

## 6. 関数近似 : 補間と補外

### 6.1 ラグランジュ補間法

互いに異なる点  $x_1, x_2, \dots, x_n$  とそれらの点における関数値  $f(x_1), f(x_2), \dots, f(x_n)$  が与えられているとする。これらの  $n$  点を補間するたかだか  $n-1$  次の補間多項式  $F_n(x)$  は、ラグランジュ基底関数  $L_k^{(n-1)}(x)$  を用いて

$$(1) F_n(x) = \sum_{k=1}^n f(x_k) L_k^{(n-1)}(x)$$

$$L_k^{(n-1)}(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_1)(x_k-x_2)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

と書ける。これは、 $n$  次の多項式

$$\pi_n(x) = (x-x_1)(x-x_2)\cdots(x-x_n)$$

を導入すると

$$\pi_n'(x_k) = (x_k-x_1)(x_k-x_2)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)$$

であるので、(1) 式は  $\pi_n(x)$  がくり出された形できつぎのようにも表すことができる。

$$(2) F_n(x) = \pi_n(x) \sum_{k=1}^n \frac{f(x_k)}{(x-x_k)\pi_n'(x_k)}$$

#### 【アルゴリズム 6.1.1】

(1) 式に基づくもの :

$(x_1, f_1), (x_2, f_2), \dots, (x_n, f_n) :=$ 与えられた  $n$  点のデータ

```

fx=0
for k=1 to n
  p=q=1
  for i=1 to n
    if i≠k then
      p:=p*(x-xi)
      q:=q*(xk-xi)
    end
  fx:=fx+fk*p/q
end
end
    
```

#### 【プログラム 6.1.1】

```

1 /* Lagrangian interpolation */
2 /* given data: n values of (x, f) */
3 /* interpolation point xx */
4 /* interpolated value f(xx) is computed */
5
6 #include <stdio.h>
7
8 #define NMAX 20
9
    
```

```

10 int n;
11 float x[NMAX], f[NMAX];
12
13 void main(void)
14 {
15     int i,k;
16     float xx, fx, p, q;
17
18     /* read input data */
19     printf("Lagrangian interpolation\n");
20     printf("number of data n ? ");
21     scanf("%d", &n);
22     for(i=0; i<n; i++) {
23         printf("x[%d], f[%d] ? ", i+1, i+1);
24         scanf("%g%g", &x[i], &f[i]);
25     }
26     printf("interpolation point xx ? ");
27     scanf("%g", &xx);
28
29     /* interpolation */
30     fx=0;
31     for(k=0; k<n; k++) {
32         p=1; q=1;
33         for(i=0; i<n; i++)
34             if(i!=k) {
35                 p *= xx-x[i];
36                 q *= x[k]-x[i];
37             }
38         fx += f[k]*p/q;
39     }
40     printf("\ninterpolated value at x=%g:\n", xx);
41     printf("f(%g)=%g\n", xx, fx);
42 }

```

### 【アルゴリズム 6.1.2】

(2) 式に基づくもの :

$(x_1, f_1), (x_2, f_2), \dots, (x_n, f_n)$  := 与えられた  $n$  点のデータ

```

for k=1 to n
    q=1
    for i=1 to n
        if i≠k then
            q:=q(xk-xi)
        dk:=q
    end
end

p=1, sum=0
for k=1 to n
    p :=p(x-xk)
    σ :=σ+fk/((x-xk) dk)
end
fx:=pσ

```

**【プログラム 6.1.2】**

```

1 /* Lagrangian interpolation          */
2 /* given data: n values of (x, f)  */
3 /* interpolation point xx           */
4 /* interpolated value f(xx) is computed */
5
6 #include <stdio.h>
7
8 #define NMAX 20
9
10 int n;
11 float x[NMAX], f[NMAX];
12
13 void main(void)
14 {
15     int i,k;
16     float xx, fx, p, q, sum, pd[NMAX];
17
18     printf("Lagrangian interpolation\n");
19     printf("number of data n ? ");
20     scanf("%d", &n);
21     for(i=0; i<n; i++) {
22         printf("x[%d], f[%d] ? ", i+1, i+1);
23         scanf("%g%g", &x[i], &f[i]);
24     }
25     printf("interpolation point xx ? ");
26     scanf("%g", &xx);
27
28     for(k=0; k<n; k++) {
29         q=1;
30         for(i=0; i<n; i++)
31             if(i!=k) q *= x[k]-x[i];
32         pd[k]=q;
33     }
34
35     p=1; sum=0;
36     for(k=0; k<n; k++) {
37         p *= xx-x[k];
38         sum += f[k]/((xx-x[k])*pd[k]);
39     }
40     fx=p*sum;
41     printf("\ninterpolated value at x=%g:\n", xx);
42     printf("f(%g)=%g\n", xx, fx);
43 }

```

**【問題】**

水の動粘性係数  $\nu$  は、常温の範囲では以下のとおりである。

T [°C]	$\nu$ [m <sup>2</sup> /s]
10.0	1.30700E-06
15.0	1.13900E-06
20.0	1.00400E-06
25.0	8.92800E-07
30.0	8.00800E-07

ラグランジュ補間により、水温 18.2°C のときの動粘性係数を求めよ。

**【実行例】**

プログラム 6.1.1, プログラム 6.1.2 とともに実行結果は以下のようになる。

```

-----実行開始-----
Lagrangian interpolation
number of data n ? 5
x[1], f[1] ? 10.0 1.307e-6
x[2], f[2] ? 15.0 1.139e-6
x[3], f[3] ? 20.0 1.004e-6
x[4], f[4] ? 25.0 8.928e-7
x[5], f[5] ? 30.0 8.008e-7
interpolation point xx ? 18.2

interpolated value at x=18.2:
f(18.2)=1.04948e-06
-----おしまい-----
    
```

**6.2 スプライン補間法**

互いに異なる点  $x_0 < x_1 < \dots < x_n$  とそれらの点における関数値  $y_0=f(x_0), y_1=f(x_1), \dots, y_n=f(x_n)$  が与えられているとする。小区間  $[x_{i-1}, x_i]$  ごとに 3 次エルミート補間関数 (1 階導関数の連続性を保証)  $s_i(x)$  を用いると、

$$s_i(x) = \left(\frac{x-x_i}{h_i}\right)^2 \left( (3y_{i-1} + h_i y'_{i-1}) + \frac{x-x_i}{h_i} (2y_{i-1} + h_i y'_{i-1}) \right) + \left(\frac{x-x_{i-1}}{h_i}\right)^2 \left( (3y_i + h_i y'_i) - \frac{x-x_{i-1}}{h_i} (2y_i - h_i y'_i) \right)$$

$$h_i = x_i - x_{i-1}$$

となる。更に隣接する小区間上の補間関数と 2 階導関数まで連続に接続する条件  $s_i''(x_i) = s_{i+1}''(x_i)$  を課すると、 $y_i'$  に関する方程式

$$\frac{1}{h_i} y'_{i-1} + \left(\frac{2}{h_i} + \frac{2}{h_{i+1}}\right) y'_i + \frac{1}{h_{i+1}} y'_{i+1} = -\frac{3}{h_i^2} y_{i-1} + \left(\frac{3}{h_i^2} - \frac{3}{h_{i+1}^2}\right) y_i + \frac{3}{h_{i+1}^2} y_{i+1}, \quad i = 1, 2, \dots, n-1$$

を得る。端点  $x=x_0, x_n$  においては、2 階微係数が零となる条件を課すると、

$$\frac{2}{h_1} y'_0 + \frac{1}{h_1} y'_1 = -\frac{3}{h_1^2} y_0 + \frac{3}{h_1^2} y_1$$

$$\frac{1}{h_n} y'_{n-1} + \frac{2}{h_n} y'_n = -\frac{3}{h_n^2} y_{n-1} + \frac{3}{h_n^2} y_n$$

となる。これらは、 $y_i'$  に関する三項方程式となっているので、アルゴリズム 4.3.1 により  $y_i'$  は求まる。点  $x$  がどの区間にあるかを判定すれば、 $x$  におけるスプライン補間関数  $s_i(x)$  の値は定まる。

**【プログラム 6.2.1】**

```

1 /* Spline Interpolation */
2 /* given data: number of intervals: n */
    
```

```

3 /*          x_i, y_i, i=0,1,...,n */
4 /*  interpolated value f(xx) is computed */
5 /*          for x_0 <= xx <= x_n */
6
7 #include <stdio.h>
8 #include <math.h>
9
10 #define NMAX 50
11
12 float yy(float xx, float x0, float x1, float y0, float y1, float dy0, float dy1)
13 {
14     float z0, z1;
15
16     z0=(xx-x0)/(x1-x0);  z1=(xx-x1)/(x0-x1);
17     return(z1*z1*((3*y0-(x0-x1)*dy0)-z1*(2*y0-(x0-x1)*dy0))
18         +z0*z0*((3*y1-(x1-x0)*dy1)-z0*(2*y1-(x1-x0)*dy1)));
19 }
20
21 void main(void)
22 {
23     int n, i, j, m;
24     float x[NMAX], y[NMAX], h[NMAX], a[NMAX], b[NMAX], c[NMAX], d[NMAX];
25     float x0, dx, xx, yyj;
26
27                                     /* read input data */
28     printf("Spline interpolation\n");
29     printf("number of intervals n ? ");
30     scanf("%d", &n);
31     for (i=0; i<=n; i++) {
32         printf("x[%d], y[%d] ? ", i, i);
33         scanf("%g%g", &x[i], &y[i]);
34
35                                     /* set coefficients of three-term equations */
36     for (i=1; i<=n; i++)
37         h[i]=x[i]-x[i-1];
38     for (i=1; i<=n; i++) {
39         a[i]=2/h[i]+2/h[i+1];
40         b[i]=1/h[i+1];
41         c[i]=1/h[i];
42         d[i]=3/(h[i]*h[i])*(y[i]-y[i-1])
43             +3/(h[i+1]*h[i+1])*(y[i+1]-y[i]);
44
45                                     /* end-point condition at x[0] */
46     a[0]=2/h[1]; b[0]=1/h[1]; c[0]=0; d[0]=3/(h[1]*h[1])*(y[1]-y[0]); /* y''=0 */
47                                     /* end-point condition at x[n] */
48     a[n]=2/h[n]; b[n]=0; c[n]=1/h[n]; d[n]=3/(h[n]*h[n])*(y[n]-y[n-1]); /* y''=0 */
49 /*  a[n]=1; b[n]=0; c[n]=0; d[n]=(y[n]-y[n-1])/h[n]; */
50
51                                     /* solve three-term equations */
52                                     /* forward */
53     b[0] /= a[0];
54     d[0] /= a[0];
55     for (i=1; i<=n; i++) {

```

```

54     a[i] -= c[i]*b[i-1];
55     d[i] -= c[i]*d[i-1];
56     d[i] /= a[i];
57     b[i] /= a[i];
58 }
59                                     /* backward */
60 for(i=n-1;i>=0;i--)
61     d[i] -=b[i]*d[i+1];
62                                     /* interpolated values */
63 printf("initial value and interval of x for interpolated results, x0, dx ? ");
64 scanf("%g%g", &x0,&dx);
65 printf("\n\tx\t interpolation\n");
66 m=(int)floor((x[n]-x0)/dx + 0.5);
67 i=1;                               /* initial set for interval counter */
68 for(j=0;j<=m;j++){
69     xx=x0+j*dx;
70 label0:if(xx<x[i] || i==n){
71     yyj= yy(xx, x[i-1], x[i], y[i-1], y[i], d[i-1], d[i]);
72     printf("%15.6e %15.6e\n", xx, yyj);
73     if(xx>=(x[n]+0.1*dx) && i==n) break;
74 }else{
75     i++;
76     goto label0;
77 }
78 }
79 }

```

**【問題】**

水の動粘性係数 $\nu$ の常温での温度依存性は 6.1 節「ラグランジュ補間法」の問題で記したとおりである。スプライン補間により、水温が 10~30[°C]のときの動粘性係数を求め、グラフ表示せよ。

**【実行例】**

```

-----実行開始-----
Spline interpolation
number of intervals n ? 4           ..... nは小区間数であることに注意
x[0], y[0] ? 10.0 1.307e-6          (データ点数はn+1)
x[1], y[1] ? 15.0 1.139e-6
x[2], y[2] ? 20.0 1.004e-6
x[3], y[3] ? 25.0 8.928e-7
x[4], y[4] ? 30.0 8.008e-7
initial value and interval of x for interpolated results, x0, dx ? 10.0 0.2
..... 初期値10, 増分0.2のxの値に対する補間値を求める

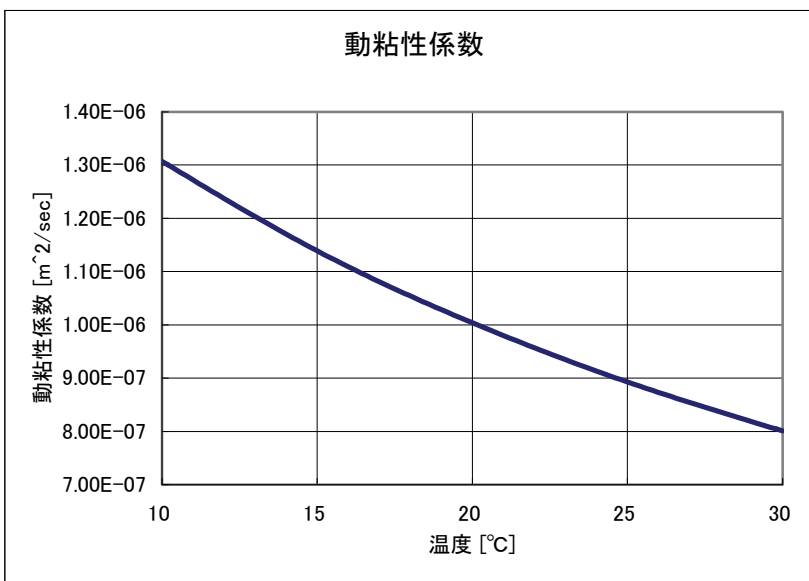
      x           interpolation
1.000000e+01    1.307000e-06
1.020000e+01    1.299981e-06
1.040000e+01    1.292965e-06
1.060000e+01    1.285955e-06
1.080000e+01    1.278954e-06
1.100000e+01    1.271963e-06
1.120000e+01    1.264988e-06

```

1. 140000e+01	1. 258029e-06
1. 160000e+01	1. 251091e-06
1. 180000e+01	1. 244176e-06
1. 200000e+01	1. 237286e-06
⋮	⋮
⋮	⋮
2. 900000e+01	8. 184259e-07
2. 920000e+01	8. 148914e-07
2. 940000e+01	8. 113631e-07
2. 960000e+01	8. 078395e-07
2. 980000e+01	8. 043190e-07
3. 000000e+01	8. 008000e-07

-----おしまい-----

動粘性係数の温度依存性はつぎのグラフに示される。



## 参考文献

- 高倉葉子：数値計算の基礎—解法と誤差—，コロナ社（2007）
- 森口繁一，伊理正夫，武市正人 編：C による算法痛論，東京大学出版会（2000）
- Heath, Michael T.: Scientific Computing, An Introductory Survey, McGraw-Hill（2002）