Numerical Analysis For Materials

Homework #7

Gilwoon Lee

ID: 20120083
Dept: Material Science and Engineering

Homework #7

Integration of given equation

1. Theory

Midpoint Rule

If $f \in C^2[a,b]$, then a number ξ in (a,b) exists with

$$\int_a^b f(x)\,dx = (b-a)f\left(\frac{a+b}{2}\right) + \frac{f''(\xi)}{24}(b-a)^3.$$

Trapezoidal Rule

If $f \in C^2[a,b]$, then a number ξ in (a,b) exists with

$$\int_a^b f(x)\,dx = (b-a)\frac{f(a)+f(b)}{2} - \frac{f''(\xi)}{12}(b-a)^3.$$

Composite Simpson's Rule

Suppose that $f \in C^4[a,b]$. Then for some μ in (a,b) we have

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{(n/2)-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(b)\right] - \frac{(b-a)h^4}{180} f^{(4)}(\mu).$$

Romberg Integration

in General $$R_{k,1} = \frac{1}{2}\left\{R_{k-1,1} + h_{k-1}\sum_{i=1}^{2^{k-2}} f(a+(2i-1)h_k\right\}$$

for $k = 2, 3, \cdots, n$.

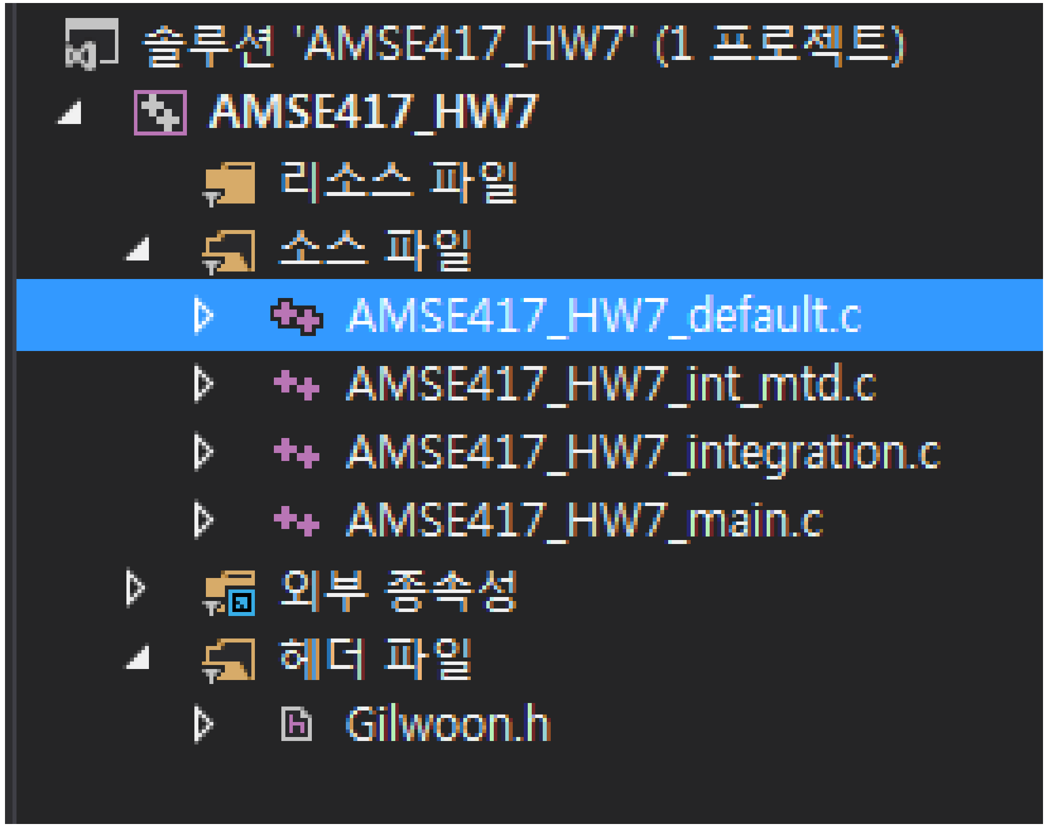
Extrapolation $$R_{k,2} = R_{k,1} + \frac{R_{k,1} - R_{k-1,1}}{3}$$

$$R_{k,j} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}$$

2. Programmed code

* Description of Program

1. Info. of Program



2. Characteristics of Program

- 헤더 및 각 기능 함수화
- 에러 범위를 조절 가능하게 함
- 각 구간 개수마다 파일로 에러를 출력하게 함

2. Programmed code

* Description of Program

3. Part of functions

```
void Midpoint(FILE *file)
{
double ans, err, para, section;
int i, num;
err = 1;
num = 1;

while (fabs(err) >= ERROR)
{
ans = 0;
section = (MAX - MIN) / num;
para = MIN;
for (i = 1; i <= num; i++)
{
ans += f((2 * para + section) / 2)*(section);
para += section;
}
err = ANSWER - ans;
fprintf(file, "%d %.8f %.9f\n", num, ans, err);

if (fabs(err) < ERROR)
{
printf("* Midpoint:%d,%.8f,%.9f\n", num, ans, err);
break;
}
num++;
}

}
```

```
void Trapezoidal(FILE *file)
{
double ans, err, para, section;
int i, num;
err = 1;
num = 1;

while (fabs(err) >= ERROR)
{
ans = 0;
section = (MAX - MIN) / num;
para = MIN;
for (i = 1; i <= num; i++)
{
ans += ((f(para) + f(para + section)) / 2)*(section);
para += section;
}
err = ANSWER - ans;
fprintf(file, "%d %.8f %.9f\n", num, ans, err);

if (fabs(err) < ERROR)
{
printf("* Trapezoidal:%d,%.8f,%.9f\n", num, ans, err);
break;
}
num++;
}
}
```

2. Programmed code

* Description of Program

3. Part of functions

```c
void Simpson(FILE *file)
{
double ans, err, para, section;
int i, num;
err = 1;
num = 2;

while (fabs(err) >= ERROR)
{ans = 0;
section = (MAX - MIN) / num;
para = MIN;
ans = f(para);
for (i = 1; i <= num - 2; i+=2)
{para += section;
ans += 4 * f(para);
para += section;
ans += 2 * f(para);}
para += section;
ans += 4 * f(para);
ans += f(MAX);
ans *= (section / 3);

err = ANSWER - ans;
fprintf(file, "%d %.8f %.9f\n", num/2, ans, err);

if (fabs(err) < ERROR)
{printf("* Simpson:%d,%.8f,%.9f\n", num/2, ans, err);
break;}
num += 2;
}
}
```

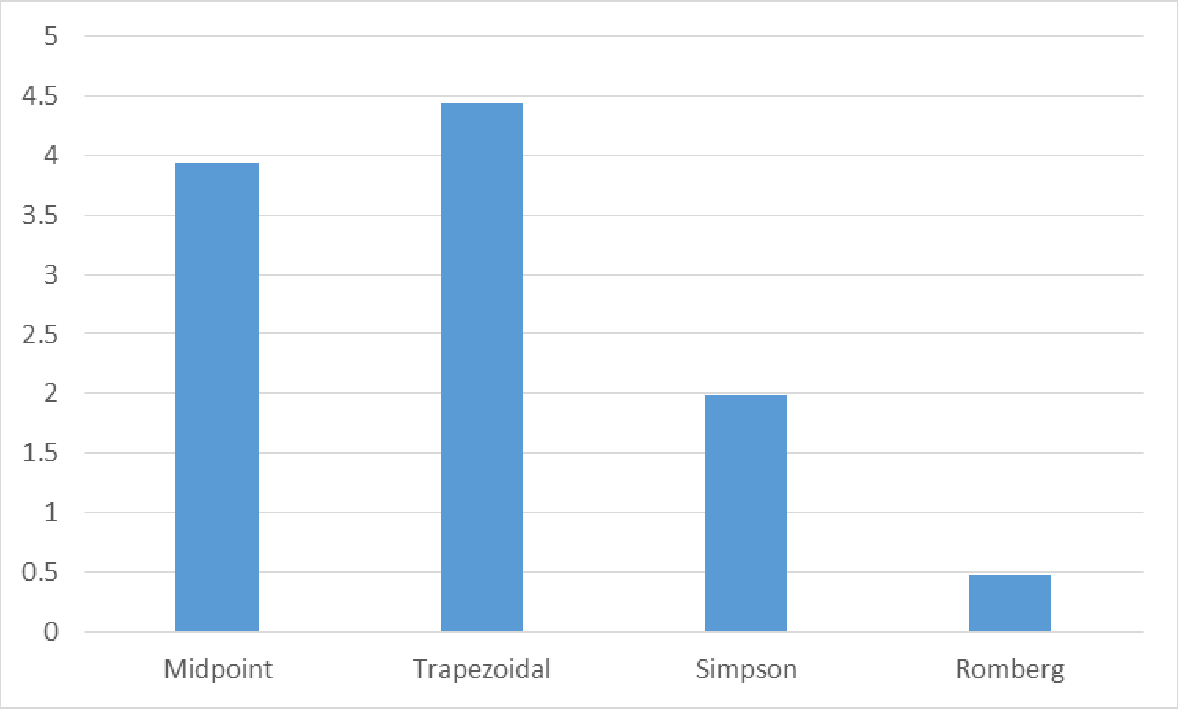
```c
void Romberg(FILE *file)
{double err, para;
double ans[ROW][COL];
double section, sum_temp;
int i, num;
err = ERROR + 1;
num = 1;

while (fabs(err) >= ERROR)
{if (num == 1)
{section = (MAX - MIN);
para = MIN;
ans[0][0] = section*(f(para) + f(para + section)) / 2;
err = ANSWER - ans[0][0];
fprintf(file, "%d %.8f %.9f\n", num, ans[0][0], err);
if (fabs(err) < ERROR) { break; }}
else
{sum_temp = 0;
section /= 2;
para = MIN;
for (i = 1; i <= pow(2, num-2); i++)
{sum_temp += f(para + (2 * i - 1)*section);}
ans[num-1][0] = (ans[num-2][0] + 2*section*sum_temp) / 2;
for (i = 2; i <= num; i++)
{ans[num - 1][i - 1] = ans[num - 1][i - 2] + (ans[num - 1][i - 2]
- ans[num - 2][i - 2]) / (pow(4, i - 1) - 1);}
err = ANSWER - ans[num - 1][num - 1];
fprintf(file, "%d %.8f %.9f\n", num, ans[num-1][num-1], err);
if (fabs(err) < ERROR) { break; }}
num++;}
printf("* Romberg:%d,%.8f,%.9f\n", num, ans[num-1][num-1], err);}
```

3. Result & Conclusion

1. 실행 결과

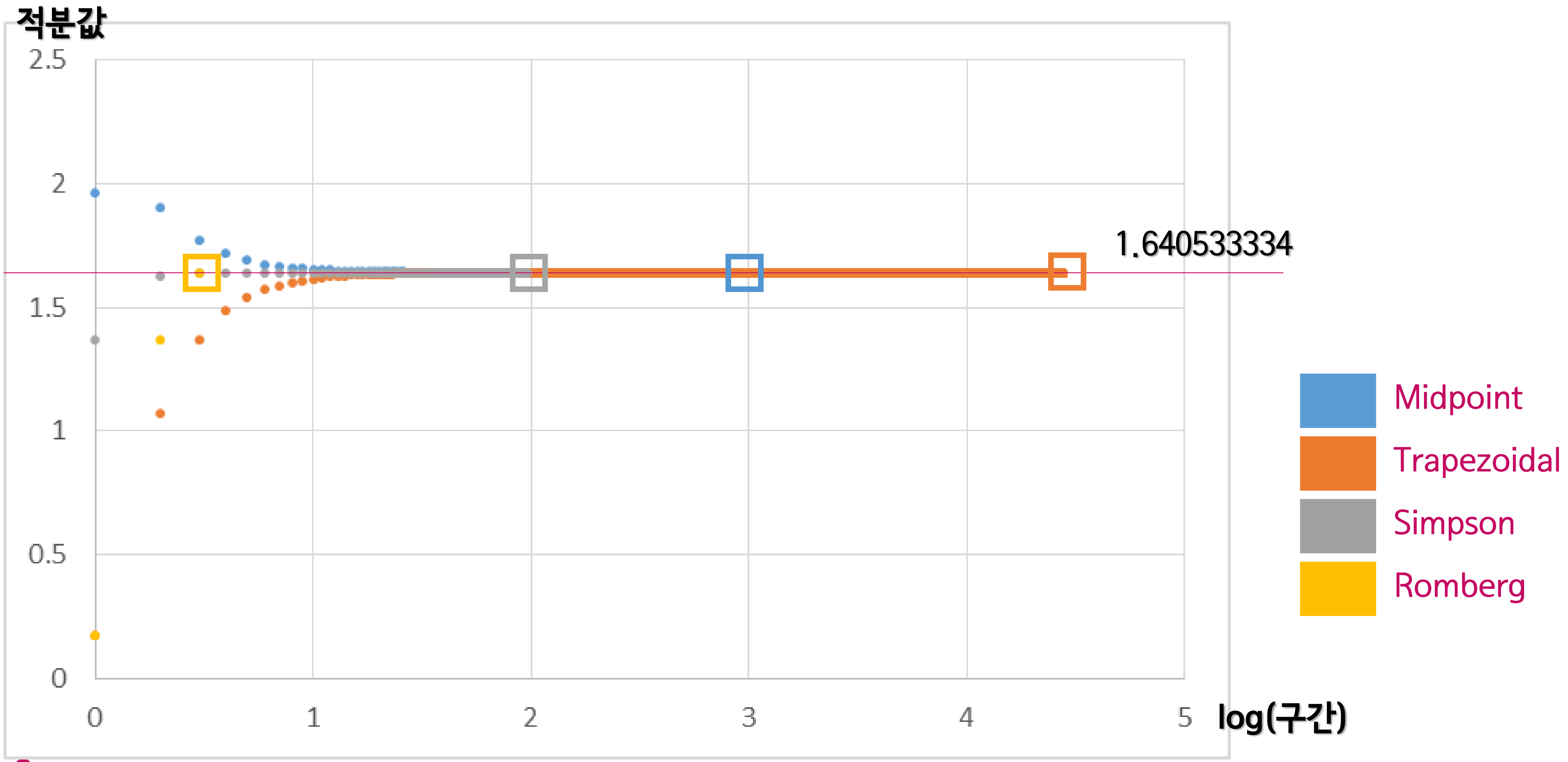
```
==================================================================
Program: Integration of given equation
Date: 2015.05.09
Made by Gilwoon Lee
POSTECH, project 7 of [AMSE417] Numerical analysis for
Development environment: Visual Studio 2013
Code language: C
==================================================================

This programm will integrate given equation.
============================
The range of integration is x = 0 to 0.8.
Allowed error is 0.00000001
* Midpoint:      8764,   1.64053335,     -0.000000010
* Trapezoidal:   27711,  1.64053333,     0.000000010
* Simpson:       96,     1.64053333,     0.000000010
* Romberg:       3,      1.64053333,     0.000000007

============================
계속하려면 아무 키나 누르십시오 . . .
```

log(구간)



3. Result & Conclusion

2. 실행 결과



4. Conclusion

1) Romberg가 가장 좋은 결과를 보여주었으며, Simpson, Midpoint, Trapezoidal 순으로 좋은 결과를 보여주었다.
2) 구간의 수에 따라서 결과를 분석해본 결과 마찬가지로 수렴의 속도가 빠른 방법도 위 순을 따르는 것을 보여주었다.