Interpolation methods

20120778
Subin Lee

Department of Materials Science & Engineering
Pohang University of Science & Technology

Assignment

- Polynomials (Lagrange method) 와 Cubic spline을 비교
- $y=\ln x$ 그래프 상에 4 개의 포인트가 주어졌을 때 $x=2$ 값을 예측하라.

Interpolation methods

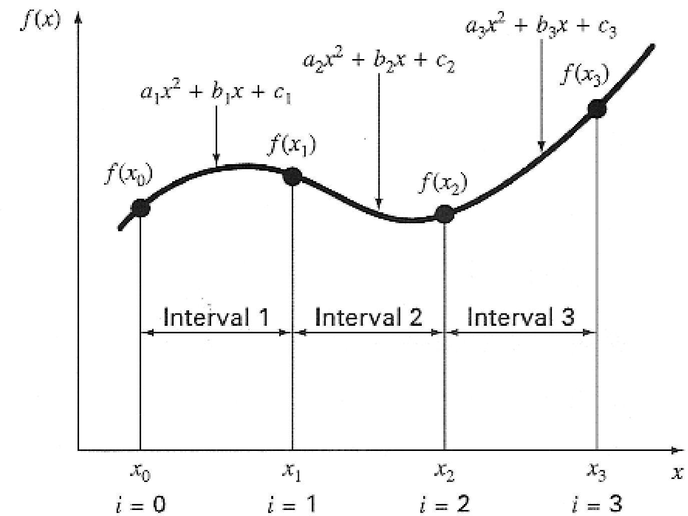
- Lagrange polynomials

$$f_n(x) = \sum_{k=0}^{n} L_{n,k}(x)\ f(x_k)$$

$$L_{n,k}(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

$$L_{n,k}(x) = \frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

- Spline methods



- 3 차 (cubic) 스플라인

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

1. 함수 값은 내부 절점에서 같아야 한다. [$2n$–2]
2. 첫 번째와 마지막 함수는 양 끝점을 통과해야 한다. [2]
3. 내부 절점에서 1 차 도함수는 같아야 한다. [n–1]
4. 내부 절점에서 2 차 도함수도 같아야 한다. [n–1]
5. 양 끝점에서의 2 차 도함수는 0 이라고 가정한다. [2]
   (2 차 도함수가 0 이 아니라면 그 정보로 조건을 대체)

Key codes: cubic spline

1. Sort input arrays

```
double spline (double ary_x[], double ary_y[], int n, double x){
    double temp_x, temp_y, d2y[n], s[n];
    FILE *fp;

    // 순서대로 정렬
    for (int i=0; i<n; i++) {
        for (int j=0; j<n-1; j++) {
            if (ary_x[j]>ary_x[j+1]) {
                temp_x=ary_x[j];
                temp_y=ary_y[j];
                ary_x[j]=ary_x[j+1];
                ary_y[j]=ary_y[j+1];
                ary_x[j+1]=temp_x;
                ary_y[j+1]=temp_y;
            }
        }
    }
    //position 찾기
    int k=0;
    while (ary_x[k]<x) {
        k=k+1;
    }
```

2. Find 2nd derivatives

$$
\begin{aligned}
&(x_i - x_{i-1}) f''(x_{i-1}) + 2(x_{i+1} - x_{i-1}) f''(x_i) + (x_{i+1} - x_i) f''(x_{i+1}) \\
&= \frac{6}{x_{i+1} - x_i}[f(x_{i+1}) - f(x_i)] + \frac{6}{x_i - x_{i-1}}[f(x_{i-1}) - f(x_i)]
\end{aligned}
$$

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

$$
\begin{bmatrix}
1 & & & & & \\
h_0 & 2(h_1 + h_0) & h_2 & & & \\
 & \ddots & \ddots & & & \\
 & & & \ddots & \ddots & \\
 & & & h_{n-1} & 2(h_{n+1} + h_{n-1}) & h_{n+1} \\
 & & & & & 1
\end{bmatrix}
\begin{Bmatrix}
f_0'' \\ f_1'' \\ \vdots \\ \vdots \\ f''_{n-2} \\ f''_{n-1}
\end{Bmatrix}
= \{\cdots\}
$$

```
for (int i=1; i<n-1; i++){
    matrix[i][n]=6*((ary_y[i+1]-ary_y[i])/(ary_x[i+1]-ary_x[i])+(ary_y[i-1]-ary_y[i])/
        (ary_x[i]-ary_x[i-1]));
}

scaled_pivot(matrix, n);
elimination(matrix, n);
substitution(matrix, n);

for (int i=0; i<n; i++) {
    d2y[i]=matrix[i][n];
}
```

3. Calculate $f_i(x)$

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

```
for (int i=1; i<n; i++) {
    double h=ary_x[i]-ary_x[i-1];

    s[i]=d2y[i-1]*(ary_x[i]-x)*(ary_x[i]-x)*(ary_x[i]-x)/(6*h)+
    d2y[i]*(x-ary_x[i-1])*(x-ary_x[i-1])*(x-ary_x[i-1])/(6*h)+
    (ary_y[i-1]/h-d2y[i-1]*h/6)*(ary_x[i]-x)+
    (ary_y[i]/h-d2y[i]*h/6)*(x-ary_x[i-1]);
```

Key codes: Lagrange polynomials

```
//***lagrange 함수
double lagrange (double ary1[], double ary2[], int n, double x){
    double sum=0.0;
    for (int i=0; i<n; i++){
        for (int j=0; j<n; j++){
            if (i !=j){
                ary2[i]=((x-ary1[j])/(ary1[i]-ary1[j]))*ary2[i];
            }
        }
        sum=sum+ary2[i];
    }
    return sum;
}
```

$$f_n(x) = \sum_{k=0}^{n} L_{n,k}(x)\ f(x_k)$$

$$L_i(x) = \prod_{j=1,\, j\neq i}^{n+1} \frac{x - x_j}{x_i - x_j}$$

$$= \frac{(x-x_1)(x-x_2)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_{n+1})}{(x_i-x_1)(x_i-x_2)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{n+1})}$$

Results and discussion

Example) x = 1, 4, 6 에서의 값에 기반을 둔 2 차 보간법을 사용하여 ln2 값을 구하라.

ln 1 = 0

ln 4 = 1.386294

ln 6 = 1.791759

$$f_2(x) = 0 + 0.4620981(x-1) + -0.0518731(x-1)(x-4)$$

$$= 0.5658444$$

```
Number of data points
3
Insert x values
1 4 6
Insert x:
2
The result by Lagrange: 0.565844346900983 (18.37% error)
The result by cubic spline: 0.531262271391754 (23.36% error)
```

```
Number of data points
4
Insert x values
1 4 6 7
Insert x:
2
The result by Lagrange: 0.613416553818629 (11.50% error)
The result by cubic spline: 0.531575722728613 (23.31% error)
```

```
Number of data points
5
Insert x values
1 4 6 7 9
Insert x:
2
The result by Lagrange: 0.636453046962152 (8.18% error)
The result by cubic spline: 0.532051488568178 (23.24% error)
```

```
Number of data points
10
Insert x values
1 4 6.2 7.1 8.3 9 10 11.2 12.0 13
Insert x:
2
The result by Lagrange: 0.677902024329512 (2.20% error)
The result by cubic spline: 0.530616383804913 (23.45% error)
```

- As increase initial data points, error decreases for only Lagrange methods

Results and discussion

```
Number of data points
4
Insert x values
1 4 6 7
Insert x:
2
The result by Lagrange: 0.613416553818629 (11.50% error)
The result by cubic spline: 0.531575722728613 (23.31% error)
```

```
Number of data points
4
Insert x values
1.4 3 6 7
Insert x:
2
The result by Lagrange: 0.668882755785609 (3.50% error)
The result by cubic spline: 0.644389004220410 (7.03% error)
```

```
Number of data points
4
Insert x values
1.4 3.5 6 7
Insert x:
2
The result by Lagrange: 0.661017007653706 (4.64% error)
The result by cubic spline: 0.625959279617199 (9.69% error)
```

```
Number of data points
4
Insert x values
1.4 2.6 6 7
Insert x:
2
The result by Lagrange: 0.676838258314518 (2.35% error)
The result by cubic spline: 0.662044612998852 (4.49% error)
```

- As the initial data points near the unknown point become close, errors effectively decrease in both case

```
Number of data points
10
Insert x values
1 3 4 6 8 7 2 5.2
4.3
7.9
Insert x:
3.7
The result by Lagrange: 149.724378964754521 (0.05% error)
The result by cubic spline: 148.625797777921377 (0.69% error)
```

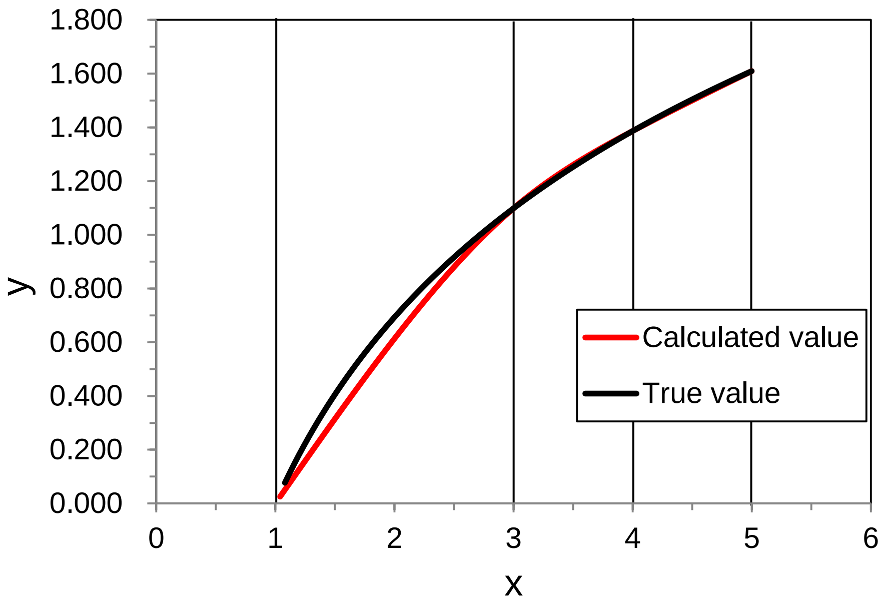
$$y = x\, exp(x)$$

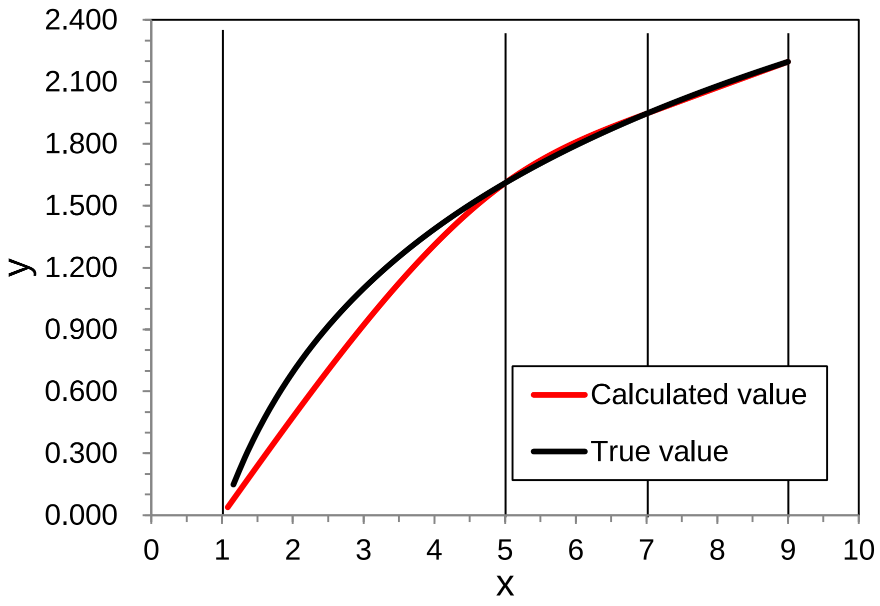
```
Number of data points
5
Insert x values
1 5 4 6 8
Insert x:
3
The result by Lagrange: 3.307274211609648 (0.35% error)
The result by cubic spline: 3.433409657268317 (4.17% error)
```

$$y = x\, log(x)$$

- For most cases Lagrange method showed better accuracy

Results and discussion





- For cubic spline methods the differences between data points are key factors