Numerical Computing

Lecture 6: Least Squares and QR Decomposition

Francisco Richter Mendoza

Università della Svizzera Italiana
Faculty of Informatics
Lugano, Switzerland

Today's Topics

- **Linear least squares** problem formulation
- **Normal equations** vs **QR decomposition**
- **Gram-Schmidt** and **Householder** algorithms
- **Numerical stability** analysis
- **Applications** to data fitting and overdetermined systems
- **Computational complexity** and practical considerations

Linear Least Squares Problem

Problem Statement

Given $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ and $b \in \mathbb{R}^m$:

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 = \min_x \sum_{i=1}^{m} (a_i^T x - b_i)^2$$

Geometric Interpretation

Find point Ax in column space of A closest to b in Euclidean norm.

Key insight: Overdetermined systems ($m > n$) generally have no exact solution.

Existence and Uniqueness

Theorem

The least squares problem $\min_x \|Ax - b\|_2$ always has a solution.
The solution is unique $\Leftrightarrow$ A has full column rank.

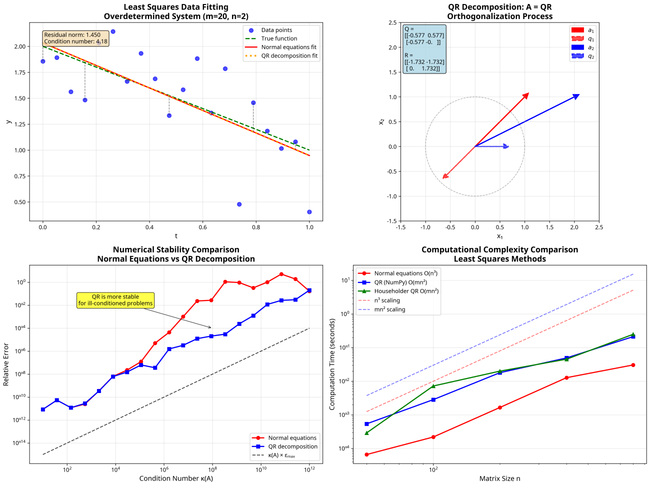
Optimality Condition

x^* is optimal $\Leftrightarrow$ residual $r^* = b - Ax^*$ is orthogonal to column space of A:

$$A^T r^* = A^T(b - Ax^*) = 0$$

Orthogonal projection: $Ax^* = P_A b$ where $P_A = A(A^T A)^{-1} A^T$

Least Squares: Geometric and Numerical Analysis



Visualization: Data fitting, QR vs normal equations stability, computational complexity, and conditioning effects.

Normal Equations Approach

Normal Equations

x^* minimizes $\|Ax - b\|_2$ if and only if:

$$A^T A x^* = A^T b$$

Algorithm

1. Form $A^T A$ and $A^T b$
2. Solve $(A^T A)x = A^T b$ (e.g., Cholesky if SPD)

Cost: $mn^2 + \frac{n^3}{3}$ operations
Problem: $\kappa(A^T A) = \kappa(A)^2$ (condition number squared!)

QR Decomposition

Definition

For $A \in \mathbb{R}^{m \times n}$ with $m \geq n$:

$$A = QR$$

where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times n}$ is upper triangular.

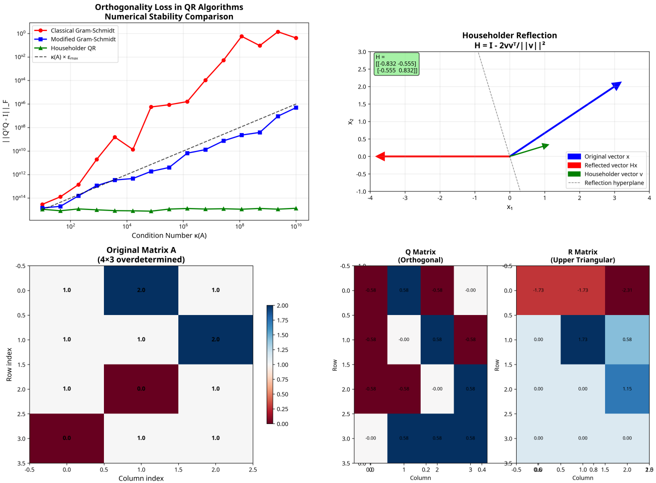
Reduced QR

$$A = Q_1 R_1$$

where $Q_1 \in \mathbb{R}^{m \times n}$ has orthonormal columns and $R_1 \in \mathbb{R}^{n \times n}$.

Key property: Orthogonal matrices preserve norms and conditioning.

QR Algorithms: Stability and Implementation



Comparison: Classical vs Modified Gram-Schmidt vs Householder reflections for numerical stability.

Gram-Schmidt Orthogonalization

Classical Gram-Schmidt

$$v_1 = a_1, \quad q_1 = \frac{v_1}{\|v_1\|} \tag{1}$$

$$v_k = a_k - \sum_{j=1}^{k-1} (q_j^T a_k) q_j, \quad q_k = \frac{v_k}{\|v_k\|} \tag{2}$$

Modified Gram-Schmidt

More stable: orthogonalize against q_j immediately after computing it.

Stability: Classical GS loses orthogonality for ill-conditioned matrices.

Cost: $2mn^2$ operations for both variants.

Householder Reflections

Definition

For nonzero $v \in \mathbb{R}^m$:

$$H = I - 2\frac{vv^T}{\|v\|^2}$$

Symmetric, orthogonal matrix that reflects across hyperplane $\perp v$.

QR via Householder

Choose $H_1, \ldots, H_n$ such that:

$$H_n \cdots H_1 A = R \quad \Rightarrow \quad A = QR \text{ with } Q = H_1 \cdots H_n$$

Advantage: Backward stable, optimal numerical properties.

Solving Least Squares via QR

QR Solution Method

If $A = QR$, then:

$$\|Ax - b\|_2 = \|QRx - b\|_2 = \|Rx - Q^T b\|_2$$

Minimum achieved when $Rx^* = Q^T b$.

Algorithm

1. Compute QR decomposition: $A = QR$
2. Compute $c = Q^T b$
3. Solve upper triangular system: $Rx = c$

Cost: $2mn^2 - \frac{2n^3}{3}$ operations (QR) + $O(n^2)$ (solve)

Numerical Stability Comparison

Condition Numbers

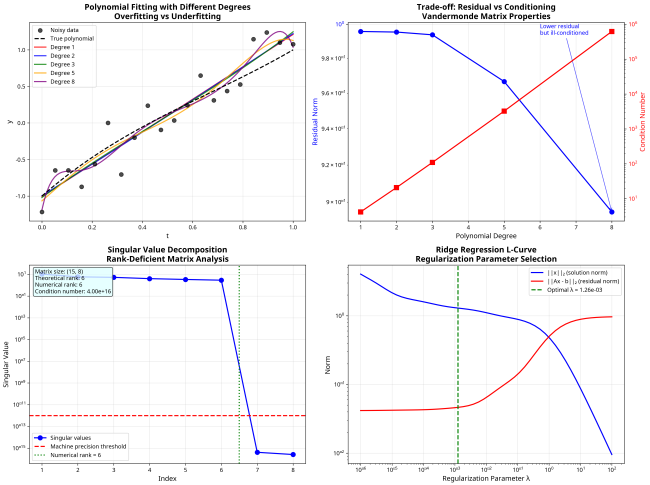
- Normal equations: $\kappa(A^T A) = \kappa(A)^2$
- QR decomposition: $\kappa(A)$ (preserved)

Backward Error Analysis

- Normal equations: $(A^T A + \Delta(A^T A))\tilde{x} = A^T b$
- QR decomposition: $(A + \Delta A)\tilde{x} = b$

Rule of thumb: Lose $\approx \log_{10}(\kappa)$ digits of accuracy.
QR advantage: Works directly with A, not $A^T A$.

Applications and Conditioning Effects



Applications: Polynomial fitting, rank-deficient problems, regularization, and L-curve analysis.

Polynomial Fitting Example

```python
# Polynomial fitting using QR decomposition
import numpy as np
from scipy.linalg import lstsq

# Generate data
t = np.linspace(0, 1, 20)
y = 1 - 2*t + 3*t**2 - t**3 + noise

# Create Vandermonde matrix for degree n
def fit_polynomial(t, y, degree):
    A = np.vander(t, degree + 1, increasing=True)
    coeffs = lstsq(A, y)[0]  # Uses QR internally
    return coeffs

# Fit different degrees
coeffs = fit_polynomial(t, y, degree=3)
```

Key insight: Higher degree $\Rightarrow$ higher condition number $\Rightarrow$ less stable fit.

Rank-Deficient Least Squares

Problem

When $\text{rank}(A) < n$, infinitely many solutions exist.

SVD Solution

$A = U\Sigma V^T$ gives minimum norm solution:

$$x^+ = V\Sigma^+ U^T b$$

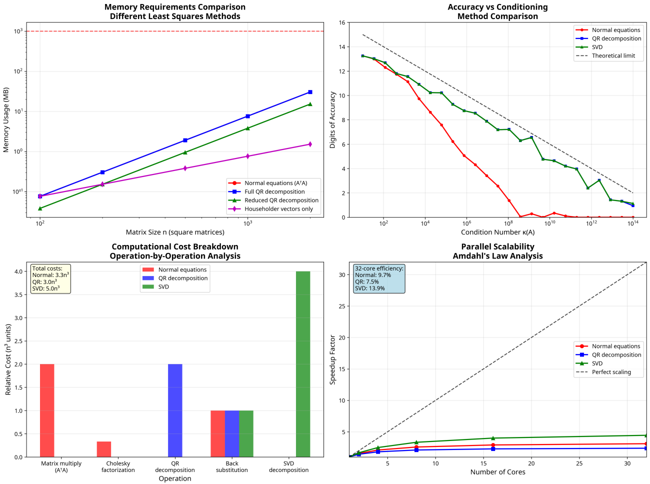
where Σ^+ is Moore-Penrose pseudoinverse.

Numerical Rank

Singular values $\sigma_i > \text{tol} \cdot \sigma_1$ determine effective rank.

Applications: Image reconstruction, data compression, regularization.

Computational Performance Analysis



Analysis: Memory usage, accuracy vs conditioning, cost breakdown, and parallel scalability.

Regularization: Ridge Regression

Ridge Problem

$$\min_x \|Ax - b\|_2^2 + \lambda \|x\|_2^2$$

Equivalent to solving augmented system:

$$\begin{pmatrix} A \\ \sqrt{\lambda} I \end{pmatrix} x = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

L-Curve Method

Plot $\|Ax - b\|_2$ vs $\|x\|_2$ for different λ.
Optimal λ at corner of L-shaped curve.

Effect: Trades off residual minimization vs solution smoothness.

When to Use Each Method

Normal Equations

Use when:

- A is well-conditioned ($\kappa(A) \lesssim 10^6$)
- Multiple right-hand sides
- $A^T A$ has special structure (sparse, banded)

QR Decomposition

Use when:

- A is ill-conditioned
- Numerical stability is critical
- General-purpose robust solver needed

SVD

Use when:

- Rank-deficient problems

Computational Complexity Summary

Operation Counts

- **Normal equations:** $mn^2 + \frac{n^3}{3}$ operations
- **QR (Householder):** $2mn^2 - \frac{2n^3}{3}$ operations
- **SVD:** $\approx 4mn^2 + 8n^3$ operations

Memory Requirements

- **Normal equations:** $O(n^2)$ (store A^TA)
- **QR:** $O(mn)$ (store Q, R)
- **Householder vectors:** $O(mn)$ (implicit Q)

Trade-off: Stability vs computational cost vs memory usage.

Key Takeaways

1. **Least squares** provides optimal solutions for overdetermined systems
2. **Normal equations** are fast but numerically unstable for ill-conditioned A
3. **QR decomposition** preserves conditioning and provides stable solutions
4. **Householder reflections** offer optimal backward stability
5. **Condition number** determines accuracy: lose $\approx \log_{10}(\kappa)$ digits
6. **Regularization** trades off fit quality vs solution smoothness

Next lecture: Iterative Methods for Linear Systems