

AMS526: Numerical Analysis I (Numerical Linear Algebra)

Lecture 7: Gram-Schmidt Orthogonalization

Xiangmin Jiao

SUNY Stony Brook

September 23, 2008

Gram-Schmidt Projections

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

$$\mathbf{q}_j = \frac{\mathbf{P}_j \mathbf{a}_j}{\|\mathbf{P}_j \mathbf{a}_j\|}$$

where

$$\mathbf{P}_j = \mathbf{I} - \hat{\mathbf{Q}}_{j-1} \hat{\mathbf{Q}}_{j-1}^* \text{ with } \hat{\mathbf{Q}}_{j-1} = [\mathbf{q}_1 \quad \mathbf{q}_2 \quad \cdots \quad \mathbf{q}_{j-1}]$$

- \mathbf{P}_j projects orthogonally onto space orthogonal to $\langle \mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_{j-1} \rangle$ and rank of \mathbf{P}_j is $m - (j - 1)$

Algorithm of Gram-Schmidt Orthogonalization

Classical Gram-Schmidt method

for $j = 1$ to n

$$\mathbf{v}_j = \mathbf{a}_j$$

for $i = 1$ to $j - 1$

$$r_{ij} = \mathbf{q}_i^* \mathbf{a}_j$$

$$\mathbf{v}_j = \mathbf{v}_j - r_{ij} \mathbf{q}_i$$

$$r_{jj} = \|\mathbf{v}_j\|_2$$

$$\mathbf{q}_j = \mathbf{v}_j / r_{jj}$$

Algorithm of Gram-Schmidt Orthogonalization

Classical Gram-Schmidt method

for $j = 1$ to n

$$\mathbf{v}_j = \mathbf{a}_j$$

for $i = 1$ to $j - 1$

$$r_{ij} = \mathbf{q}_i^* \mathbf{a}_j$$

$$\mathbf{v}_j = \mathbf{v}_j - r_{ij} \mathbf{q}_i$$

$$r_{jj} = \|\mathbf{v}_j\|_2$$

$$\mathbf{q}_j = \mathbf{v}_j / r_{jj}$$

- Classical Gram-Schmidt (CGS) is **unstable**, which means that its solution is sensitive to perturbation

Existence of QR

Theorem

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

Existence of QR

Theorem

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

Key idea of proof: If \mathbf{A} has full rank, Gram-Schmidt algorithm provides a proof itself for having reduced QR.

If \mathbf{A} does not have full rank, at some step $\mathbf{v}_j = \mathbf{0}$. We can set \mathbf{q}_j to be a vector orthogonal to \mathbf{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 $\mathbf{A} \in \mathbb{C}^{m \times n}$ ($m \geq n$) of full rank has a unique reduced QR factorization $\mathbf{A} = \widehat{\mathbf{Q}}\widehat{\mathbf{R}}$ 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 \mathbf{q}_j are determined.

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n}$ ($m \geq n$) of full rank has a unique reduced QR factorization $\mathbf{A} = \widehat{\mathbf{Q}}\widehat{\mathbf{R}}$ 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 \mathbf{q}_j are determined.

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

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n}$ ($m \geq n$) of full rank has a unique reduced QR factorization $\mathbf{A} = \widehat{\mathbf{Q}}\widehat{\mathbf{R}}$ 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 \mathbf{q}_j are determined.

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

Question: Is full QR factorization unique?

Uniqueness of QR

Theorem

Every $\mathbf{A} \in \mathbb{C}^{m \times n}$ ($m \geq n$) of full rank has a unique reduced QR factorization $\mathbf{A} = \widehat{\mathbf{Q}}\widehat{\mathbf{R}}$ 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 \mathbf{q}_j are determined.

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

Question: Is full QR factorization unique?

Question: What if \mathbf{A} does not have full rank?

Alternative view to Gram-Schmidt Projection

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

$$\mathbf{q}_j = \frac{\mathbf{P}_j \mathbf{a}_j}{\|\mathbf{P}_j \mathbf{a}_j\|}, \text{ where } \mathbf{P}_j = \mathbf{I} - \hat{\mathbf{Q}}_{j-1} \hat{\mathbf{Q}}_{j-1}^*, \hat{\mathbf{Q}}_{j-1} = [\mathbf{q}_1 | \mathbf{q}_2 | \cdots | \mathbf{q}_{j-1}]$$

- We may view \mathbf{P}_j as product of a sequence of projections

$$\mathbf{P}_j = \mathbf{P}_{\perp \mathbf{q}_{j-1}} \mathbf{P}_{\perp \mathbf{q}_{j-2}} \cdots \mathbf{P}_{\perp \mathbf{q}_1}$$

where $\mathbf{P}_{\perp \mathbf{q}} = \mathbf{I} - \mathbf{q} \mathbf{q}^*$

- Instead of computing $\mathbf{v}_j = \mathbf{P}_j \mathbf{a}_i$, one could compute $\mathbf{v}_j = \mathbf{P}_{\perp \mathbf{q}_{j-1}} \mathbf{P}_{\perp \mathbf{q}_{j-2}} \cdots \mathbf{P}_{\perp \mathbf{q}_1} \mathbf{a}_j$ instead, resulting in modified Gram-Schmidt algorithm

Modified Gram-Schmidt Algorithm

Classical Gram-Schmidt method

for $j = 1$ **to** n

$$\mathbf{v}_j = \mathbf{a}_j$$

for $i = 1$ **to** $j - 1$

$$r_{ij} = \mathbf{q}_i^* \mathbf{a}_j$$

$$\mathbf{v}_j = \mathbf{v}_j - r_{ij} \mathbf{q}_i$$

$$r_{jj} = \|\mathbf{v}_j\|_2$$

$$\mathbf{q}_j = \mathbf{v}_j / r_{jj}$$

Modified Gram-Schmidt method

for $j = 1$ **to** n

$$\mathbf{v}_j = \mathbf{a}_j$$

for $i = 1$ **to** n

$$r_{ii} = \|\mathbf{v}_j\|_2$$

$$\mathbf{q}_i = \mathbf{v}_j / r_{ii}$$

for $j = i + 1$ **to** n

$$r_{ij} = \mathbf{q}_i^* \mathbf{v}_j$$

$$\mathbf{v}_j = \mathbf{v}_j - r_{ij} \mathbf{q}_i$$

Modified Gram-Schmidt Algorithm

Classical Gram-Schmidt method

```
for  $j = 1$  to  $n$   
   $\mathbf{v}_j = \mathbf{a}_j$   
  for  $i = 1$  to  $j - 1$   
     $r_{ij} = \mathbf{q}_i^* \mathbf{a}_j$   
     $\mathbf{v}_j = \mathbf{v}_j - r_{ij} \mathbf{q}_i$   
 $r_{jj} = \|\mathbf{v}_j\|_2$   
 $\mathbf{q}_j = \mathbf{v}_j / r_{jj}$ 
```

Modified Gram-Schmidt method

```
for  $j = 1$  to  $n$   
   $\mathbf{v}_j = \mathbf{a}_j$   
  for  $i = 1$  to  $n$   
     $r_{ii} = \|\mathbf{v}_j\|_2$   
     $\mathbf{q}_i = \mathbf{v}_j / r_{ii}$   
    for  $k = i + 1$  to  $n$   
       $r_{ik} = \mathbf{q}_i^* \mathbf{v}_k$   
       $\mathbf{v}_k = \mathbf{v}_k - r_{ik} \mathbf{q}_i$ 
```

- Key difference between CGS and MGS is how r_{ij} is computed
- CGS above is column-oriented (in the sense that \mathbf{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 stable (less sensitive to round-off errors)
- \mathbf{v}_j can overwrite \mathbf{a}_j , and \mathbf{q}_i can overwrite \mathbf{v}_j

Example: CGS vs. MGS

- Consider matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ \varepsilon & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \varepsilon \end{bmatrix}$$

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

- For both CGS and MGS

$$\mathbf{v}_1 \leftarrow (1, \varepsilon, 0, 0)^T, \quad r_{11} = \sqrt{1 + \varepsilon^2} \approx 1, \quad \mathbf{q}_1 = \mathbf{v}_1 / r_{11} = (1, \varepsilon, 0, 0)^T,$$

$$\mathbf{v}_2 \leftarrow (1, 0, \varepsilon, 0)^T, \quad r_{12} = \mathbf{q}_1^T \mathbf{a}_2 (\text{or } = \mathbf{q}_1^T \mathbf{v}_2) = 1$$

$$\mathbf{v}_2 \leftarrow \mathbf{v}_2 - r_{12} \mathbf{q}_1 = (0, -\varepsilon, \varepsilon, 0)^T,$$

$$r_{22} = \sqrt{2}\varepsilon, \quad \mathbf{q}_2 = (0, -1, 1, 0) / \sqrt{2},$$

$$\mathbf{v}_3 \leftarrow (1, 0, 0, \varepsilon)^T, \quad r_{13} = \mathbf{q}_1^T \mathbf{a}_3 (\text{or } = \mathbf{q}_1^T \mathbf{v}_3) = 1$$

$$\mathbf{v}_3 \leftarrow \mathbf{v}_3 - r_{13} \mathbf{q}_1 = (0, -\varepsilon, 0, \varepsilon)^T$$

Example: CGS vs. MGS Cont'd

- For CGS:

$$r_{23} = \mathbf{q}_2^T \mathbf{a}_3 = 0, \quad \mathbf{v}_3 \leftarrow \mathbf{v}_3 - r_{23} \mathbf{q}_2 = (0, -\varepsilon, 0, \varepsilon)^T$$

$$r_{33} = \sqrt{2}\varepsilon, \quad \mathbf{q}_3 = \mathbf{v}_3 / r_{33} = (0, -1, 0, 1)^T / \sqrt{2}$$

- ▶ Note that $\mathbf{q}_2^T \mathbf{q}_3 = (0, -1, 1, 0)(0, -1, 0, 1)^T / 2 = 1/2$

- For MGS:

$$r_{23} = \mathbf{q}_2^T \mathbf{v}_3 = \varepsilon / \sqrt{2}, \quad \mathbf{v}_3 \leftarrow \mathbf{v}_3 - r_{23} \mathbf{q}_2 = (0, -\varepsilon/2, -\varepsilon/2, \varepsilon)^T$$

$$r_{33} = \sqrt{6}\varepsilon/2, \quad \mathbf{q}_3 = \mathbf{v}_3 / r_{33} = (0, -1, -1, 2)^T / \sqrt{6}$$

- ▶ Note that $\mathbf{q}_2^T \mathbf{q}_3 = (0, -1, 1, 0)(0, -1, -1, 2)^T / \sqrt{12} = 0$

Operation Count

- It is important to assess the *efficiency* of algorithms. But how?
 - ▶ We could implement different algorithms and do head-to-head comparison, but implementation details might affect true performance
 - ▶ We could estimate cost of all operations, but it is very tedious
 - ▶ Relatively simple and effective approach is to estimate amount of floating-point operations, or “flops”, and focus on asymptotic analysis as sizes of matrices approach infinity
- Count each operation $+$, $-$, $*$, $/$, and $\sqrt{\quad}$ as one flop, and make no distinction of real and complex numbers

Theorem

CGS and MGS require $\sim 2mn^2$ flops to compute a QR factorization of an $m \times n$ matrices.