

3 Interpolacija in ekstrapolacija

Pri obdelavi merskih podatkov, pridobljenih bodisi avtomatsko bodisi človeško odčitanih, pogosto naletimo na problem, kako določiti neznano vrednost med dvema izmerjenima točkama. Povedano v bolj matematičnem jeziku, imamo vrednosti posameznih točk (x_i, y_i) , ne poznamo pa funkcijske zveze $y = f(x)$.

Ogledali si bomo dva najpogosteje uporabljana primera interpolacije na funkcijah ene spremenljivke.

3.1 Polinomska interpolacija

Skazi niz N točk (x_i, y_i) lahko speljemo polinom stopnje $N - 1$, ki ga določimo z Lagrangeovo formulo:

$$P(x) = \frac{(x-x_2)(x-x_3)\dots(x-x_N)}{(x_1-x_2)(x_1-x_3)\dots(x_1-x_N)}y_1 + \frac{(x-x_1)(x-x_3)\dots(x-x_N)}{(x_2-x_1)(x_2-x_3)\dots(x_2-x_N)}y_2 + \dots + \frac{(x-x_1)(x-x_2)\dots(x-x_{N-1})}{(x_N-x_1)(x_N-x_2)\dots(x_N-x_{N-1})}y_N \quad (1)$$

Čeprav je pravilen in enostaven, se ta način interpolacije ne uporablja prav pogosto. Prvi razlog je, da metoda ne da ocene napake, drugi pa leži v nepraktičnosti uporabe, še zlasti pri večjem številu merskih točk in posledično višjih stopnjah polinomov.

3.2 Kubična spline interpolacija

Spline je v angleškem jeziku ime za odsekoma preprost kubični polinom, ki ga je moč tudi na večjem intervalu sestaviti v zvezno in gladko funkcijo. Ta metoda interpolacije je n.pr. izredno razširjena v algoritmih za digitalno obdelavo slik in prikaz računalniške grafike.

Osnove

N točk (x_i, y_i) razdeli interval $[x_1, x_N]$ na $N - 1$ delov. Skonstruirati želimo tako funkcijo $F(x)$, da je zvezna in gladka na celotnem intervalu $[x_1, x_N]$. Zapišemo jo takole:

$$F(x) = \begin{cases} f_1(x) & x_1 \leq x < x_2 \\ f_2(x) & x_2 \leq x < x_3 \\ \dots & \dots \\ f_{N-1}(x) & x_{N-1} \leq x < x_N \end{cases} \quad (2)$$

Na vsakem intervalu $[x_i, x_{i+1}]$ skonstruiramo kubični polinom $f_i(x) = a_i(x-x_i)^3 + b_i(x-x_i)^2 + c_i(x-x_i) + d_i$ za $i = 1, 2, 3, \dots, N-1$. Določiti moramo koeficiente a_i, b_i, c_i in d_i , kar je možno z uporabo samih točk ter iz zahtev po zveznosti in gladkosti funkcije $F(x)$. Potrebujemo torej še prve in druge odvode funkcij $f_i(x)$:

$$f'_i(x) = 3a_i(x-x_i)^2 + 2b_i(x-x_i) + c_i \quad (3)$$

$$f''_i(x) = 6a_i(x-x_i) + 2b_i \quad (4)$$

Lastnosti kubičnega splina:

- funkcija $F(x)$ gre skozi vse točke (x_i, y_i) , $i = 1, 2, 3, \dots, N$
- funkcija $F(x)$ je zvezna na intervalu $[x_1, x_N]$
- funkcija $F'(x)$ je zvezna na intervalu $[x_1, x_N]$
- funkcija $F''(x)$ je zvezna na intervalu $[x_1, x_N]$

Velja $F(x_i) = y_i$ in še:

$$y_i = f_i(x_i) \tag{5}$$

$$y_i = a_i(x_i - x_i)^3 + b_i(x_i - x_i)^2 + c_i(x_i - x_i) + d_i = d_i \tag{6}$$

Zveznost funkcije $F(x)$ pomeni, da se morajo vrednosti funkcij f_i na mejah podintervalov ujemati:

$$f_i(x_i) = f_{i-1}(x_i) = d_i \tag{7}$$

$$d_i = a_{i-1}(x_i - x_{i-1})^3 + b_{i-1}(x_i - x_{i-1})^2 + c_{i-1}(x_i - x_{i-1}) + d_{i-1} \tag{8}$$

če zapišemo $h = x_i - x_{i-1}$, se zgornja enačba glasi takole:

$$d_i = a_{i-1}h^3 + b_{i-1}h^2 + c_{i-1}h + d_{i-1} \tag{9}$$

Ker od splina zahtevamo gladkost krivulje, morajo biti na stičiščih podintervalov prvi odvodi enaki: $f'_i(x_i) = f'_{i-1}(x_i)$. Ker je

$$f_i(x_i) = 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i = c_i \tag{10}$$

$$f'_{i-1}(x_i) = 3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1} = c_i \tag{11}$$

Če ponovno zapišemo $h = x_i - x_{i-1}$, dobimo:

$$c_i = 3a_{i-1}h^2 + 2b_{i-1}h + c_{i-1} \tag{12}$$

Tudi iz enačbe za drugi odvod dobimo koristne podatke:

$$f''_i(x_i) = 6a_i(x_i - x_i) + 2b_i = 2b_i \tag{13}$$

Iz zahteve po zveznosti drugih odvodov $f''_i(x_i) = f''_{i+1}(x_i)$ sledi:

$$f''_i(x_{i+1}) = 6a_i(x_{i+1} - x_i) + 2b_i \tag{14}$$

$$f''_{i+1}(x_{i+1}) = 6a_i(x_{i+1} - x_i) + 2b_i \tag{15}$$

Ker velja $f''_{i+1}(x_{i+1}) = f''_i(x_{i+1}) = 2b_i$, ob upoštevanju $h = x_{i+1} - x_i$ dobimo zvezo

$$2b_{i+1} = 6a_i h + 2b_i \tag{16}$$

Zaradi bolj preglednega zapisa označimo druge odvode $f_i''(x_i)$ kot M_i in izrazimo koeficiente polinomov a_i, b_i, c_i in d_i z M_i in y_i . Že od prej vemo, da je $d_i = y_i$ in $M_i = 2b_i$, od koder sledi, da je $b_i = \frac{M_i}{2}$. Iz zveze

$$2b_{i+1} = 6a_i h + 2b_i \quad (17)$$

dobimo

$$a_i = \frac{M_{i+1} - M_i}{6h} \quad (18)$$

ter iz zveze

$$d_{i+1} = a_i h^3 + b_i h^2 + c_i h + d_i \quad (19)$$

po malenkost daljšem premetavanju členov pridemo do

$$c_i = \frac{y_{i+1} - y_i}{h} - \left(\frac{M_{i+1} + 2M_i}{6} \right) h. \quad (20)$$

Za določitev a_i, b_i, c_i in d_i ($i = 1, 2, \dots, N - 1$) imamo sedaj na voljo (skoraj) dovolj enačb. Ko v enačbo

$$c_{i+1} = 3a_i h^2 + 2b_i h + c_i \quad (21)$$

vstavimo z M_i in y_i izražene koeficiente a_i, b_i in c_i , dobimo enačbe oblike

$$M_i + 4M_{i+1} + M_{i+2} = \frac{6}{h^2} (y_i - 2y_{i+1} + y_{i+2}), \quad (22)$$

ki jih pregledneje zapišemo v matrični obliki:

$$\begin{pmatrix} 1 & 4 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{N-3} \\ M_{N-2} \\ M_{N-1} \\ M_N \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{N-4} - 2y_{N-3} + y_{N-2} \\ y_{N-3} - 2y_{N-2} + y_{N-1} \\ y_{N-2} - 2y_{N-1} + y_N \end{pmatrix} \quad (23)$$

Opazimo, da je vrstic $N - 2$, medtem ko je stolpcev N . Da bi sistem enačb lahko rešili, moramo dobiti dve novi zvezi oziroma pogoja. Prvi je

$$M_1 = M_N = 0 \quad (24)$$

kar pomeni, da sta druga odvoda na krajiščih intervala enaka nič. Takemu splinu pravimo tudi *naravni spline* in

sistem $n - 2$ enačb postane rešljiv ter dobi naslednjo obliko:

$$\begin{pmatrix} 1 & 4 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{N-3} \\ M_{N-2} \\ M_{N-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{N-4} - 2y_{N-3} + y_{N-2} \\ y_{N-3} - 2y_{N-2} + y_{N-1} \\ y_{N-2} - 2y_{N-1} + y_N \end{pmatrix} \quad (25)$$

Druga, tudi pogosto uporabljena možnost pa je, da postavimo

$$\begin{aligned} M_1 &= M_2 \\ M_{N-1} &= M_N \end{aligned} \quad (26)$$

kar v praksi pomeni, da dobi spline na krajiščih intervala $[x_1, x_N]$ parbolično obliko. Ta je ugodna, kadar imamo opravka s periodičnimi ali eksponentnimi funkcijami.

Naloga

Z uporabo podprogramov `spline.f90`, `splint.f90` ter `polint.f90` iz knjige Numerical Recipes [1, 2] interpolirajte tabelarično podano funkcijo, in sicer najprej s polinomsko in nato s spline interpolacijo.

```

SUBROUTINE polint(xa,ya,n,x,y,dy)
IMPLICIT NONE
INTEGER(4),PARAMETER :: NMAX=10
INTEGER(4) :: n,i,m,ns
REAL(8) :: dy,x,y,xa(n),ya(n)
REAL(8) :: den,dif,dift,ho,hp,w,c(NMAX),d(NMAX)

!Largest anticipated value of n. Given arrays xa and ya, each of
!length n, and given a value x, this routine returns a value y,
!and an error estimate dy. If P(x) is the polynomial of degree N-1
!such that P(xai) = ya_i, i = 1, . . . , n, then the returned value y = P(x).

ns=1
dif=abs(x-xa(1))

do i=1,n
  dift=abs(x-xa(i))
  if (dift<dif) then
    ns=i
    dif=dift
  endif
enddo

c(ns)=ya(ns)
d(ns)=ya(ns)

do m=1,n-1
  do i=1,n-m
    ho=xa(i)-x
    hp=xa(i+m)-x
    w=c(i+1)-d(i)
    den=ho-hp
    if(den==0.)pause 'failure in polint'
    den=w/den
    d(i)=hp*den
    c(i)=ho*den
  enddo
enddo

```

```

        enddo
        if (2*ns<n-m)then
            dy=c(ns+1)
        else
            dy=d(ns)
            ns=ns-1
        endif
        y=y+dy
    enddo

return
END

SUBROUTINE spline(x,y,n,yp1,ypn,y2)
IMPLICIT NONE
INTEGER(4),PARAMETER :: NMAX=500
INTEGER(4) :: n,i,k
REAL(8) :: yp1,ypn,x(n),y(n),y2(n),p,qn,sig,u(NMAX)
!Given arrays x(1:n) and y(1:n) containing a tabulated function, i.e.,
!yi = f(xi), with x1 < x2 < . . . < xN, and given values yp1 and ypn for
!the first derivative of the interpolating function at points 1 and n,
!respectively, this routine returns an array y2(1:n) of length n which
!contains the second derivatives of the interpolating function at the
!tabulated points xi. If yp1 and/or ypn are equal to 1 Å 1030 or larger,
!the routine is signaled to set the corresponding boundary condition for
!a natural spline, with zero second derivative on that boundary.
!Parameter: NMAX is the largest anticipated value of n.
if (yp1>.99e30) then
    y2(1)=0. ! The lower boundary condition is set either to be natural
    u(1)=0.
else !or else to have a specified first derivative.
    y2(1)=-0.5
    u(1)=(3./(x(2)-x(1)))*((y(2)-y(1))/(x(2)-x(1))-yp1)
endif

do i=2,n-1
!This is the decomposition loop of the tridiagonal algorithm. y2 and u
!are used for temporary storage of the decomposed factors.
    sig=(x(i)-x(i-1))/(x(i+1)-x(i-1))
    p=sig*y2(i-1)+2.
    y2(i)=(sig-1.)/p
    u(i)=(6.*((y(i+1)-y(i))/(x(i+1)-x(i)))-(y(i)-y(i-1)) &
        /(x(i)-x(i-1)))/(x(i+1)-x(i-1))-sig*u(i-1))/p
enddo

if (ypn>.99e30) then
    qn=0. !The upper boundary condition is set either to be natural
    un=0.
else
    qn=0.5
    un=(3./(x(n)-x(n-1)))*(ypn-(y(n)-y(n-1))/(x(n)-x(n-1)))
endif

y2(n)=(un-qn*u(n-1))/(qn*y2(n-1)+1.)

do k=n-1,1,-1
    y2(k)=y2(k)*y2(k+1)+u(k)
enddo

return
END

SUBROUTINE splint(xa,ya,y2a,n,x,y)
IMPLICIT NONE

INTEGER(4) :: n,k,khi,klo
REAL(8) :: x,y,xa(n),y2a(n),ya(n), a,b,h

!Given the arrays xa(1:n) and ya(1:n) of length n, which tabulate xai s
!in order), and given the array y2a(1:n), which is the output and given
!a value of x, this routine returns a cubic-spline interpolated
!
!We will find the right place in the table by This is optimal if sequential
!calls to this routine values of x. If sequential calls are in spaced, one
!would do better to store klo and khi and test if they remain next call.

```

```

klo=1
khi=n

1 continue

if ((khi-klo)>1) then
  k=(khi+klo)/2
  if(xa(k)>x)then
    khi=k
  else
    klo=k
  endif
  goto 1
endif

!klo and khi now bracket the input value h=xa(khi)-xa(klo)

if (h.eq.0.) pause 'bad xa input in splint'
!the xa s must be distinct.
a=(xa(khi)-x)/h
!Cubic spline polynomial is now evaluated.
b=(x-xa(klo))/h
y=a*ya(klo)+b*ya(khi)+((a**3-a)*y2a(klo)+(b**3-b)*y2a(khi))*(h**2)/6.

return
END

```