exercise2 (Score: 4.0 / 21.0)

Exercise 2

Suppose that a planet follows an elliptical orbit, which can be represented in a Cartesian coordinate system by the equation of the form

$$\alpha_1 y^2 + \alpha_2 xy + \alpha_3 x + \alpha_4 y + \alpha_5 = x^2. \qquad (1)$$

Based on the observation of the planet's position:

$$ \left [

```
    \begin{array}{c}
     x \\
     y
    \end{array}
  \right ] =
  \left [
    \begin{array}{cccccccccc}
```

1.02 & 0.95 & 0.87 & 0.77 & 0.67 & 0.56 & 0.44 & 0.30 & 0.16 & 0.01\ 0.39 & 0.32 & 0.27 & 0.22 & 0.18 & 0.15 & 0.13 & 0.12 & 0.13 & 0.15 \end{array} \right],$$

we want to determine the orbital parameters α_i, $i = 1, 2, \cdots, 5$, that solve the linear least squares problem of the form: $\min_{\alpha_i} \|b - A\alpha\|_2$, where the vector $b \in \mathrm{R}^{10}$, $\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]^T \in \mathrm{R}^5$ and the matrix $A \in \mathrm{R}^{10\times 5}$ can be obtained easily when we substitute the aboe data to the equation (1).

Part 0

Import necessary libraries

In [1]:

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

Part 1

Find the solution of the problem by solving the associated normal equations via Cholesky factorization.

Part 1.1

Prepare data vector x, y and store them into 1D arrays: `data_x` , `data_y` .

In [2]:

(Top)

```
'''
Hint:
    data_x = ?
    data_y = ?
'''

# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

Out[2]:

```
'\nHint:\n    data_x = ?\n    data_y = ?\n'
```

Check your `data_x` and `data_y` .

In [3]:

cell-3b704739d6fd2990 (Top)

```
print('x =', data_x)
print('y =', data_y)
### BEGIN HIDDEN TESTS
assert np.mean(data_x - np.array([1.02, 0.95, 0.87, 0.77, 0.67, 0.56, 0.44, 0.30, 0.16, 0.01])) < 1e-7
assert np.mean(data_y - np.array([0.39, 0.32, 0.27, 0.22, 0.18, 0.15, 0.13, 0.12, 0.13, 0.15])) < 1e-7
### END HIDDEN TESTS
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-a3c229d28484> in <module>
----> 1 print('x =', data_x)
      2 print('y =', data_y)
      3 ### BEGIN HIDDEN TESTS
      4 assert np.mean(data_x - np.array([1.02, 0.95, 0.87, 0.77, 0.67, 0.56, 0.44, 0.30, 0.1
6, 0.01])) < 1e-7
      5 assert np.mean(data_y - np.array([0.39, 0.32, 0.27, 0.22, 0.18, 0.15, 0.13, 0.12, 0.1
3, 0.15])) < 1e-7

NameError: name 'data_x' is not defined
```

Part 1.2

Construct the matrix A and the vector b with the data x, y and the equation (1).

In [4]:

(Top)

```python
def construct_A_and_b(x, y):
    '''
    Arguments:
        x : 1D np.array, data x
        y : 1D np.array, data y

    Returns:
        A : 2D np.array
        b : 1D np.array
    '''

    # ===== 請實做程式 =====

    # ====================
```

Comments:
No response.

Check your A and b.

In [5]:

cell-ab0180156b91fc0c (Top)

```python
A, b = construct_A_and_b(data_x, data_y)
print('A:\n', A)
print('b:\n', b)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-c220f4908bd6> in <module>
----> 1 A, b = construct_A_and_b(data_x, data_y)
      2 print('A:\n', A)
      3 print('b:\n', b)

NameError: name 'data_x' is not defined
```

Part 1.3

As the [lecture (https://ceiba.ntu.edu.tw/course/7a770d/content/cmath2019_note4_linear_system_cholesky.pdf)](https://ceiba.ntu.edu.tw/course/7a770d/content/cmath2019_note4_linear_system_cholesky.pdf) noted, to solve the noraml eqaution via Cholesky factorization we need additional **Forward substitution** and **Backward substituion** besides the **Cholesky factorization**. Please implement and check these three algorithms at below.

Algorithm 1: Implement forward substitution to solve

$$Lx = b,$$

where L is a lower triangular matrix and b is a column vector.

(Note that you need to implement it by hand, simply using some package functions is not allowed.)

In [6]:

(Top)

```python
def forward_substitution(L, b):
    '''
    Arguments:
        L : 2D lower triangular np.array
        b : 1D np.array

    Return:
        x : solution to Lx = b
    '''

    # ===== 請實做程式 =====

    # ====================
```

Comments:
No response.

Check your function.

In [7]:

cell-55c3537517a849a7 (Top)

```python
L = np.array([
    [1, 0, 0, 0],
    [2, 1, 0, 0],
    [4, 5, 6, 0],
    [1, 2, 3, 4]
])
x = np.array([11, 22, 33, 24])
print('L:\n', L)
print('x:\n', x)
print('My answer:\n', forward_substitution(L, L @ x))
```

```
L:
 [[1 0 0 0]
 [2 1 0 0]
 [4 5 6 0]
 [1 2 3 4]]
x:
 [11 22 33 24]
My answer:
 None
```

Algorithm 2: Implement backward substitution to solve

$$Rx = b,$$

where R is an upper triangular matrix and b is a column vector.

(Note that you need to implement it by hand, simply using some package functions is not allowed.)

In [8]:

(Top)

```python
def backward_substitution(R, b):
    '''
    Arguments:
        R : 2D upper triangular np.array
        b : 1D np.array

    Return:
        x : solution to Rx = b
    '''

    # ===== 請實做程式 =====

    # ====================
```

Comments:
No response.

Check your function.

In [9]:

cell-b139cd9ef4098615 (Top)

```python
R = np.array([
    [1, 2, 3],
    [0, 4, 5],
    [0, 0, 9]
])
x = np.array([11, 22, 33])
print('R:\n', R)
print('x:\n', x)
print('My answer:\n', backward_substitution(R, R @ x))
```

```
R:
 [[1 2 3]
 [0 4 5]
 [0 0 9]]
x:
 [11 22 33]
My answer:
 None
```

Algorithm 3: Implement Cholesky decompostion to decompose a nonsingualr PSD (https://www.wikiwand.com/en/Definiteness_of_a_matrix) matrix A into

$$A = R^T R,$$

where R is an upper triangular matrix.

(Note that you need to implement it by hand, simply using some package functions is not allowed.)

In [10]:

(Top)

```python
def cholesky_decomposition(A):
    '''
    Arguments:
        A : 2D np.array

    Return:
        R : 2D np.array, A = R^T R
    '''

    # ===== 請實做程式 =====

    # ====================
```

Comments:
No response.

Check your function.

In [11]:

cell-cc45a402f856cb26 (Top)

```python
# Construct a PSD matrix A
_A = np.array([
    [1, 3, 2, 4],
    [4, 2, 1, 7],
    [2, 5, 9, 0],
    [3, 5, 8, 2]
])
A = _A.T @ _A

# Do Cholesky decomposition
R = cholesky_decomposition(A)
print('A:\n', A)
print('R:\n', R)
print('A = R.T @ R:\n', R.T @ R)
```

```
A:
 [[ 30  36  48  38]
 [ 36  63  93  36]
 [ 48  93 150  31]
 [ 38  36  31  69]]
R:
 None

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-11-931d65f4a9cc> in <module>
     12 print('A:\n', A)
     13 print('R:\n', R)
---> 14 print('A = R.T @ R:\n', R.T @ R)

AttributeError: 'NoneType' object has no attribute 'T'
```

Part 1.4

Implement the function `solve_alpha` to find α from the associated the normal equation.

In [12]:

(Top)

```python
def solve_alpha(x, y):
    '''
    Arguments:
        x : 1D np.array, data x
        y : 1D np.array, data y

    Returns:
        alpha : 1D np.array

    Hints:
        1. Find matrix A, vector b
        2. Find the associated normal equation
        3. Do Cholesky decomposition
        4. Solve the equation with forward/backward substition
    '''

    # ===== 請實做程式 =====

    # ====================
```

Comments:
No response.

Solve α !

In [13]:

cell-ada65b7c60848c59 (Top)

```python
alpha = solve_alpha(data_x, data_y)
print('alpha:\n', alpha)
### BEGIN HIDDEN TESTS
assert np.mean(alpha - np.array([-2.63562548,  0.14364618,  0.55144696,  3.22294034, -0.43289427])) < 1e-
7
### END HIDDEN TESTS
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-33879121bf0c> in <module>
----> 1 alpha = solve_alpha(data_x, data_y)
      2 print('alpha:\n', alpha)
      3 ### BEGIN HIDDEN TESTS
      4 assert np.mean(alpha - np.array([-2.63562548,  0.14364618,  0.55144696,  3.22294034,
-0.43289427])) < 1e-7
      5 ### END HIDDEN TESTS

NameError: name 'data_x' is not defined
```

Part 2

Perturb the input data slightly by adding to each coordinate of each data point a uniformly distributed random number, and solve the least square problem as before with the perturbed data.

Compare the new values for the parameters with those previously computed. What effect does this difference have on the plot of the orbit ?

Part 2.1

In order to plot the orbit, we need to transform the equation (1) into a graph $z = f(x, y, \alpha)$ and then plot the contour at $z = 0$ by the tool `plt.contour` .

In [14]:

(Top)

```python
def ellipse(x, y, alpha):
    '''
    Arguments:
        x : 1D np.array, data x
        y : 1D np.array, data y
        alpha : 1D np.array, the coefficients

    Returns:
        z : 1D np.array, z=f(x, y, alpha) from equation (1)
    '''
    # ===== 請實做程式 =====

    # ====================
```

Comments:
No response.

Plot the orbit.

In [15]:

cell-c944b24065f4673f (Top)

```python
# Plot the exact data points (x,y)
plt.scatter(data_x, data_y, label='data')

# Prepare mesh data points (X,Y) to plot the orbit
X, Y = np.meshgrid(
    np.linspace(-1, 1.5, 100),
    np.linspace(0, 1.5, 100)
)
# Plot the level curve at z = 0 only
plt.contour(X, Y, ellipse(X, Y, alpha), [0])

plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-bcf2e8b78519> in <module>
      1 # Plot the exact data points (x,y)
----> 2 plt.scatter(data_x, data_y, label='data')
      3 
      4 # Prepare mesh data points (X,Y) to plot the orbit
      5 X, Y = np.meshgrid(

NameError: name 'data_x' is not defined
```

Part 2.2

Now perturb the original data with some slight, uniformly random noise and follow the steps as before to find new `perturbed_x`, `perturbed_y`, `perturbed_alpha`.

In [16]:

(Top)

```
'''
Hint:
    perturbed_x = ?
    perturbed_y = ?
    perturbed_alpha = ?
'''

# ===== 請實做程式 =====

# ====================
```

Comments:
No response.

Out[16]:

```
'\nHint:\n    perturbed_x = ?\n    perturbed_y = ?\n    perturbed_alpha = ?\n'
```

Overlay the new perturbed orbit on the plot.

In [17]:

cell-7428d2eef3884195 (Top)

```
# Plot the exact data points (x,y)
plt.scatter(data_x, data_y, label='data')

# Plot the perturbed data points
plt.scatter(perturbed_x, perturbed_y, label='perturbed_data')

# Prepare mesh data points (X,Y) to plot the orbits
X, Y = np.meshgrid(
    np.linspace(-1, 1.5, 100),
    np.linspace(0, 1.5, 100)
)

# Plot the level curve at z = 0
plt.contour(X, Y, ellipse(X, Y, alpha), [0])

# Plot the level curve at z = 0 after perturbed
plt.contour(X, Y, ellipse(X, Y, perturbed_alpha), [0])

plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-17-8c777d12918f> in <module>
      1 # Plot the exact data points (x,y)
----> 2 plt.scatter(data_x, data_y, label='data')
      3 
      4 # Plot the perturbed data points
      5 plt.scatter(perturbed_x, perturbed_y, label='perturbed_data')

NameError: name 'data_x' is not defined
```

(Top)

Part 2.3

Try some different perturbations and compare the orbits before and after your perturbation. What's your observation?

Please write down your answer here.