HW2 REPORT

2016024893 오성준

INTUITION

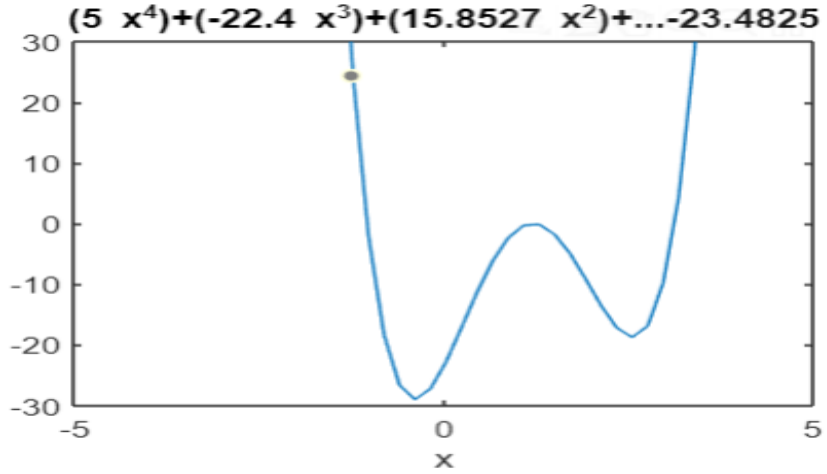
```
p1 =

   20.0000   -67.2000   31.7054   24.1615

>> roots(p1)

ans =

    2.5542
    1.2000
   -0.3942
```



- MATLAB을 이용해서 함수의 극값의 X좌표를 구해보면 -0.3942 1.2 2.5542가 나온다.
- 이 때, 그래프의 개형상 0보다 작은 값에서 GLOBAL MINIMUM을 가지므로, 우리가 구하고자 하는 값은 -0.3942라고 할 수 있다.
- 이 때, 2.5542는 LOCAL MINIMUM이라고 할 수 있다.
- 극소를 가지는 극값이 두개이므로, 함수를 2번씩만 돌리면 될 것이다.

ABOUT CODE-EXACT FUNCTION(1)

```
10      double th = 0.000001;
11      double alpha = 0.0001;
12
13      double poly[5] = { 5,-22.4,15.85272,24.161472,-23.4824832 };
14      double poly_diff[4] = { 20,-22.4 * 3,15.85272 * 2,24.161472 };
15      double poly_ddiff[3] = { 60,-22.4 * 3 * 2, 15.85272 * 2 };
16
17      double h = 0.00001;
18
19      double fun(double x)
20      {
21          double ret = 0;
22
23          for (int i = 0; i < 5; i++)
24          {
25              ret += poly[i] * pow(x, 4 - i);
26          }
27          return ret;
28      }
29
30      double fun1(double x)
31      {
32          double ret = 0;
33          for (int i = 0; i < 4; i++)
34          {
35              ret += poly_diff[i] * pow(x, 3 - i);
36          }
37          return ret;
38      }
39      double fun2(double x)
40      {
41          double ret = 0;
42          for (int i = 0; i < 3; i++)
43          {
44              ret += poly_ddiff[i] * pow(x, 2 - i);
45          }
46          return ret;
47      }
48
```

- TH와 ALPHA값을 다음과 같이 잡았다. 물론 이 값들이 작을수록 더 정확한 값을 구할 수 있겠지만, 반대로 코드가 너무 느려져서 이 정도의 값을 정했다. 코드를 구현하는 목적에 따라서 값들을 조정할 수 있을 것 같다.
- 도함수와 이계도 함수의 계수들을 구해서 함수 값을 계산하는 부분을 구현했다.

ABOUT CODE-EXACT FUNCTION(2)

```
void newton(double x)
{
    if (fun1(x) == 0)
    {
        newton_min.push_back(x);
        return;
    }

    double x_ = x - fun1(x)*alpha / fun2(x);
    double err = fabs((x_ - x) / x_);
    while (err * 100 >= th)
    {
        x = x_;
        x_ = x - fun1(x)*alpha / fun2(x);
        if (fun1(x_) == 0) break;
        err = fabs((x_ - x) / x_);
    }
    newton_min.push_back(x_);
}
```

- 전체적으로 지난번에 했던 과제와 형태가 똑같다. 다만 지금 내가 값을 구하는 함수는 이전 과제의 도함수이고 그로 인해서 이계도함수를 사용하는 것이 다를 뿐이다.
- MAIN함수에서는 -5와 5를 대입해서 LOCAL MINIMUM값을 2개를 구했다.

ABOUT CODE-APPROXIMATION(1)

```
double appf1(double x)
{
    double ret = fun(x + h) - fun(x);
    return ret / h;
}
double appf2(double x)
{
    double ret = fun(x + h) - 2 * fun(x) + fun(x - h);
    return ret / (h * h);
}
```

- APPF1과 APPF2는 각각 도함수와 이계도 함수의 APPROXIMATION을 구하는 함수이다.
- 이 때, 너무 당연한 얘기이지만 H값을 작게 할 수록 더더욱 원래의 값에 가까워질 것이다. H가 0으로 가는 극한이 각각의 함수의 정의이기 때문이다.
- H값은 0.00001로 설정하였다.

ABOUT CODE-APPROXIMATION(2)

```
void newton_approximation(double x)
{
    double one = appf1(x);
    double two = appf2(x);

    if (one == 0)
    {
        newton_app_min.push_back(x);
        return;
    }

    double x_ = x - one * alpha / two;
    double err = fabs((x_ - x) / x_);
    while (err * 100 >= th)
    {
        x = x_;
        one = appf1(x);
        two = appf2(x);
        x_ = x - one * alpha / two;
        if (fun1(x_) == 0) break;
        err = fabs((x_ - x) / x_);
    }
    newton_app_min.push_back(x_);
}
```

- 큰 차이는 없지만, 대입값들을 APPROXIMATION한 값들로 대체해서 사용한다는 차이점이 있다. 하지만, 큰 차이는 없는 것 같다.

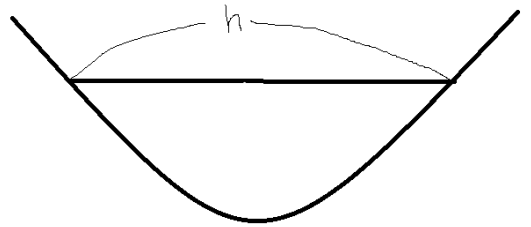
RESULT

```
exact func minimum point
-0.394193 2.55441
ans : -0.394193

approximation minimum point
-0.394198 2.5544
ans : -0.394198
```

- 두개의 METHOD 다 실제 값에 근사한 LOCAL MINIMUM들을 구해냈고, 실제로 GLOBAL MINIMUM의 좌표와 근사한 값들을 구해냈다.
- 이 코드에서 쓰인 TH, H, ALPHA와 같은 값들을 줄여주면 더 정확한 값을 구할 수 있겠지만, 속도가 더 느려진다는 단점이 있었다. 코딩의 목적과 조건에 따라서 다르게 값을 설정할 수 있을 것이다.
- 그리고 어느 METHOD가 더 정확한지는 이 예시로는 판단하기 어렵다. GLOBAL MINIMUM은 APPROXIMATION이 LOCAL은 EXACT FUNC이 더 정확하게 구했고 그 차이도 의미가 없는 수준이기 때문이다.

COMMENT ABOUT METHOD



EXACT FUNCTION

- 변곡점은 이번에도 조심해야하는 점이다. 만약 변곡점에서 도함수 값이 0이라면 이는 LOCAL MINIMUM조차 될 수 없다. 왜냐면 변곡점에서는 함수의 증감이 변화하지 않기 때문이다. LOCAL MINIMUM을 구할 때 조차 조심해서 사용해야 할 것이다.
- 역시나 도함수의 변곡점 또한 조심해서 계산해야 할 것이다. NEWTON METHOD는 함수의 개형에 굉장히 예민하기 때문이다. 대입하는 값을 조심해야 할 것

APPROXIMATION

- EXACT FUNCTION과 똑같은 내용을 가진다.
- 다만, H값을 조심해서 설정할 필요가 있는데, H값이 너무 작다면 속도가 너무 느려질 것이고, H값이 너무 크다면 오른쪽 위 그림과 같은 상황에서 근사하지도 않은 값의 도함수 값을 0으로 계산해버리는 대참사가 일어날 수 있다. 그러므로 H값을 잘 조절하는 것이 APPROXIMATION의 큰 과제이다.
- 다만 정확도 자체는 잘 구현한다면, EXACT FUNCTION과 큰 차이가 나지 않았다. 물론 정확한 값이 아니므로 비교적 부정확하지 않을까라고 예측해 본다.