :: i, n
 real(8) :: a
 real(8) :: x(11)
 real(8) :: f(11)

 a = 0.0d0

 x(:) =  (/ ( -0.5d0 + i * 0.1d0, i=0,10 ) /)
 f(:) = 0.0d0

 do i = 1, 11
  do n = 0, 4
   f(i) = f(i) + df(n,a) * ( x(i) - a )**n / factorial(n)
  end do

  write(*,*) x(i), f(i)
 end do
end program
```



復習： Fortran 副プログラム

- Fortranでは、手続きをひとまとめにした副プログラムを使うことで、何度も同じような手続きを書かなくても済むようにコードを書くことができる
- 副プログラムにはサブルーチン形式とファンクション形式がある
  - サブルーチンはcallで呼び出される
  - ファンクションは組込関数のように式中に組み込むことができる
- それぞれ一長一短があるので、目的に応じて使い分けるとよい

```
program testprogram
  implicit none
  ! --- 宣言部 ---
  ! --- 処理部 ---
  call f( a, b )
  call f( c, d )
  stop
contains
  subroutine f( x, y )
    implicit none
    ! --- 宣言部
    ! --- 処理部
  end subroutine
end program
```

復習：　グラフの描画

- GNUplotを用いる
  - $ gnuplot
  - > plot ”output.csv” u 1:2 w lp
- gnuplotコマンドの内訳
  - “”でくくったファイル名が入力ファイル
  - そのファイルの中身は空白で区切られた列とみなし、1列目を横軸、2列目をy軸とみなす
  - 線（line: l）と点（point: p）で結んだ折れ線グラフで表示する
- 他に、output.csv の中身をOfficeファイルにコピペしてグラフを描かせてみてもよい

復習：　離散/逆離散フーリエ変換

- 関数$f(x)$上のN個の点$(x_0, f(x_0))$ $(x_1, f(x_1))$…$(x_{N-1}, f(x_{N-1}))$が与えられる。周期LをΔx間隔でN等分し（$L = N\Delta x$）、m番目のxの位置$x_m = m\Delta x = \frac{mL}{N}$とおく
- 離散フーリエ変換（DFT）
  - $F_n = \frac{1}{N}\sum_{m=0}^{N-1} f(m\Delta x)\exp\left(-\frac{2\pi imn}{N}\right)$ 　　　　$(n = -N/2, \ldots, -2, -1, 0, 1, 2, \ldots, N/2)$
- 逆離散フーリエ変換（IDFT）
  - $f(x) = \sum_{n=-N/2}^{N/2} F_n \exp\left(\frac{2\pi inx}{L}\right)$

（サンプル３－２）　離散フーリエ変換

```
program discrete_Fourier_transform
  implicit none

  integer, parameter :: num = 4
  real(8), parameter :: pi = 3.1415926535897930d0
  complex(8), parameter :: i = ( 0.0d0, 1.0d0 )

  integer :: n, m
  real(8) :: f( 0:num-1 )
  complex(8) :: DFT( -num/2:num/2 )

  f( 0:num-1 ) = (/ 0.0d0, 0.5d0, 1.0d0, 0.5d0 /)
  DFT( -num/2:num/2 ) = 0.0d0

  do n = -num/2, num/2
    do m = 0, num-1
      DFT(n) = DFT(n) + f(m) / dble(num) ¥
               * exp( -2.0d0 * pi * i * dble(m*n) / dble(num) )
    end do

    write(*,*) 'F(',n,') = ',real( DFT(n) ),' + i * ',aimag( DFT(n) )
  end do
end program
```

```
F(        -2 ) =   0.0000000000000000       + i *   0.0000000000000000
F(        -1 ) =  -0.25000000000000000      + i *   2.7755575615628914E-017
F(         0 ) =   0.50000000000000000      + i *   0.0000000000000000
F(         1 ) =  -0.25000000000000000      + i *  -2.7755575615628914E-017
F(         2 ) =   0.0000000000000000       + i *   0.0000000000000000
```

復習： Fortran 複素数

- Fortranにおける複素数は、complex宣言を使用する
- 実数部と虚数部の配列のようになっている
  - 単精度・倍精度を決めることができる
- real関数で実数部を読み込め、aimag関数で虚数部を読み込める
- 複素数配列は、虚数単位を作り、a+biの形で代入する必要がある

```
program testprogram
  implicit none

  complex(8) :: x, a(2)

  x = (/ 3.0d0, -2.0d0 /)         ! x = 3 - 2i
  i = (/ 0.0d0, 1.0d0 /)          ! 虚数単位

  write(*,*) real(x)              ! 3
  write(*,*) aimag(x)             ! -2

  a(1) = 3.0d0 + 5.0d0 * i        ! 3+5i
  a(2) = -2.0d0 – 7.0d0 * i       ! -2-7i

  write(*,*) real(a(1))           ! 3
  write(*,*) aimag(a(2))          ! -7

end program
```

課題

1. サンプルプログラム3-1を参考に、$f(x) = \log x$ を$a = 1$ の周りでテイラー展開して、$n = 5$ まで近似せよ。また、$x = 0.5 \sim 1.5$ の範囲で等間隔に11点（$x = 0.5, 0.6, \ldots, 1.4, 1.5$）選び、対応する$f(x)$ の値を書き出せ。剰余項は小さいものとして無視してよい。
2. サンプルプログラム3-1を参考に、$f(x) = \sqrt{x+1}$ をマクローリン展開して、$n = 5$ まで近似せよ。また、$x = \pm 1$ の範囲で等間隔に21点（$x = -1, -0.9, \ldots, 0.9, 1$）選び、対応する$f(x)$ の値を書き出せ。剰余項は小さいものとして無視してよい。
3. サンプルプログラム3-2を参考に、三角波形$f(x) = x\ (0 \leq x \leq 1), f(x) = 2 - x\ (1 < x \leq 2)$ をもとに$x = 0 \sim 2$ の範囲で$x = 0$ から0.25毎に$N = 8$個の離散フーリエ係数を求めるプログラムを作成せよ。
4. サンプルプログラム3-2を参考に、上記のフーリエ係数を用いて逆離散フーリエ変換を行い、$x = 0 \sim 2$ の範囲で$x = 0$ から0.02毎の101点に対応する$f(x)$ の値を書き出せ。
5. サンプルプログラム3-2を参考に、ノコギリ波形$f(x) = x\ (0 \leq x \leq 1), f(x) = x - 2\ (1 < x \leq 2)$ をもとに$x = 0 \sim 2$ の範囲で$x = 0$ から0.2毎に$N = 10$個の離散フーリエ係数を求めるプログラムを作成せよ。
6. サンプルプログラム3-2を参考に、上記のフーリエ係数を用いて逆離散フーリエ変換を行い、$x = 0 \sim 2$ の範囲で$x = 0$ から0.02毎の101点に対応する$f(x)$ の値を書き出せ。

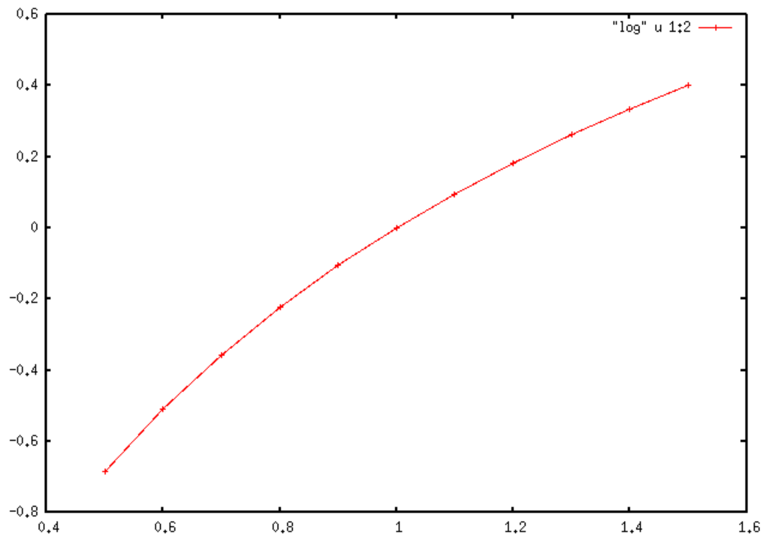
提出方法

- 〆切: 2017/07/28(金)　講義開始前まで
- メールにプログラムを添付
  - 主題: 演習3レポート(学籍番号)
  - 宛先: tyamaura@riken.jp
  - 本文: なくてもOK
  - 添付: 学籍番号_課題番号.f90 を6ファイル
    - 課題3-1: r???????_kadai03-1.f90
    - 課題3-2: r???????_kadai03-2.f90
    - 課題3-3: r???????_kadai03-3.f90
    - 課題3-4: r???????_kadai03-4.f90
    - 課題3-5: r???????_kadai03-5.f90
    - 課題3-6: r???????_kadai03-6.f90

課題1：解答例

- $y = \log x$ の微分係数を考える
- 内挿の始点を考える

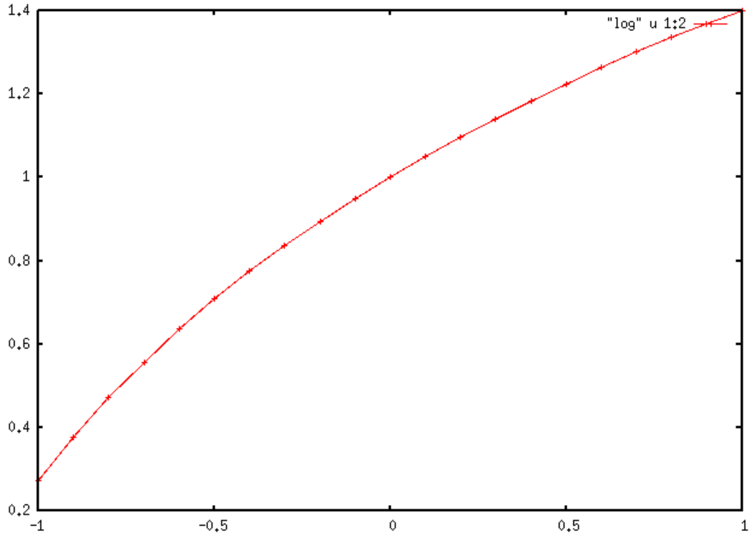
```
program taylor_expansion
  implicit none

  ! -- definition

  integer :: i, j, n
  real(8) :: a
  real(8) :: x(11)
  real(8) :: f(11)

  ! --- main process ---

  a = 1.0d0

  x(:) = (/ (0.5d0+0.1d0*j,j=0,10) /)
  f(:) = 0.0d0

  do i = 1, 11
    do n = 0, 4
      f(i) = f(i) + df(n,a) * ( x(i) - a )**n / factorial(n)
    end do

    write(*,*) x(i), f(i)
  end do

! --- 中略

   tmp(0:4) = (/ log(a), &
                1.0d0/a, &
                -1.0/a**2, &
                2.0d0/a**3, &
                -6.0d0/a**4 /)

    res = tmp(k)

    return
end program
```



課題2：解答例

- $y = \sqrt{x+1}$ の微分係数を考える
- 内挿の始点を考える

```
program taylor_expansion
  implicit none

  ! -- definition

  integer :: i, j, n
  real(8) :: a
  real(8) :: x(21)
  real(8) :: f(21)

  ! --- main process ---

  a = 1.0d0

  x(:) = (/ (-1.0d0+0.1d0*j,j=0,20) /)
  f(:) = 0.0d0

  do i = 1, 11
    do n = 0, 4
      f(i) = f(i) + df(n,a) * ( x(i) - a )**n / factorial(n)
    end do

    write(*,*) x(i), f(i)
  end do

! --- 中略

    tmp(0:4) = (/ (a+1.0d0)**(0.5d0), &
                 0.5d0*(a+1.0d0)**(-0.5d0), &
                 -0.25d0*(a+1.0d0)**(-1.5d0), &
                 0.375d0*(a+1.0d0)**(-2.5d0), &
                 -0.9375d0*(a+1.0d0)**(-3.5d0) /)

    res = tmp(k)

    return
end program
```



課題3：解答例

- 三角波を想定
- 離散フーリエ変換に与える点を増やす
- 元の関数から、$f(x)$ に相当する値を計算し、プログラムに書き込む

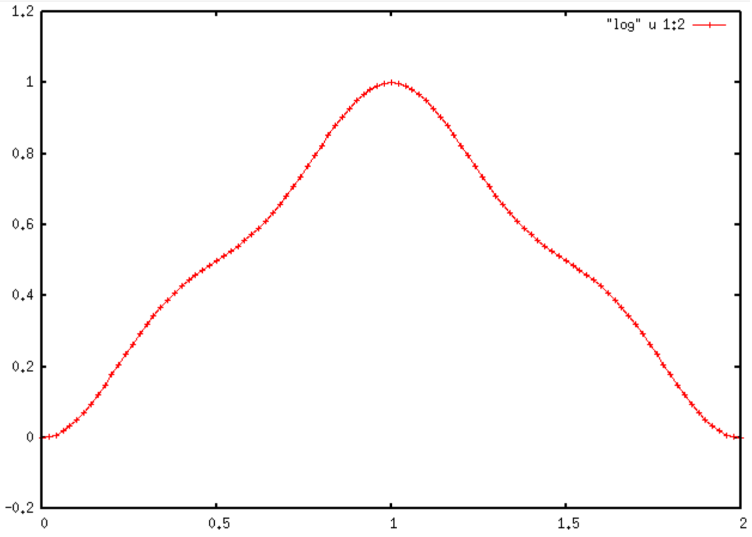
```
program discrete_Fourier_transform
 implicit none

 ! --- definition ---

 integer, parameter :: num = 8
 real(8), parameter :: pi = 3.141592653589793d0
 complex(8), parameter :: i = ( 0.0d0, 1.0d0 )

 integer :: n, m
 real(8) :: f( 0:num-1 )
 complex(8) :: DFT( -num/2:num/2 )

 ! --- main process ---

 f( 0:num-1 ) = (/ 0.00d0, 0.25d0, 0.50d0, 0.75d0, 1.00d0, &
                  0.75d0, 0.50d0, 0.25d0 /)
 DFT( -num/2:num/2 ) = 0.0d0

 do n = -num/2, num/2
  do m = 0, num-1
   DFT(n) = DFT(n) + f(m) / dble(num) &
      * exp( -2.0d0 * pi * i * dble(m*n) / dble(num) )
  end do

  write(*,*) 'F(',n,') = ',real( DFT(n) ),' + i * ',aimag( DFT(n) )
 end do

 stop
end program
```

```
F(        -4 ) =   0.0000000000000000      + i *   1.2325951644078309E-032
F(        -3 ) =  -3.6611652351681609E-002 + i *  -7.6327832942979512E-017
F(        -2 ) =   4.8321344214028144E-019 + i *  -3.4694469519536142E-018
F(        -1 ) = -0.21338834764831846      + i *   2.0816681711721685E-017
F(         0 ) =  0.50000000000000000      + i *   0.0000000000000000
F(         1 ) = -0.21338834764831846      + i *  -2.0816681711721685E-017
F(         2 ) =   4.8321344214028144E-019 + i *   3.4694469519536142E-018
F(         3 ) =  -3.6611652351681609E-002 + i *   7.6327832942979512E-017
F(         4 ) =   0.0000000000000000      + i *  -1.2325951644078309E-032
```

課題4：解答例

- 課題3の離散フーリエ係数を設定し、IDFTを解く
- 三角波のような波形が見える

```
program inverse_discrete_Fourier_transform
 implicit none

 ! --- definition ---

 integer, parameter :: num = 8
 real(8), parameter :: pi = 3.141592653589793d0
 complex(8), parameter :: i = ( 0.0d0, 1.0d0 )

 integer :: j, n, m
 real(8) :: L
 real(8) :: x( 0:100 )
 real(8) :: IDFT( 0:100 )
 complex(8) :: F( -num/2:num/2 )

 ! --- main process ---

 F(-num/2:num/2) = (/  &
  0.0000000000000000     + i *  1.2325951644078309E-032, &
  -3.6611652351681609E-002 + i * -7.6327832942979512E-017, &
  4.8321344214028144E-019 + i * -3.4694469519536142E-018, &
  -0.21338834764831846     + i *  2.0816681711721685E-017, &
  0.50000000000000000     + i *  0.0000000000000000,     &
  -0.21338834764831846     + i * -2.0816681711721685E-017, &
  4.8321344214028144E-019 + i *  3.4694469519536142E-018, &
  -3.6611652351681609E-002 + i *  7.6327832942979512E-017, &
  0.0000000000000000     + i * -1.2325951644078309E-032 /)

 x( 0:100 ) = (/ (j*0.02, j=0,100) /)

 L = 2.0d0
 IDFT(:) = 0.0d0

 do m = 0, 100
  do n = -num/2, num/2
   IDFT(m) = IDFT(m) + F(n)* exp( 2.0d0 * pi * i * dble(n) * x(m) / L )
  end do

  write(*,*) x(m), IDFT(m)
 end do

 stop
end program
```



課題5：解答例

- ノコギリ波形を想定
- 離散フーリエ変換に与える点を考える
- 元の関数から、$f(x)$ に相当する値を計算し、プログラムに書き込む

```
program discrete_Fourier_transform
 implicit none

 ! --- definition ---

 integer, parameter :: num = 10
 real(8), parameter :: pi = 3.141592653589793d0
 complex(8), parameter :: i = ( 0.0d0, 1.0d0 )

 integer :: j, n, m
 real(8) :: f( 0:num-1 )
 complex(8) :: DFT( -num/2:num/2 )

 ! --- main process ---

 f( 0:num-1 ) = (/ 0.0d0,  0.2d0,  0.4d0,  0.6d0,  0.8d0, &
                  1.0d0, -0.8d0, -0.6d0, -0.4d0, -0.2d0 /)
 DFT( -num/2:num/2 ) = 0.0d0

 do n = -num/2, num/2
   do m = 0, num-1
     DFT(n) = DFT(n) + f(m) / dble(num) &
       * exp( -2.0d0 * pi * i * dble(m*n) / dble(num) )
   end do

   write(*,*) 'F(',n,') = ',real( DFT(n) ),' + i * ',aimag( DFT(n) )
 end do

 stop
end program
```

```
F(        -5 ) =  -0.10000000000000002      + i *    6.1232339957367648E-017
F(        -4 ) =   0.10000000000000001      + i *   -3.2491969623290685E-002
F(        -3 ) =  -9.9999999999999978E-002  + i *    7.2654252800536140E-002
F(        -2 ) =   9.9999999999999978E-002  + i *  -0.13763819204711741
F(        -1 ) =  -9.9999999999999950E-002  + i *   0.30776835371752531
F(         0 ) =   0.10000000000000002      + i *    0.0000000000000000
F(         1 ) =  -9.9999999999999950E-002  + i *  -0.30776835371752531
F(         2 ) =   9.9999999999999978E-002  + i *   0.13763819204711741
F(         3 ) =  -9.9999999999999978E-002  + i *   -7.2654252800536140E-002
F(         4 ) =   0.10000000000000001      + i *    3.2491969623290685E-002
F(         5 ) =  -0.10000000000000002      + i *   -6.1232339957367648E-017
```

課題6：解答例

- 課題5の離散フーリエ係数を設定し、IDFTを解く
- ノコギリ波のような波形が見える

```
program inverse_discrete_Fourier_transform
 implicit none

 ! --- definition ---

 integer, parameter :: num = 10
 real(8), parameter :: pi = 3.141592653589793d0
 complex(8), parameter :: i = ( 0.0d0, 1.0d0 )

 integer :: j, n, m
 real(8) :: L
 real(8) :: x( 0:100 )
 real(8) :: IDFT( 0:100 )
 complex(8) :: F( -num/2:num/2 )

 ! --- main process ---

 F(-num/2:num/2) = (/ &
  -0.10000000000000002     + i *   6.1232339957367648E-017, &
  0.10000000000000001      + i *  -3.2491969623290685E-002, &
  -9.9999999999999978E-002 + i *   7.2654252800536140E-002, &
  9.9999999999999978E-002 + i * -0.13763819204711741,     &
  -9.9999999999999950E-002 + i *  0.30776835371752531,     &
  0.10000000000000002      + i *   0.0000000000000000,     &
  -9.9999999999999950E-002 + i * -0.30776835371752531,     &
  9.9999999999999978E-002 + i *  0.13763819204711741,     &
  -9.9999999999999978E-002 + i *  -7.2654252800536140E-002, &
  0.10000000000000001      + i *   3.2491969623290685E-002, &
  -0.10000000000000002     + i *  -6.1232339957367648E-017 /)

 x( 0:100 ) = (/ (j*0.02d0, j=0,100) /)

 L = 2.0d0
 IDFT(:) = 0.0d0

 do m = 0, 100
  do n = -num/2, num/2
   IDFT(m) = IDFT(m) + F(n)* exp( 2.0d0 * pi * i * dble(n) * x(m) / L )
  end do

  write(*,*) x(m), IDFT(m)
 end do

 stop
end program
```