GIAN Course on Solving Linear Systems and Computing Generalized Inverses Using Recurrent Neural Networks
June 09-19, 2025, IIT Indore,
(The Least Squares Problem and SVD)

Sk. Safique Ahmad

June 17, 2025

Second Step of the Algorithm

- The second step operates on the submatrix obtained by ignoring the first row and column.
- Otherwise, it is identical to the first step:
  - Compute and apply Householder reflector.
  - Identify pivot and possibly swap columns.
- When columns are interchanged, the **full columns** are swapped, not just the parts in the submatrix.
- This is equivalent to performing the interchange **before** the QR process starts.

Case: Matrix Has Full Rank

- If the matrix has full rank, the algorithm terminates after m steps.
- The result is a decomposition:

$$A\Pi = QR$$

- Where:
  - Π is a column permutation matrix,
  - $Q \in \mathbb{R}^{n \times n}$ is orthogonal,
  - $R \in \mathbb{R}^{n \times m}$ is upper triangular and nonsingular.

Case: Matrix Does Not Have Full Rank

- If A does not have full rank, at some step we will encounter $\tau_i = 0$.
- This occurs when all entries in the remaining submatrix are zero.
- Suppose this occurs after r steps.
- Let $Q_i \in \mathbb{R}^{n \times n}$ denote the reflector used at step i.

Structure of R and Reflectors

- Let $R_H \in \mathbb{R}^{r \times r}$ be the upper triangular part constructed from the first r steps.
- Then:

$$R = \begin{bmatrix} R_H & * \\ 0 & 0 \end{bmatrix}$$

- The diagonal entries of R_H are $-\tau_1, -\tau_2, \ldots, -\tau_r$, all nonzero.
- Clearly, $\text{rank}(R) = r$.

Final Form of the Decomposition

- Let:
$$Q = Q_1 Q_2 \cdots Q_r$$
- Then:
$$Q^T = Q_r Q_{r-1} \cdots Q_1$$
- Therefore:
$$Q^T A = R \quad \text{and} \quad A = QR$$
- Since $\text{rank}(A) = \text{rank}(R) = r$, we conclude:
$$\text{rank}(A) = r$$

Theorem 3.3.11

Theorem: Let $A \in \mathbb{R}^{n \times m}$ with rank $r > 0$. Then there exist matrices:

- $\Pi \in \mathbb{R}^{m \times m}$: a permutation matrix,
- $Q \in \mathbb{R}^{n \times n}$: orthogonal,
- $R \in \mathbb{R}^{n \times m}$: upper triangular,

such that:

$$A\Pi = QR$$

where:

$$R = \begin{bmatrix} R_H & * \\ 0 & 0 \end{bmatrix}, \quad R_H \in \mathbb{R}^{r \times r} \text{ is nonsingular.}$$

Least Squares Problem Setup

- Given $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$, we seek $x \in \mathbb{R}^m$ that minimizes:

$$\|Ax - b\|_2$$

- If A has **full column rank**, the solution is unique.
- If A is **rank-deficient** (i.e., $\text{rank}(A) = r < m$), the problem has **infinitely many solutions**.

QR Decomposition with Column Pivoting

- Apply QR with column pivoting: $A\Pi = QR$
- $Q \in \mathbb{R}^{n \times n}$: orthogonal
- $R = \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix}$, where:
  - $R_1 \in \mathbb{R}^{r \times r}$ is upper triangular and nonsingular,
  - $r = \text{rank}(A) < m$
- $\Pi \in \mathbb{R}^{m \times m}$: permutation matrix

Reduced Least Squares System

- Let $y = \Pi^T x$, then:

$$QRy \approx b \Rightarrow Ry \approx Q^T b$$

- Partition $y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$, where:

$$R = \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix}, \quad y_1 \in \mathbb{R}^r, \quad y_2 \in \mathbb{R}^{m-r}$$

- Solve:

$$R_1 y_1 = (Q^T b)_{1:r}$$

- y_2 is **free** (arbitrary) $\Rightarrow$ infinite solutions.

General Solution Form

- General solution to the least squares problem:

$$x = \Pi \begin{bmatrix} R_1^{-1}(Q^T b)_{1:r} \\ \text{free vector } y_2 \end{bmatrix}$$

- The solution set forms an affine subspace:

$$x = x_{\text{particular}} + \text{null}(A)$$

- The set of solutions is infinite due to $\dim(\text{null}(A)) = m - r > 0$

Minimum Norm Solution

- Among infinite solutions, one may choose the one with minimum $\|x\|_2$
- This is called the **minimum norm least squares solution**:

$$x_{\min} = A^{\dagger} b$$

  where $A^{\dagger}$ is the Moore–Penrose pseudoinverse.
- In QR terms:

$$x_{\min} = \Pi \begin{bmatrix} R_1^{-1} (Q^T b)_{1:r} \\ 0 \end{bmatrix}$$

MATLAB Code: Infinite Least Squares Solutions

```
% Rank-deficient matrix A and vector b
A = [1 2 3 4; 2 4 6 8; 3 6 9 12];  % rank 2
b = [1; 2; 3];

% QR decomposition with column pivoting
[Q, R, P] = qr(A, 'vector');

% Determine rank numerically
tol = max(size(A)) * eps(norm(R, 'fro'));
r = sum(abs(diag(R)) > tol);

% Solve R1 * y1 = Q^T * b (first r components)
R1 = R(1:r, 1:r);
Qt_b = Q' * b;
b1 = Qt_b(1:r);
y1 = R1 \ b1;
```

```
x = zeros(size(A,2), 1); x(P) = y;
disp('Minimum-norm solution:'); disp(x);
```

Classical Gram-Schmidt Algorithm

- Given linearly independent vectors $v_1, v_2, \ldots, v_m \in \mathbb{R}^n$
- Produces orthonormal vectors $q_1, q_2, \ldots, q_m$ such that:

$$\text{span}\{q_1, \ldots, q_i\} = \text{span}\{v_1, \ldots, v_i\}, \quad \text{for } i = 1, \ldots, m$$

- Algorithm:

$$\begin{aligned}
&q_1 = \frac{v_1}{\|v_1\|} \\
&\text{for } k = 2 \text{ to } m \\
&\quad \text{for } j = 1 \text{ to } k-1 \\
&\quad\quad r_{jk} = q_j^\top v_k \\
&\quad\quad v_k = v_k - r_{jk} q_j \\
&\quad \text{end} \\
&\quad r_{kk} = \|v_k\|, \quad q_k = \frac{v_k}{r_{kk}} \\
&\text{end}
\end{aligned}$$

Gram-Schmidt as QR Decomposition

- Let $A = [v_1, v_2, \ldots, v_m] \in \mathbb{R}^{n \times m}$
- The Gram-Schmidt process gives:

$$A = QR$$

where:

  - $Q = [q_1, q_2, \ldots, q_m]$ is orthonormal ($Q^\top Q = I$)
  - R is upper triangular with entries $r_{jk} = q_j^\top v_k$

- Each v_k can be written as:

$$v_k = \sum_{j=1}^{k} r_{jk} q_j$$

- In matrix form:

$$A = QR \quad \text{(Gram-Schmidt gives QR)}$$

MATLAB Code for Classical Gram-Schmidt

```
function [Q, R] = classical_gs(A)
    [n, m] = size(A);
    Q = zeros(n, m);
    R = zeros(m, m);

    for k = 1:m
        v = A(:,k);
        for j = 1:k-1
            R(j,k) = Q(:,j)' * A(:,k);
            v = v - R(j,k) * Q(:,j);
        end
        R(k,k) = norm(v);
        Q(:,k) = v / R(k,k);
    end
end
```

Summary

- Classical Gram-Schmidt orthogonalizes vectors sequentially.
- It is numerically unstable for nearly linearly dependent vectors.
- The resulting decomposition $A = QR$ links the process directly to matrix factorization.
- Modified Gram-Schmidt improves numerical stability.

Modified Gram-Schmidt Overview

- Modified Gram-Schmidt (MGS) is a numerically more stable variant of the classical method.
- Orthogonalizes column by column using updated vectors.
- Better handles near-linear dependence in columns.
- Produces $A = QR$ where:
  - Q: orthonormal columns
  - R: upper triangular matrix

MATLAB Code: Modified Gram-Schmidt

```
function [Q, R] = modified_gs(A)
    [n, m] = size(A);
    Q = zeros(n, m);
    R = zeros(m, m);
    V = A;

    for i = 1:m
        R(i,i) = norm(V(:,i));
        Q(:,i) = V(:,i) / R(i,i);
        for j = i+1:m
            R(i,j) = Q(:,i)' * V(:,j);
            V(:,j) = V(:,j) - R(i,j) * Q(:,i);
        end
    end
end
```

Example Matrix

- Let

$$A = \begin{bmatrix} 1 & 1 \\ 10^{-10} & 0 \\ 0 & 10^{-10} \end{bmatrix}$$

- Columns of A are nearly linearly dependent
- Classical Gram-Schmidt fails due to loss of orthogonality
- Modified Gram-Schmidt maintains orthogonality

Numerical Stability Comparison

- Compute $Q^\top Q$:
  - Classical GS: $Q^\top Q \neq I$ (orthogonality lost)
  - Modified GS: $Q^\top Q \approx I$
- Stability matters in ill-conditioned problems
- Use ‘norm(Q’*Q - eye(size(Q,2)))‘ in MATLAB to test

Remark

- Modified Gram-Schmidt is more stable than the classical version.
- Especially important when vectors are nearly linearly dependent.
- For numerical work, prefer MGS or Householder QR over classical GS.

Given Vectors

- $\mathbf{v}_1 = \begin{bmatrix} 3 \\ -3 \\ 3 \\ -3 \end{bmatrix}$
- $\mathbf{v}_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$
- Define $S = \text{span}\{\mathbf{v}_1, \mathbf{v}_2\} \subset \mathbb{R}^4$

Step (a): Gram-Schmidt Process

$$\mathbf{u}_1 = \mathbf{v}_1$$

$$r_{11} = \|\mathbf{u}_1\| = \sqrt{3^2 + (-3)^2 + 3^2 + (-3)^2} = \sqrt{36} = 6$$

$$\mathbf{q}_1 = \frac{\mathbf{u}_1}{r_{11}} = \frac{1}{6}\begin{bmatrix} 3 \\ -3 \\ 3 \\ -3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}$$

Continue Gram-Schmidt

$$
\begin{aligned}
r_{12} &= \mathbf{q}_1^T \mathbf{v}_2 = [0.5, -0.5, 0.5, -0.5] \cdot [1, 2, 3, 4]^T \\
&= 0.5(1) + (-0.5)(2) + 0.5(3) + (-0.5)(4) = -1
\end{aligned}
$$

$$
\mathbf{u}_2 = \mathbf{v}_2 - r_{12}\mathbf{q}_1 = \mathbf{v}_2 + \mathbf{q}_1 = \begin{bmatrix} 1.5 \\ 1.5 \\ 3.5 \\ 3.5 \end{bmatrix}
$$

$$
r_{22} = \|\mathbf{u}_2\| = \sqrt{1.5^2 + 1.5^2 + 3.5^2 + 3.5^2} = \sqrt{32} = 4\sqrt{2}
$$

$$
\mathbf{q}_2 = \frac{\mathbf{u}_2}{r_{22}} = \frac{1}{4\sqrt{2}} \begin{bmatrix} 1.5 \\ 1.5 \\ 3.5 \\ 3.5 \end{bmatrix}
$$

Step (b): Construct Q and R

$$Q = \begin{bmatrix} 0.5 & \frac{1.5}{4\sqrt{2}} \\ -0.5 & \frac{1.5}{4\sqrt{2}} \\ 0.5 & \frac{3.5}{4\sqrt{2}} \\ -0.5 & \frac{3.5}{4\sqrt{2}} \end{bmatrix}, \quad R = \begin{bmatrix} 6 & -1 \\ 0 & 4\sqrt{2} \end{bmatrix}$$

Then, $V = QR$

QR Decomposition Method Comparison

- Let $V \in \mathbb{R}^{30\times 20}$ with entries:

$$V(i,j) = \left(\frac{j}{20}\right)^{i-1}, \quad i = 1,\ldots,30,\ j = 1,\ldots,20$$

- Such matrices are called **Vandermonde matrices**.
- Highly ill-conditioned:

$$\kappa_2(V) \approx 3 \times 10^{13}$$

- Indicates columns are nearly linearly dependent.

Numerical Experiment

- Goal: Compare orthogonality of Q from QR decomposition methods.
- Metric: $\|I - Q^\top Q\|_2$
- IEEE double-precision unit roundoff: $u \approx 10^{-16}$
- Expect error for stable methods: $\approx \kappa(V) \cdot u \approx 3 \times 10^{-3}$

QR Method Comparison on Vandermonde Matrix

- Comparison of orthogonality error $\|I - Q^\top Q\|_2$:

Method	$\|I - Q^\top Q\|_2$
Classical Gram-Schmidt	12.4
Modified Gram-Schmidt	$\approx 3 \times 10^{-4}$
Householder QR (Reflectors)	$\approx 1.9 \times 10^{-15}$

Table: *

QR decomposition results for highly ill-conditioned Vandermonde matrix

Remark

- **Classical Gram-Schmidt:** fails in preserving orthogonality for ill-conditioned matrices.
- **Modified Gram-Schmidt:** more stable but still sensitive to ill-conditioning.
- **Householder QR:** highly stable, preferred in practice.
- Recommendation: **Use Householder QR** or SVD for high-accuracy applications.

Singular Value Decomposition (SVD), Moore Penrose Inverse

Bases and Matrices in the SVD

The Singular Value Decomposition is a highlight of linear algebra. A is any $m \times n$ matrix, square or rectangular. Its rank is r. We will diagonalize this A, but not by $X^{-1}AX$.

The eigenvectors in X have three big problems:

- They are usually not orthogonal,
- there are not always enough eigenvectors, and $Ax = \lambda x$ requires A to be a square matrix.
- The singular vectors of A solve all those problems in a perfect way.

Theorem 4.1.1 (SVD Theorem)

Let $A \in \mathbb{R}^{n \times m}$ be a nonzero matrix of rank r. Then:
Singular Value Decomposition (SVD) There exist orthogonal matrices:

$$U \in \mathbb{R}^{n \times n}, \quad V \in \mathbb{R}^{m \times m}$$

and a "diagonal" matrix:

$$\Sigma \in \mathbb{R}^{n \times m}$$

such that:

$$A = U \Sigma V^{\top}$$

Structure of the SVD

- $U = [\mathbf{u}_1\ \mathbf{u}_2\ \ldots\ \mathbf{u}_n]$ with $U^\top U = I_n$
- $V = [\mathbf{v}_1\ \mathbf{v}_2\ \ldots\ \mathbf{v}_m]$ with $V^\top V = I_m$
- Σ has the form:

$$\Sigma = \begin{bmatrix} \sigma_1 & & & & \\ & \sigma_2 & & & \\ & & \ddots & & \\ & & & \sigma_r & \\ & & & & \mathbf{0} \end{bmatrix}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ are the singular values of A.

Geometric Interpretation

- A maps the orthonormal basis vectors of $\mathbb{R}^m$ (columns of V) to scaled orthogonal vectors in $\mathbb{R}^n$ (columns of U).
- Each σ_i represents the stretching factor along the direction $\mathbf{v}_i$.
- The rank r of A equals the number of nonzero singular values.

Remark

- Every real matrix $A \in \mathbb{R}^{n \times m}$ has a Singular Value Decomposition.
- $A = U\Sigma V^\top$, where:
  - $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{m \times m}$ are orthogonal,
  - $\Sigma \in \mathbb{R}^{n \times m}$ is diagonal with singular values.
- The SVD is a fundamental tool in numerical linear algebra, data compression, and PCA.

Theorem (Geometric SVD Theorem)

Let $A \in \mathbb{R}^{n \times m}$ be a nonzero matrix of rank r. Then:

- There exists an orthonormal basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ of $\mathbb{R}^m$
- And an orthonormal basis $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ of $\mathbb{R}^n$
- And singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$

Such that:

$$A\mathbf{v}_i = \sigma_i \mathbf{u}_i \quad \text{for } i = 1, \ldots, r$$

$$A\mathbf{v}_i = \mathbf{0} \quad \text{for } i = r+1, \ldots, m$$

$$A^T\mathbf{u}_i = \sigma_i \mathbf{v}_i \quad \text{for } i = 1, \ldots, r$$

$$A^T\mathbf{u}_i = \mathbf{0} \quad \text{for } i = r+1, \ldots, n$$

Geometric Interpretation

- A maps the unit vectors $\mathbf{v}_i$ in $\mathbb{R}^m$ to scaled orthogonal vectors $\sigma_i \mathbf{u}_i$ in $\mathbb{R}^n$.
- The first r directions are scaled by $\sigma_i > 0$, and the rest are mapped to 0.
- The image of the unit sphere in $\mathbb{R}^m$ under A is a hyperellipse in $\mathbb{R}^n$.

Exercise 4.1.5: From Algebraic SVD to Geometric SVD

Let $A = U\Sigma V^\top$ be the SVD of A, where:

- Columns of $V = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$
- Columns of $U = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$

Then:

$$A\mathbf{v}_i = U\Sigma V^\top \mathbf{v}_i = \sigma_i \mathbf{u}_i \quad (i = 1, \ldots, r)$$

- The matrix multiplication $AV = U\Sigma$ implies that each $\mathbf{v}_i$ is mapped to $\sigma_i \mathbf{u}_i$.
- When $\sigma_i = 0$, $A\mathbf{v}_i = \mathbf{0}$.

Thus, the geometric form follows directly from the standard SVD expression.

SVD and the Four Fundamental Subspaces

The SVD provides orthonormal bases for the fundamental subspaces of $A \in \mathbb{R}^{n \times m}$.

- $\mathcal{R}(A) = \text{Col}(A) = \text{span}\{\mathbf{u}_1, \ldots, \mathbf{u}_r\} \subseteq \mathbb{R}^n$
- $\mathcal{N}(A) = \text{Null}(A) = \text{span}\{\mathbf{v}_{r+1}, \ldots, \mathbf{v}_m\} \subseteq \mathbb{R}^m$
- $\mathcal{R}(A^\top) = \text{Row}(A) = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_r\} \subseteq \mathbb{R}^m$
- $\mathcal{N}(A^\top) = \text{Left Null Space} = \text{span}\{\mathbf{u}_{r+1}, \ldots, \mathbf{u}_n\} \subseteq \mathbb{R}^n$

These follow directly from the SVD:

$$A = U\Sigma V^\top$$

Corollary: Rank-Nullity Relation

Corollary 4.1.9 Let $A \in \mathbb{R}^{n \times m}$. Then:

$$\dim(\mathcal{R}(A)) + \dim(\mathcal{N}(A)) = m$$

- That is, the sum of the dimensions of the column space and the null space equals the number of columns.
- This result, also known as the **Rank-Nullity Theorem**, follows from the orthogonality and completeness of the columns of $V \in \mathbb{R}^{m \times m}$.
- Similarly, $\dim(\mathcal{R}(A^\top)) + \dim(\mathcal{N}(A^\top)) = n$.

(a) Structure of Rank-1 Matrix

Proof: Let $A \in \mathbb{R}^{n \times m}$ have rank 1.

- Then all columns of A lie in $\text{Range}(A) = \text{span}(\mathbf{u}_1)$.
- Choose $\|\mathbf{u}_1\| = 1$, then $A = \mathbf{u}_1 \mathbf{w}^T$ for some $\mathbf{w} \in \mathbb{R}^m$.
- Define $\mathbf{v}_1 = \frac{\mathbf{w}}{\|\mathbf{w}\|}$, $\sigma_1 = \|\mathbf{w}\|$, so:

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$$

(b) Orthonormal Extension

- Extend $\mathbf{u}_1$ to an orthonormal basis of $\mathbb{R}^n$: $U = [\mathbf{u}_1 \ \cdots] \in \mathbb{R}^{n \times n}$
- Similarly, extend $\mathbf{v}_1$ to $V = [\mathbf{v}_1 \ \cdots] \in \mathbb{R}^{m \times m}$
- Define:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \end{bmatrix}_{n \times m} \Rightarrow A = U \Sigma V^T$$

- This gives the SVD of a rank-1 matrix.

(c) Leading Singular Value and Vector

Let $A \in \mathbb{R}^{n \times m}$, $\text{rank}(A) = r > 1$

- Let $\mathbf{v}_1$ maximize $\|A\mathbf{v}\|_2$ over unit vectors.
- Then $\mathbf{u}_1 = \frac{A\mathbf{v}_1}{\|A\mathbf{v}_1\|}$, and define:

$$\sigma_1 = \|A\mathbf{v}_1\| = \|A\|_2$$

- Let $U = [\mathbf{u}_1 \ \cdots]$, $V = [\mathbf{v}_1 \ \cdots]$, define:

$$B = U^T AV = \begin{bmatrix} \sigma_1 & \mathbf{z}^T \\ \mathbf{0} & A_1 \end{bmatrix} \Rightarrow A = UBV^T$$

(d) Showing $\mathbf{z} = \mathbf{0}$

- Suppose $B = \begin{bmatrix} \sigma_1 & \mathbf{z}^T \\ 0 & A_1 \end{bmatrix}$
- Take $\mathbf{x} = \begin{bmatrix} \cos\theta \\ \sin\theta \mathbf{w} \end{bmatrix}$, $\|\mathbf{w}\| = 1$
- Then $\|B\mathbf{x}\|^2 = \sigma_1^2 \cos^2\theta + \|\mathbf{z}^T\mathbf{w}\|^2 \sin^2\theta + \|A_1\mathbf{w}\|^2 \sin^2\theta$
- Since σ_1 is the largest singular value, optimization implies $\mathbf{z} = 0$

(e) Completing the SVD Inductively

- Since $\mathbf{z} = 0$, $B = \begin{bmatrix} \sigma_1 & 0 \\ 0 & A_1 \end{bmatrix}$
- $\text{rank}(A_1) = r - 1$. By induction, SVD of $A_1 = U_1 \Sigma_1 V_1^T$
- Embed into full SVD of A as:

$$A = U \begin{bmatrix} \sigma_1 & 0 \\ 0 & \Sigma_1 \end{bmatrix} V^T$$

- This gives an SVD for general $A \in \mathbb{R}^{n \times m}$ of rank r.

Theorem 4.1.10: Condensed SVD

Condensed Singular Value Decomposition (SVD)
Let $A \in \mathbb{R}^{n \times m}$ be a nonzero matrix of rank r. Then:
Condensed SVD Form There exist:

- $U_r \in \mathbb{R}^{n \times r}$ with orthonormal columns ($U_r^\top U_r = I_r$)
- $V_r \in \mathbb{R}^{m \times r}$ with orthonormal columns ($V_r^\top V_r = I_r$)
- A diagonal matrix $\Sigma_r \in \mathbb{R}^{r \times r}$ with positive entries $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$

such that:

$$A = U_r \Sigma_r V_r^\top$$

Exercise: Proving the Condensed SVD

- Start from the full SVD:

$$A = U\Sigma V^\top$$

where $U \in \mathbb{R}^{n\times n}$, $\Sigma \in \mathbb{R}^{n\times m}$, $V \in \mathbb{R}^{m\times m}$

- Partition as:

$$U = [U_r\ \bar{U}], \quad \Sigma = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}, \quad V = [V_r\ \bar{V}]$$

- Then:

$$A = U\Sigma V^\top = U_r \Sigma_r V_r^\top$$

by removing the zero blocks.

Example: Compute the SVD of $A = \begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix}$

The rank is $r = 2$, so A has two positive singular values σ_1 and σ_2. We will find:
- $\sigma_1 > \lambda_{\max} = 5$ - $\sigma_2 < \lambda_{\min} = 3$
Begin by computing $A^T A$ and AA^T:

$$A^T A = \begin{bmatrix} 3 & 4 \\ 0 & 5 \end{bmatrix}^T \begin{bmatrix} 3 & 0 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix}, \quad AA^T = \begin{bmatrix} 9 & 12 \\ 12 & 41 \end{bmatrix}$$

Both matrices have the same trace (50) and determinant (225). Their eigenvalues are:

$$\sigma_1^2 = 45, \quad \sigma_2^2 = 5 \Rightarrow \sigma_1 = \sqrt{45}, \quad \sigma_2 = \sqrt{5}$$

Then $\sigma_1\sigma_2 = 15$, which is the determinant of A.
Now, we find the eigenvectors of A^TA:
For $\sigma_1^2 = 45$:

$$\begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 45 \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

For $\sigma_2^2 = 5$:

$$\begin{bmatrix} 25 & 20 \\ 20 & 25 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 5 \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Rightarrow v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Then we compute $u_i = \frac{Av_i}{\sigma_i}$ to get the columns of U.
This gives the full SVD: $A = U\Sigma V^T$.
The right singular vectors are:

$$v_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}, \quad v_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$$

Now compute:

$$Av_1 = A \cdot \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}3\\4\end{bmatrix} + \frac{1}{\sqrt{2}}\begin{bmatrix}0\\5\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}3\\9\end{bmatrix} = \sqrt{45} \cdot \frac{1}{\sqrt{10}}\begin{bmatrix}1\\3\end{bmatrix} = \sigma_1 u_1$$

$$Av_2 = A \cdot \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}3\\4\end{bmatrix} - \frac{1}{\sqrt{2}}\begin{bmatrix}0\\5\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}3\\-1\end{bmatrix} = \sqrt{5} \cdot \frac{1}{\sqrt{10}}\begin{bmatrix}3\\-1\end{bmatrix} = \sigma_2 u_2$$

This gives:

$$u_1 = \frac{1}{\sqrt{10}} \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \quad u_2 = \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

The singular value decomposition is:

$$A = U\Sigma V^T$$

Where:

$$U = \frac{1}{\sqrt{10}} \begin{bmatrix} 1 & 3 \\ 3 & -1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \sqrt{45} & 0 \\ 0 & \sqrt{5} \end{bmatrix}, \quad V = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{7}$$

U and V contain orthonormal bases for the column space and row space of A. These bases diagonalize A:

$$AV = U\Sigma \Rightarrow U^T AV = \Sigma$$

A as a Sum of Rank-One Matrices

$$\sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T = \sqrt{45} \cdot \frac{1}{\sqrt{10}} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \end{bmatrix} + \sqrt{5} \cdot \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$= \frac{\sqrt{45}}{\sqrt{20}} \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} + \frac{\sqrt{5}}{\sqrt{20}} \begin{bmatrix} 3 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} = A$$

Consider:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Observations:

- All eigenvalues of A are 0.
- Only one eigenvector: $(1, 0, 0, 0)^T$.
- Singular values: $\sigma = 3, 2, 1, 0$
- Singular vectors are columns of the identity matrix.

This example shows how the SVD provides much more structural insight than the eigen-decomposition, especially for non-symmetric or defective matrices.

SVD Setup

Let $A \in \mathbb{R}^{n \times m}$ with rank r, and let:

$$A = U \Sigma V^T$$

where:

- $U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{m \times m}$ are orthogonal,
- $\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n \times m}$
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$

Transforming the Problem

We want to solve the least squares problem:

$$\min_x \|Ax - b\|_2$$

Using the orthogonality of U and V, let:

$$c = U^T b, \quad y = V^T x$$

Then:

$$\|Ax - b\|_2 = \|U\Sigma V^T x - b\|_2 = \|\Sigma y - c\|_2$$

Minimizing the Residual

The residual becomes:

$$\|\Sigma y - c\|_2^2 = \sum_{i=1}^{r} (\sigma_i y_i - c_i)^2 + \sum_{i=r+1}^{n} c_i^2$$

This is minimized when:

$$y_i = \frac{c_i}{\sigma_i}, \quad i = 1, \ldots, r$$

$y_{r+1}, \ldots, y_m$ arbitrary (do not affect residual)

Minimum Norm Solution

To find the solution x with minimal $\|x\|_2$, we must minimize $\|y\|_2$.
This is achieved when:

$$y_{r+1} = \cdots = y_m = 0$$

Hence, the minimum-norm least-squares solution is:

$$x = Vy = \sum_{i=1}^{r} \frac{c_i}{\sigma_i} \mathbf{v}_i$$

or equivalently:

$$x = A^+ b$$

where $A^+ = V\Sigma^+ U^T$ is the Moore-Penrose pseudoinverse.

Moore-Penrose Pseudoinverse

Let $A \in \mathbb{R}^{n \times m}$ be a matrix of rank r, with SVD:

$$A = U \Sigma V^T$$

Then the Moore-Penrose pseudoinverse $A^\dagger$ is given by:

$$A^\dagger = V \Sigma^\dagger U^T$$

where $\Sigma^\dagger \in \mathbb{R}^{m \times n}$ is formed by:

$$\Sigma^\dagger = \begin{bmatrix} 1/\sigma_1 & & & \\ & \ddots & & \\ & & 1/\sigma_r & \\ & & & \mathbf{0} \end{bmatrix}$$

with $\sigma_1, \ldots, \sigma_r > 0$ the nonzero singular values of A.

Exercise– Matrix Form of Pseudoinverse

Given the full SVD of A:

$$A = U\Sigma V^T$$

Then:

$$A^\dagger U = V\Sigma^\dagger$$

Because U is orthogonal:

$$A^\dagger = V\Sigma^\dagger U^T$$

This representation is exact and satisfies all four Moore-Penrose conditions.

Condensed Form of the SVD

Let $U_r \in \mathbb{R}^{n \times r}$, $V_r \in \mathbb{R}^{m \times r}$ denote the first r columns of U and V, and $\Sigma_r \in \mathbb{R}^{r \times r}$ the diagonal matrix of nonzero singular values.
Then:

$$A = U_r \Sigma_r V_r^T$$

$$A^\dagger = V_r \Sigma_r^{-1} U_r^T$$

This form is efficient and commonly used in numerical computation.

Why Use the Pseudoinverse?

- Solves the least-squares problem:

$$\min_x \|Ax - b\|_2 \Rightarrow x = A^\dagger b$$

- Works even when A is not full-rank.
- Provides the minimum-norm solution when there are infinitely many.
- The pseudoinverse is essential in data fitting, control theory, and machine learning.

Exercise: Moore-Penrose Characterization

Theorem: Let $A \in \mathbb{R}^{n \times m}$, and let $B \in \mathbb{R}^{m \times n}$. Then $B = A^\dagger$ if and only if:

$$
\begin{aligned}
&(1) \quad ABA = A \\
&(2) \quad BAB = B \\
&(3) \quad (AB)^T = AB \\
&(4) \quad (BA)^T = BA
\end{aligned}
$$

Proof Outline:

- If $B = A^\dagger$ from SVD, all four properties hold.
- Conversely, any matrix B satisfying these four conditions must be $A^\dagger$.

Setup: SVD of A

Let $A \in \mathbb{R}^{n \times m}$ with full column rank m, and let

$$A = U \Sigma V^T$$

be the singular value decomposition, where:

- $U \in \mathbb{R}^{n \times n}$, $U^T U = I_n$
- $V \in \mathbb{R}^{m \times m}$, $V^T V = I_m$
- $\Sigma = \begin{bmatrix} \text{diag}(\sigma_1, \ldots, \sigma_m) \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n \times m}$

SVDs of Related Matrices

Matrix	**SVD Expression**	**Singular Values**
$A^T A$	$V\Sigma^T \Sigma V^T$	σ_i^2
$(A^T A)^{-1}$	$V(\Sigma^T \Sigma)^{-1} V^T$	$1/\sigma_i^2$
$(A^T A)^{-1} A^T$	$V\Sigma^{-1} U^T$	$1/\sigma_i$
$A(A^T A)^{-1}$	$U\Sigma^{-1} V^T$	$1/\sigma_i$

Observations

- $A^T A$ and $(A^T A)^{-1}$ are symmetric and positive definite.
- $(A^T A)^{-1} A^T = A^\dagger$: the Moore-Penrose pseudoinverse of A.
- $A(A^T A)^{-1}$ is the pseudoinverse of A^T.
- All SVDs use the same orthogonal matrices U and V, but scale differently.

Diagonal Structure of A^TA and AA^T

We always start with A^TA and AA^T. They are diagonal (with easy v's and u's):

$$A^TA = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 9 \end{bmatrix}, \quad AA^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The eigenvalues of A^TA (and AA^T) are $\sigma^2 = 9, 4, 1$ (nonzero), corresponding to singular values $\sigma_1 = 3$, $\sigma_2 = 2$, $\sigma_3 = 1$.

Their corresponding orthonormal eigenvectors (in order of decreasing singular values) are:

$$U = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The first columns u_1 and v_1 have 1's in positions 3 and 4. Then the matrix $u_1 \sigma_1 v_1^T$ picks out the largest number in A, which is $A_{3,4} = 3$.
Thus the SVD of A is:

$$A = U\Sigma V^T = 3u_1 v_1^T + 2u_2 v_2^T + 1u_3 v_3^T$$

Effect of Removing a Zero Row

Suppose we remove the last row of A (which is entirely zeros). Then A becomes a 3×4 matrix and AA^T becomes 3×3. Its fourth row and column disappear.
However, the eigenvalues of A^TA and AA^T remain the same: $\lambda = 1, 4, 9$, so the singular values are still $\sigma = 3, 2, 1$. We just remove the last row of Σ, and the last row and column of U:

$$A_{3\times 4} = U_{3\times 3}\Sigma_{3\times 4}V^T_{4\times 4}$$

The SVD naturally accommodates rectangular matrices.

Stability of Singular Values vs. Eigenvalue Instability

The 4×4 matrix A provides a powerful illustration of the instability of eigenvalues. Suppose the $(4,1)$ entry of A is changed slightly—from 0 to $\frac{1}{60{,}000}$.
Consider the matrix:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \\ \frac{1}{60{,}000} & 0 & 0 & 0 \end{bmatrix}$$

This small change in the $(4,1)$ entry (only $1/60{,}000$) creates a much larger effect in the eigenvalues of A. Originally, with a zero in the $(4,1)$ position, the eigenvalues of A were all zero:

$$\lambda = 0, 0, 0, 0$$

After the change, the eigenvalues move to four points on a circle in the complex plane centered at the origin, with radius $\frac{1}{10}$:

At the other extreme, when $A^T A = AA^T$ (i.e., A is a **normal matrix**), the eigenvectors of A are orthogonal, and the eigenvalues are completely stable.
Singular Values Are Stable.
By contrast, the singular values of A remain stable under small perturbations. In this case, the new singular values are:

$$\sigma = 3, 2, 1, \frac{1}{60{,}000}$$

The singular vectors (U and V) remain essentially unchanged. The fourth piece of the SVD is:

$$\sigma_4 u_4 v_4^T = \frac{1}{60{,}000} u_4 v_4^T$$

—mostly zeros, except for the new small entry.

Singular Vectors of A and Eigenvectors of $S = A^T A$

Equations (5)–(6) showed that the right singular vectors v_i of A are eigenvectors q_i of $S = A^T A$. The eigenvalues λ_i of S are exactly σ_i^2, where σ_i are the singular values of A. The rank r of S equals the rank of A.

The SVD produces beautiful, parallel, orthonormal sets:

- $\{q_i\}$: orthonormal eigenvectors of $S = A^T A$
- $\{v_i\}$: right singular vectors of A (equal to q_i)
- $\{u_i\}$: left singular vectors of A

Motivation for Revisiting the SVD Derivation.

We revisit this to address two reasons:

1. If λ is a repeated eigenvalue of S, we must find two orthonormal eigenvectors corresponding to it.
2. We want to understand how SVD successively picks off rank-one terms: $\sigma_1 u_1 v_1^T$, then $\sigma_2 u_2 v_2^T$, etc.—each ordered by importance.

Variational Characterization of λ_1 and σ_1

Largest eigenvalue λ_1 of $S = A^T A$:

$$\lambda_1 = \max_{\|x\|=1} x^T S x = \max_{\|x\|=1} x^T A^T A x = \max_{\|x\|=1} \|Ax\|^2$$

The maximizer is the eigenvector $x = q_1$ (also v_1), and $Sq_1 = \lambda_1 q_1$.
Largest singular value σ_1 of A:

$$\sigma_1 = \max_{\|x\|=1} \|Ax\| = \sqrt{\lambda_1}$$

Again, the maximizing vector is $x = v_1$, and $Av_1 = \sigma_1 u_1$.
This is how SVD identifies the dominant direction in which A acts—picking the vector v_1 along which A stretches the most.

One-at-a-Time Approach for λ_2 and σ_2

The same variational approach used to find λ_1 and σ_1 also applies to the second eigenvalue and singular value:

$$\lambda_2 = \max_{\substack{x \perp q_1 \\ x \neq 0}} \frac{x^T S x}{x^T x} \quad \text{(winning } x \text{ is } q_2\text{)} \tag{10}$$

$$\sigma_2 = \max_{\substack{x \perp v_1 \\ x \neq 0}} \frac{\|Ax\|}{\|x\|} \quad \text{(winning } x \text{ is } v_2\text{)} \tag{11}$$

When $S = A^T A$, we find that $\lambda_1 = \sigma_1^2$ and $\lambda_2 = \sigma_2^2$. Why does this strategy succeed?

Rayleigh Quotient and the Eigenvalue Problem

Begin with the Rayleigh quotient:

$$r(x) = \frac{x^T S x}{x^T x}$$

To find the x that maximizes $r(x)$, we compute the gradient of $r(x)$ and set the derivatives $\frac{\partial r}{\partial x_i} = 0$. Though messy, the result is a clean vector equation:

$$Sx = r(x)x \tag{12}$$

This means the maximizing x is an eigenvector of S, and $r(x)$ reaches its maximum at the largest eigenvalue λ_1.

Connection to Singular Values

Now observe that:

$$\|Ax\| = \sqrt{x^T A^T A x} = \sqrt{x^T S x}$$

So maximizing $\|Ax\|/\|x\|$ is the same as maximizing $\sqrt{x^T S x / x^T x}$. Therefore, the maximizer $x = v_1$ from equation (9) is the same as q_1 from (8), and $\sigma_1 = \sqrt{\lambda_1}$.

Orthogonality and the Search for q_2

We now explain why q_2 (and v_2) are the solutions to (10) and (11). Because they are orthogonal to q_1 (and v_1), they lie in the subspace orthogonal to the top singular vector. Choose any orthogonal matrix Q_1 such that:

$$Q_1 = [q_1 \; q_2 \; \dots \; q_n]$$

where the columns $q_2, \dots, q_n$ are orthonormal and orthogonal to q_1. Then:

$$SQ_1 = \begin{bmatrix} Sq_1 & Sq_2 & \dots & Sq_n \end{bmatrix} = \begin{bmatrix} \lambda_1 q_1 & Sq_2 & \dots & Sq_n \end{bmatrix}$$

Representing S in this new basis:

$$Q_1^T S Q_1 = \begin{bmatrix} \lambda_1 & \mathbf{w}^T \\ \mathbf{w} & S_{n-1} \end{bmatrix} \tag{13}$$

Because S is symmetric, so is $Q_1^T S Q_1$, which forces $\mathbf{w} = 0$, and thus:

$$Q_1^T S Q_1 = \begin{bmatrix} \lambda_1 & 0 \\ 0 & S_{n-1} \end{bmatrix}$$

This isolates S_{n-1} as a smaller symmetric matrix (dimension $n-1$), in which we can now solve for its top eigenvalue λ_2 and eigenvector q_2.

By Induction or Recursion

We can now repeat this process:

ameChoose x orthogonal to $q_1, q_2, \ldots, q_{k-1}$Maximize $x^T S x / x^T x$The result is q_k, the k^{th} eigenvector, and the maximum value is λ_k

This iterative or inductive construction yields the complete orthonormal eigenbasis $\{q_1, \ldots, q_n\}$ and eigenvalues $\{\lambda_1, \ldots, \lambda_n\}$, proving the **Spectral Theorem**: every symmetric matrix S can be diagonalized as:

$$S = Q \Lambda Q^T$$

The Same Logic for SVD

Exactly the same reasoning applies to the SVD of A. Once v_1 is found (the direction that A stretches the most), we constrain our search for the next singular vector to the subspace orthogonal to v_1, then v_2, and so on—just like in the eigenvalue case.
This recursive procedure gives the complete SVD:

$$A = U\Sigma V^T$$

Each piece $\sigma_i u_i v_i^T$ corresponds to a direction and strength of action by A, building the full picture one direction at a time.

Next: Computing the Singular Values and Vectors

In the next section, we'll ask: How are the λ_i and σ_i actually computed in practice? And how does this relate to the **geometry of an ellipse** and the **compression of images** by SVD?

Computing the Eigenvalues of S and the SVD of A

The singular values σ_i of A are the square roots of the eigenvalues λ_i of $S = A^T A$.

This crucial connection ties the Singular Value Decomposition (SVD) directly to the symmetric eigenvalue problem—a good thing, because symmetric matrices are stable and well-behaved.

Avoiding the Cost of Squaring

Even though $S = A^T A$ gives us the λ_i, computing S explicitly can be expensive and numerically unstable (squaring magnifies errors). So instead of forming S directly, we work with A using techniques that preserve its singular values.

Orthogonal Similarity: Tridiagonal Form for S

The first idea is to reduce S to a simpler matrix with the same eigenvalues. This is done by a similarity transformation:

$$S' = Q^{-1}SQ = Q^T SQ \quad \text{(when } Q \text{ is orthogonal)}$$

Since $Q^T = Q^{-1}$, this transformation preserves eigenvalues, and because S is symmetric, S' is also symmetric. In particular, we aim to choose Q so that $Q^T SQ$ is tridiagonal—a symmetric matrix with nonzero entries only on the main diagonal and the two adjacent diagonals. Section 11.3 discusses how to construct such a Q using a series of 2×2 Givens rotations or Householder reflections.

Bidiagonalization for A: Two Orthogonal Matrices

What is the SVD analog of this? For A, we do not want to change its singular values σ_i. We use two orthogonal matrices, Q_1 and Q_2, such that:

$$A' = Q_1^T A Q_2$$

Then:

$$(A')^T A' = (Q_1^T A Q_2)^T (Q_1^T A Q_2) = Q_2^T A^T Q_1 Q_1^T A Q_2 = Q_2^T S Q_2$$

Since Q_2 is orthogonal, the eigenvalues of S and $Q_2^T S Q_2$ are the same. So the singular values of A remain unchanged. The big advantage here is that we can reduce A to a **bidiagonal matrix**—nonzero entries only on the main diagonal and the first superdiagonal.

$$Q_1^T A Q_2 = \text{bidiagonal matrix}$$

This step is a core part of practical SVD algorithms. It reduces the complexity of the problem while preserving the structure we care about (the σ_i).

Bidiagonal vs. Tridiagonal: A Neat Connection

This reduction beautifully mirrors the process for symmetric matrices:

$$(\text{bidiagonal})^T \cdot (\text{bidiagonal}) = \text{tridiagonal}$$

So reducing A to bidiagonal form gives $A^T A$ in tridiagonal form—another instance of the strong relationship between the SVD and the symmetric eigenvalue problem.

Final Step: Diagonalizing

The last step is to diagonalize these reduced matrices—transform the tridiagonal (for S) or bidiagonal (for A) matrix into a fully diagonal form Λ or Σ. This requires more advanced iterative techniques, including:

- QR algorithm (for eigenvalues)
- Divide-and-conquer methods
- Implicitly shifted QR for faster convergence

The difficulty of this problem is rooted in solving the characteristic polynomial:

$$\det(S - \lambda I) = 0$$

This is a degree-n polynomial, where n might be in the hundreds or thousands. No closed-form solutions exist for large n, but modern algorithms approach the diagonal form iteratively and efficiently.

Built-In Commands

All of this is hidden inside simple commands in most numerical software:

- `eig(S)` finds the eigenvalues λ_i
- `svd(A)` computes the full singular value decomposition $A = U\Sigma V^T$

These functions internally use orthogonal transformations to reduce A or S to bidiagonal or tridiagonal form, then iterate toward the diagonal.

Remark

While we often think of SVD as a mysterious black box, the path to computing it is highly structured:

1. Reduce A to bidiagonal form using orthogonal matrices.
2. Use iterative methods to diagonalize the bidiagonal matrix.

The connection to the symmetric eigenvalue problem through $S = A^T A$ is central, and the stability of singular values—unlike the potential instability of eigenvalues—makes SVD a powerful and reliable tool for numerical linear algebra.

Review of the Key Ideas

1. The SVD factors a matrix A into $U\Sigma V^T$, with r singular values $\sigma_1 \geq \cdots \geq \sigma_r > 0$.
2. The squared singular values $\sigma_1^2, \ldots, \sigma_r^2$ are the nonzero eigenvalues of both AA^T and A^TA.
3. The orthonormal columns of U and V are the eigenvectors of AA^T and A^TA, respectively.
4. These columns form orthonormal bases for the four fundamental subspaces of A:
   - Col(A) and Null(A^T) from U
   - Row(A) and Null(A) from V
5. These bases diagonalize the matrix: $Av_i = \sigma_i u_i$ for $i \leq r$, which gives $AV = U\Sigma$.
6. The SVD expresses A as a sum of rank-one matrices:

$$A = \sigma_1 u_1 v_1^T + \cdots + \sigma_r u_r v_r^T$$

where σ_1 is the maximum of the ratio:

$$\max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

Worked Examples

1. Orthogonal columns
2. Orthonormal columns
3. Triangular columns

Where do the rank, the pivots, and the singular values of A come into this picture?
Solution: These three factorizations are basic to linear algebra, pure or applied:

1. Singular Value Decomposition: $A = U\Sigma V^T$
2. Gram-Schmidt Orthogonalization: $A = QR$
3. Gaussian Elimination: $A = LU$

You might prefer to separate out singular values σ_i, heights h_i, and pivots d_i:

1. $A = U\Sigma V^T$ with unit vectors in U and V. The singular values σ_i are in Σ.
2. $A = QHR$ with unit vectors in Q and diagonal 1's in R. The heights h_i are in H.
3. $A = LDU$ with diagonal 1's in L and U. The pivots d_i are in D.

Each h_i tells the height of column i above the plane of columns 1 to $i-1$. The volume of the full n-dimensional box (where $r = m = n$) comes from $A = U\Sigma V^T = LDU = QHR$:

$$|\det A| = |\text{product of } \sigma_i| = |\text{product of } d_i| = |\text{product of } h_i|$$

Solution: Start from the SVD: $A = U\Sigma V^T$. Remember that multiplying by an orthogonal matrix does not change length:

$$\|Qx\| = \|x\| \quad \text{because} \quad \|Qx\|^2 = x^T Q^T Q x = x^T x = \|x\|^2.$$

This applies to $Q = U$ and $Q = V^T$. In between is the diagonal matrix Σ.
For Ax:

$$\|Ax\| = \|U\Sigma V^T x\| = \|\Sigma V^T x\| \leq \sigma_1 \|V^T x\| = \sigma_1 \|x\|.$$

An eigenvector has $Ax = \lambda x$. So, the above inequality implies:

$$|\lambda|\|x\| \leq \sigma_1\|x\|.$$

Therefore, $|\lambda| \leq \sigma_1$.
Also, apply this to the unit vector $x = (1, 0, \ldots, 0)$. Now Ax is the first column of A. Then by the inequality, this column has length $\leq \sigma_1$. Every entry must have $|a_{ij}| \leq \sigma_1$.

Thus, equation (14) shows that the maximum value of $\frac{\|Ax\|}{\|x\|}$ equals σ_1.

Section 11.2 will explain how the ratio $\frac{\sigma_{max}}{\sigma_{min}}$ governs the roundoff error in solving $Ax = b$. MATLAB warns you if this "condition number" is large. Then x is unreliable.

QR-Algorithm for computing Eigenvalues

The QR Algorithm

The QR algorithm computes a Schur decomposition of a matrix.

It is certainly one of the most important algorithms in eigenvalue computations [**?**]. However, it is applied to *dense* (or: full) matrices only.

The QR algorithm consists of two separate stages. First, by means of a similarity transformation, the original matrix is transformed in a finite number of steps to Hessenberg form or—in the Hermitian/symmetric case—to real tridiagonal form.

This first stage of the algorithm prepares its second stage: the actual QR iterations that are applied to the Hessenberg or tridiagonal matrix. The overall complexity (in terms of floating-point operations) of the algorithm is $\mathcal{O}(n^3)$, which, as we will see, is not entirely trivial to obtain.

The major limitation of the QR algorithm is that already the first stage usually generates complete fill-in for general sparse matrices. It can therefore not be applied to large sparse matrices, simply because of excessive memory requirements. On the other hand, the QR algorithm computes *all* eigenvalues (and eventually eigenvectors), which is rarely desired in sparse matrix computations anyway.

The treatment of the QR algorithm in these lecture notes on large-scale eigenvalue computation is justified in two respects. First, there are of course large or even huge dense eigenvalue problems. Second, the QR algorithm is employed in most other algorithms to solve "internal" small auxiliary eigenvalue problems.

The Basic QR Algorithm

In 1958, Rutishauser [**?**] of ETH Zurich experimented with a similar algorithm to the one we are about to present, but based on the LR factorization, i.e., Gaussian elimination without pivoting. That algorithm was not successful, as the LR factorization (nowadays called LU factorization) is not stable without pivoting.
Francis [**?**] noticed that the QR factorization would be the preferred choice and devised the QR algorithm with many of the bells and whistles used nowadays.

Before presenting the complete picture, we start with a basic iteration, given in Algorithm 4.1, discuss its properties, and improve on it step by step until we arrive at Francis' algorithm.
We notice first that

$$A_k = R_k Q_k = Q_k^* A_{k-1} Q_k, \tag{1}$$

and hence A_k and A_{k-1} are unitarily similar. The matrix sequence $\{A_k\}$ converges (under certain assumptions) towards an upper triangular matrix [**?**].
Let us assume that...

Algorithm 4.1: Basic QR Algorithm

Let $A \in \mathbb{C}^{n \times n}$. This algorithm computes an upper triangular matrix T and a unitary matrix U such that

$$A = UTU^*$$

is the Schur decomposition of A. Basic QR Algorithm
[Set $A_0 := A$ and $U_0 := I$ $k = 1, 2, \ldots$ Compute QR factorization: $A_{k-1} = Q_k R_k$ Update matrix: $A_k := R_k Q_k$ Update unitary: $U_k := U_{k-1} Q_k$ Set $T := A_\infty$ and $U := U_\infty$

Assume that the eigenvalues are mutually different in magnitude, so we can number them such that

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|.$$

Then—as we will show in Chapter 8—the elements of A_k below the diagonal converge to zero as

$$|a_{ij}^{(k)}| = \mathcal{O}\left(\left|\frac{\lambda_i}{\lambda_j}\right|^k\right), \quad i > j. \tag{2}$$

From equation (4.1) we observe:

$$A_k = Q_k^* A_{k-1} Q_k = Q_k^* Q_{k-1}^* A_{k-2} Q_{k-1} Q_k = \cdots = Q_k^* \cdots Q_1^* A_0 Q_1 \cdots Q_k, \tag{3}$$

and hence

$$A_k = U_k^* A_0 U_k,$$

where $U_k = Q_1 Q_2 \cdots Q_k$. With the same assumption on the eigenvalues, the sequence A_k converges to an upper triangular matrix, and U_k converges to the matrix of Schur vectors.

We conduct two MATLAB experiments to illustrate the convergence rate given in Equation (4.2). To that end, we construct a random 4×4 matrix with eigenvalues $1, 2, 3$, and 4. The matrix A_0 is formed via similarity transformation of the diagonal matrix $D = \mathrm{diag}(4, 3, 2, 1)$:

```
([4 3 2 1]); rand('seed',0); format short e S = rand(4); S = (S - 0.5) * 2; A = S
    * D / S;
for i = 1:20 [Q, R] = qr(A); A = R * Q; end
```

This iteration yields a sequence of matrices $A^{(k)}$. Below are the first few iterates:

$$A^{(0)} = \begin{bmatrix} -4.4529\text{e}-01 & 4.9063\text{e}+00 & 1.3354\text{e}+01 & -6.6617\text{e}+00 \\ -5.2052\text{e}+00 & 1.6107\text{e}+00 & 1.8630\text{e}+00 & 2.5660\text{e}-01 \\ 1.5265\text{e}-01 & 1.4907\text{e}+00 & -8.7871\text{e}-01 & 1.6668\text{e}+00 \\ -6.0021\text{e}-02 & -1.4130\text{e}+00 & 9.3153\text{e}-01 & 2.0428\text{e}+00 \end{bmatrix}$$

$$A^{(1)} = \begin{bmatrix} -6.3941\text{e}+00 & 5.9284\text{e}+00 & -1.5294\text{e}+00 & 1.9850\text{e}-01 \\ 2.4815\text{e}-01 & 4.7396\text{e}+00 & 1.1945\text{e}+01 & -7.0043\text{e}+00 \\ -2.8484\text{e}+00 & -2.2056\text{e}+01 & 6.5900\text{e}+00 & 1.2184\text{e}+00 \\ 4.9975\text{e}-01 & 2.3126\text{e}+01 & -2.1236\text{e}+00 & 6.3036\text{e}+00 \end{bmatrix}$$

$$A(2) = \begin{bmatrix} 4.7396e+00 & 1.4907e+00 & -2.1236e+00 & 2.3126e+01 \\ -4.3101e-01 & 2.4307e+00 & 2.2544e+00 & -8.2867e-01 \\ 1.2803e-01 & 2.4287e-01 & 1.6398e+00 & -1.8290e+00 \\ -4.8467e-02 & -5.8164e-02 & -1.0994e-01 & 1.1899e+00 \end{bmatrix}$$

$$A(3) = \begin{bmatrix} 4.3289e+00 & 1.0890e+00 & -3.9478e+00 & -2.2903e+01 \\ -1.8396e-01 & 2.7053e+00 & 1.9060e+00 & -1.2062e+00 \\ 6.7951e-02 & 1.7100e-01 & 1.6852e+00 & 2.5267e+00 \\ 1.3063e-02 & 2.2630e-02 & 7.9186e-02 & 1.2805e+00 \end{bmatrix}$$

$$A(4) = \begin{bmatrix} 4.1561e+00 & 7.6418e-01 & -5.1996e+00 & 2.2582e+01 \\ -9.4175e-02 & 2.8361e+00 & 1.5788e+00 & 2.0983e+00 \\ 3.5094e-02 & 1.1515e-01 & 1.7894e+00 & -2.9819e+00 \\ -3.6770e-03 & -8.7212e-03 & -5.7793e-02 & 1.2184e+00 \end{bmatrix}$$

$$A(5) = \begin{bmatrix} 4.0763e+00 & 5.2922e-01 & -6.0126e+00 & -2.2323e+01 \\ -5.3950e-02 & 2.9035e+00 & 1.3379e+00 & -2.5358e+00 \\ 1.7929e-02 & 7.7393e-02 & 1.8830e+00 & 3.2484e+00 \\ 1.0063e-03 & 3.2290e-03 & 3.7175e-02 & 1.1372e+00 \end{bmatrix}$$

$$A(6) = \begin{bmatrix} 4.0378e+00 & 3.6496e-01 & -6.4924e+00 & 2.2149e+01 \\ -3.3454e-02 & 2.9408e+00 & 1.1769e+00 & 2.7694e+00 \\ 9.1029e-03 & 5.2173e-02 & 1.9441e+00 & -3.4025e+00 \\ -2.6599e-04 & -1.1503e-03 & -2.1396e-02 & 1.0773e+00 \end{bmatrix}$$

$$A(7) = \begin{bmatrix} 4.0189e+00 & 2.5201e-01 & -6.7556e+00 & -2.2045e+01 \\ -2.1974e-02 & 2.9627e+00 & 1.0736e+00 & -2.9048e+00 \\ 4.6025e-03 & 3.5200e-02 & 1.9773e+00 & 3.4935e+00 \\ 6.8584e-05 & 3.9885e-04 & 1.1481e-02 & 1.0411e+00 \end{bmatrix}$$

$$A(8) = \begin{bmatrix} 4.0095e+00 & 1.7516e-01 & -6.8941e+00 & 2.1985e+01 \\ -1.5044e-02 & 2.9761e+00 & 1.0076e+00 & 2.9898e+00 \\ 2.3199e-03 & 2.3720e-02 & 1.9932e+00 & -3.5486e+00 \\ -1.7427e-05 & -1.3602e-04 & -5.9304e-03 & 1.0212e+00 \end{bmatrix}$$

$$A(9) = \begin{bmatrix} 4.0048e+00 & 1.2329e-01 & -6.9655e+00 & -2.1951e+01 \\ -1.0606e-02 & 2.9845e+00 & 9.6487e-01 & -3.0469e+00 \\ 1.1666e-03 & 1.5951e-02 & 1.9999e+00 & 3.5827e+00 \\ 4.3933e-06 & 4.5944e-05 & 3.0054e-03 & 1.0108e+00 \end{bmatrix}$$

$$A(10) = \begin{bmatrix} 4.0024e+00 & 8.8499e-02 & -7.0021e+00 & 2.1931e+01 \\ -7.6291e-03 & 2.9899e+00 & 9.3652e-01 & 3.0873e+00 \\ 5.8564e-04 & 1.0704e-02 & 2.0023e+00 & -3.6041e+00 \\ -1.1030e-06 & -1.5433e-05 & -1.5097e-03 & 1.0054e+00 \end{bmatrix}$$

$$A(11) = \begin{bmatrix} 4.0013e+00 & 6.5271e-02 & -7.0210e+00 & -2.1920e+01 \\ -5.5640e-03 & 2.9933e+00 & 9.1729e-01 & -3.1169e+00 \\ 2.9364e-04 & 7.1703e-03 & 2.0027e+00 & 3.6177e+00 \\ 2.7633e-07 & 5.1681e-06 & 7.5547e-04 & 1.0027e+00 \end{bmatrix}$$

$$A(12) = \begin{bmatrix} 4.0007e+00 & 4.9824e-02 & -7.0308e+00 & 2.1912e+01 \\ -4.0958e-03 & 2.9956e+00 & 9.0396e-01 & 3.1390e+00 \\ 1.4710e-04 & 4.7964e-03 & 2.0024e+00 & -3.6265e+00 \\ -6.9154e-08 & -1.7274e-06 & -3.7751e-04 & 1.0014e+00 \end{bmatrix}$$

$$A(13) = \begin{bmatrix} 4.0003e+00 & 3.9586e-02 & -7.0360e+00 & -2.1908e+01 \\ -3.0339e-03 & 2.9971e+00 & 8.9458e-01 & -3.1558e+00 \\ 7.3645e-05 & 3.2052e-03 & 2.0019e+00 & 3.6322e+00 \\ 1.7298e-08 & 5.7677e-07 & 1.8857e-04 & 1.0007e+00 \end{bmatrix}$$

$$A(14) = \begin{bmatrix} 4.0002e+00 & 3.2819e-02 & -7.0388e+00 & 2.1905e+01 \\ -2.2566e-03 & 2.9981e+00 & 8.8788e-01 & 3.1686e+00 \\ 3.6855e-05 & 2.1402e-03 & 2.0014e+00 & -3.6359e+00 \\ -4.3255e-09 & -1.9245e-07 & -9.4197e-05 & 1.0003e+00 \end{bmatrix}$$

$$A(15) = \begin{bmatrix} 4.0001e+00 & 2.8358e-02 & -7.0404e+00 & -2.1902e+01 \\ -1.6832e-03 & 2.9987e+00 & 8.8305e-01 & -3.1784e+00 \\ 1.8438e-05 & 1.4284e-03 & 2.0010e+00 & 3.6383e+00 \\ 1.0815e-09 & 6.4192e-08 & 4.7062e-05 & 1.0002e+00 \end{bmatrix}$$

$$A(16) = \begin{bmatrix} 4.0001e+00 & 2.5426e-02 & -7.0413e+00 & 2.1901e+01 \\ -1.2577e-03 & 2.9991e+00 & 8.7953e-01 & 3.1859e+00 \\ 9.2228e-06 & 9.5295e-04 & 2.0007e+00 & -3.6399e+00 \\ -2.7039e-10 & -2.1406e-08 & -2.3517e-05 & 1.0001e+00 \end{bmatrix}$$

$$A(17) = \begin{bmatrix} 4.0000e+00 & 2.3503e-02 & -7.0418e+00 & -2.1900e+01 \\ -9.4099e-04 & 2.9994e+00 & 8.7697e-01 & -3.1917e+00 \\ 4.6126e-06 & 6.3562e-04 & 2.0005e+00 & 3.6409e+00 \\ 6.7600e-11 & 7.1371e-09 & 1.1754e-05 & 1.0000e+00 \end{bmatrix}$$

$$A(18) = \begin{bmatrix} 4.0000e+00 & 2.2246e-02 & -7.0422e+00 & 2.1899e+01 \\ -7.0459e-04 & 2.9996e+00 & 8.7508e-01 & 3.1960e+00 \\ 2.3067e-06 & 4.2388e-04 & 2.0003e+00 & -3.6416e+00 \\ -1.6900e-11 & -2.3794e-09 & -5.8750e-06 & 1.0000e+00 \end{bmatrix}$$

$$A(19) = \begin{bmatrix} 4.0000e+00 & 2.1427e-02 & -7.0424e+00 & -2.1898e+01 \\ -5.2787e-04 & 2.9997e+00 & 8.7369e-01 & -3.1994e+00 \\ 1.1535e-06 & 2.8265e-04 & 2.0002e+00 & 3.6421e+00 \\ 4.2251e-12 & 7.9321e-10 & 2.9369e-06 & 1.0000e+00 \end{bmatrix}$$

$$A(20) = \begin{bmatrix} 4.0000e+00 & 2.0896e-02 & -7.0425e+00 & 2.1898e+01 \\ -3.9562e-04 & 2.9998e+00 & 8.7266e-01 & 3.2019e+00 \\ 5.7679e-07 & 1.8846e-04 & 2.0002e+00 & -3.6424e+00 \\ -1.0563e-12 & -2.6442e-10 & -1.4682e-06 & 1.0000e+00 \end{bmatrix}$$

Looking at the element-wise quotients of the last two matrices, one recognizes the convergence rates claimed in (3.2).

$$A(20)./A(19) = \begin{bmatrix} 1.0000 & 0.9752 & 1.0000 & -1.0000 \\ 0.7495 & 1.0000 & 0.9988 & -1.0008 \\ 0.5000 & 0.6668 & 1.0000 & -1.0001 \\ -0.2500 & -0.3334 & -0.4999 & 1.0000 \end{bmatrix}$$

The elements above and on the diagonal are relatively stable.
If we run the same little MATLAB script but with the initial diagonal matrix D replaced by

$$D = \mathrm{diag}(5, 2, 2, 1),$$

then we obtain:

$$A(19) = \begin{bmatrix} 5.0000\mathrm{e}{+00} & 4.0172\mathrm{e}{+00} & -9.7427\mathrm{e}{+00} & -3.3483\mathrm{e}{+01} \\ -4.2800\mathrm{e}{-08} & 2.0000\mathrm{e}{+00} & 2.1100\mathrm{e}{-05} & -4.3247\mathrm{e}{+00} \\ 1.3027\mathrm{e}{-08} & 7.0605\mathrm{e}{-08} & 2.0000\mathrm{e}{+00} & 2.1769\mathrm{e}{+00} \\ 8.0101\mathrm{e}{-14} & -2.4420\mathrm{e}{-08} & 4.8467\mathrm{e}{-06} & 1.0000\mathrm{e}{+00} \end{bmatrix}$$

$$A(20) = \begin{bmatrix} 5.0000\mathrm{e}{+00} & 4.0172\mathrm{e}{+00} & -9.7427\mathrm{e}{+00} & 3.3483\mathrm{e}{+01} \\ -1.7120\mathrm{e}{-08} & 2.0000\mathrm{e}{+00} & 1.0536\mathrm{e}{-05} & 4.3247\mathrm{e}{+00} \\ 5.2106\mathrm{e}{-09} & 3.3558\mathrm{e}{-08} & 2.0000\mathrm{e}{+00} & -2.1769\mathrm{e}{+00} \\ -1.6020\mathrm{e}{-14} & 1.2210\mathrm{e}{-08} & -2.4234\mathrm{e}{-06} & 1.0000\mathrm{e}{+00} \end{bmatrix}$$

So, again the eigenvalues are visible on the diagonal of $A(20)$. The element-wise quotients of $A(20)$ relative to $A(19)$ are:

$$\frac{A(20)}{A(19)} = \begin{bmatrix} 1.0000 & 1.0000 & 1.0000 & -1.0000 \\ 0.4000 & 1.0000 & 0.4992 & -1.0000 \\ 0.4000 & 0.4754 & 1.0000 & -1.0000 \\ -0.2000 & -0.5000 & -0.5000 & 1.0000 \end{bmatrix}$$

$$A(20)./A(19) = \begin{bmatrix} 1.0000 & 1.0000 & 1.0000 & -1.0000 \\ 0.4000 & 1.0000 & 0.4993 & -1.0000 \\ 0.4000 & 0.4753 & 1.0000 & -1.0000 \\ -0.2000 & -0.5000 & -0.5000 & 1.0000 \end{bmatrix}$$

Notice that equation (3.2) does not state a convergence rate for the element at position (3, 2).

These small numerical experiments demonstrate that the convergence rates given in (3.2) are indeed observed during an actual run of the basic QR algorithm.

The conclusions we can draw from this are the following:

1. **The convergence of the algorithm is slow.** In fact, it can be arbitrarily slow if eigenvalues are very close to each other.

2. **The algorithm is expensive.** Each iteration step requires the computation of the QR factorization of a full $n \times n$ matrix, i.e., each single iteration step has a complexity of $\mathcal{O}(n^3)$.

   Even if we assume that the number of steps is proportional to n, the total complexity would be $\mathcal{O}(n^4)$. The latter assumption is not even guaranteed, as mentioned in point 1.

In the following, we aim to improve on both issues. First, we seek a matrix structure that is preserved by the QR algorithm and that reduces the cost of a single iteration step.

Then, we want to enhance the convergence properties of the algorithm.

Given a real matrix $A \in \mathbb{R}^{n \times n}$, we aim to find an orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ such that

$$H = Q^T A Q$$

where H is an upper Hessenberg matrix (i.e., all entries below the first subdiagonal are zero).

This is achieved through a series of similarity transformations using **Householder reflectors**.

Hessenberg Reduction Algorithm

For $k = 1$ to $n - 2$, perform the following:

1. Let $x = A_{k+1:n,k} \in \mathbb{R}^{n-k}$ be the subvector of the k-th column of A, starting from the $(k+1)$-th row.
2. Construct a Householder reflector $H_k = I - 2\frac{vv^T}{v^Tv}$, where $v = x + \text{sign}(x_1)\|x\|_2 e_1$, and e_1 is the first basis vector in $\mathbb{R}^{n-k}$.
3. Expand H_k to size $n \times n$ by embedding it into an identity matrix:

$$Q_k = \begin{bmatrix} I_k & 0 \\ 0 & H_k \end{bmatrix}$$

4. Apply the similarity transformation:

$$A \leftarrow Q_k^T A Q_k$$

5. Accumulate $Q = Q_1 Q_2 \cdots Q_{n-2}$

At the end of this process, the matrix A is transformed into an upper Hessenberg matrix H, and the orthogonal matrix Q satisfies:

$$H = Q^T A Q$$

- The transformation preserves eigenvalues because it is a similarity transformation.
- Only $n-2$ Householder reflectors are required.
- The resulting Hessenberg matrix is the starting point for the Implicit QR Algorithm.

Each Householder transformation affects an $(n-k) \times (n-k)$ submatrix, requiring about $4(n-k)^2$ flops.

$$\text{Total cost: } \sum_{k=1}^{n-2} 4(n-k)^2 \approx \frac{10}{3} n^3 \text{ flops}$$

Example: Hessenberg Reduction via Householder Reflector

Given Matrix

Let

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

We aim to reduce A to upper Hessenberg form using a Householder similarity transformation. We want to zero out the $(3,1)$ entry. Let

$$x = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \quad \text{(the subvector of column 1 below the diagonal)}$$

Step 2: Construct Householder Vector

Compute

$$v = x + \|x\|_2 \cdot e_1 = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \sqrt{1^2 + (-2)^2} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \sqrt{5} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 + \sqrt{5} \\ -2 \end{bmatrix}$$

Normalize v:

$$v = \frac{1}{\|v\|_2} \begin{bmatrix} 1 + \sqrt{5} \\ -2 \end{bmatrix}$$

Step 3: Form the Householder Matrix

Let

$$H = I - 2vv^T$$

embedded into the identity matrix as:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & * & * \\ 0 & * & * \end{bmatrix}$$

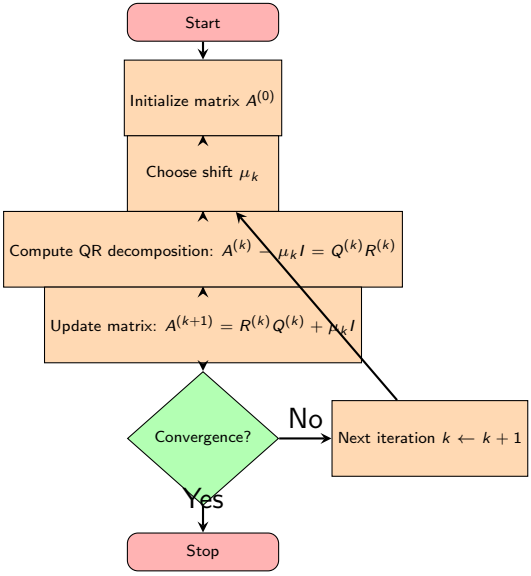
We apply the similarity transformation:

$$H = Q^T A Q$$

The resulting matrix H will have the form:

$$H = \begin{bmatrix} * & * & * \\ * & * & * \\ 0 & * & * \end{bmatrix}$$

QR Algorithm with Shifts Flowchart



QR Algorithm with Shifts

QR Algorithm with Shifts [1] **Input:** Matrix A, tolerance ε, maximum iterations max_iter Initialize $A^{(0)} \leftarrow A$, $k \leftarrow 0$ $k <$ max_iter Choose shift $\mu_k = a_{nn}^{(k)}$ (bottom-right element) Compute QR decomposition: $A^{(k)} - \mu_k I = Q^{(k)} R^{(k)}$ Update matrix: $A^{(k+1)} = R^{(k)} Q^{(k)} + \mu_k I$ all off-diagonal entries of $A^{(k+1)}$ are less than ε **break** Convergence achieved $k \leftarrow k + 1$ **Output:** Approximate eigenvalues are the diagonal elements of $A^{(k+1)}$

Thank You!