Numerical Analysis For Materials

Homework #4

Gilwoon Lee

ID: 20120083
Dept: Material Science and Engineering

Homework #3

Determining phase diagram of Ge-Si

1. Idea to determine phase-diagram of Ge-Si

$T_{m,Ge} < T < T_{m,Si}$ 범위

B/C: $T_{m,Ge} == T$, $T_{m,Si} - 0.00001 < T$

반복

주어진 범위를 50등분

Newton's mtd

$X=\{x_s, x_l\}$의 초기값 설정

반복

Jacobian matrix (J(X))

$J^{-1}(X)$ } HW#3

$X_{new} = X - J^{-1}(X)*F(X)$

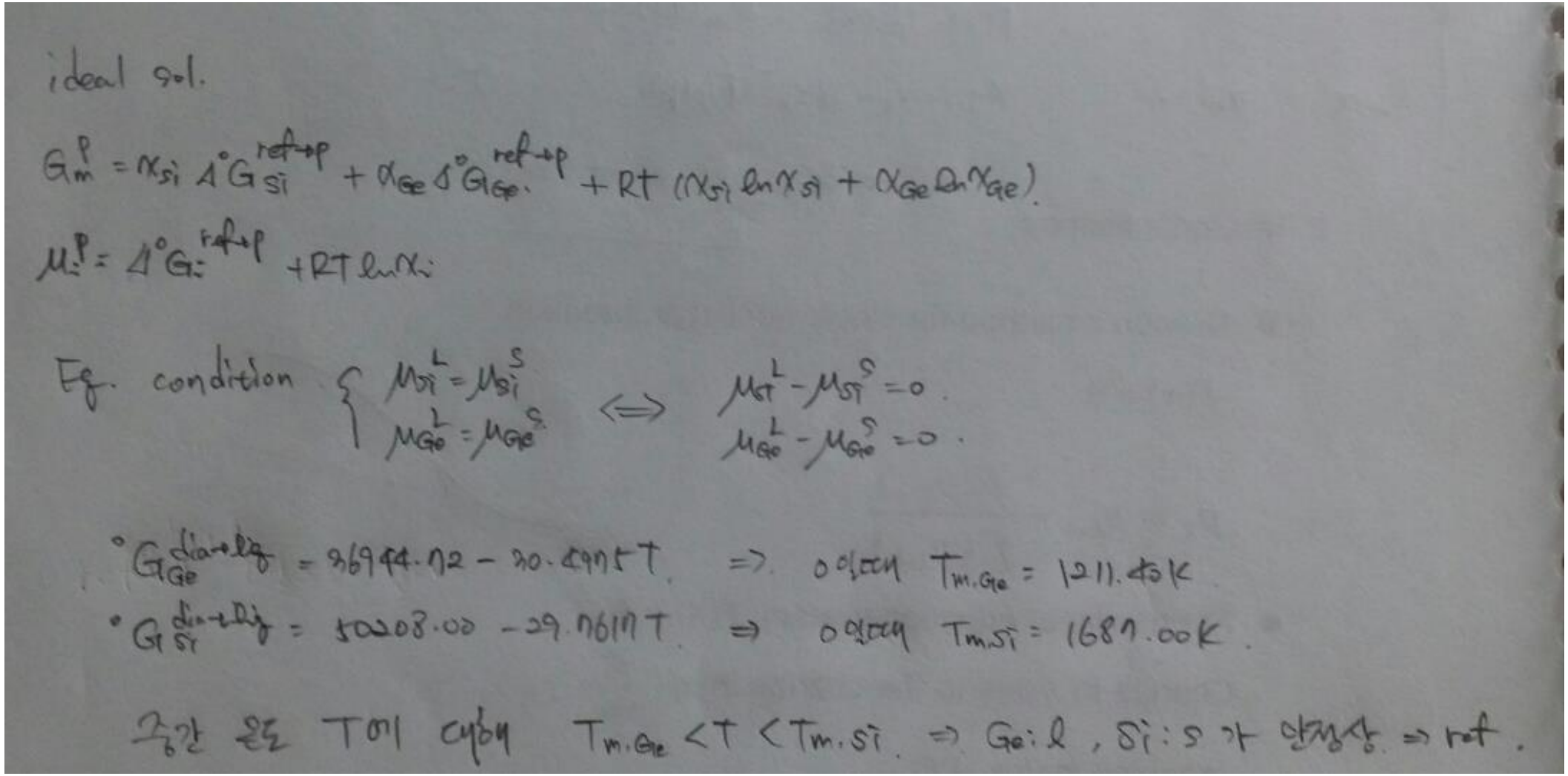
B/C: MIN보다 작으면 x = MIN + epsilon
MAX보다 커지면 x = MAX - epsilon으로 조정

최종 $X=\{x_s, x_l\}$값 출력

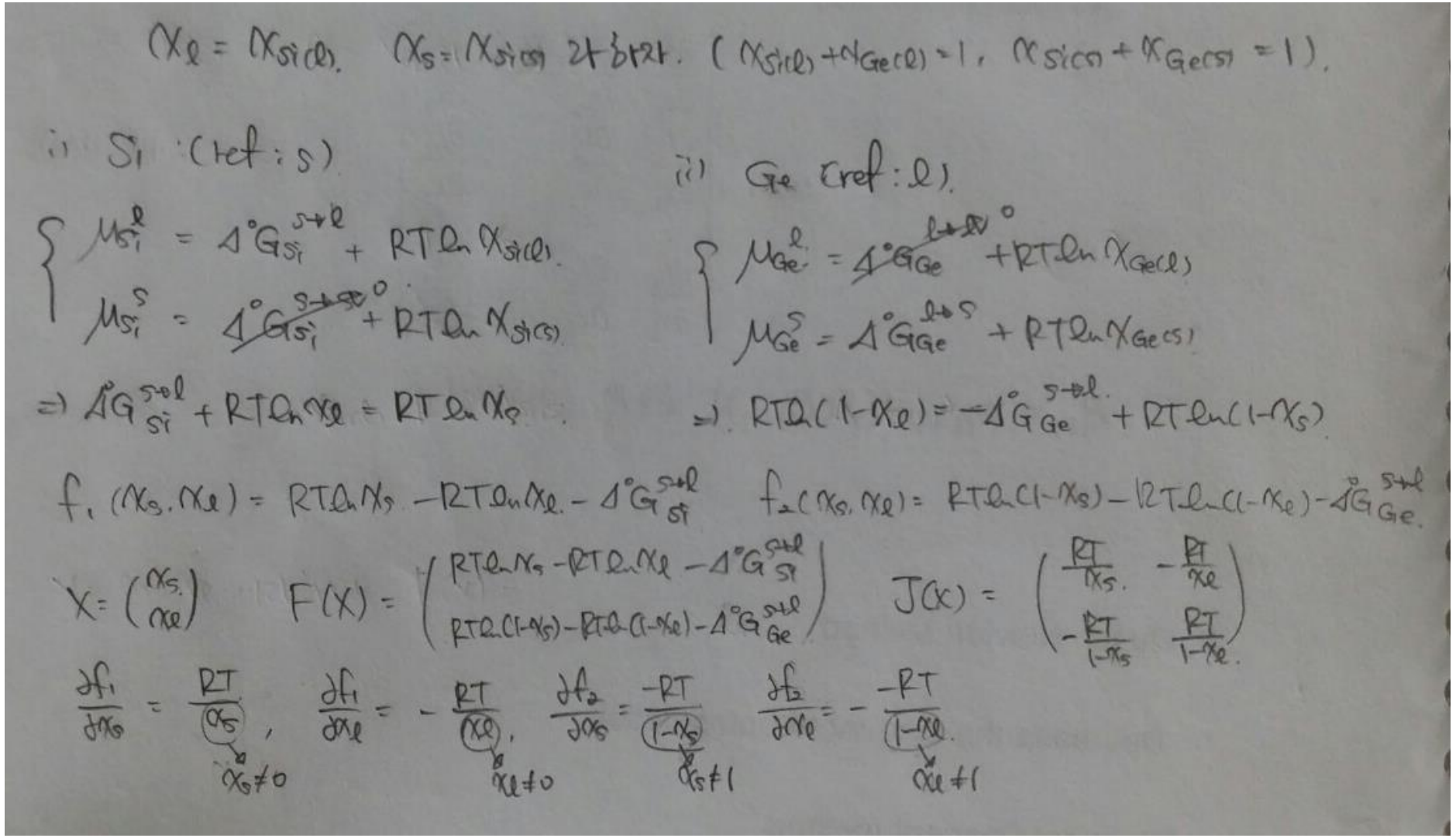
B/C: fabs(x_s_new - x_s)<epsilon &&
fabs(x_l_new - x_l)<epsilon

HW#2

1. Idea to determine phase-diagram of Ge-Si



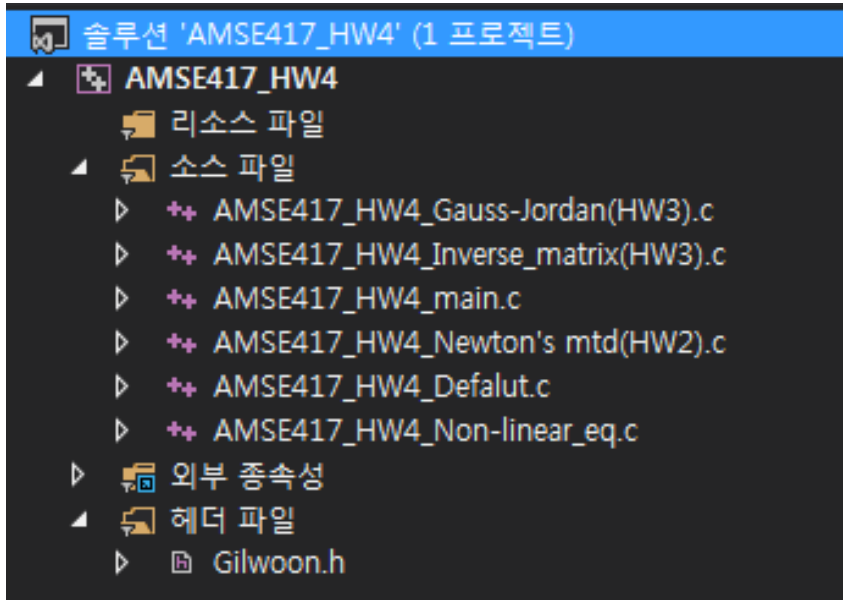
1. Idea to determine phase-diagram of Ge-Si



2. Programmed code

* Description of Program

1. Info. of Program



2. Characteristics of Program

- 시간 출력 가능
- 헤더 및 각 기능 함수화
- 크기제한 없음(malloc 이용)
- 파일로 정보 출력 (fprintf 이용)

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;

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

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

//시간 측정 시작
time_start = clock();
//Phase diagram의 정보 출력
Phase_diagram_data(file);
//걸린 시간 출력
time_end = clock();
printf("Data is saved in ""Info.txt.""\nCheck the file.\n");
exe_time = (double)(time_end - time_start) / CLOCKS_PER_SEC;
fprintf(file, "\nexecusion time: %.4f sec\n", exe_time);
fprintf(file, "============================\n");

fclose(file);
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(FILE *file);
void Program_info(FILE *file);

//Phase diagram information 관련 함수 모음
void Phase_diagram_data(FILE *file);
double f_1(double T, double x_s, double x_l);
double f_2(double T, double x_s, double x_l);
double G_Si(double T);
double G_Ge(double T);

//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);

//Newton's mtd 관련 함수 모음
void Newton_mtd(double MIN, double MAX, double x_s, double x_l, int turn, double T, FILE *file);

#endif
```

Phase diagram 정보 출력 & Material genome

HW#3 약간 수정

HW#2 대폭 수정 (2차원으로 확장)

앞으로 사용할 함수를 모아놓았다.

2. Programmed code

5. AMSE417_HW4_Non-linear_eq.c

```c
#define _CRT_SECURE_NO_WARNINGS //Secure warning을 skip하는 명령문
#define epsilon 0.00001
#define R 8.314

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

void Phase_diagram_data(FILE *file)
{double T_min = 36944.72 / 30.4975;
double T_max = 50208.00 / 29.7617;
double T = 0;
double x_s = 0;
double x_l = 0;

fprintf(file, "Temp, Turn, x_s, x_l\n");
fprintf(file, "============================\n");

T = T_min;
while(T < T_max)
{if (T == T_min)
{fprintf(file, "%f ", T);
fprintf(file, "BC 0.000000 0.000000\n");}
else if (T_max - epsilon<T)
{fprintf(file, "%f K, ", T_max);
fprintf(file, "BC 1.000000 1.000000\n");}
else
{x_s = (T - T_min) / (T_max - T_min) + epsilon;
x_l = (T - T_min) / (T_max - T_min) - epsilon;
fprintf(file, "%f , ", T);
Newton_mtd(0, 1, x_s, x_l, 0, T, file);}
T += (T_max - T_min) / 50;}}
```

```c
double G_Si(double T)
{
double G_Si = 50208.00 - 29.7617*T;
return G_Si;
}

double G_Ge(double T)
{
double G_Ge = 36944.72 - 30.4975*T;
return G_Ge;
}

double f_1(double T, double x_s, double x_l)
{
double f_1 = R*T*log(x_s) - R*T*log(x_l) - G_Si(T);
return f_1;
}

double f_2(double T, double x_s, double x_l)
{
double f_2 = R*T*log(1 - x_s) - R*T*log(1 - x_l) - G_Ge(T);
return f_2;
}
```

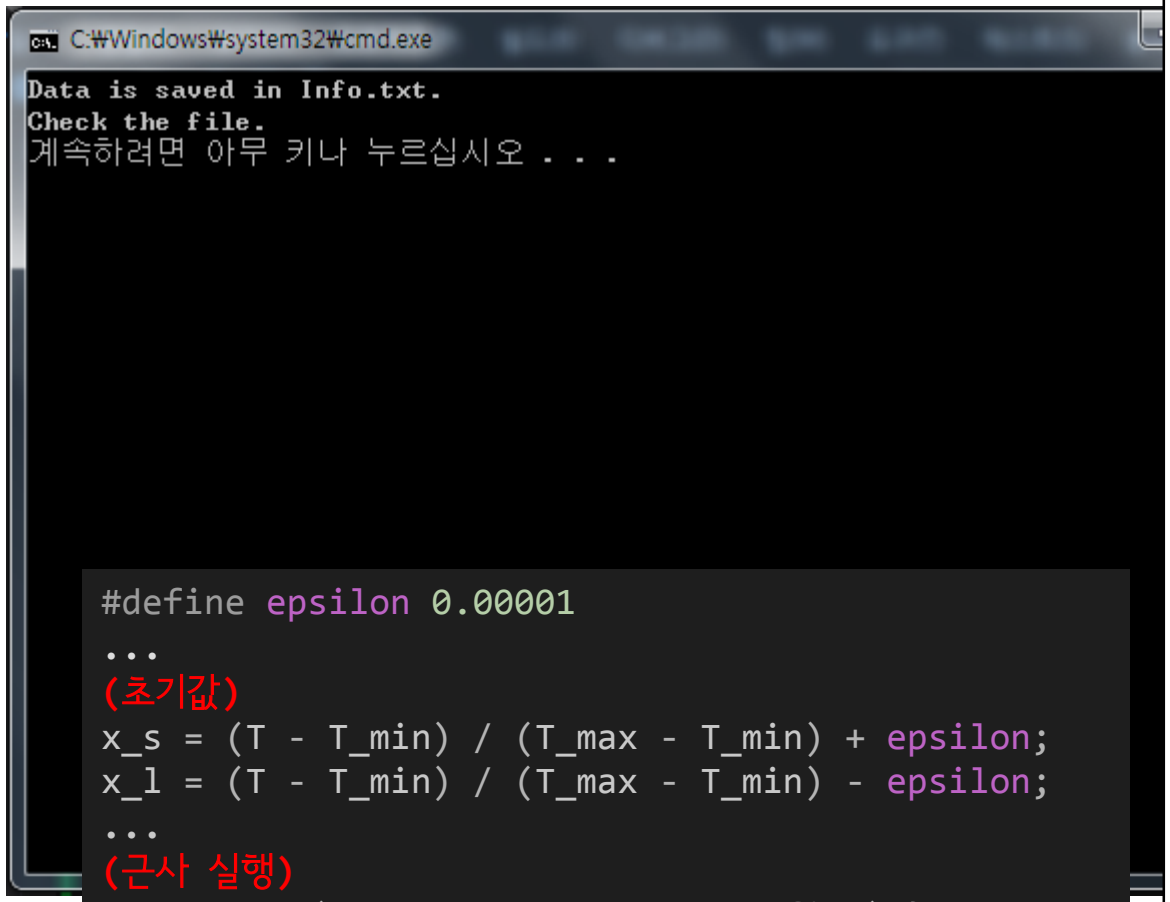
2. Programmed code

6. AMSE417_HW4_Newton's mtd(HW2).c

```c
#define _CRT_SECURE_NO_WARNINGS //Secure warning을 skip하는 명령문
#define epsilon 0.0000001
#define R 8.314

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

void Newton_mtd(double MIN, double MAX, double x_s, double x_l, int turn,
double T, FILE *file)//p: 최소, 최대 구간 정의, turn:loop를 돌린 횟수
{double x_s_new = 0;
double x_l_new = 0;
double **J = NULL;
double row_max = 2;
double col_max = 2;
int i;
int row;
int col;
double D;
//Matrix의 2-D 공간 공간 생성
i = 0;
J = (double**)malloc(sizeof(double*)*row_max);
while (i<row_max)
{J[i++] = (double*)malloc(sizeof(double)*col_max);}
//Boundary condition
if (MIN >= x_s)
{x_s = MIN + epsilon;}
else if (MIN >= x_l)
{x_l = MIN + epsilon;}
else if (MAX <= x_s)
{x_s = MAX - epsilon;}
else if (MAX <= x_l)
{x_l = MAX - epsilon;}

//Jacobian
J[0][0] = R*T / x_s;
J[0][1] = -R*T / x_l;
J[1][0] = -R*T / (1 - x_s);
J[1][1] = R*T / (1 - x_l);

Inverse_given_matrix(J, row_max, col_max);

x_s_new = x_s - (J[0][0] * f_1(T, x_s, x_l) + J[0][1] * f_2(T, x_s, x_l));
x_l_new = x_l - (J[1][0] * f_1(T, x_s, x_l) + J[1][1] * f_2(T, x_s, x_l));

if ((fabs(x_s_new - x_s)<epsilon && fabs(x_l_new - x_l)<epsilon) ||
turn>499)//구간 차이가 0.0001 미만 또는 횟수 초과일 경우 출력
{fprintf(file, "%d %.6f %.6f\n", turn, x_s_new, x_l_new);}
else//아래부터는 recursion을 진행
{if (MIN >= x_s_new)
{Newton_mtd(MIN, MAX, MIN+epsilon, x_l_new, ++turn, T, file);}
else if (MIN >= x_l_new)
{Newton_mtd(MIN, MAX, x_s_new, MIN + epsilon, ++turn, T, file);}
else if (MAX <= x_s_new)
{Newton_mtd(MIN, MAX, MAX - epsilon, x_l_new, ++turn, T, file);}
else if (MAX <= x_l_new)
{Newton_mtd(MIN, MAX, x_s_new, MAX - epsilon, ++turn, T, file);}
else
{Newton_mtd(MIN, MAX, x_s_new, x_l_new, ++turn, T, file);}}

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

3. Result & Conclusion

1. 실행 결과

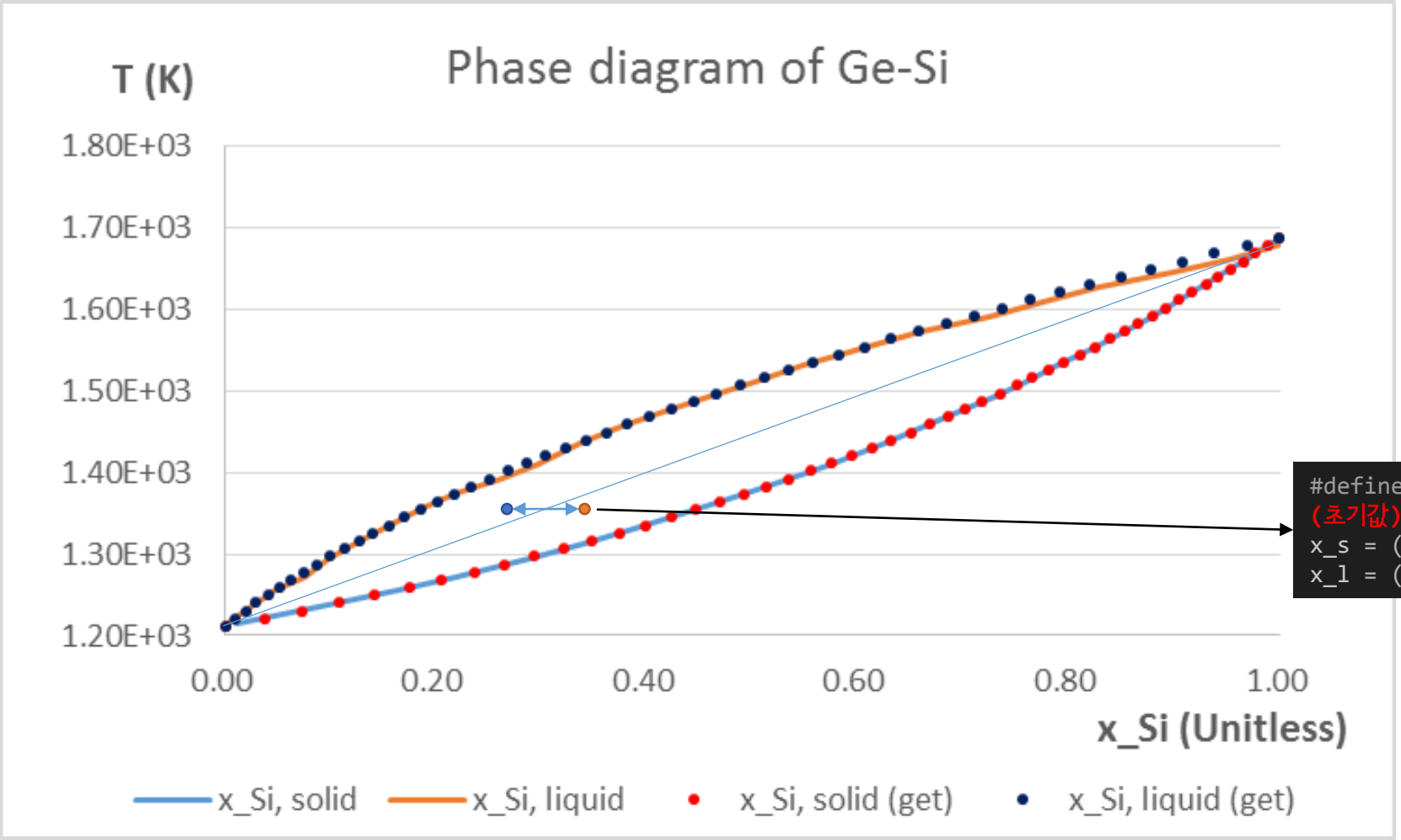



```
#define epsilon 0.00001
...
(초기값)
x_s = (T - T_min) / (T_max - T_min) + epsilon;
x_l = (T - T_min) / (T_max - T_min) - epsilon;
...
(근사 실행)
Newton_mtd(0, 1, x_s, x_l, 0, T, file);}
(구간)
T += (T_max - T_min) / 50;}}
```

```
================================================================================
================================================================================
Program: Determining phase diagram of Ge-Si
Date: 2015.03.30
Made by Gilwoon Lee
POSTECH, project 4 of [AMSE417] Numerical analysis for materials
Development environment: Visual Studio 2013
Code language: C
================================================================================

This programm will determine phase diagram of Ge-Si.
============================
Temp, Turn, x_s, x_l
============================
1211.401590 BC 0.000000 0.000000
1220.913567 23 0.037455 0.009550
1230.425543 25 0.073547 0.019484
1239.937519 25 0.108347 0.029805
1249.449496 26 0.141924 0.040516
1258.961472 27 0.174343 0.051622
1268.473448 27 0.205664 0.063126
1277.985425 27 0.235942 0.075032
1287.497401 28 0.265231 0.087343
1297.009377 28 0.293580 0.100062
1306.521354 28 0.321036 0.113192
1316.033330 29 0.347641 0.126737
1325.545306 24 0.373439 0.140699
1335.057283 24 0.398466 0.155082
1344.569259 24 0.422760 0.169887
1354.081235 24 0.446355 0.185118
1363.593212 23 0.469284 0.200778
1373.105188 23 0.491576 0.216867
1382.617164 23 0.513261 0.233390
1392.129140 23 0.534367 0.250348
1401.641117 22 0.554918 0.267744
1411.153093 22 0.574940 0.285578
1420.665069 22 0.594454 0.303854
1430.177046 21 0.613484 0.322573
1439.689022 21 0.632050 0.341737
1449.200998 21 0.650172 0.361348
1458.712975 22 0.667867 0.381407
1468.224951 23 0.685154 0.401916
1477.736927 23 0.702049 0.422875
1487.248904 23 0.718568 0.444288
1496.760880 24 0.734727 0.466154
1506.272856 24 0.750540 0.488474
1515.784833 24 0.766019 0.511251
1525.296809 24 0.781179 0.534485
1534.808785 25 0.796032 0.558177
1544.320762 25 0.810589 0.582327
1553.832738 25 0.824862 0.606937
1563.344714 25 0.838861 0.632007
1572.856691 25 0.852596 0.657539
1582.368667 26 0.866078 0.683531
1591.880643 26 0.879315 0.709986
1601.392620 26 0.892317 0.736903
1610.904596 26 0.905091 0.764282
1620.416572 26 0.917646 0.792125
1629.928549 26 0.929989 0.820431
1639.440525 26 0.942128 0.849200
1648.952501 26 0.954071 0.878433
1658.464478 27 0.965823 0.908130
1667.976454 30 0.977391 0.938290
1677.488430 28 0.988781 0.968913
1687.000407 BC 1.000000 1.000000

execusion time: 0.0150 sec
============================
```

3. Result & Conclusion

2. 결과 Plotting



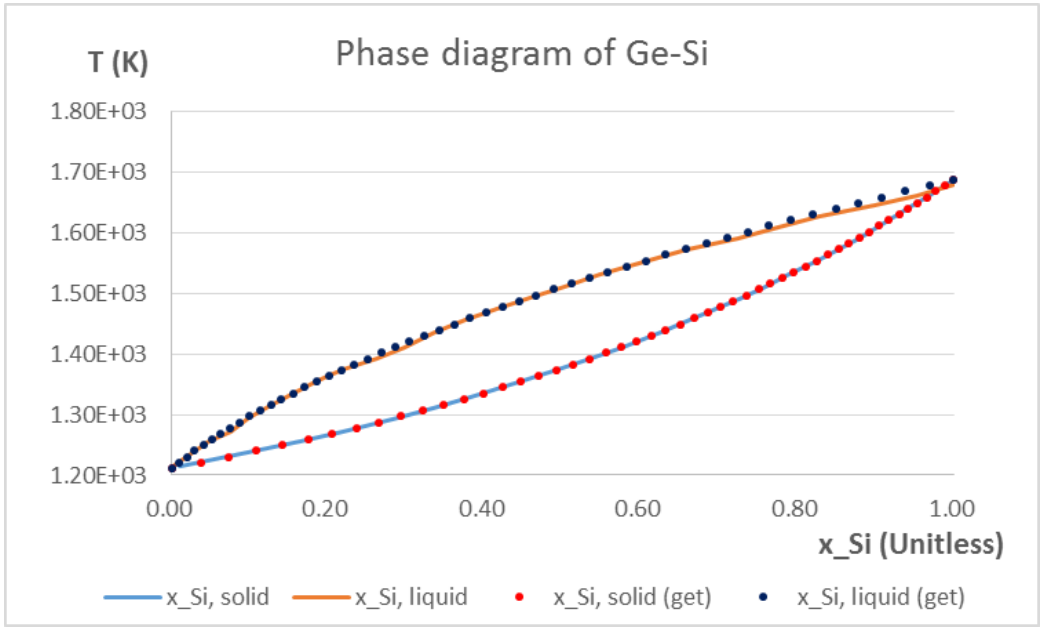
실선은 주어진 data,
각 점은 계산을 통해 얻은 data

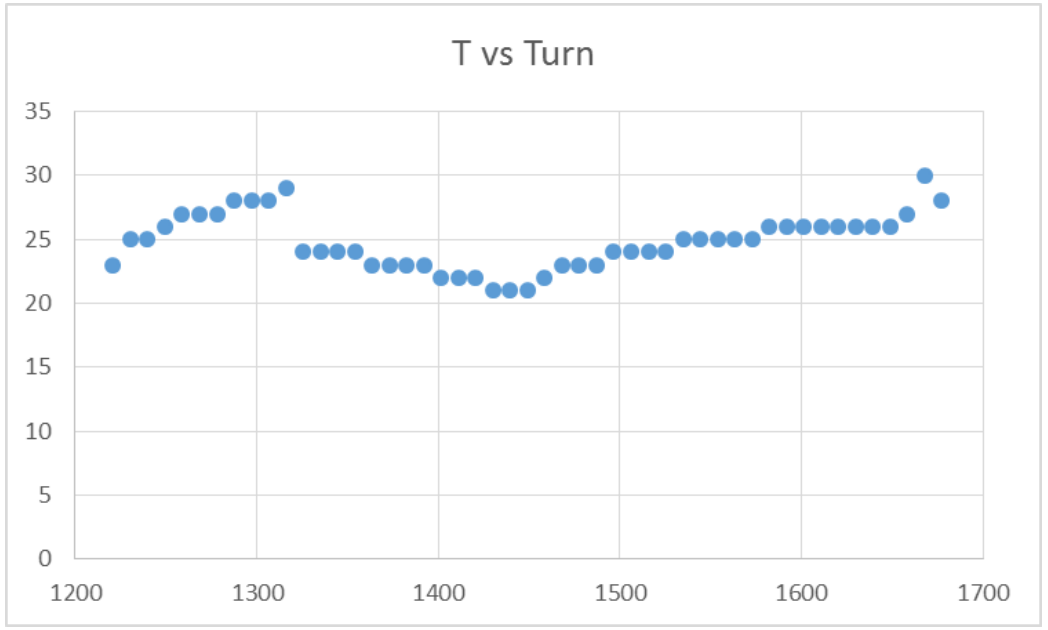
계산 값과 주어진 값이 거의 일치하나, liquidus line의 경우 mole fraction이 커질 수록 약간 경향에서 벗어남을 확인하였다.

```
#define epsilon 0.00001
(초기값)
x_s = (T - T_min) / (T_max - T_min) + epsilon;
x_l = (T - T_min) / (T_max - T_min) - epsilon;
```

3. Result & Conclusion

2. 결과 Plotting





온도에 따른 근사 횟수를 비교해보았으며, 중간의 경우에 가장 빠르게 근사하고, 초반을 제외하고는 양 끝에서는 근사의 횟수가 점점 늘어나는 경향이 보인다.

3. Result & Conclusion

3. 초기값 변화

```
#define epsilon 0.00001
(초기값)
x_s = (T - T_min) / (T_max - T_min) + epsilon;
x_l = (T - T_min) / (T_max - T_min) - epsilon;
```

초기 조건으로 평균 24.79592회, 표준편차 2.16005회에 근사 한다.

```
#define epsilon 0.00001
(초기값)
x_s = (T - T_min) / (T_max - T_min);
x_l = (T - T_min) / (T_max - T_min);
```

초기 값을 같은 값으로 줄 경우 D 값이 0이 되어 횟수가 초과되는 것을 확인할 수 있었다.

```
#define epsilon 0.00001
(초기값)
x_s = epsilon;
x_l = 1 - epsilon;
```

값은 잘 근사하나 평균 32.22449회, 표준편차 2.382797회에 근사 한다.

```
#define epsilon 0.00001
(초기값)
x_s = 0.3;
x_l = 0.7;
```

값은 잘 근사하나 평균 26.55102회, 표준편차 2.180035회에 근사 한다.

```
---------------------------
Temp, Turn, x_s, x_l
===========================
1211.401590 BC 0.000000 0.000000
1220.913567 500 -0.028578 -0.028578
1230.425543 500 -0.056715 -0.056715
1239.937519 500 -0.084420 -0.084420
1249.449496 500 -0.111703 -0.111703
1258.961472 500 -0.138574 -0.138574
1268.473448 500 -0.165042 -0.165042
1277.985425 500 -0.191116 -0.191116
1287.497401 500 -0.216804 -0.216804
1297.009377 500 -0.242116 -0.242116
1306.521354 500 -0.267060 -0.267060
1316.033330 500 -0.291642 -0.291642
1325.545306 500 -0.315872 -0.315872
1335.057283 500 -0.339757 -0.339757
1344.569259 500 -0.363303 -0.363303
1354.081235 500 -0.386519 -0.386519
1363.593212 500 -0.409411 -0.409411
```

3. Conclusion

* Problem answer

주어진 열역학적 data를 이용하여 phase diagram을 이론 값과 비슷하게 plotting 할 수 있었으며, 횟수도 평균 24.79592회, 표준편차 2.16005회 만에 근사하였다.

* 초기 값에 따른 횟수 비교

1) 반복 횟수에 걸리는 횟수가 비례한다고 생각하였을 때, 양 끝 점을 linear하게 잇는 선에서 +- epsilon 값을 가질 때 가장 잘 나올 것으로 예측하였고, 몇 가지 case를 check해 보았을 때 맞을 것으로 예측된다.
2) 값이 서로 같은 경우 determinant를 0으로 만들어 결과가 잘 나오지 않았다. 비슷하게 determinant가 0이 되는 다른 case를 넣을 경우 결과가 나오지 않을 것으로 생각된다.
3) 임의의 고정된 값을 넣었을 때도 잘 예측하나, 횟수는 조금 더 많아지는 것을 확인하였고, 각 온도에서 횟수 간의 표준편차는 2.x 정도의 값을 유지하는 것을 확인하였다.


Thank
you!