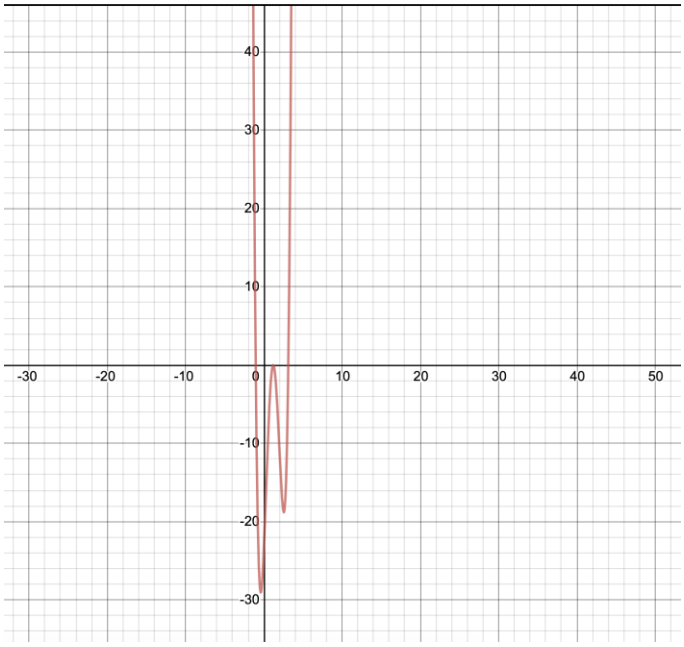
HW2

2017029970 우원진



이번에도 최솟값을 찾기 위한 대략적인 그래프의 그림을 알기위해 Desmos라는 사이트를 이용했다.

```
def function(x):
    return (
        5 * pow(x, 4)
        - 22.4 * pow(x, 3)
        + 15.85272 * pow(x, 2)
        + 24.161472 * x
        - 23.4824832
    )
```

먼저 위의 사진처럼 주어진 식에 x값을 대입 했을때의 값을 구하기 위해 function 이라는 함수를 만들었다.

```python
def exactFirstDerivative(x):
    return 20 * pow(x, 3) - 67.2 * pow(x, 2) + 31.70544 * x + 24.161472


def exactSecondDerivative(x):
    return 60 * pow(x, 2) - 134.4 * x + 31.70544


def exactDerivative(x):
    variable = x
    while True:
        result = exactFirstDerivative(variable)
        if result <= 0.0001 and result >= -0.0001:
            return variable

        variable = variable - exactFirstDerivative(variable) / exactSecondDerivative(
            variable
        )
```

먼저 계산을 통해 정확한 미분 했을때의 식을 구하여 값을 구하는 과정이다. exactFirstDerivative와 exactSecondDerivative 함수는 각각 주어진함수를 한번 미분 했을때와 두번 미분했을때의 식이다. 여기서 오차의 범위는 0.0001로 두었고, 알파값은 테일러 시리즈를 가지고 구했을때는 2가 나왔지만 임의로 1로 두었다. 모멘텀은 따로 두지 않았다.

```python
def approximationFirstDerivative(x):
    return (function(x + 0.01) - function(x)) / 0.01


def approximationSecondDerivative(x):
    return (function(x + 0.01) - 2 * function(x) + function(x - 0.01)) / 0.0001


def approximationDerivative(x):
    variable = x
    while True:
        result = approximationFirstDerivative(variable)
        if result <= 0.0001 and result >= -0.0001:
            return variable
        variable = variable - approximationFirstDerivative(
            variable
        ) / approximationSecondDerivative(variable)
```

위의 그림은 정확한 미분이 아닌 미분을 하는 과정인 근사치를 통해 하는것이다. 여기서 h는 0.01로 뒀다. 처음에 h를 0.1로 두니 오차가 너무크게 발생해서 조금 줄였다. 마찬가지로 결과값의 오차허용범위는 0.0001로 두었고, 알파는 1, 모멘텀은 따로 두지않았다.

```python
def main():
    print("Using ExactDerivative")
    print("Find using exactDerivative with -1 : " + str(exactDerivative(-1)))
    print("Find using exactDerivative with 2 : " + str(exactDerivative(2)))
    print("")
    print("Using Approximation")
    print(
        "Find using approximateDerivative with -1 : " + str(approximationDerivative(-1))
    )
    print(
        "Find using approximateDerivative with 2 : " + str(approximationDerivative(2))
    )


if __name__ == "__main__":
    main()
```

이는 그래프의 모형을 통해 최솟값이 -1, 2근처에 있다는 것을 알아냈고 이를가지고 결과 값을 구하는 과정이다.

아래는 결과값이다.

```
uwonjin-ui-MacBookPro:수치해석 과제 woowonjin$ python hw2.py
Using ExactDerivative
Find using exactDerivative with -1 : -0.394153721168
Find using exactDerivative with 2 : 2.55415353986

Using Approximation
Find using approximateDerivative with -1 : -0.399145359023
Find using approximateDerivative with 2 : 2.5491443549
uwonjin-ui-MacBookPro:수치해석 과제 woowonjin$
```

두 가지 중에서 확실히 실제 미분값을 가지고 결과값을 구한것이 실제값에 더욱 근접했다.