[EEC3600-001] 수치해석		
소속: 전기전자공학부	학번: 12191529	이름: 장준영
Python Programming for Numerical Analysis		**HW number: #4**

4.1. (4.1.1) Least Squares with orthogonal columns. (4.1.2) Least Squares and QR Factorization

a. 문제

Problem 4.1.1 *(least squares with orthogonal columns)* Suppose the $m \times n$ matrix Q has orthogonal columns, i.e., $Q^\top Q = I_{n\times n}$.

(a) Show that $\hat{x} = Q^\top b$ is the vector that minimizes $\|Qx - b\|_2^2$.

(b) What is the computational complexity of computing $\hat{x}$, given Q and b, and how does it compare to the complexity of a general least squares problems with an $m \times n$ coefficient matrix?

문제 4.1.1

Problem 4.1.2 *(least squares and QR factorization)* Suppose the $A \in \mathbb{R}^{m\times n}$ has linearly independent columns and QR factorization $A = QR$, and $b \in \mathbb{R}^m$. The vector $A\hat{x}$ is the linear combination of the columns of A that is closest to the vector b, i.e., it is the projection of b onto the set of linear combinations of the columns of A.

(a) Show that $A\hat{x} = QQ^\top b$, where $QQ^\top$ is called the projection matrix (can you think of its geometry?)

(b) Show that $\|A\hat{x} - b\|_2^2 = \|b\|_2^2 - \|Q^\top b\|^2$. (This is that square of the distance between b and the closest linear combination of the columns of A.)

문제 4.1.2

b. 풀이

4.1.1 (a) $\min_x \|Qx-b\|_2^2$ 에서:

$$\begin{aligned}\|Qx-b\|_2^2 &= (Qx-b)^T(Qx-b)\\ &= x^TQ^TQx - 2b^TQx + b^Tb\\ &= x^Tx - 2b^TQx + b^Tb\end{aligned}$$

$Q^TQ = I$ (Q : Orthonormal Matrix)

$\rightarrow q_i^Tq = \begin{cases} 1 & (i=k) \\ 0 & (i\neq k)\end{cases}$

이를 x에 대해 미분한 후, 0 이 되는 x를 찾자.

$$\therefore \nabla_x(x^Tx - 2b^TQx) = 2x - 2Q^Tb = 0 \quad ; \quad \boxed{x = Q^Tb}$$

$\nabla_x(a^Tx) = a$

(b) ① $x = Q^Tb$: $O(mn)$

- $Q \in \mathbb{R}^{m\times n}$
- $b \in \mathbb{R}^m$
- $Q^Tb \in \mathbb{R}^n$

$\Rightarrow$ ■ 연산 개수 : $Q^T \in \mathbb{R}^{n\times m}$ 이므로 Q^Tb를 계산하려면

- 각 결과 원소 하나는 m번의 곱셈과 덧셈 : m
- 총 n개 원소 계산 : n

② 일반적인 Least Square Method : $O(mn^2+n^3)$

■ Normal equation : $\min_x \|Ax-b\|_2^2 \Rightarrow A^TAx = A^Tb$

(i) A^TA 계산

- $A^T \in \mathbb{R}^{n\times m}$, $A \in \mathbb{R}^{m\times n}$
- 결과 : $n\times n$ 행렬
- 각 원소는 m개의 곱셈, 덧셈

$\Rightarrow O(mn^2+n^3)$

(ii) A^Tb 계산

- $\mathbb{R}^{n\times m}\cdot\mathbb{R}^{m\times 1} = \mathbb{R}^{n\times 1}$ $\Rightarrow O(mn)$ (별로 크지 않음.)

(iii) $A^TAx = A^Tb$

- 가우스 소거법, LU 등 $\Rightarrow O(n^3)$

문제 4.1.1

4.1.2 (a) $A = QR$ 에서 $\begin{cases} Q: \text{orthogonal} \in \mathbb{R}^{m\times n} \rightarrow \underbrace{Q^TQ = I} \\ R: \text{Upper Triangular Matrix} \end{cases}$

∴ Normal Equation 에서 A는 선형 독립 이므로

$$A^TA\hat{x} = A^Tb \; ; \; \hat{x} = (A^TA)^{-1}A^Tb$$

$$; \; \hat{x} = \{(QR)^TQR\}^{-1}(QR)^Tb$$

$$= (R^T\underbrace{Q^TQ}_{I}R)^{-1}R^TQ^Tb$$

$$= R^{-1}\underbrace{(R^T)^{-1}R^T}_{I}Q^Tb \quad = R^{-1}Q^Tb$$

따라서 $A\hat{x} = (QR)\hat{x} = \underbrace{(QR)(R^{-1}}_{I}Q^T)b = QQ^Tb$ ∎

(b) 벡터 b는 두 부분으로 분해할 수 있다.

- $b_{\parallel} = A\hat{x} = QQ^Tb$: A의 열공간 방향.
- $b_{\perp} = b - A\hat{x} = b - QQ^Tb$: A의 열공간에 수직.

$$\therefore \|QQ^Tb\|^2 = (QQ^Tb)^T(QQ^Tb) = (b^TQQ^T)(QQ^Tb)$$
$$= b^TQQ^Tb$$
$$= (Q^Tb)^T(Q^Tb) = \|Q^Tb\|^2$$

따라서 $\|A\hat{x}-b\|^2 = \|b_{\perp}\|^2 = \|b\|^2 - \|QQ^Tb\|^2 = \|b\|^2 - \|Q^Tb\|^2$ ∎

문제 4.1.2

4.2. **(Permutation Matrix) B가 A의 행 교환으로 생성된 행렬일 때, A의 역행렬을 알고 있다면 B의 역행렬을 (직접 구하지 말고) 구하시오.**

a. 문제

Problem 4.2 Let

$$\mathbf{A} = \begin{bmatrix} 3 & -1 & 2 \\ 0 & 1 & 3 \\ -2 & 2 & -4 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 1 & 3 \\ 3 & -1 & 2 \\ -2 & 2 & -4 \end{bmatrix}$$

(note that $\mathbf{B}$ is obtained by interchanging the first two rows of $\mathbf{A}$). Knowing that

$$\mathbf{A}^{-1} = \begin{bmatrix} 0.5 & 0 & 0.25 \\ 0.3 & 0.4 & 0.45 \\ -0.1 & 0.2 & -0.15 \end{bmatrix}$$

determine $\mathbf{B}^{-1}$.

(Caution!) Do not directly compute the matrix inversion, but use A^{-1} to compute B^{-1}.

b. 풀이

sol) $\underbrace{B = PA}$ $\therefore P$: Permutation 행렬.

여기서는 1행 ⇔ 2행 교환을 수행하는 정사각행렬!

$$\Rightarrow P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

따라서 $B^{-1} = A^{-1}P^{-1}$, 그런데 $P^{-1} = P$ 이므로 $B^{-1} = A^{-1}P$

→ B^{-1}는 A^{-1}의 1열 ⇔ 2열 교환과 같다

답: $B^{-1} = \begin{bmatrix} 0 & 0.5 & 0.25 \\ 0.4 & 0.3 & 0.45 \\ 0.2 & -0.1 & -0.15 \end{bmatrix}$

4.3. (Least Square Method 적용 1) 질량-연비 관계 추정

a. 문제

Problem 4.3 The following table displays the mass M and average fuel consumption ϕ of motor vehicles manufactured by Ford and Honda in 2008. Fit a straight line $\phi = a + bM$ to the data and compute the standard deviation.

Model	M(kg)	ϕ(km/liter)
Focus	1198	11.90
Crown Victoria	1715	6.80
Expedition	2530	5.53
Explorer	2014	6.38
F-150	2136	5.53
Fusion	1492	8.50
Taurus	1652	7.65
Fit	1168	13.60
Accord	1492	9.78
CR-V	1602	8.93
Civic	1192	11.90
Ridgeline	2045	6.38

b. Python 코드

```python
import numpy as np

# 데이터 입력
M = np.array([1198, 1715, 2530, 2014, 2136, 1492, 1652, 1168, 1492, 1602, 1192, 2045], dtype=float)
phi = np.array([11.90, 6.80, 5.53, 6.38, 5.53, 8.50, 7.65, 13.60, 9.78, 8.93, 11.90, 6.38], dtype=float)

# A 행렬 구성: [1, M]
A = np.vstack((np.ones_like(M), M)).T

# A^T A, A^T b 계산
ATA = A.T @ A
ATb = A.T @ phi

# 가우스 소거법 + 역행렬로 해결 가능
def gaussian_elimination_solve(A, b):
    A = A.astype(float)
    b = b.astype(float)
    n = len(b)
    # Forward Elimination
    for i in range(n):
        for j in range(i+1, n):
            factor = A[j,i] / A[i,i]
            A[j,i:] = A[j,i:] - factor * A[i,i:]
            b[j] = b[j] - factor * b[i]
    # Backward Substitution
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (b[i] - np.dot(A[i,i+1:], x[i+1:])) / A[i,i]
    return x

# x = [a, b]
x = gaussian_elimination_solve(ATA.copy(), ATb.copy())
a, b = x

# 예측값 계산 및 표준 편차 계산
phi_pred = a + b * M
sigma = np.sqrt(np.mean((phi - phi_pred)**2))
```

c. Python 출력 결과

```
\Users\SAMSUNG\OneDrive\Desktop\대학\Solution 모음\25-1\수치해석\HW\HW_4\hw4_3.py" "
회귀계수: a = 18.4099, b = -0.005833
```

d. 문제 해결 과정

1) $A^T A, A^T \emptyset$ 각각 계산.

2) 가우스 소거법 -> Backward Substitution 사용하여 $x = \begin{bmatrix} a \\ b \end{bmatrix}$ 구하기.

4.4. (Least Square Method 적용 2) Linear vs Quadratic

a. 문제

Problem 4.4 Fit a straight line and a quadratic to the data in the following table. Which

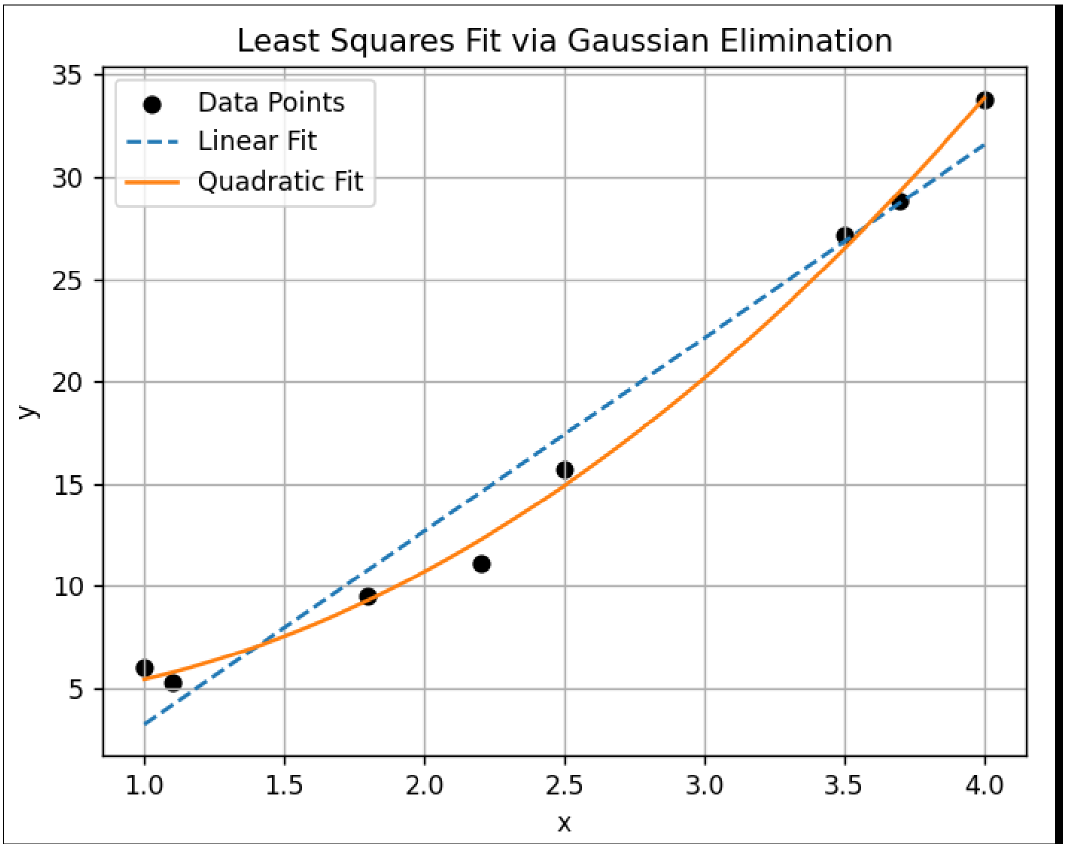
x	1.0	2.5	3.5	4.0	1.1	1.8	2.2	3.7
y	6.008	15.722	27.130	33.772	5.257	9.549	11.098	28.828

is a better fit?

b. Python 코드

```python
# 가우스 소거 + 백워드 서브스티튜션
def gaussian_elimination(A, b):
    A = A.astype(float)
    b = b.astype(float)
    n = len(b)
    # Forward Elimination
    for i in range(n):
        for j in range(i+1, n):
            factor = A[j, i] / A[i, i]
            A[j, i:] -= factor * A[i, i:]
            b[j] -= factor * b[i]
    # Backward Substitution
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (b[i] - A[i, i+1:] @ x[i+1:]) / A[i, i]
    return x
# 데이터
x = np.array([1.0, 2.5, 3.5, 4.0, 1.1, 1.8, 2.2, 3.7])
y = np.array([6.008, 15.722, 27.130, 33.772, 5.257, 9.549, 11.098, 28.828])

# 1차 회귀: A = [1, x]
A1 = np.vstack((np.ones_like(x), x)).T
ATA1 = A1.T @ A1
ATy1 = A1.T @ y
x1 = gaussian_elimination(ATA1.copy(), ATy1.copy())
y_pred1 = A1 @ x1
rss1 = np.sum((y - y_pred1)**2)

# 2차 회귀: A = [1, x, x^2]
A2 = np.vstack((np.ones_like(x), x, x**2)).T
ATA2 = A2.T @ A2
ATy2 = A2.T @ y
x2 = gaussian_elimination(ATA2.copy(), ATy2.copy())
y_pred2 = A2 @ x2
rss2 = np.sum((y - y_pred2)**2)

# 결과 출력
print("1차 회귀식: y = {:.4f} + {:.4f}x".format(x1[0], x1[1]))
print("RSS = {:.4f}".format(rss1))

print("2차 회귀식: y = {:.4f} + {:.4f}x + {:.4f}x^2".format(x2[0], x2[1], x2[2]))
print("RSS = {:.4f}".format(rss2))
```

c. Python 출력 결과




```
1차 회귀식: y = -6.1899 + 9.4385x
RSS = 30.2015
2차 회귀식: y = 4.4057 + -1.0689x + 2.1081x^2
RSS = 3.3043
```

d. 문제 해결 과정

1) $y = a + bx$에 대한 선형회귀 수행.
2) $y = a + bx + cx^2$에 대한 이차회귀 수행.
3) Python 코드 동작과정 설명
   - A^TA, A^Tb 각각 계산.
   - 가우스 소거법 -> Backward Substitution 사용하여 계수 최적화.
   - 각 방법에 대한 $rss = \sum(y - \hat{y})^2$ 계산. -> 비교.

4.5. (Least Square Method 적용 3) sin, cos 계수 추정

a. 문제

Problem 4.5 Determine a and b for which $f(x) = a\sin(\pi x/2) + b\cos(\pi x/2)$ fits the following data in the least-squares sense.

x	-0.5	-0.19	0.02	0.20	0.35	0.50
y	-3.558	-2.874	-1.995	-1.040	-0.068	0.677

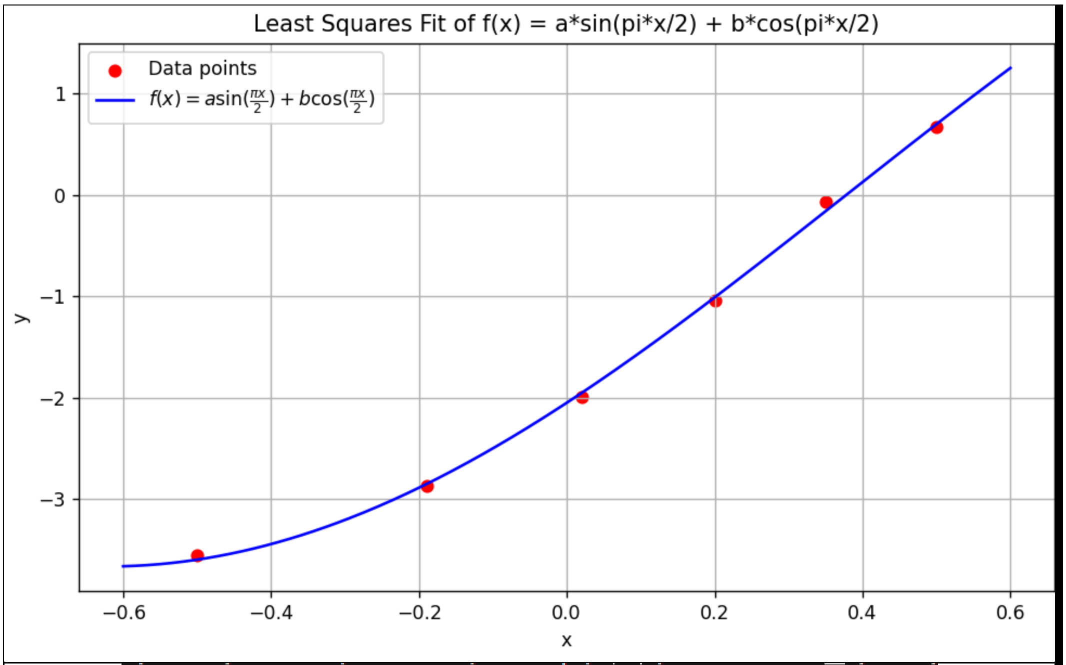
b. Python 코드

```python
# 가우스 소거법 함수
def gaussian_elimination(A, b):
    A = A.astype(float)
    b = b.astype(float)
    n = len(b)
    for i in range(n):
        for j in range(i+1, n):
            factor = A[j,i] / A[i,i]
            A[j,i:] -= factor * A[i,i:]
            b[j] -= factor * b[i]
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (b[i] - A[i,i+1:] @ x[i+1:]) / A[i,i]
    return x

# 데이터
x_vals = np.array([-0.5, -0.19, 0.02, 0.20, 0.35, 0.50])
y_vals = np.array([-3.558, -2.874, -1.995, -1.040, -0.068, 0.677])

# 디자인 행렬 A = [sin(πx/2), cos(πx/2)]
A = np.column_stack((
    np.sin(np.pi * x_vals / 2),
    np.cos(np.pi * x_vals / 2)
))
```

```python
# 정규 방정식
ATA = A.T @ A
ATy = A.T @ y_vals

# 계수 추정
params = gaussian_elimination(ATA.copy(), ATy.copy())
a, b = params

# 예측값
y_pred = A @ params
rss = np.sum((y_vals - y_pred)**2)

print(f"회귀식: f(x) = {a:.4f} * sin(πx/2) + {b:.4f} * cos(πx/2)")
print(f"RSS = {rss:.6f}")
```

```python
# 시각화용 x축 범위 및 예측 함수 정의
x_plot = np.linspace(-0.6, 0.6, 300)
y_plot = a * np.sin(np.pi * x_plot / 2) + b * np.cos(np.pi * x_plot / 2)

# 플로팅
plt.figure(figsize=(8, 5))
plt.scatter(x_vals, y_vals, color='red', label='Data points')
plt.plot(x_plot, y_plot, label=r'$f(x) = a \sin(\frac{\pi x}{2}) + b \cos(\frac{\pi x}{2})$', color='blue')
plt.title("Least Squares Fit of f(x) = a*sin(pi*x/2) + b*cos(pi*x/2)")
plt.xlabel("x")
plt.ylabel("y")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

c. Python 출력 결과




```
회귀식: f(x) = 3.0385 * sin(πx/2) + -2.0496 * cos(πx/2)
RSS = 0.013635
```

d. 문제 해결 과정

1) sin함수, cos함수에 각각 주어진 x 데이터 입력하여 계수행렬 A 생성.
2) Ax=y 풀기.
   - $A^T A, A^T b$ 각각 계산.
   - 가우스 소거법 -> Backward Substitution 사용하여 계수 최적화.

4.6. (Least Square Method 적용 4) 방사능 물질의 반감기 추정

a. 문제

Problem 4.6 The intensity of radiation of a radioactive substance was measured at half-year intervals. The results were

t (years)	0	0.5	1	1.5	2	2.5
γ	1.000	0.994	0.990	0.985	0.979	0.977
t (years)	3	3.5	4	4.5	5	5.5
γ	0.972	0.969	0.967	0.960	0.956	0.952

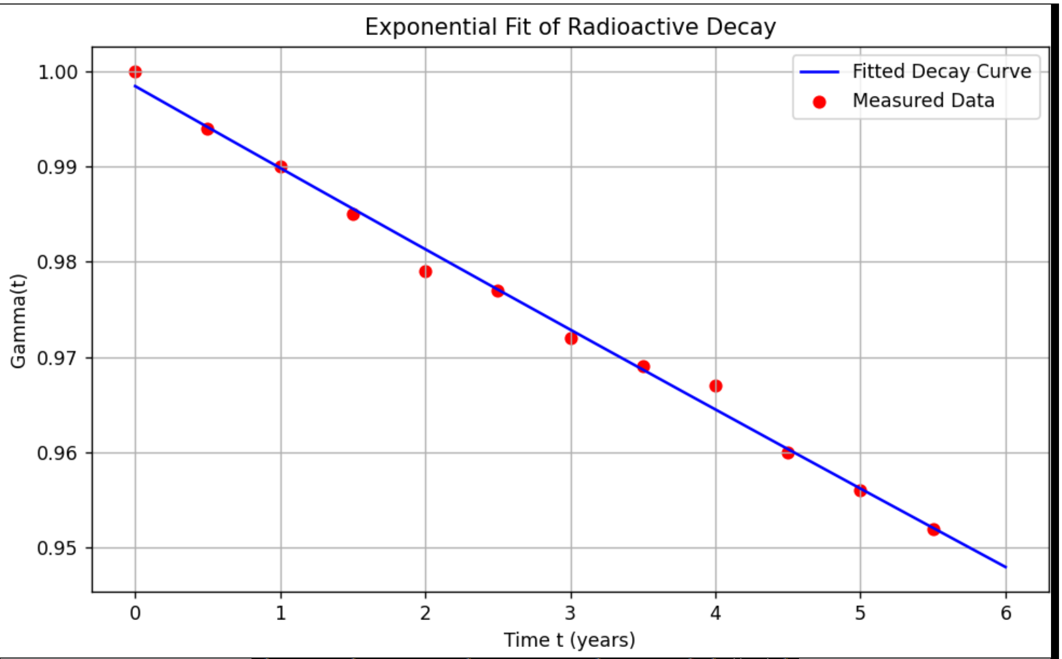
where γ is the relative intensity of radiation. Knowing that radioactivity decays exponentially with time, $\gamma(t) = ae^{-bt}$, estimate the radioactive half-life of the substance.

b. Python 코드

```python
# 가우스 소거법 함수
def gaussian_elimination(A, b):
    A = A.astype(float)
    b = b.astype(float)
    n = len(b)
    for i in range(n):
        for j in range(i+1, n):
            factor = A[j,i] / A[i,i]
            A[j,i:] -= factor * A[i,i:]
            b[j] -= factor * b[i]
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (b[i] - A[i,i+1:] @ x[i+1:]) / A[i,i]
    return x

# 데이터
t = np.array([0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5])
gamma = np.array([1.000, 0.994, 0.990, 0.985, 0.979, 0.977,
                  0.972, 0.969, 0.967, 0.960, 0.956, 0.952])

# ln(γ) = ln a - bt 형태로 변환
Y = np.log(gamma)
A = np.vstack((np.ones_like(t), -t)).T  # Y = c - bt
```

```python
# 정규방정식
ATA = A.T @ A
ATY = A.T @ Y

# 계수 추정
params = gaussian_elimination(ATA.copy(), ATY.copy())
c, b = params
a = np.exp(c)
half_life = np.log(2) / b

# 출력
print(f"계수: a = {a:.6f}, b = {b:.6f}")
print(f"Radioactive Half-life ≈ {half_life:.4f} years")

# 예측값 및 시각화
t_plot = np.linspace(0, 6, 200)
gamma_fit = a * np.exp(-b * t_plot)
```

c. Python 출력 결과




```
계수: a = 0.998416, b = 0.008640
Radioactive Half-life ≈ 80.2295 years
```

d. 문제 해결 과정

1) $\gamma(t) = ae^{-bt}$; $\ln\{\gamma(t)\} = lna - bt$ $\quad\quad \therefore Y = c - bt \ (c = lna)$
2) Ax=y 풀기.
   - $A^T A, A^T b$ 각각 계산.
   - 가우스 소거법 -> Backward Substitution 사용하여 계수 최적화.

4.7. (Least Square Method 적용 5) 다변수

a. 문제

Problem 4.7 Linear regression can be extended to data that depend on two or more variables (called multiple linear regression). If the dependent variable is z and independent variables are x and y, the data to be fitted have the form

x_1	y_1	z_1
x_2	y_2	z_2
x_3	y_3	z_3
$\vdots$	$\vdots$	$\vdots$
x_n	y_n	z_n

Instead of a straight line, the fitting function now represents a plane:

$$f(x, y) = a + bx + cy\,.$$

Show that the normal equations for the coefficients are

$$\begin{bmatrix} n & \Sigma x_i & \Sigma y_i \\ \Sigma x_i & \Sigma x_i^2 & \Sigma x_i y_i \\ \Sigma y_i & \Sigma x_i y_i & \Sigma y_i^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \Sigma z_i \\ \Sigma x_i z_i \\ \Sigma y_i z_i \end{bmatrix}.$$

b. 풀이

sol) $z_i = a + bx_i + cy_i$

이를 행렬로 쓰면 $z = A \cdot \theta$:

$$\theta = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}, \quad A = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix}$$

Normal Equation 을 계산하면:

$$A^T A \theta = A^T z \;;\; ① A^T A = \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix}}_{\in \mathbb{R}^{3\times n}} \underbrace{\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix}}_{\in \mathbb{R}^{n\times 3}} = \boxed{\underbrace{\begin{bmatrix} n & \sum x_i & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 \end{bmatrix}}_{\in \mathbb{R}^{3\times 3}}}$$

$$② A^T z = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix} = \boxed{\begin{bmatrix} \sum z_i \\ \sum x_i z_i \\ \sum y_i z_i \end{bmatrix}}$$