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」

2017.6.2

演習2

山浦 剛（tyamaura@riken.jp）

講義資料ページ

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

曲線の推定

- N次多項式ラグランジュ補間
  - $y = p_N(x) = \sum_{j=0}^{N} y_j \frac{(x-x_0)(x-x_1)\dots(x-x_{j-1})(x-x_{j+1})\dots(x-x_N)}{(x_j-x_0)(x_j-x_1)\dots(x_j-x_{j-1})(x_j-x_{j+1})\dots(x_j-x_N)} = \sum_{j=0}^{N} y_j \prod_{n=0}^{N} \frac{(x-x_n)}{(x_j-x_n)} \qquad (n \neq j)$
- 3次自然スプライン補間
  - $S(x) = S_j(x) = a_j(x-x_j)^3 + b_j(x-x_j)^2 + c_j(x-x_j) + d_j \qquad (x_j \leq x \leq x_{j+1})$
- 直線を仮定した最小2乗法
  - $y = \tilde{A}x + \tilde{B}$

（補足）スプライン補間

- 係数行列
  - $$\begin{bmatrix} 2(h_0+h_1) & h_1 & & & \mathbf{0} \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{N-3} & 2(h_{N-3}+h_{N-2}) & h_{N-2} \\ \mathbf{0} & & & h_{N-2} & 2(h_{N-2}+h_{N-1}) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{N-2} \\ v_{N-1} \end{bmatrix}$$
- 定数定義
  - $h_j = x_{j+1} - x_j$ $\quad \{j = 0,1,2,\dots,N-1\}$
  - $v_j = 6\left\{\frac{y_{j+1}-y_j}{h_j} - \frac{y_j-y_{j-1}}{h_{j-1}}\right\}$ $\quad \{j = 1,2,3,\dots,N-1\}$
  - $a_j = \frac{u_{j+1}-u_j}{6(x_{j+1}-x_j)}, \quad b_j = \frac{u_j}{2}, \quad c_j = \frac{y_{j+1}-y_j}{x_{j+1}-x_j} - \frac{1}{6}(u_{j+1}+2u_j)(x_{j+1}-x_j), \quad d_j = y_j$ $\quad \{j = 0,1,2,\dots,N-1\}$

（補足）直線の最小2乗法

- 最適な定数$(\tilde{A}, \tilde{B})$
  - $\tilde{A} = \frac{N\sum_{i=1}^{N} x_i y_i - \sum_{i=1}^{N} x_i \sum_{i=1}^{N} y_i}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2} = \frac{\sum_{i=1}^{N} x_i y_i - N\overline{X}\,\overline{Y}}{\sum_{i=1}^{N} x_i^2 - N\overline{X}^2} = \frac{\sum_{i=1}^{N}(x_i - \overline{X})(y_i - \overline{Y})}{\sum_{i=1}^{N}(x_i - \overline{X})^2}$
  - $\tilde{B} = \frac{\sum_{i=1}^{N} x_i^2 \sum_{i=1}^{N} y_i - \sum_{i=1}^{N} x_i y_i \sum_{i=1}^{N} x_i}{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2} = \overline{Y} - \overline{X}\tilde{A}$

（サンプル） ラグランジュ補間

```
program sample_lagrange
  implicit none

  ! parameter
  integer, parameter :: num_pt = 3
  integer, parameter :: num_lp = 100

  real(8), parameter :: px(num_pt) = (/ -1.0d0,   0.0d0,  1.0d0  /)
  real(8), parameter :: py(num_pt) = (/  0.05d0,  0.0d0,  0.05d0 /)
  real(8), parameter :: dx = 2.0d0

  ! work
  integer :: i, j, n

  real(8) :: x, y
  real(8) :: lx

  ! open file for data output
  open(unit=10, file='output_lagrange.csv')
```

```
  do i = 1, num_lp+1
    x = -1.0d0 + dx / dble( num_lp ) * dble( i - 1 )
    y = 0.0d0

    ! execute lagrange interpolation
    do j = 1, num_pt
      lx = 1.0d0

      do n = 1, num_pt
        if( n /= j ) lx = lx * ( x - px(n) ) / ( px(j) - px(n) )
      end do

      y = y + lx * py(j)
    end do

    ! write data to file
    write(unit=10, fmt='(2f20.16)') x, y
  end do
end program
```

（サンプル） スプライン補間

```
program sample_spline
  implicit none

  ! parameter
  integer, parameter :: num_pt = 5
  integer, parameter :: num_lp = 100

  real(8), parameter :: px(num_pt) = &
    (/ -1.0d0,  -0.5d0,  0.0d0,   0.5d0,   1.0d0  /)
  real(8), parameter :: py(num_pt) = &
    (/  0.05d0,  0.2d0,  0.0d0,  -0.2d0,  -0.05d0 /)

  ! work
  integer :: i, j

  real(8) :: u( num_pt   )
  real(8) :: v( num_pt-2 )
  real(8) :: h( num_pt-1 )

  real(8) :: h2( num_pt-2, num_pt-2 )

  real(8) :: a, b, c, d
  real(8) :: x, y
```

```
  ! determine the U vector
  u(:) = 0.0d0

  do j = 1, num_pt-1
    h(j) = px(j+1) - px(j)
  end do
  do j = 1, num_pt-2
    v(j) = ( ( py(j+2) - py(j+1) ) &
          / h(j+1) - ( py(j+1) - py(j) ) / h(j) ) * 6.0d0
  end do

  h2(:,:) = 0.0d0

  do j = 1, num_pt-2
  do i = 1, num_pt-2
    if( i == j+1 ) h2(i,j) = h(j+1)
    if( i == j   ) h2(i,j) = 2.0d0 * ( h(j+1) + h(j) )
    if( i == j-1 ) h2(i,j) = h(j)
  end do
  end do

  call MATRIX_SOLVER_tridiagonal( &
          h2(:,:), v(:), u(2:num_pt-1) )
```

```
  ! open file for data output
  open(unit=10, file='output_spline.csv')

  do j = 1, num_pt-1
    a = ( u(j+1) - u(j) ) / ( 6.0d0 * ( px(j+1) - px(j) ) )
    b = u(j) / 2.0d0
    c = ( py(j+1) - py(j) ) / ( px(j+1) - px(j) ) &
      - ( u(j+1) + 2.0d0 * u(j) ) * ( px(j+1) - px(j) ) / 6.0d0
    d = py(j)

    do i = 1, num_lp / num_pt + 1
      x = px(j) + ( px(j+1) - px(j) ) / dble( num_lp / num_pt ) * dble( i - 1 )
      y = a * ( x - px(j) )**3 + b * ( x - px(j) )**2 + c * ( x - px(j) ) + d

      ! write data to file
      write(unit=10, fmt='(2f20.16)') x, y
    end do

  end do
end program
```

Fortran 配列演算

- Fortranは配列を四則演算の要素にとることが可能
  - C/C++ ではできない、Fortranの特徴の1つ
- 配列同士の演算操作や、配列を実数倍することなどもできるので、行列演算などをdoループを使わなくても表記できる
- 配列の要素数は一致している必要がある
- 多次元配列でも同様に操作可能
- Cと違い、配列の要素番号は1から始まる

```
! 配列に配列を足したものを配列に代入する
  A(1:10) = B(1:10) + C(1:10)

! 配列と整数、配列と実数の演算も可能
  D(2:4) = E(5:7) * 5.0

! 配列番号を省略することも可能
  F(:) = G(:) + H(:) * I(:)

! 多次元配列でも同様
  J(:,:,:) = K(:,:,:) - L(:,:,:)
```

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
```

グラフの描画

- GNUplotを用いる
  - $ gnuplot
  - > plot ”output_lagrange.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

課題

1. サンプルプログラム2-1を参考に、$x = \pm\frac{\pi}{2}$ の範囲で、$y = 1 - \cos x$ をラグランジュ補間で曲線を推定せよ。推定のために用いるデータは５点以上用いること。
   - real(8), parameter :: pi = 3.1415926535897930d0
2. サンプルプログラム2-1を参考に、$x = \pm 1$ の範囲で、ルンゲ関数（$y = \frac{1}{1+25x^2}$）をラグランジュ補間で曲線を推定せよ。推定のために用いるデータは7点以上用いること。
3. サンプルプログラム2-2を参考に、$x = \pm 1$ の範囲で、ルンゲ関数（$y = \frac{1}{1+25x^2}$）を3次自然スプライン補間で曲線を推定せよ。推定のために用いるデータは7点以上用いること。
4. 下記の表のデータから$y = Ax + B$ を理論式として、最小2乗法で理論式の係数$\tilde{A}, \tilde{B}$を求めるプログラムを作成せよ。

x	15.961	11.967	1.180	5.476	14.408	10.591	10.343	6.421	9.574	5.280	9.691	10.532	15.659	8.814	15.644	11.662
y	79.637	56.528	3.464	23.768	68.656	46.281	48.538	28.989	47.785	22.847	49.276	47.392	84.165	43.825	82.966	55.900

提出方法

- 〆切：2017/06/16（金） 講義開始前まで
- メールにプログラムを添付
  - 主題：演習2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4ファイル
    - 課題2-1：r???????_kadai02-1.f90
    - 課題2-2：r???????_kadai02-2.f90
    - 課題2-3：r???????_kadai02-3.f90
    - 課題2-4：r???????_kadai02-4.f90

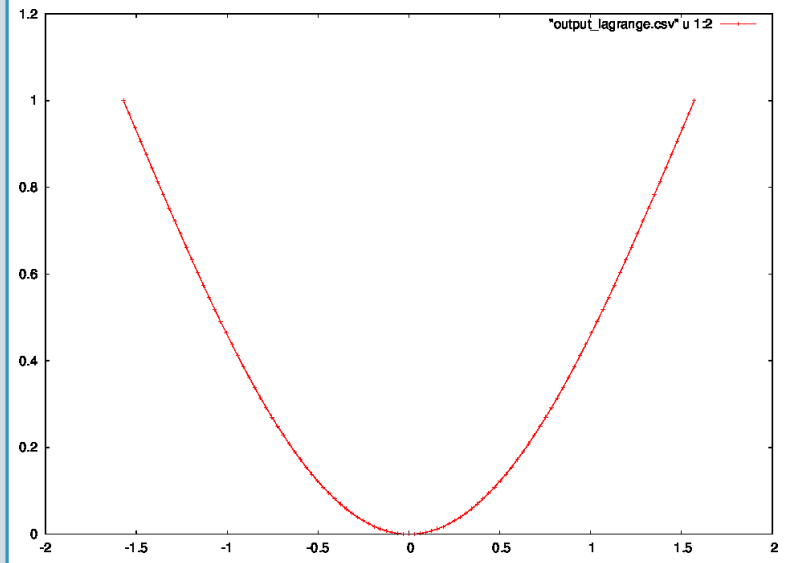
前期中間試験

- 6月5日（月）2時間目（50分）
- 出題範囲
  - 第1章「数値解析への案内」始め～ 第3章「曲線の推定」終わりまで
  - 教科書（pp.1～pp.55）、関連する講義ノートのいずれも含む
- 出題レベル
  - 教科書の章末問題程度
  - 知識を問う問題、計算問題、数値計算プログラムに関する問題

課題1：解答例

- $y = 1 - \cos(x)$ の5点を考える
  - $(x_1, y_1) = (-\frac{\pi}{2}, 1)$
  - $(x_2, y_2) = (-\frac{\pi}{3}, \frac{1}{2})$
  - $(x_3, y_3) = (0, 0)$
  - $(x_4, y_4) = (\frac{\pi}{3}, \frac{1}{2})$
  - $(x_5, y_5) = (\frac{\pi}{2}, 1)$
- 内挿の始点を考える

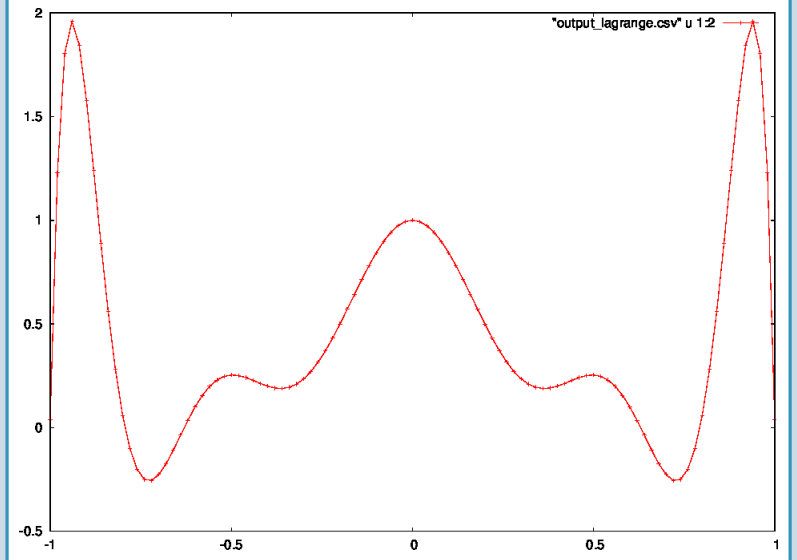
```
program sample_lagrange
  implicit none

  integer, parameter :: num_pt = 5
  integer, parameter :: num_lp = 100

  real(8), parameter :: pi = 3.141592653589793d0
  real(8), parameter :: px(num_pt) = &
    (/ -pi/2.0d0, -pi/3.0d0, 0.0d0, pi/3.0d0, pi/2.0d0 /)
  real(8), parameter :: py(num_pt) = &
    (/ 1.0d0, 0.5d0, 0.0d0, 0.5d0, 1.0d0 /)

  real(8), parameter :: dx = pi

  integer :: i, j, n

  real(8) :: x, y
  real(8) :: lx

  open(unit=10, file='output_lagrange.csv')

  do i = 1, num_lp+1
    x = -pi/2.0d0 + dx / dble( num_lp ) * dble( i - 1 )
    y = 0.0d0

    do j = 1, num_pt
      lx = 1.0d0

      do n = 1, num_pt
        if( n /= j ) lx = lx * ( x - px(n) ) / ( px(j) - px(n) )
      end do

      y = y + lx * py(j)
    end do

    write(unit=10, fmt='(2f20.16)') x, y
  end do

end program
```



課題2：解答例

- $y = \frac{1}{1+25x^2}$ の11点を考える
  - $(x_1, y_1) = (-1.0, 0.0385)$
  - $(x_2, y_2) = (-0.8, 0.0588)$
  - $(x_3, y_3) = (-0.6, 0.1)$
  - $(x_4, y_4) = (-0.4, 0.2)$
  - $(x_5, y_5) = (-0.2, 0.5)$
  - $(x_6, y_6) = (0.0, 1.0)$
  - $(x_7, y_7) = (0.2, 0.5)$
  - ……
- 内挿の始点を考える

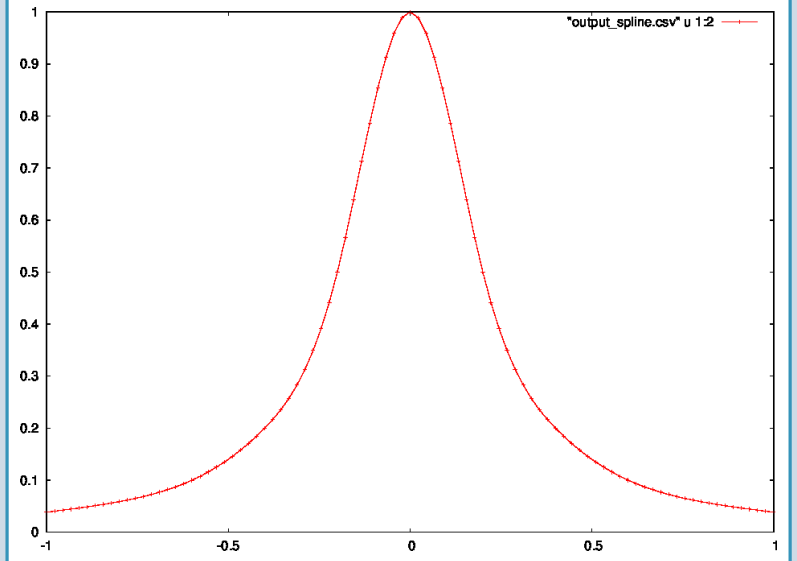
```
program sample_lagrange
  implicit none

  ! parameter
  integer, parameter :: num_pt = 11
  integer, parameter :: num_lp = 100

  real(8), parameter :: px(num_pt) = &
    (/ -1.0d0, -0.8d0, -0.6d0, -0.4d0, -0.2d0, 0.0d0, &
       0.2d0,  0.4d0,  0.6d0,  0.8d0,  1.0d0 /)
  real(8), parameter :: py(num_pt) = &
    (/ 0.0385d0, 0.0588d0, 0.1d0, 0.2d0, 0.5d0, 1.0d0, &
       0.5d0, 0.2d0,  0.1d0, 0.0588d0, 0.0385d0 /)

  real(8), parameter :: dx = 2.0d0

  integer :: i, j, n

  real(8) :: x, y
  real(8) :: lx

  open(unit=10, file='output_lagrange.csv')

  do i = 1, num_lp+1
    x = -1.0d0 + dx / dble( num_lp ) * dble( i - 1 )
    y = 0.0d0

    do j = 1, num_pt
      lx = 1.0d0

      do n = 1, num_pt
        if( n /= j ) lx = lx * ( x - px(n) ) / ( px(j) - px(n) )
      end do

      y = y + lx * py(j)
    end do

    write(unit=10, fmt='(2f20.16)') x, y
  end do

end program
```



課題3：解答例

- $y = \frac{1}{1+25x^2}$ の11点を考える
  - $(x_1, y_1) = (-1.0, 0.0385)$
  - $(x_2, y_2) = (-0.8, 0.0588)$
  - $(x_3, y_3) = (-0.6, 0.1)$
  - $(x_4, y_4) = (-0.4, 0.2)$
  - $(x_5, y_5) = (-0.2, 0.5)$
  - $(x_6, y_6) = (0.0, 1.0)$
  - $(x_7, y_7) = (0.2, 0.5)$
  - ……

```
program sample_spline
  implicit none
  integer, parameter :: num_pt = 11
  integer, parameter :: num_lp = 100
  real(8), parameter :: px(num_pt) = &
    (/ -1.0d0, -0.8d0, -0.6d0, -0.4d0, -0.2d0, 0.0d0, &
        0.2d0,  0.4d0,  0.6d0,  0.8d0,  1.0d0 /)
  real(8), parameter :: py(num_pt) = &
    (/ 0.0385d0, 0.0588d0,  0.1d0,    0.2d0,  0.5d0, 1.0d0, &
       0.5d0, 0.2d0,  0.1d0, 0.0588d0, 0.0385d0 /)

  integer :: i, j
  real(8) :: u( num_pt   )
  real(8) :: v( num_pt-2 )
  real(8) :: h( num_pt-1 )
  real(8) :: h2( num_pt-2, num_pt-2 )
  real(8) :: a, b, c, d
  real(8) :: x, y

  do j = 1, num_pt-1
    h(j) = px(j+1) - px(j)
  end do
  do j = 1, num_pt-2
    v(j) = ( ( py(j+2) - py(j+1) ) / h(j+1) - ( py(j+1) - py(j) ) / h(j) ) * 6.0d0
  end do

  h2(:,:) = 0.0d0
  do j = 1, num_pt-2
  do i = 1, num_pt-2
    if( i == j+1 ) h2(i,j) = h(j+1)
    if( i == j   ) h2(i,j) = 2.0d0 * ( h(j+1) + h(j) )
    if( i == j-1 ) h2(i,j) = h(j)
  end do
  end do

  u(:) = 0.0d0
  call MATRIX_SOLVER_tridiagonal( h2(:,:), v(:), u(2:num_pt-1) )
  open(unit=10, file='output_spline.csv')

  do j = 1, num_pt-1
    a = ( u(j+1) - u(j) ) / ( 6.0d0 * ( px(j+1) - px(j) ) )
    b = u(j) / 2.0d0
    c = ( py(j+1) - py(j) ) / ( px(j+1) - px(j) ) &
      - ( u(j+1) + 2.0d0 * u(j) ) * ( px(j+1) - px(j) ) / 6.0d0
    d = py(j)

    do i = 1, num_lp / num_pt + 1
      x = px(j) + ( px(j+1) - px(j) ) / dble( num_lp / num_pt ) * dble( i - 1 )
      y = a * ( x - px(j) )**3 + b * ( x - px(j) )**2 + c * ( x - px(j) ) + d
      write(unit=10, fmt='(2f20.16)') x, y
    end do
  end do
end program
```



課題4：解答例

- 点の標本平均を求める
  - Sum関数が便利
- 平均を引いた偏差配列を作成
- XとYの偏差積の和を求める
- Xの偏差平方和を求める
- 係数A, Bを求める

```
program sample_least_square_fitting
  implicit none

  integer, parameter :: num_pt = 16  ! number of data points

  real(8), parameter :: px(num_pt) = &
   (/ 15.961, 11.967,  1.180,  5.476, 14.408, 10.591, &
      10.343,  6.421,  9.574,  5.280, 9.691, 10.532, &
      15.659,  8.814, 15.644, 11.662 /)
  real(8), parameter :: py(num_pt) = &
   (/ 79.637, 56.528,  3.464, 23.768, 68.656, 46.281, &
      48.538, 28.989, 47.785, 22.847, 49.276, 47.392, &
      84.165, 43.825, 82.966, 55.900 /)

  integer :: i

  real(8) :: a, b
  real(8) :: cv, xv
  real(8) :: xm, xa(num_pt)
  real(8) :: ym, ya(num_pt)

  xm = sum( px(:) ) / real( num_pt )
  ym = sum( py(:) ) / real( num_pt )

  do i = 1, num_pt
    xa(i) = px(i) - xm
    ya(i) = py(i) - ym
  end do

  cv = 0.0d0
  xv = 0.0d0

  do i = 1, num_pt
    cv = cv + xa(i) * ya(i)
    xv = xv + xa(i)**2
  end do

  a = cv / xv
  b = ym - xm * a

  write(*,*) '( gradient, intercept ) = ', a, b
end program
```

```
( gradient, intercept ) = 5.3713004751836824 -5.4122096019567394
```