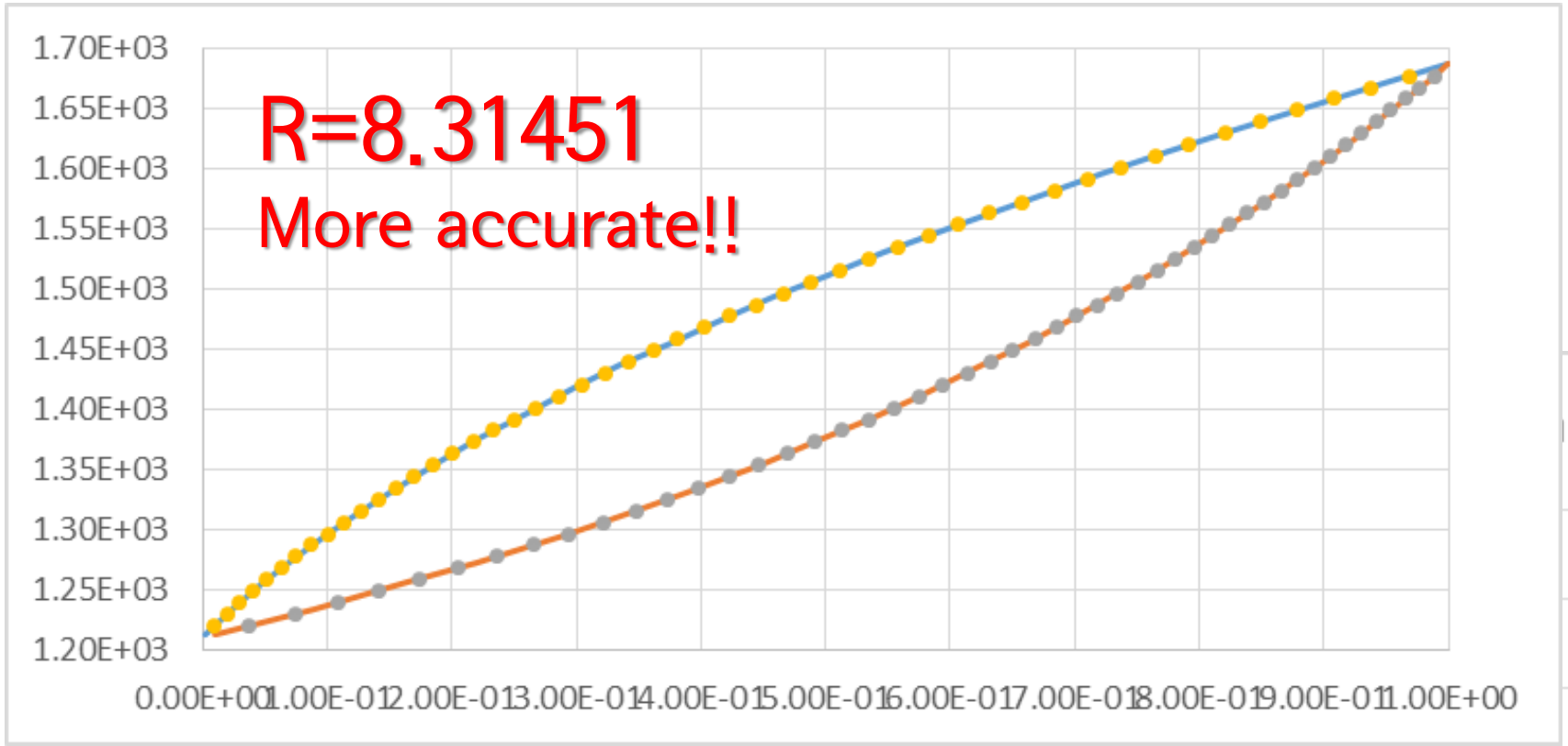
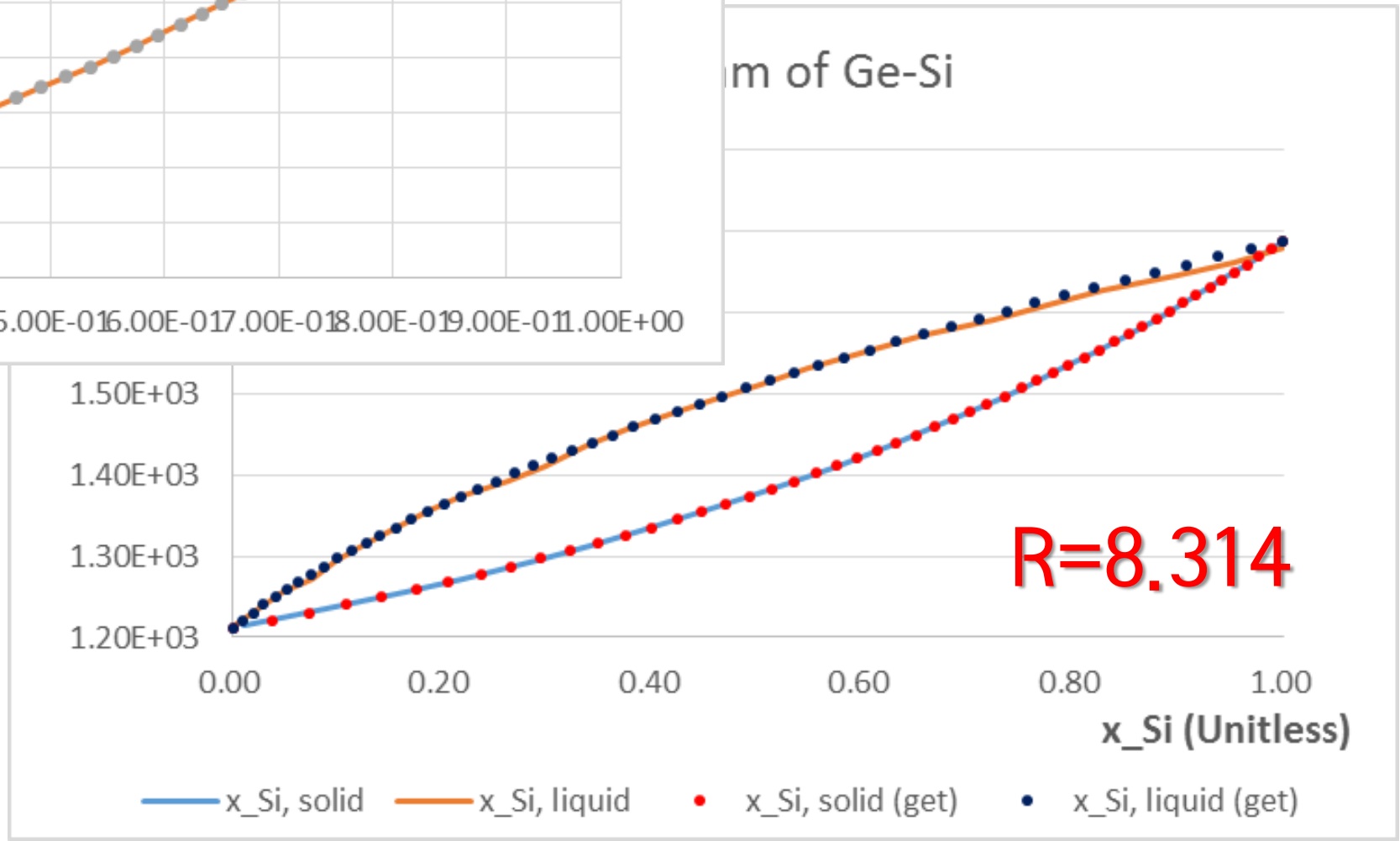
Numerical Analysis For Materials

Homework #5

Gilwoon Lee

ID: 20120083
Dept: Material Science and Engineering


R=8.31451
More accurate!!
1.70E+03
1.65E+03
1.60E+03
1.55E+03
1.50E+03
1.45E+03
1.40E+03
1.35E+03
1.30E+03
1.25E+03
1.20E+03
0.00E+00
1.00E-01
2.00E-01
3.00E-01
4.00E-01
5.00E-01
6.00E-01
7.00E-01
8.00E-01
9.00E-01
1.00E+00


m of Ge-Si
R=8.314
1.50E+03
1.40E+03
1.30E+03
1.20E+03
0.00
0.20
0.40
0.60
0.80
1.00
x_Si (Unitless)
x_Si, solid
x_Si, liquid
x_Si, solid (get)
x_Si, liquid (get)

Homework #5

Interpolation for given value

1. Idea to construct Interpolation - Cubic spline

Given data

주어진 B.C 조건

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]

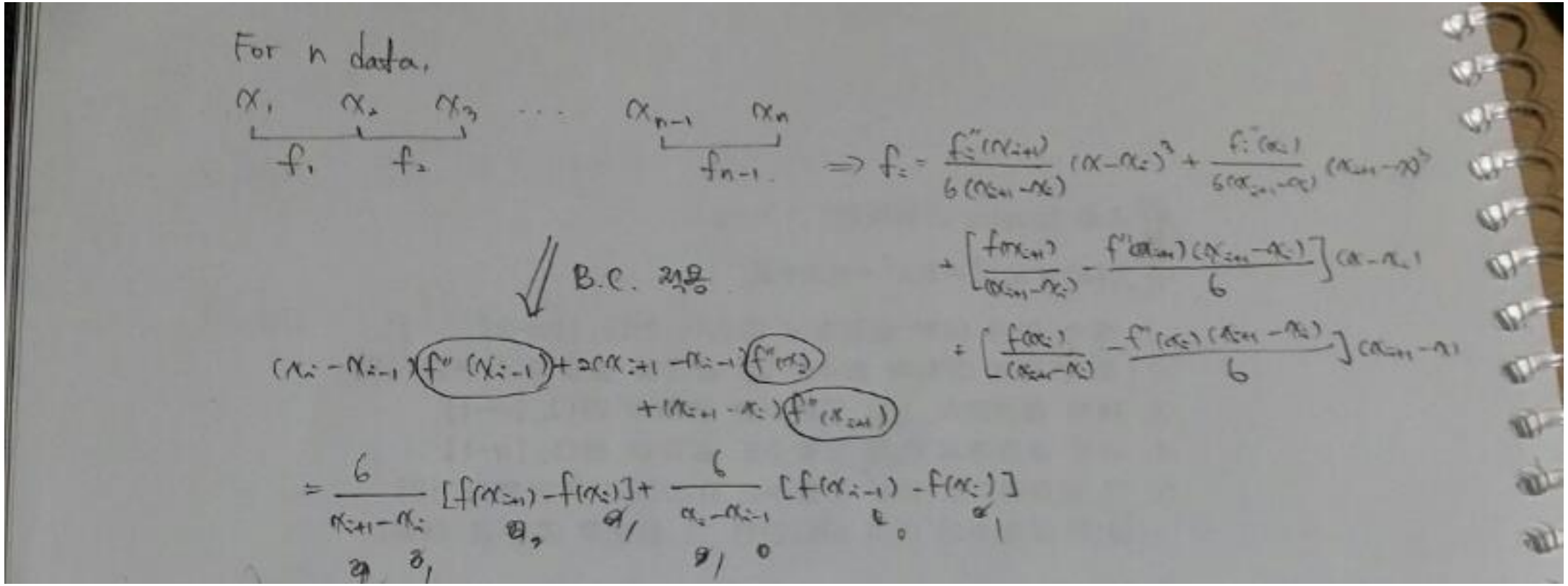
(2 차 도함수가 0 이 아니라면 그 정보로 조건을 대체)

Ax=b 꼴의 선형연립방정식 -> 직사각행렬이므로 pseudo-inverse가 필요
(n by n-2) * (n by 1) = (n-2 by 1)

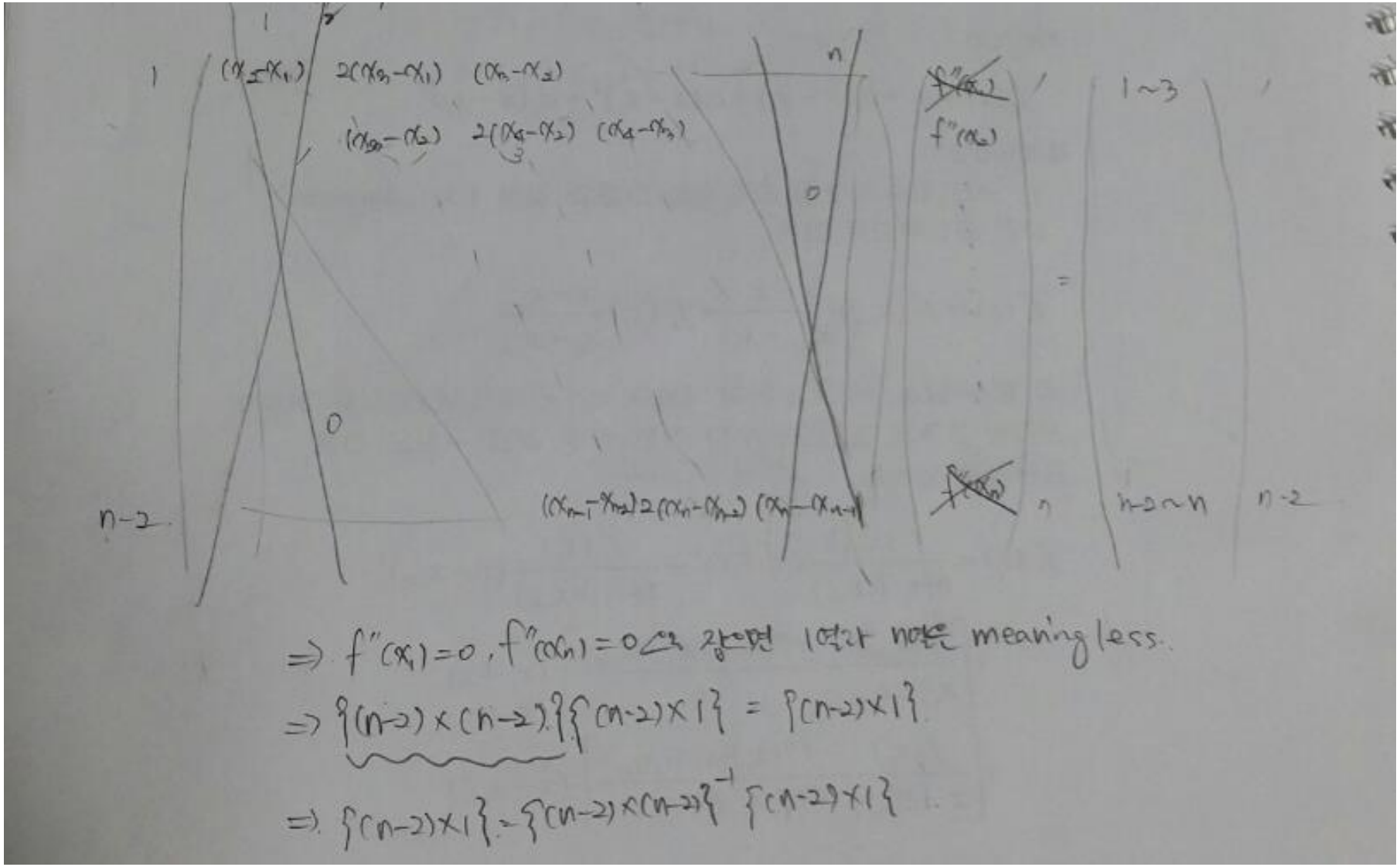
적절한 처리를 통한 Ax=b 꼴의 함수 생성
(n-2 by n-2) * (n by 1) = (n-2 by 1)

x 값은 이차 미분 값 -> 주어진 식으로 주어진 구간에서 값을 구할 수 있음

1. Idea to construct Interpolation - Cubic spline



1. Idea to construct Interpolation - Cubic spline



1. Idea to construct Interpolation - Lagrange

nth Lagrange Interpolating Polynomial

$$P_n(x) = f(x_0)L_{n,0}(x) + \cdots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x),$$

where

$$L_{n,k}(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{k-1})(x - x_{k+1})\cdots(x - x_n)}{(x_k - x_0)(x_k - x_1)\cdots(x_k - x_{k-1})(x_k - x_{k+1})\cdots(x_k - x_n)}$$

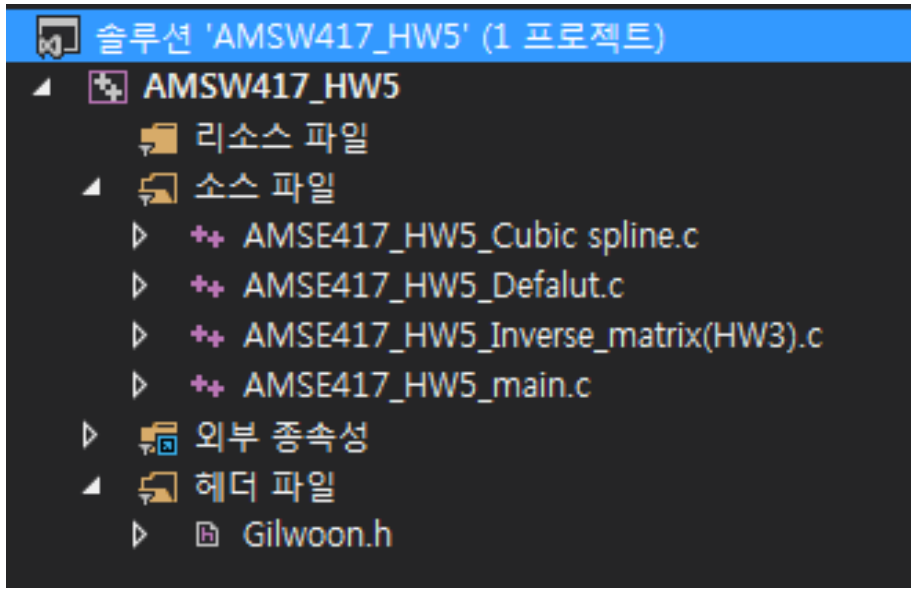
for each $k = 0, 1, \ldots, n$.

```
for (i = 0; i < row_max; i++)
{p = 1;
q = 1;
for (j = 0; j < row_max; j++)
{if (i != j)
{p *= (x - Raw_data[j][0]);}
if (i != j)
{q *= (Raw_data[i][0] - Raw_data[j][0]);}}
L_n[i] = p / q;}
```

2. Programmed code - Cubic spline

* Description of Program

1. Info. of Program



2. Characteristics of Program

- 시간 출력 가능
- 헤더 및 각 기능 함수화
- 크기제한 없음(malloc 이용)
- Sorting을 추가하려 했으나... 실패ㅠ

3. Process of main()

```c
#define _CRT_SECURE_NO_WARNINGS //Secure warning을 skip하는 명령문

#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include"Gilwoon.h"

int main()
{
clock_t time_start = 0;//시간출력 관련 변수들
clock_t time_end = 0;
double exe_time = 0;

Program_explanation();//Program explanation
Program_info();//Program 한 줄 설명

//시간 측정 시작
time_start = clock();

//Phase diagram의 정보 출력
Cubic_spline();

//걸린 시간 출력
time_end = clock();
exe_time = (double)(time_end - time_start) / CLOCKS_PER_SEC;
printf("\nexecusion time: %.4f sec\n", exe_time);
printf("============================\n");

return 0;
}
```

2. Programmed code

4. Header

```
#ifndef __GILWOON_H__//헤더가 정의 되지 않았을 때만 아래를 실행
#define __GILWOON_H__//헤더 정의

//defalut information 출력 관련 함수 모음
void Program_explanation();
void Program_info();

//Cubic spline 관련 함수 모음
void Cubic_spline();

//Inverse matrix 관련 함수 모음
void Inverse_given_matrix(double **Matrix_given, int row_max, int col_max);
void Normalization(double **Matrix_given, double **Inverse_matrix, int row_max, int col_max);
void Pivoting(double **Matrix_given, double **Inverse_matrix, int row_max, int col_max);
void Lower(double **Matrix_given, double **Inverse_matrix, int row_max, int col_max);
void Upper(double **Matrix_given, double **Inverse_matrix, int row_max, int col_max);

#endif
```

Cubic spline 관련

HW#3 약간 수정

앞으로 사용할 함수를 모아놓았다.

2. Programmed code

5. AMSE417_HW5_Cubic spline.c

```c
#define _CRT_SECURE_NO_WARNINGS //Secure warning을 skip하는 명령문

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include"Gilwoon.h"

void Cubic_spline()
{double **Matrix_cal = NULL;
double *Matrix_vec = NULL;
double *Diff_2 = NULL;
double **Raw_data = NULL;
double row_max, col_max;
double f_i, x;
int i, j, k;
double two = 2;

FILE *file;
file = fopen("Info.txt", "r");
if (file == 0)
{printf("No file.");
return -1;}

//data 크기
fscanf(file, "%lf", &row_max);
col_max = row_max;

//Matrix의 2-D 공간 공간 생성
i = 0;
Matrix_cal = (double**)malloc(sizeof(double*)*(row_max - 2));
while (i<row_max-2)
{Matrix_cal[i++] = (double*)malloc(sizeof(double)*(col_max - 2));}
i = 0;
Raw_data = (double**)malloc(sizeof(double*)*(row_max));
while (i<row_max)
{Raw_data[i++] = (double*)malloc(sizeof(double) * (two));}
Matrix_vec = (double*)malloc(sizeof(double)*(row_max - 2));
Diff_2 = (double*)malloc(sizeof(double)*(row_max));
```

```c
//raw data
for (i = 0; i < row_max; i++)
{fscanf(file, "%lf %lf", &Raw_data[i][0], &Raw_data[i][1]);}

fclose(file);

//Matrix_cal 생성
for (i = 0; i < row_max- 2; i++)
{if (i == 0)
{j = i - 1;
Matrix_cal[i][j + 1] = 2 * (Raw_data[i + 2][0] - Raw_data[i][0]);
Matrix_cal[i][j + 2] = Raw_data[i + 2][0] - Raw_data[i + 1][0];
for (k = j + 3; k < col_max - 2; k++)
{Matrix_cal[i][k] = 0;}}
if (i == col_max-3)
{j = i - 1;
Matrix_cal[i][j] = Raw_data[i + 1][0] - Raw_data[i][0];
Matrix_cal[i][j + 1] = 2 * (Raw_data[i + 2][0] - Raw_data[i][0]);
for (k = 0; k < j; k++)
{Matrix_cal[i][k] = 0;}}
else
{j = i - 1;
Matrix_cal[i][j] = Raw_data[i+1][0] - Raw_data[i][0];
Matrix_cal[i][j+1] = 2 * (Raw_data[i+2][0] - Raw_data[i][0]);
Matrix_cal[i][j+2] = Raw_data[i+2][0] - Raw_data[i+1][0];
for (k = 0; k < j; k++)
{Matrix_cal[i][k] = 0;}
for (k = j + 3; k < col_max - 2; k++)
{Matrix_cal[i][k] = 0;}}}
```

2. Programmed code

5. AMSE417_HW5_Cubic spline.c

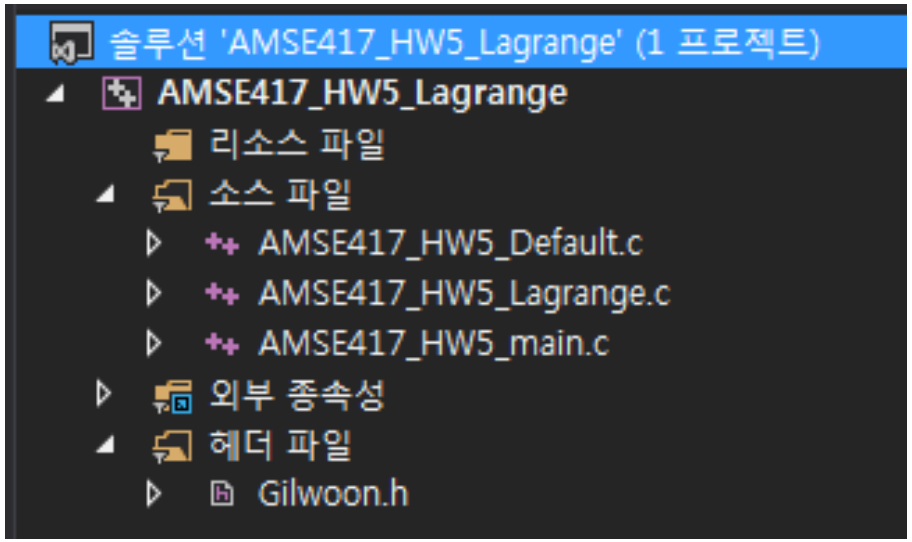
```
for (i = 0; i < col_max - 2; i++)
{Matrix_vec[i] = (6 * (Raw_data[i + 2][1] - Raw_data[i + 1][1]) /
(Raw_data[i + 2][0] - Raw_data[i + 1][0]))
+ (6 * (Raw_data[i][1] - Raw_data[i + 1][1]) / (Raw_data[i + 1][0] -
Raw_data[i][0]));}

Inverse_given_matrix(Matrix_cal, row_max-2, col_max-2);

//2차 미분 값
Diff_2[0] = 0;
Diff_2[(int)(row_max) - 1] = 0;
for (i = 0; i < row_max - 2; i++)
{Diff_2[i + 1] = 0;}
for (i = 0; i < row_max - 2; i++)
{for (j = 0; j < col_max - 2; j++)
{Diff_2[i + 1] += Matrix_cal[i][j] * Matrix_vec[j];}}

//data 판별
printf("To get f(x), enter x: ");
scanf("%lf", &x);

i = 0;
while (i<row_max)
{if (x<Raw_data[i + 1][0] && x>Raw_data[i][0])
{f_i = Diff_2[i + 1] * pow((x - Raw_data[i][0]), 3) / (6 * (Raw_data[i +
1][0] - Raw_data[i][0]))
+ Diff_2[i] * pow((Raw_data[i + 1][0] - x), 3) / (6 * (Raw_data[i + 1][0] -
Raw_data[i][0]))
+ (Raw_data[i + 1][1] / (Raw_data[i + 1][0] - Raw_data[i][0]) - (Diff_2[i +
1] * (Raw_data[i + 1][0] - Raw_data[i][0]) / 6)) * (x - Raw_data[i][0])
+ (Raw_data[i][1] / (Raw_data[i + 1][0] - Raw_data[i][0]) - (Diff_2[i] *
(Raw_data[i + 1][0] - Raw_data[i][0]) / 6)) * (Raw_data[i + 1][0] - x);
printf("Range is %f < x < %f\n", Raw_data[i][0], Raw_data[i + 1][0]);
printf("The answer is: %f", f_i);
break;}
i++;}
if (i == row_max)
{printf("range out");}
```

```
//Matrix의 2-D 공간 free
i = 0;
/*
while (i < row_max-2)
{
free(Matrix_cal[i++]);
}*/
free(Matrix_cal);
i = 0;
while (i < row_max)
{
free(Raw_data[i++]);
}
free(Raw_data);
free(Matrix_vec);
free(Diff_2);
}
```

2. Programmed code - Lagrange

* Description of Program

1. Info. of Program



2. Characteristics of Program

- 시간 출력 가능
- 헤더 및 각 기능 함수화
- 크기제한 없음(malloc 이용)
- 마찬가지, Sorting을 추가하려 했으나... 실패ㅠ

3. Process of main()

```c
#define _CRT_SECURE_NO_WARNINGS //Secure warning을 skip하는 명령문

#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include"Gilwoon.h"

int main()
{
clock_t time_start = 0;//시간출력 관련 변수들
clock_t time_end = 0;
double exe_time = 0;
float MIN_given = 0;//범위 지정 변수
float MAX_given = 0;

Program_explanation();//Program explanation
Program_info();//Program 한 줄 설명

//시간 측정 시작
time_start = clock();

//Phase diagram의 정보 출력
Lagrange();

//걸린 시간 출력
time_end = clock();
exe_time = (double)(time_end - time_start) / CLOCKS_PER_SEC;
printf("\nexecusion time: %.4f sec\n", exe_time);
printf("============================\n");

return 0;
}
```

2. Programmed code

4. Header

```
#ifndef __GILWOON_H__//헤더가 정의 되지 않았을 때만 아래를 실행
#define __GILWOON_H__//헤더 정의

//defalut information 출력 관련 함수 모음
void Program_explanation();
void Program_info();

//Cubic spline 관련 함수 모음
void Lagrange();

#endif
```

Lagrange 관련

앞으로 사용할 함수를 모아놓았다.

2. Programmed code

5. AMSE417_HW5_Lagrange.c

```c
#define _CRT_SECURE_NO_WARNINGS //Secure warning을 skip하는 명령문

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include"Gilwoon.h"

void Lagrange()
{
double **Raw_data = NULL;
double *L_n = NULL;
double row_max, col_max;
double f = 0;
double x;
int i, j, k;
double two = 2;
double p = 1;
double q = 1;

FILE *file;
file = fopen("Info.txt", "r");
if (file == 0)
{printf("No file.");
return -1;}

//data 크기
fscanf(file, "%lf", &row_max);
col_max = row_max;

//Matrix의 2-D 공간 공간 생성
i = 0;
Raw_data = (double**)malloc(sizeof(double*)*(row_max));
while (i<row_max)
{
Raw_data[i++] = (double*)malloc(sizeof(double) * (two));
}
L_n = (double*)malloc(sizeof(double*)*(row_max));
```

```c
//raw data
for (i = 0; i < row_max; i++)
{
fscanf(file, "%lf %lf", &Raw_data[i][0], &Raw_data[i][1]);
}

fclose(file);


//data 판별
printf("To get f(x), enter x: ");
scanf("%lf", &x);

for (i = 0; i < row_max; i++)
{
L_n[i] = 0;
}
for (i = 0; i < row_max; i++)
{
p = 1;
q = 1;
for (j = 0; j < row_max; j++)
{
if (i != j)
{
p *= (x - Raw_data[j][0]);
}
if (i != j)
{
q *= (Raw_data[i][0] - Raw_data[j][0]);
}
}
L_n[i] = p / q;
}
```

2. Programmed code

5. AMSE417_HW5_Lagrange.c

```
i = 0;
while (i<row_max)
{
if (x<Raw_data[i + 1][0] && x>Raw_data[i][0])
{
for (j = 0; j < row_max; j++)
{
f += Raw_data[j][1]*L_n[j];
}
printf("Range is %f < x < %f\n", Raw_data[i][0], Raw_data[i + 1][0]);
printf("The answer is: %f", f);
break;
}
i++;
}
if (i == row_max)
{
printf("range out");
}


//Matrix의 2-D 공간 free
i = 0;
while (i < row_max)
{
free(Raw_data[i++]);
}
free(Raw_data);
//free(L_n);
}
```

3. Result & Conclusion

1. 실행 결과(임의값)

```
4
3.0 2.5
4.5 1.0
7.0 2.5
9.0 0.5
```

Cubic Spline

```
This programm will do interpolation of given data.
============================
To get f(x), enter x: 5
Range is 4.500000 < x < 7.000000
The answer is: 1.102890
```

교재: 1.102886 -> 0.000004 만큼의 오차 발생

Lagrange

```
This programm will do interpolation of given data.
============================
To get f(x), enter x: 5
Range is 4.500000 < x < 7.000000
The answer is: 1.151852
```

Cubic Spline과는 큰 오차 발생 -> x=7에 더 가까운 경향

교재 값은 spline으로 구한 값이므로,
어떤 값이 더 정확한 값인지는 알 수 없다.

3. Result & Conclusion

2. 실행 결과(ln x)

```
3
1 0
4 1.386294
6 1.791759
```

f(x)=ln(x)

Cubic Spline

```
This programm will do interpolation of given data.
============================
To get f(x), enter x: 2
Range is 1.000000 < x < 4.000000
The answer is: 0.531262
```

참값: ln2=0.693147 -> 0.161885 만큼의 오차 발생

Lagrange
better

```
This programm will do interpolation of given data.
============================
To get f(x), enter x: 2
Range is 1.000000 < x < 4.000000
The answer is: 0.565844
```

참값: ln2=0.693147 -> 0.127303 만큼의 오차 발생
마찬가지로 4의 값에 더 가깝게 나타나는 현상

3. Result & Conclusion

2. 실행 결과(ln x, 유효숫자 차이)

참값: ln2=0.693147

Lagrange

```
3
1 0
4 1.39
6 1.79
```

```
This programm will do interpolati
============================
To get f(x), enter x: 2
Range is 1.000000 < x < 4.000000
The answer is: 0.568667
```

유효숫자 소수 2자리

```
3
1 0
4 1.3863
6 1.7918
```

```
This programm will do interpolati
============================
To get f(x), enter x: 2
Range is 1.000000 < x < 4.000000
The answer is: 0.565840
```

유효숫자 소수 4자리

```
3
1 0
4 1.386294
6 1.791759
```

```
This programm will do interpolati
============================
To get f(x), enter x: 2
Range is 1.000000 < x < 4.000000
The answer is: 0.565844
```

유효숫자 소수 6자리

```
3
1 0
4 1.38629436
6 1.79175947
```

```
This programm will do interpolati
============================
To get f(x), enter x: 2
Range is 1.000000 < x < 4.000000
The answer is: 0.565844
```

유효숫자 소수 8자리

유효숫자가 어느 정도를 넘으면 오차의 차이가 발생하지 않음

3. Result & Conclusion

3. 실행 결과(Phase diagram data)

```
10
1258.961472 0.174338
1306.521354 0.321028
1354.081235 0.446346
1401.641117 0.554909
1449.200998 0.650163
1496.76088 0.73472
1544.320762 0.810583
1591.880643 0.879311
1639.440525 0.942126
```

참값 데이터
1373.9466621 0.49350961

Cubic Spline

```
This programm will do interpolation of given data.
============================
To get f(x), enter x: 1373.9466621
Range is 1354.081235 < x < 1401.641117
The answer is: 0.493467
```

참값에 비해 0.000043 만큼의 오차 발생

Lagrange
better

```
This programm will do interpolation of given data.
============================
To get f(x), enter x: 1373.9466621
Range is 1354.081235 < x < 1401.641117
The answer is: 0.493509
```

참값에 비해 0.000001 만큼의 오차 발생

3. Result & Conclusion

3. 실행 결과(Phase diagram data)

```
20
1211.40159      0
1235.181531     0.091101
1258.961472     0.174338
1282.741413     0.250701
1306.521354     0.321028
1330.301294     0.386038
1354.081235     0.446346
1377.861176     0.502484
1401.641117     0.554909
1425.421058     0.60402
1449.200998     0.650163
1472.980939     0.693641
1496.76088      0.73472
1520.540821     0.773632
1544.320762     0.810583
1568.100702     0.845756
1591.880643     0.879311
1615.660584     0.911392
1639.440525     0.942126
1663.220466     0.971628
```

참값 데이터
1373.9466621 0.49350961

Cubic Spline

```
This programm will do interpolation of given data.
==========================
To get f(x), enter x: 1373.9466621
Range is 1354.081235 < x < 1401.641117
The answer is: 0.493467
```

```
This programm will do interpolation of given data.
============================
To get f(x), enter x: 1373.9466621
Range is 1354.081235 < x < 1377.861176
The answer is: 0.493515
```

참값에 비해 0.000043 -> 0.000006 만큼의 오차 발생

Lagrange
better

```
This programm will do interpolation of given data.
==========================
To get f(x), enter x: 1373.9466621
Range is 1354.081235 < x < 1401.641117
The answer is: 0.493509
```

```
This programm will do interpolation of given data.
============================
To get f(x), enter x: 1373.9466621
Range is 1354.081235 < x < 1377.861176
The answer is: 0.493510
```

참값에 비해 0.000001 -> 0.000001 만큼의 오차 발생

전체 Data 개수가 조밀하게 증가할 경우 정확도가 증가하는 것을 확인 가능!

3. Result & Conclusion

4. 실행 결과(1/x)

```
10
1       1
2       0.5
3       0.333333333
4       0.25
5       0.2
6       0.166666667
7       0.142857143
8       0.125
9       0.111111111
10      0.1
```

$f(x)=1/x$

참값 데이터
5.7 0.175439

Cubic Spline

```
================================
To get f(x), enter x: 5.7
Range is 5.000000 < x < 6.000000
The answer is: 0.175546
```

참값에 비해 0.000107 만큼의 오차 발생

Lagrange
better

```
================================
To get f(x), enter x: 5.7
Range is 5.000000 < x < 6.000000
The answer is: 0.175473
```

참값에 비해 0.000034 만큼의 오차 발생

3. Result & Conclusion

4. 실행 결과(1/x)

```
6
1          1
4          0.25
5.5        0.181818182
6          0.166666667
7.5        0.133333333
9          0.111111111
```

$f(x)=1/x$

참값 데이터
5.7 0.175439

Cubic Spline

```
============================
To get f(x), enter x: 5.7
Range is 5.000000 < x < 6.000000
The answer is: 0.175546
```

```
============================
To get f(x), enter x: 5.7
Range is 5.500000 < x < 6.000000
The answer is: 0.176057
```

참값에 비해 0.000107->0.000618 만큼의 오차 발생

Lagrange
better

```
============================
To get f(x), enter x: 5.7
Range is 5.000000 < x < 6.000000
The answer is: 0.175473
```

```
============================
To get f(x), enter x: 5.7
Range is 5.500000 < x < 6.000000
The answer is: 0.175495
```

참값에 비해 0.000034->0.000056 만큼의 오차 발생

Data 개수가 원하는 구간에서 조밀하게 증가할 경우, curve에 의해 정확도 감소

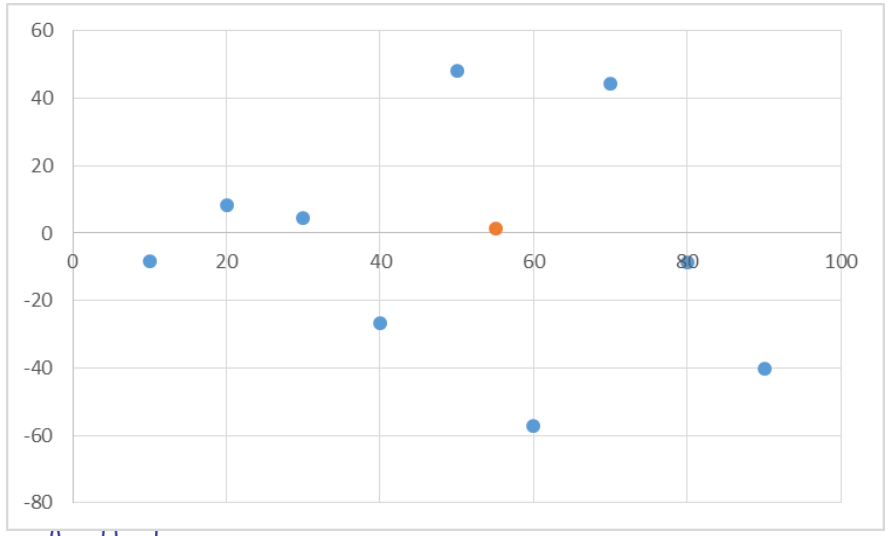
3. Result & Conclusion

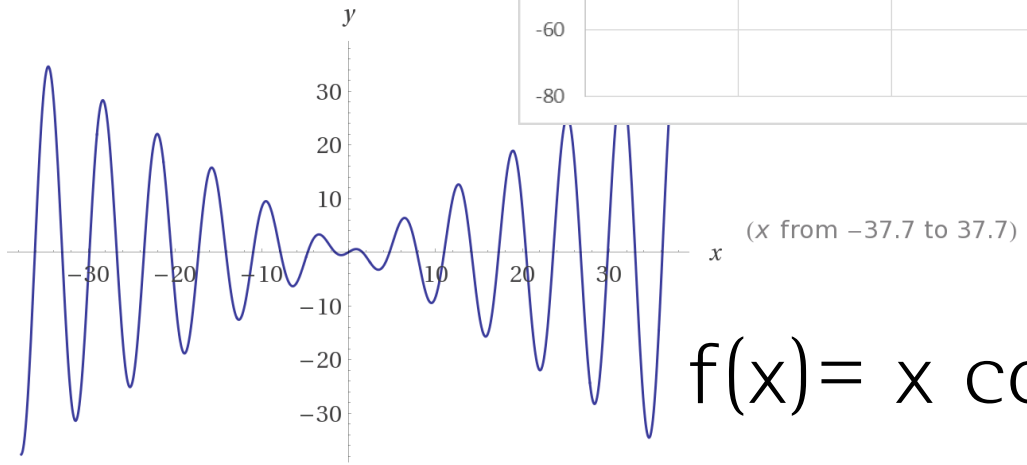
5. 실행 결과(xcosx)

```
9
10        -8.390715291
20        8.161641236
30        4.627543497
40        -26.67752247
50        48.24830142
60        -57.14477882
70        44.33234422
80        -8.830979507
90        -40.32662545
```

참값 데이터

55 1.216971594





$f(x) = x \cos(x)$

Cubic Spline

```
================================
To get f(x), enter x: 55
Range is 50.000000 < x < 60.000000
The answer is: -7.362098
```

참값에 비해 8.579070 만큼의 오차 발생

Lagrange

better

```
================================
To get f(x), enter x: 55
Range is 50.000000 < x < 60.000000
The answer is: 2.583706
```

참값에 비해 1.366734 만큼의 오차 발생

복잡한 함수일 수록 보간법을 이용한 계산의 정확도가 감소

3. Conclusion

1) 임의의 값에 대한 분석을 시행하였을 때, spline의 경우 교재의 값을 나타내었으며, 다만 Lagrange의 경우에는 값이 서로 많이 차이 나는 것을 확인하였다. (뒤에서 체크한 데이터의 경우 많은 차이를 보이지는 않았다.)
   이 경우에는 다른 데이터에 비해 함수의 값이 복잡하게 분포하는 경우였으며, 정확한 값을 알 수 없어 이에 대한 결론은 내릴 수 없었다.
2) ln2에 대한 실험을 하였을 때, spline의 경우 오차가 0.161885, Lagrange의 경우 오차가 0.127303로 더 작게 나오는 것을 확인하였다.
   유효숫자의 차이에 의한 결과도 분석하였으며, data의 유효숫자에 따라 값이 영향을 받으나 6자리 정도가 넘어가면서 부터는 값의 차이가 많이 생기지 않음을 확인하였다.
3) Phase diagram에 대한 실험을 하였을 때, data의 개수를 범위를 일정하게 두고 10개에서 20개로 늘렸을 때, spline의 경우 오차가 0.000043 -> 0.000006, Lagrange의 경우 0.000001 -> 0.000001로 오차의 값이 줄어드는 것을 확인하였다. 마찬가지로 여기서도 Lagrange가 더 좋은 결과를 나타내었다.
4) 1/x에 대한 실험을 하였을 때, 원하는 구간 주위를 좁게 할 경우 근사 하는 function의 curve에 의해 오히려 값이 더 나빠지는 것을 확인하였다. spline의 경우 오차가 0.000107->0.000618, Lagrange의 경우 0.000034->0.000056로 오차 값이 증가하였다.
5) 1)의 결과를 보강하기 위해 진동하며 발산하는 함수인 x cos(x)를 생각하였다. 이 중 급격하게 떨어지는 구간의 점을 잡았을 때, 값이 두 방법 모두 정확하게 예측하지 못하였으나, 여전이 Lagrange의 결과가 더 좋았다.


Thank
you!