

AMS526: Numerical Analysis I (Numerical Linear Algebra)

Lecture 8: Householder Reflectors and Givens Rotations

Xiangmin Jiao

SUNY Stony Brook

September 25, 2008

Review: Gram-Schmidt Algorithms

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

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

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

- What is key difference between CGS and MGS?
- Which of the two is more numerically stable?

Gram-Schmidt as Triangular Orthogonalization

- Every step of Gram-Schmidt can be viewed as multiplication with triangular matrix. For example, at first step:

$$\underbrace{[\mathbf{v}_1 | \mathbf{v}_1 | \cdots | \mathbf{v}_n]}_{R_1} = [\mathbf{q}_1 | \mathbf{v}_1^{(2)} | \cdots | \mathbf{v}_n^{(2)}],$$

- Gram-Schmidt therefore multiplies triangular matrices to orthogonalize column vectors, and in turns can be viewed as *triangular orthogonalization*

$$\mathbf{A} \underbrace{R_1 R_2 \cdots R_n}_{\widehat{R}^{-1}} = \widehat{Q}$$

where R_i is triangular matrix

- A “dual” approach would be *orthogonal triangularization*, i.e., multiply \mathbf{A} by unitary matrices to make it triangular matrix.

Householder Triangularization

- Method introduced by Alston Scott Householder in 1958
- It multiplies unitary matrices to make column triangular, e.g.

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{Q_1} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{Q_2} \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & 0 & \times \\ & 0 & \times \\ & 0 & \times \end{bmatrix} \xrightarrow{Q_3} \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \\ & & 0 \\ & & 0 \end{bmatrix}$$

$\mathbf{A} \qquad \mathbf{Q}_1\mathbf{A} \qquad \mathbf{Q}_2\mathbf{Q}_1\mathbf{A} \qquad \mathbf{Q}_3\mathbf{Q}_2\mathbf{Q}_1\mathbf{A}$

- After n steps, we get a product of unitary matrices

$$\underbrace{Q_n \cdots Q_2 Q_1}_{Q^*} \mathbf{A} = \mathbf{R}$$

and in turn we get full QR factorization $\mathbf{A} = \mathbf{QR}$

- Q_k introduces zeros below diagonal of k th column while preserving zeros below diagonal in preceding columns
- The key question is how to find Q_k

Householder Reflectors

- First, consider Q_1 : $Q_1 \mathbf{a}_1 = \|\mathbf{a}_1\| \mathbf{e}_1$, where $\mathbf{e}_1 = (1, 0, \dots, 0)^T$. Why the length is $\|\mathbf{a}_1\|$?
- Q_1 reflects \mathbf{a}_1 across hyperplane H orthogonal to $\mathbf{v} = \|\mathbf{a}_1\| \mathbf{e}_1 - \mathbf{a}_1$, and therefore

$$Q_1 = I - 2 \frac{\mathbf{v} \mathbf{v}^*}{\mathbf{v}^* \mathbf{v}}$$

- More generally,

$$Q_k = \begin{bmatrix} I & \mathbf{0} \\ \mathbf{0} & F \end{bmatrix}$$

where I is $(k-1) \times (k-1)$ and F is $(m-k+1) \times (m-k+1)$ such that $F \mathbf{x} = \|\mathbf{x}\|_2 \mathbf{e}_1$, where \mathbf{x} is $(a_{k,k}, a_{k,k+1}, \dots, a_{k,m})^T$

- What is F ? It has similar form as Q_1 with $\mathbf{v} = \|\mathbf{x}\| \mathbf{e}_1 - \mathbf{x}$.

Choice of Reflectors

- We could choose F such that $F\mathbf{x} = -\|\mathbf{x}\|\mathbf{e}_1$ instead of $F\mathbf{x} = \|\mathbf{x}\|\mathbf{e}_1$, or more generally, $F\mathbf{x} = z\|\mathbf{x}\|\mathbf{e}_1$ with $|z| = 1$ for $z \in \mathbb{C}$
- This leads to an infinite number of possible QR factorizations of A
- If we require $z \in \mathbb{R}$, we still have two choices
- Numerically, it is undesirable for $\mathbf{v}^*\mathbf{v}$ to be close to zero for $\mathbf{v} = z\|\mathbf{x}\|\mathbf{e}_1 - \mathbf{x}$, and $\|\mathbf{v}\|$ is larger if $z = -\text{sign}(x_1)$
- For completeness, if $x_1 = 0$, set z to 1 (instead of 0)

Householder Algorithm

Householder QR Factorization

for $k = 1$ to n

$$\mathbf{x} = \mathbf{A}_{k:m,k}$$

$$\mathbf{v}_k = \mathbf{sign}(x_1) \|\mathbf{x}\| \mathbf{e}_1 + \mathbf{x}$$

$$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|$$

$$\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$$

- Note that $\mathbf{sign}(x) = 1$ if $x \geq 0$ and $= -1$ if $x < 0$
- Leave \mathbf{R} in place of \mathbf{A}
- Matrix \mathbf{Q} is not formed explicitly but reflection vector \mathbf{v}_k is stored

Householder Algorithm

Householder QR Factorization

for $k = 1$ to n

$$\mathbf{x} = \mathbf{A}_{k:m,k}$$

$$\mathbf{v}_k = \mathbf{sign}(x_1) \|\mathbf{x}\| \mathbf{e}_1 + \mathbf{x}$$

$$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|$$

$$\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$$

- Note that $\mathbf{sign}(x) = 1$ if $x \geq 0$ and $= -1$ if $x < 0$
- Leave \mathbf{R} in place of \mathbf{A}
- Matrix \mathbf{Q} is not formed explicitly but reflection vector \mathbf{v}_k is stored
- Question: Can \mathbf{A} be reused to store both \mathbf{R} and \mathbf{v}_k completely?

Householder Algorithm

Householder QR Factorization

for $k = 1$ to n

$$\mathbf{x} = \mathbf{A}_{k:m,k}$$

$$\mathbf{v}_k = \text{sign}(x_1) \|\mathbf{x}\| \mathbf{e}_1 + \mathbf{x}$$

$$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|$$

$$\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$$

- Note that $\text{sign}(x) = 1$ if $x \geq 0$ and $= -1$ if $x < 0$
- Leave \mathbf{R} in place of \mathbf{A}
- Matrix \mathbf{Q} is not formed explicitly but reflection vector \mathbf{v}_k is stored
- Question: Can \mathbf{A} be reused to store both \mathbf{R} and \mathbf{v}_k completely?
- Answer: We can use lower diagonal portion of \mathbf{A} to store $\mathbf{v}_{k,k+1:m}$ but need additional storage for $v_{k,k}$

Householder Algorithm

Householder QR Factorization

for $k = 1$ to n

$$\mathbf{x} = \mathbf{A}_{k:m,k}$$

$$\mathbf{v}_k = \text{sign}(x_1) \|\mathbf{x}\| \mathbf{e}_1 + \mathbf{x}$$

$$\mathbf{v}_k = \mathbf{v}_k / \|\mathbf{v}_k\|$$

$$\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$$

- Note that $\text{sign}(x) = 1$ if $x \geq 0$ and $= -1$ if $x < 0$
- Leave \mathbf{R} in place of \mathbf{A}
- Matrix \mathbf{Q} is not formed explicitly but reflection vector \mathbf{v}_k is stored
- Question: Can \mathbf{A} be reused to store both \mathbf{R} and \mathbf{v}_k completely?
- Answer: We can use lower diagonal portion of \mathbf{A} to store $\mathbf{v}_{k,k+1:m}$ but need additional storage for $v_{k,k}$
- Question: What happens if \mathbf{v}_k is $\mathbf{0}$ in line 3?

Applying or Forming Q

- Compute $Q^*b = Q_n \cdots Q_1 b$

Implicit calculation of Q^*b

for $k = 1$ to n

$$b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$$

Applying or Forming Q

- Compute $Q^*b = Q_n \cdots Q_1 b$

Implicit calculation of Q^*b

for $k = 1$ to n

$$b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$$

- Compute $Qx = Q_1 Q_2 \cdots Q_n x$

Implicit calculation of Qx

for $k = n$ downto 1

$$b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$$

- Question: How to form Q and \hat{Q} , respectively?

Applying or Forming Q

- Compute $Q^*b = Q_n \cdots Q_1 b$

Implicit calculation of Q^*b

for $k = 1$ to n

$$b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$$

- Compute $Qx = Q_1 Q_2 \cdots Q_n x$

Implicit calculation of Qx

for $k = n$ downto 1

$$b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$$

- Question: How to form Q and \hat{Q} , respectively?
- Answer: Apply $x = I_{m \times m}$ or first n columns of I , respectively

Operation Count

- Most work done at step $\mathbf{A}_{k:m,k:n} = \mathbf{A}_{k:m,k:n} - 2\mathbf{v}_k(\mathbf{v}_k^* \mathbf{A}_{k:m,k:n})$
- Flops per iteration:
 - ▶ $\sim 2(m-k)(n-k)$ for dot products $\mathbf{v}_k^* \mathbf{A}_{k:m,k:n}$
 - ▶ $\sim (m-k)(n-k)$ for outer product $2\mathbf{v}_k(\cdot \cdot \cdot)$
 - ▶ $\sim (m-k)(n-k)$ for subtraction
 - ▶ $\sim 4(m-k)(n-k)$ total
- Including outer loop, total flops is

$$\begin{aligned}\sum_{k=1}^n 4(m-k)(n-k) &= 4 \sum_{k=1}^n (mn - km - kn + k^2) \\ &\sim 4mn^2 - 4(m+n)n^2/2 + 4n^3/3 \\ &= 2mn^2 - \frac{2}{3}n^3\end{aligned}$$

- If $m \approx n$, it is more efficient than Gram-Schmidt method, but if $m \gg n$, similar to Gram-Schmidt

Givens Rotations

- Instead of using reflection, we can rotate \mathbf{x} to obtain $\|\mathbf{x}\|\mathbf{e}_1$
- A Givens rotation $\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$, which rotates $\mathbf{x} \in \mathbb{R}^2$ counterclockwise by θ
- Choose θ to be angle between $(x_i, x_j)^T$ and $(1, 0)^T$, and we have

$$\begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{bmatrix}$$

where

$$\cos \theta = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \quad \sin \theta = \frac{-x_j}{\sqrt{x_i^2 + x_j^2}}$$

Givens QR

- Introduce zeros in column bottom-up, one zero at a time

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{(4,5)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \end{bmatrix} \xrightarrow{(3,4)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \mathbf{x} & \mathbf{x} & \mathbf{x} \\ 0 & \mathbf{x} & \mathbf{x} \\ & \times & \times \end{bmatrix} \dots$$

- To zero a_{ij} , left-multiply matrix F with $F_{i:i+1,i:i+1}$ being rotation matrix and $F_{kk} = 1$ for $k \neq i, i+1$
- Flop count of Givens QR is $3mn^2 - n^3$, which is about 50% more expensive than Householder QR