수치해석 HW#1

Bisection과 Newton-Raphson법의 비교

Python 사용

컴퓨터소프트웨어학부 2015005187 최철훈

목차

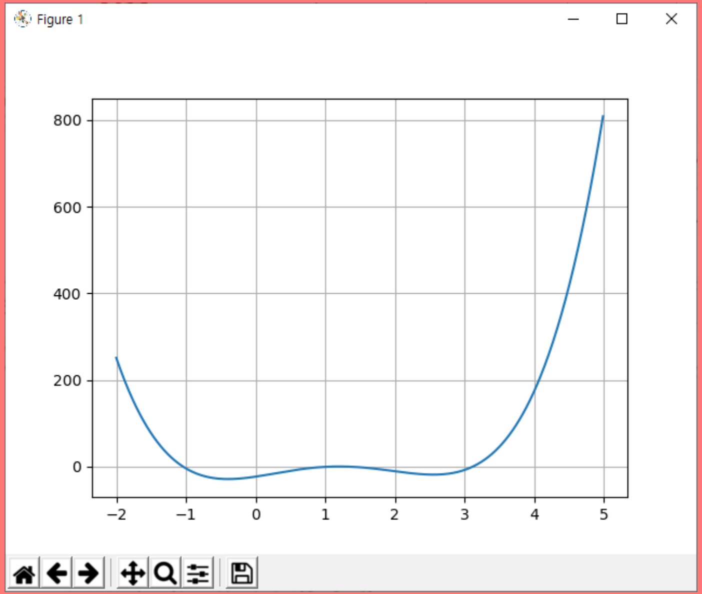
근 추측하기

먼저, f(x) = $5x^4 - 22.4x^3 + 15.85272x^2 + 24.161472x - 23.4824832$의 근의 위치를 추측하기 위해 그래프를 그렸다.

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(-2, 5, 0.01)
y = 5*x**4 - 22.4*x**3 + 15.85272*x**2 + 24.161472*x - 23.4824832

plt.figure()
plt.plot(x, y)
plt.grid()
plt.show()
```

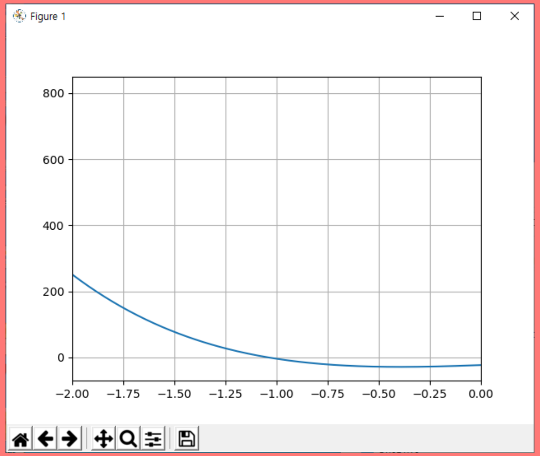


위 그래프를 보면 f(x)의 근은 -1, 1, 3에 근접해 있으며 1근방에서 중근을 가짐을 확인할 수 있다.

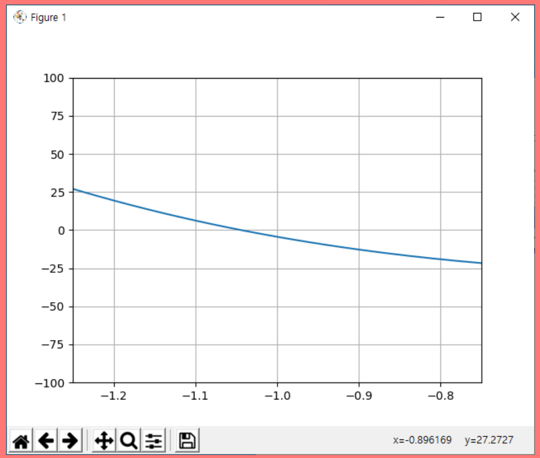
근 추측하기 -1 근방의 근 추측

-1 근방의 근을 좀 더 정확하게 알기 위해 -1 근방을 점점 확대하여 그래프를 그렸다.

```
plt.xlim(-2, 0)
```

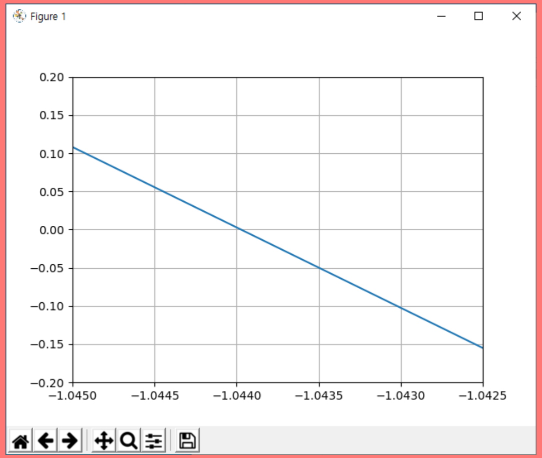



```
plt.xlim(-1.25, -0.75)
plt.ylim(-100, 100)
```



.....

```
plt.xlim(-1.045, -1.0425)
plt.ylim(-0.2, 0.2)
```



위와 같이 확대하여 -1.044에 근사한 값임을 알 수 있다.

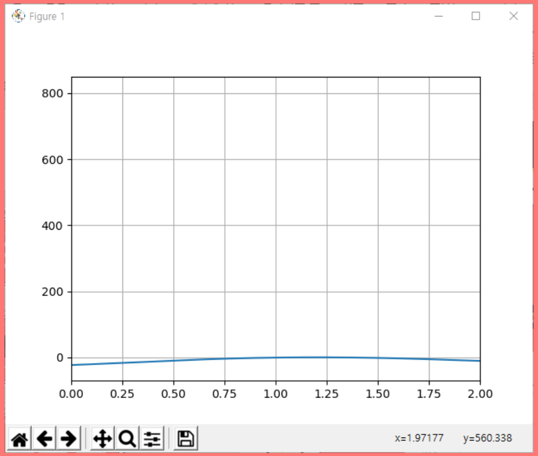
근 추측하기 1 근방의 근 추측

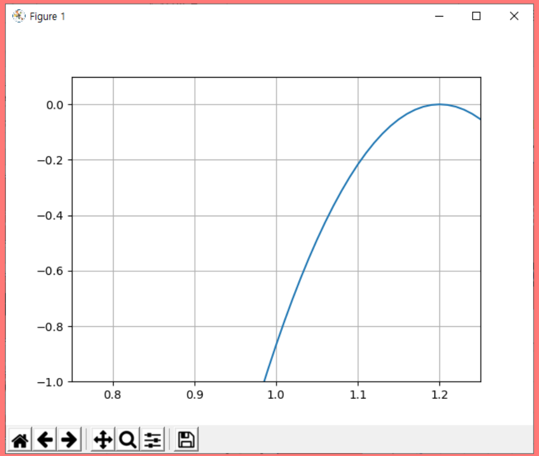
1 근방의 근을 좀 더 정확하게 알기 위해 1 근방을 점점 확대하여 그래프를 그렸다.

```
plt.xlim(0, 2)
```

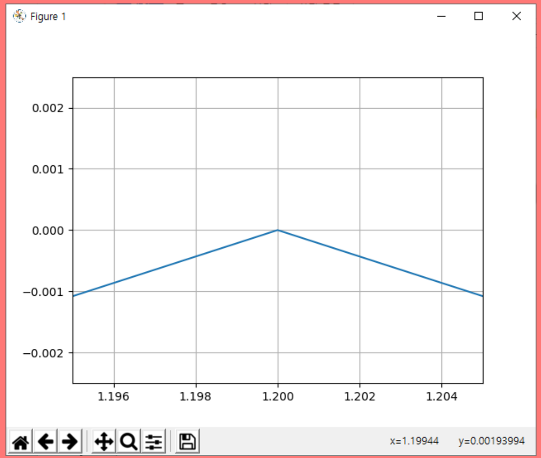
```
plt.xlim(0.75, 1.25)
plt.ylim(-1, 0.1)
```

```
plt.xlim(1.195, 1.205)
plt.ylim(-0.0025, 0.0025)
```





.....

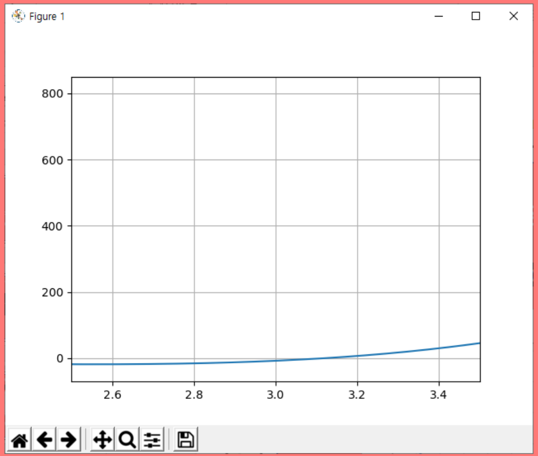


위와 같이 확대하여 1.2에 근사한 값임을 알 수 있다.

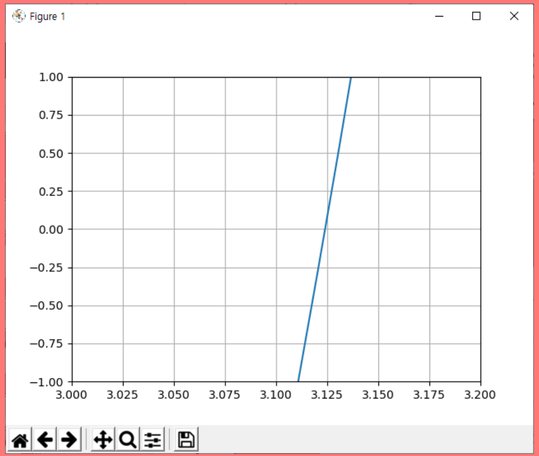
근 추측하기 3 근방의 근 추측

3 근방의 근을 좀 더 정확하게 알기 위해 3 근방을 점점 확대하여 그래프를 그렸다.

```
plt.xlim(2.5, 3.5)
```

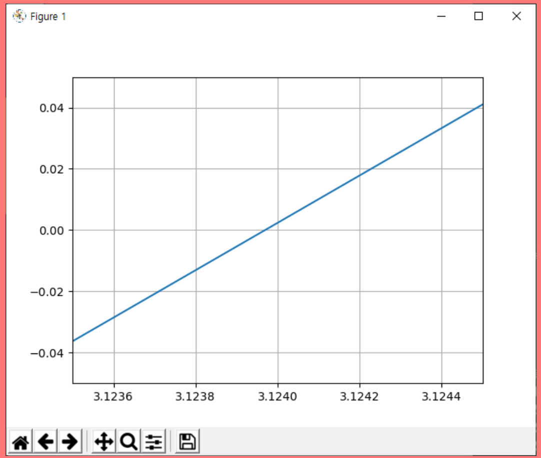



```
plt.xlim(3.0, 3.2)
plt.ylim(-1, 1)
```



.....

```
plt.xlim(3.1235, 3.1245)
plt.ylim(-0.05, 0.05)
```



위와 같이 확대하여 3.124에 근사한 값임을 알 수 있다.

Bisection을 이용하여 근 구하기 코드

Bisection을 구현한 코드이다. 정확한 근을 찾지 못하면 interval이 0.0001보다 작을 경우 interval을 반환하도록 하였다.

```
def fx(x):
    result = 5*x**4 - 22.4*x**3 + 15.85272*x**2 + 24.161472*x - 23.4824832
    return result
min = float(input("Input interval's Min value : "))
max = float(input("Input interval's Max value : "))
sc = fx(min) * fx(max)
while True:
    if min >= max:
        print("Your input is wrong interval")
        min = float(input("Input interval's Min value : "))
        max = float(input("Input interval's Max value : "))
        sc = fx(min) * fx(max)
        continue

    if sc == 0:
        if fx(min) == 0:
            print("The root is", min)
            break
        elif fx(max) == 0:
            print("The root is", max)
            break

    elif sc > 0:
        print("Your input is wrong interval")
        min = float(input("Input interval's Min value : "))
        max = float(input("Input interval's Max value : "))
        sc = fx(min) * fx(max)
        continue
```

```
elif sc < 0:
    if max - min < 0.0001:
        print("The root exists between %f, %f" % (min, max))
        break
    while True:
        max2 = max - (max - min)/2
        sc = fx(min) * fx(max2)
        if sc > 0:
            min = max2
            sc = fx(min) * fx(max)
            continue

        elif sc < 0:
            if max - min < 0.0001:
                print("The root exists between %f, %0.12f" % (min, max))
                break
            max = max2
            continue

        elif sc == 0:
            if fx(min) == 0:
                print("The root is", min)
                break
            elif fx(max2) == 0:
                print("The root is", max2)
                break
    break
```

Bisection을 이용하여 근 구하기 결과

각각 근사치를 중심으로 interval을 5번 이등분하는 초기 interval로 설정하였다.

```
We will solve the equation 5x^4 - 22.4x^3 + 15.85272x^2 + 24.161472x - 23.4824832
What method do you want?
Bisection : 1, Newton-Raphson : 2
Your Choice : 1
Your choice is Bisection
Input interval's Min value : -1.055
Input interval's Max value : -1.023
The root is -1.044
```

-1.044가 근이다.

```
We will solve the equation 5x^4 - 22.4x^3 + 15.85272x^2 + 24.161472x - 23.4824832
What method do you want?
Bisection : 1, Newton-Raphson : 2
Your Choice : 1
Your choice is Bisection
Input interval's Min value : -0.9
Input interval's Max value : 2.3
Your input is wrong interval
Input interval's Min value :
```

1.2는 중근이어서 sign change가 일어나지 않으므로 값을 찾지 못한다.

```
We will solve the equation 5x^4 - 22.4x^3 + 15.85272x^2 + 24.161472x - 23.4824832
What method do you want?
Bisection : 1, Newton-Raphson : 2
Your Choice : 1
Your choice is Bisection
Input interval's Min value : 3.003
Input interval's Max value : 3.135
The root exists between 3.123979, 3.124042968750
```

근이 3.123979와 3.124042968750 사이에 있다.

Newton-Rhapson을 이용하여 근 구하기 코드

Newton-Rhapson법을 구현한 코드이다. 정확한 근을 찾지 못하면 x_{i+1}과 x_i의 차이가 0.0001%의 오차율을 보이면 이 때의 x_{i+1}값을 반환하도록 하였다.

```python
init = float(input("Input initial value : "))
x = symbols('x')
fx = 5*x**4 - 22.4*x**3 + 15.85272*x**2 + 24.161472*x - 23.4824832
while True:
    v = fx.subs({x: init})
    if v == 0:
        print("The root is", init)
        break
    elif v != 0:
        lim = Derivative(fx, x)
        d = lim.doit().subs({x: init})
        nv = init - v/d
        if abs((nv - init)/nv) * 100 < 0.0001:
            print("The approximated root is", nv)
            break
        init = nv
```

Newton-Rhapson을 이용하여 근 구하기 결과

각각 근사치에서 가장 가까운 정수를 택하여 initial value로 설정하였다.

```
We will solve the equation 5x^4 - 22.4x^3 + 15.85272x^2 + 24.161472x - 23.4824832
What method do you want?
Bisection : 1, Newton-Raphson : 2
Your Choice : 2
Your choice is Newton-Raphson
Input initial value : -1
The approximated root is -1.04400000000000
```

-1.044가 근이다.

```
We will solve the equation 5x^4 - 22.4x^3 + 15.85272x^2 + 24.161472x - 23.4824832
What method do you want?
Bisection : 1, Newton-Raphson : 2
Your Choice : 2
Your choice is Newton-Raphson
Input initial value : 1
The approximated root is 1.19999923518475
```

1.19999923518475가 근사값이다.

```
We will solve the equation 5x^4 - 22.4x^3 + 15.85272x^2 + 24.161472x - 23.4824832
What method do you want?
Bisection : 1, Newton-Raphson : 2
Your Choice : 2
Your choice is Newton-Raphson
Input initial value : 3
The approximated root is 3.12400000000075
```

3.12400000000075가 근사값이다.

Bisection과 Newton-Rhapson의 비교

1. Bisection은 중근을 구할 수 없지만 Newton-Rhapson법은 구할 수 있다.

2. Bisection은 interval과 stop condition에 따라서 얼마나 정확한 근을 구할지가 결정되지만 Newton-Rhapson법은 근과 initial value사이에 변곡점만 있지 않으면 stop condition에 따라서만 얼마나 정확한 근을 구할지가 결정된다.

마무리

감사합니다.

https://github.com/cheol-hoon/Numerical_Analysis