

# Assignment 6

2024017201 최선웅

12034 수치해석

## 필요한 Package

```
pip install numpy matplotlib
```

- numpy 패키지: 대규모 수치 연산을 빠르고 쉽게 처리하는 라이브러리, 행렬 연산, 벡터 연산, 데이터 파일 읽기/쓰기, 최소제곱법 계산 수행
- matplotlib 패키지: 피팅 결과와 오차를 그림으로 확인하고, 그래프를 저장 및 출력할 수 있게 함

## Main.py 코드

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

plt.rcParams['font.sans-serif'] = ['DejaVu Sans']
plt.rcParams['axes.unicode_minus'] = False

def linear_fitting(filename):
    """
    Linear data fitting function using least squares method

    Input: filename - data file name (fitdata1.dat, fitdata2.dat, fitdata3.dat)

    Model:
        x' = a1*x + a2*y + a3
        y' = a4*x + a5*y + a6
    
```

```

"""
print(f"\n{'='*60}")
print(f"Processing file: {filename}")
print('='*60)

try:
    data = np.loadtxt(filename)
except FileNotFoundError:
    print(f"Error: {filename} not found.")
    return None

x = data[:, 0]
y = data[:, 1]
x_prime = data[:, 2]
y_prime = data[:, 3]

N = len(x)
print(f"Number of data points (N): {N}")

A = np.zeros((2*N, 6))
b = np.zeros(2*N)

A[0::2, 0] = x      # a1 coefficient
A[0::2, 1] = y      # a2 coefficient
A[0::2, 2] = 1      # a3 coefficient

A[1::2, 3] = x      # a4 coefficient
A[1::2, 4] = y      # a5 coefficient
A[1::2, 5] = 1      # a6 coefficient

b[0::2] = x_prime   # x' values
b[1::2] = y_prime   # y' values

a, residuals, rank, s = np.linalg.lstsq(A, b, rcond=None)

print("\n==== Estimated Parameters ===")
print(f"a1 = {a[0]:.8f}")

```

```

print(f"a2 = {a[1]:.8f}")
print(f"a3 = {a[2]:.8f}")
print(f"a4 = {a[3]:.8f}")
print(f"a5 = {a[4]:.8f}")
print(f"a6 = {a[5]:.8f}")

x_pred = a[0]*x + a[1]*y + a[2]
y_pred = a[3]*x + a[4]*y + a[5]

error_x = np.sum((x_pred - x_prime)**2)
error_y = np.sum((y_pred - y_prime)**2)
total_error = error_x + error_y

print("\n==== Error Analysis ===")
print(f"Squared error for x' equation: {error_x:.8f}")
print(f"Squared error for y' equation: {error_y:.8f}")
print(f"Total squared error: {total_error:.8f}")

print("\n==== Data Comparison (First 5 samples) ===")
print(f"{'i':<4} {'x':<12} {'y':<12} {'x_pred':<12} {'x_actual':<12} {'error':<12}")
print("-" * 64)
for i in range(min(5, N)):
    err = abs(x_pred[i] - x_prime[i])
    print(f"{i:<4} {x[i]:<12.4f} {y[i]:<12.4f} {x_pred[i]:<12.4f} {x_prime[i]:<12.4f} {err:<12.6f}")

return {
    'parameters': a,
    'x_pred': x_pred,
    'y_pred': y_pred,
    'total_error': total_error,
    'x_original': x,
    'y_original': y,
    'x_prime': x_prime,
    'y_prime': y_prime
}

```

```

def plot_results(results, filename):
    """
    Visualize results as plots
    """
    if results is None:
        return

    a = results['parameters']
    x = results['x_original']
    y = results['y_original']
    x_prime = results['x_prime']
    y_prime = results['y_prime']
    x_pred = results['x_pred']
    y_pred = results['y_pred']

    fig, axes = plt.subplots(2, 2, figsize=(12, 10))
    fig.suptitle(f"Linear Fitting Results: {filename}", fontsize=14, fontweight='bold')

    ax = axes[0, 0]
    ax.scatter(range(len(x_prime)), x_prime, label='Actual x_prime', alpha=0.6, s=30)
    ax.scatter(range(len(x_pred)), x_pred, label='Predicted x_prime', alpha=0.6, s=30)
    ax.set_xlabel('Data Index')
    ax.set_ylabel('x_prime Value')
    ax.set_title('x_prime Transformation (Actual vs Predicted)')
    ax.legend()
    ax.grid(True, alpha=0.3)

    ax = axes[0, 1]
    ax.scatter(range(len(y_prime)), y_prime, label='Actual y_prime', alpha=0.6, s=30)
    ax.scatter(range(len(y_pred)), y_pred, label='Predicted y_prime', alpha=0.6, s=30)
    ax.set_xlabel('Data Index')

```

```

        ax.set_ylabel('y_prime Value')
        ax.set_title('y_prime Transformation (Actual vs Predicted)')
        ax.legend()
        ax.grid(True, alpha=0.3)

        ax = axes[1, 0]
        error_x = x_pred - x_prime
        ax.scatter(range(len(error_x)), error_x, alpha=0.6, s=30, color='red')
        ax.axhline(y=0, color='k', linestyle='--', linewidth=0.8)
        ax.set_xlabel('Data Index')
        ax.set_ylabel('Error (Predicted - Actual)')
        ax.set_title('x_prime Error Distribution')
        ax.grid(True, alpha=0.3)

        ax = axes[1, 1]
        error_y = y_pred - y_prime
        ax.scatter(range(len(error_y)), error_y, alpha=0.6, s=30, color='red')
        ax.axhline(y=0, color='k', linestyle='--', linewidth=0.8)
        ax.set_xlabel('Data Index')
        ax.set_ylabel('Error (Predicted - Actual)')
        ax.set_title('y_prime Error Distribution')
        ax.grid(True, alpha=0.3)

plt.tight_layout()

output_filename = filename.replace('.dat', '_result.png')
plt.savefig(output_filename, dpi=150, bbox_inches='tight')
print(f"\nPlot saved: {output_filename}")
plt.show()

def main():
    """
    Main function: process all data files
    """
    print("\n" + "="*60)
    print("Linear Data Fitting (Least Squares Method)")

```

```

print("=*60)

filenames = ['fitdata1.dat', 'fitdata2.dat', 'fitdata3.dat']

all_results = {}

for filename in filenames:
    results = linear_fitting(filename)
    if results is not None:
        all_results[filename] = results
        plot_results(results, filename)

print("\n" + "*60)
print("Final Summary")
print("*60)

for filename, results in all_results.items():
    a = results['parameters']
    print(f"\n{filename}:")
    print(f"  a1={a[0]:.6f}, a2={a[1]:.6f}, a3={a[2]:.6f}")
    print(f"  a4={a[3]:.6f}, a5={a[4]:.6f}, a6={a[5]:.6f}")
    print(f"  Total squared error: {results['total_error']:.6f}")

if __name__ == "__main__":
    main()

```

해당 파일은 최소제곱법을 이용해 주어진 데이터에서 2차원 선형 변환 모델의 계수를 추정하고, 결과를 시각화하는 프로그램이다.

주요 흐름은 다음과 같다.

1. 데이터 로딩: 데이터 파일( `.dat` )에서  $(x, y)$ 와 변환된  $(x', y')$ 값을 읽어온다.
2. 모델 형태:

- $x' = a_1x + a_2y + a_3$
- $y' = a_4x + a_5y + a_6$

- 즉,  $(x, y)$ 를 입력하여 선형 변환으로  $(x', y')$ 을 예측하는 식이다.
- 최소제곱해 구하기: `numpy` 의 `lstsq` 로  $A$ 행렬과  $b$ 벡터를 만들어 이 모델 계수( $a_1 \sim a_6$ )를 찾는다. 이 과정에서 실제 변환값과 예측값의 오차 제곱합이 최소가 되도록 계수를 구한다.
  - 결과 출력: 추정된 계수, 오차 분석을 출력하고 일부 데이터를 비교한다.
  - 시각화: 실제값과 예측값 비교, 오차분포 등을 하위 플롯으로 그림 파일로 저장한다.

## 출력값

(assignment6) Assignment6 python main.py

```
=====
=====
Linear Data Fitting (Least Squares Method)
=====

=====
Processing file: fitdata1.dat
=====

Number of data points (N): 77

==== Estimated Parameters ===
a1 = 0.98188846
a2 = 0.00254056
a3 = -0.37517351
a4 = 0.00125034
a5 = 0.98216284
a6 = 1.15771487

==== Error Analysis ===
Squared error for x' equation: 1636.62952713
Squared error for y' equation: 2411.06592586
Total squared error: 4047.69545299
```

```

==== Data Comparison (First 5 samples) ====
i  x      y      x_pred   x_actual   error
-----
0 -333.0600 -220.3900 -327.9629 -328.0000  0.037142
1 -268.8100 -220.0100 -264.8756 -264.0000  0.875559
2 -201.1800 -222.4400 -198.4766 -200.0000  1.523384
3 -137.7300 -221.3300 -136.1730 -136.0000  0.172973
4 -74.6500  -220.2900 -74.2328  -72.0000  2.232806

```

Plot saved: fitdata1\_result.png

```
=====
=====
```

Processing file: fitdata2.dat

```
=====
=====
```

Number of data points (N): 77

==== Estimated Parameters ====

a1 = 0.97990699  
 a2 = 0.00045180  
 a3 = -1.19222732  
 a4 = -0.00106942  
 a5 = 0.98034647  
 a6 = 0.49156818

==== Error Analysis ====

Squared error for x' equation: 21.43512155  
 Squared error for y' equation: 25.53397845  
 Total squared error: 46.96910000

==== Data Comparison (First 5 samples) ====

i	x	y	x_pred	x_actual	error
0	-333.9200	-220.4200	-328.5024	-328.0000	0.502356
1	-267.8000	-221.0100	-263.7112	-264.0000	0.288828
2	-203.1900	-220.4300	-200.3991	-200.0000	0.399119

```
3 -136.8800 -220.7200 -135.4216 -136.0000 0.578383  
4 -72.5800 -221.3400 -72.4139 -72.0000 0.413878
```

Plot saved: fitdata2\_result.png

```
=====
```

```
=====
```

Processing file: fitdata3.dat

```
=====
```

```
=====
```

Number of data points (N): 77

== Estimated Parameters ==

a1 = 0.98080637

a2 = 0.00054523

a3 = -0.94445869

a4 = -0.00071673

a5 = 0.97910785

a6 = 0.42893554

== Error Analysis ==

Squared error for x' equation: 77.31373680

Squared error for y' equation: 56.72232004

Total squared error: 134.03605684

== Data Comparison (First 5 samples) ==

i	x	y	x_pred	x_actual	error
0	-333.0600	-220.3900	-327.7320	-328.0000	0.268010
1	-268.8100	-220.0100	-264.7150	-264.0000	0.714973
2	-201.1800	-222.4400	-198.3844	-200.0000	1.615636
3	-137.7300	-221.3300	-136.1516	-136.0000	0.151594

1762

Plot saved: fitdata3\_result.png

```
=====
```

```
=====
```

## Final Summary

---

---

fitdata1.dat:

a1=0.981888, a2=0.002541, a3=-0.375174  
a4=0.001250, a5=0.982163, a6=1.157715  
Total squared error: 4047.695453

fitdata2.dat:

a1=0.979907, a2=0.000452, a3=-1.192227  
a4=-0.001069, a5=0.980346, a6=0.491568  
Total squared error: 46.969100

fitdata3.dat:

a1=0.980806, a2=0.000545, a3=-0.944459  
a4=-0.000717, a5=0.979108, a6=0.428936  
Total squared error: 134.036057

이 출력은 선형 변호나 계수 추정 및 예측 오차 분석 결과이다. 각 데이터셋(fitdata1, fitdata2, fitdata3)에 대해 다음 항목을 보여준다.

### 1. Estimated Parameters (추정 계수)

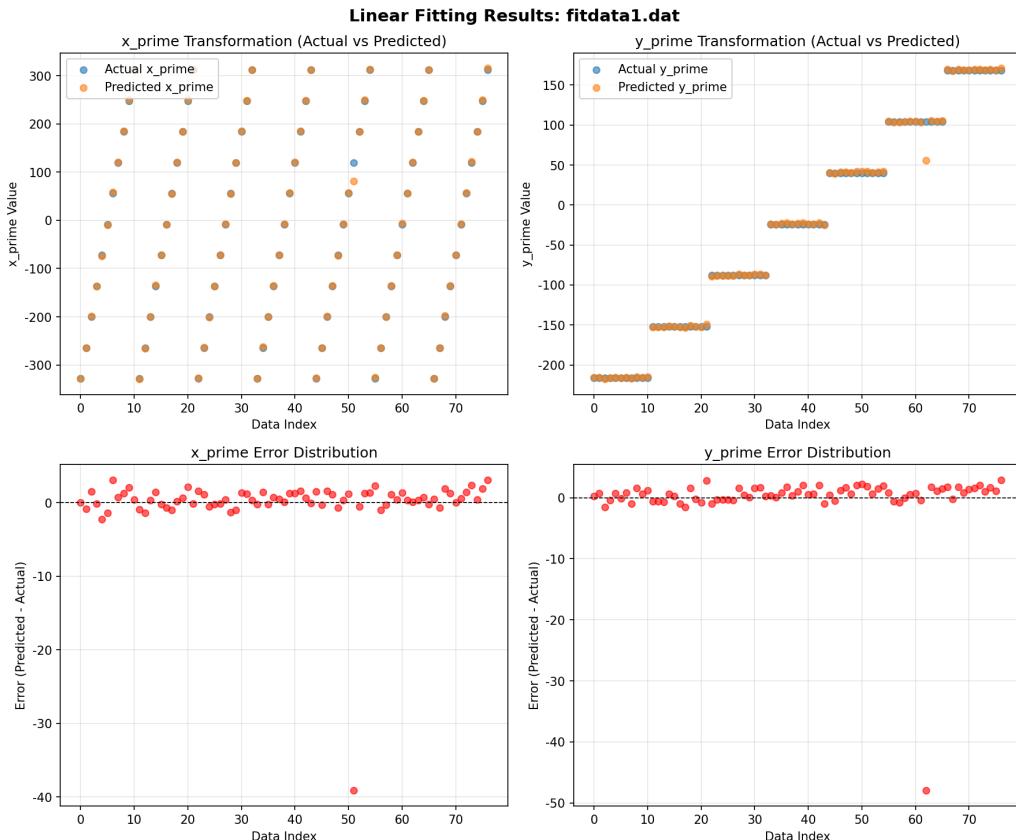
- 각 파일마다  $a_1, a_2, a_3, a_4, a_5, a_6$  값을 구함.
- 이 값들은 선형 변환 관계  
 $x' = a_1x + a_2y + a_3$   
 $y' = a_4x + a_5y + a_6$ 의 최적 계수이다.
- 대부분  $a_1$ 과  $a_5$ 는 약 0.98,  $a_2$ 와  $a_4$ 는 0에 가깝다. 즉,  $x', y'$ 은 주로 원래  $x, y$ 에 비례해 변환된다.

### 2. Error Analysis (오차 분석)

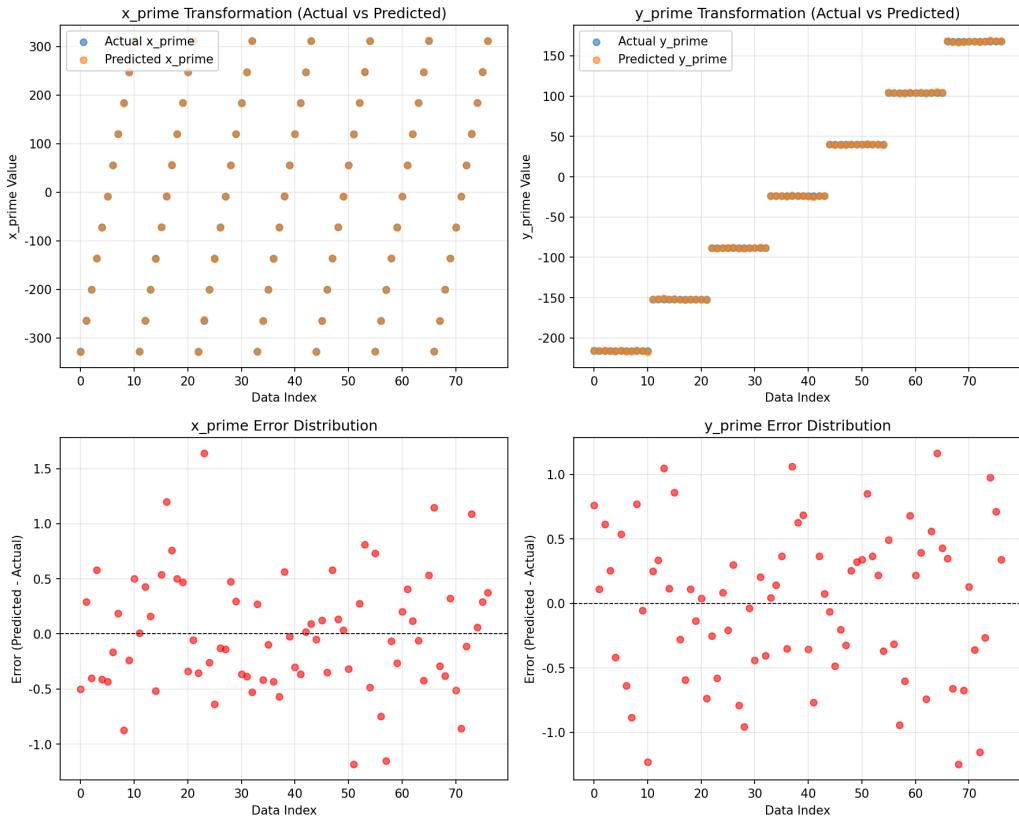
- Squared Error는 실제값과 예측값의 차이의 제곱합을 의미한다. 값이 작을수록 예측이 실제값에 잘 부합한다.
- `fitdata1.dat` 는 약 4048, `fitdata2.dat` 는 약 47, `fitdata3.dat` 는 약 134로, `fitdata1.dat` 의 오차가 상대적으로 크다.

### 3. Data Comparison (샘플 비교)

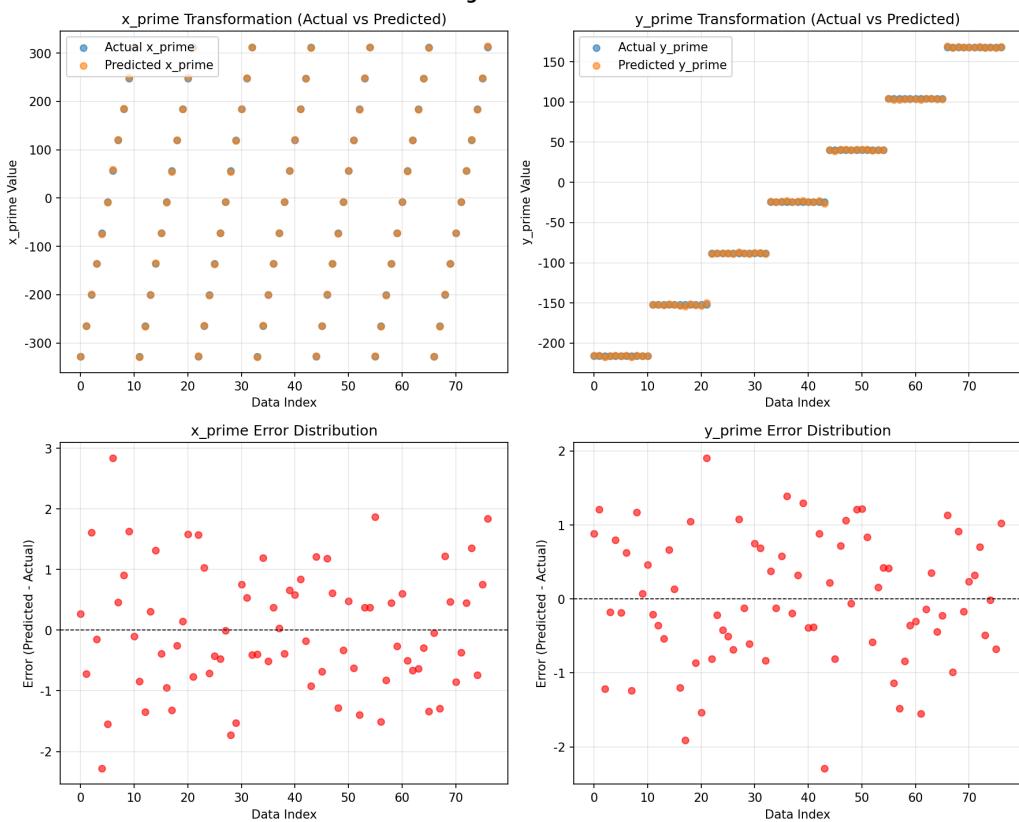
- 처음 5개의 샘플의  $x, y$ 와 예측값( $x_{\text{pred}}$ ), 실제값( $x_{\text{actual}}$ ), 오차를 출력한다.
- 오차는 대부분 0과 2사이로 작고, 실제값, 예측값, 원본값이 매우 비슷하다.



### Linear Fitting Results: fitdata2.dat



### Linear Fitting Results: fitdata3.dat



순서대로 `fitdata1.dat`, `fitdata2.dat`, `fitdata3.dat` 의 선형 회귀 분석 결과를 시각화한 자료이다.

## 왜 `fitdata1.dat`의 오차가 큰가?

`fitdata1.dat` 의 오차는 나머지 둘에 비하여 크게 나타났다. 다음은 해당 원인을 간단하게 분석하였다.

- `fitdata1.dat` 의  $y'$ 값이 대부분의 행에서 동일하게  $-216.00, -152.00, -88.00, -24.00, 40.00, 104.00, 168.00$ 등으로  $y$ 값에 관계없이 같은 값이다.
- 즉,  $x$ 와  $y$ 가 바뀌어도  $y'$ 의 값은 대부분 같은 값이다. 그런데 모델은  $y' = a_4x + a_5y + a_6$  형태로 모든 데이터를 맞추어야 하므로, 실제 데이터 패턴과 모델 구조가 잘 맞지 않다.