


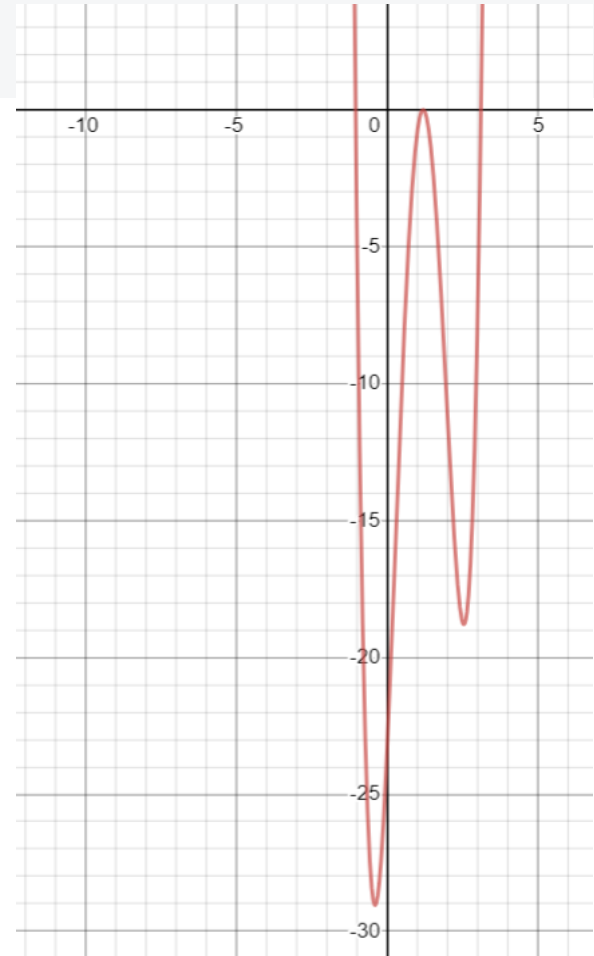


수치해석 HW2

2018008613 안상욱

그래프의 대략적인 모양

 $5x^4 - 22.4x^3 + 15.85272x^2 + 24.161472x - 23.4824832$



- Desmos라는 사이트에서 위 식에 대한 그래프를 대략적으로 그려보았을 때, 최솟값을 갖도록 하는 x 의 값이 -1에서 0 사이에 하나, 2에서 3 사이에 하나 위치하는 것을 확인할 수 있었다.

Using 1st and 2nd derivatives

$$x_{i+1} = x_i - \alpha \frac{f^{(1)}(x_i)}{f^{(2)}(x_i)} + \Delta x_i$$

Moment

먼저, $f'(x)$ 와 $f''(x)$ 를 직접 구한 뒤 Newton Method를 사용해 최솟값을 구해보았다. 여기서 alpha의 값을 0.5로 두고

Moment의 값을 $((x_{i-1}) - (x_i)) / 2$ 로 조금씩 줄어들도록 설정해 최솟값을 넘어서서 진동하지 않도록 구현해주었다.

두 최솟값은 각각 -1과 0 사이, 2와 3 사이에 있었으므로 초기의 x값을 각각 0, 3으로 설정해주었다.

C 프로그램 구현

```
#define LIM 0.000001
#define ALPHA 0.5

double f(double x)
{
    return 5 * x*x*x*x - 22.4*x*x*x + 15.85272*x*x + 24.161472*x - 23.4824832;
}

double f1(double x)
{
    return 20 * x*x*x - 67.2*x*x + 31.70544*x + 24.161472;
}

double f2(double x)
{
    return 60 * x*x - 134.4*x + 31.70544;
}
```

먼저 다음과 같이 f 함수, f' 함수, f'' 함수를 각각 f , $f1$, $f2$ 로 정의해주었다. 그리고 LIM 값은 x_i 에서 x_{i+1} 로의 변화량이 LIM보다 작아질 동안 Newton Method를 반복하기 위해 설정해주었다. 그리고 ALPHA의 값을 앞에서 말한 것처럼 0.5로 설정해주었다.

C 프로그램 구현

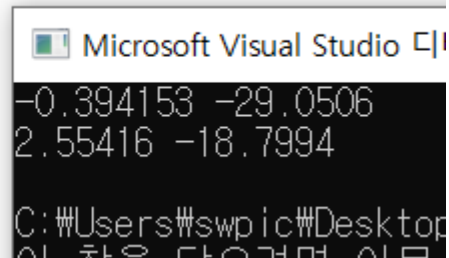
```
double Newton(double x)
{
    double x1;
    double moment = 0;
    while (1)
    {
        x1 = x - ALPHA * f1(x) / f2(x) + moment;
        if (abs(x1 - x) < LIM)
            break;
        moment = (x - x1) / 2;
        x = x1;
    }
    return x1;
}
```

그리고 Newton method를 실행할 Newton 함수를 구현해주었다.

x_1 은 x_{i+1} 을 의미하고, x 는 x_i 를 의미한다. 그리고 moment의 값은 초기값을 0으로 설정해 준 뒤, 한 번 실행할 때마다 $((x_{i-1}) - (x_i)) / 2$ 값으로 변경해주었다. 그리고 $x_{i+1} - x_i$ 의 절댓값이 아까 설정해준 0.000001의 값을 가지는 LIM보다 작아질 때까지 반복해주었다. 반복이 끝나면 x_i 의 값을 return해주었다.

결과

```
int main()
{
    cout << Newton(0) <<" " << f(Newton(0)) << "\n";
    cout << Newton(3) <<" " << f(Newton(3)) << "\n";
}
```



```
Microsoft Visual Studio 디
-0.394153 -29.0506
2.55416 -18.7994
C:\Users\swpic\Desktop
아. 참을. 다음. 려면. 아. 문.
```

0과 3에서 Newton Method를 이용해 최솟값을 찾아보았다. 그 결과 x 가 -0.394153일 때 함수값 -29.0506을 가지고, x 가 2.55416일 때 함수값 -18.7994를 가짐을 확인할 수 있었다. 즉, x 가 -0.394153일 때 global minimum 값인 -29.0506을 가지고, x 가 2.55416일 때 local minimum 값인 -18.7994를 가짐을 확인할 수 있었다.

Using Approximation

$$x_{i+1} = x_i - \alpha \frac{f^{(1)}(x_i)}{f^{(2)}(x_i)} + \Delta x_i$$

Moment

앞의 방법과 유사하게 alpha의 값을 0.5로, Moment의 값을 $((x_{i-1}) - (x_i)) / 2$ 로 동일하게 설정해주었다.

다만, $f''(x)$ 의 값을 구할 때 approximation을 이용해서 값을 구하도록 구현했다.

두 최솟값은 앞의 방법과 동일하게 각각 -1과 0 사이, 2와 3 사이에 있었으므로 초기의 x값을 각각 0, 3으로 설정해주었다.

C 프로그램 구현

```
#define LIM 0.000001
#define ALPHA 0.5
#define H 0.0000001

double f(double x)
{
    return 5 * x*x*x*x - 22.4*x*x*x + 15.85272*x*x + 24.161472*x - 23.4824832;
}

double f1(double x)
{
    return 20 * x*x*x - 67.2*x*x + 31.70544*x + 24.161472;
}

double af2(double x)
{
    return (f(x + H) - 2 * f(x) + f(x - H)) / (H*H);
}
```

앞에서 말했듯이 f, f1 함수는 동일하게 구현했고, ALPHA 값과 LIM값을 동일하게 설정해주었다. 그러나 f2 함수 대신 approximation을 이용해 f''(x)의 값을 얻도록 af2 함수를 구현했다. 이 때 H의 값을 0.0000001로 설정해주었다.

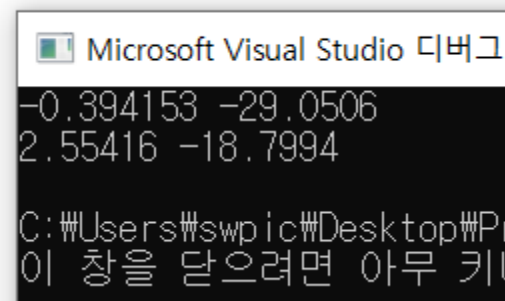
C 프로그램 구현

```
double Newton(double x)
{
    double x1;
    double moment = 0;
    while (1)
    {
        x1 = x - ALPHA * f1(x) / af2(x) + moment;
        if (abs(x1 - x) < LIM)
            break;
        moment = (x - x1) / 2;
        x = x1;
    }
    return x1;
}
```

Newton 함수도 동일하게 구현해주었는데, $f_2(x)$ 함수 대신 $af_2(x)$ 함수를 이용했다.

결과

```
int main()
{
    cout << Newton(0) << " " << f(Newton(0)) << "\n";
    cout << Newton(3) << " " << f(Newton(3)) << "\n";
}
```



Microsoft Visual Studio 디버그

```
-0.394153 -29.0506
2.55416 -18.7994
```

C:\Users\swpic\Desktop\P...
이 창을 닫으려면 아무 키

0과 3에서 Newton Method를 이용해 최솟값을 찾아보았다. 그 결과 x 가 -0.394153일 때 함수값 -29.0506을 가지고, x 가 2.55416일 때 함수값 -18.7994를 가짐을 확인할 수 있었다. 즉, x 가 -0.394153일 때 global minimum 값인 -29.0506을 가지고, x 가 2.55416일 때 local minimum 값인 -18.7994를 가짐을 확인할 수 있었다.

비교 및 분석

```
int main()
{
    cout << Newton(0) << " " << f(Newton(0)) << "\n";
    cout << Newton(3) << " " << f(Newton(3)) << "\n";
}
```

Microsoft Visual Studio 디버그

```
-0.394153 -29.0506
2.55416 -18.7994
C:\Users\swpic\Desktop\P
이 창을 닫으려면 아무 키
```

```
int main()
{
    cout << Newton(0) << " " << f(Newton(0)) << "\n";
    cout << Newton(3) << " " << f(Newton(3)) << "\n";
}
```

Microsoft Visual Studio 디버그

```
-0.394153 -29.0506
2.55416 -18.7994
C:\Users\swpic\Desktop\P
이 창을 닫으려면 아무 키
```

결과를 비교해봤을 때 Using 1st and 2nd derivatives로 최솟값을 구했을 때와 Using Approximation로 최솟값을 구했을 때의 값이 정확히 일치하는 것을 확인해볼 수 있었다.

실제 값과 비교

Result:

$$20x^3 - 67.2x^2 + 31.7054x + 24.1615$$

Roots:

$$x \approx -0.394153$$

$$x \approx 1.2$$

$$x \approx 2.55415$$

```
int
```

```
int main()
```

```
{  
    cout << Newton(0) << " " << f(Newton(0)) << "\n";
```

```
    cout << Newton(3) << " " << f(Newton(3)) << "\n";  
}
```

Microsoft Visual Studio 디버그

```
-0.394153 -29.0506  
2.55416 -18.7994
```

```
C:\Users\sswpc\Desktop\P  
이 창을 닫으려면 아무 키
```

방정식의 근을 구해주는 사이트에서 $f'(x)$ 의 근을 구해서 극소, 극대를 가지는 점을 찾아보았다. 그 결과 앞에서 구한 Local minimum, Global minimum을 가지는 x 값과 거의 일치하는 것을 확인해 볼 수 있었다.

감사합니다!