现代数学概论【科学计算】

Lecture 2 - Matrix Analysis

Xian-Liang Hu

School of Mathematical Sciences, Zhejiang University, CHINA.

http://www.mathweb.zju.edu.cn:8080/xlhu/sc.html

4. Eigenvalue Problem
   - The problem
   - Numerical Method

5. Linear Least Square
   - The Problem
   - Numerical Method

6. Matrix Factorization
   - Orthogonalization
   - Singular Value Decomposition
   - Applications

Eigenvalue Problem

Definition

Given an $n \times n$ matrix A representing a linear transformation on an n-dimensional vector space, we wish to find a nonzero vector x and a scalar λ such that

$$A\mathrm{x} = \lambda \mathrm{x},$$

where scalar λ is an *eigenvalue*, and x is the corresponding (*right*) *eigenvector*.

- *left eigenvector* y: $\mathrm{y}^T A = \lambda \mathrm{y}^T$
- *spectrum* $\lambda(A)$:all the eigenvalues of a matrix A
- *spectral radius* $\rho(A)$: $\max |\lambda| : \lambda \in \lambda(A)$

例 4.1 弹簧-重物系统 考虑图 4.1 的弹簧-重物系统，其中包括三个质量分别为 m_1, m_2, m_3 的重物，它们的垂直位置分别为 y_1, y_2, y_3，由三个弹性系数分别为 k_1, k_2, k_3 的弹簧相连. 根据牛顿第二定律，系统运动满足下面的常微分方程

$$\boldsymbol{M}\boldsymbol{y}'' + \boldsymbol{K}\boldsymbol{y} = \boldsymbol{0},$$

其中

$$\boldsymbol{M} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}$$

称为质量矩阵，而

$$\boldsymbol{K} = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix}$$

称为刚性矩阵.

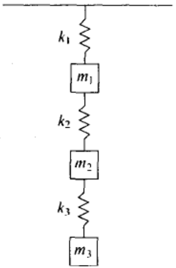


图 4.1 弹簧-重物系统

这个系统以自然频率 ω 做谐波运动，解的分量由

$$y_k(t) = x_k e^{i\omega t}$$

给出，其中 x_k 是振幅，$k=1,2,3$，$i=\sqrt{-1}$. 为确定频率 ω 及振动的波型（即振幅 x_k），注意到对解的每个分量，有

$$y_k''(t) = -\omega^2 x_k e^{i\omega t},$$

将这个关系代入微分方程，得代数方程

$$\boldsymbol{K}\boldsymbol{x} = \omega^2 \boldsymbol{M}\boldsymbol{x},$$

Evaluation of λ

To solve $A\mathrm{x} = \lambda\mathrm{x}$, it is equivalent to solve

$$(A - \lambda\mathrm{I})\mathrm{x} = 0,$$

where x be nonzero $\iff$ coefficient matrix is singular, that is, the roots of

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

are the eigenvalues of A. That is

Eigenvalue Problem $\iff$ Roots Problem

Example (Characteristic Polynomial)

$$\boldsymbol{A} = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}.$$

The characteristic polynomial of matrix $\boldsymbol{A}$ is

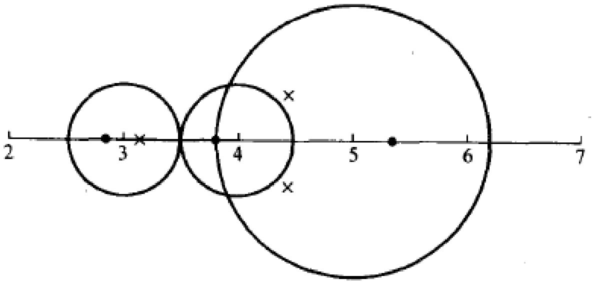
$$\begin{aligned} \det\left(\begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) &= \det\left(\begin{bmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{bmatrix} \right) \\ &= (3-\lambda)(3-\lambda) - 1 \times 1 \\ &= \lambda^2 - 6\lambda + 8 \\ &= (\lambda - 4)(\lambda - 2) = 0. \end{aligned}$$

- The eigenvalues are $\lambda_1 = 4$ and $\lambda_2 = 2$.

Roughly estimation for λ

Theorem (Gershgorin's Theorem)

The eigenvalues of an $n \times n$ matrix A are all contained within the union of n disks, with the kth disk centered at a_{kk} and having radius $\sum_{j \neq k} |a_{kj}|$.



Proof: Let λ be any eigenvalue, with corresponding eigenvector x, normalized so that $\|x\|_{\inf} = 1$. Let x_k be an entry of x such that $|x_k| = 1$ (at least one component has magnitude 1, by definition of the inf-norm). Because $Ax = \lambda x$, we have

$$(\lambda - a_{kk})\, x_k = \sum_{j \neq k} a_{kj} x_j$$

so that

$$|\lambda - a_{kk}| \leq \sum_{j \neq k} |a_{kj}| \cdot |x_j| \leq \sum_{j \neq k} |a_{kj}| .$$

Example

$$A_1 = \begin{bmatrix} 4.0 & -0.5 & 0.0 \\ 0.6 & 5.0 & -0.6 \\ 0.0 & 0.5 & 3.0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 4.0 & 0.5 & 0.0 \\ 0.6 & 5.0 & 0.6 \\ 0.0 & 0.5 & 3.0 \end{bmatrix}$$

Proof: Let λ be any eigenvalue, with corresponding eigenvector x, normalized so that $\|x\|_{\text{inf}} = 1$. Let x_k be an entry of x such that $|x_k| = 1$ (at least one component has magnitude 1, by definition of the inf-norm). Because $Ax = \lambda x$, we have

$$(\lambda - a_{kk})\, x_k = \sum_{j \neq k} a_{kj} x_j$$

so that

$$|\lambda - a_{kk}| \leq \sum_{j \neq k} |a_{kj}| \cdot |x_j| \leq \sum_{j \neq k} |a_{kj}| \,.$$

Example

$$A_1 = \begin{bmatrix} 4.0 & -0.5 & 0.0 \\ 0.6 & 5.0 & -0.6 \\ 0.0 & 0.5 & 3.0 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} 4.0 & 0.5 & 0.0 \\ 0.6 & 5.0 & 0.6 \\ 0.0 & 0.5 & 3.0 \end{bmatrix}$$

Sensitive Analysis

Consider the perturbed eigenvalue problem

$$(A+E)(\mathrm{x}+\delta \mathrm{x}) = (\lambda+\delta\lambda)(\mathrm{x}+\delta \mathrm{x})$$

Certain simplification(negelect high order term and left multiply with y^H) leads to

$$\delta\lambda \approx \frac{\mathrm{y}^H E \mathrm{x}}{\mathrm{y}^H \mathrm{x}}.$$

Finally,

$$|\delta\lambda| \leq \frac{\|\mathrm{y}\|_2 \|\mathrm{x}\|_2}{|\mathrm{y}^H \mathrm{x}|} \|E\|_2 = \frac{1}{\cos\theta} \|E\|_2,$$

where θ is the angle between x and y.

- It is sensitive if the right and left eigenvectors are nearly orthogonal.
- For real symmetric/complex Hermitian matrices, it is always well-conditioned

Sensitive Analysis

Consider the perturbed eigenvalue problem

$$(A+E)(\mathrm{x}+\delta\mathrm{x}) = (\lambda+\delta\lambda)(\mathrm{x}+\delta\mathrm{x})$$

Certain simplification(negelect high order term and left multiply with y^H) leads to

$$\delta\lambda \approx \frac{\mathrm{y}^H E\mathrm{x}}{\mathrm{y}^H\mathrm{x}}.$$

Finally,

$$|\delta\lambda| \leq \frac{\|\mathrm{y}\|_2\|\mathrm{x}\|_2}{|\mathrm{y}^H\mathrm{x}|}\|E\|_2 = \frac{1}{\cos\theta}\|E\|_2,$$

where θ is the angle between x and y.

- It is sensitive if the right and left eigenvectors are nearly orthogonal.
- For real symmetric/complex Hermitian matrices, it is always well-conditioned

Sensitive Analysis

Consider the perturbed eigenvalue problem

$$(A+E)(\mathrm{x}+\delta\mathrm{x}) = (\lambda+\delta\lambda)(\mathrm{x}+\delta\mathrm{x})$$

Certain simplification(negelect high order term and left multiply with y^H) leads to

$$\delta\lambda \approx \frac{\mathrm{y}^H E\mathrm{x}}{\mathrm{y}^H\mathrm{x}}.$$

Finally,

$$|\delta\lambda| \leq \frac{\|\mathrm{y}\|_2\|\mathrm{x}\|_2}{|\mathrm{y}^H\mathrm{x}|}\|E\|_2 = \frac{1}{\cos\theta}\|E\|_2,$$

where θ is the angle between x and y.

- It is sensitive if the right and left eigenvectors are nearly orthogonal.
- For real symmetric/complex Hermitian matrices, it is always well-conditioned

Practical Numerical Methods

- 求解特征值可粗略地分为: 直接法 和 迭代法
- 注意直接法中也需要用到迭代,如果其中用一个固定的迭代次数收敛几乎不失败，也称为直接法
- 迭代法通常用于稀疏矩阵,或能方便地执行矩阵向量乘法的隐式算子情形

1. Power Iteration

Given x_0 = arbitrary nonzero vector
for $k = 0, 1, \cdots$ *until convergence* **do**
  $y_k = Ax_{k-1}$;
  $x_k = y_k / \|y_k\|_2$;
  $\lambda_k = x_k^T A x_k$;
end

- It is simple both for understanding and implementation.
- Only the largest eigenvalue of A could be obtained.
- Question: how to calculate the eigenvalue which close to given σ?

1. Power Iteration

```
Given x_0 = arbitrary nonzero vector
for k = 0, 1, ··· until convergence do
    y_k = A x_{k-1};
    x_k = y_k / ||y_k||_2;
    λ_k = x_k^T A x_k;
end
```

$$
\begin{array}{l}
\text{Given } \mathrm{x}_0 = \text{arbitrary nonzero vector}\\
\textbf{for } k = 0, 1, \cdots \textit{ until convergence } \textbf{do}\\
\quad \mathrm{y}_k = A\mathrm{x}_{k-1};\\
\quad \mathrm{x}_k = \mathrm{y}_k / \|\mathrm{y}_k\|_2;\\
\quad \lambda_k = \mathrm{x}_k^T A \mathrm{x}_k;\\
\textbf{end}
\end{array}
$$

- It is simple both for understanding and implementation.
- Only the largest eigenvalue of A could be obtained.
- Question: how to calculate the eigenvalue which close to given σ?

1. Power Iteration

```
Given x_0 = arbitrary nonzero vector
for k = 0, 1, ··· until convergence do
    y_k = A x_{k-1};
    x_k = y_k / ||y_k||_2;
    λ_k = x_k^T A x_k;
end
```

- It is simple both for understanding and implementation.
- Only the largest eigenvalue of A could be obtained.
- Question: how to calculate the eigenvalue which close to given σ?

1. Power Iteration

Given x_0 = arbitrary nonzero vector
for $k = 0, 1, \cdots$ *until convergence* **do**
    $y_k = Ax_{k-1}$;
    $x_k = y_k / \|y_k\|_2$;
    $\lambda_k = x_k^T A x_k$;
end

- It is simple both for understanding and implementation.
- Only the largest eigenvalue of A could be obtained.
- Question: how to calculate the eigenvalue which close to given σ?

Example (Normalized Power Iteration.)

We apply power iteration to matrix

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

with starting vector

$$\boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

As is shown in the right table, the approximate eigenvector is normalized at each iteration, thereby avoiding geometric growth or decay of its components.

k	$\boldsymbol{x}_k^T$		$\lVert \boldsymbol{y}_k \rVert_\infty$
0	0.000	1.0	
1	0.333	1.0	3.000
2	0.333	1.0	3.000
3	0.600	1.0	3.333
4	0.778	1.0	3.600
5	0.882	1.0	3.778
6	0.969	1.0	3.939
7	0.984	1.0	3.969
8	0.992	1.0	3.984
9	0.996	1.0	3.992

Example (Normalized Power Iteration.)

We apply power iteration to matrix

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

with starting vector

$$\boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

As is shown in the right table, the approximate eigenvector is normalized at each iteration, thereby avoiding geometric growth or decay of its components.

k	$\boldsymbol{x}_k^T$		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	0.333	1.0	3.000
2	0.333	1.0	3.000
3	0.600	1.0	3.333
4	0.778	1.0	3.600
5	0.882	1.0	3.778
6	0.969	1.0	3.939
7	0.984	1.0	3.969
8	0.992	1.0	3.984
9	0.996	1.0	3.992

2. Inverse Iteration

$x_0 =$ arbitrary nonzero vector ;
for $k = 0, 1, \cdots$ *until convergence* **do**
  solve $\boldsymbol{A}\boldsymbol{y}_k = \boldsymbol{x}_{k-1}$ for $\boldsymbol{y}_k$;
  $x_k = y_k / \|y_{ik}\|_2$;
end

Algorithm 1: Inverse Iteration

- It calculates the smallest one, compared with the power iteration.
- Question: how to calculate all other eigenvalues?

k	x_k^T		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	−0.333	1.0	
2	−0.333	1.0	0.375
3	−0.600	1.0	0.417
4	−0.778	1.0	0.450
5	−0.882	1.0	0.485
6	−0.969	1.0	0.492
7	−0.984	1.0	0.496
8	−0.992	1.0	0.498
9	−0.996	1.0	0.499

2. Inverse Iteration

x_0 = arbitrary nonzero vector ;
for $k = 0, 1, \cdots$ *until convergence* **do**
    solve $\boldsymbol{A}\boldsymbol{y}_k = \boldsymbol{x}_{k-1}$ for $\boldsymbol{y}_k$;
    $x_k = y_k / \|y_{ik}\|_2$;
end

Algorithm 2: Inverse Iteration

- It calculates the smallest one, compared with the power iteration.
- Question: how to calculate all other eigenvalues?

k	$\boldsymbol{x}_k^T$		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	−0.333	1.0	
2	−0.333	1.0	0.375
3	−0.600	1.0	0.417
4	−0.778	1.0	0.450
5	−0.882	1.0	0.485
6	−0.969	1.0	0.492
7	−0.984	1.0	0.496
8	−0.992	1.0	0.498
9	−0.996	1.0	0.499

2. Inverse Iteration

x_0 = arbitrary nonzero vector ;
for $k = 0, 1, \cdots$ *until convergence* **do**
    solve $\boldsymbol{A}\boldsymbol{y}_k = \boldsymbol{x}_{k-1}$ for $\boldsymbol{y}_k$;
    $x_k = y_k / \|y_{ik}\|_2$;
end

Algorithm 3: Inverse Iteration

- It calculates the smallest one, compared with the power iteration.
- Question: how to calculate all other eigenvalues?

k	$\boldsymbol{x}_k^T$		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	−0.333	1.0	
2	−0.333	1.0	0.375
3	−0.600	1.0	0.417
4	−0.778	1.0	0.450
5	−0.882	1.0	0.485
6	−0.969	1.0	0.492
7	−0.984	1.0	0.496
8	−0.992	1.0	0.498
9	−0.996	1.0	0.499

Calculate p eigenvalues simultaneously?

Given Z_0 as an $n \times p$ orthogonal matrix;
for $i = 0, 1, \cdots$ *until convergence* **do**
  Calculate $Y_{i+1} = AZ_i$;
  Decompose $Z_{i+1}R_{i+1} = Y_{i+1}$;
end

- 可利用QR分解进行计算;
- 可同时求得一个p–维不变子空间($p > 1$，由Z_{i+1}列向量张成);
- 也称为子空间迭代或同时迭代

Calculate p eigenvalues simultaneously?

Given Z_0 as an $n \times p$ orthogonal matrix;
for $i = 0, 1, \cdots$ *until convergence* **do**
    Calculate $Y_{i+1} = AZ_i$;
    Decompose $Z_{i+1}R_{i+1} = Y_{i+1}$;
end

- 可利用QR分解进行计算;
- 可同时求得一个p–维不变子空间($p > 1$，由Z_{i+1}列向量张成);
- 也称为子空间迭代或同时迭代

Calculate p eigenvalues simultaneously?

Given Z_0 as an $n \times p$ orthogonal matrix;
for $i = 0, 1, \cdots$ *until convergence* **do**
  Calculate $Y_{i+1} = AZ_i$;
  Decompose $Z_{i+1}R_{i+1} = Y_{i+1}$;
end

- 可利用QR分解进行计算;
- 可同时求得一个p–维不变子空间($p > 1$，由Z_{i+1}列向量张成);
- 也称为子空间迭代或同时迭代

3.QR Iteration

Given A_0 ;
for $k = 1, 2, \cdots$ *until convergence* **do**
    Compute QR factorization;
    $\boldsymbol{Q}_k \boldsymbol{R}_k = \boldsymbol{A}_{k-1}$;
    $\boldsymbol{A}_k = \boldsymbol{R}_k \boldsymbol{Q}_k$;
end

- $A_k = R_k Q_k = (Q_k^T Q_k) R_k Q_k = Q_k^T (Q_k R_k) Q_k = Q_k^T A_{k-1} Q_k$
- $\{A_k\}_{k=1}^{\infty}$ will converge to an upper triangular matrix whose diagonal elements are all the eigenvalues.
- A_i等同于用正交迭代隐式计算矩阵$Z_i^T Z_i$，且数值稳定
- 带位移的QR迭代可加快收敛（当选择的位移接近特征值时二次收敛）

3.QR Iteration

```
Given A_0 ;
for k = 1, 2, ··· until convergence do
    Compute QR factorization;
    Q_k R_k = A_{k-1};
    A_k = R_k Q_k;
end
```

- $A_k = R_k Q_k = (Q_k^T Q_k) R_k Q_k = Q_k^T (Q_k R_k) Q_k = Q_k^T A_{k-1} Q_k$
- $\{A_k\}_{k=1}^{\infty}$ will converge to an upper triangular matrix whose diagonal elements are all the eigenvalues.
- A_i等同于用正交迭代隐式计算矩阵$Z_i^T Z_i$，且数值稳定
- 带位移的QR迭代可加快收敛（当选择的位移接近特征值时二次收敛）

3.QR Iteration

```
Given A_0 ;
for k = 1, 2, ··· until convergence do
    Compute QR factorization;
    Q_k R_k = A_{k-1};
    A_k = R_k Q_k;
end
```

- $A_k = R_k Q_k = (Q_k^T Q_k) R_k Q_k = Q_k^T (Q_k R_k) Q_k = Q_k^T A_{k-1} Q_k$
- $\{A_k\}_{k=1}^{\infty}$ will converge to an upper triangular matrix whose diagonal elements are all the eigenvalues.
- A_i等同于用正交迭代隐式计算矩阵$Z_i^T Z_i$，且数值稳定
- 带位移的QR迭代可加快收敛（当选择的位移接近特征值时二次收敛）

3.QR Iteration

```
Given A_0 ;
for k = 1, 2, ··· until convergence do
    Compute QR factorization;
    Q_k R_k = A_{k-1};
    A_k = R_k Q_k;
end
```

- $A_k = R_k Q_k = (Q_k^T Q_k) R_k Q_k = Q_k^T (Q_k R_k) Q_k = Q_k^T A_{k-1} Q_k$
- $\{A_k\}_{k=1}^{\infty}$ will converge to an upper triangular matrix whose diagonal elements are all the eigenvalues.
- A_i等同于用正交迭代隐式计算矩阵$Z_i^T Z_i$，且数值稳定
- 带位移的QR迭代可加快收敛（当选择的位移接近特征值时二次收敛）

Example (QR Iteration.)

Calculate the eigenvalues of A, which is $\lambda_1 = 4, \lambda_2 = 3, \lambda_3 = 2, \lambda_4 = 1$.

$$A = \begin{bmatrix} 2.9766 & 0.3945 & 0.4198 & 1.1159 \\ 0.3945 & 2.7328 & -0.3097 & 0.1129 \\ 0.4198 & -0.3097 & 2.5675 & 0.6079 \\ 1.1159 & 0.1129 & 0.6079 & 1.7231 \end{bmatrix}$$

The first iteration: calculate QR factorization and then do the reverse product

$$A_1 = \begin{bmatrix} 3.7703 & 0.1745 & 0.5126 & -0.3934 \\ 0.1745 & 2.7675 & -0.3872 & 0.0539 \\ 0.5126 & -0.3872 & 2.4019 & -0.1241 \\ -0.3934 & 0.0539 & -0.1241 & 1.0603 \end{bmatrix}$$

- Most of the off-diagonal entries are now smaller in magnitude
- the diagonal entries are somewhat closer to the eigenvalues

Example (QR Iteration.)

Calculate the eigenvalues of A, which is $\lambda_1 = 4, \lambda_2 = 3, \lambda_3 = 2, \lambda_4 = 1$.

$$A = \begin{bmatrix} 2.9766 & 0.3945 & 0.4198 & 1.1159 \\ 0.3945 & 2.7328 & -0.3097 & 0.1129 \\ 0.4198 & -0.3097 & 2.5675 & 0.6079 \\ 1.1159 & 0.1129 & 0.6079 & 1.7231 \end{bmatrix}$$

The first iteration: calculate QR factorization and then do the reverse product

$$A_1 = \begin{bmatrix} 3.7703 & 0.1745 & 0.5126 & -0.3934 \\ 0.1745 & 2.7675 & -0.3872 & 0.0539 \\ 0.5126 & -0.3872 & 2.4019 & -0.1241 \\ -0.3934 & 0.0539 & -0.1241 & 1.0603 \end{bmatrix}$$

- Most of the off-diagonal entries are now smaller in magnitude
- the diagonal entries are somewhat closer to the eigenvalues

Example (QR Iteration.)

Calculate the eigenvalues of A, which is $\lambda_1 = 4, \lambda_2 = 3, \lambda_3 = 2, \lambda_4 = 1$.

$$A = \begin{bmatrix} 2.9766 & 0.3945 & 0.4198 & 1.1159 \\ 0.3945 & 2.7328 & -0.3097 & 0.1129 \\ 0.4198 & -0.3097 & 2.5675 & 0.6079 \\ 1.1159 & 0.1129 & 0.6079 & 1.7231 \end{bmatrix}$$

The first iteration: calculate QR factorization and then do the reverse product

$$A_1 = \begin{bmatrix} 3.7703 & 0.1745 & 0.5126 & -0.3934 \\ 0.1745 & 2.7675 & -0.3872 & 0.0539 \\ 0.5126 & -0.3872 & 2.4019 & -0.1241 \\ -0.3934 & 0.0539 & -0.1241 & 1.0603 \end{bmatrix}$$

- Most of the off-diagonal entries are now smaller in magnitude
- the diagonal entries are somewhat closer to the eigenvalues

The second iteration yields

$$A_2 = \begin{bmatrix} 3.9436 & 0.0143 & 0.3046 & 0.1038 \\ 0.0143 & 2.8737 & -0.3362 & -0.0285 \\ 0.3046 & -0.3362 & 2.1785 & 0.0083 \\ 0.1038 & -0.0285 & 0.0083 & 1.0042 \end{bmatrix}$$

and the third iteration reaches to

$$A_3 = \begin{bmatrix} 3.9832 & -0.0356 & 0.1611 & -0.0262 \\ -0.0356 & 2.9421 & -0.2432 & 0.0098 \\ 0.1611 & -0.2432 & 2.0743 & 0.0047 \\ -0.0262 & 0.0098 & 0.0047 & 1.0003 \end{bmatrix}$$

- The off-diagonal entries are now fairly small,
- The diagonal entries are quite close to the eigenvalues,
- A few more iterations will lead to a satisfactory accuracy.

The second iteration yields

$$A_2 = \begin{bmatrix} 3.9436 & 0.0143 & 0.3046 & 0.1038 \\ 0.0143 & 2.8737 & -0.3362 & -0.0285 \\ 0.3046 & -0.3362 & 2.1785 & 0.0083 \\ 0.1038 & -0.0285 & 0.0083 & 1.0042 \end{bmatrix}$$

and the third iteration reaches to

$$A_3 = \begin{bmatrix} 3.9832 & -0.0356 & 0.1611 & -0.0262 \\ -0.0356 & 2.9421 & -0.2432 & 0.0098 \\ 0.1611 & -0.2432 & 2.0743 & 0.0047 \\ -0.0262 & 0.0098 & 0.0047 & 1.0003 \end{bmatrix}$$

- The off-diagonal entries are now fairly small,
- The diagonal entries are quite close to the eigenvalues,
- A few more iterations will lead to a satisfactory accuracy.

The second iteration yields

$$A_2 = \begin{bmatrix} 3.9436 & 0.0143 & 0.3046 & 0.1038 \\ 0.0143 & 2.8737 & -0.3362 & -0.0285 \\ 0.3046 & -0.3362 & 2.1785 & 0.0083 \\ 0.1038 & -0.0285 & 0.0083 & 1.0042 \end{bmatrix}$$

and the third iteration reaches to

$$A_3 = \begin{bmatrix} 3.9832 & -0.0356 & 0.1611 & -0.0262 \\ -0.0356 & 2.9421 & -0.2432 & 0.0098 \\ 0.1611 & -0.2432 & 2.0743 & 0.0047 \\ -0.0262 & 0.0098 & 0.0047 & 1.0003 \end{bmatrix}$$

- The off-diagonal entries are now fairly small,
- The diagonal entries are quite close to the eigenvalues,
- A few more iterations will lead to a satisfactory accuracy.

If A is symmetric ...

- The best estimate for the corresponding eigenvalue λ can be considered as an $n \times 1$ linear least squares approximation problem

$$\boldsymbol{x}\lambda \approx \boldsymbol{A}\boldsymbol{x},$$

where $\boldsymbol{x}$ is an approximate eigenvector for a real matrix $\boldsymbol{A}$.

- From the normal equation $\boldsymbol{x}^T\boldsymbol{x}\lambda = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$,the least squares solution is

$$\lambda = \frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}.$$

- The Rayleigh quotient $\boldsymbol{x}_k^T\boldsymbol{A}\boldsymbol{x}_k/\boldsymbol{x}_k^T\boldsymbol{x}_k$ gives a better approximation

If A is symmetric ...

- The best estimate for the corresponding eigenvalue λ can be considered as an $n \times 1$ linear least squares approximation problem

$$\boldsymbol{x}\lambda \approx \boldsymbol{A}\boldsymbol{x},$$

where $\boldsymbol{x}$ is an approximate eigenvector for a real matrix $\boldsymbol{A}$.

- From the normal equation $\boldsymbol{x}^T\boldsymbol{x}\lambda = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$,the least squares solution is

$$\lambda = \frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}.$$

- The Rayleigh quotient $\boldsymbol{x}_k^T\boldsymbol{A}\boldsymbol{x}_k/\boldsymbol{x}_k^T\boldsymbol{x}_k$ gives a better approximation

If A is symmetric ...

- The best estimate for the corresponding eigenvalue λ can be considered as an $n \times 1$ linear least squares approximation problem

$$\boldsymbol{x}\lambda \approx \boldsymbol{A}\boldsymbol{x},$$

where $\boldsymbol{x}$ is an approximate eigenvector for a real matrix $\boldsymbol{A}$.

- From the normal equation $\boldsymbol{x}^T\boldsymbol{x}\lambda = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$,the least squares solution is

$$\lambda = \frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}.$$

- The Rayleigh quotient $\boldsymbol{x}_k^T\boldsymbol{A}\boldsymbol{x}_k/\boldsymbol{x}_k^T\boldsymbol{x}_k$ gives a better approximation

4.Rayleigh Quotient Iteration

Given arbitrary nonzero vector x_0, s.t. $\|x_0\|_2 = 1$;
Calculate $\rho_0 = \frac{x_0^T A x_0}{x_0^T x_0}$, and specify TOL as the tolerance;
for $k = 1, \cdots$ *until* $\|Ax_{i+1} - \rho_{i+1} x_{i+1}\|_2 < TOL$ **do**
  $\sigma_k = x_{k-1}^T A x_{k-1} / \boldsymbol{x}_{k-1}^T \boldsymbol{x}_{k-1}$;
  Solve $(A - \sigma_k I)\boldsymbol{y}_k = \boldsymbol{x}_{k-1}$ for $\boldsymbol{y}_k$;
  $\boldsymbol{x}_k = \boldsymbol{y}_k / \|\boldsymbol{y}_k\|_\infty$;
end

- 该算法等同于逆迭代算法中位移取为瑞利商$\rho(x, A) := \frac{x^T A x}{x^T x}$
- 采用初值$x_0 = [0, \cdots, 0, 1]^T$时与QR迭代得到的序列相同
- 对单重特征值的计算是局部立方收敛

4.Rayleigh Quotient Iteration

Given arbitrary nonzero vector x_0, s.t. $\|x_0\|_2 = 1$;
Calculate $\rho_0 = \frac{x_0^T A x_0}{x_0^T x_0}$, and specify TOL as the tolerance;
for $k = 1, \cdots$ *until* $\|Ax_{i+1} - \rho_{i+1} x_{i+1}\|_2 < TOL$ **do**
  $\sigma_k = x_{k-1}^T A x_{k-1} / \boldsymbol{x}_{k-1}^T \boldsymbol{x}_{k-1}$;
  Solve $(A - \sigma_k I)\boldsymbol{y}_k = \boldsymbol{x}_{k-1}$ for $\boldsymbol{y}_k$;
  $\boldsymbol{x}_k = \boldsymbol{y}_k / \|\boldsymbol{y}_k\|_\infty$;
end

- 该算法等同于逆迭代算法中位移取为瑞利商$\rho(x, A) := \frac{x^T A x}{x^T x}$
- 采用初值$x_0 = [0, \cdots, 0, 1]^T$时与QR迭代得到的序列相同
- 对单重特征值的计算是局部立方收敛

Example (Rayleigh Quotient Iteration)

Using the matrix

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

and a randomly chosen starting vector $\boldsymbol{x}_0$, Rayleigh quotient iteration converges to the accuracy shown in only two iterations:

k	$\boldsymbol{x}_k^T$		σ_k
0	0.807	0.397	3.792
1	0.924	1.000	3.997
2	1.000	1.000	4.000

- It converges to the dominant eigenvalue $\lambda_1 = 4$ much faster.

Example (Rayleigh Quotient Iteration)

Using the matrix

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

and a randomly chosen starting vector $\boldsymbol{x}_0$, Rayleigh quotient iteration converges to the accuracy shown in only two iterations:

k	$\boldsymbol{x}_k^T$		σ_k
0	0.807	0.397	3.792
1	0.924	1.000	3.997
2	1.000	1.000	4.000

- It converges to the dominant eigenvalue $\lambda_1 = 4$ much faster.

Other numerical iterative methods

- 三对角QR迭代：目前是求对称三对角矩阵所有特征值最快($O(n)^2$)的方法($n = 25$以内);Matlab指令eig,LAPACK程序ssyev(稠密)和sstev(三对角阵)
- 二分法和逆迭代(Bisection and inverse iteration)：二分法只求对称三对角阵特征值的一个子集（给定区间）；逆迭代可求相应的特征向量。最坏的情形（许多特征值很接近）不能保证精度；LAPACK程序ssyevx
- 分而治之(Divide-and-conquer)：求解$n > 25$对称三对角矩阵所有特征值和特征向量的最快方法 ($平均O(n)^{2.3}$，最坏$O(n)^3$)；LAPACK程序sstevd
- Jacobi方法：求特征值问题最古老的方法(1846年)；可反复利用Givens变换实现；通常比任何方法都慢($O(n^3)$)变换实现；但结果更精确；

Other numerical iterative methods

- 三对角QR迭代：目前是求对称三对角矩阵**所有**特征值最快($O(n)^2$)的方法($n = 25$以内);Matlab指令eig,LAPACK程序ssyev(稠密)和sstev(三对角阵)
- 二分法和逆迭代(Bisection and inverse iteration)：二分法只求对称三对角阵特征值的一个子集（给定区间）；逆迭代可求相应的特征向量。最坏的情形（许多特征值很接近）不能保证精度；LAPACK程序ssyevx
- 分而治之(Divide-and-conquer)：求解$n > 25$对称三对角矩阵所有特征值和特征向量的最快方法
($平均O(n)^{2.3}$，最坏$O(n)^3$)；LAPACK程序sstevd
- Jacobi方法：求特征值问题最古老的方法(1846年)；可反复利用Givens变换实现；通常比任何方法都慢($O(n^3)$)变换实现；但结果更精确；

Other numerical iterative methods

- 三对角QR迭代：目前是求对称三对角矩阵**所有**特征值最快($O(n)^2$)的方法($n = 25$以内);Matlab指令eig,LAPACK程序ssyev(稠密)和sstev(三对角阵)
- 二分法和逆迭代(Bisection and inverse iteration)：二分法只求对称三对角阵特征值的一个子集（给定区间）；逆迭代可求相应的特征向量。最坏的情形（许多特征值很接近）不能保证精度；LAPACK程序ssyevx
- 分而治之(Divide-and-conquer)：求解$n > 25$对称三对角矩阵所有特征值和特征向量的最快方法
  (平均$O(n)^{2.3}$，最坏$O(n)^3$)；LAPACK程序sstevd
- Jacobi方法：求特征值问题最古老的方法(1846年)；可反复利用Givens变换实现；通常比任何方法都慢($O(n^3)$)变换实现；但结果更精确；

Other numerical iterative methods

- 三对角QR迭代：目前是求对称三对角矩阵**所有**特征值最快($O(n)^2$)的方法($n = 25$以内);Matlab指令eig,LAPACK程序ssyev(稠密)和sstev(三对角阵)
- 二分法和逆迭代(Bisection and inverse iteration)：二分法只求对称三对角阵特征值的一个子集（给定区间）；逆迭代可求相应的特征向量。最坏的情形（许多特征值很接近）不能保证精度；LAPACK程序ssyevx
- 分而治之(Divide-and-conquer)：求解$n > 25$对称三对角矩阵所有特征值和特征向量的最快方法
  (平均$O(n)^{2.3}$，最坏$O(n)^3$)；LAPACK程序sstevd
- Jacobi方法：求特征值问题最古老的方法(1846年)；可反复利用Givens变换实现；通常比任何方法都慢($O(n^3)$)变换实现；但结果更精确；

广义特征值问题

求λ和x，满足

$$Ax = \lambda Bx$$

例:在结构振动问题中，A与B分别为刚度矩阵和质量矩阵。其中，特征值表示结构的振动的自然频率、特征向量对于形状。

求解的几类方法

1. 若B非奇异$(B^{-1}A)x = \lambda x$ 或A非奇异$(A^{-1}B)x = \frac{1}{\lambda}x$
2. 若$B = LL^T$对称,则$(L^{-1}AL^{-T})y = \lambda y$ 且$L^T x = y$
3. 其他情况，用QZ算法稳定！

广义特征值问题

求λ和x，满足

$$Ax = \lambda Bx$$

例:在结构振动问题中，A与B分别为刚度矩阵和质量矩阵。其中，特征值表示结构的振动的自然频率、特征向量对于形状。

求解的几类方法

1. 若B非奇异$(B^{-1}A)x = \lambda x$ 或A非奇异$(A^{-1}B)x = \frac{1}{\lambda}x$
2. 若$B = LL^T$对称,则$(L^{-1}AL^{-T})y = \lambda y$ 且$L^Tx = y$
3. 其他情况，用QZ算法稳定！

其他参考教材

关于特征值问题及其求解方法有不少专著

- Yousef Saad: Numerical Methods for Large Eigenvalue Problems(2nd Edition)
- James Demmel, et. al. : Templates for the solution of algebraic eigenvalue problems
- J.H. Wilkinson: The Algebraic Eigenvalue Problem

  Wilkinson Prize for Numerical Software

Example (Over-determined System)

例 3.1　超定方程组　测量员要测量在某个基准点上 3 个山头的高度．首先从基准点观测，测量员测得它们的高度分别为 $x_1=1237\text{ft}$，$x_2=1914\text{ft}$，$x_3=2417\text{ft}$．(ft：英尺)

为进一步确认初始的测量数据，测量员爬上第一座小山，测得第二座小山相对于第一座的高度为 $x_2-x_1=711\text{ft}$，第三座相对于第一座的高度为 $x_3-x_1=1177\text{ft}$．最后，测量员爬上第二座小山，测得第三座小山相对于第二座小山的高度是 $x_3-x_2=475\text{ft}$[redacted]

- To Model/Describe with linear equation system:

$$Ax=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\\-1&1&0\\-1&0&1\\0&-1&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}\approx\begin{bmatrix}1237\\1941\\2417\\711\\1177\\475\end{bmatrix}=b$$

Example (Over-determined System)

例 3.1 超定方程组 测量员要测量在某个基准点上 3 个山头的高度. 首先从基准点观测，测量员测得它们的高度分别为 $x_1=1237\text{ft}$，$x_2=1914\text{ft}$，$x_3=2417\text{ft}$.（ft：英尺）

为进一步确认初始的测量数据，测量员爬上第一座小山，测得第二座小山相对于第一座的高度为 $x_2-x_1=711\text{ft}$，第三座相对于第一座的高度为 $x_3-x_1=1177\text{ft}$. 最后，测量员爬上第二座小山，测得第三座小山相对于第二座小山的高度是 $x_3-x_2=475\text{ft}$[redacted]

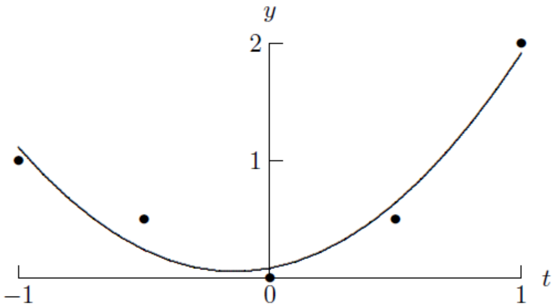
- To Model/Describe with linear equation system:

$$Ax = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \approx \begin{bmatrix} 1237 \\ 1941 \\ 2417 \\ 711 \\ 1177 \\ 475 \end{bmatrix} = b$$

Example (Curve Fitting)

To find a Quadratic Curve, which crossing the given points

t	-1.0	-0.5	0.0	0.5	1.0
y	1.0	0.5	0.0	0.5	2.0

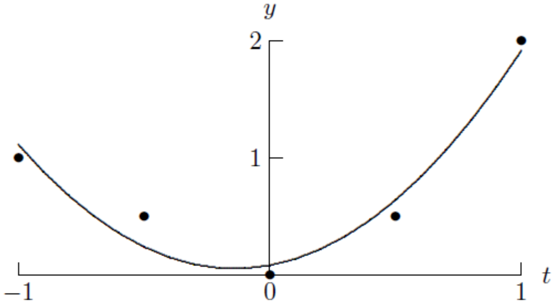


- Similar problem:regression, non-polynomial fitting, etc.

Example (Curve Fitting)

To find a Quadratic Curve, which crossing the given points

t	-1.0	-0.5	0.0	0.5	1.0
y	1.0	0.5	0.0	0.5	2.0



- Similar problem:regression, non-polynomial fitting, etc.

The Problem

Writing the linear system in matrix-vector notation

$$A\mathrm{x} = \mathrm{b},$$

where

- A is an $m \times n$ matrix with $m \gg n$
- b is an $m \times 1$ vector
- x an $n \times 1$ vector

Regularity

1. If *span*(A) is convex set, $\mathrm{r} = \mathrm{b} - A\mathrm{x}$ is convex mapping, then there exists unique solution;
2. The solution to an $m \times n$ least squares problem $Ax \approx b$ is unique if, and only if, A has full column rank, i.e., $\mathrm{rank}(A) = \mathrm{n}$;
3. If $\mathrm{rank}(A) < \mathrm{n}$, then A is said to be *rank-deficient*, and though a solution of the corresponding least squares problem must still exist, it cannot be unique in this case.

Normal Equation - analytic point of view

To minimize L_2 norm of the residual vector $r = b - Ax$, define

$$\phi(x) := \|r\|_2^2 = r^T r = (b - Ax)^T (b - Ax) = b^T b - 2x^T A^T b + x^T A^T Ax,$$

which is minimized by $\nabla\phi(x) = 0$. Then we have

$$\nabla\phi(x) = 2A^T Ax - 2A^T b = 0.$$

That is to solve linear equation system

$$(A^T A)\mathrm{x} = A^T \mathrm{b},$$

where $A^T A$ is

- $n \times n$
- symmetric
- positive defined

Normal Equation - analytic point of view

To minimize L_2 norm of the residual vector $r = b - Ax$, define

$$\phi(x) := \|r\|_2^2 = r^T r = (b - Ax)^T (b - Ax) = b^T b - 2x^T A^T b + x^T A^T Ax,$$

which is minimized by $\nabla\phi(x) = 0$. Then we have

$$\nabla\phi(x) = 2A^T Ax - 2A^T b = 0.$$

That is to solve linear equation system

$$(A^T A)\mathrm{x} = A^T \mathrm{b},$$

where $A^T A$ is

- $n \times n$
- symmetric
- positive defined

Normal Equation - analytic point of view

To minimize L_2 norm of the residual vector $r = b - Ax$, define

$$\phi(x) := \|r\|_2^2 = r^T r = (b - Ax)^T (b - Ax) = b^T b - 2x^T A^T b + x^T A^T Ax,$$

which is minimized by $\nabla\phi(x) = 0$. Then we have

$$\nabla\phi(x) = 2A^T Ax - 2A^T b = 0.$$

That is to solve linear equation system

$$(A^T A)\mathrm{x} = A^T \mathrm{b},$$

where $A^T A$ is

- $n \times n$
- symmetric
- positive defined

Normal Equation - analytic point of view

To minimize L_2 norm of the residual vector $r = b - Ax$, define

$$\phi(x) := \|r\|_2^2 = r^T r = (b - Ax)^T (b - Ax) = b^T b - 2x^T A^T b + x^T A^T Ax,$$

which is minimized by $\nabla\phi(x) = 0$. Then we have

$$\nabla\phi(x) = 2A^T Ax - 2A^T b = 0.$$

That is to solve linear equation system

$$(A^T A)\mathrm{x} = A^T \mathrm{b},$$

where $A^T A$ is

- $n \times n$
- symmetric
- positive defined

Solution to Example 1

The normal equation is

$$A^T A\mathbf{x} = \begin{bmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -651 \\ 2177 \\ 4069 \end{bmatrix} = A^T\mathbf{b}.$$

- $\mathrm{x}^T = [1236, 1943, 2416]^T$,
- $\|\mathrm{r}\|_2^2 = 35.$

Solution to Example 1

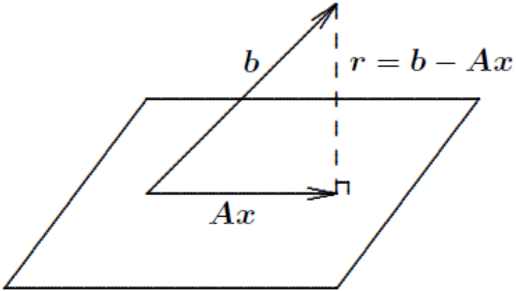
The normal equation is

$$A^T A\mathrm{x} = \begin{bmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -651 \\ 2177 \\ 4069 \end{bmatrix} = A^T\mathrm{b}.$$

- $\mathrm{x}^T = [1236, 1943, 2416]^T$,
- $\|\mathrm{r}\|_2^2 = 35.$

Orthogonal Projection

For case $m > n$ and b $\notin$ $span(A)$, it is clear that r $\perp$ $(Ax - \text{b})$, which is equal to



$$A^T(Ax - \text{b}) = 0.$$

Sensitivity(敏感性) and Condition Number(条件数)

Pseudo inverse of A

$$A^{+} = \left(A^{T}A\right)^{-1}A^{T},$$

where

- $A^{+}A = I$
- $P = AA^{+}$ is an orthogonal projector onto $\mathrm{span}(A)$
- the least square solution is given by $x = A^{+}b$
- the condition number is

$$\mathrm{cond}(A) = \|A\|_2 \cdot \left\|A^{+}\right\|_2 .$$

  - 体现了与秩亏损矩阵的接近程度
  - 正规方程组的条件数与原问题条件数几乎成平方关系！

Sensitivity(敏感性) and Condition Number(条件数)

Pseudo inverse of A

$$A^+ = \left(A^T A\right)^{-1} A^T,$$

where

- $A^+ A = I$
- $P = AA^+$ is an orthogonal projector onto $\mathrm{span}(A)$
- the least square solution is given by $x = A^+ b$
- the condition number is

$$\mathrm{cond}(A) = \|A\|_2 \cdot \left\|A^+\right\|_2 .$$

  - 体现了与秩亏损矩阵的接近程度
  - 正规方程组的条件数与原问题条件数几乎成平方关系！

RHS Perturbation for solving $A^T A\mathrm{x} = A^T \mathrm{b}$

A perturbation of Right Hand Side(RHS) $b + \delta b$ causes

$$A^T A(\mathrm{x} + \delta\mathrm{x}) = A^T(\mathrm{b} + \delta\mathrm{b}).$$

Noticing that $A^T Ax = A^T b$ and by trivial calculation, we have

$$\frac{\|\delta\mathrm{x}\|_2}{\|\mathrm{x}\|_2} \le \cdots \le cond(A)\left(\frac{\|\mathrm{b}\|_2}{\|A\mathrm{x}\|_2}\right)\left(\frac{\|\delta\mathrm{b}\|_2}{\|\mathrm{b}\|_2}\right)$$

Considering Example 1

$$cond(A) = \|A\|_2 \cdot \|A^+\|_2 = 2 \cdot 1 = 2$$

$$\cos(\theta) = \frac{\|\mathrm{b}\|_2}{\|A\mathrm{x}\|_2} \approx 0.99999868.$$

Both the condition number and the angle θ is small, so it is *well-conditioned* !

RHS Perturbation for solving $A^TAx = A^Tb$

A perturbation of Right Hand Side(RHS) $b + \delta b$ causes

$$A^TA(x + \delta x) = A^T(b + \delta b).$$

Noticing that $A^TAx = A^Tb$ and by trivial calculation, we have

$$\frac{\|\delta x\|_2}{\|x\|_2} \le \cdots \le cond(A) \left(\frac{\|b\|_2}{\|Ax\|_2}\right) \left(\frac{\|\delta b\|_2}{\|b\|_2}\right)$$

Considering Example 1

$$cond(A) = \|A\|_2 \cdot \|A^+\|_2 = 2 \cdot 1 = 2$$

$$\cos(\theta) = \frac{\|b\|_2}{\|Ax\|_2} \approx 0.99999868.$$

Both the condition number and the angle θ is small, so it is *well-conditioned* !

RHS Perturbation for solving $A^T A\mathrm{x} = A^T\mathrm{b}$

A perturbation of Right Hand Side(RHS) $b + \delta b$ causes

$$A^T A(\mathrm{x} + \delta\mathrm{x}) = A^T(\mathrm{b} + \delta\mathrm{b}).$$

Noticing that $A^T Ax = A^T b$ and by trivial calculation, we have

$$\frac{\|\delta\mathrm{x}\|_2}{\|\mathrm{x}\|_2} \leq \cdots \leq cond(A)\left(\frac{\|\mathrm{b}\|_2}{\|A\mathrm{x}\|_2}\right)\left(\frac{\|\delta\mathrm{b}\|_2}{\|\mathrm{b}\|_2}\right)$$

Considering Example 1

$$cond(A) = \|A\|_2 \cdot \|A^+\|_2 = 2 \cdot 1 = 2$$

$$\cos(\theta) = \frac{\|\mathrm{b}\|_2}{\|A\mathrm{x}\|_2} \approx 0.99999868.$$

Both the condition number and the angle θ is small, so it is *well-conditioned* !

RHS Perturbation for solving $A^T Ax = A^T b$

A perturbation of Right Hand Side(RHS) $b + \delta b$ causes

$$A^T A(x + \delta x) = A^T (b + \delta b).$$

Noticing that $A^T Ax = A^T b$ and by trivial calculation, we have

$$\frac{\|\delta x\|_2}{\|x\|_2} \leq \cdots \leq cond(A) \left(\frac{\|b\|_2}{\|Ax\|_2} \right) \left(\frac{\|\delta b\|_2}{\|b\|_2} \right)$$

Considering Example 1

$$cond(A) = \|A\|_2 \cdot \|A^+\|_2 = 2 \cdot 1 = 2$$

$$\cos(\theta) = \frac{\|b\|_2}{\|Ax\|_2} \approx 0.99999868.$$

Both the condition number and the angle θ is small, so it is *well-conditioned* !

Example (Sensitivity and Conditioning)

An pseudoinverse of the previous example is given by

$$A^{+} = \left(A^{T}A\right)^{-1}A^{T} = \frac{1}{4}\begin{bmatrix} 2 & 1 & 1 & -1 & -1 & 0 \\ 1 & 2 & 1 & 1 & 0 & -1 \\ 1 & 1 & 2 & 0 & 1 & 1 \end{bmatrix}.$$

The matrix norms can be computed to obtain

$$\|A\|_2 = 2, \left\|A^{+}\right\|_2 = 1, \quad \mathrm{cond}(A) = \|A\|_2 \cdot \left\|A^{+}\right\|_2 = 2.$$

From the ratio

$$\cos(\theta) = \frac{\|b\|_2}{\|Ax\|_2} \approx 0.99999868,$$

so that the angle θ between b and y is about 0.001625, and it is well-conditioned.

Example (Sensitivity and Conditioning)

An pseudoinverse of the previous example is given by

$$A^+ = \left(A^T A\right)^{-1} A^T = \frac{1}{4}\begin{bmatrix} 2 & 1 & 1 & -1 & -1 & 0 \\ 1 & 2 & 1 & 1 & 0 & -1 \\ 1 & 1 & 2 & 0 & 1 & 1 \end{bmatrix}.$$

The matrix norms can be computed to obtain

$$\|A\|_2 = 2, \left\|A^+\right\|_2 = 1, \quad \text{cond}(A) = \|A\|_2 \cdot \left\|A^+\right\|_2 = 2.$$

From the ratio

$$\cos(\theta) = \frac{\|b\|_2}{\|Ax\|_2} \approx 0.99999868,$$

so that the angle θ between b and y is about 0.001625, and it is well-conditioned.

Example (Sensitivity and Conditioning)

An pseudoinverse of the previous example is given by

$$A^+ = \left(A^T A\right)^{-1} A^T = \frac{1}{4}\begin{bmatrix} 2 & 1 & 1 & -1 & -1 & 0 \\ 1 & 2 & 1 & 1 & 0 & -1 \\ 1 & 1 & 2 & 0 & 1 & 1 \end{bmatrix}.$$

The matrix norms can be computed to obtain

$$\|A\|_2 = 2, \left\|A^+\right\|_2 = 1, \quad \mathrm{cond}(A) = \|A\|_2 \cdot \left\|A^+\right\|_2 = 2.$$

From the ratio

$$\cos(\theta) = \frac{\|\mathrm{b}\|_2}{\|A\mathrm{x}\|_2} \approx 0.99999868,$$

so that the angle θ between b and y is about 0.001625, and it is well-conditioned.

Matrix Perturbation for solving $A^T A x = A^T b$

For a perturbed matrix $A + E$, the solution is given by the normal equations

$$(A+E)^T (A+E)(x+\delta x) = (A+E)^T b.$$

By dropping high order term, it is simplified to

$$\delta x \approx \left(A^T A\right)^{-1} E^T r - A^+ E x.$$

Thus,

$$\begin{aligned}\frac{\|\delta x\|_2}{\|x\|_2} &\lessapprox \left\|\left(A^T A\right)^{-1}\right\|_2 \cdot \|E\|_2 \cdot \frac{\|r\|_2}{\|x\|_2} + \left\|A^+\right\|_2 \cdot \|E\|_2 \\ &= [cond(A)]^2 \frac{\|E\|_2}{\|A\|_2} \frac{\|r\|_2}{\|A\|_2 \cdot \|x\|_2} + cond(A) \frac{\|E\|_2}{\|A\|_2} \\ &\leq \left([cond(A)]^2 \frac{\|r\|_2}{\|Ax\|_2} + cond(A)\right) \frac{\|E\|_2}{\|A\|_2} \\ &= \left([cond(A)]^2 \tan(\theta) + cond(A)\right) \frac{\|E\|_2}{\|A\|_2}.\end{aligned}$$

Matrix Perturbation for solving $A^TAx = A^Tb$

For a perturbed matrix $A+E$, the solution is given by the normal equations

$$(A+E)^T(A+E)(x+\delta x) = (A+E)^T b.$$

By dropping high order term, it is simplified to

$$\delta x \approx \left(A^TA\right)^{-1} E^T r - A^+ Ex.$$

Thus,

$$\begin{aligned}
\frac{\|\delta x\|_2}{\|x\|_2} &\lessapprox \left\|\left(A^TA\right)^{-1}\right\|_2 \cdot \|E\|_2 \cdot \frac{\|r\|_2}{\|x\|_2} + \left\|A^+\right\|_2 \cdot \|E\|_2 \\
&= \left[cond\,(A)\right]^2 \frac{\|E\|_2}{\|A\|_2} \frac{\|r\|_2}{\|A\|_2 \cdot \|x\|_2} + cond(A)\frac{\|E\|_2}{\|A\|_2} \\
&\leq \left(\left[cond\,(A)\right]^2 \frac{\|r\|_2}{\|Ax\|_2} + cond(A)\right)\frac{\|E\|_2}{\|A\|_2} \\
&= \left(\left[cond\,(A)\right]^2 \tan(\theta) + cond(A)\right)\frac{\|E\|_2}{\|A\|_2}.
\end{aligned}$$

Matrix Perturbation for solving $A^TAx = A^Tb$

For a perturbed matrix $A+E$, the solution is given by the normal equations

$$(A+E)^T(A+E)(x+\delta x) = (A+E)^T b.$$

By dropping high order term, it is simplified to

$$\delta x \approx \left(A^TA\right)^{-1} E^T r - A^+ E x.$$

Thus,

$$\begin{aligned}\frac{\|\delta x\|_2}{\|x\|_2} &\lessapprox \left\|\left(A^TA\right)^{-1}\right\|_2 \cdot \|E\|_2 \cdot \frac{\|r\|_2}{\|x\|_2} + \|A^+\|_2 \cdot \|E\|_2 \\ &= \left[cond\left(A\right)\right]^2 \frac{\|E\|_2}{\|A\|_2} \frac{\|r\|_2}{\|A\|_2 \cdot \|x\|_2} + cond(A)\frac{\|E\|_2}{\|A\|_2} \\ &\le \left(\left[cond\left(A\right)\right]^2 \frac{\|r\|_2}{\|Ax\|_2} + cond(A)\right)\frac{\|E\|_2}{\|A\|_2} \\ &= \left(\left[cond\left(A\right)\right]^2 \tan\left(\theta\right) + cond(A)\right)\frac{\|E\|_2}{\|A\|_2}.\end{aligned}$$

Matrix Perturbation for solving $A^TAx = A^Tb$

For a perturbed matrix $A+E$, the solution is given by the normal equations

$$(A+E)^T(A+E)(x+\delta x) = (A+E)^T b.$$

By dropping high order term, it is simplified to

$$\delta x \approx \left(A^TA\right)^{-1}E^Tr - A^+Ex.$$

Thus,

$$\begin{aligned}
\frac{\|\delta x\|_2}{\|x\|_2} &\lessapprox \left\|\left(A^TA\right)^{-1}\right\|_2 \cdot \|E\|_2 \cdot \frac{\|r\|_2}{\|x\|_2} + \|A^+\|_2 \cdot \|E\|_2 \\
&= \left[cond\,(A)\right]^2 \frac{\|E\|_2}{\|A\|_2} \frac{\|r\|_2}{\|A\|_2 \cdot \|x\|_2} + cond(A)\frac{\|E\|_2}{\|A\|_2} \\
&\le \left(\left[cond\,(A)\right]^2 \frac{\|r\|_2}{\|Ax\|_2} + cond(A)\right)\frac{\|E\|_2}{\|A\|_2} \\
&= \left(\left[cond\,(A)\right]^2 \tan(\theta) + cond(A)\right)\frac{\|E\|_2}{\|A\|_2}.
\end{aligned}$$

Example (Condition-Squaring Effect)

$$A = \begin{bmatrix} 1 & 1 \\ \epsilon & -\epsilon \\ 0 & 0 \end{bmatrix}, E = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\epsilon & \epsilon \end{bmatrix},$$

where $\epsilon \ll 1$, say around $\sqrt{\epsilon_{mach}}$, for which we have

$$cond(A) = 1/\epsilon, \|E\|_2 / \|A\|_2 = \epsilon.$$

- If $b = [1, 0, \epsilon]^T$, then $\|\delta x\|_2 / \|x\|_2 = 0.5$. There is no condition-squaring effect, since the residual is small and $\tan(\theta) \approx \epsilon$. Suppressive !
- If $b = [1, 0, 1]^T$,then we have $\|\delta x\|_2 / \|x\|_2 = 0.5/\epsilon$. Perturbation is about equal to $[\text{cond}(A)]^2$ times the relative perturbation in A, and the norm of the residual is about 1 and $\tan(\theta) \approx 1$. Sensitive !

Example (Condition-Squaring Effect)

$$A = \begin{bmatrix} 1 & 1 \\ \epsilon & -\epsilon \\ 0 & 0 \end{bmatrix}, E = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\epsilon & \epsilon \end{bmatrix},$$

where $\epsilon \ll 1$, say around $\sqrt{\epsilon_{mach}}$, for which we have

$$cond(A) = 1/\epsilon, \|E\|_2 / \|A\|_2 = \epsilon.$$

- If $b = [1, 0, \epsilon]^T$, then $\|\delta x\|_2 / \|x\|_2 = 0.5$. There is no condition-squaring effect, since the residual is small and $\tan(\theta) \approx \epsilon$. Suppressive !
- If $b = [1, 0, 1]^T$, then we have $\|\delta x\|_2 / \|x\|_2 = 0.5/\epsilon$. Perturbation is about equal to $[\text{cond}(A)]^2$ times the relative perturbation in A, and the norm of the residual is about 1 and $\tan(\theta) \approx 1$. Sensitive !

Example (Condition-Squaring Effect)

$$A = \begin{bmatrix} 1 & 1 \\ \epsilon & -\epsilon \\ 0 & 0 \end{bmatrix}, E = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\epsilon & \epsilon \end{bmatrix},$$

where $\epsilon \ll 1$, say around $\sqrt{\epsilon_{mach}}$, for which we have

$$cond(A) = 1/\epsilon, \|E\|_2 / \|A\|_2 = \epsilon.$$

- If $b = [1, 0, \epsilon]^T$, then $\|\delta x\|_2 / \|x\|_2 = 0.5$. There is no condition-squaring effect, since the residual is small and $\tan(\theta) \approx \epsilon$. Suppressive !
- If $b = [1, 0, 1]^T$,then we have $\|\delta x\|_2 / \|x\|_2 = 0.5/\epsilon$. Perturbation is about equal to $[\text{cond}(A)]^2$ times the relative perturbation in A, and the norm of the residual is about 1 and $\tan(\theta) \approx 1$. Sensitive !

Practical issues for numerical calculation

To solve the linear least square problem, one can convert the linear least square problem into

1. Augment Linear System
2. Triangular Least Squares Problems

1.Augment Linear System(化矩为方)

The solution x and the residual r satisfying

$$\begin{aligned} r + Ax &= b \\ A^T r &= 0 \end{aligned}$$

In the matrix form $$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ 0 \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

Introduce a scaling parameter α for the residual, it yields

$$\begin{bmatrix} \alpha I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r/\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

- Generally, $\alpha = \max_{i,j} |a_{i,j}|/1000$ 。
- Be prohibitive in cost (proportional to $(m+n)^3$)
- Effectively in MATLAB for large sparse linear least squares problems

1.Augment Linear System(化矩为方)

The solution x and the residual r satisfying

$$\begin{aligned} \mathrm{r} + A\mathrm{x} &= \mathrm{b} \\ A^T\mathrm{r} &= 0 \end{aligned}$$

In the matrix form
$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathrm{r} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathrm{b} \\ 0 \end{bmatrix}.$$

Introduce a scaling parameter α for the residual, it yields

$$\begin{bmatrix} \alpha I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathrm{r}/\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} \mathrm{b} \\ 0 \end{bmatrix}.$$

- Generally, $\alpha = \max_{i,j} |a_{i,j}|/1000$。
- Be prohibitive in cost (proportional to $(m+n)^3$)
- Effectively in MATLAB for large sparse linear least squares problems

1.Augment Linear System(化矩为方)

The solution x and the residual r satisfying

$$\begin{aligned} \mathrm{r} + A\mathrm{x} &= \mathrm{b} \\ A^T\mathrm{r} &= 0 \end{aligned}$$

In the matrix form
$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathrm{r} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathrm{b} \\ 0 \end{bmatrix}.$$

Introduce a scaling parameter α for the residual, it yields

$$\begin{bmatrix} \alpha I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathrm{r}/\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} \mathrm{b} \\ 0 \end{bmatrix}.$$

- Generally, $\alpha = \max_{i,j} |a_{i,j}|/1000$。
- Be prohibitive in cost (proportional to $(m+n)^3$)
- Effectively in MATLAB for large sparse linear least squares problems

1.Augment Linear System(化矩为方)

The solution x and the residual r satisfying

$$
\begin{aligned}
\mathrm{r} + A\mathrm{x} &= \mathrm{b} \\
A^T \mathrm{r} &= 0
\end{aligned}
$$

In the matrix form
$$
\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathrm{r} \\ 0 \end{bmatrix} = \begin{bmatrix} \mathrm{b} \\ 0 \end{bmatrix}.
$$

Introduce a scaling parameter α for the residual, it yields

$$
\begin{bmatrix} \alpha I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathrm{r}/\alpha \\ 0 \end{bmatrix} = \begin{bmatrix} \mathrm{b} \\ 0 \end{bmatrix}.
$$

- Generally, $\alpha = \max_{i,j} |a_{i,j}|/1000$。
- Be prohibitive in cost (proportional to $(m+n)^3$)
- Effectively in MATLAB for large sparse linear least squares problems

2. Triangular Least Squares Problems

In the over-determined case, $m > n$, such a problem has the form

$$\begin{bmatrix} R \\ O \end{bmatrix} x \approx \begin{bmatrix} c_1 \\ c_2 \end{bmatrix},$$

where R is $n \times n$ and upper triangular. The least squares residual is then given by

$$\|r\|_2^2 = \|c_1 - Rx\|_2^2 + \|c_2\|_2^2 \,.$$

We have no control over the second term $\|c_2\|_2^2$, however, the first term can be forced to be zero by choosing x, s.t.

$$Rx = c_1,$$

which can be solved simply by back-substitution。 Therefore

$$\|r\|_2^2 = \|c_2\|_2^2 \,.$$

2. Triangular Least Squares Problems

In the over-determined case, $m > n$, such a problem has the form

$$\begin{bmatrix} R \\ O \end{bmatrix} x \approx \begin{bmatrix} c_1 \\ c_2 \end{bmatrix},$$

where R is $n \times n$ and upper triangular. The least squares residual is then given by

$$\|r\|_2^2 = \|c_1 - Rx\|_2^2 + \|c_2\|_2^2 .$$

We have no control over the second term $\|c_2\|_2^2$, however, the first term can be forced to be zero by choosing x, s.t.

$$Rx = c_1,$$

which can be solved simply by back-substitution。 Therefore

$$\|r\|_2^2 = \|c_2\|_2^2 .$$

QR Factorization

Theorem

For an $m \times n$ matrix A with $m > n$, has the form

$$A = Q \begin{bmatrix} R \\ O \end{bmatrix},$$

If we partition Q as $Q = [Q_1, Q_2]$, where Q_1 contains the first n columns and Q_2 contains the remaining $m - n$ columns of Q, then we have

$$A = Q \begin{bmatrix} R \\ O \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ O \end{bmatrix} = Q_1 R.$$

- It is sometimes called the *reduced*, or "economy size" QR factorization of A.

QR Factorization

Theorem

For an $m \times n$ matrix A with $m > n$, has the form

$$A = Q \begin{bmatrix} R \\ O \end{bmatrix},$$

If we partition Q as $Q = [Q_1, Q_2]$, where Q_1 contains the first n columns and Q_2 contains the remaining $m - n$ columns of Q, then we have

$$A = Q \begin{bmatrix} R \\ O \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ O \end{bmatrix} = Q_1 R.$$

- It is sometimes called the *reduced*, or "economy size" QR factorization of A.

QR Factorization - conti.

- Indeed, let $A = [\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_n]$ in a column-wise form, and its orthogonal operation yields $Q[\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_n]$，记$r_{ji} = \mathrm{q}_j^T \mathrm{a}_i$, that is

$$\mathrm{a}_i = \sum_{j=1}^{i} r_{ji} \mathrm{q}_j,$$

- If A has full column rank, then R is nonsingular. The columns of Q_1 form an orthonormal basis for span(A) and Q_2 for its orthogonal complement span$(A)^{\perp}$.
- Practically, Givens rotation, Householder rotation or Gram-Schmidt orthogonalization are both works for this purpose, i.e.,

$$\gg [\mathrm{Q},\mathrm{R}] = \mathrm{qr}(\mathrm{A});$$

QR Factorization - conti.

- Indeed, let $A = [\mathrm{a}_1, \mathrm{a}_2, \ldots, \mathrm{a}_n]$ in a column-wise form, and its orthogonal operation yields $Q[\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_n]$，记$r_{ji} = \mathrm{q}_j^T \mathrm{a}_i$, that is

$$\mathrm{a}_i = \sum_{j=1}^{i} r_{ji} \mathrm{q}_j,$$

- If A has full column rank, then R is nonsingular. The columns of Q_1 form an orthonormal basis for span(A) and Q_2 for its orthogonal complement span$(A)^\perp$.
- Practically, Givens rotation, Householder rotation or Gram-Schmidt orthogonalization are both works for this purpose, i.e.,

» [Q,R] = qr(A);

Example (Example for QR Factorization)

$$
\begin{aligned}
A &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & \frac{3}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & 0 & \frac{2}{\sqrt{3}} \end{bmatrix} := QR
\end{aligned}
$$

4. Eigenvalue Problem

The problem

Numerical Method

5. Linear Least Square

The Problem

Numerical Method

6. Matrix Factorization

Orthogonalization

Singular Value Decomposition

Applications

Orthogonalization Methods

Approaches to computing the QR factorization of a matrix are similar to LU factorization. Eventually, we will use orthogonal transformations rather than elementary elimination matrices so that the Euclidean norm will be preserved.

Several classical orthogonalization methods are commonly used, including

1. Householder transformations (elementary reflectors)
2. Givens transformations (plane rotations)
3. Gram-Schmidt orthogonalization

Orthogonalization Methods

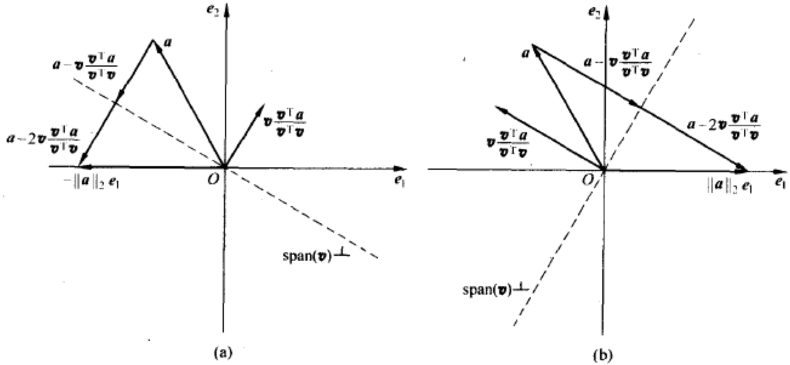
Approaches to computing the QR factorization of a matrix are similar to LU factorization. Eventually, we will use orthogonal transformations rather than elementary elimination matrices so that the Euclidean norm will be preserved. Several classical orthogonalization methods are commonly used, including

1. Householder transformations (elementary reflectors)
2. Givens transformations (plane rotations)
3. Gram-Schmidt orthogonalization

1.Householder transformation

To annihilates(to be 0 by transformation) certain components of a given vector a:

$$Ha = \alpha e_1, \qquad \text{where } H = I - \frac{2}{v^T v} vv^T.$$



- Geometrically, it could be done by taking $v = a - \|a\| e_1$

More explanation

Choose v, such that all the components of a except the first are annihilated, i.e.,

$$Ha = [\alpha, 0, \cdots, 0]^T = \alpha[1, 0, \cdots, 0]^T = \alpha e_1.$$

Remembering $H = H^T = H^{-1}$, we have

$$\alpha e_1 = Ha\left(I - 2\frac{vv^T}{v^Tv}\right)a = a - 2v\frac{v^Ta}{v^Tv},$$

Hence

$$v = (a - \alpha e_1)\frac{v^Tv}{2v^Ta}.$$

- Since the scalar factor is irrelevant in determining v, it holds that

$$v = a - \alpha e_1.$$

- $\alpha = \pm\|a\|_2$. To preserve the norm, i.e.,$\alpha = -\operatorname{sign}(a)_2$, which leads to

$$v = a - \|a\|_2 e_1$$

More explanation

Choose v, such that all the components of a except the first are annihilated, i.e.,

$$Ha = [\alpha, 0, \cdots, 0]^T = \alpha[1, 0, \cdots, 0]^T = \alpha e_1.$$

Remembering $H = H^T = H^{-1}$, we have

$$\alpha e_1 = Ha \left(I - 2\frac{vv^T}{v^T v} \right) a = a - 2v\frac{v^T a}{v^T v},$$

Hence

$$v = (a - \alpha e_1)\frac{v^T v}{2v^T a}.$$

- Since the scalar factor is irrelevant in determining v, it holds that

$$v = a - \alpha e_1.$$

- $\alpha = \pm\|a\|_2$. To preserve the norm, i.e.,$\alpha = -\operatorname{sign}(a)_2$, which leads to

$$v = a - \|a\|_2 e_1$$

More explanation

Choose v, such that all the components of a except the first are annihilated, i.e.,

$$Ha = [\alpha, 0, \cdots, 0]^T = \alpha[1, 0, \cdots, 0]^T = \alpha e_1.$$

Remembering $H = H^T = H^{-1}$, we have

$$\alpha e_1 = Ha \left(I - 2\frac{vv^T}{v^T v} \right) a = a - 2v\frac{v^T a}{v^T v},$$

Hence

$$v = (a - \alpha e_1) \frac{v^T v}{2v^T a}.$$

- Since the scalar factor is irrelevant in determining v, it holds that

$$v = a - \alpha e_1.$$

- $\alpha = \pm \|a\|_2$. To preserve the norm, i.e.,$\alpha = -\operatorname{sign}(a)_2$, which leads to

$$v = a - \|a\|_2 e_1$$

More explanation

Choose v, such that all the components of a except the first are annihilated, i.e.,

$$Ha = [\alpha, 0, \cdots, 0]^T = \alpha[1, 0, \cdots, 0]^T = \alpha e_1.$$

Remembering $H = H^T = H^{-1}$, we have

$$\alpha e_1 = Ha\left(I - 2\frac{vv^T}{v^Tv}\right)a = a - 2v\frac{v^Ta}{v^Tv},$$

Hence

$$v = (a - \alpha e_1)\frac{v^Tv}{2v^Ta}.$$

- Since the scalar factor is irrelevant in determining v, it holds that

$$v = a - \alpha e_1.$$

- $\alpha = \pm\|a\|_2$. To preserve the norm, i.e.,$\alpha = -\operatorname{sign}(a)_2$, which leads to

$$v = a - \|a\|_2 e_1$$

More explanation

Choose v, such that all the components of a except the first are annihilated, i.e.,

$$Ha = [\alpha, 0, \cdots, 0]^T = \alpha[1, 0, \cdots, 0]^T = \alpha e_1.$$

Remembering $H = H^T = H^{-1}$, we have

$$\alpha e_1 = Ha \left(I - 2\frac{vv^T}{v^T v} \right) a = a - 2v\frac{v^T a}{v^T v},$$

Hence

$$v = (a - \alpha e_1)\frac{v^T v}{2v^T a}.$$

- Since the scalar factor is irrelevant in determining v, it holds that

$$v = a - \alpha e_1.$$

- $\alpha = \pm \|a\|_2$. To preserve the norm, i.e.,$\alpha = -\operatorname{sign}(a)_2$, which leads to

$$v = a - \|a\|_2 e_1$$

Example (Householder Transformation.)

To illustrate the construction just described, we determine a Householder transformation that annihilates all but the first component of the vector

$$a = [2, 1, 2]^T.$$

- Following the foregoing recipe, we choose the vector

$$\mathrm{v} = \mathrm{a} - \alpha \mathrm{e}_1 = [2, 1, 2]^T - (-3) \times [1, 0, 0]^T = [5, 1, 2]^T$$

- To confirm that the Householder transformation performs as expected,

$$Ha = a - 2\frac{v^T a}{v^T v} v = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 2 \times \frac{15}{30} \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix},$$

- There is *no need to* calculate the matrix H explicitly!

Example (Householder Transformation.)

To illustrate the construction just described, we determine a Householder transformation that annihilates all but the first component of the vector

$$a = [2, 1, 2]^T.$$

- Following the foregoing recipe, we choose the vector

$$\mathrm{v} = \mathrm{a} - \alpha \mathrm{e}_1 = [2, 1, 2]^T - (-3) \times [1, 0, 0]^T = [5, 1, 2]^T$$

- To confirm that the Householder transformation performs as expected,

$$Ha = a - 2\frac{v^T a}{v^T v} v = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 2 \times \frac{15}{30} \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix},$$

- There is *no need to* calculate the matrix H explicitly!

Example (Householder Transformation.)

To illustrate the construction just described, we determine a Householder transformation that annihilates all but the first component of the vector

$$a = [2, 1, 2]^T.$$

- Following the foregoing recipe, we choose the vector

$$\mathrm{v} = \mathrm{a} - \alpha \mathrm{e}_1 = [2, 1, 2]^T - (-3) \times [1, 0, 0]^T = [5, 1, 2]^T$$

- To confirm that the Householder transformation performs as expected,

$$Ha = a - 2\frac{v^T a}{v^T v} v = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 2 \times \frac{15}{30} \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix},$$

- There is *no need to* calculate the matrix H explicitly!

Example (Householder Transformation.)

To illustrate the construction just described, we determine a Householder transformation that annihilates all but the first component of the vector

$$a = [2, 1, 2]^T.$$

- Following the foregoing recipe, we choose the vector

$$\mathrm{v} = \mathrm{a} - \alpha \mathrm{e}_1 = [2, 1, 2]^T - (-3) \times [1, 0, 0]^T = [5, 1, 2]^T$$

- To confirm that the Householder transformation performs as expected,

$$Ha = a - 2\frac{v^T a}{v^T v} v = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 2 \times \frac{15}{30} \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix},$$

- There is *no need to* calculate the matrix H explicitly!

To annihilates only one single component of a given vector $[a_1, a_2]^T$:

$$Ga = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sqrt{a_1^2 + a_2^2} \\ 0 \end{bmatrix},$$

where

$$c = \cos(\theta) = \frac{a_1}{\sqrt{a_1^2 + a_2^2}}, \quad s = \sin(\theta) = \frac{a_2}{\sqrt{a_1^2 + a_2^2}}$$

- G is a orthogonal matrix, so that it keeps $\|\cdot\|_2$ unchanged.
- Could be applied to implement QR factorization.
- Require more computational effort and memory consuming than H-Rotation.

2.Givens Rotation

To annihilates only one single component of a given vector $[a_1, a_2]^T$:

$$Ga = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sqrt{a_1^2 + a_2^2} \\ 0 \end{bmatrix},$$

where

$$c = \cos(\theta) = \frac{a_1}{\sqrt{a_1^2 + a_2^2}}, \quad s = \sin(\theta) = \frac{a_2}{\sqrt{a_1^2 + a_2^2}}$$

- G is a orthogonal matrix, so that it keeps $\|\cdot\|_2$ unchanged.
- Could be applied to implement QR factorization.
- Require more computational effort and memory consuming than H-Rotation.

Example (Givens Rotation)

To illustrate the construction just described, we determine a Givens rotation that annihilates the second component of the vector $[4, 3]^T$

For this problem, we can safely compute the cosine and sine directly, obtaining

$$\cos(\theta) = \frac{a_1}{\sqrt{a_1^2 + a_2^2}} = \frac{4}{5} = 0.8, \qquad \sin(\theta) = \frac{a_2}{\sqrt{a_1^2 + a_2^2}} = \frac{3}{5} = 0.6.$$

Thus, the rotation is given by

$$G = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix}.$$

To confirm that the rotation performs as expected, we compute

$$Ga = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix},$$

which shows the 0 component and the L_2-norm is preserved.

Example (Givens Rotation)

To illustrate the construction just described, we determine a Givens rotation that annihilates the second component of the vector $[4, 3]^T$

For this problem, we can safely compute the cosine and sine directly, obtaining

$$\cos(\theta) = \frac{a_1}{\sqrt{a_1^2 + a_2^2}} = \frac{4}{5} = 0.8, \qquad \sin(\theta) = \frac{a_2}{\sqrt{a_1^2 + a_2^2}} = \frac{3}{5} = 0.6.$$

Thus, the rotation is given by

$$G = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix}.$$

To confirm that the rotation performs as expected, we compute

$$Ga = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix},$$

which shows the 0 component and the L_2-norm is preserved.

Example (Givens Rotation)

To illustrate the construction just described, we determine a Givens rotation that annihilates the second component of the vector $[4, 3]^T$

For this problem, we can safely compute the cosine and sine directly, obtaining

$$\cos(\theta) = \frac{a_1}{\sqrt{a_1^2 + a_2^2}} = \frac{4}{5} = 0.8, \qquad \sin(\theta) = \frac{a_2}{\sqrt{a_1^2 + a_2^2}} = \frac{3}{5} = 0.6.$$

Thus, the rotation is given by

$$G = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix}.$$

To confirm that the rotation performs as expected, we compute

$$Ga = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix},$$

which shows the 0 component and the L_2-norm is preserved.

3.Gram-Schmidt Orthogonalization

// 1. Classical G-S for $k = 1 \rightarrow n$ **do**

- $q_k = a_k$;
- **for** $j = 1 \rightarrow k-1$ **do**
    - $r_{jk} = q_j^T a_k$;
    - $q_k = q_k - r_{jk} q_j$;
- **end**
- $r_{kk} = \|q_k\|_2$;
- **if** $r_{kk} == 0$ **then**
    - quit
- **else**
    - $q_k = q_k / r_{kk}$

// 2. Modified G-S ;

for $k = 1 \rightarrow n$ **do**

- $r_{kk} = \|a_k\|_2$;
- **if** $r_{kk} == 0$ **then**
    - quit
- **else**
    - $q_k = a_k / r_{kk}$
- **end**
- **for** $j = k+1 \rightarrow n$ **do**
    - $r_{kj} = q_k^T a_j$;
    - $a_j = a_j - r_{kj} q_k$;

// 3. G-S based QR;

for $i \rightarrow n$ **do**

- $q_i = a_i$;
- **for** $j = 1 \rightarrow i$ **do**
    - $r_{ji} = q_j^T a_i$;
    - $q_i = q_i - r_{ji} q_j$;
- **end**
- $r_{ii} = \|q_i\|_2$;
- **if** $r_{ii} == 0$ **then**
    - quit
- **else**
    - $q_i = q_i / r_{ii}$

3.Gram-Schmidt Orthogonalization

// 1. Classical G-S **for**

$k = 1 \to n$ **do**

  $q_k = a_k$;

  for $j = 1 \to k-1$ **do**

    $r_{jk} = q_j^T a_k$;

    $q_k = q_k - r_{jk} q_j$;

  end

  $r_{kk} = \|q_k\|_2$;

  if $r_{kk} == 0$ **then**

    quit

  else

    $q_k = q_k / r_{kk}$

// 2. Modified G-S ;

for $k = 1 \to n$ **do**

  $r_{kk} = \|a_k\|_2$;

  if $r_{kk} == 0$ **then**

    quit

  else

    $q_k = a_k / r_{kk}$

  end

  for $j = k+1 \to n$

   do

    $r_{kj} = q_k^T a_j$;

    $a_j = a_j - r_{kj} q_k$;

// 3. G-S based QR;

for $i \to n$ **do**

  $q_i = a_i$;

  for $j = 1 \to i$ **do**

    $r_{ji} = q_j^T a_i$;

    $q_i = q_i - r_{ji} q_j$;

  end

  $r_{ii} = \|q_i\|_2$;

  if $r_{ii} == 0$ **then**

    quit

  else

    $q_i = q_i / r_{ii}$

3.Gram-Schmidt Orthogonalization

// 1. Classical G-S for

for $k = 1 \to n$ **do**

$\quad$ $q_k = a_k$;

$\quad$ **for** $j = 1 \to k-1$ **do**

$\quad\quad$ $r_{jk} = q_j^T a_k$;

$\quad\quad$ $q_k = q_k - r_{jk} q_j$;

$\quad$ **end**

$\quad$ $r_{kk} = \|q_k\|_2$;

$\quad$ **if** $r_{kk} == 0$ **then**

$\quad\quad$ quit

$\quad$ **else**

$\quad\quad$ $q_k = q_k / r_{kk}$

// 2. Modified G-S ;

for $k = 1 \to n$ do

$\quad$ $r_{kk} = \|a_k\|_2$;

$\quad$ if $r_{kk} == 0$ then

$\quad\quad$ quit

$\quad$ else

$\quad\quad$ $q_k = a_k / r_{kk}$

$\quad$ end

$\quad$ for $j = k+1 \to n$

$\quad$ do

$\quad\quad$ $r_{kj} = q_k^T a_j$;

$\quad\quad$ $a_j = a_j - r_{kj} q_k$;

// 3. G-S based QR;

for $i \to n$ **do**

$\quad$ $q_i = a_i$;

$\quad$ **for** $j = 1 \to i$ **do**

$\quad\quad$ $r_{ji} = q_j^T a_i$;

$\quad\quad$ $q_i = q_i - r_{ji} q_j$;

$\quad$ **end**

$\quad$ $r_{ii} = \|q_i\|_2$;

$\quad$ **if** $r_{ii} == 0$ **then**

$\quad\quad$ quit

$\quad$ **else**

$\quad\quad$ $q_i = q_i / r_{ii}$

Comments

1. Classical G-S suffers loss of orthogonality in finite-precision arithmetic
2. Separate storage is required for A, Q, and R, since original a_k are needed in inner loop, so q_k cannot overwrite columns of A
3. Both deficiencies are improved by modified Gram-Schmidt, with each vector orthogonalized in turn against all subsequent vectors, so q_k can overwrite a_k
4. Write out the algorithm for QR factorization by modified G-S procedure?

Example (Modified Gram-Schmidt QR Factorization)

We illustrate this by the solution of least squares problem given by A.

Step 1: Normalizing the first column of A:

$$r_{11} = \|a_1\|_2 = 1.7321, \quad q_1 = a_1/r_{11} = [0.5774, 0, 0, -0.5774, -0.5774, 0]^T.$$

and orthogonalizing the first column against the subsequent columns

$$r_{12} = q_1^T a_2 = -0.5774, r_{13} = q_1^T a_3 = -0.5774.$$

Subtracting these multiples of q_1 from the second and third columns, we have

$$A := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix} \Rightarrow \begin{bmatrix} 0.5774 & 0.3333 & 0.3333 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.5774 & 0.6667 & -0.3333 \\ -0.5774 & -0.3333 & 0.6667 \\ \vdots & \vdots & \vdots \end{bmatrix}.$$

Example (Modified Gram-Schmidt QR Factorization)

We illustrate this by the solution of least squares problem given by A.

Step 1: Normalizing the first column of A:

$$r_{11} = \|\mathrm{a}_1\|_2 = 1.7321, \quad \mathrm{q}_1 = \mathrm{a}_1/r_{11} = [0.5774, 0, 0, -0.5774, -0.5774, 0]^T.$$

and orthogonalizing the first column against the subsequent columns

$$r_{12} = \mathrm{q}_1^T \mathrm{a}_2 = -0.5774, r_{13} = \mathrm{q}_1^T \mathrm{a}_3 = -0.5774.$$

Subtracting these multiples of q_1 from the second and third columns, we have

$$A := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0.5774 & 0.3333 & 0.3333 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -0.5774 & 0.6667 & -0.3333 \\ -0.5774 & -0.3333 & 0.6667 \\ 0 & -1 & 1 \end{bmatrix}.$$

Normalizing the second column, we have $r_{22} = \|a_2\|_2 = 1.6330$ and

$$q_2 = a_2/r_{22} = [0.2041, 0.6124, 0, 0.4082, -0.2041, -0.6124]^T.$$

Orthogonalizing the second column against the third column, we obtain

$$r_{23} = q_2^T a_3 = -0.8165.$$

Subtracting this multiple of q_2 from the third column and replacing the second column with q_2, we obtain the transformed matrix

$$\begin{bmatrix} 0.5774 & 0.2041 & 0.5 \\ 0 & 0.6124 & 0.5 \\ 0 & 0 & 1 \\ -0.5774 & -0.4082 & 0 \\ -0.5774 & -0.2041 & 0.5 \\ 0 & -0.6124 & 0.5 \end{bmatrix}.$$

Normalizing the second column, we have $r_{22} = \|a_2\|_2 = 1.6330$ and

$$q_2 = a_2/r_{22} = [0.2041, 0.6124, 0, 0.4082, -0.2041, -0.6124]^T.$$

Orthogonalizing the second column against the third column, we obtain

$$r_{23} = q_2^T a_3 = -0.8165.$$

Subtracting this multiple of q_2 from the third column and replacing the second column with q_2, we obtain the transformed matrix

$$\begin{bmatrix} 0.5774 & 0.2041 & 0.5 \\ 0 & 0.6124 & 0.5 \\ 0 & 0 & 1 \\ -0.5774 & -0.4082 & 0 \\ -0.5774 & -0.2041 & 0.5 \\ 0 & -0.6124 & 0.5 \end{bmatrix}.$$

Finally, normalize the third column $r_{33} = \|a_3\|_2 = 1.4142$ and

$$q_3 = a_3/r_{33} = [0.3536, 0.3536, 0.7071, 0, 0.3536, 0.3536]^T.$$

Replacing the third column with q_3, the reduced QR factorization yields

$$A = \begin{bmatrix} 0.5774 & 0.2041 & 0.3536 \\ 0 & 0.6124 & 0.3536 \\ 0 & 0 & 0.7071 \\ -0.5774 & 0.4082 & 0 \\ -0.5774 & -0.2041 & 0.3536 \\ 0 & -0.6124 & 0.3536 \end{bmatrix} \begin{bmatrix} 1.7321 & -0.5774 & -0.5774 \\ 0 & 1.6330 & -0.8165 \\ 0 & 0 & 1.4142 \end{bmatrix} = Q_1 R.$$

This is well-conditioned problem, it yields

$$Q_1^T b = \begin{bmatrix} -376 \\ 1200 \\ 3417 \end{bmatrix} = c_1.$$

Finally, normalize the third column $r_{33} = \|a_3\|_2 = 1.4142$ and

$$q_3 = a_3/r_{33} = [0.3536, 0.3536, 0.7071, 0, 0.3536, 0.3536]^T.$$

Replacing the third column with q_3, the reduced QR factorization yields

$$A = \begin{bmatrix} 0.5774 & 0.2041 & 0.3536 \\ 0 & 0.6124 & 0.3536 \\ 0 & 0 & 0.7071 \\ -0.5774 & 0.4082 & 0 \\ -0.5774 & -0.2041 & 0.3536 \\ 0 & -0.6124 & 0.3536 \end{bmatrix} \begin{bmatrix} 1.7321 & -0.5774 & -0.5774 \\ 0 & 1.6330 & -0.8165 \\ 0 & 0 & 1.4142 \end{bmatrix} = Q_1 R.$$

This is well-conditioned problem, it yields

$$Q_1^T b = \begin{bmatrix} -376 \\ 1200 \\ 3417 \end{bmatrix} = c_1.$$

Finally, normalize the third column $r_{33} = \|\mathrm{a}_3\|_2 = 1.4142$ and

$$\mathrm{q}_3 = \mathrm{a}_3/r_{33} = [0.3536, 0.3536, 0.7071, 0, 0.3536, 0.3536]^T.$$

Replacing the third column with q_3, the reduced QR factorization yields

$$A = \begin{bmatrix} 0.5774 & 0.2041 & 0.3536 \\ 0 & 0.6124 & 0.3536 \\ 0 & 0 & 0.7071 \\ -0.5774 & 0.4082 & 0 \\ -0.5774 & -0.2041 & 0.3536 \\ 0 & -0.6124 & 0.3536 \end{bmatrix} \begin{bmatrix} 1.7321 & -0.5774 & -0.5774 \\ 0 & 1.6330 & -0.8165 \\ 0 & 0 & 1.4142 \end{bmatrix} = Q_1 R.$$

This is well-conditioned problem, it yields

$$Q_1^T \mathrm{b} = \begin{bmatrix} -376 \\ 1200 \\ 3417 \end{bmatrix} = \mathrm{c}_1.$$

Finally, normalize the third column $r_{33} = \|a_3\|_2 = 1.4142$ and

$$q_3 = a_3/r_{33} = [0.3536, 0.3536, 0.7071, 0, 0.3536, 0.3536]^T.$$

Replacing the third column with q_3, the reduced QR factorization yields

$$A = \begin{bmatrix} 0.5774 & 0.2041 & 0.3536 \\ 0 & 0.6124 & 0.3536 \\ 0 & 0 & 0.7071 \\ -0.5774 & 0.4082 & 0 \\ -0.5774 & -0.2041 & 0.3536 \\ 0 & -0.6124 & 0.3536 \end{bmatrix} \begin{bmatrix} 1.7321 & -0.5774 & -0.5774 \\ 0 & 1.6330 & -0.8165 \\ 0 & 0 & 1.4142 \end{bmatrix} = Q_1 R.$$

This is well-conditioned problem, it yields

$$Q_1^T b = \begin{bmatrix} -376 \\ 1200 \\ 3417 \end{bmatrix} = c_1.$$

Theorem

The Singular Value Decomposition (SVD) of an $m \times n$ matrix A has the form

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

where U is an $m \times \times m$ orthogonal matrix($U^H U = I$), V is an $n \times n$ orthogonal matrix($V^H V = I$), and $\Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_n)$ is an $m \times n$ diagonal matrix,

$$\text{where } \sigma_{ij} = \begin{cases} 0, & \text{for } i \neq j \\ \sigma \geq 0, & \text{for } i = j \end{cases}.$$

- singular values of A: diagonal entries σ_i
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$
- Columns u_i of U(v_i of V) is the left(right) singular vectors corresponding σ_i.

Theorem

The Singular Value Decomposition (SVD) of an $m \times n$ matrix A has the form

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

where U is an $m \times \times m$ orthogonal matrix($U^H U = I$), V is an $n \times n$ orthogonal matrix($V^H V = I$), and $\Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_n)$ is an $m \times n$ diagonal matrix,

$$\text{where } \sigma_{ij} = \begin{cases} 0, & \text{for } i \neq j \\ \sigma \geq 0, & \text{for } i = j \end{cases}.$$

- singular values of A: diagonal entries σ_i
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$
- Columns u_i of U(v_i of V) is the left(right) singular vectors corresponding σ_i.

Example

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 2^{-\frac{1}{2}} & -2^{-\frac{1}{2}} \\ 2^{-\frac{1}{2}} & 2^{-\frac{1}{2}} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 2^{-\frac{1}{2}} & -2^{-\frac{1}{2}} \\ 2^{-\frac{1}{2}} & 2^{-\frac{1}{2}} \end{bmatrix} := U\Sigma V^T$$

Example

Rank-deficient

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 5 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

The SVD of A is given by $U\Sigma V^T$, which is

$$\begin{bmatrix} 0.141 & 0.825 & -0.420 & -0.351 \\ 0.344 & 0.426 & 0.298 & 0.782 \\ 0.547 & 0.028 & 0.664 & -0.509 \\ 0.750 & -0.371 & -0.542 & 0.079 \end{bmatrix} \begin{bmatrix} 25.5 & 0 & 0 \\ 0 & 1.29 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.504 & 0.574 & 0.644 \\ -0.761 & -0.057 & 0.646 \\ 0.408 & -0.816 & 0.408 \end{bmatrix}$$

- It is equivalent to least square problem $\min \|Ax - b\|_2$

Example

Rank-deficient

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 5 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

The SVD of A is given by $U\Sigma V^T$, which is

$$\begin{bmatrix} 0.141 & 0.825 & -0.420 & -0.351 \\ 0.344 & 0.426 & 0.298 & 0.782 \\ 0.547 & 0.028 & 0.664 & -0.509 \\ 0.750 & -0.371 & -0.542 & 0.079 \end{bmatrix} \begin{bmatrix} 25.5 & 0 & 0 \\ 0 & 1.29 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.504 & 0.574 & 0.644 \\ -0.761 & -0.057 & 0.646 \\ 0.408 & -0.816 & 0.408 \end{bmatrix}$$

- It is equivalent to least square problem $\min \|Ax - b\|_2$

Example

Rank-deficient

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 5 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

The SVD of A is given by $U\Sigma V^T$, which is

$$\begin{bmatrix} 0.141 & 0.825 & -0.420 & -0.351 \\ 0.344 & 0.426 & 0.298 & 0.782 \\ 0.547 & 0.028 & 0.664 & -0.509 \\ 0.750 & -0.371 & -0.542 & 0.079 \end{bmatrix} \begin{bmatrix} 25.5 & 0 & 0 \\ 0 & 1.29 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.504 & 0.574 & 0.644 \\ -0.761 & -0.057 & 0.646 \\ 0.408 & -0.816 & 0.408 \end{bmatrix}$$

- It is equivalent to least square problem $\min \|Ax - b\|_2$

Near Rank Deficiency

Example

Consider a 3×2 matrix

$$A = \begin{bmatrix} 0.641 & 0.242 \\ 0.321 & 0.121 \\ 0.962 & 0.363 \end{bmatrix}$$

Computing its QR factorization with

$$R = \begin{bmatrix} 1.1997 & 0.4527 \\ 0 & 0.0002 \end{bmatrix}$$

- R is extremely close to singular.
- For practical purposes, $rank(A) = 1$ rather than 2.

Near Rank Deficiency

Example

Consider a 3×2 matrix

$$A = \begin{bmatrix} 0.641 & 0.242 \\ 0.321 & 0.121 \\ 0.962 & 0.363 \end{bmatrix}$$

Computing its QR factorization with

$$R = \begin{bmatrix} 1.1997 & 0.4527 \\ 0 & 0.0002 \end{bmatrix}$$

- R is extremely close to singular.
- For practical purposes, $rank(A) = 1$ rather than 2.

Apply to the square matrix A

Indeed $\|A\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_{max}$, $\quad cond(A) = \sigma_{max}/\sigma_{min}$ if A is square.

For instance,

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

It can be factor with $U\Sigma V^T =$

$$\begin{bmatrix} 0.392 & -0.920 & -0.021 \\ 0.240 & 0.081 & 0.967 \\ 0.888 & 0.384 & -0.253 \end{bmatrix} \begin{bmatrix} 5.723 & 0 & 0 \\ 0 & 1.068 & 0 \\ 0 & 0 & 0.327 \end{bmatrix} \begin{bmatrix} 0.645 & -0.224 & 0.731 \\ -0.567 & -0.501 & 0.653 \\ 0.513 & 0.836 & -0.196 \end{bmatrix}$$

Then $\|A\|_2 = 5.723$, $\qquad cond(A) = 5.723/0.327 = 17.5$

Apply to the square matrix A

Indeed $\|A\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_{max}$, $\quad cond(A) = \sigma_{max}/\sigma_{min}$ if A is square.

For instance,

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

It can be factor with $U\Sigma V^T =$

$$\begin{bmatrix} 0.392 & -0.920 & -0.021 \\ 0.240 & 0.081 & 0.967 \\ 0.888 & 0.384 & -0.253 \end{bmatrix} \begin{bmatrix} 5.723 & 0 & 0 \\ 0 & 1.068 & 0 \\ 0 & 0 & 0.327 \end{bmatrix} \begin{bmatrix} 0.645 & -0.224 & 0.731 \\ -0.567 & -0.501 & 0.653 \\ 0.513 & 0.836 & -0.196 \end{bmatrix}$$

Then $\|A\|_2 = 5.723$, $\qquad cond(A) = 5.723/0.327 = 17.5$

Apply to the square matrix A

Indeed $\|A\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_{max}$, $\quad cond(A) = \sigma_{max}/\sigma_{min}$ if A is square.

For instance,

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

It can be factor with $U\Sigma V^T =$

$$\begin{bmatrix} 0.392 & -0.920 & -0.021 \\ 0.240 & 0.081 & 0.967 \\ 0.888 & 0.384 & -0.253 \end{bmatrix} \begin{bmatrix} 5.723 & 0 & 0 \\ 0 & 1.068 & 0 \\ 0 & 0 & 0.327 \end{bmatrix} \begin{bmatrix} 0.645 & -0.224 & 0.731 \\ -0.567 & -0.501 & 0.653 \\ 0.513 & 0.836 & -0.196 \end{bmatrix}$$

Then $\|A\|_2 = 5.723$, $\qquad cond(A) = 5.723/0.327 = 17.5$

Application to Image Compression

- Digital image $\boldsymbol{I}$ with resolution n^2 pixels $\Longleftrightarrow$ matrix $\boldsymbol{A}$ sized $n \times n$
- Lower-Rank Approximation: only keep $\sigma_1 \cdots, \sigma_k$, which means that

$$A = U\Sigma V^T \approxeq \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T + \ldots$$

- In this sense, $k \times (2n+1)$ (8-bit) integers are require to recover the original $n \times n$ image I, and the image compression ratio:

$$\rho = \frac{n^2}{(2n+1)k}.$$

- Demo: a MATLAB script

```
load clown.mat;
[U,S,V]=svd(X);
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
```

Application to Image Compression

- Digital image $\boldsymbol{I}$ with resolution n^2 pixels $\Longleftrightarrow$ matrix $\boldsymbol{A}$ sized $n \times n$
- Lower-Rank Approximation: only keep $\sigma_1 \cdots, \sigma_k$, which means that

$$A = U\Sigma V^T \approxeq \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T + \dots$$

- In this sense, $k \times (2n+1)$ (8-bit) integers are require to recover the original $n \times n$ image I, and the image compression ratio:

$$\rho = \frac{n^2}{(2n+1)k}.$$

- Demo: a MATLAB script

```
load clown.mat;
[U,S,V]=svd(X);
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
```

Application to Image Compression

- Digital image $\boldsymbol{I}$ with resolution n^2 pixels $\Longleftrightarrow$ matrix $\boldsymbol{A}$ sized $n \times n$
- Lower-Rank Approximation: only keep $\sigma_1 \cdots, \sigma_k$, which means that

$$A = U\Sigma V^T \approxeq \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T + \ldots$$

- In this sense, $k \times (2n+1)$ (8-bit) integers are require to recover the original $n \times n$ image I, and the image compression ratio:

$$\rho = \frac{n^2}{(2n+1)k}.$$

- Demo: a MATLAB script

```
load clown.mat;
[U,S,V]=svd(X);
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
```

Application to Image Compression

- Digital image $\boldsymbol{I}$ with resolution n^2 pixels $\iff$ matrix $\boldsymbol{A}$ sized $n \times n$
- Lower-Rank Approximation: only keep $\sigma_1 \cdots, \sigma_k$, which means that

$$A = U\Sigma V^T \approxeq \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_k u_k v_k^T + \dots$$

- In this sense, $k \times (2n+1)$ (8-bit) integers are require to recover the original $n \times n$ image I, and the image compression ratio:

$$\rho = \frac{n^2}{(2n+1)k}.$$

- Demo: a MATLAB script

```
load clown.mat;
[U,S,V]=svd(X);
image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)');
```

Example (Image Compression with SVD($k = 100$))

source

k=100 compress ratio=2.5575

```
%% MATLAB code to demostrate Image Compression with SVD
clear all ;close all ; clc ;
I = imread('lena.jpg');
[m,n] = size(I);
k = 100;

Id = double(I);
[u,s,v] = svd (Id);

s = diag(s); plot(s,'.','Color','k'); % check the value s
s1 = s; s1(k:end) = 0; s1 = diag(s1);
Ig = uint8(u*s1*v');
compressratio = n^2/(k*(2*n+1));

figure ; % show the original image and that recovered with SVD components
subplot (1,2,1) ; imshow(mat2gray(Id)); title ('source')
subplot (1,2,2) ; imshow(Ig); title (['compress ratio',num2str(compressratio)])
```

A coarser approximation/compression($k = 25$)

source

k=20 compress ratio=12.7875

Conclusion & Homework 2

4. Eigenvalue Problem
    - The problem
    - Numerical Method
5. Linear Least Square
    - The Problem
    - Numerical Method
6. Matrix Factorization
    - Orthogonalization
    - Singular Value Decomposition
    - Applications

Homework:

1. "Exercises" of Chapter **3**:
   3.3, 3.7, 3.17, 3.20, 3.28
2. "Exercises" of Chapter **4**:
   4.2, 4.14, 4.17, 4.22, 4.32

Conclusion & Homework 2

4. Eigenvalue Problem
   - The problem
   - Numerical Method
5. Linear Least Square
   - The Problem
   - Numerical Method
6. Matrix Factorization
   - Orthogonalization
   - Singular Value Decomposition
   - Applications

Homework:

1. "Exercises" of Chapter **3**:
   3.3, 3.7, 3.17, 3.20, 3.28
2. "Exercises" of Chapter **4**:
   4.2, 4.14, 4.17, 4.22, 4.32

Thanks for your attentation!