2024학년도 2학기

문제해결프로그래밍 강의
11주차

조기필

2024.11.21

지난시간 복습

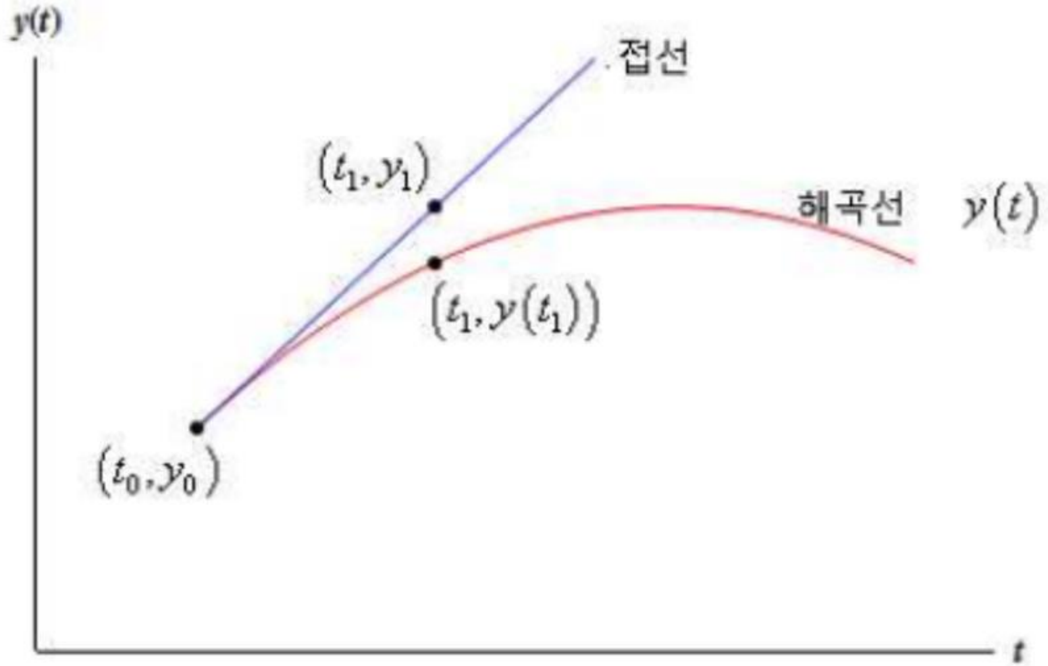
1. 미분방정식 수치해석 기법

문제 3 : COVID-19 수학 모델

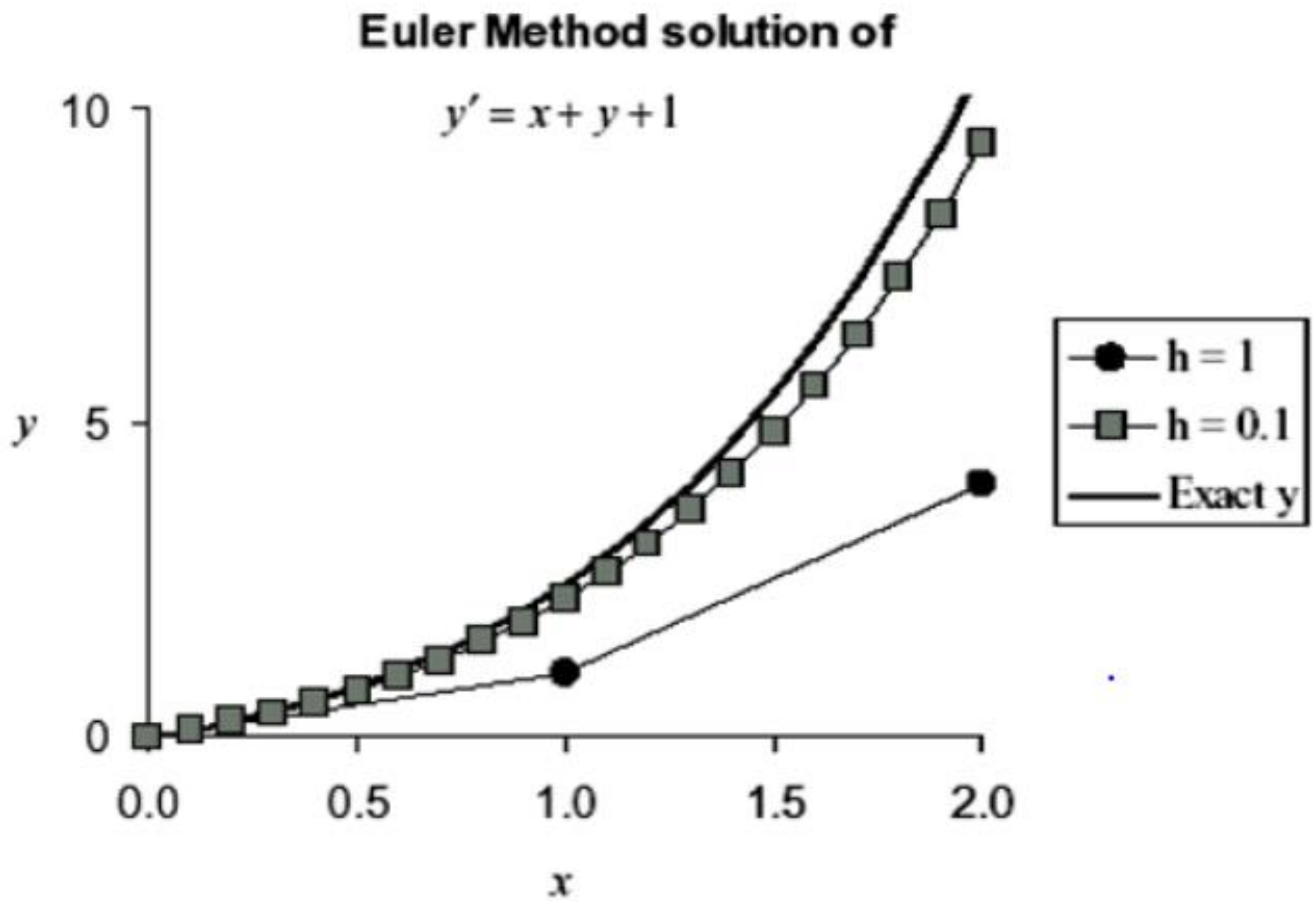
Euler 방법 : 등분된 x값으로부터 근사해를 얻는 방법

초기값 문제 $\begin{cases} y' = f(x, y) \\ y(x_0) = y_0 \end{cases}$

$$y_1 = y_0 + f(t_0, y_0)(t_1 - t_0)$$



문제 3 : COVID-19 수학 모델



문제 3 : COVID-19 수학 모델

오일러 방법 (Euler method) 계산

$$\frac{d}{dt}S(t) = -\beta S(t)\frac{I(t)}{N}$$

$$y_1 = y_0 + f(t_0, y_0)(t_1 - t_0)$$

$$\frac{d}{dt}I(t) = \beta S(t)\frac{I(t)}{N} - \gamma(t)I(t)$$

$$\frac{d}{dt}R(t) = \gamma(t)I(t)$$

$$S(0) = S_0,\ I(0) = I_0,\ R(0) = R_0,$$

```
S0 = Euler_y[0,0]
I0 = Euler_y[0,1]
R0 = Euler_y[0,2]
```

```
S1 = S0 + (-beta*S0*I0/N)*(time[1]-time[0])
I1 = I0 + (beta*S0*I0/N - gamma*I0)*(time[1]-time[0])
R1 = R0 + (gamma*I0)*(time[1]-time[0])

Euler_y[1,:] = np.array([S1, I1, R1])
```

문제 3 : COVID-19 수학 모델

테일러 급수 (Taylor series)

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots, \quad = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

모든 함수는 그 함수의 미분들의 한 점에서의 값으로 계산된 항의 무한합으로 나타낼 수 있다.

$x = t_{i+1}, a = t_i$,
$f(x) = f(t_{i+1}) = y_{i+1},\ f(a) = f(t_i) = y_i$, 이라고 하면

$$y_{i+1} = y_i + \frac{dy}{dt}(t_{i+1} - t_i) + \frac{1}{2!}\frac{d^2y}{dt^2}(t_{i+1} - t_i)^2$$

$$+ \frac{1}{3!}\frac{d^3y}{dt^3}(t_{i+1} - t_i)^3 + \frac{1}{4!}\frac{d^4y}{dt^4}(t_{i+1} - t_i)^4 + \cdots$$

문제 3 : COVID-19 수학 모델

만약 두 단위시간의 간격 $t_{i+1} - t_i$가 작은 값이라면 (예를들어 1보다 작은 0에 가까운 수)

$$y_{i+1} = y_i + \frac{dy}{dt}(t_{i+1} - t_i) + \frac{1}{2!}\frac{d^2y}{dt^2}(t_{i+1} - t_i)^2$$

$$+\frac{1}{3!}\frac{d^3y}{dt^3}(t_{i+1} - t_i)^3 + \frac{1}{4!}\frac{d^4y}{dt^4}(t_{i+1} - t_i)^4 + \cdots$$

점점 0에 가까워 진다.

오일러 방법
(Euler method)

$$y_{i+1} \simeq y_i + f(t_i, y_i)h + \frac{1}{2!}f'(t_i, y_i)h^2 + \frac{1}{3!}f''(t_i, y_i)h^3 + \frac{1}{4!}f'''(t_i, y_i)h^4$$

문제 3 : COVID-19 수학 모델

룽게-쿠타 방법 (Runge-Kutta method)

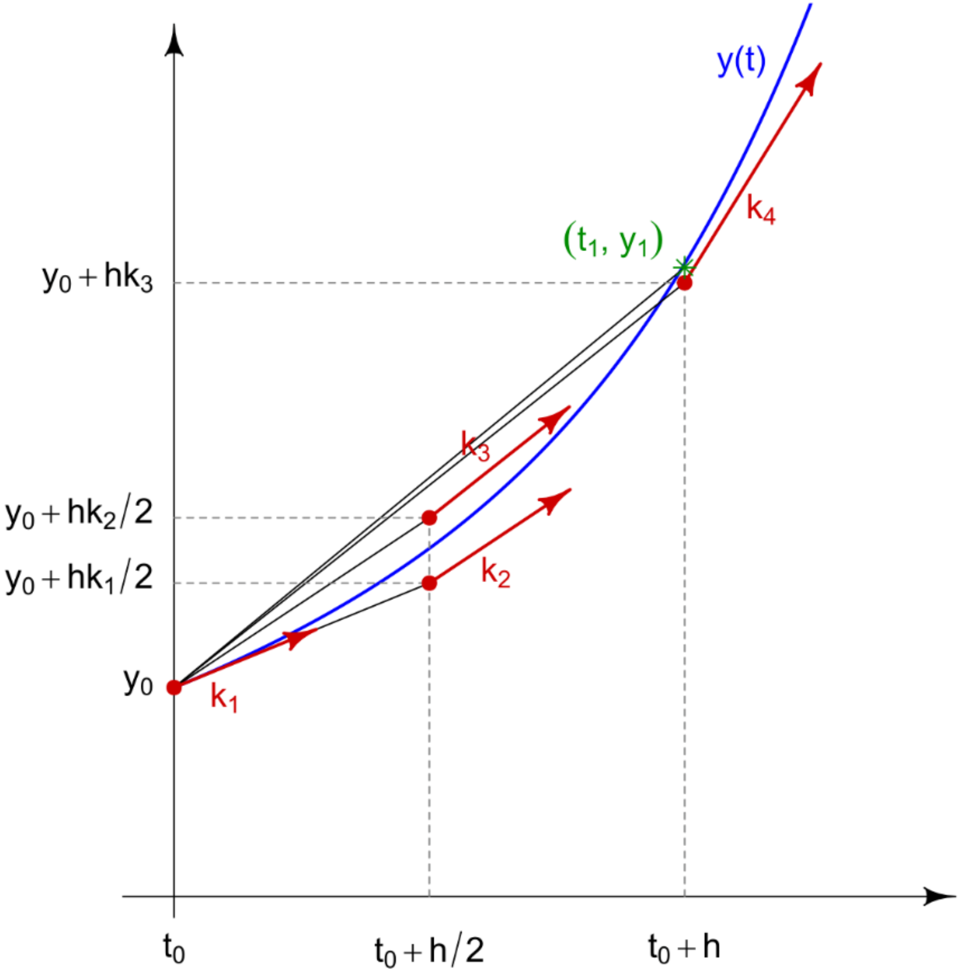
$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2 + a_3 k_3 + a_4 k_4)h$$

$$k_1 = f(t_i, y_i)$$

$$k_2 = f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}k_1 h)$$

$$k_3 = f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}k_2 h)$$

$$k_2 = f(t_i + h, y_i + k_3 h)$$



문제 3 : COVID-19 수학 모델

그럼 어떻게 룽게-쿠타 방법을 적용할까?

$$\frac{d}{dt}S(t) = -\beta S(t)\frac{I(t)}{N}$$

$$\frac{d}{dt}I(t) = \beta S(t)\frac{I(t)}{N} - \gamma I(t)$$

$$\frac{d}{dt}R(t) = \gamma I(t)$$

$S(0) = S_0,\ I(0) = I_0,\ R(0) = R_0,$

룽게-쿠타 방법 (Runge-Kutta method)

$$\frac{dy}{dt} = f(t, y), \qquad y(t_i) = y_i$$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

이때 여기서,

$$k_1 = f(t_i, y_i)$$

$$k_2 = f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}k_1 h)$$

$$k_3 = f(t_i + \tfrac{1}{2}h, y_i + \tfrac{1}{2}k_2 h)$$

$$= f(t_i + h, y_i + k_3 h)$$

문제 3 : COVID-19 수학 모델

기초감염재생산수(R_0, Basic reproduction number)

최초 감염자가 감염기간 동안에 감염시킬 수 있는 2차 감염자의 수

R_0 =	감염 확률	x	접촉빈도	x	전파기간
	transmissibility		number of contact per day		Duration of infectious period

감염 확산 방지 정책

문제 3 : COVID-19 수학 모델

단위 시간당 접촉

환자들의 감염 전파기간

$$\mathcal{R}_0 = \boldsymbol{\alpha} \times \boldsymbol{p} \times \frac{\mathbf{1}}{\boldsymbol{\gamma}} = \frac{\boldsymbol{\beta}}{\boldsymbol{\gamma}}$$

접촉 당 감염될 확률

문제 3 : COVID-19 수학 모델

질병이 사라지는 조건 ⟺ $R_0 < 1$

질병이 사라지지 않고 계속 남아 있을 조건 ⟺ $R_0 > 1$

오늘의 학습내용

1. 코로나-19 데이터를 이용한 $\boldsymbol{\beta}$ 추정하기

문제 3 : COVID-19 수학 모델

단위 시간당 접촉

환자들의 감염 전파기간

$$\mathcal{R}_0 = \boldsymbol{\alpha} \times \boldsymbol{p} \times \frac{\mathbf{1}}{\boldsymbol{\gamma}} \qquad = \frac{\boldsymbol{\beta}}{\boldsymbol{\gamma}}$$

접촉 당 감염될 확률

질병이 사라지는 조건 ⟺ $\boldsymbol{R_0} < 1$

γ 는 회복률은 질병자체의 고유한 회복기간이 필요하므로 인위적으로 낮추기 어려움

그래서, R_0 를 1보다 작게 만들기 위해 현재 β(감염상수)가 얼마인지를 알아야 한다.

문제 3 : COVID-19 수학 모델

그럼 $\boldsymbol{\beta}$ 를 어떻게 구해야 할까?

감염자 실데이터를 활용해서 날짜별 감염자 수에 가장 가깝게 시뮬레이션이 되는

$\boldsymbol{\beta}$를 찾으면 된다.

그럼 감염자 실데이터를 알아보자.

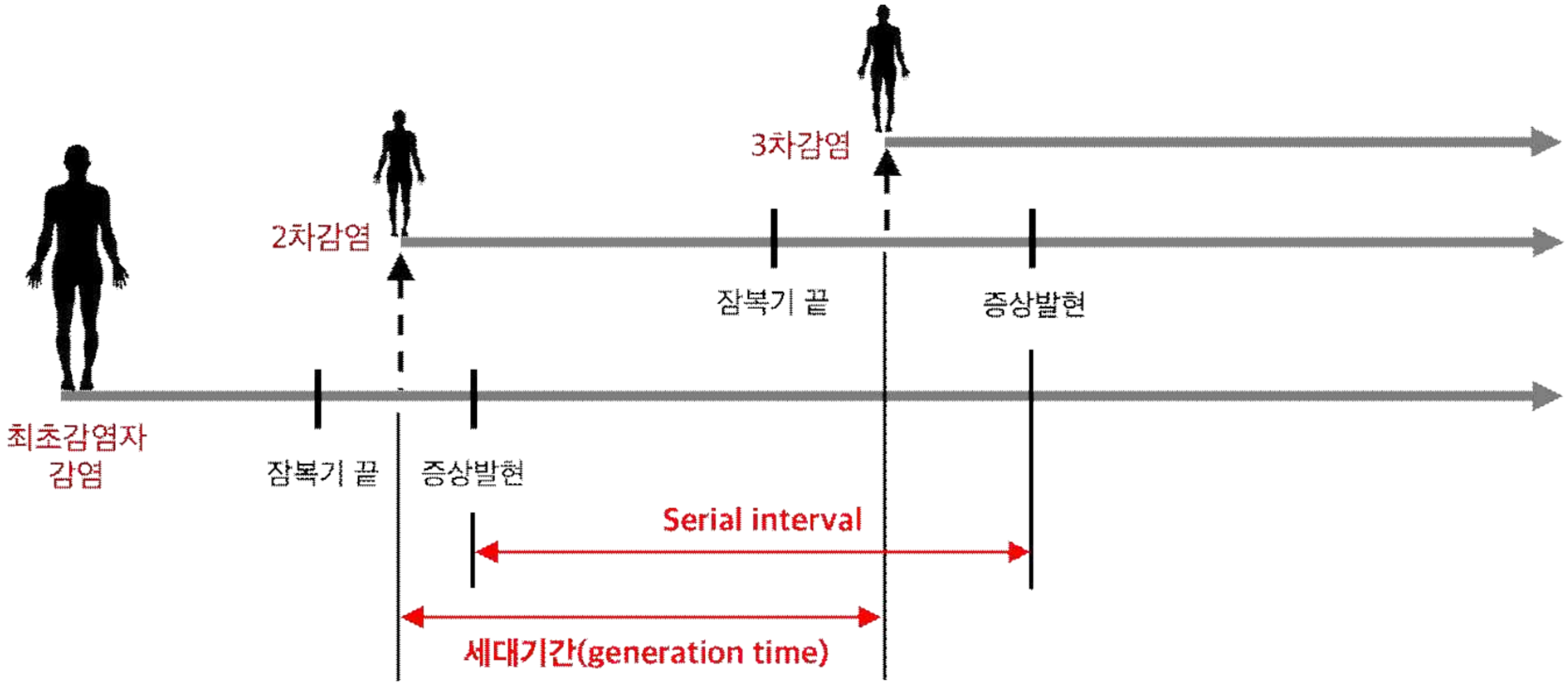
https://ncov.kdca.go.kr/bdBoardListR.do?brdId=1&brdGubun=11

문제 3 : COVID-19 수학 모델

COVID-19 감염자 데이터 (2020년 8월)

날짜	3일	4일	5일	6일	7일	8일	9일	10일	11일	12일	13일	14일	15일	16일
감염된 사람 수 (명)	3	13	13	23	9	30	30	17	23	35	47	85	154	267

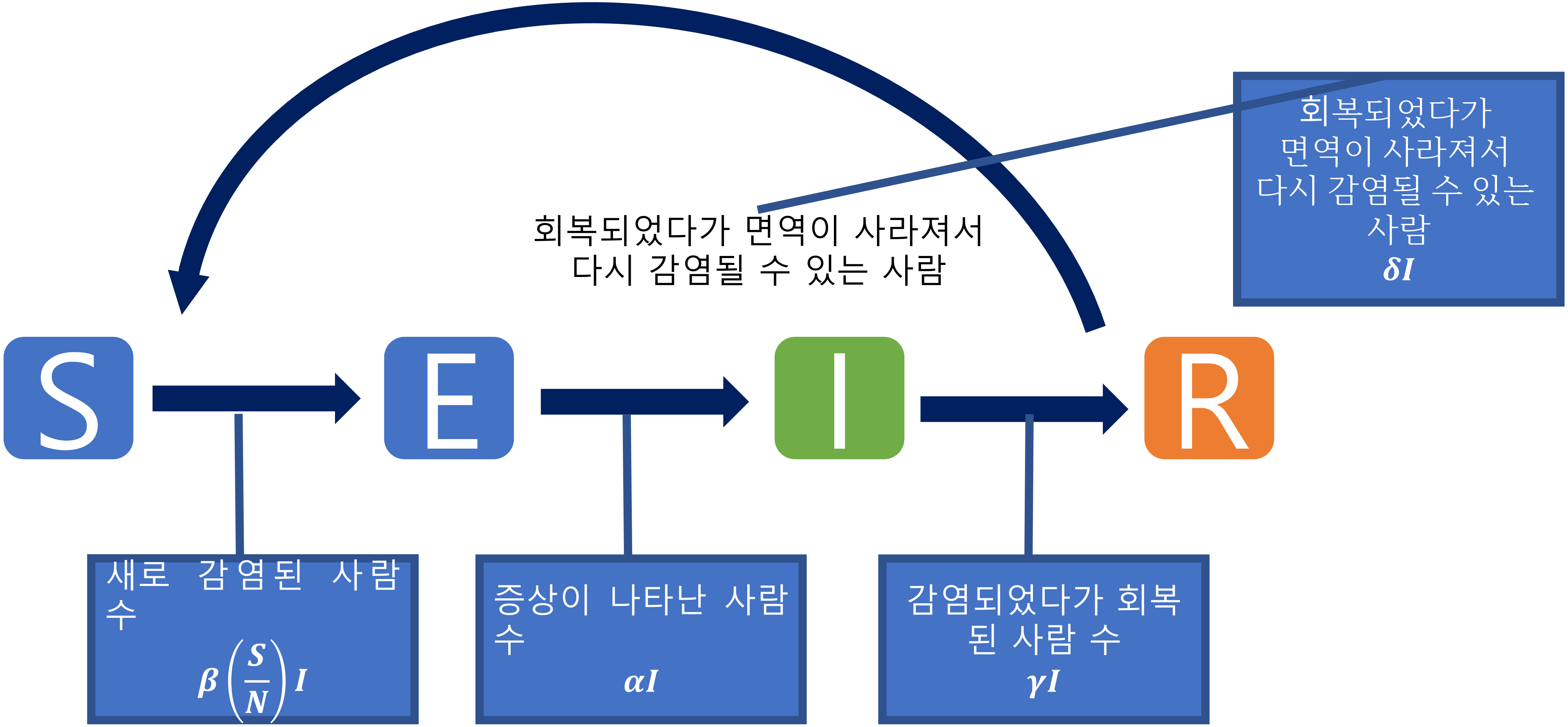
문제 3 : COVID-19 수학 모델



Q 코로나19 바이러스의 잠복기는 어떻게 되나요?

A ▫ 코로나19의 잠복기는 1~14일 (평균 5~7일)이며, 증상 발생 1~3일 전부터 호흡기 검체에서 바이러스가 검출됩니다.

문제 3 : COVID-19 수학 모델



문제 3 : COVID-19 수학 모델

SEIR 모델 (잠복기가 추가된 모델)

$$\frac{d}{dt}S(t) = -\beta S(t)\frac{I(t)}{N} + \delta R(t)$$

$$\frac{d}{dt}E(t) = \beta S(t)\frac{I(t)}{N} - \alpha\, E(t)$$

$$\frac{d}{dt}I(t) = \alpha\, E(t) - \gamma I(t)$$

$$\frac{d}{dt}R(t) = \gamma I(t) - \delta R(t)$$

문제 3 : COVID-19 수학 모델

기본 라이브러리와 COVID-19 신규 감염자 데이터를 불러오자.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
data = pd.read_excel('drive/MyDrive/Colab Notebooks/Problem_solver/COVID19_data.xlsx')
```

문제 3 : COVID-19 수학 모델

시간 변수들을 정의하자.

```
t0 = 0
tf = 13
n = 13
h = (tf-t0)/n
```

시작시간

끝시간

총 시간간격 개수

시간 간격

```
time = np.linspace(t0,tf,n+1)
```

수식안에 변수 정의하자.

```
N = 51840000
beta = 1.0
gamma = 1/14
delta = 1/229
alpha = 1/5
```

평균 잠복기를 5일로 가정하자.

문제 3 : COVID-19 수학 모델

8월 3일에 처음 3명의 감염자가 확인되었다고 가정하자. 그럼 초기 감염자수 (I=3)이고,

미분방정식에서 신규 감염자는 $\alpha\, E(t)$ 이므로 E=15로 가정할 수 있다.

```
I0 = data.loc[0,'Cases']
y = np.zeros((n+1,4))
initial_value = np.array([N-I0/alpha-I0, I0/alpha, I0, 0.0])
y[0,:] = initial_value
```

```
data
```

	date	Cases
0	2020-08-03	3
1	2020-08-04	13
2	2020-08-05	13

문제 3 : COVID-19 수학 모델

SEIR 모델 함수화

```python
def f(y_t):
  S, E, I, R = y_t
  dS = -beta*S*I/N + delta*R
  dE = beta*S*I/N - alpha*E
  dI = alpha*E - gamma*I
  dR = gamma*I - delta*R
  output = np.array([dS, dE, dI, dR])
  return output
```

SEIR 모델 (잠복기가 추가된 모델)

$$\frac{d}{dt}S(t) = -\beta S(t)\frac{I(t)}{N} + \delta R(t)$$

$$\frac{d}{dt}E(t) = \beta S(t)\frac{I(t)}{N} - \alpha E(t)$$

$$\frac{d}{dt}I(t) = \alpha E(t) - \gamma I(t)$$

$$\frac{d}{dt}R(t) = \gamma I(t) - \delta R(t)$$

문제 3 : COVID-19 수학 모델

룽게-쿠타 방법 (Runge-Kutta method) 함수화

```
def rk4(f, y_t, h):
  k1 = f(y_t)
  k2 = f(y_t+k1*h/2)
  k3 = f(y_t+k2*h/2)
  k4 = f(y_t+k3*h)
  y_t1 = y_t+h*(k1+2*k2+2*k3+k4)/6
  return y_t1
```

문제 3 : COVID-19 수학 모델

$\boldsymbol{\beta}$ 를 실제 신규 감염자수에 맞게 추정하기 위해서

모델의 신규감염자 수($\boldsymbol{\beta}$) ≈ 실제 신규 감염자수

```
pred_cases = np.zeros(len(time))
```

모델의 신규 감염자 수를 정의하자.

```
pred_cases[0] = 10
```

모델의 첫 시간의 초기 감염자 수가 10이므로
초기값을 10으로 지정하자.

문제 3 : COVID-19 수학 모델

```
for i in range(n):
  y[i+1,:] = rk4(f, y[i,:], h)
  pred_cases[i+1] = alpha*y[i,1]
```

모델의 신규 감염자 수는 $\alpha E(t)$ 로 나타낼 수 있다.

SEIR 모델 (잠복기가 추가된 모델)

$$\frac{d}{dt}S(t) = -\beta S(t)\frac{I(t)}{N} + \delta R(t)$$

$$\frac{d}{dt}E(t) = \beta S(t)\frac{I(t)}{N} - \alpha E(t)$$

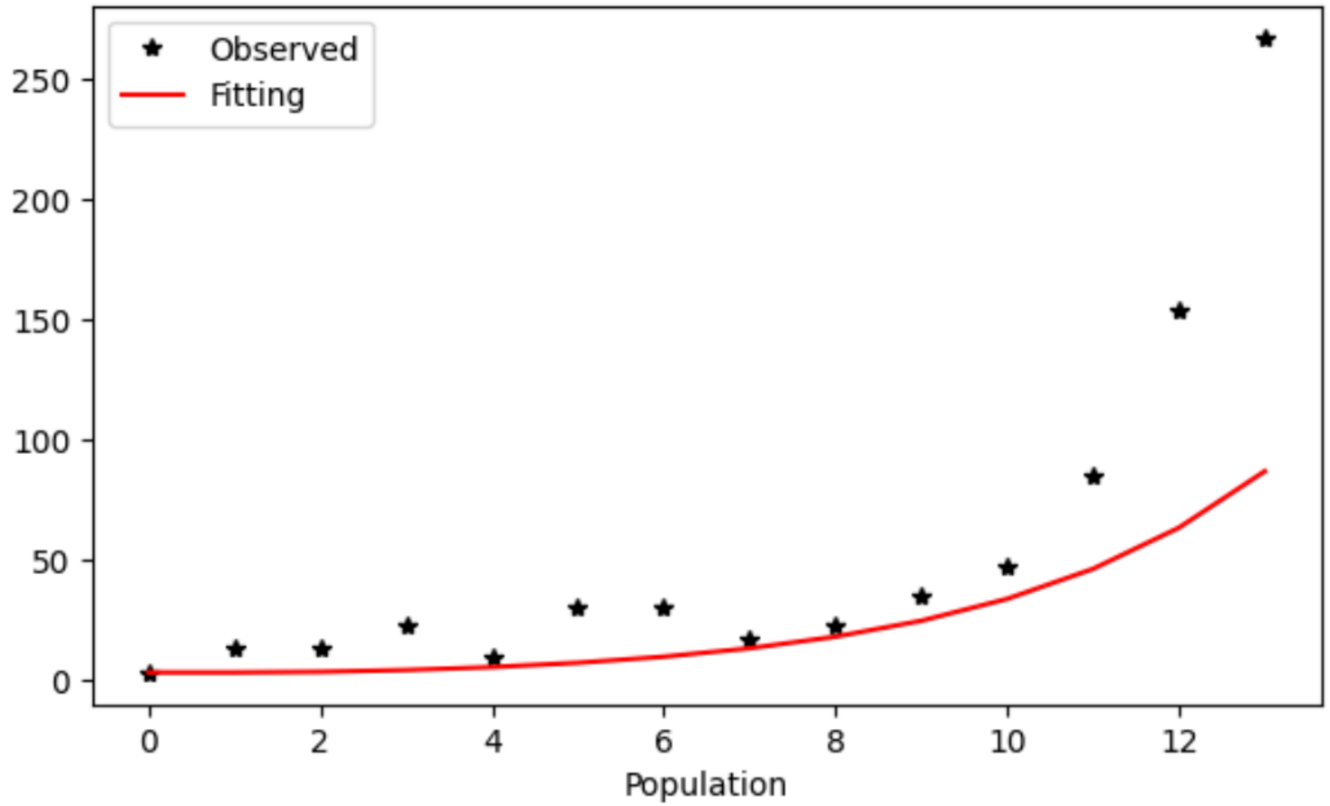
$$\frac{d}{dt}I(t) = \alpha E(t) - \gamma I(t)$$

$$\frac{d}{dt}R(t) = \gamma I(t) - \delta R(t)$$

문제 3 : COVID-19 수학 모델

모델의 신규감염자 수와 실제 신규 감염자수를 비교해보자.

```
plt.figure(figsize=(7,4))
plt.plot(time,data['Cases'],'*k')
plt.plot(time,pred_cases,'r')
plt.xlabel('time')
plt.xlabel('Population')
plt.legend(('Observed','Fitting'),loc='best')
plt.show()
```



문제 3 : COVID-19 수학 모델

모델의 신규감염자 수와 실제 신규 감염자수가 잘 맞는가?

아니라면 각자 데이터에 잘 맞게 $\boldsymbol{\beta}$ 의 값을 바꾸어가며 맞추어보자.

문제 3 : COVID-19 수학 모델

그럼 어떻게 데이터에 가장 알맞은 $\boldsymbol{\beta}$ 를 추정할 수 있을까?

↓

데이터의 신규 감염자수와 모델의 감염자 수의 차이를 최소화

↓

최소값을 찾는 문제 (최적화 문제)

문제 3 : COVID-19 수학 모델

최적화 문제의 수학적 표현

$$\min \quad f(\mathrm{X})$$

$$\textit{subject to} \quad g_i(x) \leq 0, \quad i = 1, 2, \dots, m$$

$$h_j(x) = 0, \quad j = 1, 2, \dots, p$$

$$x_l \leq x \leq x_u$$

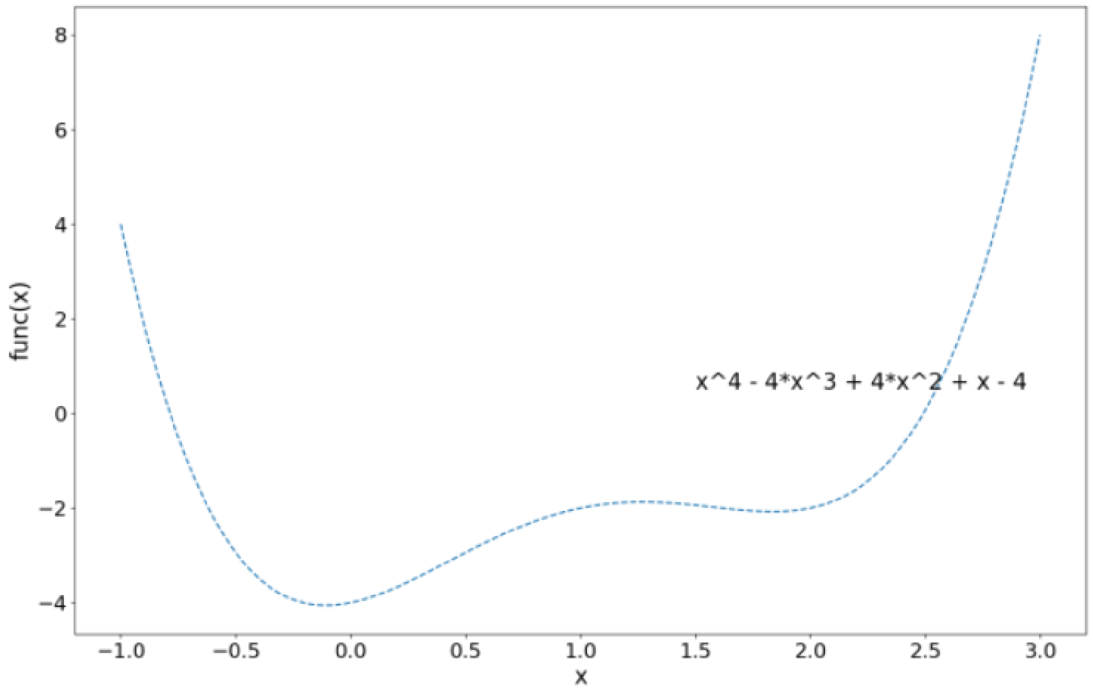
$$\textit{where} \quad \mathrm{X} = \left[x_1, x_2, x_3, \dots, x_n\right]^T$$

문제 3 : COVID-19 수학 모델

최적화 문제

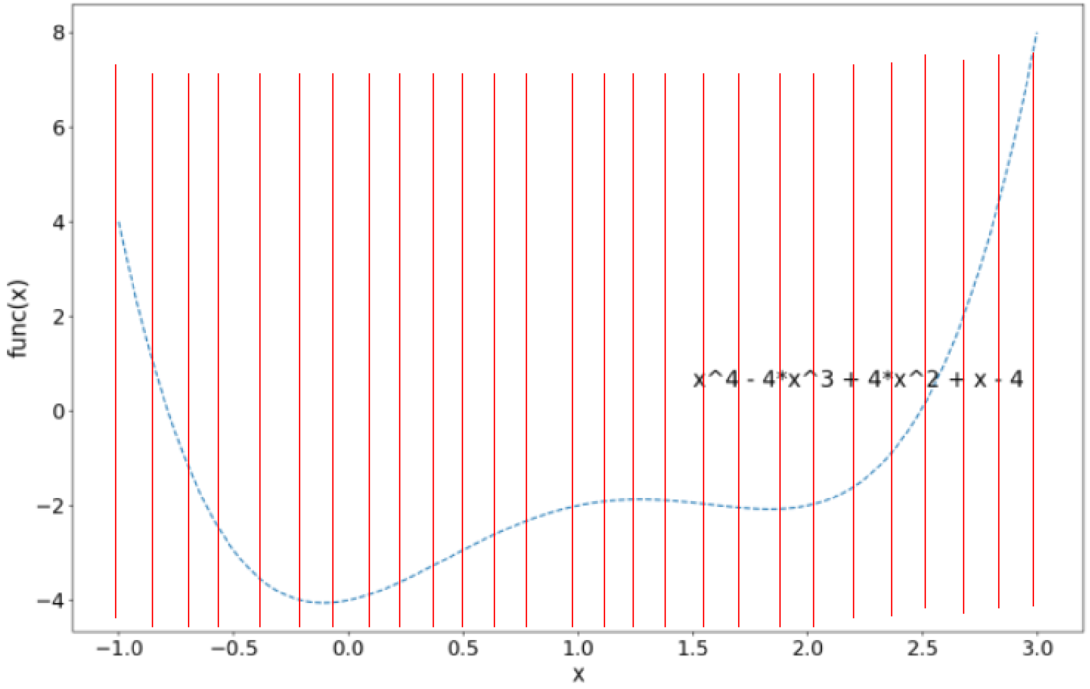
수학적으로 해결하기 어렵지만 컴퓨터로 해결하는 것 어렵지 않다.

예를 들어,

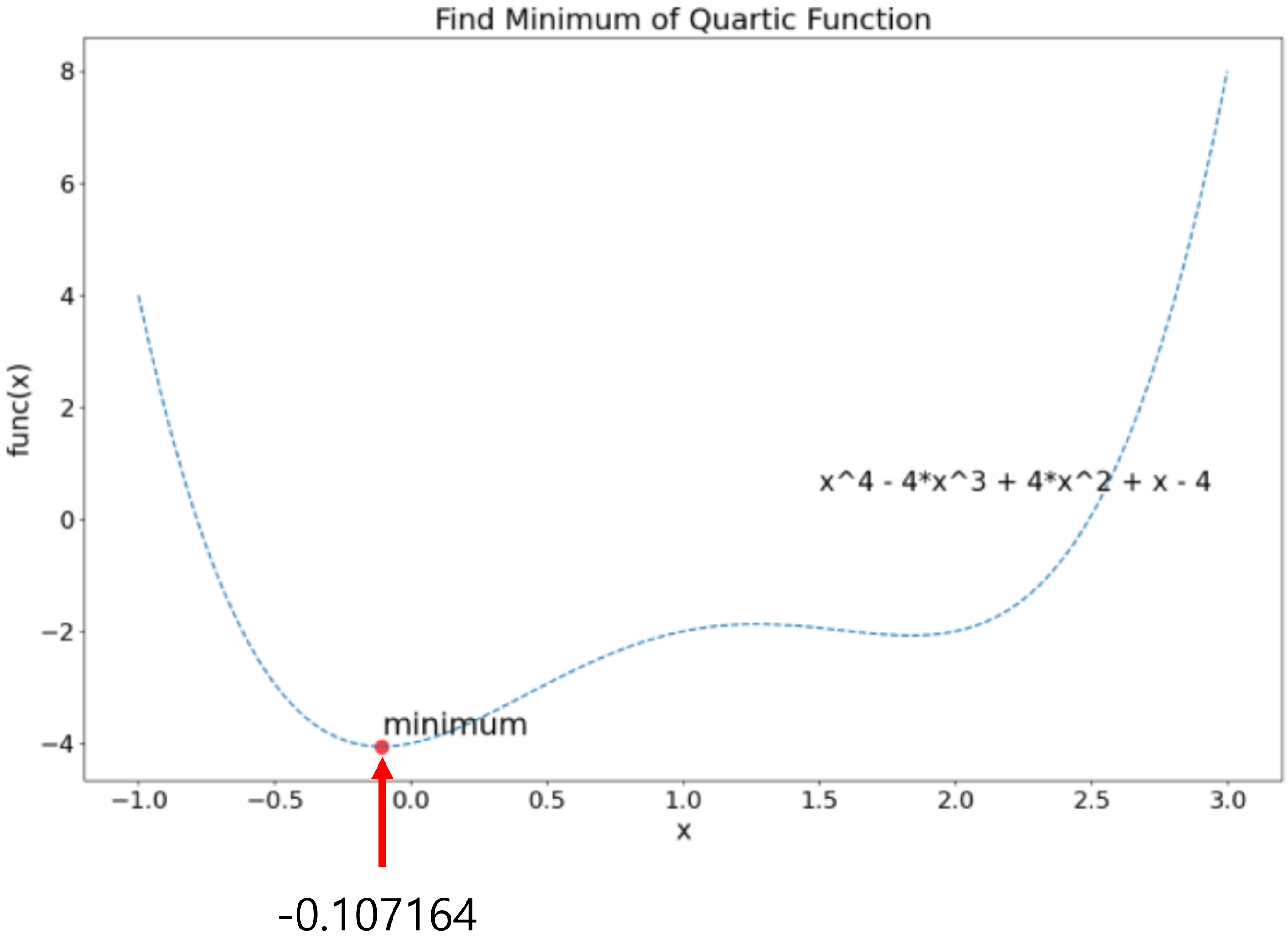


문제 3 : COVID-19 수학 모델

촘촘하게 모든 격자의 최적화할 함수 값을 확인



문제 3 : COVID-19 수학 모델



X의 간격을 **0.000001**으로 목적 함수를 **4백만번** 계산하면 최소값을 쉽게 확인할 수 있다.

문제 3 : COVID-19 수학 모델

시간간격을 아주 작게해서 모든 값을 계산하면 최적화 문제 해결 하지만....

1. 변수가 아주 많다면? 예를 들어 10개
2. 시간이 아주 길다면? 예를 들어 1000(단위)시간

계산 횟수 = (단위시간간격개수*시간)^(변수개수)

$$= (1{,}000{,}000 \times 1000)^{10} = 10^{90}$$

문제 3 : COVID-19 수학 모델

2022년 110경

초당 연산 1,000,000,000,000,000,000…엑사급 슈퍼컴 시대 개막

곽노필 기자 +구독

등록 :2022-06-03 10:14
수정 :2022-06-03 11:41

가+

미 프런티어, 1초에 110경2천조번…최강 슈퍼컴 등극
25년만에 1조에서 100경까지…100만배 성능 향상
기후모델·신에너지·약물 개발 등에서 큰 역할 기대
한국은 500위 안에 6대…초당 2.5경번 연산이 최고

미국 테네시주 오크리지국립연구소에 설치된 세계 최강 슈퍼컴퓨터 '프런티어'. 오크리지국립연구소 제공

1초에 100경번.

초당 연산 횟수가 엑사(100경=1,000,000,000,000,000,000=10^18)급에 이르는 슈퍼컴퓨터 시대가 개막됐다.

미래&과학 많이 보는 기사

1. 화성의 삼각주서 생명체

2023년 119경

슈퍼컴퓨터 연산속도 랭킹

순위	명칭	개발국	처리속도(회)
1	프론티어	미국	119.4경
2	후가쿠	일본	44.2경
3	루미	핀란드	30.9경
4	네오날도	이탈리아	23.8경
5	서밋	미국	14.8경
6	시에라	미국	9.4경
7	선웨이타이후라이트	중국	9.3경
8	팔무타	미국	7.0경
9	세레네	미국	6.3경
10	텐허 2A	중국	6.1경

㈜ 처리속도는 초당

$$\frac{10^{90}}{10^{18}} = 10^{72}\text{초}$$

$$\approx 3 \times 10^{64}\text{년}$$

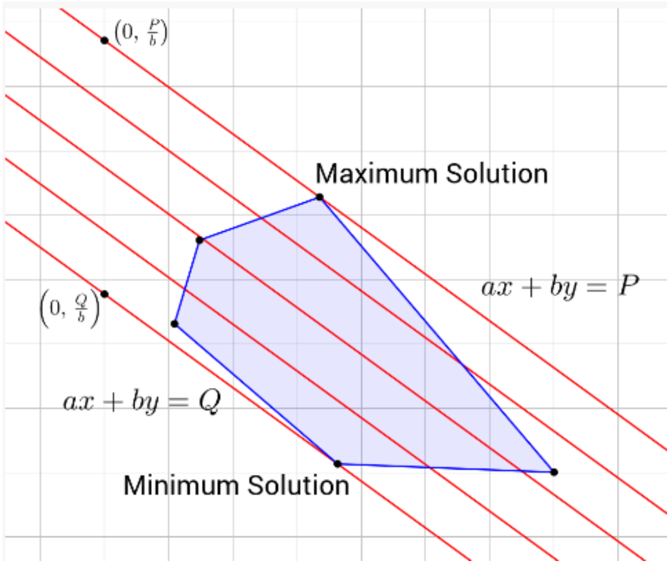
문제 3 : COVID-19 수학 모델

수치적으로 최적화란?

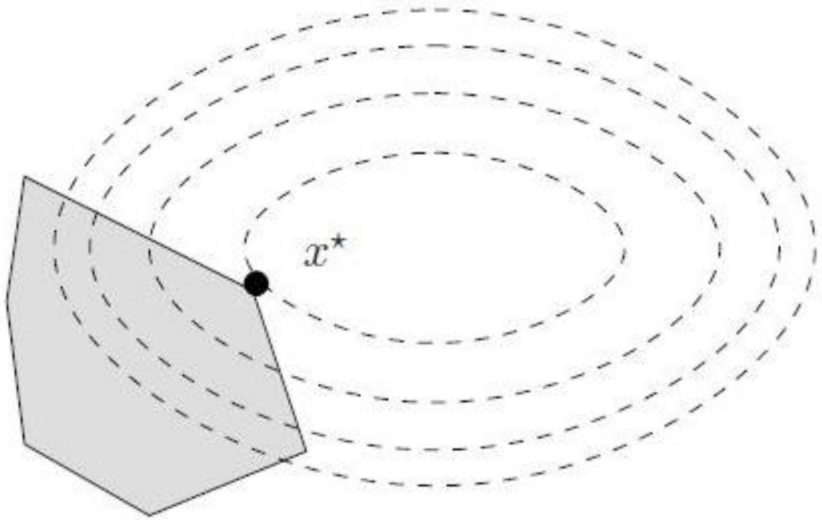
가능한 적은 반복 계산으로 최적화 (최소 또는 최대) 값을 찾는 방법

문제 3 : COVID-19 수학 모델

수치적으로 최적화의 기본 아이디어



선형계획법



2차계획법

문제 3 : COVID-19 수학 모델

파이썬의 scipy.optimize 패키지

Types of optimization problem		Function	Method
Local Opt. (국소)	unconstrained (비제약)	minimize	Nelder–Mead BFGS Newton–CG trust–ncg trust–krylov trust–exact
	constrained (제약)		trust–constr SLSQP
Global Opt. (전역)	derivative–based (미분 적용)	basinhopping shgo	–
	metahuristic	brute(?) differential_evolution dual_annealing	–

문제 3 : COVID-19 수학 모델

minimize 함수 이용한 최적화 시뮬레이션

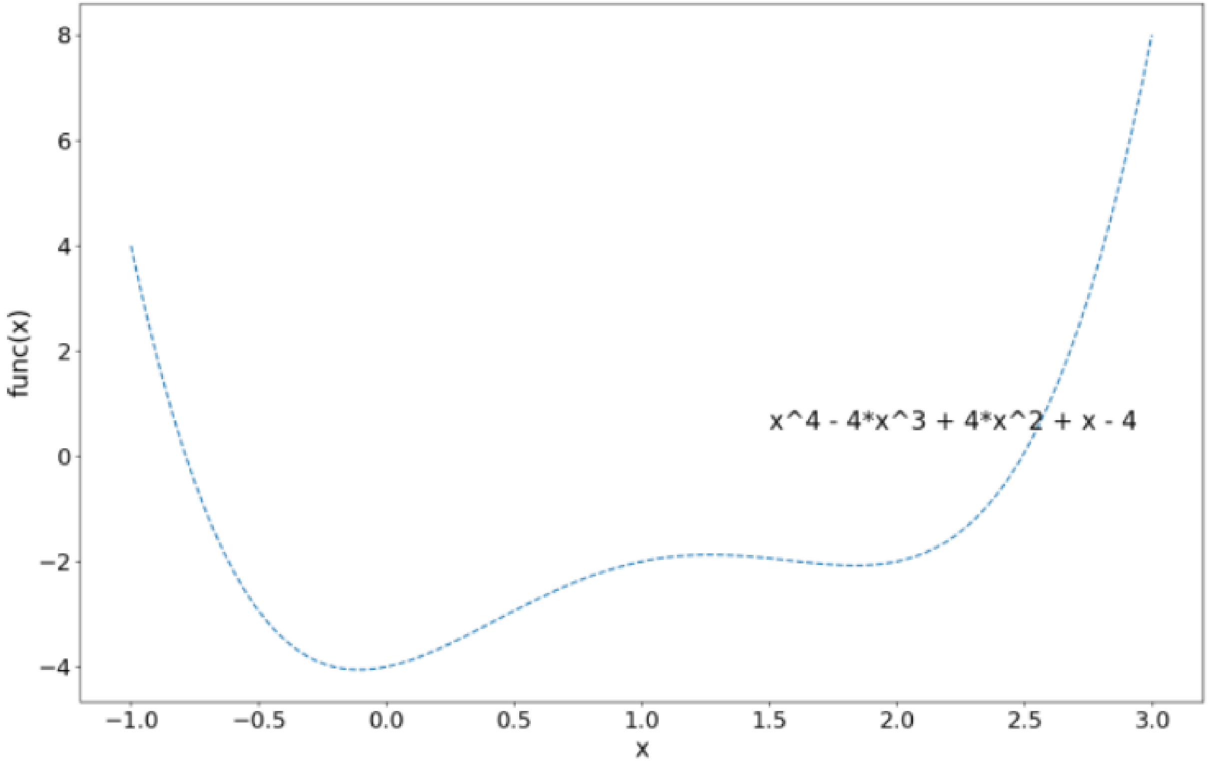
문제 3 : COVID-19 수학 모델

Minimize 함수 사용 방법

```
scipy.optimize.minimize(
        fun, → 목적함수
        x0, → 초깃값
핵심    args=( ), → 초깃값 외에 목적함수에 전달할 매개변수
        method=None, → 최적화 해 찾기 종류
        jac=None, hess=None, hessp=None,
        bounds=None, → 경계값
보조    constraints=( ), → 제약조건
        tol=None, callback=None, options=None )
```

문제 3 : COVID-19 수학 모델

예제 : 목적함수가 다음과 같은 4차 함수라고 하자.



이 목적함수의 최소값을 구해보자.

문제 3 : COVID-19 수학 모델

```
from scipy.optimize import minimize
```

minimize 라이브러리

목적함수를 정의하자.

목적함수 $f(x) = x^4 - 4 \times x^3 + 4 \times x^2 + x - 4$

$\Downarrow$

```
#목적함수
def obj_fun(x):
  return x**4 - 4*x**3 + 4*x**2 + x - 4
```

문제 3 : COVID-19 수학 모델

목적함수의 매개변수인 초기값을 정의하자.

만약 목적함수의 정보가 있다면, 최대한 원하는 값에 가까운

즉, 최소가 되게 하는 x에 가까운 값을 초기값으로 정의하는 것이 좋다.

아무 정보가 없다면, 조건에 위배되지 않는 초기값을 아무값이나 주자.

```
#초기값
x0 = 0.5
```

3. 추정하는 방법을 선택하자.

여기서 우리는 'SLSQP' 방법으로 실습할 것이다.

문제 3 : COVID-19 수학 모델

Minimize 함수 사용 방법

```
scipy.optimize.minimize(
    fun, → 목적함수
    x0, → 초깃값
    args=( ), → 초깃값 외에 목적함수에 전달할 매개변수
    method=None, → 최적화 해 찾기 종류
    jac=None, hess=None, hessp=None,
    bounds=None, → 경계값
    constraints=( ), → 제약조건
    tol=None, callback=None, options=None )
```

핵심: fun, x0, args, method

보조: bounds, constraints

```
sol = minimize(obj_fun, x0, method='SLSQP')
```

문제 3 : COVID-19 수학 모델

```
sol
```

```
message: Optimization terminated successfully
success: True
 status: 0
    fun: -4.056172844149885
      x: [-1.071e-01]
    nit: 6
    jac: [ 9.382e-04]
   nfev: 14
   njev: 6
```

x: 최적화 해

success: 최적화에 성공하면 True 반환

status: 종료 상태. 최적화에 성공하면 0 반환

message: 메시지 문자열

fun: x 위치에서의 함수의 값

jac: x 위치에서의 자코비안(그레디언트) 벡터의 값

hess_inv: x 위치에서의 헤시안 행렬의 역행렬의 값

nfev: 목적함수 호출 횟수

njev: 자코비안 계산 횟수

nhev: 헤시안 계산 횟수

nit: x 이동 횟수

문제 3 : COVID-19 수학 모델

```
sol.x
```

```
array([-0.10707227])
```

점뒤에 변수를 호출하는 것으로 원하는 결과만 프린트할 수 있다

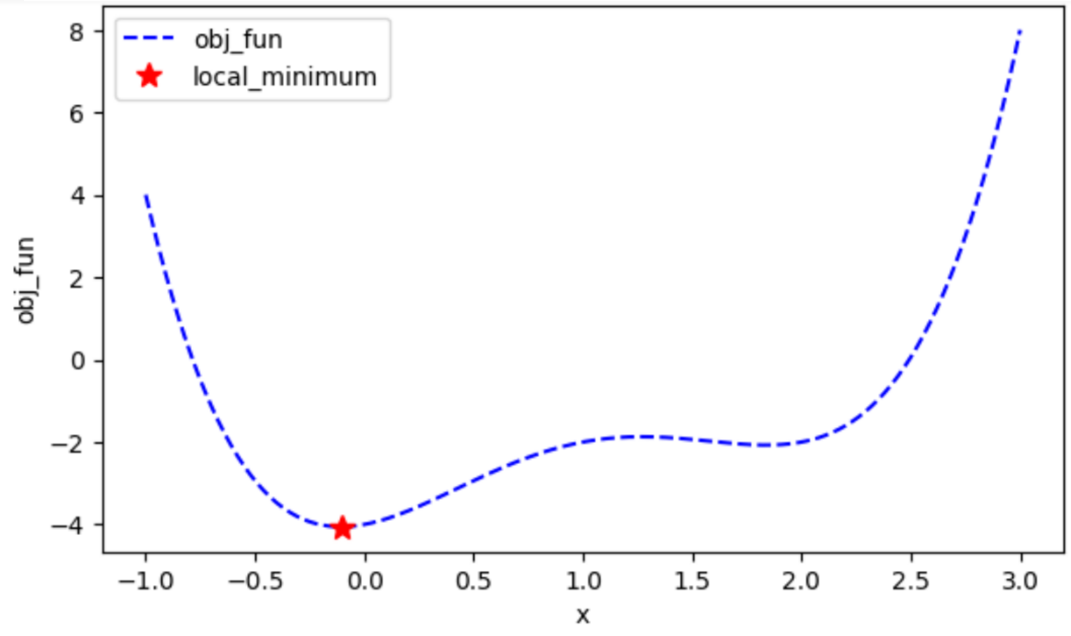
```
message: Optimization terminated successfully
success: True
 status: 0
    fun: -4.056172844149885
      x: [-1.071e-01]
    nit: 6
    jac: [ 9.382e-04]
   nfev: 14
   njev: 6
```

문제 3 : COVID-19 수학 모델

진짜로 최소값이 나왔는지 그림으로 확인해보자.

```
plt.figure(figsize=(7,4))
x1 = np.linspace(-1,3,401)
x2 = sol.x
y1 = obj_fun(x1)
y2 = sol.fun
plt.plot(x1, y1, '--b', x2, y2, '*r', markersize=10)
plt.xlabel('x')
plt.ylabel('obj_fun')
plt.legend(('obj_fun', 'local_minimum'), loc='best')
plt.show()
```

- `x1 = np.linspace(-1,3,401)` ← 예제 그림처럼 -1~3까지 그리기 위해 x격자 설정
- `x2 = sol.x` ← 최소값 결과는 점을 찍을 것이므로 결과에 해당하는 sol.x 를 x값으로
- `y1 = obj_fun(x1)` ← 목적함수의 그림을 그리기 위해 목적함수에 -1~3 격자 입력
- `y2 = sol.fun` ← 최소값의 결과인 fun변수 호출



문제 3 : COVID-19 수학 모델

Minimize함수에서 범위는 튜플형으로 값을 입력해야한다.

튜플(tuple)은 몇 가지 점을 제외하곤 리스트와 거의 비슷하며 리스트와 다른 점은 다음과 같다.

- 리스트는 [ ]으로 둘러싸지만 튜플은 ( )으로 둘러싼다.
- 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 그 값을 바꿀 수 없다.

튜플의 모습은 다음과 같다.

```
>>> t1 = ()
>>> t2 = (1,)
>>> t3 = (1, 2, 3)
>>> t4 = 1, 2, 3
>>> t5 = ('a', 'b', ('ab', 'cd'))
```

리스트와 모습은 거의 비슷하지만 튜플에서는 리스트와 다른 2가지 차이점을 찾아볼 수 있다. t2 = (1,)처럼 단지 1개의 요소만을 가질 때는 요소 뒤에 콤마(,)를 반드시 붙여야 한다는 것과 t4 = 1, 2, 3처럼 괄호( )를 생략해도 무방하다는 점이다.

문제 3 : COVID-19 수학 모델

```
bound = (0,3)
sol_2 = minimize(obj_fun, x0, method='SLSQP', bounds=(bound,))
```

범위가 하나지만 쉼표를 붙이고
괄호를 닫는다.

문제 3 : COVID-19 수학 모델

```
sol_2
```

```
 message: Optimization terminated successfully
 success: True
  status: 0
     fun: -4.0
       x: [ 0.000e+00]
     nit: 2
     jac: [ 1.000e+00]
    nfev: 4
    njev: 2
```

x: array([-0.10707227])

기존의 최소값을 만드는 x가 0으로 변했다.

문제 3 : COVID-19 수학 모델

항상 우리가 원하는 최소화 점을 찾지 못할 수 있다.

x0=1.5일때,

```
x0 = 1.5
```

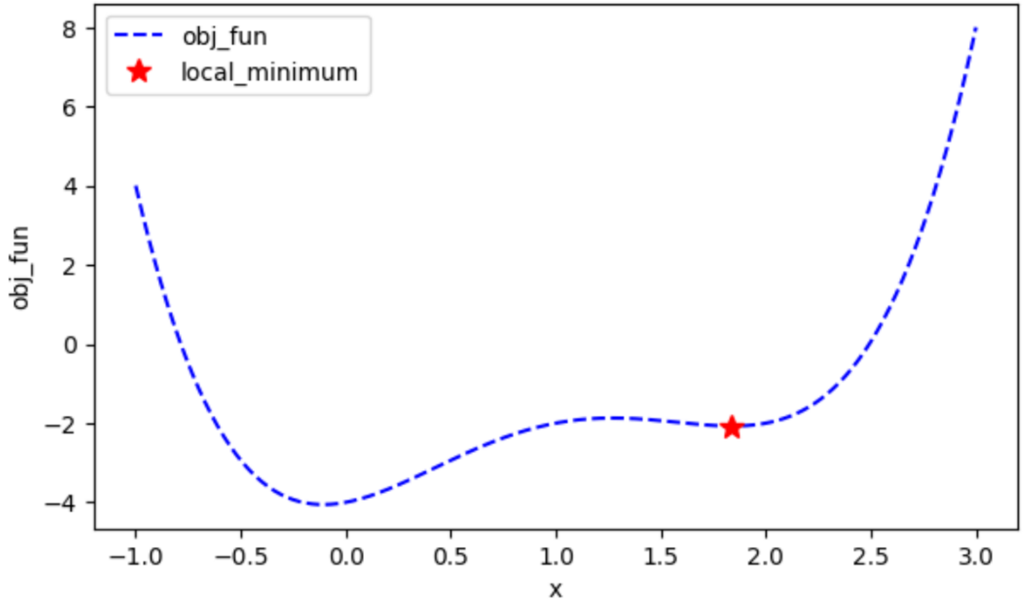
```
sol = minimize(obj_fun, x0, method='SLSQP')
```

를 실행시켜보자. 결과가 같은가? → NO!!

문제 3 : COVID-19 수학 모델

```
sol
```

```
 message: Optimization terminated successfully
 success: True
  status: 0
     fun: -2.073341682590727
       x: [ 1.838e+00]
     nit: 5
     jac: [ 9.373e-04]
    nfev: 11
    njev: 5
```



결과가 같은가? → NO!!

문제 3 : COVID-19 수학 모델

한번에 global한 최소값을 찾는 방법은?

```
scipy.optimize.basinhopping(func, x0, niter=100, T=1.0, stepsize=0.5,
minimizer_kwargs=None, take_step=None, accept_test=None, callback=None, interval=50,
disp=False, niter_success=None, seed=None, *, target_accept_rate=0.5, stepwise_factor=0.9)
```

문제 3 : COVID-19 수학 모델

minimize와 basinhopping의 결과를 비교해보자.

```
from scipy.optimize import basinhopping
```

```
sol_2 = basinhopping(obj_fun, x0)
```

```
sol_2
```

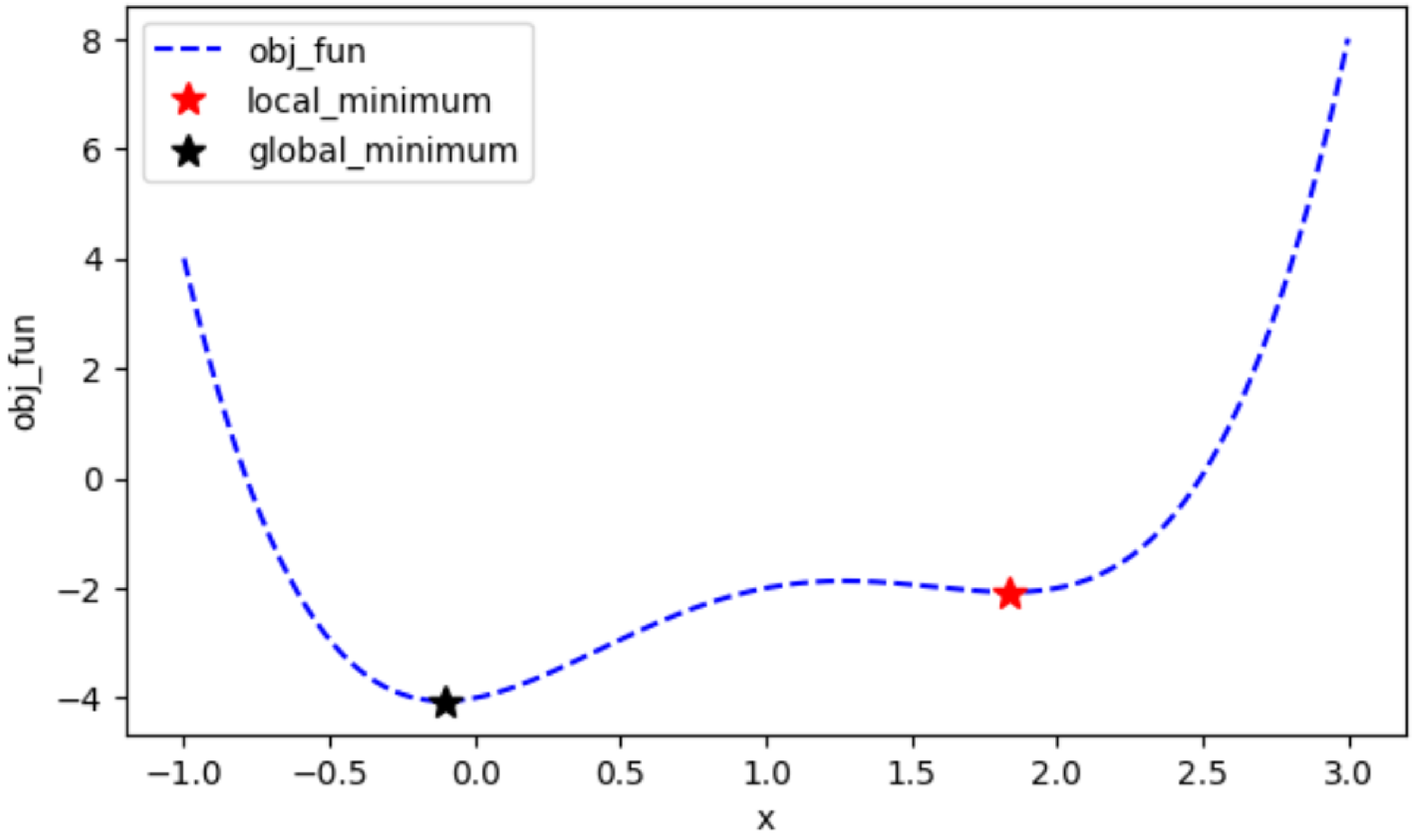
```
                       message: ['requested number of basinhopping iterations completed successfully']
                       success: True
                           fun: -4.056172885244464
                             x: [-1.072e-01]
                           nit: 100
         minimization_failures: 0
                          nfev: 1336
                          njev: 668
  lowest_optimization_result:  message: Optimization terminated successfully.
                               success: True
                                status: 0
                                   fun: -4.056172885244464
                                     x: [-1.072e-01]
                                   nit: 5
                                   jac: [ 0.000e+00]
                              hess_inv: [[ 9.339e-02]]
                                  nfev: 14
                                  njev: 7
```

문제 3 : COVID-19 수학 모델

```
plt.figure(figsize=(7,4))
x1 = np.linspace(-1,3,401)
x2 = sol.x
x3 = sol_2.x
y1 = obj_fun(x1)
y2 = sol.fun
y3 = sol_2.fun
plt.plot(x1, y1, '--b', x2, y2, '*r', markersize=10)
plt.plot(x3, y3, '*k', markersize=10)
plt.xlabel('x')
plt.ylabel('obj_fun')
plt.legend(('obj_fun', 'local_minimum', 'global_minimum'), loc='best')
plt.show()
```

결과 비교 위해 두 최소화값 점찍기

문제 3 : COVID-19 수학 모델



최적화를 이용한 beta 추정 시뮬레이션

문제 3 : COVID-19 수학 모델

그럼 어떻게 데이터에 가장 알맞은 $\boldsymbol{\beta}$ 를 추정할 수 있을까?

⬇

데이터의 신규 감염자수와 모델의 감염자 수의 차이를 최소화

⬇

최소값을 찾는 문제 (최적화 문제)

문제 3 : COVID-19 수학 모델

데이터의 신규 감염자수와 모델의 감염자 수의 차이를 최소화

↓

데이터의 신규 감염자수와 모델의 감염자 수의 RMSE를 최소화

↓

RMSE를 함수화하여 Minimize 사용

문제 3 : COVID-19 수학 모델

가장 알맞은 β를 추정해야 함으로 RMSE를 β 에 대한 함수로 나타내자.

1. 모델을 β 에 대한 함수로 변경하자.

```
def f(y_t):
  S, E, I, R = y_t
  dS = -beta*S*I/N + delta*R
  dE = beta*S*I/N - alpha*E
  dI = alpha*E - gamma*I
  dR = gamma*I - delta*R
  output = np.array([dS, dE, dI, dR])
  return output
```

```
def f_beta(y_t, x):
  S, E, I, R = y_t
  dS = (-x*S*I/N + delta*R).item()
  dE = (x*S*I/N - alpha*E).item()
  dI = (alpha*E - gamma*I).item()
  dR = (gamma*I - delta*R).item()
  return np.array([dS, dE, dI, dR])
```

문제 3 : COVID-19 수학 모델

2. rk4를 $\boldsymbol{\beta}$ 에 대한 함수로 변경하자.

```
def rk4(f, y_t, h):
  k1 = f(y_t)
  k2 = f(y_t+k1*h/2)
  k3 = f(y_t+k2*h/2)
  k4 = f(y_t+k3*h)
  y_t1 = y_t+h*(k1+2*k2+2*k3+k4)/6
  return y_t1
```

→

```
def rk4_beta(f, y_t, h, x):
  k1 = f_beta(y_t, x)
  k2 = f_beta(y_t+k1*h/2, x)
  k3 = f_beta(y_t+k2*h/2, x)
  k4 = f_beta(y_t+k3*h, x)
  y_t1 = y_t+h*(k1+2*k2+2*k3+k4)/6
  return y_t1
```

3. MSE 함수 가져오자.

```
from sklearn.metrics import mean_squared_error
```

문제 3 : COVID-19 수학 모델

4. 실제 데이터와 모델의 예측 값을 $\boldsymbol{\beta}$ 에 대한 RMSE 함수를 정의하자.

```
pred_cases = np.zeros(len(time))
```

```
pred_cases[0] = I0
```

```
for i in range(n):
  y[i+1,:] = rk4(f, y[i,:], h)
  pred_cases[i+1] = alpha*y[i,1]
```

```
def MSE_beta(x):
  pred_cases = np.zeros(len(time))
  pred_cases[0]=I0
  for i in range(n):
    y[i+1,:] = rk4_beta(f_beta, y[i,:], h, x)
    pred_cases[i+1] = alpha*y[i,1]
  result = mean_squared_error(data['Cases'],pred_cases)
  return result
```

문제 3 : COVID-19 수학 모델

5. RMSE 함수를 목적함수로 만들자.

```
def Obj_beta(x):
    return MSE_beta(x)
```

6. 목적함수를 Minimize에 적용하자.

```
bound=(0,10)
```

```
param = minimize(Obj_beta, 1.0, method='SLSQP', bounds=(bound,))
```

```
param
```

```
 message: Optimization terminated successfully
 success: True
  status: 0
     fun: 251.92722631402904
       x: [ 1.392e+00]
     nit: 9
     jac: [ 6.561e-04]
    nfev: 24
    njev: 9
```

최소가 되는 MSE : 251.927

최적의 $\boldsymbol{\beta} = \mathbf{1.392}$

문제 3 : COVID-19 수학 모델

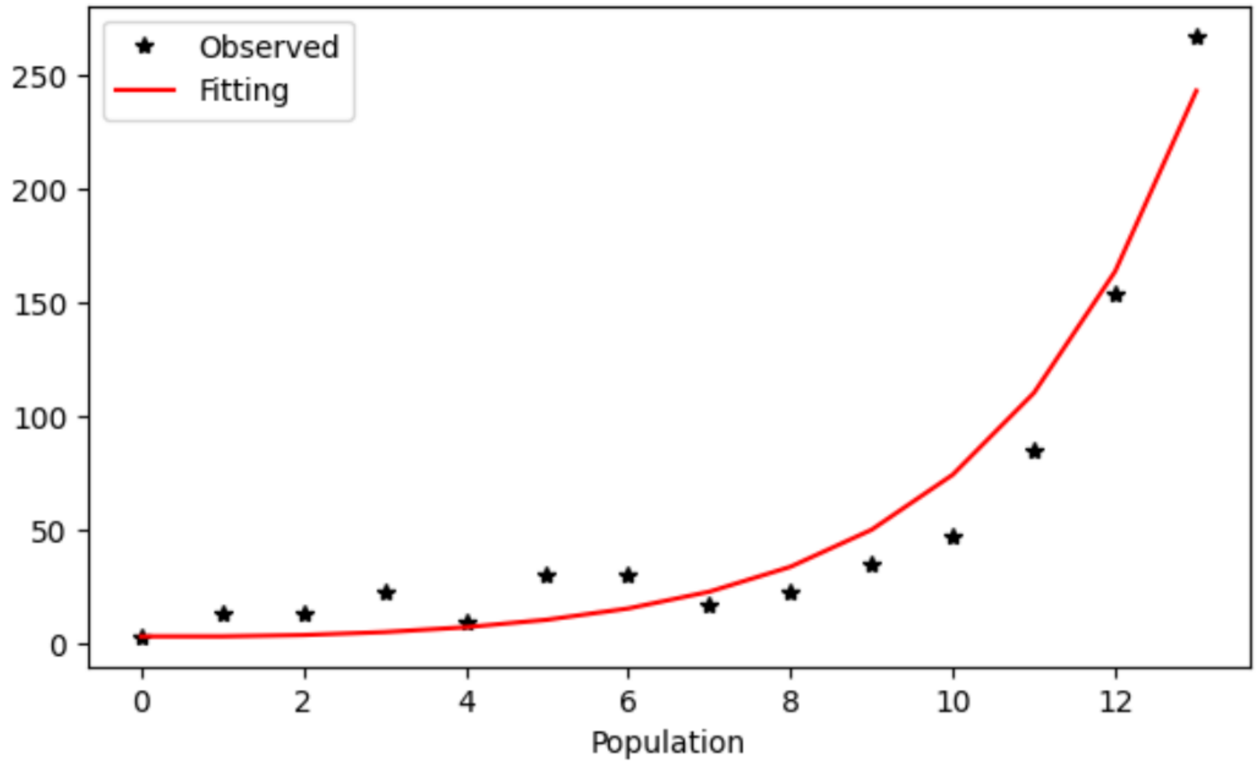
7. 앞서 구한 최적의 $\boldsymbol{\beta}$ 를 모델에 적용하여 SEIR 모델을 시뮬레이션 하자.

```
pred_cases = np.zeros(len(time))
pred_cases[0]=10
for i in range(n):
  y[i+1,:] = rk4_beta(f_beta, y[i,:], h, param.x)
  pred_cases[i+1] = alpha*y[i,1]
```

문제 3 : COVID-19 수학 모델

결과를 확인해보자.

```
plt.figure(figsize=(7,4))
plt.plot(time,data['Cases'],'*k')
plt.plot(time,pred_cases,'r')
plt.xlabel('time')
plt.xlabel('Population')
plt.legend(('Observed','Fitting'),loc='best')
plt.show()
```



Thank you