Chapter 4

The Least Squares Method

with error analysis and Python code

4.1 Introduction

The method of least squares is a fundamental technique for data fitting and regression analysis. It provides the best-fitting curve to a set of data points by minimizing the sum of the squares of the residuals (differences between observed and predicted values). Given data points:

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

and a model $y = f(x, \boldsymbol{\beta})$ (with parameters $\boldsymbol{\beta}$), the least squares objective is

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i - f(x_i, \boldsymbol{\beta})]^2.$$

4.2 Linear Least Squares

4.2.1 Mathematical Formulation

For the linear model

$$y = \beta_0 + \beta_1 x + \varepsilon,$$

we minimize

$$S(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2.$$

The normal equations give analytic solutions:

$$\beta_1 = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2}, \qquad \beta_0 = \bar{y} - \beta_1 \bar{x}.$$

4.2.2 Numerical Example

Data:

x_i	0	1	2	3	4	5
y_i	2.1	2.9	3.7	4.5	5.1	5.9

Fitted line (calculated via linear least squares):

$$\boxed{\hat{y} = 2.148 + 0.754\,x}$$

(values rounded to 3 decimals).

4.2.3 Residuals and Error Analysis (Linear)

The following table shows x_i, measured y_i, predicted $\hat{y}_i$, and the residual $r_i = y_i - \hat{y}_i$.

x_i	y_i	$\hat{y}_i$	r_i
0	2.100	2.148	-0.048
1	2.900	2.902	-0.002
2	3.700	3.656	0.044
3	4.500	4.410	0.090
4	5.100	5.165	-0.065
5	5.900	5.919	-0.019

Table 4.1: Linear fit residuals (rounded to 3 decimals).

Summary metrics for the linear fit:

$$\text{SSE} = \sum_i r_i^2 \approx 0.016762,$$
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum r_i^2} \approx 0.052855,$$
$$R^2 = 1 - \frac{\text{SSE}}{\sum (y_i - \bar{y})^2} \approx 0.998319.$$

4.2.4 Python implementation (linear + metrics)

```python
import numpy as np
import matplotlib.pyplot as plt
# Dr. Samir Kenouche - 30/10/2025
# Data
x = np.array([0,1,2,3,4,5])
y = np.array([2.1,2.9,3.7,4.5,5.1,5.9])

# Fit using least squares (design matrix with columns [x, 1])
A = np.vstack([x, np.ones(len(x))]).T
slope, intercept = np.linalg.lstsq(A, y, rcond=None)[0]

# Predictions and residuals
y_pred = slope * x + intercept
residuals = y - y_pred

# Metrics
sse = np.sum(residuals**2)
rmse = np.sqrt(np.mean(residuals**2))
```

```
ss_tot = np.sum((y - y.mean())**2)
r2 = 1 - sse/ss_tot

print(f"Linear fit: y = {intercept:.6f} + {slope:.6f} x")
print("SSE =", sse)
print("RMSE =", rmse)
print("R^2 =", r2)

# Plot
plt.scatter(x, y, label='Data')
plt.plot(x, y_pred, label='LS fit')
plt.xlabel('x'); plt.ylabel('y')
plt.legend(); plt.grid(True)
plt.title('Linear Least Squares Fit')
plt.savefig('linear_fit.png'); plt.show()
```

4.3 Nonlinear Least Squares

4.3.1 Formulation

For a nonlinear model $y = f(x, \boldsymbol{\beta})$, minimization of

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i - f(x_i, \boldsymbol{\beta})]^2$$

is typically performed by iterative algorithms (Gauss–Newton, Levenberg–Marquardt, etc.). The gradient is

$$\nabla S = -2J^T(\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})),$$

where J is the Jacobian matrix of partial derivatives $\partial f / \partial \beta_j$.

4.3.2 Example: Exponential Model

Fit

$$y = ae^{bx}$$

to data:

x_i	0	1	2	3	4	5
y_i	2.0	2.7	3.8	5.5	7.4	10.1

Using a log-transform as an initial estimator (and commonly used practical approach), we estimate

$$\ln y = \ln a + bx \quad \Rightarrow \quad b \approx 0.328325, \quad a \approx 1.985521.$$

Predicted values (rounded) give a very good fit.

4.3.3 Residuals and Error Analysis (Exponential)

Table of observed, predicted, and residuals for the exponential model:

x_i	y_i	$\hat{y}_i = ae^{bx_i}$	r_i
0	2.000	1.986	0.014
1	2.700	2.757	-0.057
2	3.800	3.829	-0.029
3	5.500	5.317	0.183
4	7.400	7.383	0.017
5	10.100	10.252	-0.152

Table 4.2: Exponential fit residuals (rounded to 3 decimals).

Summary metrics for the exponential fit:

$$\text{SSE} \approx 0.061393,$$
$$\text{RMSE} \approx 0.101154,$$
$$R^2 \approx 0.998704.$$

4.3.4 Python implementation (nonlinear - exponential)

Below is Python code using `scipy.optimize.curve_fit` if available; if `scipy` is not present, the log-transform approach (shown) provides good initial estimates and is included as a fallback.

```
import numpy as np
import matplotlib.pyplot as plt
# Dr. Samir Kenouche - 30/10/2025
x = np.array([0,1,2,3,4,5])
y = np.array([2.0,2.7,3.8,5.5,7.4,10.1])

# Preferred: use scipy.optimize.curve_fit if available
try:
    from scipy.optimize import curve_fit

    def model(x, a, b):
        return a * np.exp(b * x)

    params, cov = curve_fit(model, x, y, p0=(2.0, 0.3))
    a, b = params
    y_fit = model(x, a, b)
except Exception:
    # Fallback: linearize by taking logs (works when y_i > 0)
    logy = np.log(y)
    b, loga = np.polyfit(x, logy, 1) # logy = b*x + loga
    a = np.exp(loga)
    y_fit = a * np.exp(b * x)

residuals = y - y_fit
sse = np.sum(residuals**2)
rmse = np.sqrt(np.mean(residuals**2))
ss_tot = np.sum((y - y.mean())**2)
r2 = 1 - sse/ss_tot
```

```python
print(f"Exponential fit: a = {a:.6f}, b = {b:.6f}")
print("SSE =", sse)
print("RMSE =", rmse)
print("R^2 =", r2)

# Plot
plt.scatter(x, y, label='Data')
plt.plot(x, y_fit, label=f'Fit: y={a:.3f} e^{{{b:.3f} x}}')
plt.xlabel('x'); plt.ylabel('y')
plt.legend(); plt.grid(True)
plt.title('Nonlinear (Exponential) Least Squares Fit')
plt.savefig('nonlinear_fit.png'); plt.show()
```

4.4 Discussion

- Both fits show high R^2 (near 0.999) for these small, well-behaved example datasets.
- The linear fit residuals are small and nearly symmetric, indicating a good linear approximation.
- The exponential model fits the accelerating growth in the second dataset; using a log-transform gives a very good initial estimate and is often used in practice. For strict nonlinear LS on original y-space, Levenberg–Marquardt (via `scipy.optimize.curve_fit`) is recommended.
- RMSE and SSE provide absolute error scale; R^2 indicates the fraction of variance explained.

4.5 Conclusion

This chapter presented the theory and practice of least squares (linear and nonlinear), two worked numerical examples, full error analysis (residuals, SSE, RMSE, R^2), and Python code to reproduce the computations and plots. The document can be used either as a standalone report or integrated as a chapter in a larger numerical methods document.

Appendix: Mathematical Formulation of Linear and Nonlinear Least Squares

.1 General Least Squares Principle

The least squares method is based on minimizing the total squared deviations between observed data y_i and model predictions $f(x_i, \boldsymbol{\beta})$:

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i - f(x_i, \boldsymbol{\beta})]^2.$$

The goal is to find the parameter vector $\boldsymbol{\beta}^*$ such that:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} S(\boldsymbol{\beta}).$$

This formulation applies to both linear and nonlinear models.

.2 Linear Least Squares Formulation

.2.1 Model and Matrix Representation

Consider the general linear model:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_m x_{im} + \varepsilon_i, \quad i = 1, 2, \ldots, n.$$

In matrix notation:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1m} \\ 1 & x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}.$$

.2.2 Minimization and Normal Equations

The objective function is:

$$S(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}).$$

Differentiating with respect to $\boldsymbol{\beta}$:

$$\frac{\partial S}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) = 0.$$

This leads to the **normal equations**:

$$\boxed{\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y}.}$$

If $\mathbf{X}^T\mathbf{X}$ is invertible, the least squares estimator is:

$$\boxed{\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}.}$$

.2.3 Geometric Interpretation

The fitted values are:

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{P}\mathbf{y},$$

where $\mathbf{P} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is the projection matrix. Thus:

$$\mathbf{y} = \hat{\mathbf{y}} + \mathbf{r}, \quad \text{with } \mathbf{r} = \mathbf{y} - \hat{\mathbf{y}}.$$

Since $\mathbf{X}^T\mathbf{r} = 0$, the residuals are orthogonal to the space spanned by the columns of $\mathbf{X}$. Hence, least squares can be interpreted as an **orthogonal projection** of $\mathbf{y}$ onto the column space of $\mathbf{X}$.

.2.4 Variance and Covariance of Estimates

If the errors ε_i are independent and identically distributed with variance σ^2, then:

$$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}.$$

An unbiased estimate of σ^2 is:

$$\hat{\sigma}^2 = \frac{S(\hat{\boldsymbol{\beta}})}{n - m - 1}.$$

Confidence intervals for each parameter can then be constructed using this estimated variance.

.3 Nonlinear Least Squares Formulation

.3.1 Model Definition

For nonlinear regression models:

$$y_i = f(x_i, \boldsymbol{\beta}) + \varepsilon_i,$$

the function $f(x_i, \boldsymbol{\beta})$ is nonlinear in at least one component of $\boldsymbol{\beta}$.

The objective function is:

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} [y_i - f(x_i, \boldsymbol{\beta})]^2.$$

.3.2 Jacobian and Gradient

Define residuals:

$$r_i(\boldsymbol{\beta}) = y_i - f(x_i, \boldsymbol{\beta}), \quad \mathbf{r}(\boldsymbol{\beta}) = \begin{bmatrix} r_1(\boldsymbol{\beta}) \\ r_2(\boldsymbol{\beta}) \\ \vdots \\ r_n(\boldsymbol{\beta}) \end{bmatrix}.$$

Then:

$$S(\boldsymbol{\beta}) = \mathbf{r}^T(\boldsymbol{\beta})\mathbf{r}(\boldsymbol{\beta}).$$

The gradient is given by:

$$\nabla S = -2\mathbf{J}^T\mathbf{r},$$

where **J** is the **Jacobian matrix**:

$$\mathbf{J}_{ij} = \frac{\partial f(x_i, \boldsymbol{\beta})}{\partial \beta_j}.$$

.3.3 Iterative Solution Methods

Since no analytical solution exists, iterative optimization algorithms are applied.

(a) Gauss–Newton Method Linearizing the model around the current estimate $\boldsymbol{\beta}_k$:

$$f(x_i, \boldsymbol{\beta}) \approx f(x_i, \boldsymbol{\beta}_k) + \sum_j J_{ij}(\boldsymbol{\beta}_k)(\beta_j - \beta_{k,j}).$$

Minimizing S with respect to $\Delta\boldsymbol{\beta} = \boldsymbol{\beta} - \boldsymbol{\beta}_k$ yields:

$$(\mathbf{J}^T\mathbf{J})\Delta\boldsymbol{\beta} = \mathbf{J}^T\mathbf{r}.$$

Update rule:

$$\boxed{\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \Delta\boldsymbol{\beta}.}$$

(b) Levenberg–Marquardt Method A more stable variant introduces a damping parameter λ:

$$(\mathbf{J}^T\mathbf{J} + \lambda I)\Delta\boldsymbol{\beta} = \mathbf{J}^T\mathbf{r}.$$

This combines the speed of Gauss–Newton with the stability of gradient descent.

(c) Convergence Criteria Iterations stop when:

$$\|\Delta\boldsymbol{\beta}\| < \varepsilon_1, \quad |S(\boldsymbol{\beta}_{k+1}) - S(\boldsymbol{\beta}_k)| < \varepsilon_2, \quad k \geq k_{\max}.$$

.3.4 Covariance and Uncertainty of Nonlinear Estimates

After convergence, the covariance matrix of parameter estimates is approximated by:

$$\mathrm{Cov}(\hat{\boldsymbol{\beta}}) \approx \hat{\sigma}^2(\mathbf{J}^T\mathbf{J})^{-1},$$

with:

$$\hat{\sigma}^2 = \frac{S(\hat{\boldsymbol{\beta}})}{n - p}.$$

This approximation assumes near-linearity around the optimum and provides confidence intervals for fitted parameters.

.4 Comparison Summary

Feature	**Linear Least Squares**	**Nonlinear Least Squares**
Model form	$y = \mathbf{X}\boldsymbol{\beta} + \varepsilon$	$y = f(x, \boldsymbol{\beta}) + \varepsilon$
Objective	$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$	$\|\mathbf{y} - \mathbf{f}(\boldsymbol{\beta})\|^2$
Solution type	Analytical (Normal Equations)	Iterative (Gauss–Newton, LM)
Derivative requirement	Not required	Requires Jacobian $\mathbf{J}$
Computation cost	Low (Matrix inversion)	High (Multiple iterations)
Convergence	Exact if $\mathbf{X}^T\mathbf{X}$ invertible	Depends on initial guess
Error estimation	$(\mathbf{X}^T\mathbf{X})^{-1}\sigma^2$	$(\mathbf{J}^T\mathbf{J})^{-1}\sigma^2$

Table 3: Comparison between linear and nonlinear least squares formulations.

.5 Geometric Interpretation (Summary)

In both cases, least squares minimizes the distance between observed data and the model manifold:

- In the linear case, the model space is a flat subspace (a hyperplane) of $\mathbb{R}^n$.
- In the nonlinear case, the model space is a curved manifold; iterative algorithms project the data onto this manifold.