AMS526: Numerical Analysis I
(Numerical Linear Algebra for Computational and Data Sciences)
Lecture 10: Review for Midterm #1; Gram-Schmidt Orthogonalization

Xiangmin Jiao

Stony Brook University
Outline

1. Review of Midterm #1

2. Gram-Schmidt Orthogonalization (NLA§7)

3. Modified Gram-Schmidt Orthogonalization (NLA§8)
Midterm #1

- Wednesday, Oct. 5th, 2016 in classroom
- It will cover material up to Cholesky factorization
- It is a closed-book exam
- You can bring a single-sided, one-page, letter-size cheat sheet, which you must prepare by yourself
Fundamental Concepts

- Norms, orthogonality, conditioning, stability
- Conditioning of problems
- Stability and backward stability of algorithms
- Efficiency of algorithms, operation counts
- Singular value decomposition, properties, and relationship with eigenvalue problems
- Orthogonal projection matrices, orthogonal matrices
Algorithms

- Matrix multiplication
- Triangular systems
- Gaussian elimination with/without pivoting
- Cholesky factorization and $LDL^T$ factorization
- Understand when they work, how they work, why they work, and how well they work
Outline

1. Review of Midterm #1

2. Gram-Schmidt Orthogonalization (NLA§7)

3. Modified Gram-Schmidt Orthogonalization (NLA§8)
Motivation

**Question:** Given a linear system $Ax \approx b$ where $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) has full rank, how to solve the linear system?

1. One approach is to solve normal equation $A^TAx = A^Tb$ directly using Cholesky factorization. It is unstable, but is very efficient if $m \gg n$ ($m^2n + \frac{1}{3}n^3$).

2. Another possible solution is to use SVD: $A = U\Sigma V^T$, so $x = V\Sigma^{-1}U^Tb$. It is stable, but is inefficient.

A more robust approach is to use QR factorization, which decomposes $A$ into product of two simple matrices $Q$ and $R$, where columns of $Q$ are orthonormal and $R$ is upper triangular.
Motivation

**Question:** Given a linear system $Ax \approx b$ where $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) has full rank, how to solve the linear system?

1. One approach is to solve normal equation $A^T Ax = A^T b$ directly using Cholesky factorization. It is unstable, but is very efficient if $m \gg n$ ($mn^2 + \frac{1}{3}n^3$).

2. Another possible solution is to use SVD:

   $$A = U \Sigma V^T, \quad \text{so} \quad x = V \Sigma^{-1} U^T b.$$

   It is stable, but is inefficient.

A more robust approach is to use QR factorization, which decomposes $A$ into product of two simple matrices $Q$ and $R$, where columns of $Q$ are orthonormal and $R$ is upper triangular.
Two Different Versions of QR

There are two versions of QR

- Full QR factorization: $A \in \mathbb{R}^{m \times n}$ ($m \geq n$)
  \[
  A = QR
  \]
  where $Q \in \mathbb{R}^{m \times m}$ is orthogonal and $R \in \mathbb{R}^{m \times n}$ is upper triangular

- Reduced (or thin) QR factorization: $A \in \mathbb{R}^{m \times n}$ ($m \geq n$)
  \[
  A = \hat{Q}\hat{R}
  \]
  where $Q \in \mathbb{R}^{m \times n}$ contains orthonormal vectors and $R \in \mathbb{R}^{n \times n}$ is upper triangular

- What space do $\{q_1, q_2, \ldots, q_j\}$, $j \leq n$ span?
Two Different Versions of QR

There are two versions of QR

- **Full QR factorization:** \( A \in \mathbb{R}^{m \times n} \quad (m \geq n) \)
  \[
  A = QR
  \]
  where \( Q \in \mathbb{R}^{m \times m} \) is orthogonal and \( R \in \mathbb{R}^{m \times n} \) is upper triangular

- **Reduced (or thin) QR factorization:** \( A \in \mathbb{R}^{m \times n} \quad (m \geq n) \)
  \[
  A = \hat{Q}\hat{R}
  \]
  where \( Q \in \mathbb{R}^{m \times n} \) contains orthonormal vectors and \( R \in \mathbb{R}^{n \times n} \) is upper triangular

- **What space do \( \{q_1, q_2, \ldots, q_j\}, \quad j \leq n \) span?**
  
  - Answer: For full rank \( A \), first \( j \) column vectors of \( A \), i.e.,
    \[
    \langle q_1, q_2, \ldots, q_j \rangle = \langle a_1, a_2, \ldots, a_j \rangle.
    \]
Gram-Schmidt Orthogonalization

- A method to construct QR factorization is to orthogonalize the column vectors of $A$:
- Basic idea:
  - Take first column $a_1$ and normalize it to obtain vector $q_1$;
  - Take second column $a_2$, subtract its orthogonal projection to $q_1$, and normalize to obtain $q_2$;
  - ... 
  - Take $j$th column of $a_j$, subtract its orthogonal projection to $q_1, \ldots, q_{j-1}$, and normalize to obtain $q_j$;

$$v_j = a_j - \sum_{i=1}^{j-1} q_i^T a_j q_i, \quad q_j = v_j/\|v_j\|.$$ 

- This idea is called **Gram-Schmidt orthogonalization**.
Gram-Schmidt Projections

- Orthogonal vectors produced by Gram-Schmidt can be written in terms of projectors
  \[ q_j = \frac{P_j a_j}{\|P_j a_j\|} \]

  where

  \[ P_j = I - \hat{Q}_{j-1} \hat{Q}_{j-1}^T \text{ with } \hat{Q}_{j-1} = \begin{bmatrix} q_1 & q_2 & \cdots & q_{j-1} \end{bmatrix} \]

- \( P_j \) projects orthogonally onto space orthogonal to \( \langle q_1, q_2, \ldots, q_{j-1} \rangle \)
  and rank of \( P_j \) is \( m - (j - 1) \)
Existence of QR

Theorem

Every $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) has full QR factorization, hence also a reduced QR factorization.
Existence of QR

Theorem

Every $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) has full QR factorization, hence also a reduced QR factorization.

Key idea of proof: If $A$ has full rank, Gram-Schmidt algorithm provides a proof itself for having reduced QR. If $A$ does not have full rank, at some step $v_j = 0$. We can set $q_j$ to be a vector orthogonal to $q_i$, $i < j$. To construct full QR from reduced QR, just continue Gram-Schmidt an additional $m - n$ steps.
Uniqueness of QR

Theorem

Every \( A \in \mathbb{R}^{m \times n} \) of full rank has a unique reduced QR factorization \( A = \hat{Q}\hat{R} \) with \( r_{jj} > 0 \).

Key idea of proof: Proof is provided by Gram-Schmidt iteration itself. If the signs of \( r_{jj} \) are determined, then \( r_{ij} \) and \( q_j \) are determined.
Uniqueness of QR

**Theorem**

Every \( A \in \mathbb{R}^{m \times n} \) \((m \geq n)\) of full rank has a unique reduced QR factorization \( A = \hat{Q} \hat{R} \) with \( r_{jj} > 0 \).

**Key idea of proof**: Proof is provided by Gram-Schmidt iteration itself. If the signs of \( r_{jj} \) are determined, then \( r_{ij} \) and \( q_j \) are determined.

**Question**: Why do we require \( r_{jj} > 0 \)?

Xiangmin Jiao
Numerical Analysis I
Theorem

Every $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) of full rank has a unique reduced QR factorization $A = \hat{Q}\hat{R}$ with $r_{jj} > 0$.

Key idea of proof: Proof is provided by Gram-Schmidt iteration itself. If the signs of $r_{jj}$ are determined, then $r_{ij}$ and $q_j$ are determined.

Question: Why do we require $r_{jj} > 0$?

Question: Is full QR factorization unique?
Uniqueness of QR

Every \( A \in \mathbb{R}^{m \times n} (m \geq n) \) of full rank has a unique reduced QR factorization \( A = \hat{Q} \hat{R} \) with \( r_{jj} > 0 \).

**Key idea of proof**: Proof is provided by Gram-Schmidt iteration itself. If the signs of \( r_{jj} \) are determined, then \( r_{ij} \) and \( q_j \) are determined.

**Question**: Why do we require \( r_{jj} > 0 \)?

**Question**: Is full QR factorization unique?

**Question**: What if \( A \) does not have full rank?
Outline

1. Review of Midterm #1
2. Gram-Schmidt Orthogonalization (NLA§7)
3. Modified Gram-Schmidt Orthogonalization (NLA§8)
Algorithm of Gram-Schmidt Orthogonalization

Classical Gram-Schmidt method

\[
\text{for } j = 1 : n \\
\quad v_j = a_j \\
\quad \text{for } i = 1 : j - 1 \\
\quad \quad r_{ij} = q_i^T a_j \\
\quad \quad v_j = v_j - r_{ij} q_i \\
\quad \quad r_{jj} = \|v_j\|_2 \\
\quad q_j = v_j / r_{jj}
\]

- GS require $\sim 2mn^2$ flops to compute QR factorization of $m \times n$ matrix
- Classical Gram-Schmidt (CGS) is **unstable**, which means that its solution is sensitive to perturbation
Alternative View of Gram-Schmidt Projection

- Orthogonal vectors produced by Gram-Schmidt can be written in terms of projectors

\[ q_j = \frac{P_j a_j}{\|P_j a_j\|}, \text{ where } P_j = I - \hat{Q}_{j-1} \hat{Q}_{j-1}^T, \hat{Q}_{j-1} = [q_1 | q_2 | \cdots | q_{j-1}] \]

- We may view \( P_j \) as product of a sequence of projections

\[ P_j = P_{\perp q_{j-1}} P_{\perp q_{j-2}} \cdots P_{\perp q_1} \]

where \( P_{\perp q} = I - q q^T \)

- Instead of computing \( v_j = P_j a_i \), one could compute \( v_j = P_{\perp q_{j-1}} P_{\perp q_{j-2}} \cdots P_{\perp q_1} a_j \) instead, resulting in modified Gram-Schmidt algorithm
Modified Gram-Schmidt Algorithm

**Classical Gram-Schmidt method**

```plaintext
for j = 1 : n
    v_j = a_j
    for i = 1 : j - 1
        r_ij = q_i^T a_j
        v_j = v_j - r_ij q_i
    rjj = ||v_j||_2
    q_j = v_j / rjj
```

**Modified Gram-Schmidt method**

```plaintext
for j = 1 : n
    v_j = a_j
    for i = 1 : n
        r_ii = ||v_i||_2
        q_i = v_i / r_ii
    for j = i + 1 : n
        r_ij = q_i^T v_j
        v_j = v_j - r_ij q_i
```

Key difference between CGS and MGS is how \( r_{ij} \) is computed.

CGS above is column-oriented (in the sense that \( R \) is computed column by column) and MGS above is row-oriented, but this is NOT the main difference between CGS and MGS. There are also column-oriented MGS and row-oriented CGS.

MGS is numerically “more stable” than CGS, but neither is stable.
Modified Gram-Schmidt Algorithm

Classical Gram-Schmidt method

\[
\text{for } j = 1 : n \\
\quad v_j = a_j \\
\text{for } i = 1 : j - 1 \\
\quad r_{ij} = q_i^T a_j \\
\quad v_j = v_j - r_{ij} q_i \\
\quad r_{jj} = \|v_j\|_2 \\
\quad q_j = v_j / r_{jj}
\]

Modified Gram-Schmidt method

\[
\text{for } j = 1 : n \\
\quad v_j = a_j \\
\text{for } i = 1 : n \\
\quad r_{ii} = \|v_i\|_2 \\
\quad q_i = v_i / r_{ii} \\
\text{for } j = i + 1 : n \\
\quad r_{ij} = q_i^T v_j \\
\quad v_j = v_j - r_{ij} q_i
\]

- Key difference between CGS and MGS is how \( r_{ij} \) is computed.
- CGS above is column-oriented (in the sense that \( R \) is computed column by column) and MGS above is row-oriented, but this is NOT the main difference between CGS and MGS. There are also column-oriented MGS and row-oriented CGS.
- MGS is numerically “more stable” than CGS, but neither is stable.
- In MGS, \( v_j \) can overwrite \( a_j \), and \( q_j \) can overwrite \( v_j \).
Example: CGS vs. MGS

- Consider matrix

\[
A = \begin{bmatrix}
1 & 1 & 1 \\
\varepsilon & 0 & 0 \\
0 & \varepsilon & 0 \\
0 & 0 & \varepsilon \\
\end{bmatrix}
\]

where \( \varepsilon \) is small such that \( 1 + \varepsilon^2 = 1 \) with round-off error

- For both CGS and MGS

\[
v_1 \leftarrow (1, \varepsilon, 0, 0)^T, \quad r_{11} = \sqrt{1 + \varepsilon^2} \approx 1, \quad q_1 = v_1/r_{11} = (1, \varepsilon, 0, 0)^T,
\]

\[
v_2 \leftarrow (1, 0, \varepsilon, 0)^T, \quad r_{12} = q_1^T a_2 (\text{or } = q_1^T v_2) = 1
\]

\[
v_2 \leftarrow v_2 - r_{12} q_1 = (0, -\varepsilon, \varepsilon, 0)^T,
\]

\[
r_{22} = \sqrt{2}\varepsilon, \quad q_2 = (0, -1, 1, 0)/\sqrt{2},
\]

\[
v_3 \leftarrow (1, 0, 0, \varepsilon)^T, \quad r_{13} = q_1^T a_3 (\text{or } = q_1^T v_3) = 1
\]

\[
v_3 \leftarrow v_3 - r_{13} q_1 = (0, -\varepsilon, 0, \varepsilon)^T
\]
Example: CGS vs. MGS Cont’d

● For CGS:

\[ r_{23} = q_2^T a_3 = 0, \quad v_3 \leftarrow v_3 - r_{23} q_2 = (0, -\varepsilon, 0, \varepsilon)^T \]

\[ r_{33} = \sqrt{2}\varepsilon, \quad q_3 = v_3 / r_{33} = (0, -1, 0, 1)^T / \sqrt{2} \]

▶ Note that \( q_2^T q_3 = (0, -1, 1, 0)(0, -1, 0, 1)^T / 2 = 1/2 \)

● For MGS:

\[ r_{23} = q_2^T v_3 = \varepsilon / \sqrt{2}, \quad v_3 \leftarrow v_3 - r_{23} q_2 = (0, -\varepsilon / 2, -\varepsilon / 2, \varepsilon)^T \]

\[ r_{33} = \sqrt{6}\varepsilon / 2, \quad q_3 = v_3 / r_{33} = (0, -1, -1, 2)^T / \sqrt{6} \]

▶ Note that \( q_2^T q_3 = (0, -1, 1, 0)(0, -1, -1, 2)^T / \sqrt{12} = 0 \)