TF2202 Komputasi Rekayasa

Interpolasi dan Pencocokan Kurva

Tim Praktikum Komputasi Rekayasa 2021
Teknik Fisika
Institut Teknologi Bandung

1 Polinomial Interpolasi Newton

Soal 1. Implementasikan fungsi atau subroutin dalam Python untuk implementasi algoritma pada Gambar 18.7 untuk implementasi polinomial Newton. Uji hasil yang Anda dapatkan dengan menggunakan data-data yang diberikan pada contoh 18.2 (polinomial kuadrat) dan 18.3 (polinomial kubik) pada Chapra. Lengkapi jawaban Anda dengan membuat plot seperti pada Gambar 18.4 dan 18.6 pada Chapra.

Soal 2. Gunakan data pada soal Chapra 18.5 untuk mengevaluasi nilai $f(x) = \ln(x)$ pada $x = 2$ dengan menggunakan polinomial kubik. Coba variasikan titik-titik yang digunakan (*base points*) dan perhatikan nilai estimasi kesalahan yang diberikan oleh fungsi/subrutin yang sudah Anda buat pada soal sebelumnya.

2 Polinomial Interpolasi Lagrange

Soal 3. Implementasikan fungsi atau subroutin dalam Python untuk implementasi algoritma pada Gambar 18.11 untuk implementasi polinomial Lagrange. Uji hasil yang Anda dapatkan dengan menggunakan data-data yang diberikan pada contoh 18.2 (polinomial kuadrat) dan 18.3 (polinomial kubik) pada Chapra.

Soal 4. Kerjakan Contoh 18.7 pada Chapra dengan menggunakan fungsi/subrutin interpolasi Lagrange yang sudah Anda buat pada soal sebelumnya sehingga Anda dapat mereproduksi Gambar 18.12 pada Chapra. Pada Contoh 18.17 Anda diminta untuk mengestimasi kecepatan penerjun pada $t = 10$ s, yang berada di antara dua titik data terakhir. Untuk polinom orde-1, gunakan dua titik data terakhir, untuk orde-2 gunakan tiga titik data terakhir, dan seterusnya sampai orde-4.

3 Interpolasi Spline Kubik

Soal 5. Implementasikan fungsi atau subroutin dalam Python untuk implementasi algoritma pada Gambar 18.18 untuk implementasi spline kubik (natural). Uji hasil yang Anda dapatkan dengan menggunakan data-data yang diberikan pada contoh 18.10.

4 Soal Tambahan

Soal 6. Chapra Latihan 18.5

Diberikan data berikut:

x	1.6	2	2.5	3.2	4	4.5
$f(x)$	2	8	14	15	8	2

(a) Hitung $f(2.8)$ dengan menggunakan polinomial interpolasi Newton dengan orde 1 sampai 3.

(b) Gunakan Pers. (18.18) pada Chapra untuk mengestimasi kesalahan untuk setiap prediksi.

Soal 7. Chapra Latihan 18.11 Gunakan interpolasi invers dengan menggunakan polinomial interpolasi kubik dan metode bagi dua (*bisection*) untuk menentukan nilai x yang memenuhi $f(x) = 0.23$ untuk data dalam tabel berikut.

x	2	3	4	5	6	7
y	0.5	0.333	0.25	0.2	0.1667	1.1429

Soal 8. Chapra Latihan 18.26 Fungsi Runge dapat dinyatakan sebagai berikut:

$$f(x) = \frac{1}{1 + 25x^2}$$

(a) Buat plot dari fungsi tersebut interval dari $x = -1$ sampai $x = 1$.

(b) Buat polinomial interpolasi Lagrange orde 4 dengan menggunakan nilai fungsi yang disampel secara seragam: $x = -1, -0.5, 0, 0.5, 1$. Buat juga plot dari polinomial tersebut. Gunakan untuk menghitung nilai $f(0.8)$.

(c) Ulangi bagian sebelumnya dengan menggunakan polinom orde 5 sampai 10.

Soal 9. Chapra Latihan 18.27 Fungsi *humps* dapat ditulis sebagai berikut.

$$f(x) = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$

Hitung nilai fungsi ini pada titik-titik dalam interval x = 0 sampai x = 1 dengan jarak antar titik 0.1. Gunakan interpolasi spline kubik pada data yang Anda hasilkan dan buat plot dari fungsi *humps* dengan hasil interpolan spline yang Anda dapatkan.

Soal 10. Chapra Latihan 17.5 Gunakan regresi kuadrat terkecil untuk mencocokkan garis lurus ke data berikut.

x	6	7	11	15	17	21	23	29	29	37	39
x	29	21	29	14	21	15	7	7	13	0	3

Plot data dan garis lurus (persamaan linear) yang Anda dapatkan dalam satu plot. Hitung juga koefisien korelasi dan determinasi.

Soal 11. Chapra Latihan 17.6 Gunakan regresi kuadrat terkecil untuk mencocokkan polinomial orde-1 sampai dengan orde-8

x	1	2	3	4	5	6	7	8	9
x	1	1.5	2	3	4	5	8	10	13

Plot data dan polinomial yang Anda dapatkan, untuk tiap polinomial ada 1 plot. Hitung juga koefisien determinasi untuk masing-masing regresi yang Anda lakukan. Regresi orde berapa yang paling baik menurut Anda?

5 Kode Fortran

5.1 Interpolasi Newton

```
subroutine newton_interp(N, x, y, xi, yint, ea)
  implicit none
  integer :: N
  real(8) :: x(0:N)
  real(8) :: y(0:N)
  real(8) :: ea(0:N-1)
  real(8) :: yint(0:N)
  real(8) :: xi
  !
  real(8) :: fdd(0:N,0:N)
  integer :: i, j, order
  real(8) :: yint2, xterm

  do i = 0,n
    fdd(i,0) = y(i)
  enddo

  do j = 1,n
    do i = 0, n-j
      fdd(i,j) = ( fdd(i+1,j-1) - fdd(i,j-1) ) / ( x(i+j) - x(i) )
    enddo
  enddo

  xterm = 1.d0
  yint(0) = fdd(0,0)
  do order = 1,N
    xterm = xterm * ( xi - x(order-1) )
    yint2 = yint(order-1) + fdd(0,order)*xterm
    ea(order-1) = yint2 - yint(order-1)
    yint(order) = yint2
  enddo

  return
end subroutine
```

5.2 Contoh 18.2

```
program example
  implicit none
  integer :: N
  real(8) :: x(3), y(3)
  real(8) :: xi
```

```
  real(8), allocatable :: ea(:)
  real(8), allocatable :: yint(:)
  !
  real(8), allocatable :: fdd(:,:)

  x = (/ 1.d0, 4.d0, 6.d0 /)
  y = (/ 0.d0, 1.386294d0, 1.791759d0 /)

  N = 2
  xi = 2.d0

  allocate( ea(0:N-1) )
  allocate( yint(0:N) )

  call newton_interp(N, x, y, xi, yint, ea)

  write(*,*) 'yint = ', yint
  write(*,*) 'ea   = ', ea

  deallocate( ea )
  deallocate( yint )

end program
```

5.3 Contoh 18.3

```
program example
  implicit none
  integer :: N
  real(8) :: x(4), y(4)
  real(8) :: xi
  real(8), allocatable :: ea(:)
  real(8), allocatable :: yint(:)
  !
  real(8), allocatable :: fdd(:,:)

  x = (/ 1.d0, 4.d0, 6.d0 , 5.d0 /)
  y = (/ 0.d0, 1.386294d0, 1.791759d0, 1.609438d0 /)

  N = 3
  xi = 2.d0

  allocate( ea(0:N-1) )
  allocate( yint(0:N) )

  call newton_interp(N, x, y, xi, yint, ea)

  write(*,*) 'yint = ', yint
  write(*,*) 'ea   = ', ea

  write(*,*) abs( yint(N) - log(xi) )

  deallocate( ea )
  deallocate( yint )

end program
```

5.4 Interpolasi Lagrange

```
subroutine lagrange_interp(N, x, y, xx, res)
  implicit none
  integer :: N
  real(8) :: x(0:N), y(0:N)
  real(8) :: xx, res
```

```
  !
  real(8) :: ss, pp
  integer :: i, j

  ss = 0.d0
  do i = 0,N
    pp = y(i)
    do j = 0,N
      if( i /= j ) then
        pp = pp*( xx - x(j) ) / ( x(i) - x(j) )
      endif
    enddo
    ss = ss + pp
  enddo
  res = ss
  return
end subroutine
```

5.5 Interpolasi Spline Kubik Natural

```
subroutine interp_nat_cubic_spline( N, x, y, d2x, xu, yu, dy, d2y )
  !
  implicit none
  !
  integer :: N
  real(8) :: x(0:N), y(0:N)
  real(8) :: d2x(0:N)
  real(8) :: xu
  real(8) :: yu
  real(8) :: dy, d2y
  !
  integer :: i
  logical :: flag, is_in_interval
  real(8) :: c1, c2, c3, c4, t1, t2, t3, t4

  !write(*,*) 'size d2x = ', size(d2x)
  !write(*,*) 'd2x = ', d2x

  flag = .false.
  i = 1
  do while(.true.)
    !
    is_in_interval = (xu >= x(i-1)) .and. (xu <= x(i))
    !
    if( is_in_interval ) then
      !write(*,*) 'interval = ', i
      !
      c1 = d2x(i-1)/6.d0/( x(i) - x(i-1) )
      c2 = d2x(i)/6.d0/( x(i) - x(i-1) )
      c3 = y(i-1)/(x(i) - x(i-1)) - d2x(i-1)*(x(i) - x(i-1))/6.d0
      c4 = y(i)/(x(i) - x(i-1)) - d2x(i)*(x(i) - x(i-1))/6.d0
      !
      t1 = c1*( x(i) - xu )**3
      t2 = c2*( xu - x(i-1) )**3
      t3 = c3*( x(i) - xu )
      t4 = c4*( xu - x(i-1) )
      !
      yu = t1 + t2 + t3 + t4
      !
      t1 = -3.d0*c1*( x(i) - xu )**2
      t2 = 3.d0*c2*( xu - x(i-1) )**2
      t3 = -c3
      t4 = c4
      !
      dy = t1 + t2 + t3 + t4
```

```
      !
      t1 = 6.d0*c1*( x(i) - xu )
      t2 = 6.d0*c2*( xu - x(i-1) )
      !
      d2y = t1 + t2
      !
      flag = .true.
      !
    else
      !
      i = i + 1
    endif
    !
    if( i == N + 1 .or. flag ) then
      exit ! break out of the loop
    endif
  enddo

  if( .not. flag ) then
    write(*,*) "ERROR: xu is outside range of spline"
    return
  endif

  return
end subroutine


subroutine decomp_trid(N, e, f, g)
  implicit none
  integer :: N
  real(8) :: e(N), f(N), g(N)
  integer :: k
  do k = 2,N
    e(k) = e(k)/f(k-1)
    f(k) = f(k) - e(k)*g(k-1)
  enddo
  return
end subroutine

! should be called after calling decomp_trid
subroutine subs_trid(N, e, f, g, r, x)
  implicit none
  integer :: N
  real(8) :: e(N), f(N), g(N), r(N)
  real(8) :: x(N) ! output
  integer :: k

  ! Forward subs
  do k = 2,N
    r(k) = r(k) - e(k)*r(k-1)
  enddo

  ! back subs
  x(N) = r(N)/f(N)
  do k = N-1,1,-1
    x(k) = ( r(k) - g(k)*x(k+1) ) / f(k)
  enddo
  return
end subroutine

subroutine gen_trid_matrix(N, x, y, e, f, g, r)
  implicit none
  integer :: N
  real(8) :: x(0:N), y(0:N)
  real(8) :: e(N-1), f(N-1), g(N-1), r(N-1)
```

```
  integer :: i

  f(1) = 2.d0*( x(2) - x(0) )
  g(1) = x(2) - x(1)
  r(1) = 6.d0/( x(2) - x(1) ) * ( y(2) - y(1) )
  r(1) = r(1) + 6.d0/( x(1) - x(0) ) * ( y(0) - y(1) )
  !
  do i = 2,N-2
    e(i) = x(i) - x(i-1)
    f(i) = 2.d0*( x(i+1) - x(i-1) )
    g(i) = x(i+1) - x(i)
    r(i) = 6.d0/( x(i+1) - x(i) ) * ( y(i+1) - y(i) )
    r(i) = r(i) + 6.d0/( x(i) - x(i-1) ) * ( y(i-1) - y(i) )
  enddo
  !
  e(n-1) = x(n-1) - x(n-2)
  f(n-1) = 2.d0*( x(n) - x(n-2) )
  r(n-1) = 6.d0/( x(n) - x(n-1) ) * ( y(n) - y(n-1) )
  r(n-1) = r(n-1) + 6.d0/( x(n-1) - x(n-2) ) * ( y(n-2) - y(n-1) )


  return
end subroutine
```