HW1 REPORT

2016024893 오성준

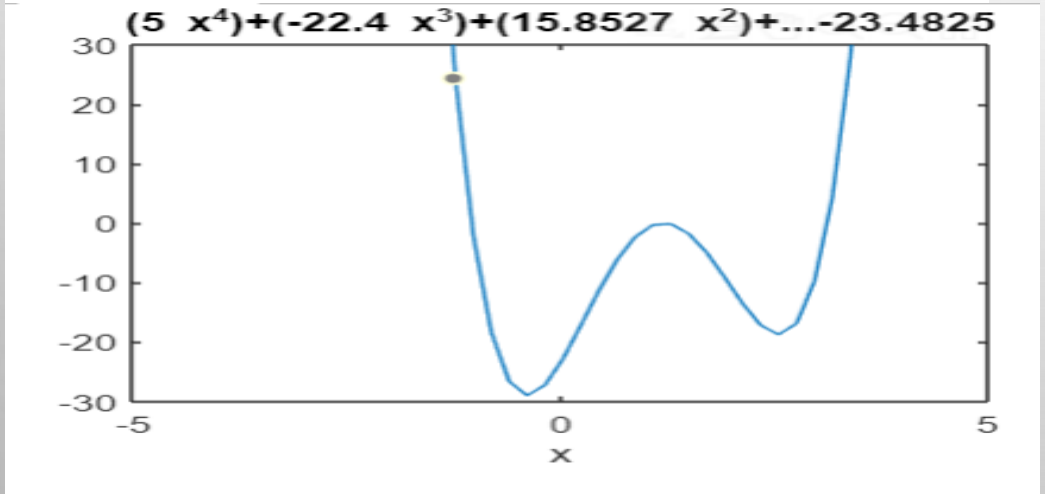
ABOUT CODE-BISECTION(1)

```
void bisection(double left, double right)
{
    if (fun(left) * fun(right) >= 0)
    {
        cout << left << " " << right << " : ";
        cout << "wrong input₩n";
        return;
    }

    double mid=left, mid_=right;
    double err = fabs((mid - mid_)/mid);
    while (err * 100 >= th)
    {
        mid_ = mid;
        mid = (left + right) / 2;
        if (fun(mid) == 0) break;
        else if (fun(mid) * fun(left) < 0) right = mid;
        else left = mid;
        err = fabs((mid - mid_) / mid);
    }

    bisection_root.push_back(mid);
}
```

- BRACKET을 통해서 BISECTION을 구현했습니다.
- BRACKET에서 곱이 양수가 되면 WRONG INPUT을 프린트하도록 만들었습니다.
- MID_가 X_NEW에 해당하고 MID가 X_OLD에 해당하는 변수입니다.
- STOP SIGN에 해당하는 TH는 0.0001입니다.

ABOUT CODE-BISECTION(2)

```
//Bisection
bisection(0, 3);
bisection(-5,0);
bisection(0, 5);
cout << "bisection₩n";
for (double i : bisection_root) cout << i << endl;
```



- 3가지 범위로 나누어서 BISECTION을 실행했습니다.
- F(-5)>0, F(5)>0, F(0)<0이라서 -5와 0 그리고 0과 5사이를 LEFT RIGHT로 잡아서 실행했습니다. 이렇게 양쪽 끝에 있는 근을 구할 수 있었습니다.
- 하지만 이 때 그 사이에 있는 두근은 구해지지 않았고 극값을 구해본 결과 1.2에서 주어진 함수가 근을 가지는 것을 알게 되었습니다. 0,3은 BISECTION이 극값이 근일 경우 구할 수 없다는 걸 보이기 위해서 넣었습니다.

ABOUT CODE-NEWTON(1)

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

    double x_ = x - fun(x) / fun_(x);
    double err = fabs((x_ - x) / x_);
    while (err * 100 >= th)
    {
        x = x_;
        x_ = x - fun(x) / fun_(x);
        if (fun(x_) == 0) break;
        err= fabs((x_ - x) / x_);
    }
    newton_root.push_back(x_);
}
```

- PPT에 있는데로 함수를 구현했습니다.
- 이 때 쓰이는 TH는 BISECTION에서 쓴 것과 동일합니다.
- FUN_은 도함수의 함수 값을 나타냅니다.

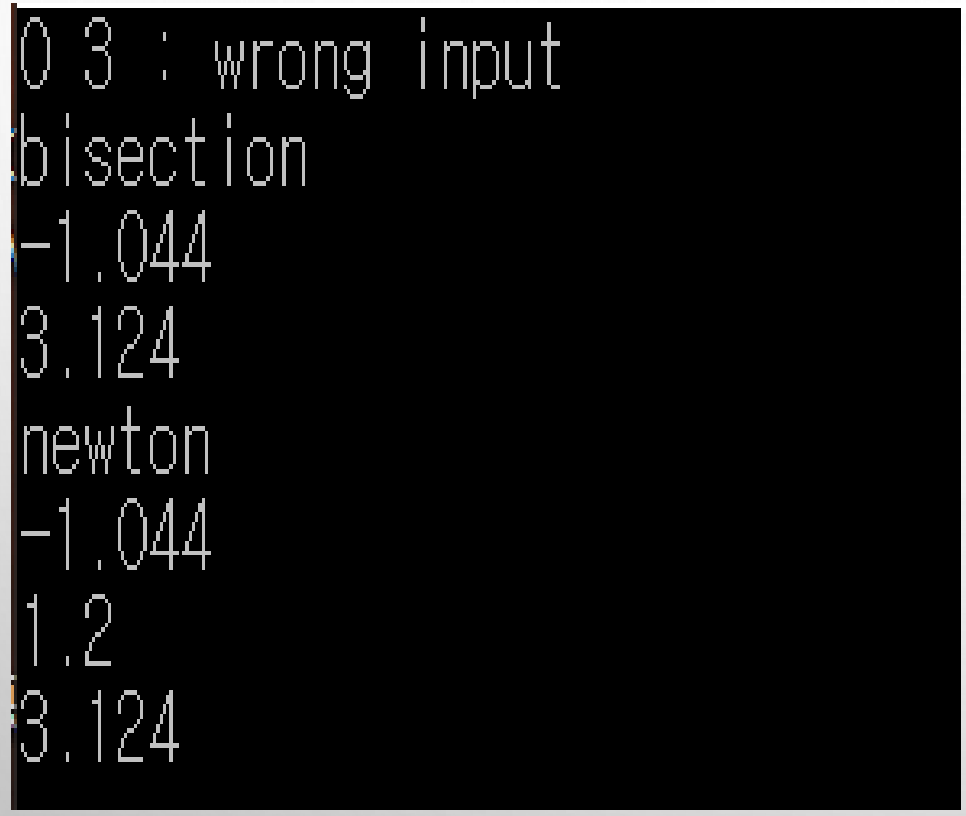
ABOUT CODE-NEWTON(2)

```
//Newton
newton(-5);
newton(0.3);
newton(2.6);
cout << "newton₩n";
for (double i : newton_root)cout << i << endl;
```

```
>> roots(p)

ans =

    2.5542
    1.2000
   -0.3942

>> p=[60 -22.4*6, 15.85272*2]

p =

   60.0000 -134.4000   31.7054

>> roots(p)

ans =

    1.9720
    0.2680

>> |
```

- -5와 0.3 2.6을 사용해서 NEWTON METHOD를 실행했습니다.
- 이계도함수로 변곡점의 X좌표를 구한 결과 1.9720과 0.2680이 나왔고 NEWTON METHOD에선 문제가 있을 수 있기 때문에 이 값들을 피하는 선에서 값을 정했습니다.
- 이미 BISECTION에서 1.2에서 중근을 가진다는 것을 알았기 때문에 3개의 값만을 집어 넣었습니다.

RESULT



- BISECTION은 0과 3범위에 있는 1.2를 구할 수가 없었습니다. 중근은 구할 수 없기 때문입니다. 대신 나머지 값들은 구했습니다.
- NEWTON METHOD는 모든 값을 구할 수 있었습니다.

COMMENT ABOUT METHOD

BISECTION

- 중근을 구할 수 없습니다.
- 그렇다면 중근이 존재하는지 확인하기 위해서 극 값을 구해야 하는데, 이 때, 도함수가 우리가 값을 구하기 쉬운 함수라는 보장이 없습니다. 반복해서 미분을 해야 하는 불상사가 발생할 수 있을 것 같습니다.
- 범위를 잡기위해서 여러 개의 값을 넣어봐야하는 상황이 생길 수도 있습니다. 특히 근의 개수가 정해져 있지 않다면, 근들이 촘촘히 모여 있다면 전부를 구하기 힘들 수도 있을 것 같습니다.

NEWTON METHOD

- 그래도 중근을 구할 수 있습니다. 점들만 잘 잡는다면 무리 없이 근들을 구할 수 있습니다.
- 다만 그를 위해서 함수를 그려서 판단해야 하는 점과 그 마저도 그리기 힘들 수 있다는 게 제 의견입니다. 특히나 이계도 함수의 값들을 체크해줘야 하는데, 함수가 복잡해지는 순간 저희 수준에서는 힘들 수도 있을 것 같습니다.

비고

- 제가 REPORT에 사용한 MATLAB들은 그저 단지 구한 값이 맞는지 확인하기 위해서 사용한 것임을 알려드립니다. REPORT에 쓴 것은 손으로 계산한 것보단 가독성이 좋을것이라 생각해서 입니다.
- 감사합니다.