수치해석
(2019학년도 1학기)

[4주/1차시 학습내용]: Ch. 5. Bracketing Methods : Graphical Method

국민대학교 소프트웨어학부

Problem Statement

- 번지 점퍼 회사에서 문제가 발생했다.
- 번지 점퍼를 하는 사람들 중에 심각한 척추 부상을 당하는 경우가 다수 발생하였다.
- 그래서 번지 점퍼 회사는 여러분을 회사의 신입 사원으로 입사 시켜, 이러한 문제를 해결하도록 업무를 지시하였다.

```
Problem of significant vertebrae injury if the free-fall velocity exceeds
36 m/s after 4 s of free fall.
```

Find out the mass

- Find out the mass that exceeds the 36m/s after 4sec.
- m에 68, 69, 70,...을 넣어 4초에 36m/s 를 지나는 지 확인한다.

```
g=9.8; cd=0.25; t=4
m=68
np.sqrt(g*m/cd)*np.tanh(np.sqrt(g*cd/m)*t)
33.07583215932410 8

m=69
v=np.sqrt(g*m/cd)*np.tanh(np.sqrt(g*cd/m)*t)

33.14823174860153 2

m=70
v=np.sqrt(g*m/cd)*np.tanh(np.sqrt(g*cd/m)*t)
33.24783965767994
```

Graphical Method

- 데이터 시각화 (Data Visualization)에 의해 근을 구하는 방법을 소개한다.

Graphical Method

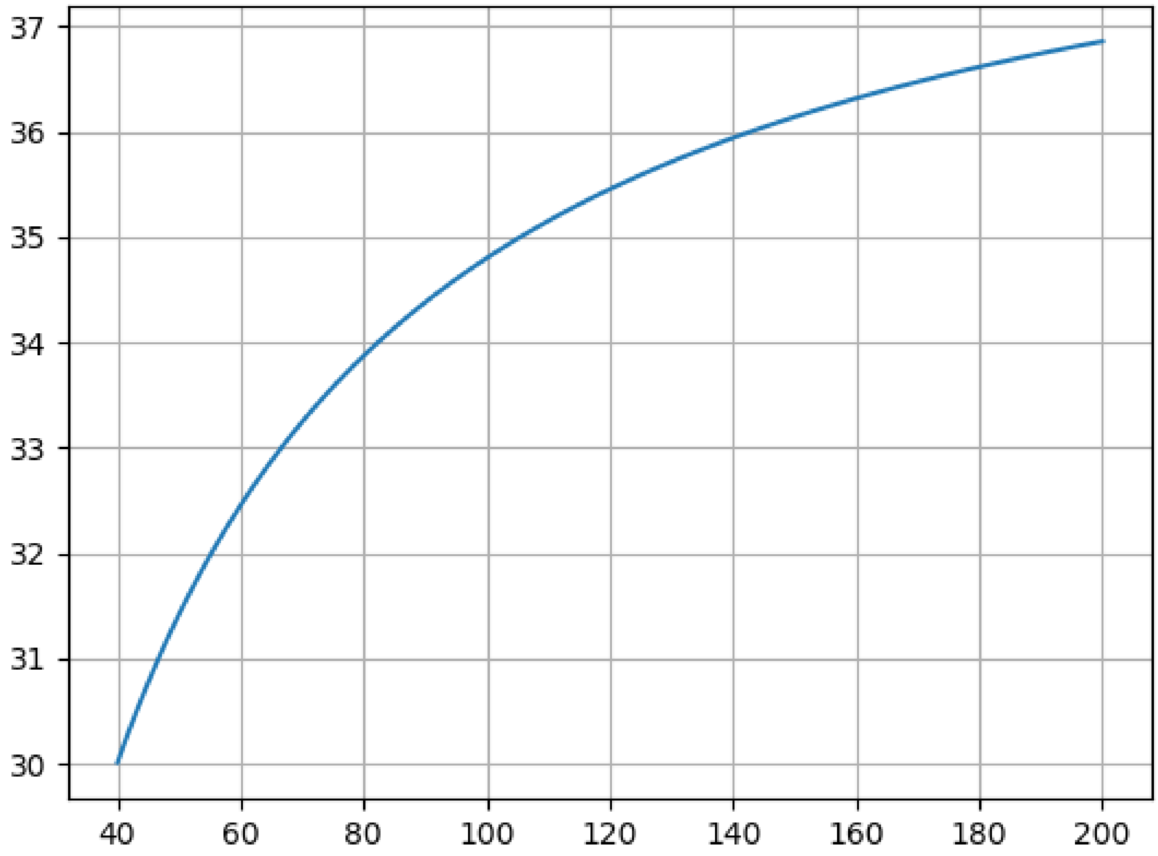
- https://github.com/SCKIMOSU/Numerical-Analysis/blob/master/Data_visualization.py

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

g=9.81; cd=0.25; t=4; v=36

m=np.linspace(40, 200, 100)
fm=np.sqrt(g*m/cd)*np.tanh(np.sqrt(g*cd/m)*t)
plt.figure(1)

plt.plot(m, fm)
plt.grid()
plt.show()

fm1=np.sqrt(g*m/cd)*np.tanh(np.sqrt(g*cd/m)*t)-v
k=np.linspace(0,0,100)
plt.figure(2)
plt.plot(m,fm1, m, k)
plt.grid()
plt.show()
```

Graphical Method

- `m=np.linspace(40, 200, 100)`
- 100개의 m(몸무게)에 대해 일일이 시각화를 해서 4초 때, 36 m/s를 지나는 지 확인하는 방법
- 140kg 조금 지나서 150kg 사이에서 4초일 때, 36m/s를 그래프가 지나가는 것을 보여 준다.
- **<u>데이터 시각화에 의해 근을 구하는 방법이다.</u>**
- x축은 40kg에서 200kg의 몸무게를 가진 사람을 나타낸다
- y축은 40kg에서 200kg의 몸무게를 가진 사람의 4초일 때의 체감 속도를 나타낸다.

Graphical Method



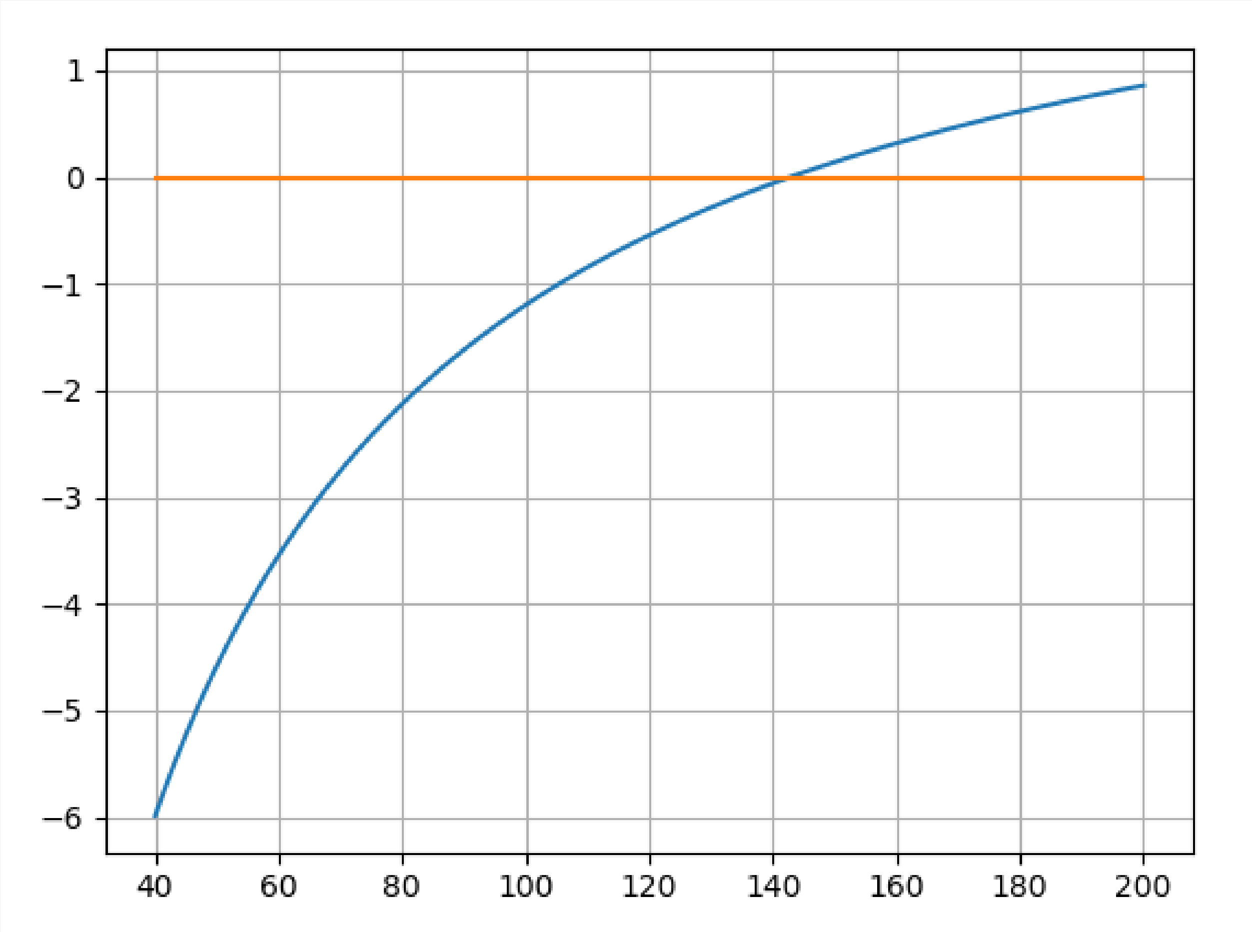
Graphical Method

- `fm1=np.sqrt(g*m/cd)*np.tanh(np.sqrt(g*cd/m)*t)-v`
- x축은 40kg에서 200kg의 몸무게를 가진 사람을 나타낸다
- y축은 40kg에서 200kg의 몸무게를 가진 사람의 4초일 때의 체감 속도에서 36m/s 뺀 값을 나타낸다.
- 이 그래프는 f(m)=0의 의미를 나타낸다.
- m이 어떤 값을 가질 때, f(m)은 0을 지나간다.
- 우리는 4초일 때 f(m) 그래프가 0을 지나가는 몸무게 m을 찾는 것이 목표이다.
- 140kg 조금 지나서 150kg 사이에서 4초일 때, 그래프가 0을 지나가는 것을 보여 준다.
- 데이터 시각화 방법으로 근은 140과 150 사이에 있다.

Root (근, 해)을 구하는 과정: 근 알고리즘

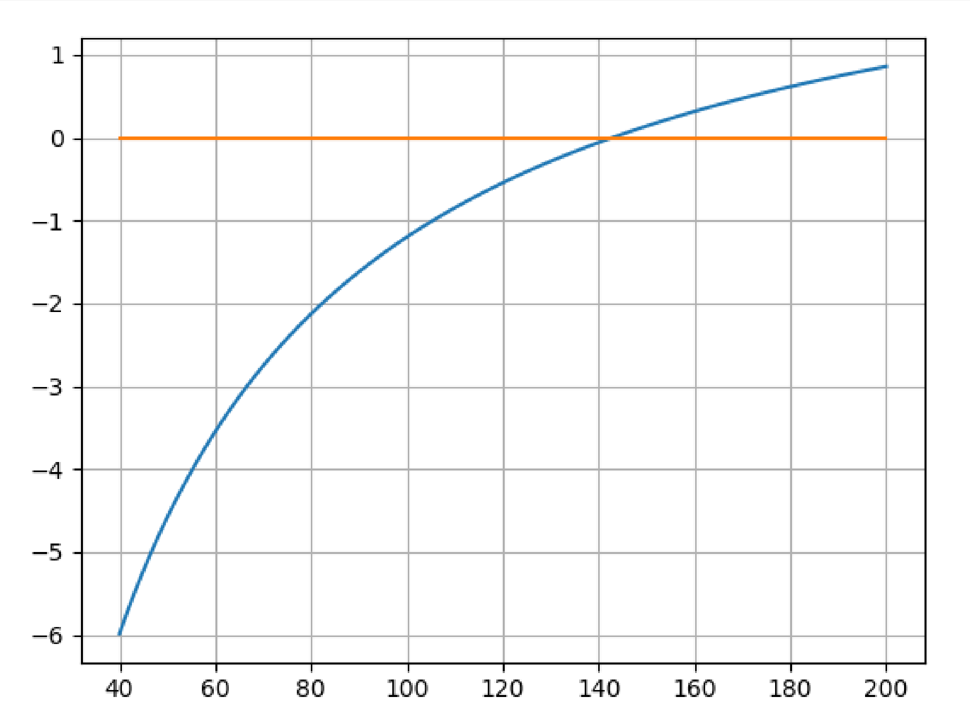
- 함수 f(x), f(m) 를 0으로 만드는 x, 또는 m을 찾아내는 과정
- 번지 점프의 체감 속도 **함수를 0으로 만드는 몸무게 m인 사람을 찾는 과정**
- 번지 점프의 체감 속도 함수에서 36을 뺀 **함수를 0으로 만드는 몸무게 m인 사람을 찾는 과정**
- **f(x)=0을 만드는 x를 찾는 과정**
- **수치해석에서는 f(x)=0을 만드는 정확한 x값를 찾는 것이 아니라, f(x)=0을 만드는 근사값 x를 찾는다**
- **이유는 실생활에서 만나는, 풀어야 할 f()함수가 수학적으로 유도하기에 어렵고, 또는 풀기에도 어렵기 때문이다.**
- **그러나 풀어야 하는 임무를 가지고 있기 때문에 근사 값을 구하기 위한 다양한 수치해석 알고리즘이 제안된다. (근사화 전략)**

Graphical Method



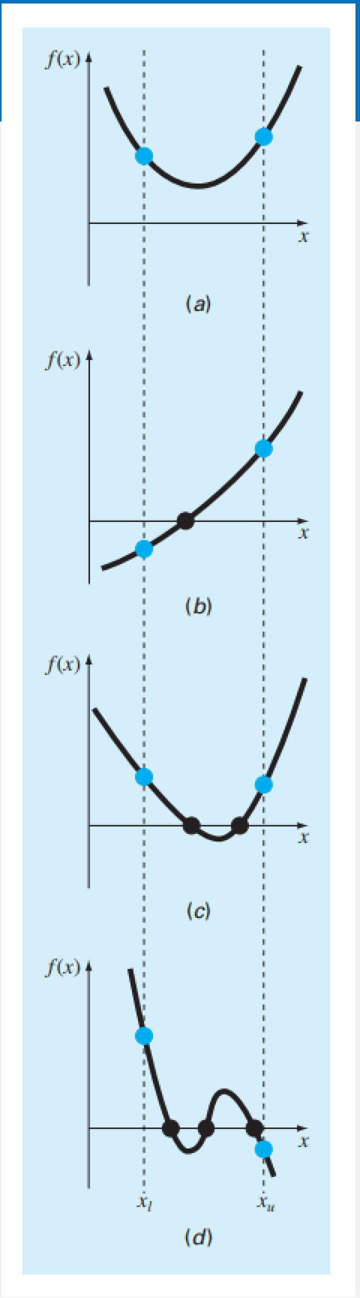
Graphical Method

- `k=np.linspace(0,0,100)`
- `plt.plot(m,fm1, m, k)`
- x축에 새로운 라인을 하난 더 그려 넣어보면, 그래프가 x축과 만나는 점에 대한 개념이 생긴다.



Graphical Method 고찰

- 장점
  - 비교적 쉬운 방법으로 근의 개략적인 위치를 알 수 있다.
  - 다른 수치해석 알고리즘이 시작할 수 있는 기본적인 근의 위치를 파악 할 수 있게 한다
- 단점
  - 근의 측정값이 정확하지 않다
- 특징
  - 근을 추측할 때 trial and error 방법을 사용한다.
  - $f(x_l)$과 $f(x_u)$의 개념, 구간법이 나타난다.
  - $f(x_l)$과 $f(x_u)$의 부호가 다르면, (b)와 (d) 처럼 근이 한 개 있거나, 홀수 개(3개)의 근이 있음을 파악할 수 있다.
  - $f(x_l)$ 과 $f(x_u)$의 부호가 같으면, (a)와 (c)처럼 근이 없거나, 짝수 개(2개)의 근이 있다.



수치 알고리즘의 **근사화 전략**

- 1. 연속 시스템(Continuous System)을 이산 시스템(Discrete System)으로 근사한다.
- 2. 적분을 합으로 대체한다
- 3. 도함수를 유한 차분(Infinite Difference)로 대체한다.
- 4. 비선형을 선형과 수정으로 대체한다.
- 5. 문제를 다른 문제로 변형한다.
- 6. trial and error의 반복을 통해서 참 값에 근사한다.
  - Graphical Method, Bracketing Method (구간법), Open Method (개방법)

비선형을 선형으로 대체한다. (예제)

- https://github.com/SCKIMOSU/Numerical-Analysis/blob/master/sin_x.py

- x가 0과 0.1사이의 값을 가질 때, f(x)=sin(x)의 값을 구한다
-
```
import numpy as np
points = [0.01*i for i in np.arange(1,11)]

for x in points:
    print (x, np.sin(x), '%.2f' % (abs(x-
np.sin(x))/np.sin(x)*100))
    # Formatting Numbers as Strings
    # print('The order total comes to %.2f' % 123.444)
    # The order total comes to 123.44
    # a = "abcdefghijklmnopqrstuvwxyz"
    # print('%.3s' % a)
    # abc
```

비선형을 선형으로 대체한다. (예제)

- 첫 번째 열은 x값, 두 번째 열은 sin(x), 세 번째 열은 x와 sin(x) 사이의 상대 차(백분율)이다.
- 이 차이는 언제나 20%보다 작다.
- 따라서 이 정도의 정밀도에 만족한다면, sin(x)를 x로 대체할 수 있다

0.01 0.009999833334166664 0.00
0.02 0.01999866669333308 0.01
0.03 0.02999550020249566 0.02
0.04 0.03998933418663416 0.03
0.05 0.04997916927067833 0.04
0.06 0.059964006479444595 0.06
0.07 0.06994284733753277 0.08
0.08 0.0799146939691727 0.11
0.09 0.08987854919801104 0.14
0.1 0.09983341664682815 0.17