"cemmsharp", "msharpmath" Revised on 2012.12.06

[102] 005

제 5 장 보간과 곡선근사

www.msharpmath.com *
invite authors for improvement **

* Msharpmath Inc.
** author's address

수치해석 강좌 담당교수께서는 원본파일을 강의의 목적에 맞게 설명방법, 예제, 그림, 연습문제 등을 원하는 방향으로 수정하여 www.msharpmath.com 에 출간하시면 강의의 진행에 도움되실 것입니다. 학생들에게는 항상 무료로 다운로드 받을 수 있는 인터넷 교재를 제공하고, 담당교수께서는 언제든지 수정출간할 수 있는 장점을 가진 인터넷 출간에 부디 많은 학과에서 참여하시기를 부탁드립니다. 인터넷 출간은 Chapter 별로 이루어지므로 원하시는 분야만을 골라서 출간할 수 있습니다. 따로 각 담당강좌별로 저자 고유번호가 부여되어 쉽게 검색하고 다운받을 수 있습니다.

이 장에서는 불연속적으로 주어진 자료 (x_i, f_i)를 이용하여 독립변수의 값이 주어지지 않는 점에서의 함수값을 보간 (補間, interpolation)하는 방법과 전체 데이터를 가장 효과적으로 근사할 수 있는 곡선근사 (曲線近似, curve fitting)에 대해서 알아본다.

일반적으로 함수값의 보간은 주어진 자료 (x_i, f_i) 를 미리 선택한 기저함수 (基底函數, base function) ϕ_i의 선형결합

$$\sum_i w_i \phi_i$$

의 형태로 표현하는 것을 말하며, 이것은 곡선근사에 대해서도 동일하게 성립한다. 일반적으로 기저함수로는 다항식

$$1, x, x^2, \cdots, x^n$$

을 선택한다. 만일 선형결합이 아닌 다른 형태로 보간을 나타내는 경우에는 비선형 보간 또는 비선형 곡선근사 등으로 분류한다.

한편 독립변수 x의 분포가 등간격으로 주어지는 경우와 그렇지 않은 경우로 구분할 수 있는데 단순히 보간과 곡선근사를 구체적으로 결정하기 위한 목적이라면 어느 쪽도 그다지 문제가 되지 않는다. 그러나 보간과 곡선근사의 이론을 다른 수치해석 분야에 적용할 때는 수치적인 결과가 아니라 수식적인 결과가 보다 중요한 역할을 하게 된다. 이러한 이유로 이 장에서 공부하는 대부분의 내용은 수식적인 형태가 중요한 내용으로 된다. 만일 수치적인 결과가 중요하다면 cemm#에서 제공하는 도트 함수 .interp(x,y) 을 이용하면 대부분의 실용적인 결과를 얻는 데에는 부족함이 없을 것이다.

이 장에서 가장 먼저 공부하는 라그랑주 보간은 실용적인 측면보다는 이론적인 측면에서 중요한 위치를 차지하는 내용이다. 다음으로 공부하는 뉴턴의 전진/후진보간법은 등간격의 경우 나타나는 이론으로 앞으로 공부하게 될 수치적분 또는 미분방정식 분야에서 기초가 되기 때문에 중요하다. 그러나 비등간격의 경우에는 원래의 목적 그대로 보간과 곡선근사에서만 의미를 가진다.

한편 데이터 (x_i, f_i)가 미리 주어지지 않은 상태에서 알고 있는 함수 $f(x)$의 최적화된 보간을 구하고 싶을 경우에는 체비쉐프 보간을 이용하는 것이 가장 효과적이라는 사실을 5-8절에서 확인할 것이다. 이와 유사하게 독립변수 x의 주어진 점 x_i에서 함수값 뿐아니라 도함수값 $f'(x_i)$가 알려져 있는 경우에는 에르미트 보간이 가장 정확한 방법임을 5-9절에서 논의한다. 그리고 5-7절에서는 구간마다 각각의 다항식으로 보간하는 스플라인 보간에 대해서 알아보기로 한다. 주어진 데이터를 이용하여 곡선근사를 하는 방법에 대해서는 5-5절에서 논의할 것이다.

특히 데이터의 보간에서 나타나는 이항계수에 대한 연산은 부록에서 다룰 것이므로 여기서는 추가적인 논의를 생략하기로 한다.

```
//=========================================================================
// syntax in cemm#
//-------------------------------------------------------------------------
.dtable(x,f)          // forward divided-difference table
.interp(x,f)          // interpolating polynomial
.corrcoef(f1,f2)      // correlation coefficient
(m,n).dfdx(kth)       // finite difference for kth derivative
(m,n).dfdxm(kth)      // finite difference matrix
.spline(x,f)          // spline matrix with intervals
.spline1(x,f)         // 1st-order spline matrix with intervals
.polyfit(x,f,kth)     // regression by polynomial of kth degree
.regress(x,f,mftn)    // regression by matrix functions, return matrix
p %% c = p.syndiv(c)  // coefficients for synthetic division

// ascending-order polynomial
    p(x) = a0 + a1x + a2x^2 + a3x^3 + ...+ anx^n + ...
    poly(a0,a1,a2, ..., an);     // list of coefficients
    poly( [a0,a1,a2,...,an] );   // a single matrix
    poly( A );                   // A is a matrix

// descending-order polynomial
    .[an, ...,a2,a1,a0];   // general order, n >= 4
    .[ a       ]           // constant polynomial
    .[ a,b     ]           // 1st-order poly, line y = ax+b
    .[ a,b,c   ]    // 2nd-order poly, parabola y = ax^2 + bx + c
    .[ a,b,c,d ]    // 3rd-order poly, cubic y = ax^3 + bx^2 + cx +d
    poly(an, ...,a2,a1,a0).rev;
// 'poly' by known roots
    [ x1,x2, ... , xn ] .roots  // (x-x1)(x-x2) .. (x-xn), real x's only
    .roots(x1,x2, ... , xn)   // (x-x1)(x-x2) .. (x-xn)
    .roots( A )          // A is a matrix including complex numbers

// Newton polynomial
    p(x) = a0 + a1(x - x1) + a2(x - x1)(x - x2)
                           +a3(x - x1)(x - x2)(x - x3) + ...
    poly(a0,a1,a2,a3, ...) .newton( [x1,x2,x3, ...] )


// piecewise polynomials over a number of intervals
// x11<=x<=x12: p1(x)=a_10 + a_11 x + a_12 x^2 +...
// x21<=x<=x22: p2(x)=a_20 + a_21 x + a_22 x^2 +...
// ...
// xn1<=x<=xn2: pn(x)=a_n0 + a_n1 x + a_n2 x^2 + ...
// is stacked as a spline matrix in which
```

```
//              the first two columns are intervals
//       [ x11, x12, a10 a11 a12 ... ]
//  A =  [ x21, x22, a20 a21 a22 ... ]
//       [  ...                      ]
//       [ xn1, xn2, an0 an1 an2 ... ]
    A.spl(x)     // evaluation for double
    A.spl(B)     // evaluation for matrix
    A.spldiff    // differentiate piecewise polynomials
    A.splint     // integrate piecewise polynomials
    A.splplot ;  // plot piecewise polynomials
```

5-1 기본 개념

■ 이산 데이터. 공학적인 많은 응용문제는 알려진 함수로부터 추출하거나 실험적으로 결정된 이산 데이터 (discrete data)를 처리하게 된다. 하나의 독립변수 x 인 경우로 국한하여 생각하면, 이산 데이터의 대표적인 형태는

독립변수	x_0	x_1	x_2	...	x_i	...	x_{n-1}	x_n
종속변수	f_0	f_1	f_2	...	f_i	...	f_{n-1}	f_n

으로 주어진다. 위에서 $f_i = f(x_i)$ 는 어떤 함수 $f(x)$ 로부터 결정되며 총 $n+1$ 개의 데이터가 $(x_i, f_i),\ i = 0, 1, \ldots, n$ 으로 주어지는 경우에 대해서 고려한다. 특별한 언급이 없으면 데이터의 격자점 x_i 는 일반적으로 크기가 증가하는 순서로 주어진 것으로 간주한다. 그리고 $h_i = x_i - x_{i-1}$ 는 두 개의 연속된 격자점 사이의 격자크기로 정의한다. 만일 주어진 데이터가 균일한 격자크기로 배열된 경우에는 등간격 격자라고 한다.

■ 보간과 곡선근사. 이산 데이터가 위에서 언급한 바와 같이 주어지면 $x \neq x_i$ 인 위치에서의 함수값은 당연히 미지값으로 된다. 이 장의 주 목적은 임의의 x 점에 대한 함수값 $f(x)$ 에 대한 근사값을 결정하는 것이다. 이러한 의미에서 그림 5-1에는 보간 (補間, interpolation)과 곡선근사 (曲線近似, curve-fitting)의 두 가지 경우를 도시하였다.

그림 5-1에 보인 바와 같이 주어진 데이터 점을 모두 지나는 경우를 보간, 그렇지 않은 경우를 곡선근사라고 흔히 구별한다. 이때 보간 또는 곡선근사는 대부분의 경우 미리 선택한 기저함수 (base function) ϕ_i 에 대한 일차결합 (linear combination)

$$f(x) = \sum_i w_i \phi_i \tag{1}$$

을 주로 이용한다. 위에서 기저함수로 x 의 거듭제곱 (즉 $\phi_i = x^i$)을 사용하는 경우에는 다항식보간 (polynomial interpolation)이라고 한다. 뒤에서 공부하겠지만 라그랑주 보간, 뉴턴 전진/후진 보간, 체비쉐프 보간, 에르미트 보간, 스플라인 보간 등이 다항식 보간에 속한다. 마찬가지로 곡선근사의 경우에도 다항식이 가장 많이 사용된다.

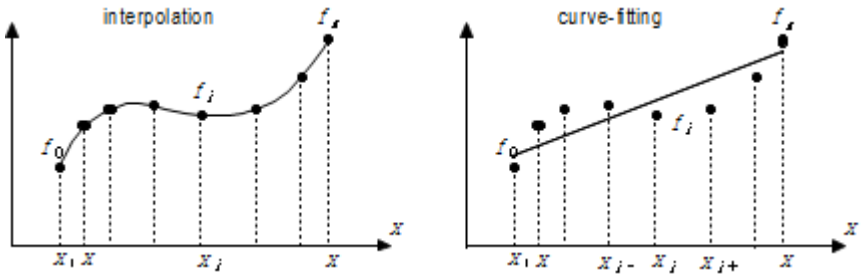


그림 5-1 보간과 곡선근사

■ 다항식과 테일러급수. 다항식은 보간과 곡선근사 모두에서 중요하기 때문에 추후의 논의를 위하여 간단히 다항식에 대해서 정리한다. n 차 다항식의 표준형은

$$P_n(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n \tag{2}$$

으로 나타내고, 계수 a_n 은 선행계수 (先行係數, leading coefficient)라고 한다. 특히 $a_n = 1$ 인 다항식은 일원다항식 (一元多項式, monic polynomial)으로 부른다.

주어진 $n+1$개의 데이터에 대해서 다항식보간은 다음의 선형방정식

$$\begin{aligned} P_n(x_0) &= a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n = f_0 \\ P_n(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n = f_1 \\ &\cdots \\ P_n(x_n) &= a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n = f_n \end{aligned} \tag{3}$$

을 풀어 미지의 계수 a_i를 구할 수 있다. 그러나, 이러한 접근방법은 심각한 마무리오차의 영향으로 실제로는 사용되지 않는다 (x_i의 거듭제곱은 i가 커질수록 아주 큰 수가 되거나 아주 작은 수로 된다). 이러한 이유로 다항식보간은 다음에서 설명하는 방법을 이용한다.

다항식에 관하여 한 가지 주목할 내용은 총 $n+1$개의 데이터 점을 지나는 n차 다항식은 겉보기의 표현에 무관하게 유일하게 주어진다는 유일성 정리 (uniqueness theorem)이다. 이러한 성질은 다항식보간에 관련된 오차해석의 통합된 접근방법을 가능하게 한다.

점 $x = x_i$에 관한 테일러급수는

$$f(x) = f(x_i) + (x - x_i)f'(x_i) + (x - x_i)^2 \frac{f''(x_i)}{2!} + \cdots + (x - x_i)^n \frac{f^{(n)}(x_i)}{n!} + R_n \tag{4}$$

$$R_n = (x - x_i)^{n+1} \frac{f^{(n+1)}(\xi)}{(n+1)!} \tag{5}$$

으로 쓸 수 있으며, 위에서 ξ는 x와 x_i의 사이에 위치하는 값이다. 나머지 항 R_n은 다항식보간의 오차해석에서 나중에 중요한 역할을 할 것이다. 그리고 뒤에서는 보간다항식과 테일러급수 사이의 유사성에 대해서 논의한다.

■ 부분구간 다항식. 구간별로 각각 정의된 다항식을 부분구간 다항식 (Piecewise continuous polynomials)이라 하고, 각 구간과 대응하는 다항식으로 다음과 같이 정의된다.

$$\begin{aligned} x_{11} \le x \le x_{12} &: p_1(x) = a_{10} + a_{11}x + a_{12}x^2 + \ \ldots + a_{1n}x^n + \ \ldots \\ x_{21} \le x \le x_{22} &: p_2(x) = a_{20} + a_{21}x + a_{22}x^2 + \ \ldots + a_{2n}x^n + \ \ldots \\ &\vdots \\ x_{m1} \le x \le x_{m2} &: p_m(x) = a_{m0} + a_{m1}x + a_{m2}x^2 + \ \ldots + a_{mn}x^n + \ \ldots \end{aligned}$$

cemm#에서는 스플라인 행렬 (spline matrix)을 정의하고 부분구간 다항식을 처리한다. 처음 두 열은 구간의 시작과 끝을 나타내고 나머지 행에 오름차순으로 다항식을 나타낸다.

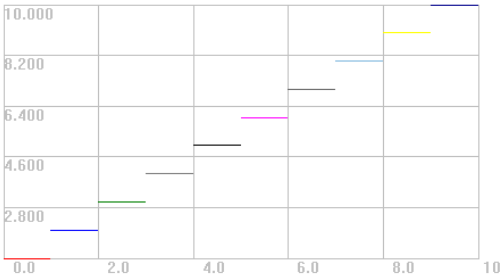
$$\mathbf{A} = \begin{bmatrix} x_{11} & x_{12} & a_{10} & a_{11} & \cdots \\ x_{21} & x_{22} & a_{20} & a_{21} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & a_{m0} & a_{m1} & \cdots \end{bmatrix}$$

이러한 스플라인 행렬에 대해서 함수값 (실수 및 행렬)과 미분, 적분, 그림 등을 처리하는 행렬함수가 다음과 같이 정의된다.

```
A.spl(x)     // evaluation for double
A.spl(B)     // evaluation for matrix
A.spldiff    // differentiate piecewise polynomials
A.splint     // integrate piecewise polynomials
A.splplot ;  // plot piecewise polynomials
```

예를 들어 계단함수는 다음과 같이 그린다.

```
#> x = (0:9).tr;
#> [ x, x+1, x+1 ].splplot;   // spline plot
```

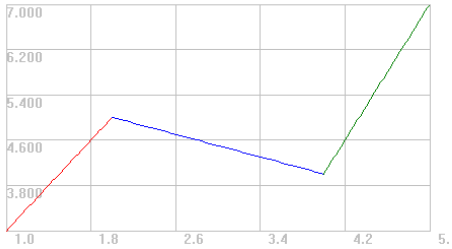


이산데이터 $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ 는 흔히 두 점을 직선으로 연결하여 처리하며 대표적인 부분구간 다항식으로 된다. 아래에는 함수값, 미분, 적분 등의 계산과 그림을 그리는 예를 보인다.

```
#> x = [ 1, 2, 4, 5 ];;
#> y = [ 3, 5, 4, 7 ];;
#> P = .spline1(x,y);  // spline1 for the 1st-order splines
    P =
[           1           2           1           2 ]
[           2           4           6        -0.5 ]
[           4           5          -8           3 ]
```

$1 \le x \le 2 : p_1(x) = 1 + 2x$
$2 \le x \le 4 : p_2(x) = 6 - 0.5x$
$4 \le x \le 5 : p_3(x) = -8 + 3x$

```
#> P.splplot;    // plot piecewise polynomials
```




```
#> P.spl(2.5);                  // p2(2.5) = 6 - 0.5(2.5)
    ans =        4.75

#> P.spl( [ 2.5, 3, 4.5 ] );  // evaluation by matrix
    ans =  [          4.75          4.5          5.5 ]

#> Q = P.splint;
Q =
 [              1              2             -2              1              1 ]
 [              2              4             -7              6          -0.25 ]
 [              4              5             21             -8            1.5 ]
```

$$1 \le x \le 2 : q_1(x) = \int_1^x p_1(z)dz = \int_1^x 1 + 2z\, dz = -2 + x + x^2$$

$$2 \le x \le 4 : q_2(x) = \int_2^x p_2(z)dz + q_1(2) = -7 + 6x - \frac{1}{4}x^2$$

$$4 \le x \le 5 : q_3(x) = \int_4^x p_3(z)dz + q_2(4) = 21 - 8x + \frac{3}{2}x^2$$

위의 결과에서 ".splint"의 결과는 아래의 구간 적분을 의미한다.

```
#> Q.spl(4.5) - Q.spl(2.5);
    ans =          8.9375
```

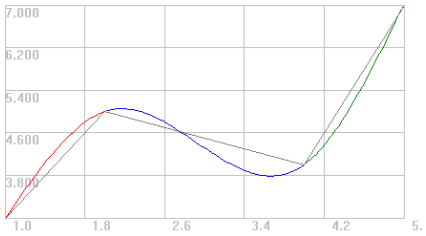
$$\int_{2.5}^{4} p_2(x)dx + \int_{4}^{4.5} p_3(x)\,dx = \left[-7 + 6x - \frac{1}{4}x^2\right]_{2.5}^{4} + \left[21 - 8x + \frac{3}{2}x^2\right]_{4}^{4.5}$$
$$= 8.9375$$

3 차다항식을 이용한 부분구간다항식 (스플라인)도 역시 가능하다.

```
#> x = [ 1, 2, 4, 5 ];;
#> y = [ 3, 5, 4, 7 ];;
#> P = .spline(x,y);  // 1st and 2nd columns for interval
    P =
 [       1       2           1       0.625      2.0625     -0.6875 ]
 [       2       4       -10.5      17.875     -6.5625        0.75 ]
 [       4       5        89.5     -57.125     12.1875     -0.8125 ]
```

$$1 \le x \le 2 : p_1(x) = 1 + 0.625x + 2.0625x^2 - 0.6875x^3$$
$$2 \le x \le 4 : p_2(x) = -10.5 + 17.875x - 6.5625x^2 + 0.75x^3$$
$$4 \le x \le 5 : p_3(x) = 89.5 - 57.125x + 12.1875x^2 - 0.8125x^3$$

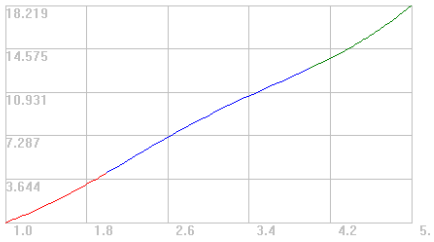
```
#> P.splplot;   // .spline(x,y).splplot;
#> [ x', y' ].plot+ ;
```



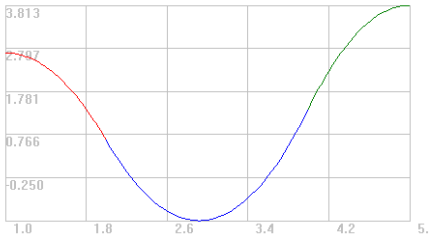

```
#> P.splint.spl(4.5) - P.splint.spl(2.5) ;
    ans =       8.5068359
```

$$\int_{2.5}^{4} p_2(x)dx + \int_{4}^{4.5} p_3(x)\,dx$$
$$= \int_{2.5}^{4} -10.5 + 17.875x - 6.5625x^2 + 0.75x^3 dx$$
$$+ \int_{4}^{4.5} 89.5 - 57.125x + 12.1875x^2 - 0.8125x^3\,dx$$
$$= 8.5068359$$

```
#> P.splint.splplot;  // .spline(x,y) .splint .splplot;  integrate
```




```
#> P.spldiff.splplot;  // .spline(x,y) .spldiff .splplot;  differentiate
```



5-2 라그랑주 보간

이 절에서는 독립변수 x에 대한 함수 $f(x)$의 불연속적인 값이 데이터 $(x_i, f_i),\ i = 0, 1, \ldots, n$ 으로 주어져 있을 때 데이터를 다항식으로 보간하는 라그랑주 보간에 대해서 알아본다.

■ 선형보간. 두 개의 데이터가

$$(x_0, f_0), \qquad (x_1, f_1)$$

으로 주어져 있을 때 선형보간 (linear interpolation)은

$$P_1(x) = ax + b \tag{6}$$

인 1차식을 이용하여 보간한다. 주어진 데이터를 윗식에 대입하면

$$P_1(x_0) = ax_0 + b = f_0,\ P_1(x_1) = ax_1 + b = f_1$$

이고, 상수 a, b 를 구하면

$$P_1(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0) \tag{7}$$

을 얻는다. 일반화를 위하여 윗식을 정리하면

$$P_1(x) = f_0\left[1 - \frac{x - x_0}{x_1 - x_0}\right] + \frac{f_1}{x_1 - x_0}(x - x_0)$$

즉

$$P_1(x) = f_0\frac{x - x_1}{x_0 - x_1} + f_1\frac{x - x_0}{x_1 - x_0} \tag{8}$$

와 같이 1차식보간이 결정된다. 그러나 다항식보간의 유일성정리에 의해서 식(7)과 식(8)은 표현만 다를 뿐 동일한 보간을 의미한다.

만일 주어진 데이터가 알려진 함수 $f(x)$ 로부터 구한 것이라면 선형보간의 오차 $e(x)$는

$$\begin{aligned} f(x) &= P_1(x) + e(x) \quad \Rightarrow \\ f(x) &= f_0 + (x - x_0)\frac{f_1 - f_0}{x_1 - x_0} + e(x) \end{aligned} \tag{9}$$

으로부터 정의할 수 있다. 이때 오차해석을 위해서는 식(8) 보다는 식(7)을 이용하는 것이 훨씬 편리하다. 한편 오차 $e(x)$ 는 두 점 $x = x_0$ 와 $x = x_1$에서 정확히 0으로 되므로 식(9)는

$$f(x) = f_0 + (x - x_0)\frac{f_1 - f_0}{x_1 - x_0} + (x - x_0)(x - x_1)\eta(x)$$

와 같이 표현할 수 있다. 위에서 $\eta(x)$는 다시

$$\eta = \frac{f''(\xi)}{2!}, \qquad x_0 \le \xi \le x_1 \tag{10}$$

인 형태로 나타낼 수 있으므로 (증명은 생략)

선형보간:

$$f(x) = f_0 + (x - x_0)\frac{f_1 - f_0}{x_1 - x_0} + (x - x_0)(x - x_1)\frac{f''(\xi)}{2!} \tag{11}$$

으로 나타낼 수 있다. 위에서 마지막 항은 선형보간과 연관된 오차를 의미한다.

이때 식(4)에 주어진 테일러급수를 아래와 같이 다시 쓰고

테일러급수:

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + (x - x_0)(x - x_0)\frac{f''(\xi)}{2!} \tag{12}$$

식(11)과 비교하면 흥미로운 유사성을 찾을 수 있으며 이러한 유사성은 뒤에서 다시 논의하기로 한다.

■ 라그랑주 보간. 위의 식(8)에서

$$L_0(x) = \frac{x - x_1}{x_0 - x_1}, \qquad L_1(x) = \frac{x - x_0}{x_1 - x_0} \tag{13}$$

와 같이 정의하면 이 다항식은

$$\begin{aligned} L_0(x_0) &= 1, \quad L_0(x_1) = 0 \\ L_1(x_0) &= 0, \quad L_1(x_1) = 1 \end{aligned} \tag{14}$$

인 성질을 가진다. 이러한 다항식을 라그랑주 정규다항식 (Lagrange normalized polynomial)이라고 하며, 이것을 이용하여 데이터를 보간하는 방법을 라그랑주 보간 (Lagrange interpolation)이라고 한다.

일반적으로 $n+1$ 개의 점 $x = x_0, x_1, x_2, \dots, x_n$ 에서 데이터 $f_0, f_1, f_2, \dots, f_n$ 이 주어질 때 라그랑주 정규다항식은

$$L_i(x) = \frac{(x - x_0)(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n)}{(x_i - x_0)(x_i - x_1)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)} \tag{15}$$

으로 정의되며

$$L_i(x_j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{16}$$

인 성질을 가진다. 이러한 성질을 이용하면 라그랑주 보간은

$$P_n(x) = f_0 L_0(x) + f_1 L_1(x) + \cdots + f_n L_n(x) \tag{17}$$

으로 된다. 위의 보간은

$$P_n(x_i) = \cdots + f_{i-1} L_{i-1}(x_i) + f_i L_i(x_i) + f_{i+1} L_{i+1}(x_i) \ldots = f_i$$

으로부터 알 수 있듯이 $x = x_i$ 에서 정확히 $f_i = f(x_i)$ 인 값을 가지므로 주어진 데이터의 모든 점을 지나며, $L_i(x)$ 의 차수가 n 이므로 n 차 보간다항식이다.

라그랑주 보간은 주어진 독립변수의 간격 $x_i - x_{i-1}$ 의 크기에 관계가 없으며 또한 x_i 의 크기 순서에도 무관한 장점을 가진다. 그러나, 라그랑주 보간은 주어진 x에 대해서 모든 $L_i(x)$의 값을 계산해야 하기 때문에 시간이 많이 걸리는 단점으로 보간의 목적으로는 사용되지 않고 수치미분 등의 응용분야에서 기초적인 방법으로서 사용된다.

■ 라그랑주 보간의 오차. 함수 $f(x)$ 로부터 얻어진 데이터 점들 (x_i, f_i)의 라그랑주 보간에서의 오차 $e(x) = f(x) - P_n(x)$는

$$e(x) = \frac{(x - x_0)(x - x_1) \ldots (x - x_n)}{(n+1)!} f^{(n+1)}(\xi), \quad x_0 \leq \xi \leq x_n \tag{18}$$

으로 주어진다.

위의 결과로부터 함수 $f(x)$ 가 n 차 이하의 다항식이라면 $(n+1)$ 계도함수는 0이므로 오차는 0이 된다. 라그랑주 보간의 오차에 영향을 주는 것은 보간영역의 구간크기 $x_n - x_0$, 데이터 점의 간격 $x_i - x_{i-1}$, 데이터의 개수 $n+1$ 등이다. 보통 n 의 값이 커지면 최대오차 $|e(x)|$은 감소하지만 경우에 따라서는 증가할 수도 있다. 라그랑주 보간에는 다음과 같은 문제점이 있다.

- 단일보간에 드는 계산량이 많다.
- 주어진 점 x 를 바꾸는 경우 이전의 계산결과는 사용되지 않는다.
- 데이터를 추가 또는 감소할 때마다 모든 계산을 새로 반복해야 한다.

• 오차의 평가가 쉽지 않다.

(예제 1) $f(x)=\sin(x)$에 대한 데이터가

x	0.5	1.0	1.5
$f(x)$	0.479526	0.841471	0.997495

으로 주어질 때 라그랑주 보간을 이용하여 $x=1.3$에서의 값을 구하라.

식(15)와 식(17)을 이용하여 계산하면 되며, $n=2$ 인 경우이므로

$$P_2(x)=0.479526\frac{(x-1.0)(x-1.5)}{(0.5-1.0)(0.5-1.5)}+0.841471\frac{(x-0.5)(x-1.5)}{(1.0-0.5)(1.0-1.5)} +0.997495\frac{(x-0.5)(x-1.0)}{(1.5-0.5)(1.5-1.0)} \tag{19}$$

와 같이 2차 보간다항식이 주어진다. 따라서 $x=1.3$ 에서의 보간값은

$$P_2(1.3)=0.479526\frac{(1.3-1.0)(1.3-1.5)}{(0.5-1.0)(0.5-1.5)}+0.841471\frac{(1.3-0.5)(1.3-1.5)}{(1.0-0.5)(1.0-1.5)} +0.997495\frac{(1.3-0.5)(1.3-1.0)}{(1.5-0.5)(1.5-1.0)}=0.959796$$

으로 된다. 한편 $|f'''(x)|=|\sin(x)|\le 1$ 임을 이용하면, 오차는 식(18)로부터 평가할 수 있다.

$$\begin{aligned}|e(x=1.3)| &= \left|(x-0.5)(x-1.0)(x-1.5)\frac{f'''(\xi)}{3!}\right|_{x=1.3} \\ &= \left|(1.3-0.5)(1.3-1.0)(1.3-1.5)\frac{f'''(\xi)}{3!}\right| \\ &= 0.008|f'''(\xi)|\end{aligned} \tag{20}$$

다시 말해서 오차는 $|e(x=1.3)|\le 0.008$으로 되고, 이것은 정확한 오차

$$e_{exact}=f(1.3)-P_2(1.3)=\sin(1.3)-0.959796=0.003762\le 0.008$$

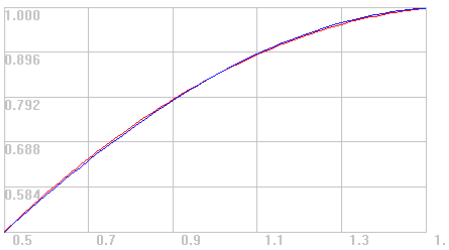
와 비교하여 타당한 결과이다. 일반적으로, 식(18)에 포함된 미분항 $f^{(n+1)}(\xi)/(n+1)!$의 알려지지 않은 거동으로 인하여 오차의 평가는 매우 어렵다.

☑ cemm# code

x	0.5	1.0	1.5
$f(x)$	0.479526	0.841471	0.997495

에 대해서 보간곡선과 원래의 함수 $f(x) = \sin(x)$ 를 함께 그려보자.

```
#> .interp( [ 0.5, 1.0, 1.5 ], [ 0.479526, 0.841471, 0.997495 ] ) (1.3)
    ans =      0.95979592
#> .interp( [ 0.5, 1.0, 1.5 ], [ 0.479526, 0.841471, 0.997495 ] )
       .plot(0.5,1.5);
#> plot+ .x(0.5,1.5) ( sin(x) );
```



5-3 뉴턴 전진/후진 보간

■ 전진차분과 전진차분표. 일반성을 잃지 않고 세 개의 점 x_1, x_2, x_3을 $x_2 = x_1 + h,\ x_3 = x_1 + (1+\alpha)h$ 와 같이 배치해보자 (여기서 $\alpha \neq 0$ 는 아직 알려지지 않은 값이다). 그러면 점 $x = x_1$에서 함수 $f(x)$의 미분은

$$f'(x_1) = \lim_{x_2 \to x_1} \frac{f(x_2) - f(x_1)}{x_2 - x_1} \tag{21}$$

으로 정의된다. 여기서 $x_2 - x_1$ 의 크기가 유한할 경우 윗식의 우변에서 분수형태

$$\Delta^1 f_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{f_2 - f_1}{x_2 - x_1} \tag{22}$$

을 전진분할차분 (前進分割差分, forward divided difference) 또는 간단히 전진차분이라고 한다. 이때 $\Delta^1 f_1$ 에서 '전진' 이라는 용어는 첨자가

증가하는 값 f_2에서 f_1을 빼는 형태로 되기 때문에 붙은 이름이다. 그러면 식(21)은 보다 간단히

$$\lim_{h\to 0}\Delta^1 f_1 = f'(x_1) \tag{23}$$

으로 쓸 수 있다. 마찬가지로 함수의 2계미분 (증명은 심화논의 참조)

$$\frac{1}{2!}f''(x_1) = \lim_{x_3,x_2\to x_1}\frac{\frac{f(x_3)-f(x_2)}{x_3-x_2}-\frac{f(x_2)-f(x_1)}{x_2-x_1}}{x_3-x_1} \tag{24}$$

와 연관된 전진차분은

$$\Delta^2 f_1 = \frac{\frac{f_3-f_2}{x_3-x_2}-\frac{f_2-f_1}{x_2-x_1}}{x_3-x_1} = \frac{\Delta^1 f_2-\Delta^1 f_1}{x_3-x_1} \tag{25}$$

으로 정의된다. 위에서, 식(24)의 극한값은 α 값에는 의존하지 않는다는 점에 주목한다 [상세한 내용은 심화논의 참조, $x_2 = x_1 + h, x_3 = x_1 + (1+\alpha)h$]. 다시 말해서

$$\lim_{h\to 0}\Delta^2 f_1 = \frac{f''(x_1)}{2!} \tag{26}$$

이 성립한다. 위의 과정은 $x_0, x_1, x_2, \dots, x_n$ 에 대해서 주어진 데이터 $f_0, f_1, f_2, \dots, f_n$ 에 관한 고차의 분할차분 (divided-difference)에 대해서도 쉽게 확장할 수 있다. 일반적으로 전진분할차분의 극한거동은

$$\lim_{h\to 0}\Delta^k f_i = \frac{f^{(k)}(x_i)}{k!} \tag{27}$$

으로 쓸 수 있다. 여기서 $h\to 0$인 조건은 반드시 $|x_{k+i}-x_i|\to 0$인 조건과 동일한 것을 의미한다. 점 $x_0, x_1, x_2, \dots, x_n$ 에서 주어진 데이터 $f_0, f_1, f_2, \dots, f_n$ 에 대한 전진차분을 모두 계산하여 표 5-1과 같이 나타낸 것을 전진분할차분표 (前進分割差分表, forward divided difference table) 또는 전진차분표라고 한다.

Table 5-1 Forward divided-difference table

x_0	f_0	$\Delta^1 f_0 = \frac{f_1 - f_0}{x_1 - x_0}$	$\Delta^2 f_0 = \frac{\Delta^1 f_1 - \Delta^1 f_0}{x_2 - x_0}$	$\Delta^3 f_0 = \frac{\Delta^2 f_1 - \Delta^2 f_0}{x_3 - x_0}$
x_1	f_1	$\Delta^1 f_1 = \frac{f_2 - f_1}{x_2 - x_1}$	$\Delta^2 f_1 = \frac{\Delta^1 f_2 - \Delta^1 f_1}{x_3 - x_1}$	$\Delta^3 f_1 = \frac{\Delta^2 f_2 - \Delta^2 f_1}{x_4 - x_1}$
x_2	f_2	$\Delta^1 f_2 = \frac{f_3 - f_2}{x_3 - x_2}$	$\Delta^2 f_2 = \frac{\Delta^1 f_3 - \Delta^1 f_2}{x_4 - x_2}$	....
x_3	f_3	$\Delta^1 f_3 = \frac{f_4 - f_3}{x_4 - x_3}$		
...	...	...	$\Delta^j f_i \equiv \frac{\Delta^{j-1} f_{i+1} - \Delta^{j-1} f_i}{x_{i+j} - x_i}$	
x_{n-1}	f_{n-1}	$\Delta^1 f_{n-1} = \frac{f_n - f_{n-1}}{x_n - x_{n-1}}$		
x_n	f_n			

```
matrix divdiff(&x,&f)
{
  n = x.len;
  tab = y._1;  // compulsory column vector

  for.j(2,n) {
    tab |= 0;  // one more column initialized by zero
    for.i(1,n-j+1) tab(i,j) = (tab(i+1,j-1)-tab(i,j-1))/(x(i+j-1)-x(i));
  }
  return tab;
}
```

☑ 여기서 $\Delta^k f_i$ 는 표준적인 표현은 아니라는 점에 주의한다. 일반적으로 대부분의 교재에서는 전진분할차분을 $f[x_1, x_2] = \Delta^1 f_1$, $f[x_1, x_2, x_3] = \Delta^2 f_1$으로 나타낸다.

■ 전진차분표를 이용한 보간. 주어진 데이터에 대한 전진차분표가 만들어지면 이것을 이용하여 보간을 할 수 있다. 예를 들어 $(x_0, f_0), (x_1, f_1), (x_2, f_2)$에 대한 보간을 아래의 2차 보간다항식

$$P_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) \tag{28}$$

을 이용하여 구해보자 [보간식 $P_2(x)$ 의 아래첨자는 다항식의 차수를 의미한다]. 먼저 $x = x_0$를 대입하면

$$a_0 = f_0 = \Delta^0 f_0 \tag{29}$$

을 얻는다. 다시 $x = x_1$을 대입하면

$$f_1 = f_0 + a_1(x_1 - x_0) \quad \Rightarrow \quad a_1 = \frac{f_1 - f_0}{x_1 - x_0} = \Delta^1 f_0 \tag{30}$$

으로 계수를 얻는다. 마지막으로 $x = x_2$를 대입하면

$$f_2 = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \tag{31}$$

이고, 윗식을 a_2에 대해서 풀면

$$\begin{aligned} a_2(x_2 - x_0)(x_2 - x_1) &= f_2 - f_1 + f_1 - f_0 - \frac{f_1 - f_0}{x_1 - x_0}(x_2 - x_1 + x_1 - x_0) \\ &= f_2 - f_1 - \frac{f_1 - f_0}{x_1 - x_0}(x_2 - x_1) \end{aligned}$$

즉

$$a_2 = \frac{1}{x_2 - x_0}\left(\frac{f_2 - f_1}{x_2 - x_1} - \frac{f_1 - f_0}{x_1 - x_0}\right) = \frac{\Delta^1 f_1 - \Delta^1 f_0}{x_2 - x_0} = \Delta^2 f_0 \tag{32}$$

가 된다. 따라서 최종보간은

$$P_2(x) = \Delta^0 f_0 + (x - x_0)\Delta^1 f_0 + (x - x_0)(x - x_1)\Delta^2 f_0 \tag{33}$$

으로 쓸 수 있다. 이것을 $n+1$개의 데이터로 확장하면 n차 보간다항식은

$$\begin{aligned} P_n(x) = \Delta^0 f_0 &+ (x - x_0)\Delta^1 f_0 + (x - x_0)(x - x_1)\Delta^2 f_0 \\ &+ \cdots + (x - x_0)(x - x_1)\cdots(x - x_{n-1})\Delta^n f_0 \end{aligned} \tag{34}$$

```
// Newton polynomial
p(x) = a0 + a1(x - x0) + a2(x - x0)(x - x1) + a3(x - x0)(x - x1)(x - x2) + ...
#> poly(a0,a1,a2,a3,...) .newton( [x0,x1,x2, ...] )
```

와 같이 간단히 나타낼 수 있다.

위와 같이 전진차분표를 이용하여 보간을 하는 방법을 뉴턴의 전진보간법 (Newton's forward interpolation)이라고 한다. 한편, n 차

보간다항식 $P_n(x)$를 이용하는 뉴턴의 전진보간법은 다음의 장점을 가지고 있다.

- 전진차분표를 한 번 만들어 두면 데이터가 바뀌지 않는 한 계속 사용되므로 효율적이다.
- 데이터 (x_{n+1}, f_{n+1}) 가 추가되면 기존의 분할차분표에 f_{n+1}가 포함되는 항만을 계산하여 추가하면 전진차분표가 만들어진다.
- 다른 x값에 대한 보간을 하더라도 전진차분표를 다시 만들 필요가 없다.
- 임의의 구간을 선택하여 보간을 할 때 기존의 전진차분표를 그대로 사용할 수 있다.

예를 들어 세 개의 점 x_2, x_3, x_4 에 대해서 2차식보간을 할 때는 전진차분표로부터 값을 찾아서 다음과 같이 보간하면 된다.

$$P_2(x) = \Delta^0 f_2 + (x - x_2)\Delta^1 f_2 + (x - x_2)(x - x_3)\Delta^2 f_2 \tag{35}$$

■ 차항규칙 (next-term rule). n차의 보간다항식에 의한 보간에 따른 오차는 이미 식(18)로부터

$$e(x) = (x - x_0)(x - x_1)\cdots(x - x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}, \quad x_0 \le \xi \le x_n$$

임을 알고 있다. 그런데 $(n+1)$계 도함수 $f^{(n+1)}(\xi)$은 일반적으로 알려져 있지 않지만 식(27)에 주어진 극한거동을 이용하면 보간점 x 의 부근에서 최대오차를 근사적으로

$$|e(x)|_{max} = |(x - x_0)(x - x_1)\cdots(x - x_n)| \times |\Delta^{n+1} f|_{max} \tag{36}$$

와 같이 나타낼 수 있다. 이때 분할차분표로부터 보간점 부근에서 $|\Delta^{n+1} f|_{max}$ 값을 가져올 수 있다고 가정한다. 이것은 $|\Delta^{n+1} f|_{max}$ 의 값이 분할차분표의 다음 열에서 얻어지므로 흔히 차항규칙 (次項規則, next-term rule)이라고도 부른다.

(예제 2) $f(x) = e^x$에 대한 데이터가

x_i	0.0	0.1	0.3	0.4	0.6	0.7	1.0
f_i	1.000000	1.105170	1.349858	1.491824	1.822118	2.013752	2.718281

으로 주어질 때 전진차분표를 구하라.

표 5-1을 참조하여 계산하면

$$\Delta^1 f_0 = \frac{f_1 - f_0}{x_1 - x_0} = \frac{1.105170 - 1}{0.1 - 0} = 1.0517$$
$$\Delta^1 f_1 = \frac{f_2 - f_1}{x_2 - x_1} = \frac{1.349858 - 1.105170}{0.3 - 0.1} = 1.22344$$
$$\Delta^1 f_2 = \frac{f_3 - f_2}{x_3 - x_2} = \frac{1.491824 - 1.349858}{0.4 - 0.3} = 1.41966$$
$$\cdots$$

$$\Delta^2 f_0 = \frac{\Delta^1 f_1 - \Delta^1 f_0}{x_2 - x_0} = \frac{1.22344 - 1.0517}{0.3 - 0} = 0.572467$$
$$\Delta^2 f_1 = \frac{\Delta^1 f_2 - \Delta^1 f_1}{x_2 - x_0} = \frac{1.41966 - 1.22344}{0.4 - 0.1} = 0.654067 \qquad (37)$$
$$\cdots$$

등으로 되고, 결과를 정리하면 표 5-2에 보인 바와 같다.

Table 5-2 Forward divided-difference table (example 2)

i	x_i	$\Delta^0 f$	$\Delta^1 f$	$\Delta^2 f$	$\Delta^3 f$	$\Delta^4 f$	$\Delta^5 f$	$\Delta^6 f$
0	0.0	1.000000	1.05170	0.572467	0.204000	0.055444	0.011825	0.002094
1	0.1	1.105170	1.22344	0.654067	0.237267	0.063722	0.013920	
2	0.3	1.349858	1.41966	0.772700	0.275500	0.076250		
3	0.4	1.491824	1.65147	0.882900	0.328875			
4	0.6	1.822118	1.91634	1.080225				
5	0.7	2.013752	2.34843					
6	1.0	2.718281						

☑ (예제 2)를 풀기 위한 cemm# code는 아래와 같다.

```
#> x = [ 0, 0.1, 0.3, 0.4, 0.6, 0.7, 1 ];
#> f = [ 1, 1.10517, 1.349858, 1.491824, 1.822118, 2.013752, 2.718281 ];
#> f`./x`;  // first-order difference
    ans =  [    1.0517   1.22344   1.41966   1.65147   1.91634   2.34843 ]
#> divdiff(x,f);  // user-function listed above
    ans =
 [       1    1.0517  0.572467     0.204 0.0554444 0.0118254  0.002094 ]
 [ 1.10517   1.22344  0.654067  0.237267 0.0637222 0.0139198         0 ]
 [ 1.34986   1.41966    0.7727    0.2755   0.07625         0         0 ]
 [ 1.49182   1.65147    0.8829  0.328875         0         0         0 ]
```

```
 [ 1.82212   1.91634   1.08022         0         0         0         0 ]
 [ 2.01375   2.34843         0         0         0         0         0 ]
 [ 2.71828         0         0         0         0         0         0 ]
#> dtab = .dtable(x,f);  // dot-function .dtable
    dtab =
 [         1    1.0517  0.572467     0.204 0.0554444 0.0118254  0.002094 ]
 [   1.10517   1.22344  0.654067  0.237267 0.0637222 0.0139198         0 ]
 [   1.34986   1.41966    0.7727    0.2755   0.07625         0         0 ]
 [   1.49182   1.65147    0.8829  0.328875         0         0         0 ]
 [   1.82212   1.91634   1.08022         0         0         0         0 ]
 [   2.01375   2.34843         0         0         0         0         0 ]
 [   2.71828         0         0         0         0         0         0 ]
```

(예제 3) 앞의 예제에서 주어진 데이터와 전진차분표(표 5-2)를 이용하여 $i = 3,4,5$ 구간에서의 2차 보간다항식을 구하라. 그리고, $x = 0.5$ 에서의 보간값을 하라.

표 5-2로부터 $i = 3,4,5$구간에 대해서는

$$x_3 = 0.4 \qquad x_4 = 0.6 \qquad x_5 = 0.7$$
$$\Delta^0 f_3 = 1.491824 \qquad \Delta^1 f_3 = 1.65147 \qquad \Delta^2 f_3 = 0.882900$$

이고, 다음항(next term)은 $\Delta^3 f_3 = 0.328875$ 으로 주어진다. 따라서 2차 보간식은

$$\begin{aligned} P_2(x) &= \Delta^0 f_3 + (x - x_3)\Delta^1 f_3 + (x - x_3)(x - x_4)\Delta^2 f_3 \\ &= 1.491824 + 1.65147(x - 0.4) \\ &\quad +0.882900(x - 0.4)(x - 0.6) \\ &= 1.043132 + 0.76857x + 0.8829x^2 \end{aligned} \tag{38}$$

```
#> p2 = poly(1.491824, 1.65147, 0.8829).newton([0.4,0.6]);
    p2 = poly( 1.04313  0.76857  0.8829 )
       = 0.8829x^2 +0.76857x +1.04313
```

이다. 따라서 $x = 0.5$ 에서의 보간값은

$$P_2(0.5) = 1.648142 \tag{39}$$

```
#> p2(0.5);
    ans =        1.648142
```

으로 된다. 계속해서, 보간오차는 식(36)을 이용하여 추정한다. 2차 보간

식의 경우 데이터는 $i=3$ 에서 $i=5$까지로 주어지므로 최대오차는

$$|e(0.5)|_{max,P_2} = |(x-x_3)(x-x_4)(x-x_5)|_{x=0.5} \times |\Delta^3 f|_{max} \\ = |(0.5-0.4)(0.5-0.6)(0.5-0.7)|(0.328875) = 0.000657 \tag{40}$$

으로 되고, 엄밀한 오차 $\left|e^{0.5} - P_2(0.5)\right| = 0.000579$와 부합한다

☑ cemm# code

```
#> p2 = dtab..(4)(1,2,3).apoly ;  // coefficients of Newton
    p2 = poly( 1.49182  1.65147  0.8829 )
       = 0.8829x^2 +1.65147x +1.49182
#> p = dtab..(4)(1,2,3).apoly.newton(x.(4:6));  // [ x(4),x(5),x(6) ]
    p = poly( 1.04313  0.76857  0.8829 )
       = 0.8829x^2 +0.76857x +1.04313
#> p(0.5);
    ans =        1.648142
// x.(4:6) = x.entry(4:6) = [ x(4),x(5),x(6) ]
#> p = .interp( x.(4:6),f.(4:6) );
    p = poly( 1.04313  0.76857  0.8829 )
       = 0.8829x^2 +0.76857x +1.04313
```

■ 후진차분과 후진차분표. 전진차분과 유사하게 후진차분 (後進差分, backward difference)은

$$\nabla^1 f_2 = \Delta^1 f_1 = \frac{f(x_2)-f(x_1)}{x_2-x_1} = \frac{f_2-f_1}{x_2-x_1} \tag{41}$$

으로 정의된다. 그러나 전진차분과 후진차분은 서로 독립적이지 못하며

$$\nabla^j f_i = \Delta^j f_{i-j} \tag{42}$$

인 관계가 있다. 표 5-3에는 함수 $f(x)$의 후진차분표를 정리하였다. 그러면 후진차분표에 의한 뉴턴의 후진보간은 다음과 같게 된다.

$$P_n(x) = \nabla^0 f_n + (x-x_n)\nabla^1 f_n + (x-x_n)(x-x_{n-1})\nabla^2 f_n + \cdots \\ + (x-x_n)(x-x_{n-1})\cdots(x-x_1)\nabla^n f_n \tag{43}$$

또한 유일성정리로부터 뉴턴의 후진보간은 전진보간의 경우와 동일한

내용이다. 그리고 오차해석은 식(36)에 주어진 것과 유사하며

$$|e(x)|_{max} = |(x-x_0)(x-x_1)\cdots(x-x_n)| \times |\nabla^{n+1} f|_{max} \tag{44}$$

와 같이 후진 분할차분 연산자를 이용하여 표현된다는 점만 다르다.

Table 5-3 Backward divided-difference table

x_0	f_0				
x_1	f_1	$\nabla^1 f_1 = \frac{f_1 - f_0}{x_1 - x_0}$			
x_2	f_2	$\nabla^1 f_2 = \frac{f_2 - f_1}{x_2 - x_1}$	$\nabla^2 f_2 = \frac{\nabla^1 f_2 - \nabla^1 f_1}{x_2 - x_0}$		
x_3	f_3	$\nabla^1 f_3 = \frac{f_3 - f_2}{x_3 - x_2}$	$\nabla^2 f_3 = \frac{\nabla^1 f_3 - \nabla^1 f_2}{x_3 - x_1}$	$\nabla^3 f_3 = \frac{\nabla^2 f_3 - \nabla^2 f_2}{x_3 - x_0}$	
...	...				
x_{n-1}	f_{n-1}			$\nabla^j f_i \equiv \frac{\nabla^{j-1} f_i - \nabla^{j-1} f_{i-1}}{x_i - x_{i-j}}$	
x_n	f_n	$\nabla^1 f_n = \frac{f_n - f_{n-1}}{x_n - x_{n-1}}$			

5-4 등간격에 대한 뉴턴 보간

■ 등간격에서의 전진차분. 점 $x_s = x_0 + sh,\ s = 0,1,2,3,\ldots,n$ 이 등간격으로 배치되어 있는 경우 일정한 간격을 h 라고 할 때 $x = x_0 + sh$ 를 이용하면 전진차분을 훨씬 간단한 형태로 나타낼 수 있다. 등간격인 경우에 대해서 식(25)를 다시 쓰면

$$\Delta^2 f_1 = \frac{\frac{f_3 - f_2}{x_3 - x_2} - \frac{f_2 - f_1}{x_2 - x_1}}{x_3 - x_1} = \frac{\frac{f_3 - f_2}{h} - \frac{f_2 - f_1}{h}}{2h} = \frac{1}{2!\,h^2}(f_1 - 2f_2 + f_3) \tag{45}$$

으로 된다. 여기서

$$\Delta^k = \frac{1}{k!\,h^k}\Delta_*^k \tag{46}$$

인 관계가 성립하도록 $*$를 아래첨자로 가지는 전진차분연산자 Δ_*^k 를

정의하면

$$\Delta^2 f_1 = \frac{1}{2!\,h^2}\,\Delta_*^2 f_1 = \frac{1}{2!\,h^2}(f_3 - 2f_2 + f_1) \tag{47}$$

으로 쓸 수 있다. 마찬가지로 고계의 전진차분에 대해서도

$$\begin{aligned}\Delta^3 f_1 &= \frac{1}{3!\,h^3}\,\Delta_*^3 f_1 = \frac{1}{3!\,h^3}(f_4 - 3f_3 + 3f_2 - f_1)\\ \Delta^4 f_1 &= \frac{1}{4!\,h^4}\,\Delta_*^4 f_1 = \frac{1}{4!\,h^4}(f_5 - 4f_4 + 6f_3 - 4f_2 + f_1)\end{aligned} \tag{48}$$

등을 얻는다. 표 5-4에는 전진차분연산자 Δ_*^k에 대한 차분결과를 정리하였다. 계수의 변화를 살펴보면 크기는 파스칼의 삼각형과 같고 부호는 교대로 바뀌는 것을 알 수 있다.

Table 5-4 Forward difference table

$\Delta_*^0 f_i = f_i$
$\Delta_*^1 f_i = f_{i+1} - f_i$
$\Delta_*^2 f_i = f_{i+2} - 2f_{i+1} + f_i$
$\Delta_*^3 f_i = f_{i+3} - 3f_{i+2} + 3f_{i+1} - f_i$
$\Delta_*^4 f_i = f_{i+4} - 4f_{i+3} + 6f_{i+2} - 4f_{i+1} + f_i$
$\Delta_*^5 f_i = f_{i+5} - 5f_{i+4} + 10f_{i+3} - 10f_{i+2} + 5f_{i+1} - f_i$

차분연산자 Δ_*^k에 관하여

$$f^{(k)}(x_i) = \lim_{h\to 0}\frac{1}{h^k}\,\Delta_*^k f_i \tag{49}$$

임에 주목한다. 그리고 등간격인 경우 $x = x_0 + sh$로부터

$$s = \frac{x - x_0}{h} \tag{50}$$

와 같이 좌표변환하면

$$(x - x_0) = hs$$

$$(x-x_0)(x-x_1)=(x-x_0)[(x-x_0)-h]=h^2 s(s-1)$$
$$\vdots$$
$$(x-x_0)(x-x_1)(x-x_2)\cdots(x-x_{k-1})$$
$$=h^k s(s-1)(s-2)\cdots(s-k+1)$$

등으로 표현되고, $\Delta^k = \frac{1}{k!h^k}\Delta_*^k$ 이므로 식(34)에서 h 의 거듭제곱은 서로 상쇄되어 식(34)는

$$\begin{aligned} P_n(x) &= \Delta^0 f_0 + (x-x_0)\Delta^1 f_0 + (x-x_0)(x-x_1)\Delta^2 f_0 \\ &\quad + \cdots + (x-x_0)(x-x_1)\cdots(x-x_{n-1})\Delta^n f_0 \\ &= \Delta_*^0 f_0 + s\,\Delta_*^1 f_0 + \frac{s(s-1)}{2!}\Delta_*^2 f_0 + \cdots \\ &\quad + \frac{s(s-1)\cdots(s-n+1)}{n!}\Delta_*^n f_0 \\ &= \binom{s}{0}\Delta_*^0 f_0 + \binom{s}{1}\Delta_*^1 f_0 + \binom{s}{2}\Delta_*^2 f_0 + \cdots + \binom{s}{n}\Delta_*^n f_0 \\ &= \sum_{k=0}^{n}\binom{s}{k}\Delta_*^k f_0 \end{aligned} \tag{51}$$

와 같이 등간격에서 뉴턴의 전진차분 보간다항식으로 된다. 여기서 이항계수 (二項係數, binomial coefficient)는 아래와 같이 정의된다.

$$\binom{s}{k} = \frac{1}{k!}s(s-1)(s-2)\cdots(s-k+1) \tag{52}$$

이항계수 $\binom{s}{k}$ 는 s 에 관한 k 차 다항식이며, 이에 대한 연산은 부록에 상세하게 설명되어 있다. 한편 cemm#에서 이항다항식을 처리하는 예를 보인다.

```
#> poly.binom(5).iratio ;
ans = poly( 0  1/5  -5/12  7/24  -1/12  1/120  )
```

$$\begin{aligned} \binom{s}{5} &= \frac{1}{5!}s(s-1)(s-2)(s-3)(s-4) \\ &= \frac{1}{5}s - \frac{5}{12}s^2 + \frac{7}{24}s^3 - \frac{1}{12}s^4 + \frac{1}{120}s^5 \end{aligned}$$

■ 등간격에서의 후진차분. 전진차분과 마찬가지로 후진차분에 대하여도 새로운 독립변수 s 를

$$s = \frac{x - x_n}{h} \tag{53}$$

와 같이 정의하면

$$\begin{aligned}
&(x - x_n) = hs \\
&(x - x_n)(x - x_{n-1}) = (x - x_n)[(x - x_n) + h] = h^2 s(s+1) \\
&(x - x_n)(x - x_{n-1})(x - x_{n-2}) \cdots (x - x_{n-k+1}) \\
&\qquad = h^k s(s+1)(s+2) \cdots (s+k-1)
\end{aligned}$$

으로 된다. 여기서 $\nabla^k = \frac{1}{k!h^k}\nabla_*^k$ 으로 정의하면 식(43)에서 h의 거듭제곱은 서로 상쇄되고, 따라서 식(43)은

$$\begin{aligned}
P_n(x) &= \nabla^0 f_n + (x - x_n)\nabla^1 f_n + (x - x_n)(x - x_{n-1})\nabla^2 f_n \\
&\quad + \cdots + (x - x_n)(x - x_{n-1}) \cdots (x - x_1)\nabla^n f_n \\
&= \nabla_*^0 f_n + s\nabla_*^1 f_n + \frac{s(s+1)}{2!}\nabla_*^2 f_n + \cdots + \binom{s+n-1}{n}\nabla_*^n f_n \\
&= \sum_{k=0}^{n} \binom{s+k-1}{k}\nabla_*^k f_n
\end{aligned} \tag{54}$$

와 같이 등간격에서 뉴턴의 후진차분 보간다항식을 얻을 수 있다. 그리고 표 5-5에는 후진차분연산자를 정리하였다.

Table 5-5 Backward difference table

$\nabla_*^0 f_i = f_i$
$\nabla_*^1 f_i = f_i - f_{i-1}$
$\nabla_*^2 f_i = f_i - 2f_{i-1} + f_{i-2}$
$\nabla_*^3 f_i = f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3}$
$\nabla_*^4 f_i = f_i - 4f_{i-1} + 6f_{i-2} - 4f_{i-3} + f_{i-4}$
$\nabla_*^5 f_i = f_i - 5f_{i-1} + 10f_{i-2} - 10f_{i-3} + 5f_{i-4} - f_{i-5}$

■ 두 종류의 차분연산자. 위에서 도입한 두 가지의 차분연산자 (* 아래첨자를 가진 것과 없는 것), 예를 들어

$$\Delta^1 f_i = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}, \quad \Delta_*^1 f_i = f_{i+1} - f_i \tag{55}$$

에서는 구간이 일정한 경우

$$\Delta^k = \frac{1}{k!\,h^k}\Delta_*^k, \qquad \nabla^k = \frac{1}{k!\,h^k}\nabla_*^k$$
$$\lim_{h\to 0}\Delta^k f_i = \lim_{h\to 0}\nabla^k f_i = \frac{f^{(k)}(x_i)}{k!} \tag{56}$$
$$\lim_{h\to 0}\frac{1}{k!\,h^k}\Delta_*^k f_i = \lim_{h\to 0}\frac{1}{k!\,h^k}\nabla_*^k f_i = \frac{f^{(k)}(x_i)}{k!}$$

인 관계가 성립하므로 서로 독립이 아니다. 그럼에도 불구하고 두 가지 표현을 함께 사용하는 이유는 Δ_*^k 또는 ∇_*^k는 뒤에서 다루는 수치미분 등의 이론분야에서 유리하기 때문이다. 보간한 수치값만을 생각하면 어느 쪽을 사용하여도 같은 결과이므로 문제되지 않는다.

(예제 4) $f(x) = \tan^{-1}x$ 에 대한 데이터가 $h = 0.2$ 인 등간격으로 $x_0 = 0$ 부터 $i = 0,1,2,3$으로 주어져 있을 때 전진차분 및 후진차분을 이용하여 $x = 0.27$에서의 보간값을 각각 구하라.

데이터가 주어져 있지 않으므로 프로그램에서 값을 생성하여 예제를 풀기로 한다. 먼저 전진차분표를 만들면

i	x_i	$\Delta_*^0 f$	$\Delta_*^1 f$	$\Delta_*^2 f$	$\Delta_*^3 f$
0	0	0.000000	0.197396	-0.014285	-0.008913
1	0.2	0.197396	0.183111	-0.023198	
2	0.4	0.380506	0.159913		
3	0.6	0.540420			

이므로, 전진보간은 $s = \frac{0.27-0}{0.2} = 1.35$ 에 대해서

$$P_3(s) = 0 + 0.197396s - \frac{0.014285s(s-1)}{2} - \frac{0.008913s(s-1)(s-2)}{3!}$$
$$= 0.201567s - 0.002686s^2 - 0.001485s^3 \tag{57}$$

이다. 따라서

$$P_3(s = 1.35) = 0.263565 \tag{58}$$

을 얻는다. 마찬가지로 후진차분표를 만들면

i	x_i	∇_*^0	∇_*^1	∇_*^2	∇_*^3
0	0	0.000000			
1	0.2	0.197396	0.197396		
2	0.4	0.380506	0.183111	-0.014285	
3	0.6	0.540420	0.159913	-0.023198	-0.008913

이고, $s = \frac{0.27-0.6}{0.2} = -1.65$을 이용하여

$$P_3(s) = 0.54042 + 0.159913s - \frac{0.023198s(s+1)}{2} - \frac{0.008913s(s+1)(s+2)}{3!} = 0.54042 + 0.145343s - 0.016055s^2 - 0.001485s^3 \quad (59)$$

즉, $P_3(s=-1.65) = 0.263565$ 가 된다. 참값은 $\tan^{-1} 0.27 = 0.263712$이다.

☑ 실제의 계산과정은 임의의 간격에 대한 보간다항식을 이용하도록 한다. 이것은 등간격에 대한 보간다항식은 일반적으로 수치이론을 개발하는 데에 주로 사용되기 때문이다. 임의의 간격에 대한 프로그램에 추가하여 다시 등간격 보간을 개발하는 것은 소모적이라고 할 수 있다. 따라서 cemm# code의 실행을 첨부한다.

```
#> .interp( [ 0, 0.2, 0.4, 0.6 ],
           [ 0, 0.197396, 0.380506, 0.54042 ])(0.27);
ans =      0.26356561
```

(예제 5) 구간 $-5 \le x \le 5$ 에서 함수 $f(x) = 1/(1+x^2)$ 를 고려하자. 구간을 등간격으로 나누고 다항식의 차수가 $n = 6,8,10$ 인 경우에 대해서 보간다항식을 구하고 그려보라.

cemm#을 아래와 같이 활용한다.

```
double f(x) = 1/(1+x^2);   // function definition

plot.x(-5,5) ( f(x) );     // function plot in red line
for.n(6,10,2) {
```

```
  x=(-5,5).span(n+1);;      // n-th degree has (n+1) points
  y=f++(x);;                // function upgrade to accept matrix
  p = .interp(x,y).trun12;
  p.plot+(-5,5);
}
    ans = poly( 1  0  -0.351364  0  0.0335319  0  -0.000840633 )
        = -0.000840633x^6 +0.0335319x^4 -0.351364x^2 +1
    ans = poly( 1  0  -0.528121  0  0.0981875  0  -0.00658016  0
                0.000137445 )
        = 0.000137445x^8 -0.00658016x^6 +0.0981875x^4 -0.528121x^2 +1
    ans = poly( 1  0  -0.674208  0  0.197376  0  -0.0244118  0
                0.00126697  0  -2.26244e-005 )
        = -2.26244e-005x^10 +0.00126697x^8 -0.0244118x^6 +0.197376x^4
          -0.674208x^2 +1
```

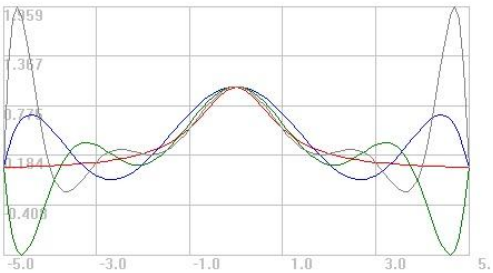


Figure 5-2 Interpolation of $f(x) = 1/(1+x^2)$

구간의 양끝에서 보간다항식의 행동은 전혀 보간식이라고 보기 어렵다. 따라서 고차다항식을 이용한 보간은 그다지 바람직하지 않다. 따라서 부분구간에 대해서 저차다항식을 이용한 보간 (즉, 스플라인 보간)을 이용하는 것이 추천되며 이것은 선택과정에서 다룬다.

5-5 최소제곱 회귀

회귀 (回歸, regression)란 주어진 데이터에 대한 최적의 근사를 얻는 방법으로 데이터와의 차이의 제곱을 이용하여 회귀곡선 (regression curve)을 구하는 것을 최소제곱 회귀라고 한다.

■ 회귀의 원리. 총 n개의 데이터가

$$(x_1, f_1), (x_2, f_2), \cdots, (x_n, f_n)$$

으로 주어져 있을 때 r개의 기저함수 (base function)

$$u_1(x), u_2(x), u_3(x), \cdots, u_r(x)$$

의 선형결합

$$u(x) = a_1u_1 + a_2u_2 + \cdots + a_ru_r \tag{60}$$

을 이용하여 주어진 데이터의 최적근사를 다음과 같이 구해보자. 먼저 주어진 데이터와의 절대오차 $e_i = |f_i - u(x_i)|$ 의 제곱의 합, 즉 데이터 편차의 제곱의 합

$$S = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} [f_i - a_1u_1(x_i) - a_2u_2(x_i) - \cdots - a_ru_r(x_i)]^2 \tag{61}$$

을 최소로 하는 계수 $a_1, a_2, a_3, \cdots, a_r$은 $j = 1,2,3, \cdots r$ 에 대해서

$$\frac{\partial S}{\partial a_j} = (-2)\sum_{i=1}^{n} u_j(x_i)[f_i - a_1u_1(x_i) - a_2u_2(x_i) - \cdots - a_ru_r(x_i)] = 0 \tag{62}$$

즉

$$\sum_{i=1}^{n} u_j(x_i)[a_1u_1(x_i) + a_2u_2(x_i) + \cdots + a_ru_r(x_i)] = \sum_{i=1}^{n} u_j(x_i)f_i \tag{63}$$

으로부터 구할 수 있다. 이것이 회귀곡선을 결정하는 원리이며 기저함수 $u_i(x)$의 선택에 따라 1차회귀, 다항식회귀, 비선형회귀 등으로 구별된다.

■ 회귀의 정확성과 평가. 회귀곡선이 원래의 데이터와 어느 정도 부합하는가를 판단하기 위한 기준으로 상관계수 (相關係數, correlation coefficient)

$$r^2 = \frac{S_0 - S}{S_0} \tag{64}$$

을 정의한다. 위에서 S_0는 데이터평균으로부터의 편차의 제곱의 합

$$S_0 = \sum_{i=1}^{n}(f_i - f_{ave})^2, \quad f_{ave} = \frac{1}{n}\sum_{i=1}^{n} f_i \tag{65}$$

을 의미하며 이것은 곡선근사법에 관계없는 데이터 자체의 성질이다. 상관계수의 최대값은 $S = 0$ 일때의 $r = 1$ 로 이때는 완전한 상관관계에 있다고 한다 (다시 말해서 회귀곡선이 주어진 데이터의 모든 점들을 지난다). 일반적으로 r 의 값이 1에 가까울수록 상관관계가 좋아진다.

■ 1차회귀 . 1차회귀는 근사곡선을

$$u(x) = a + bx \tag{66}$$

으로 두고 계수 a, b 를 결정한다. 위의 근사곡선에 대해서 식(61)은

$$S = \sum_{i=1}^{n}[f_i - a - bx_i]^2 \tag{67}$$

이고, 따라서 최소조건인 식(62)는

$$\frac{\partial S}{\partial a} = (-2)\sum_{i=1}^{n}(f_i - a - bx_i) = (-2)\left[\sum_{i=1}^{n} f_i - a\sum_{i=1}^{n} 1 - b\sum_{i=1}^{n} x_i\right] = 0$$
$$\frac{\partial S}{\partial b} = (-2)\sum_{i=1}^{n} x_i(f_i - a - bx_i) = (-2)\left[\sum_{i=1}^{n} x_i f_i - a\sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} {x_i}^2\right] = 0$$

다시 말해서

$$\begin{aligned} a\sum_{i=1}^{n} 1 + b\sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} f_i \\ a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} {x_i}^2 &= \sum_{i=1}^{n} x_i f_i \end{aligned} \tag{68}$$

와 같이 연립 선형방정식으로 정리된다. 위의 방정식은 쉽게 풀 수 있으며 이와 같이 구한 a, b 를 이용하면 1차회귀곡선 $u(x) = a + bx$ 가 결정된다.

(예제 6) 4개의 데이터가 아래와 같이 주어질 때 1차회귀곡선을 구하고

상관관계가 어느 정도인가 알아보라.

x_i	1	2	3	4
f_i	2	3	5	9

식(68)을 그대로 이용하면 된다. 주어진 계수를 구하면

$$\sum_{i=1}^{4} x_i = 1+2+3+4 = 10 \qquad \sum_{i=1}^{4} x_i^2 = 1^2+2^2+3^2+4^2 = 30$$

$$\sum_{i=1}^{4} f_i = 2+3+5+9 = 19 \qquad \sum_{i=1}^{4} x_i f_i = 1(2)+2(3)+3(5)+4(9) = 59$$

이므로

$$\begin{cases} 4a + 10b = 19 \\ 10a + 30b = 59 \end{cases} \Rightarrow \quad a = -1, \quad b = 2.3 \tag{69}$$

을 얻고, 따라서 1차회귀는

$$f(x) = -1 + 2.3x \tag{70}$$

```
#> .polyfit ( [ 1,2,3,4 ], [ 2,3,5,9 ], 1);  // '.polyfit(x,f, kth)'
    ans = poly( -1  2.3 )
        = 2.3x -1
```

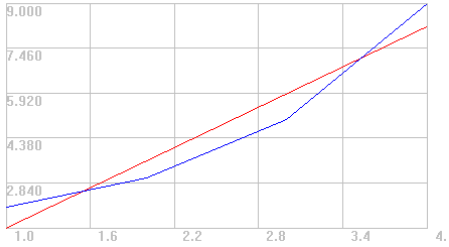
으로 된다. 그리고

$$\begin{aligned} f_{ave} &= \frac{1}{4}(2+3+5+9) = 4.75 \\ S_0 &= (2-4.75)^2 + (3-4.75)^2 + (5-4.75)^2 + (9-4.75)^2 \\ &= 28.75 \\ S &= (2-1.3)^2 + (3-3.6)^2 + (5-5.9)^2 + (9-8.2)^2 = 2.3 \\ r &= \sqrt{\frac{S_0 - S}{S_0}} = \left(\frac{28.75-2.3}{28.75}\right)^2 = 0.959166 \end{aligned} \tag{71}$$

이므로 상관계수 $r = 0.959166$ 은 비교적 1에 가까우므로 상관관계는 우수하다.

☑ cemm# code

```
#> x = [ 1,2,3,4 ];  f = [ 2,3,5,9 ];  I = .ones(1,4);
    x =  [         1         2         3         4 ]
    f =  [         2         3         5         9 ]
    I =  [         1         1         1         1 ]

#> A = [ I**I, I**x; x**I, x**x ] ;  b = [ I**f ; x**f ] ;
    A =
 [         4        10 ]
 [        10        30 ]
    b =
 [        19 ]
 [        59 ]

#> p = poly( A \ b ) ;
    p = poly( -1  2.3 )
        = 2.3x -1

#> fave = f.mean1;          fave =            4.75
#> S0 = (f-fave).norm22;      S0 =           28.75
#> S = (f-p(x)).norm22;        S =             2.3
#> r = sqrt( (S0-S)/S0 );      r =       0.9591663
#> .corrcoef(f,p(x));  // built-in function for correlation coefficient
    ans =       0.9591663

#> p.plot(1,4);
#> plot+( x,f );
```



■ 행렬표현에 의한 일반화. 위에서 구한 1차회귀는 비교적 계산이 간단한 경우이지만 데이터가 많아지고 기저함수의 개수가 증가하면 계산이 쉽지 않게 된다. 따라서 행렬을 이용하여 회귀과정을 일반화하기로 한다.

물론 식(63)을 바로 행렬형태로 나타내면 되지만 논의를 간단히 하기 위해서 1차회귀의 결과로부터 일반화하기로 한다.

먼저, 식(68)로 주어진 결과를

$$\begin{bmatrix} \sum_{i=1}^{n} 1 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} f_i \\ \sum_{i=1}^{n} x_i f_i \end{bmatrix} \tag{72}$$

와 같이 행렬형태로 나타낸다. 그러면 위의 행렬에서 각 원소는

$$\begin{bmatrix} 1 & 1 & \cdots & \cdots & 1 \\ x_1 & x_2 & \cdots & \cdots & x_n \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & \cdots & 1 \\ x_1 & x_2 & \cdots & \cdots & x_n \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} \tag{73}$$

으로부터 결정된 것임을 쉽게 알 수 있다. 그런데 위의 경우는 사용된 기저함수가 $u_1 = 1, u_2 = x$이므로, 일반적인 기저함수 $u_1, u_2, \ldots, u_r$의 경우로 확장하면

$$\begin{aligned} &\begin{bmatrix} u_1(x_1) & u_1(x_2) & \cdots & \cdots & u_1(x_n) \\ u_2(x_1) & u_2(x_2) & \cdots & \cdots & u_2(x_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_r(x_1) & u_r(x_2) & \cdots & \cdots & u_r(x_n) \end{bmatrix} \begin{bmatrix} u_1(x_1) & u_2(x_1) & \cdots & u_r(x_1) \\ u_1(x_2) & u_2(x_2) & \cdots & u_r(x_2) \\ u_1(x_3) & u_2(x_3) & \cdots & u_r(x_3) \\ \vdots & \vdots & \vdots & \vdots \\ u_1(x_n) & u_2(x_n) & \cdots & u_r(x_n) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_r \end{bmatrix} \\ &= \begin{bmatrix} u_1(x_1) & u_1(x_2) & \cdots & \cdots & u_1(x_n) \\ u_2(x_1) & u_2(x_2) & \cdots & \cdots & u_2(x_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_r(x_1) & u_r(x_2) & \cdots & \cdots & u_r(x_n) \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} \end{aligned} \tag{74}$$

으로 나타날 것이다. 따라서 $r \times n$ 행렬 $\mathbf{U}$, $r \times 1$ 행렬 $\mathbf{a}$, $n \times 1$ 행렬 $\mathbf{f}$ 를

$$\mathbf{U} = \begin{bmatrix} u_1(x_1) & u_1(x_2) & \cdots & \cdots & u_1(x_n) \\ u_2(x_1) & u_2(x_2) & \cdots & \cdots & u_2(x_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_r(x_1) & u_r(x_2) & \cdots & \cdots & u_r(x_n) \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_r \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{bmatrix} \tag{75}$$

와 같이 정의하면

$$\mathbf{U}\mathbf{U}^T\mathbf{a} = \mathbf{U}\mathbf{f} \tag{76}$$

인 행렬표현을 얻게 된다(행렬 $\mathbf{U}\mathbf{U}^T$ 는 대칭행렬이다). 따라서 회귀의 과정은 주어진 데이터 $\mathbf{x},\mathbf{f}$ 를 식(76)에 대입하고 계수벡터 $\mathbf{a}$ 에 대해서 풀면 식(60)으로 주어지는 회귀곡선을 얻는 것이다.

☑ 식(75)에 정의된 U항은 cemm#에서 함수 업그레이드로 처리한다.

```
#> matrix uterm(double x) = [ u1(x); u2(x); … ; ur(x) ];
#> uterm++(  [x1,x2, … , xn ] );
```

위와 같이 double을 인수로 하고 matrix를 리턴하는 함수가 matrix를 인수로 할 수 있도록 업그레이드 하는 경우는 Kronecker의 곱과 유사하게 처리된다. 예를 들어,

```
#> matrix ut(double x) = [ x; 1; x^2 ];
#> ut( 3 );
    ans =
 [            3 ]
 [            1 ]
 [            9 ]
#> ut++( [ 3,4,5,6 ] );   // function upgrade by postfix ++
    ans =
 [            3      4      5      6 ]
 [            1      1      1      1 ]
 [            9     16     25     36 ]
```

■ 다항식 및 비선형회귀. 다항식회귀는 x 의 거듭제곱 $1,x,x^2,\cdots,x^r$을 기저함수로 하여 회귀곡선을

$$u(x) = a_0 + a_1x + a_2x^2 + \cdots + a_rx^r \tag{77}$$

으로 두는 것을 말한다. 이것을 식(76)과 연관하여 쓰면 다항식회귀는

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ x_1 & x_2 & x_3 & \cdot & x_{n-1} & x_n \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x_1^r & x_2^r & x_3^r & \cdot & x_{n-1}^r & x_n^r \end{bmatrix} \tag{78}$$

와 같이 $(r+1)\times n$ Vandermonde 행렬 $\mathbf{U}$ 를 이용하여 정의된다.

다항식회귀에서 차수가 높아지면 행렬이 불량조건이 될 수 있고 마무리오차도 커지는 등 그다지 좋은 결과를 얻기 힘들다. 따라서 다항식 회귀는 4차 또는 5차 정도까지만 사용하는 것이 보통이다.

(예제 7) 앞의 (예제 6)에 대해서 2차 및 3차회귀곡선을 구하라.

앞의 예제로부터 데이터는

x_i	1	2	3	4
f_i	2	3	5	9

으로 주어지므로 2차회귀에 대해서

$$\mathbf{U}\mathbf{U}^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ x_1^2 & x_2^2 & x_3^2 & x_4^2 \end{bmatrix} \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix}$$
$$= \begin{bmatrix} 4 & 10 & 30 \\ 10 & 30 & 100 \\ 30 & 100 & 354 \end{bmatrix}$$

이다. 계속해서 식(76)은

$$\begin{bmatrix} 4 & 10 & 30 \\ 10 & 30 & 100 \\ 30 & 100 & 354 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 5 \\ 9 \end{bmatrix} = \begin{bmatrix} 19 \\ 59 \\ 203 \end{bmatrix} \tag{79}$$

와 같게 된다. 이것을 풀면 2차회귀곡선

$$f(x) = a_0 + a_1 x + a_2 x^2 = 2.75 - 1.45x + 0.75x^2 \tag{80}$$

을 얻는다. 3차회귀곡선에 대해서도 같은 과정을 반복하면

$$\begin{bmatrix} 4 & 10 & 30 & 100 \\ 10 & 30 & 100 & 354 \\ 30 & 100 & 354 & 1300 \\ 100 & 354 & 1300 & 4890 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 19 \\ 59 \\ 203 \\ 737 \end{bmatrix} \tag{81}$$

으로부터

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^4 = \frac{1}{6}(6 + 8x - 3x^2 + x^3) \tag{82}$$

을 얻는다. 한편 상관관계를 구하기 위해서 $f_{ave} = 4.75, S_0 = 28.75$ 을 이용하여 식(64)로 정의된 상관계수를 구하면 아래와 같다.

	linear	quadratic	cubic
S	2.3	0.05	0
r	0.959166	0.99913	1

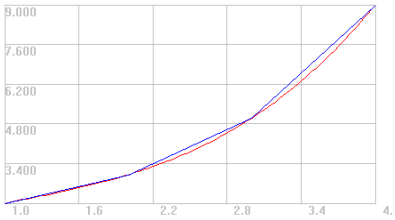
☑ 실제로 주어진 데이터가 4개이므로 3차회귀곡선은 모든 점을 통과하는 3차다항식과 동일하기 때문에 $r = 1$이 된 것이다. cemm#에 내장된 도트함수 '.polyfit(A,B,kth)' 을 이용하면 아래와 같다.

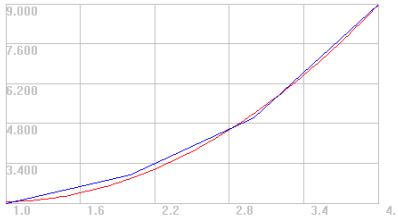
```
#> x = [            1            2            3            4 ];;
#> f = [            2            3            5            9 ];;
#> .polyfit(x,f,1);   // dot function '.polyfit(A,B,kth)'
    ans = poly( -1  2.3 )
        = 2.3x -1

#> .polyfit(x,f,2);
    ans = poly( 2.75  -1.45  0.75 )
        = 0.75x^2 -1.45x +2.75

#> .polyfit(x,f,3);
    ans = poly( 1  1.33333  -0.5  0.166667 )
        = 0.166667x^3 -0.5x^2 +1.33333x +1
```

함수를 스스로 작성하여 문제를 해결하는 경우에는 다음과 같이 cemm# code를 작성할 수 있다.

```
%> UU'a = Uf,  polynomial regression
poly polreg(matrix &x,&f, double n) {
  U = x.vander(n);
  p = poly((U*U') \ (U*f'))  ;    // from matrix to poly
  return  p;
}
#> x = [ 1,2,3,4 ];  f = [ 2,3,5,9 ];
for.n(4,3,-1) {
  cut;
  p = polreg(x,f,n);  // correlation coefficient
  .corrcoef( f,p(x));
  p.plot(1,4);
  plot+(x,f) ;
}
```

```
    x =  [         1         2         3         4 ]
    f =  [         2         3         5         9 ]
-------------------------------------------------------------
    a = poly( 1  1.33333  -0.5  0.166667 )
        = 0.166667x^3 -0.5x^2 +1.33333x +1
    ans =                1
-------------------------------------------------------------
    a = poly( 2.75  -1.45  0.75 )
        = 0.75x^2 -1.45x +2.75
    ans =       0.99913006
```





(예제 8) 앞의 (예제 6)에 대해서 기저함수 $u_1(x) = 1, u_2 = e^{0.4x}$ 를 이용하여 회귀곡선을 구하라.

앞의 예제로부터 데이터는

x_i	1	2	3	4
f_i	2	3	5	9

```
#> x = [ 1,2,3,4 ];  f = [ 2,3,5,9 ];
    x =  [         1         2         3         4 ]
    f =  [         2         3         5         9 ]
```

으로 주어지므로 두 개의 기저함수 $u_1(x) = 1, u_2 = e^{0.4x}$ 를 가지는 비선형회귀에 대해서

$$\mathbf{U} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ e^{0.4x_1} & e^{0.4x_2} & e^{0.4x_3} & e^{0.4x_4} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1.491825 & 2.225541 & 3.320117 & 4.953032 \end{bmatrix}$$

```
#> U = [ .ones(1,4); exp(0.4*x) ];
    U =
 [         1         1         1         1 ]
```

```
[    1.49182    2.22554    3.32012    4.95303 ]
```

임을 이용하면

$$\mathbf{U}\mathbf{U}^T = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1.491825 & 2.225541 & 3.320117 & 4.953032 \end{bmatrix} \begin{bmatrix} 1 & 1.491825 \\ 1 & 2.225541 \\ 1 & 3.320117 \\ 1 & 4.953032 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 11.990515 \\ 11.990515 & 42.734280 \end{bmatrix}$$

```
#> U*U' ;
    ans =
 [         4    11.9905 ]
 [    11.9905    42.7343 ]
```

을 얻는다. 이것을 이용하여 식(76)을 계산하면

$$\begin{bmatrix} 4 & 11.990515 \\ 11.990515 & 42.734280 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 19 \\ 70.838149 \end{bmatrix} \tag{83}$$

```
#> U*f' ;
ans =
 [         19 ]
 [    70.8381 ]
```

으로부터

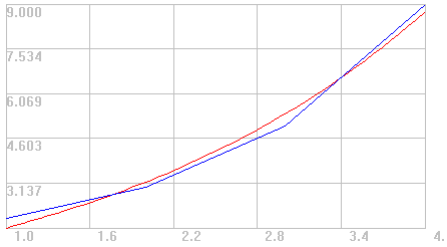
$$f(x) = -1.378061 + 2.044303e^{0.4x} \tag{84}$$

```
#> xcoef = (U*U') \ (U*f');
    xcoef =
 [  -1.37806 ]
 [     2.0443 ]
```

인 회귀곡선을 얻는다. 추가로 상관계수를 구하면 $r = 0.99357$ 로 비교적 우수한 회귀곡선임을 알 수 있다.

```
#> F = xcoef' * U ;  // interpolated values
    F =  [    1.67168    3.17162    5.40926    8.74744 ]

#> .corrcoef(f,F) ;  // correlation coefficient
    ans =       0.99357009

#> x = (1,4).span(50);;
#> plot[x] ( -1.378061 + 2.044303*exp(0.4*x) );
```

```
#> plot+( x,f );
```



☑ cemm# code

```
#> x = [ 1,2,3,4 ];  f = [ 2,3,5,9 ];
    x =  [            1            2            3            4 ]
    f =  [            2            3            5            9 ]

#> matrix mftn(double x) = [ 1 ; exp(0.4*x) ];  // base functions
#> a = .regress(x,f, mftn);                    // built-in dot function
    a =  [      -1.37806        2.0443 ]

#> mftn++(x);   //  matrix(double) can be upgraded by ++ postfix
    ans =
 [            1            1            1            1 ]
 [      1.49182      2.22554      3.32012      4.95303 ]

#> F = a * mftn++(x);     // fitted values
    F =  [       1.67168       3.17162       5.40926       8.74744 ]

#> .corrcoef(f,F);           // correlation coefficient
    ans =      0.99357009
```

★ 비선형 적합곡선. 주어진 자료를 기저함수에 대한 선형결합

$$y = c_1u_1 + c_2u_2 + \cdots + c_nu_n \tag{85}$$

의 형태보다는 비선형 적합곡선

$$y = \alpha e^{\beta x}, \quad y = \alpha x^{\beta}, \quad y = \frac{\alpha}{x+\beta} \tag{86}$$

을 이용하는 것이 편리할 때가 있다. 이러한 경우에는 좌표변환을 먼저 수행하여야 한다. 지수함수

$$y = \alpha e^{\beta x}$$

의 경우

$$\ln y = \ln \alpha + \beta x \quad \Rightarrow \quad z = \ln y, \quad a_1 = \ln \alpha, \quad a_2 = \beta \tag{87}$$

으로 변형하면

$$z = a_1 + a_2 x$$

인 형태의 1차회귀곡선을 구하는 문제로 변환된다. 또한 거듭제곱함수

$$y = \alpha x^{\beta}$$

의 경우에는

$$\ln y = \ln \alpha + \beta \ln x \quad \Rightarrow \quad z = \ln y, \; u = \ln x, \quad a_1 = \ln \alpha, \; a_2 = \beta$$

으로 변형하면 된다. 마지막으로 분수함수

$$y = \frac{\alpha}{x + \beta}$$

의 경우에는 먼저

$$xy + \beta y = \alpha \quad \Rightarrow \quad u = xy, \quad u + \beta y = \alpha \tag{88}$$

와 같이 변수변환을 한 다음

$$y = \frac{\alpha}{\beta} - \frac{1}{\beta} u \quad \Rightarrow \quad \alpha_1 = \frac{\alpha}{\beta}, \; \alpha_2 = -\frac{1}{\beta} \tag{89}$$

으로 쓰면 마찬가지로 1차회귀곡선을 구하는 문제로 변환된다.

아래에서는 위의 방법을 이용하여 비선형회귀곡선을 구해보기로 한다. 자료는 몇 개의 점에 대한 $\tan^{-1} x$의 값으로 아래와 같이

i	x_i	$f_i = f(x_i)$
1	0.5	0.463648
2	0.6	0.540420
3	0.7	0.610726
4	0.8	0.674741

5	0.9	0.732815

으로 주어진다. 지수함수 $y = \alpha e^{\beta x}$인 형태를 이용하는 경우에는

```
#> x = 0.5 : 0.1 : 0.9;
#> f = [ 0.463648, 0.54042, 0.610726, 0.674741, 0.732815 ];
#> z = log(f);
    x =  [         0.5         0.6         0.7         0.8         0.9 ]
    f =  [    0.463648     0.54042    0.610726    0.674741    0.732815 ]
    z =  [    -0.76863   -0.615409   -0.493107   -0.393426   -0.310862 ]

#> a = .polyfit(x,z, 1);   // dot function, equal to  .regress(x,z,1)
    a = poly( -1.31255  1.13752 )
        = 1.13752x -1.31255

#> a[0]= exp(a[0]);;  a;  // coefficients
    a = poly( 0.269133  1.13752 )
        = 1.13752x +0.269133
```

으로 cemm#을 작성하고 실행하면 회귀곡선

$$f(x) = 0.269133e^{1.137520x}$$

을 얻는다. 상관관계는 $r = 0.992591$으로 매우 우수하다.

5-6 연습문제

【1】4개의 데이터가 아래와 같이 주어져 있다

i	x_i	f_i
0	1	1
1	3	4
2	6	3
3	7	2

3차식을 이용한 라그랑주 보간으로 $x = 5$에서의 보간값을 구하라.

【2】 $f(x) = \sin(x)$ 에 대한 데이터가

i	x_i	$f_i = f(x_i)$
0	0.1	0.099833
1	0.2	0.198669
2	0.4	0.389418
3	0.5	0.479426
4	0.8	0.717356

으로 주어져 있다. 전진차분표를 만들어라.

【3】(연습문제 2)에서 구한 전진차분표를 이용하여 $x = 0.3$ 에서의 보간값을 아래의 경우에 대해서 구하고, $\sin(0.3)$과 값을 비교하라.
(a) $i = 1$ 을 시작점으로 하여 두 개의 데이터를 이용 (1차식)
(b) $i = 1$ 을 시작점으로 하여 세 개의 데이터를 이용 (2차식)
(c) $i = 1$ 을 시작점으로 하여 네 개의 데이터를 이용 (3차식)

【4】등간격 $h = 0.1$ 으로 배치된 $x_i = 0, 0.1, 0.2, 0.3, 0.4, 0.5$ 에 대해서 $f(x) = \cosh(x)$의 값이

i	x_i	$f_i = f(x_i)$
0	0.0	1
1	0.1	1.005004
2	0.2	1.020067
3	0.3	1.045339
4	0.4	1.081072
5	0.5	1.127626

형태의 자료로 주어진다. 6개의 데이터를 모두 사용하는 다항식보간 (다시 말해서 5차식을 이용)으로 $x = 0.35$ 에서의 보간값을 구하라.

【5】구간 $1.6 \le x \le 4.2$ 에서 5개의 체비쉐프 점을 결정하라. 이 점을 이용하여 $x\sin(x)$ 의 보간식을 구하고 $x = 3$에서의 보간값을 구하라.

【6】 $x_i = 1, 1.1, 1.2, 1.3, 1.4, 1.5$ 으로 주어지는 데이터 점 ($i =$

0,1,2,3,4,5)에 대해서 함수값 $\ln(x)$ 와 도함수값 $1/x$ 에 대한 자료가

i	x_i	f_i	f_i'
0	1.0	0	1
1	1.1	0.095310	0.909091
2	1.2	0.182322	0.833333
3	1.3	0.262364	0.769231
4	1.4	0.336472	0.714286
5	1.5	0.405465	0.666667

으로 주어진다. 그리고, 에르미트 보간을 이용하여 각 소구간의 중점에서의 보간값을 구하고 엄밀해와 비교하라.

【7】 함수 $f(x) = x + e^x$에 대해서

i	x_i	$f_i = f(x_i)$
0	1.2	4.520117
1	1.4	5.455200
2	1.6	6.553032
3	1.8	7.849647

와 같이 자료가 주어져 있다. 자연 스플라인보간을 하여 $x = 1.3,1.5,1.7$ 에서의 보간값을 각각 구하라.

【8】 함수 $\tan^{-1} x$에 대해서

i	x_i	$f_i = f(x_i)$
1	1	0.785398
2	2	1.107149
3	3	1.249046
4	4	1.325818
5	5	1.373401
6	6	1.405648

와 같이 주어지는 자료가 있다. 세 개의 기저함수 $1, x, x^2$ 을 이용하는

회귀곡선 $y = a_0 + a_1 x + a_2 x^2$ 을 구하라. 상관관계는 어느 정도인가.

【9】 아래에 주어진 데이터를 이용하여 $y = \alpha x^{\beta}$, $y = \frac{\alpha}{x+\beta}$인 형태로 각각 적합하라.

i	x_i	$f_i = f(x_i)$
1	0.5	0.463648
2	0.6	0.540420
3	0.7	0.610726
4	0.8	0.674741
5	0.9	0.732815

심화논의

【10】 $k = 2$인 경우에 대해서 극한거동 $\lim_{h\to 0} \Delta^k f_i = f^k(x_i)/k!$을 보여라.

(풀이) 임의의 α 에 대해서 $x_2 = x_1 + h, x_3 = x_2 + \alpha h$ 으로 두고 극한을 구한다. 그러면

$$\begin{aligned}
\lim_{h\to 0} \Delta^2 f_1 &= \lim_{h\to 0} \frac{\frac{f_3 - f_2}{x_3 - x_2} - \frac{f_2 - f_1}{x_2 - x_1}}{x_3 - x_1} \\
&= \lim_{h\to 0} \frac{\frac{f[x_1 + (1+\alpha)h] - f(x_1 + h)}{\alpha h} - \frac{f(x_1 + h) - f(x_1)}{h}}{(1+\alpha)h} \\
&= \lim_{h\to 0} \frac{f[x_1 + (1+\alpha)h] - (1+\alpha) f(x_1 + h) + \alpha f(x_1)}{\alpha(1+\alpha)h^2} \\
&= \lim_{h\to 0} \frac{(1+\alpha) f'[x_1 + (1+\alpha)h] - (1+\alpha) f'(x_1 + h)}{2\alpha(1+\alpha)h} \\
&= \lim_{h\to 0} \frac{f'[x_1 + (1+\alpha)h] - f'(x_1 + h)}{2\alpha h} \\
&= \lim_{h\to 0} \frac{(1+\alpha) f''[x_1 + (1+\alpha)h] - f''(x_1 + h)}{2\alpha} \\
&= \frac{1}{2} f''(x_1)
\end{aligned}$$

임을 알 수 있다. 특히 $|x_3 - x_1| \to 0$이 성립하는 한 변수 $\alpha \neq 0$의 값은 극한거동에 영향을 주지 않는다는 것을 알 수 있다.

【11】 구간 $x_{i-1} \le x \le x_i$에서 양쪽 경계에서의 함수값 f_{i-1}, f_i와 2계도함수 f''_{i-1}, f''_i가 주어져 있다. 주어진 조건을 만족하는 다항식 $s(x)$를 구하라.

(풀이) 3차 다항식의 2계 미분은 직선이다. 따라서 양쪽 경계에서 2계도함수 f''_{i-1}, f''_i의 값이 주어져 있다면, $s''(x)$는 라그랑주 보간에 의해서 두 점 (x_{i-1}, f''_{i-1}), (x_i, f''_i)을 지나는 다음의 1차식

$$s''(x) = f''_{i-1}\frac{x - x_i}{x_{i-1} - x_i} + f''_i\frac{x - x_{i-1}}{x_i - x_{i-1}}$$

으로 결정된다. 윗식을 두 번 적분하면 미지의 1차식을 포함하는 아래의 식으로 된다.

$$s_i(x) = f''_{i-1}\frac{(x - x_i)^3}{6(x_{i-1} - x_i)} + f''_i\frac{(x - x_{i-1})^3}{6(x_i - x_{i-1})} + A\frac{x - x_{i-1}}{x_i - x_{i-1}} + B\frac{x - x_i}{x_{i-1} - x_i}$$

여기서, 양쪽 경계에서 함수값 (x_{i-1}, f_{i-1}), (x_i, f_i) 가 알려져 있으므로 적분상수 A, B 는

$$f(x_i) = f_i \quad \Rightarrow \quad f_i = f''_i\frac{(x_i - x_{i-1})^2}{6} + A$$

$$f(x_{i-1}) = f_{i-1} \quad \Rightarrow \quad f_{i-1} = f''_{i-1}\frac{(x_{i-1} - x_i)^2}{6} + B$$

으로부터 구할 수 있고, 적분상수를 대입하면 3차 다항식은 아래와 같다.

$$s_i(x) = f''_{i-1}\frac{(x - x_i)^3}{6(x_{i-1} - x_i)} + \left[\frac{f_{i-1}}{x_{i-1} - x_i} - f''_{i-1}\frac{x_{i-1} - x_i}{6}\right](x - x_i)$$
$$+ f''_i\frac{(x - x_{i-1})^3}{6(x_i - x_{i-1})} + \left[\frac{f_i}{x_i - x_{i-1}} - f''_i\frac{x_i - x_{i-1}}{6}\right](x - x_{i-1})$$

【12】 다음의 두 함수

$$s_i(x) = f''_{i-1}\frac{(x-x_i)^3}{6(x_{i-1}-x_i)} + \left[\frac{f_{i-1}}{x_{i-1}-x_i} - f''_{i-1}\frac{x_{i-1}-x_i}{6}\right](x-x_i)$$
$$+f''_i\frac{(x-x_{i-1})^3}{6(x_i-x_{i-1})} + \left[\frac{f_i}{x_i-x_{i-1}} - f''_i\frac{x_i-x_{i-1}}{6}\right](x-x_{i-1})$$

및

$$s_{i+1}(x) = f''_i\frac{(x-x_{i+1})^3}{6(x_i-x_{i+1})} + \left[\frac{f_i}{x_i-x_{i+1}} - f''_i\frac{x_i-x_{i+1}}{6}\right](x-x_{i+1})$$
$$+f''_{i+1}\frac{(x-x_i)^3}{6(x_{i+1}-x_i)} + \left[\frac{f_{i+1}}{x_{i+1}-x_i} - f''_{i+1}\frac{x_{i+1}-x_i}{6}\right](x-x_i)$$

가 $x = x_i$ 에서 기울기가 연속이 되는 조건을 구하라.

(풀이) 함수 $s_i(x)$ 를 미분하고 $x = x_i$ 를 대입하면

$$s'_i(x_i) = \left[\frac{f_{i-1}}{x_{i-1}-x_i} - f''_{i-1}\frac{x_{i-1}-x_i}{6}\right] + f''_i\frac{x_i-x_{i-1}}{2} + \left[\frac{f_i}{x_i-x_{i-1}} - f''_i\frac{x_i-x_{i-1}}{6}\right]$$

으로 된다. 계속해서 함수 $s_{i+1}(x)$ 를 미분하고 $x = x_i$ 를 대입하면

$$s'_{i+1}(x_i) = f''_i\frac{x_i-x_{i+1}}{2} + \left[\frac{f_i}{x_i-x_{i+1}} - f''_i\frac{x_i-x_{i+1}}{6}\right] + \left[\frac{f_{i+1}}{x_{i+1}-x_i} - f''_{i+1}\frac{x_{i+1}-x_i}{6}\right]$$

을 얻는다. 위의 두 식을 다시 쓰면

$$s'_i(x_i) = -f''_{i-1}\frac{x_{i-1}-x_i}{6} + f''_i\frac{3(x_i-x_{i-1})}{6} + \left[\frac{f_i-f_{i-1}}{x_i-x_{i-1}} - f''_i\frac{x_i-x_{i-1}}{6}\right]$$
$$s'_{i+1}(x_i) = f''_i\frac{3(x_i-x_{i+1})}{6} + \left[-f''_i\frac{x_i-x_{i+1}}{6}\right] + \left[\frac{f_{i+1}-f_i}{x_{i+1}-x_i} - f''_{i+1}\frac{x_{i+1}-x_i}{6}\right]$$

이고, 두 식을 같게 두면 다음을 얻는다.

$$f''_{i+1}(x_{i+1}-x_i) + 2f''_i(x_{i+1}-x_{i-1}) + f''_{i-1}(x_i-x_{i-1})$$
$$= 6\left[\frac{f_{i+1}-f_i}{x_{i+1}-x_i} - \frac{f_i-f_{i-1}}{x_i-x_{i-1}}\right]$$

```
//-------------------------------------------------------------------------
//  end of Lectures
//-------------------------------------------------------------------------
```

5-7 스플라인 보간 (선택)

■ 스플라인 보간의 원리. 스플라인 (spline)은 저차의 부분구간 다항식을 의미하며 각 소구간 $x_{i-1} \le x \le x_i$ 에서 스플라인을 이용하여 데이터를 보간하는 것을 스플라인 보간 (spline interpolation)이라고 한다. 예를 들어 1차 스플라인 보간은 각 구간을 선분으로 연결하는 방법으로 1차 다항식을 이용하는 라그랑주 보간과 동일하다.

3차 스플라인 보간의 시작단계는 소구간 $x_{i-1} \le x \le x_i$의 양쪽 경계에서 2계 도함수 f''_{i-1}, f''_i가 주어져 있다고 가정하는 것이다. 다시 말해서 각 소구간 $x_{i-1} \le x \le x_i$에서

함수값 f_{i-1}, f_i와 2계 도함수값 f''_{i-1}, f''_i

인 4개의 조건이 주어진다면 소구간 전체에 걸쳐서 3차 스플라인 $s_i(x)$ 는

$$\begin{aligned} s_i(x) = f''_{i-1}\frac{(x-x_i)^3}{6(x_{i-1}-x_i)} + \left[\frac{f_{i-1}}{x_{i-1}-x_i} - f''_{i-1}\frac{x_{i-1}-x_i}{6}\right](x-x_i) \\ + f''_i\frac{(x-x_{i-1})^3}{6(x_i-x_{i-1})} + \left[\frac{f_i}{x_i-x_{i-1}} - f''_i\frac{x_i-x_{i-1}}{6}\right](x-x_{i-1}) \end{aligned} \tag{90}$$

와 같이 결정된다 (주어진 4개의 조건으로부터 4개의 계수를 가지는 3차식을 얻는 다소 복잡한 계산이므로 유도과정은 연습문제에서 다룬다). 위와 같은 3차 스플라인을 그림 5-2에 보인 바와 같이 n 개의 모든 구간에서 결정하고 나면 시작단계에서 가정한 2계 도함수 $f''_0, f''_1, f''_2, \cdots, f''_n$을 구하는 것이 스플라인 보간의 두 번째 단계가 된다.

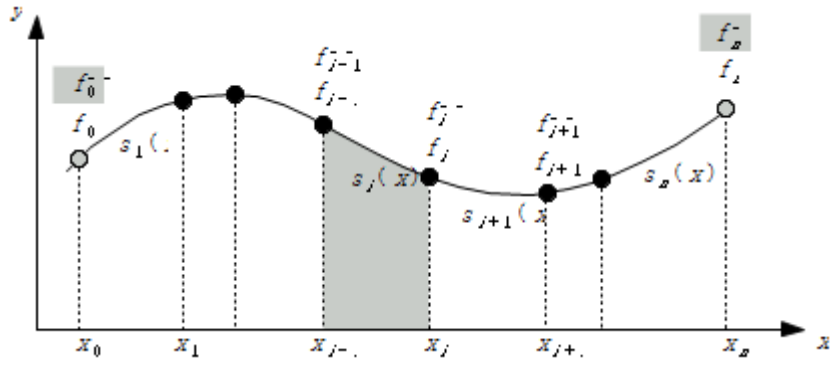


그림 5-2 스플라인 보간

그런데 보간범위 $x_0 \le x \le x_n$ 의 양쪽경계점 x_0, x_n 을 제외한 내부점 $x_i (i = 1,2,3,\dots,n-1)$ 에서는 좌우로 스플라인이 결정되어 있다 (그림 5-2 참조). 따라서 점 $x = x_i$ 에서 좌우의 스플라인 $s_i(x), s_{i+1}(x)$ 의 기울기가 연속인 조건

$$s_i'(x_i) = s_{i+1}'(x_i) \tag{91}$$

을 적용하고 풀면

$$\begin{aligned} f_{i+1}''(x_{i+1} - x_i) + 2f_i''(x_{i+1} - x_{i-1}) + f_{i-1}''(x_i - x_{i-1}) \\ = 6\left[\frac{f_{i+1} - f_i}{x_{i+1} - x_i} - \frac{f_i - f_{i-1}}{x_i - x_{i-1}}\right] \end{aligned} \tag{92}$$

인 관계식을 얻게 된다 (연습문제 참조). 위와 같은 방정식은 내부점에서만 얻을 수 있으므로 총 $n-1$ 개의 방정식이 주어지게 된다. 이것은 그러나 미지의 2계 도함수 $f_0'', f_1'', f_2'', \dots, f_n''$ 는 모두 $n+1$ 개가 있으므로 최종적으로는 양쪽경계 (x_0, x_n)에서 2개의 미지수 f_0'', f_n'' 이 남게 된다. 따라서 스플라인 보간의 마지막 단계는 추가적으로 필요한 2개의 조건, 즉 f_0'', f_n'' 을 결정하는 것이다.

스플라인 보간의 마지막 단계에 들어가기 전에 식(92)를 일단 아래와 같이

$$\begin{aligned} a_i f_i'' + b_i f_{i-1}'' + c_i f_{i+1}'' = d_i, \qquad i = 1,2,3,\dots,n-1 \\ a_i = 2(x_{i+1} - x_{i-1}) = 2(b_i + c_i) \\ b_i = x_i - x_{i-1} \\ c_i = x_{i+1} - x_i \end{aligned} \tag{93}$$

$$d_i = 6\left[\frac{f_{i+1}-f_i}{x_{i+1}-x_i} - \frac{f_i - f_{i-1}}{x_i - x_{i-1}}\right]$$

인 형태로 표준화시키기로 한다.

■ 자연 3차 스플라인. 이름으로부터 알 수 있듯이 가장 자연스럽게 생각되는 조건인

$$f_0'' = f_n'' = 0 \tag{94}$$

을 이용하는 것이다. 이 조건은 곡률이 0이므로 양쪽 경계에서 실제의 곡선보다는 편평하게 보간하는 것을 말한다. 이 경우를 식(93)과 비교하면

$$b_1 = c_{n-1} = 0 \tag{95}$$

인 조건과 동치이므로, 식(93)은 3대각행렬을 가지는 연립 선형방정식으로 되고 따라서 앞에서 공부한 TDMA로 풀 수 있다.

■ $f_0'' = f_1''$, $f_{n-1}'' = f_n''$. 양쪽 경계에서의 2계 도함수가 인접점의 2계 도함수와 같다고 하면 자연 스플라인과는 반대로 양쪽 경계에서 실제의 곡선보다 더 큰 곡률을 가지는 보간으로 된다. 이 경우를 식(93)과 비교하여

$$\begin{aligned} a_1^* &= a_1 + b_1 & b_1^* &= 0 \\ a_{n-1}^* &= a_{n-1} + c_{n-1} & c_{n-1}^* &= 0 \end{aligned} \tag{96}$$

으로 치환하면 마찬가지로 3대각행렬을 얻게 되고 따라서 TDMA로 풀 수 있다 (다른 성분은 변화가 없다).

■ 선형보외에 의한 스플라인. 내부의 2계 도함수를 이용하여 선형보외(linear extrapolation)를 하면

$$\begin{aligned} \frac{f_1'' - f_0''}{x_1 - x_0} = \frac{f_2'' - f_1''}{x_2 - x_1} \quad &\Rightarrow \quad \frac{f_1'' - f_0''}{b_1} = \frac{f_2'' - f_1''}{c_1} \\ \frac{f_n'' - f_{n-1}''}{x_n - x_{n-1}} = \frac{f_{n-1}'' - f_{n-2}''}{x_{n-1} - x_{n-2}} \quad &\Rightarrow \quad \frac{f_n'' - f_{n-1}''}{c_{n-1}} = \frac{f_{n-1}'' - f_{n-2}''}{b_{n-1}} \end{aligned}$$

으로 되고, 이것을 풀면

$$f_0'' = \frac{b_1 + c_1}{c_1} f_1'' - \frac{b_1}{c_1} f_2'', \quad f_n'' = \frac{b_{n-1} + c_{n-1}}{b_{n-1}} f_{n-1}'' - \frac{c_{n-1}}{b_{n-1}} f_{n-2}''$$

이다. 위의 표현을 식(93)에 대입하고 정리하면

$$\begin{aligned} a_1^* &= a_1 + b_1 + \frac{b_1^2}{c_1}, & c_1^* &= c_1 - \frac{b_1^2}{c_1}, & b_1^* &= 0 \\ a_{n-1}^* &= a_{n-1} + c_{n-1} + \frac{c_{n-1}^2}{b_{n-1}}, & b_{n-1}^* &= b_{n-1} - \frac{c_{n-1}^2}{b_{n-1}}, & c_{n-1}^* &= 0 \end{aligned} \tag{97}$$

와 같이 변형된다. 이 경우에도 $b_1^* = c_{n-1}^* = 0$이므로 TDMA로 풀 수 있다.

이 외에도 f_0', f_n' 의 값을 주고 이로부터 f_0'', f_n'' 을 구하는 방법, 주기적인 조건 $f_0' = f_n', f_0'' = f_n''$ 을 이용하는 방법 등이 있다. 이러한 기타의 방법에서도 위와 유사한 과정을 거치면 TDMA 또는 순환TDMA로 f_i''을 구할 수 있다.

```
//-----------------------------------------------------------------------
//  spline interpolation
//-----------------------------------------------------------------------
void spline3(matrix &x,&f,poly *P) {
  // polynomial array must be a priori declared
  n = x.len;
  a = b = c = d = .zeros(1,n);
  for.i(2,n-1) {
    b(i) = x(i)-x(i-1);
    c(i) = x(i+1)-x(i);
    a(i) = 2*(b(i)+c(i));
    d(i) = 6*( (f(i+1)-f(i))/c(i) - (f(i)-f(i-1))/b(i) );
  }
  icase = 0;  // natural condition is default
  if( icase == 1 ) { a(2) += b(2);  a(n-1) += c(n-1); }
  else if( icase == 2 ) {
    t = b(2)^2/c(2);  a(2) += b(2)+t;  c(2) -= t;
    t = c(n-1)^2/b(n-1);  a(n-1) += c(n-1)+t;  b(n-1) -= t;
  }

  f2 = .tdma(a,b,c,d,  2,n-1);   // built-in dot function ".tdma"
  for.i(2,n) {
    h = x(i)-x(i-1);
    P[i-1]
      =-f2(i-1)/(6*h)*.[1,-x(i)  ]^3+(-f(i-1)/h+f2(i-1)*h/6)*.[1,-x(i)]
       +f2(i)  /(6*h)*.[1,-x(i-1)]^3+( f(i  )/h-f2(i  )*h/6)*.[1,-x(i-1)];
  }
```

```
}
```

(예제 9) 4개의 데이터가 아래와 같을 때 3차 자연 스플라인을 구하고, $x = 3.7$에서의 보간값 $f(3.7)$을 구하라.

i	0	1	2	3
x_i	1.5	2.5	4.5	5.5
f_i	3.2	5.4	4.8	7.3

2계 도함수에 대한 첫 번째 방정식을 먼저 유도해본다. 식(93)으로부터

$$b_1 = 2.5 - 1.5 = 1, \quad c_1 = 4.5 - 2.5 = 2$$
$$a_1 = 2(b_1 + c_1) = 6, \quad d_1 = 6\left(\frac{4.8 - 5.4}{4.5 - 2.5} - \frac{5.4 - 3.2}{2.5 - 1.5}\right) = -15$$

을 얻는다. 마찬가지로 두 번째 방정식에 대해서도 계수를 구하면

$$f_0'' + 6f_1'' + 2f_2'' = -15$$
$$2f_1'' + 6f_2'' + f_3'' = 16.8$$

으로 방정식이 결정된다. 계속해서, $f_0'' = f_3'' = 0$ 을 대입하고 풀면

$$f_1'' = -3.8625, \quad f_2'' = 4.0875 \tag{98}$$

을 얻는다. 이것을 이용하면 식(90)으로부터 첫 번째 스플라인 $s_1(x)$은

$$\begin{aligned} s_1(x) &= f_0'' \frac{(x - x_1)^3}{6(x_0 - x_1)} + \left[\frac{f_0}{x_0 - x_1} - f_0'' \frac{x_0 - x_1}{6}\right](x - x_1) \\ &\quad + f_1'' \frac{(x - x_0)^3}{6(x_1 - x_0)} + \left[\frac{f_1}{x_1 - x_0} - f_1'' \frac{x_1 - x_0}{6}\right](x - x_0) \\ &= \left[\frac{f_0}{x_0 - x_1}\right](x - x_1) + f_1'' \frac{(x - x_0)^3}{6(x_1 - x_0)} + \left[\frac{f_1}{x_1 - x_0} - f_1'' \frac{x_1 - x_0}{6}\right](x - x_0) \\ &= -0.6437(x - 1.5)^3 - 3.2(x - 2.5) + 6.0437(x - 1.5) \end{aligned}$$

으로 구해진다. 나머지 스플라인은

$$s_2(x) = 0.3218(x - 4.5)^3 + 0.3406(x - 2.5) \tag{99}$$

$$-3.9875(x-4.5)+1.0375(x-2.5)$$
$$=-19.30313+22.99063x-6.9x^2+0.6625x^3$$
$$s_3(x)=103.1461-58.64219x+11.24063x^2-0.68125x^3$$

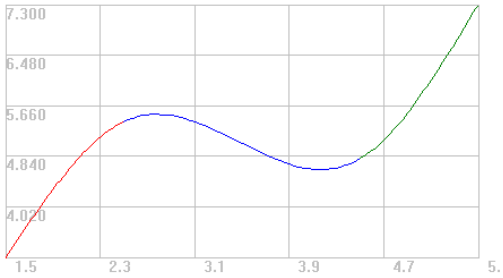
와 같이 스플라인이 각 구간별로 구해진다. 그러면 두 번째 구간에서의 스플라인을 이용하여 $f(3.7)$ 에서의 보간값을 구할 수 있다. 계산하면 다음을 얻는다.

$$s_2(3.7)=0.3218(3.7-4.5)^3+0.3406(3.7-2.5)$$
$$-3.9875(3.7-4.5)+1.0375(3.7-2.5)$$
$$=4.858800$$

```
#> x = [1.5,2.5,4.5,5.5];;
#> y = [3.2,5.4,4.8,7.3];;
#> poly P[4]; spline3(x,y,P);  P;   // user function listed above
    P = poly [4]
 [0] = poly( undefined )
 [1] = poly( 1.10703  -1.50156  2.89688  -0.64375 )
       = -0.64375x^3 +2.89688x^2 -1.50156x +1.10703
 [2] = poly( -19.3031  22.9906  -6.9  0.6625 )
       = 0.6625x^3 -6.9x^2 +22.9906x -19.3031
 [3] = poly( 103.146  -58.6422  11.2406  -0.68125 )
       = -0.68125x^3 +11.2406x^2 -58.6422x +103.146
#> P[2](3.7);
    ans =          4.8588
```

☑ 아래와 같이 cemm# code를 수정하고 실행하면 위의 모든 결과를 얻을 수 있다.

```
#> matrix.format("%12.7g");
#> x = [1.5,2.5,4.5,5.5];
#> y = [3.2,5.4,4.8,7.3];
    x =  [       1.5       2.5       4.5       5.5 ]
    y =  [       3.2       5.4       4.8       7.3 ]
#> P = .spline(x,y);  // 1st and 2nd columns for interval
    P =
 [  1.5       2.5   1.107031 -1.501563  2.896875  -0.64375 ]
 [  2.5       4.5  -19.30313  22.99063      -6.9    0.6625 ]
 [  4.5       5.5   103.1461 -58.64219  11.24063  -0.68125 ]
#> P.splplot;   // .spline(x,y).splplot;
```




```
#> s2 = P..(2)(3:6) .apoly;    // ascending polynomial
    s2 = poly( -19.3031  22.9906  -6.9  0.6625 )
       = 0.6625x^3 -6.9x^2 +22.9906x -19.3031
#> s2(3.7);   // picked up a specific polynomial
    ans =          4.8588
#> P.spl(3.7);  // A.spl(x) first search relevant interval and evaluates
    ans =          4.8588
```

위에서 도트함수 '.spline(x,y)' 가 리턴하는 행렬은 1 열과 2 열은 구간의 시작점과 끝점, 나머지는 3 차다항식의 계수를 포함하는 것에 주목한다. 경계조건을 바꾸는 경우에 따라 세 가지 다른 스플라인을 구하는 방법을 아래에 보인다. 비록 계수는 다르지만 주어진 데이터의 경우에는 결과에 차이가 크지 않다.

```
#> P = .spline(x,y);  // 1st and 2nd columns for interval
    P =
 [  1.5       2.5   1.107031 -1.501563  2.896875  -0.64375 ]
 [  2.5       4.5  -19.30313  22.99063      -6.9    0.6625 ]
 [  4.5       5.5   103.1461 -58.64219  11.24063  -0.68125 ]
#> P1 = .splinebc1(x,y);  // equation (96)
P1 =
 [         1.5         2.5       0.8625  -0.7516667        2.31  -0.5133333 ]
 [         2.5         4.5    -14.15625    18.2975      -5.515        0.53 ]
 [         4.5         5.5       81.495   -46.56333        9.02  -0.5466667 ]
#> P2 = .splinebc2(x,y);  // equation (97)
P2 =
 [         1.5         2.5     0.696875    -0.24375      1.9125      -0.425 ]
 [         2.5         4.5    -10.67813    15.13958     -4.5875   0.4416667 ]
 [         4.5         5.5     67.28437   -38.63542      7.5625  -0.4583333 ]
```

5-8 체비쉐프 보간 (선택)

앞에서 공부한 라그랑주 보간과 뉴턴의 전진.후진보간은 어떤 함수 $f(x)$ 에 대한 데이터가 지정된 점 $x_i,\ i=0,1,2,\ldots,n$ 에서 미리 주어져 있는 경우의 보간으로 이러한 보간은 수동적 (passive)이라고 한다. 반면에 주어진 구간에서 다항식보간을 할 때 오차가 가장 적게 되는 점들을 능동적(active)으로 선택하여 보간하는 경우에는 체비쉐프 보간 (Tchebyshev interpolation)이 매우 효과적이다.

■ 체비쉐프 다항식. 구간 $-1\le x\le 1$에서 정의되는 체비쉐프 다항식 (Tchebyshev polynomial)은 두 가지의 방법으로 표현된다. 첫 번째는

$$T_n(x)=\cos\left(n\cos^{-1}x\right) \tag{100}$$

와 같이 코사인 함수를 이용하고, 두 번째는 멱급수를 이용하여

$$\begin{aligned} T_0(x)&=1 & T_4(x)&=8x^4-8x^2+1\\ T_1(x)&=x & T_5(x)&=16x^5-20x^3+5x\\ T_2(x)&=2x^2-1 & T_6(x)&=32x^6-48x^4+18x^2-1\\ T_3(x)&=4x^3-3x & &\vdots \end{aligned}$$

으로 나타낸다. 고차의 체비쉐프 다항식은

$$T_n(x)=2xT_{n-1}(x)-T_{n-2}(x) \tag{102}$$

으로부터 결정할 수 있다.

■ 체비쉐프 다항식의 성질. 구간 $-1\le x\le 1$에서 정의되는 체비쉐프 다항식은 식(100)으로부터 항상

$$|T_n(x)|\le 1 \tag{103}$$

을 만족한다. 그리고 체비쉐프 다항식의 근 $T_n(x)=0$은

$$\cos(n\cos^{-1}x)=0 \quad\Rightarrow\quad n\cos^{-1}x=\left(n-\frac{1}{2}-k\right)\pi,\quad k=0,1,2,\ldots,n-1$$

즉

$$x_k = \cos\left(n - \frac{1}{2} - k\right)\frac{\pi}{n}, \quad k = 0,1,2,\dots,n-1 \tag{104}$$

으로부터 결정된다. 예를 들어 $n = 3$인 경우의 근은

$$x_0 = -0.866025, \quad x_1 = 0, \quad x_2 = 0.866025 \tag{105}$$

으로 계산된다. 그런데 여기서 한 가지 주의해야 할 것은 체비쉐프 다항식의 차수와 보간에서 사용되는 다항식의 차수가 다르다는 것이다. 왜냐하면 n 차 체비쉐프 다항식의 근은 n 개이므로 보간다항식은 $n-1$차가 되기 때문이다.

이제 체비쉐프 다항식이 함수의 보간에서 왜 중요한가에 대한 설명을 하기로 한다. 일반적으로 n 차 다항식의 최고차항의 계수가 1인 다항식을 일원 (monic) 다항식이라고 하며

$$p_n(x) = x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0 \tag{106}$$

의 형태로 나타낸다. 한편 체비쉐프 다항식을 2^{n-1}로 나누면

$$\frac{1}{2^{n-1}}T_n(x) = x^n + \text{lower terms} \tag{107}$$

와 같이 일원다항식이 되는 것을 알 수 있다. 이때 다음의 minimax 정리가 알려져 있다.

> 차수 n 인 모든 일원다항식에서 구간 $-1 \le x \le 1$에서 가장 작은 최대값을 가지는 일원다항식은 체비쉐프 다항식 $T_n(x)/2^{n-1}$이고 이때의 최대값은 $1/2^{n-1}$이다.

한편 앞에서 논의한 바와 같이 다항식보간의 경우 오차는

$$e(x) = (x - x_0)(x - x_1)(x - x_2)\dots(x - x_n)\frac{f^{n+1}(\xi)}{(n+1)!} \tag{108}$$

으로 평가되는데 오차를 최소로 하기 위해서는 아래의 일원다항식

$$|(x-x_0)(x-x_1)(x-x_2)\dots(x-x_n)| \tag{109}$$

의 절대값을 되도록 작게 만드는 다항식을 선택하여야 한다.

결국 위의 두 가지 논의를 종합하면, 구간 $-1 \le x \le 1$에서 $n+1$ 개의 점을 선택하여 다항식보간을 할 때 오차를 최소로 만드는 다항식은 체비쉐프 다항식이고, 이때 데이터를 결정하는 점 $x_0, x_1, x_2, \dots, x_n$ 은 체비쉐프 다항식의 근으로 선택하면 된다.

그러나, 체비쉐프 다항식은 결국 멱급수의 선형결합으로 표현되므로 체비쉐프 보간은 체비쉐프 다항식의 근이 되는 점 (체비쉐프 점)들에서 데이터를 획득하여 다항식보간을 하는 것으로 정의된다. 다시 말해서, 체비세프 보간은 체비쉐프 점을 이용하여 라그랑주 또는 뉴턴의 보간을 함으로써 주어진 구간에서 보간오차를 최소로 하는 방법이다.

```
//-------------------------------------------------------------------------
//  Tchebyshev interpolation
//-------------------------------------------------------------------------
poly tcheby(double a,b,n, ftn(x)) {
  x = y = .ones(n,1);
  for.k(1,n) {
    z = cos((n+0.5-k)*pi/n);
    x(k) = a+(b-a)/2*(z+1);
    y(k) = ftn(x(k));
  }
  return .interp(x,y);   // interpolation polynomial
}
```

■ 구간의 확장. 보간범위가 $a \le x \le b$ 로 주어질 경우에는

$$x = a + \frac{b-a}{2}(z+1) \tag{110}$$

으로 변환하면 $-1 \le z \le 1$ 로 되므로 체비쉐프 점들을 결정할 수 있다.

(예제 10) 구간 $1 \le x \le 3$에서 3개의 체비쉐프 점을 구하고 $x = 1.8$에서 함수 $f(x) = \tan^{-1} x$ 의 보간값을 구하라.

식(104)와 식(110)을 이용하여 체비쉐프 점을 먼저 구한다. $n = 3, k = 0,1,2$에 대해서

$$z_k = \cos\left(n - \frac{1}{2} - k\right)\frac{\pi}{n}, \quad x_k = a + \frac{b-a}{2}(z_k + 1)$$

즉

$$\begin{aligned} z_0 &= -0.866025 & z_1 &= 0 & z_2 &= 0.866025 \\ x_0 &= 1.133975 & x_1 &= 2 & x_2 &= 2.866025 \end{aligned} \tag{111}$$

와 같이 체비쉐프 점을 구한다. 계속해서 이 점에 대한 함수값을 구하고 전진차분표를 만들면

i	x_i	$\Delta^0 f$	$\Delta^1 f$	$\Delta^2 f$
0	1.133975	0.848098	0.299126	-0.087408
1	2	1.107149	0.147732	
2	2.866025	1.235088		

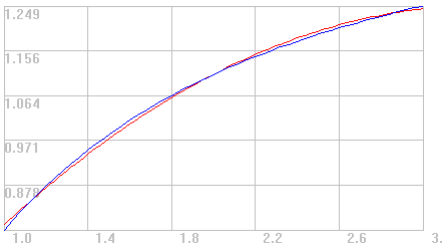
이므로 보간다항식은

$$\begin{aligned} P_2(x) &= \Delta^0 f_0 + (x - x_0)\Delta^1 f_0 + (x - x_0)(x - x_1)\Delta^2 f_0 \\ &= 0.848098 + 0.299126(x - 1.133975) \\ &\quad - 0.087408(x - 1.133975)(x - 2) \\ &= 0.310659 + 0.57306x - 0.087408x^2 \end{aligned} \tag{112}$$

이고 따라서 보간값은 $P_2(1.8) = 1.058967$ 로 된다($\tan^{-1} 1.8 = 1.063698$).

☑ cemm# code

```
#> p = tcheby(1,3, 3,atan);  p(1.8);
    p = poly( 0.310659  0.57306  -0.0874078 )
        = -0.0874078x^2 +0.57306x +0.310659
    ans =        1.0589666

#> plot.x(1,3) ( p(x), atan(x) );  // p(x) in red
```



5-9 에르미트 보간 (선택)

주어진 함수를 보간할 때 함수값 뿐아니라 함수의 기울기가 알려진 경우 주어진 기울기도 이용하여 보간할 필요가 있다. 이러한 목적으로 사용되는 보간을 에르미트 보간이라고 하며, 이 절에서는 응용분야에서 널리 사용되는 에르미트 보간에 대해서 알아본다.

■ 에르미트 보간. 구간 $x_{i-1} \le x \le x_i$ 에서 두 개의 함수값 $f(x_{i-1}), f(x_i)$ 와 두 개의 도함수값 $f'(x_{i-1}), f'(x_i)$ 이 주어진 경우를 생각해보자. 주어진 조건이 4개이므로 보간다항식은 3차가 된다. 그러나 계산을 쉽게 처리하기 위해서 먼저 좌표변환

$$s = \frac{x - x_{i-1}}{x_i - x_{i-1}} = \frac{x - x_{i-1}}{h} \tag{113}$$

을 도입하여 구간의 양끝점이 $s = 0,1$ 이 되도록 한다. 그러면 아래의 3차 다항식

$$\begin{aligned} f(x) &= f[x_{i-1} + s(x_i - x_{i-1})] \\ &= f_{i-1} + (f_i - f_{i-1})s + As(s-1) + Bs^2(s-1) \end{aligned} \tag{114}$$

은 $f(s=0) = f_{i-1}$, $f(s=1) = f_i$ 를 만족한다. 윗식을 s 에 관하여 미분하면

$$hf' = f_i - f_{i-1} + A[(s-1) + s] + B[2s(s-1) + s^2]$$

이고, $s = 0$ 과 $s = 1$ 을 대입하면

$$\begin{aligned} hf'_{i-1} &= f_i - f_{i-1} - A \\ hf'_i &= f_i - f_{i-1} + A + B \end{aligned} \tag{115}$$

으로부터

$$\begin{aligned} A &= f_i - f_{i-1} - hf'_{i-1} \\ B &= h(f'_i + f'_{i-1}) - 2(f_i - f_{i-1}) \end{aligned} \tag{116}$$

을 얻는다. 이것을 대입하고 정리하면

$$f = (1-\alpha)f_{i-1} + \alpha f_i + hs(1-s)[(1-s)f'_{i-1} - sf'_i] \\ \alpha = 3s^2 - 2s^3 \tag{117}$$

인 형태의 3차 에르미트 보간식 (3rd order Hermite interpolaton)이 얻어진다.

```
//---------------------------------------------------------------------------
//  Hermite interpolation
//---------------------------------------------------------------------------
double hermite(xx, matrix &x,&y,&y1) {
  // x : grid points
  // y : y(x)
  // y1 : y'(x)
   k = x.ipos(xx);           // (xx-x(k))*(xx-x(k+1)) < 0
   if( k == 0 ) return NaN;  // out of interval
   h = x(k+1)-x(k);
   s = (xx-x(k))/h;
   a = s*s*(3-2*s);
   yy = (1-a)*y(k)+a*y(k+1)+h*s*(1-s)*( (1-s)*y1(k)-s*y1(k+1) );
   return yy;
}
```

(예제 11) 함수 $f(x) = \tan^{-1}x, f'(x) = 1/(1+x^2)$ 의 값이 구간 $0 \le x \le 2$에서 $h = 0.2$인 등간격으로 주어져 있다. 이때 각 중점에서의 보간값을 3차 에르미트 보간으로 구하고 엄밀해와 비교하라.

$h = 0.2$ 이므로 $x_i = 0.2i,\ i = 0,1,2,3,\dots,10$ 으로 주어져 있다. 보간하는 위치는 $c_i = 0.1, 0.3, 0.5, \dots, 1.9$이다. 먼저 첫 번째 위치 $c_1 = 0.1$에 대해서 3차 에르미트 보간을 한다. 따라서

$$f_0 = \tan^{-1}0 = 0 \quad f_1 = \tan^{-1}0.2 = 0.197396$$
$$f'_0 = \frac{1}{1+0} = 0 \quad f'_1 = \frac{1}{1+0.2^2} = 0.961538$$

으로 함수값을 구하고

$$s = \frac{0.1-0}{0.2} = 0.5, \quad \alpha = 3s^2 - 2s^3 = 0.5$$

이므로

$$\begin{aligned} f &= (1-\alpha)f_{i-1} + \alpha f_i + hs(1-s)[(1-s)f'_{i-1} - sf'_i] \\ &= 0.5(0.197396) + 0.2(0.5)(0.5)[0.5(1) - 0.5(0.961538)] \\ &= 0.099660 \end{aligned} \quad (118)$$

을 얻는다. 나머지 점들에 대해서는 cemm# code 를 작성하고 실행한다.

```
#> x = (0 : 0.2 : 2)';;
#> y = atan(x);; y1 = 1 ./ (1+x.^2);;

#> xm = ym = (0.1: 0.2: 2)';;
#> for.i(1,xm.len) ym(i) = hermite(xm(i), x,y,y1);;

#> [ ym, ym-atan(xm) ];
ans =
 [    0.0996593  -9.3341e-006 ]
 [     0.291438 -1.90886e-005 ]
 [     0.463632 -1.52991e-005 ]
 [     0.610719 -7.29264e-006 ]
 [     0.732813 -1.64654e-006 ]
 [     0.832982  9.38684e-007 ]
 [     0.915102  1.70085e-006 ]
 [     0.982795  1.67692e-006 ]
 [      1.03907  1.40289e-006 ]
 [      1.08632  1.09876e-006 ]
```

x	f	$f_{ex}-f$	x	f	$f_{ex}-f$
0.1	0.099659	0.000009	1.1	0.832982	-0.000001
0.3	0.291438	0.000019	1.3	0.915102	-0.000002
0.5	0.463632	0.000015	1.5	0.982795	-0.000002
0.7	0.610719	0.000007	1.7	1.039074	-0.000001
0.9	0.732813	0.000002	1.9	1.086319	-0.000001

결과를 보면 도함수값이 함께 보간에 참여하였기 때문에 매우 정확한 보간값이 얻어지는 것을 알 수 있다.

5-10 추가적인 논의 (선택)

■ **테일러급수와 보간다항식의 유사성**. 논의에 앞서 분할차분의 극한거동과 차분연산자를 다음과 같이 정리한다.

(119a) $$\lim_{h\to 0}\Delta^k f_i = \lim_{h\to 0}\nabla^k f_i = \frac{f^{(k)}(x_i)}{k!}$$

(119b) $$\lim_{h\to 0}\frac{1}{k!\,h^k}\Delta_*^k f_i = \lim_{h\to 0}\frac{1}{k!\,h^k}\nabla_*^k f_i = \frac{f^{(k)}(x_i)}{k!}$$

계속해서 함수 $f(x)$의 테일러급수를

(120) $$\begin{aligned} f(x) &= f(x_0) + (x-x_0)f'(x_0) + (x-x_0)^2\frac{f''(x_0)}{2!} + \\ &\quad + \cdots + (x-x_0)^n\frac{f^{(n)}(x_0)}{n!} + R_n \\ R_n &= (x-x_0)^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!} \end{aligned}$$

와 같이 다시 쓰고, n차의 보간다항식을

(121) $$\begin{aligned} f(x) &= f_0 + (x-x_0)\Delta^1 f_0 + (x-x_0)(x-x_1)\Delta^2 f_0 \\ &\quad + \cdots + (x-x_0)(x-x_1)\ldots(x-x_{n-1})\Delta^n f_0 + e(x) \\ e(x) &= (x-x_0)(x-x_1)\ldots(x-x_n)\frac{f^{(n+1)}(\xi)}{(n+1)!}, \quad x_0 \le \xi \le x_n \end{aligned}$$

으로 정리한다. 그러면, 테일러급수에서의 거듭제곱 $(x-x_0)^k$ 를 보간다항식에서 선형인수의 곱 $(x-x_0)(x-x_1)\ldots(x-x_{k-1})$ 으로, 테일러급수에서의 k계미분 $f^{(k)}(x_0)/k!$ 을 보간다항식에서 $\Delta^k f_0$에 대응하는 유사성이 관찰된다.

가상적으로 무한개의 점 $x_0, x_1, \ldots, x_n, \ldots, x_\infty$ 가 하나의 점 x_0 으로 수렴하는 경우를 생각해보자 (다시 말해서 $h\to 0$ 인 경우). 그러면 무한차수의 보간다항식 $P_\infty(x)$를 가상적으로 도입할 수 있다. 그러면 이러한 무한차수의 보간다항식은 점 $x=x_0$ 에서의 테일러급수로 접근하는 것을 예상할 수 있다.

또한, 이미 언급하였듯이, 이러한 극한거동을 이용하면 보간에 의한 오차를

$$
\begin{aligned}
|e(x)|_{\max} &= |(x-x_0)(x-x_1)\dots(x-x_n)| \times \left|\frac{f^{(n+1)}(\xi)}{(n+1)!}\right|_{\max} \\
&\cong |(x-x_0)(x-x_1)\dots(x-x_n)| \times \left|\Delta^{k+1} f\right|_{\max} \\
&\cong \left|\frac{s(s-1)(s-2)\dots(s-n)}{(n+1)!}\right| \times \left|\Delta_*^{k+1} f\right|_{\max}
\end{aligned}
\tag{122}
$$

으로 쓸 수 있고 $(x_0 \le \xi \le x_n)$, 이것은 차항규칙 (次項規則, next-term rule)을 의미한다.

테일러급수와 n 차의 뉴턴보간다항식 사이의 유사성은 후진차분의 경우에도 적용된다. 이러한 논의는 비록 이론적이기는 하지만 보간다항식의 특성과 오차해석에서 매우 도움이 된다.

★ **미분과 차분의 유사성**. 미분개념과 차분개념을 비교하기 위해서 먼저 표 5-6 에는 적분과 수열의 합에 대한 잘못된 유사성을 나타내었다. 얼핏 보기에는 $\int x^k\,dx$와 $\sum s^k$ 는 거듭제곱에 대한 적분과 수열의 합이므로 동일한 구조로 생각된다. 그러나 표 5-6 의 결과를 보면 수열의 합에 대한 우변의 결과는 전혀 예측할 수 없는 형태로 나타난다. 다시 말해서 표 5-6 은 단순한 결과의 제시일 뿐 적분과 수열의 합이라는 개념의 유사구조를 나타내지 못한다.

여기서 등간격 h 와 $s = 0,1,2,3,\dots,n$ 인 정수 s 에 대해서 함수 $f(x)$ 를

$$f(x) = x(x+h)(x+2h),\ \ x = sh \tag{123}$$

으로 도입하면

$$
\begin{aligned}
f(x) - f(x-h) &= f(sh) - f[(s-1)h] \\
&= h^3[s(s+1)(s+2) - (s-1)s(s+1)] \\
&= 3h^3 s(s+1)
\end{aligned}
$$

을 얻는다. 윗식에 $s = 1,2,3,\cdots,n$ 을 계속 대입하면

$$
\begin{aligned}
f(h) - f(0) &= [3h^3 s(s+1)]_{s=1} \\
f(2h) - f(h) &= [3h^3 s(s+1)]_{s=2}
\end{aligned}
$$

$$\vdots$$

$$f(nh) - f[(n-1)h)] = [3h^3 s(s+1)]_{s=n}$$

으로 되고, 위의 식을 모두 합하면

Table 5-6 Incorrect analogy between integration and series

integration	series
$\int 1\,dx = x$	$\sum_{s=1}^{n} 1 = n$
$\int x\,dx = \frac{1}{2}x^2$	$\sum_{s=1}^{n} s = \frac{1}{2}n(n+1)$
$\int x^2 dx = \frac{1}{3}x^3$	$\sum_{s=1}^{n} s^2 = \frac{1}{6}n(n+1)(2n+1)$
$\int x^2\,dx = \frac{1}{3}x^3$	$\sum_{s=1}^{n} s^3 = \left[\frac{1}{2}n(n+1)\right]^2$

$$h^3 n(n+1)(n+2) = f(nh) = 3h^3 \sum_{s=1}^{n} s(s+1)$$

즉 h 와는 무관하게

$$\sum_{s=1}^{n} \frac{1}{2!} s(s+1) = \frac{1}{3!} n(n+1)(n+2) \tag{124}$$

인 결과를 얻는다. 그런데 함수 $f(x)$의 미분을 후진차분을 이용하여 구하면

$$\frac{df}{dx} = \lim_{h\to 0} \frac{f(x) - f(x-h)}{h}$$
$$= \lim_{h\to 0} \frac{x(x+h)(x+2h) - (x-h)x(x+h)}{h} = \lim_{h\to 0} \frac{3hx(x+h)}{h} = 3x^2$$

다시 말해서

$$\int x^2 dx = \frac{1}{3}x^3 \tag{125}$$

인 결과가 된다. 결국 적분과 수열의 합의 유사구조는 표 5-7 에 보인 바와 같이 써야 옳게 된다.

Table 5-7 Correct analogy between integration and series

integration	series
$\int 1dx = x$	$\sum_{s=1}^{n} 1 = (s)_{s=n}$
$\int x dx = \frac{1}{2}x^2$	$\sum_{s=1}^{n} s = \frac{1}{2}s(s+1)_{s=n}$
$\int \frac{1}{2!}x^2 dx = \frac{1}{3!}x^3$	$\sum_{s=1}^{n} \frac{1}{2!}s(s+1) = \frac{1}{3!}s(s+1)(s+2)_{s=n}$
$\int \frac{1}{3!}x^3 dx = \frac{1}{4!}x^4$	$\sum_{s=1}^{n} \frac{1}{3!}s(s+1)(s+2) = \frac{1}{4!}s(s+1)(s+2)(s+3)_{s=n}$

이것은

- ○ $\frac{1}{3}x^3$ 의 미분은 x^2
- ○ $\frac{1}{3}s(s+1)(s+2)$ 의 차분은 $s(s+1)$

으로 결정되는 구조를 나타낸다. 그렇다면 미분개념의 테일러급수를 차분개념으로 바꾸려면 위에서 언급한 바와 같은 유사구조를 이용하여 $h = x - x_n$ 이라고 할 때

$$f(x) = f(x_n) + hf'(x_n) + \frac{h^2}{2!}f''(x_n) + \frac{h^3}{3!}f'''(x_n) + \cdots$$
$$P_3(x) = \nabla^0 f_n + s\nabla^1 f_n + \frac{s(s+1)}{2!}\nabla^2 f_n + \frac{s(s+1)(s+2)}{3!}\nabla^3 f_n \tag{126}$$

의 형태를 얻게 된다. 위의 두 식이 가지는 유사구조를 잘 살펴보면 뉴턴의 후진차분식에서 $s(s+1)(s+2)/3!$ 와 같은 형태가 나타나는 이유를 쉽게 이해할 수 있다.

【문제】 다음의 공식을 $q=p+n-1$인 경우에 대해서 증명하라.

$$\sum_{s=a}^{b} \overbrace{(s+p)(s+p+1)\cdots(s+q)}^{n \text{ consecutive integers}} = \left[(s+p)(s+p+1)\cdots(s+q)\frac{s+q+1}{n+1}\right]_{a-1}^{b}$$

주어진 공식은 정적분과 매우 유사하다.

$$\int_a^b x^n dx = \left[\frac{x^{n+1}}{n+1}\right]_a^b$$

다시 말해서, 적분과 수열합에는 다음과 같은 유사성이 존재한다.

$$\int_a^b \underbrace{ⓧⓧ\cdots ⓧ\mathrm{dx}}_{\text{n number of x}} = \left[ⓧⓧ\cdots ⓧ\frac{ⓧ}{n+1}\right]_a^b$$

$$\sum_a^b \underbrace{◇□\cdots ♡}_{\text{n consecutive integers}} = \left[◇□\cdots ♡\frac{♡+1}{n+1}\right]_{a-1}^b$$

먼저, 다음을 정의하면

$$A_s = (s+p)(s+p+1)\cdots(s+q)(s+q+1)$$
$$A_{s-1} = (s+p-1)(s+p)(s+p+1)\cdots(s+q)$$

두 식의 차는

$$\begin{aligned} A_s - A_{s-1} &= (s+p)(s+p+1)\cdots(s+q)[(s+q+1)-(s+p-1)] \\ &= (q-p+2)(s+p)(s+p+1)\cdots(s+q) \\ &= (n+1)(s+p)(s+p+1)\cdots(s+q) \end{aligned}$$

으로 된다. 위에서, $s=a, a+1, \cdots, b-1, b$을 대입하고 모두 더하면

$$
\begin{aligned}
A_b - A_{b-1} &= (n+1)[(s+p)(s+p+1)\cdots(s+q)]_{s=b} \\
A_{b-1} - A_{b-2} &= (n+1)[(s+p)(s+p+1)\cdots(s+q)]_{s=b-1} \\
&\vdots \\
A_{a+1} - A_a &= (n+1)[(s+p)(s+p+1)\cdots(s+q)]_{s=a+1} \\
A_a - A_{a-1} &= (n+1)[(s+p)(s+p+1)\cdots(s+q)]_{s=a}
\end{aligned}
$$

다음의 공식에 도달한다.

$$
\begin{aligned}
\sum_{s=a}^{b}(s+p)(s+p+1)\cdots(s+q) &= \frac{A_b - A_{a-1}}{n+1} \\
&= \left[(s+p)(s+p+1)\cdots(s+q)\frac{s+q+1}{n+1}\right]_{a-1}^{b}
\end{aligned}
$$

수열이 항상 연속된 정수의 곱으로 주어지면 수열의 합은 항상 위에 주어진 공식으로 구할 수 있다. 예를 들어,

$$
\sum_{s=2}^{n}(s+3)(s+4) = \frac{1}{3}[(n+3)(n+4)(n+5) - 4\cdot 5\cdot 6]
$$

$$
\sum_{s=1}^{n}(s-2)(s-1)s = \frac{1}{4}[(n-2)(n-1)n(n+1) - (-2)(-1)(0)(1)]
$$

```
//---------------------------------------------------------------------------
//  end of file
//---------------------------------------------------------------------------
```