輪行予定日：2019 年 10 月 28 日
慶應義塾大学理工学部物理学科
岡崎健人

Ridge 回帰の実装

正則化パラメータ λ の値によって推定される曲線がどのように変わるのか、Fortran90 でプログラムを組んで実験した。ここでは正則化項を $\lambda\|\boldsymbol{w}\|_2^2$ とした（Ridge 回帰）。

1　正則化最小二乗法（Ridge 回帰）

トレーニングデータを $(x_i, y_i)_{i=1,\ldots,m}$ と与える。これを n 次関数 $\hat{y} = w_0 + w_1 x + w_2 x^2 + \cdots + w_n x^n = \sum_{k=0}^{n} w_k x^k$ で近似する。x の次数順に並べたベクトルを $\boldsymbol{x} = \left[1, x, x^2, \ldots, x^n\right]^\top$, 係数を並べたベクトルを $\boldsymbol{w} = [w_0, w_1, \ldots, w_n]^\top$ とすれば（これらは $n+1$ 次の列ベクトル）、この関数は $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$ となる。各トレーニングデータ (x_i, y_i) を使って単純にこの n 次多項式を計算した $\hat{y}_i = \sum_{k=0}^{n} w_k x_i^k$ を並べた m 次列ベクトルを $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_m]^\top$ とする。これは行ベクトル ${\boldsymbol{x}_i}^\top = \left[1, x_i, {x_i}^2, \ldots, {x_i}^n\right]$ を縦に並べた行列

$$X = \begin{bmatrix} {\boldsymbol{x}_1}^\top \\ {\boldsymbol{x}_1}^\top \\ \vdots \\ {\boldsymbol{x}_m}^\top \end{bmatrix} = \begin{bmatrix} 1 & x_1 & {x_1}^2 & \ldots & {x_1}^n \\ 1 & x_2 & {x_2}^2 & \ldots & {x_2}^n \\ & & \vdots & & \\ 1 & x_m & {x_m}^2 & \ldots & {x_m}^n \end{bmatrix}$$

を用いて $\hat{\boldsymbol{y}} = X\boldsymbol{w}$ とかける。ただしこの X は $m \times (n+1)$ の行列である。平均二乗誤差 $\mathrm{MSE}(\boldsymbol{w})$ を行列で表現して展開すると

$$\begin{aligned} \mathrm{MSE}(\boldsymbol{w}) &= \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{m}\|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2^2 = \frac{1}{m}\|X\boldsymbol{w} - \boldsymbol{y}\|_2^2 \\ &= \frac{1}{m}(X\boldsymbol{w} - \boldsymbol{y})^\top (X\boldsymbol{w} - \boldsymbol{y}) = \frac{1}{m}\left(\boldsymbol{w}^\top X^\top - \boldsymbol{y}^\top\right)(X\boldsymbol{w} - \boldsymbol{y}) \\ &= \frac{1}{m}\left(\boldsymbol{w}^\top X^\top X\boldsymbol{w} - \boldsymbol{y}^\top X\boldsymbol{w} - \boldsymbol{w}^\top X^\top \boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{y}\right) \end{aligned}$$

となる。ただし $\boldsymbol{y} = [y_1, \ldots, y_m]^\top$ とした。目下の目的は平均二乗誤差 $\mathrm{MSE}(\boldsymbol{w})$ を最小化する $\boldsymbol{w}$ を求めることである。行列の微分の公式

$$\frac{\partial}{\partial \boldsymbol{w}}\left(\boldsymbol{w}^\top D\boldsymbol{w}\right) = \left(D + D^\top\right)\boldsymbol{w}, \quad \frac{\partial}{\partial \boldsymbol{w}}\left(\boldsymbol{a}^\top \boldsymbol{w}\right) = \frac{\partial}{\partial \boldsymbol{w}}\left(\boldsymbol{w}^\top \boldsymbol{a}\right) = \boldsymbol{a}$$

を用いて平均二乗誤差 $\mathrm{MSE}(\boldsymbol{w})$ を微分すると

$$\frac{\partial \mathrm{MSE}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{2}{m}\left(X^\top X\boldsymbol{w} - X^\top \boldsymbol{y}\right)$$

となるので、これが $\boldsymbol{0}$ となるのは $\boldsymbol{w} = \left(X^\top X\right)^{-1} X^\top \boldsymbol{y}$ のときである。

つぎに平均二乗誤差 $\mathrm{MSE}(\boldsymbol{w})$ を最小化するのではなく、重み減衰 $\lambda\|\boldsymbol{w}\|_2^2 = \lambda\boldsymbol{w}^\top\boldsymbol{w}$ を加えたもの $J(\boldsymbol{w}) = \mathrm{MSE}(\boldsymbol{w}) + \lambda\boldsymbol{w}^\top\boldsymbol{w}$ を最小化することを考える。$\partial(\boldsymbol{w}^\top\boldsymbol{w})/\partial\boldsymbol{w} = 2\boldsymbol{w}$ より

$$\frac{\partial J(\boldsymbol{w})}{\partial\boldsymbol{w}} = \frac{\partial \mathrm{MSE}(\boldsymbol{w})}{\partial\boldsymbol{w}} + \frac{\partial\left(\lambda\boldsymbol{w}^\top\boldsymbol{w}\right)}{\partial\boldsymbol{w}} = \frac{2}{m}\left(X^\top X\boldsymbol{w} - X^\top\boldsymbol{y}\right) + 2\lambda\boldsymbol{w},$$

したがってこれが $\mathbf{0}$ となるのは $\boldsymbol{w} = \left(X^\top X + m\lambda I\right)^{-1} X^\top\boldsymbol{y}$ のときである。

プログラムを書くにあたって記号をいろいろ定義する。$A := X^\top X,\ B := A + m\lambda I,\ \boldsymbol{v} := X^\top\boldsymbol{y}$ とおく。すると目的は $(n+1)$ 本の 1 次連立方程式 $B\boldsymbol{w} = \boldsymbol{v}$ を $\boldsymbol{w}$ について解くことに帰着される。この目的は、さらに拡大係数行列 $C := \begin{bmatrix} B & \boldsymbol{v} \end{bmatrix}$ にガウスの掃き出し法を行うことに帰着される。ただし

$$X^\top = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_m \\ {x_1}^2 & {x_2}^2 & \cdots & {x_m}^2 \\ \vdots & \vdots & & \vdots \\ {x_1}^n & {x_2}^n & \cdots & {x_m}^n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_1 & {x_1}^2 & \dots & {x_1}^n \\ 1 & x_2 & {x_2}^2 & \dots & {x_2}^n \\ & & \vdots & & \\ 1 & x_m & {x_m}^2 & \dots & {x_m}^n \end{bmatrix}$$

より A の成分を具体的に書くと

$$A = X^\top X = \begin{bmatrix} s_0 & s_1 & s_2 & \cdots & s_m \\ s_1 & s_2 & s_3 & \cdots & s_{n+1} \\ s_2 & s_3 & \cdots & \cdots & s_{n+2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ s_n & s_{n+1} & s_{n+2} & \cdots & s_{n+n} \end{bmatrix}, \quad s_k = \sum_{i=1}^{m} {x_i}^k, \quad k = 0, \ldots, 2n$$

となる。つまり A は対角線に垂直な方向の成分がすべて同じであるような、$(n+1)$ 次の対角行列になっている。その成分は $a_{ij} = s_{i+j-2},\ i, j = 1, \ldots, n+1$ と定式化できる。B はその対角成分に $m\lambda$ を加えた行列である。ただし λ は任意の変数であるから、$m\lambda$ ではなく単に λ を加えることにする。

$$b_{ij} = \begin{cases} a_{ij} + \lambda, & \text{if } i = j, \\ a_{ij}, & \text{if } i \neq j. \end{cases}$$

さらに $\boldsymbol{v}$ を具体的に示すと

$$\boldsymbol{v} = X^\top\boldsymbol{y} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_m \\ {x_1}^2 & {x_2}^2 & \cdots & {x_m}^2 \\ \vdots & \vdots & & \vdots \\ {x_1}^n & {x_2}^n & \cdots & {x_m}^n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{bmatrix}, \quad v_l = \sum_{i=1}^{m} {x_i}^l y_i, \quad l = 0, \ldots, n$$

である。拡大係数行列 C は

$$C = \begin{bmatrix} B & \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{21} & \cdots & b_{n+1,1} & v_0 \\ b_{12} & \ddots & & \vdots & v_1 \\ \vdots & & \ddots & \vdots & \vdots \\ b_{1,n+1} & \cdots & \cdots & b_{n+1,n+1} & v_n \end{bmatrix} \tag{1}$$

よりその成分 $c_{ij},\ i = 1, \ldots, n+1,\ j = 1, \ldots, n+2$ は

$$c_{ij} = \begin{cases} b_{ij}, & \text{for } i, j = 1, \ldots, n+1, \\ v_{i-1}, & \text{for } i = 1, \ldots, n+1,\ j = n+2 \end{cases}$$

とかける。これは $(n+1) \times (n+2)$ の行列になる。

2　実験方法とソースコード

まずデータを読み込んで、推定したい多項式の係数 $\boldsymbol{w} = [w_0, \ldots, w_n]^\top$ と、推定曲線の (x, y) の値を出力するプログラムを作成した。それをソースコード 1 に示す。$\boldsymbol{w} = [w_0, \ldots, w_n]^\top$ を求めるのには、式 (1) の拡大係数行列 C にガウスの消去法を適用した。データの数 m, 多項式の次数 n, 正則化パラメータ λ, プロットの分割幅はプログラム作成者が入力できるようにした。m 個の訓練データは、区間 $[0, 2)$ の一様乱数を関数 $2\cos(2x) + x$ に足すことで生成した。それをソースコード 2 に示す。

正則化パラメータを $\lambda = 10^4,\ 10^2,\ 10^0,\ 10^{-2},\ 10^{-4},\ 0$ としたときに、$m = 30$ 個のデータを $n = 10,\ 20,\ 30$ 次の多項式で推定した。

ソースコード 1　リッジ回帰のソースコード

```
program ridge_regression
 implicit none

 real(8), allocatable :: x(:), y(:), data(:), s(:), v(:)
 real(8), allocatable :: a(:, :), b(:, :), c(:, :), w(:)
 real(8) :: lambda, p, q, width, delta, xmin, y_hat, x1
 integer :: i, j, n, m, k, l, i1, i2

 ! ***入力*********************
   ! データの数
   m = 30

   ! 近似したい曲線の次数
   n = 10

   ! 正則化パラメータ
   lambda = 1.0_8

 !***データの読み込み***********************
  allocate(x(m), y(m), data(2 * m))
  ! mx2 のデータ（「data.txt」など）をサイズ 2m の配列 data(2*m)として読み込む。
  ! （「./a.out < data.txt」）
  ! data(2*m)はx,y,x,y,…のような並びになっている。

  read(*,*) data

  ! data(2*m)の奇数番目をx(j), 偶数番目をy(j)とする。
  do i = 1, 2*m
    if ( mod(i, 2) == 0 ) then
      j = i / 2
      y(j) = data(i)
    else if ( mod(i, 2) == 1 ) then
      j = (i + 1) / 2
```

```
      x(j) = data(i)
    end if
  end do

! 確認済み
! do j = 1, m
! write(*,*) j, x(j), y(j)
! end do

!***拡大係数行列の準備************************************
! 解きたい連立方程式Bw=v の B と v の成分を計算しておく。
! 拡大係数行列C=[B v]を構築する。
  allocate(s(0:2 * n), a(n + 1, n + 1), b(n + 1, n + 1), v(0 : n), c(n + 1, n + 2))

! s_k = sum_{i=1}^{m} {x_i}^k, k=0~2n の計算
  do k = 0, 2 * n
    s(k) = 0.0_8
    do i = 1, m
      s(k) = s(k) + (x(i)) ** (real(k))
    end do
! write(*,*) k, s(k)
  end do

  ! 行列A の計算。その成分は a_{ij}=s_{i+j-2}, i,j=1~n+1.
  do j = 1, n + 1
    do i = 1, n + 1
      a(i, j) = s(i + j - 2)
  ! write(*,*) i, j, i + j - 2, s(i + j - 2), a(i, j) ! 確認済み
    end do
  end do

  ! 行列B の計算。行列 A の対角成分に lambda を足すだけ。
  do j = 1, n + 1
    do i = 1, n + 1
      if ( i == j ) then
        b(i, j) = a(i, j) + lambda
      else if ( i /= j ) then
        b(i, j) = a(i, j)
      end if
! write(*,*) i, j, b(i, j) - a(i, j) ! 確認済み。
    end do
  end do

! v_l = sum_{i=1}^{m} y_i {x_i}^l, l=0~n の計算
   do l = 0, n
     v(l) = 0.0_8
```

```
     do i = 1, m
       v(l) = v(l) + y(i) * (x(i)) ** real(l)
     end do
! write(*,*) l, v(l) ! 確認済み
   end do

! 拡大係数行列C=[B v]を作る
  do i = 0, n
    do j = 1, n + 2
      if (j == n + 2) then
        c(i + 1, j) = v(i)
      else if (j /= n + 2) then
        c(i + 1, j) = b(i + 1, j)
      end if
    end do
  end do
! 確認済み
! do i = 0, n
! do j = 1, n + 2
! write(*,*) i + 1, j, v(i), c(i + 1, j)
! end do
! end do

! ***ガウスの掃き出し法*************************************
! 上で作った (n+1)x(n+2)の拡大係数行列C にガウスの掃き出し法を適用する。
! あるk 行目の対角成分 c(k,k)をp とおき、その行すべてを p で割る。
! そうすればその対角成分c(k,k)は 1になる。
! k 行目以外の行について（これを i 行目とする)、c(k,k)と同じ列にあるもの (c(i,k))をq とおく。
! そのi 行目の k 列目以降の成分 c(i,j), j=k~n+2 からq*c(k,j)を引く。
! これでc(k,k)の上下はゼロになる。
! このことをすべての行 k=1~n+1 について行う。

  do k = 1, n + 1
    p = c(k, k)
    do j = k, n + 2
      c(k, j) = c(k, j) / p
    end do
    do i = 1, n + 1
      if (i /= k) then
        q = c(i, k)
        do j = k, n + 2
          c(i, j) = c(i, j) - q * c(k, j)
        end do
      end if
    end do
  end do
```

```

! 確認済み
! do i = 1, n + 1
! do j = 1, n + 2
! write(*,*) i, j, c(i, j)
! end do
! end do

! こうして得られた行列の最後の列c(i,n+2), i=1~n+1がw の解である。
  allocate(w(n + 1))
    do i = 1, n + 1
      w(i) = c(i, n + 2)
      write(*,*) i - 1, w(i) ! 確認済み
    end do

! ***曲線のプロット****************************************
! 分割数
  i1 = 200

! x の幅 width と x の最小値 xmin を計算。
  width = maxval(x) - minval(x)
  xmin = minval(x)
! write(*,*) width, xmin, maxval(x) ! 確認済み

  ! 曲線のプロットの刻み幅
  delta = width / real( i1 )

  open (18, file='line.txt', status='replace')
  do i2 = 0, i1
    y_hat = 0.0_8
    x1 = xmin + real(i2) * delta
      do i = 1, n + 1
        y_hat = y_hat + w(i) * x1 ** (real(i - 1))
      end do
    write(18, *) x1, y_hat ! ここでx に対する近似曲線 y_hat=sum_{k=0}^{n} w_k x^k の値を出力
  end do
  close(18)

  deallocate(x, y, data, s, v, a, b, c, w)

 stop
end program
```

ソースコード 2　データ生成のためのプログラム

```
program data_generator
 implicit none
```

```
integer :: m, imax, i
real(8) :: xmin, xmax, width, delta, x, y
real(8), allocatable :: random(:)


! データの個数
 m = 30

 allocate(random(m + 1))
 call random_number(random)
 ! 確認
! do i = 0, m
! write(*,*) i, 2.0_8 * random(i + 1)
! end do

! データの最小値・最大値の設定、幅、分割幅
 xmin = 0.0_8
 xmax = 6.0_8
 width = xmax - xmin
 delta = width / real(m)
! write(*,*) width

 open(18, file='generated_data.txt', status='replace')
 do i = 0, m - 1
   x = xmin + delta * real(i)
   y = 2.0_8 * cos(2.0_8 * x) + x + 2.0_8 * random(i + 1)
   write(18,*) x, y
 end do
 close(18)

 deallocate(random)

 stop
end program
```

3 出力結果

$n = 10,\ 20,\ 30$ に対して正則化パラメータを $\lambda = 10^4,\ 10^2,\ 10^0,\ 10^{-2},\ 10^{-4},\ 0$ としたときの推定結果を図 1 から図 6 に示す。与えた 30 個のデータは点で表し、推定した曲線を実線で表している。ただし推定した曲線は 200 個の点を線で繋いだものなので、鋭いピークがあるとそれが削れてい表示されている可能性がある。

$n = 10$ など n が比較的小さいとき、λ の値が大きいときちんとデータ点を推定しておらず、過少適合をおこしているのが観察できる。逆に $n = 30$ など n が大きく λ が小さいときには過剰適合を起こしている。

$n = 30,\ \lambda = 10^2$ のときの推定曲線が過剰に振動している。λ の値を少し変えて $\lambda = 99, 101$ として比較したが、そのようなことは起こっていなかった（図 7）。なぜそうなるのかはよくわからない。

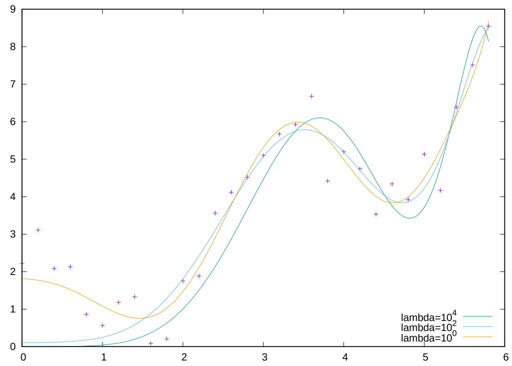

図 1　$n = 10,\ \lambda = 10^4, 10^2,\ 0^0$

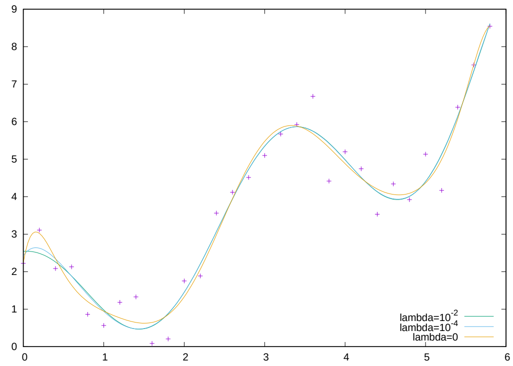

図 2　$n = 10,\ \lambda = 10^{-2}, 10^{-4}, 0$

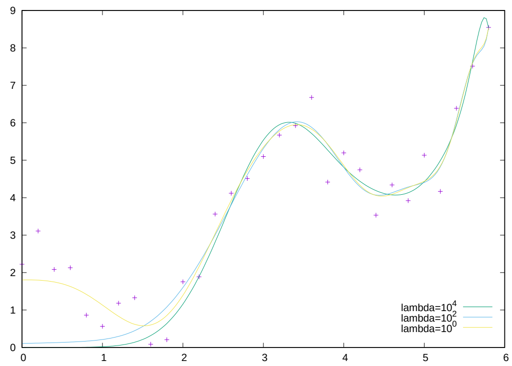

図 3　$n = 20,\ \lambda = 10^4, 10^2, 10^0$

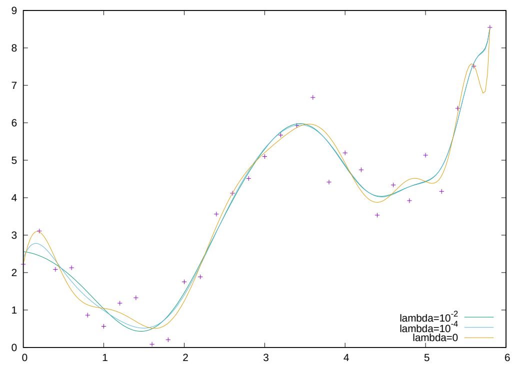

図 4　$n = 20,\ \lambda = 10^{-2}, 10^{-4}, 0$

4　今後の課題

なぜ $n = 30,\ \lambda = 10^2$ のときに異常な出力を得たのかを解決したい。

また適切な n と λ をどう決めるのかわからなかった。そこで n と λ を少しづつ変えて出力させ、テストデータを与えて汎化誤差が一番小さい n と λ を適切なものとする、というようなプログラムも書けるだろう。

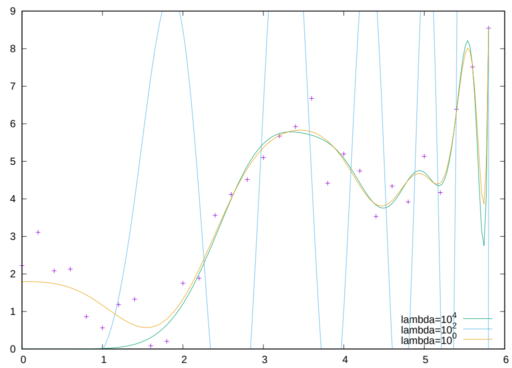

図 5　$n = 30,\ \lambda = 10^4, 10^2, 10^0$

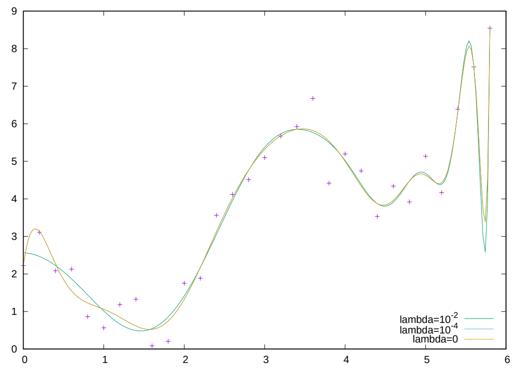

図 6　$n = 30,\ \lambda = 10^{-2}, 10^{-4}, 0$

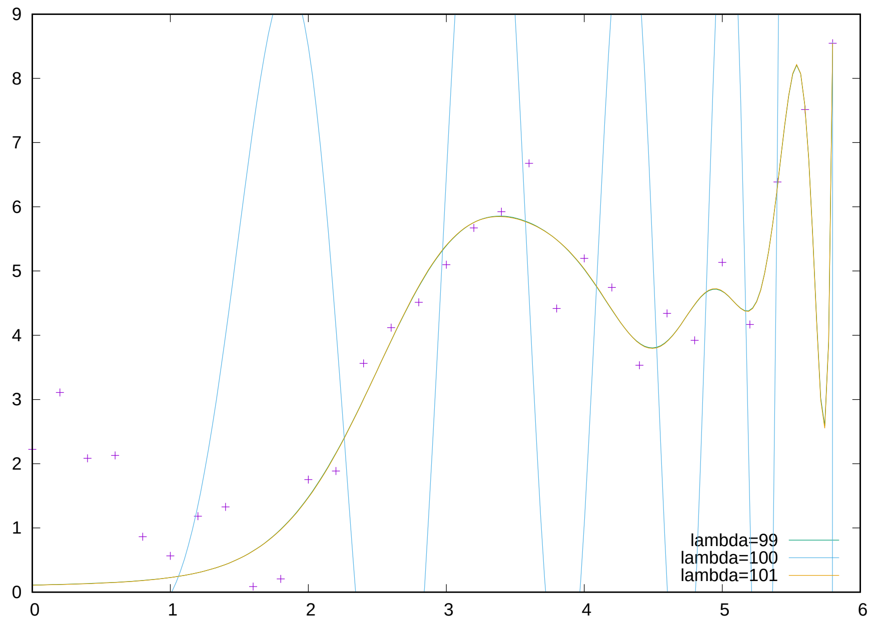

図 7　$n = 30$ のときに $\lambda = 99, 100, 101$ を比較したもの