Notes on Gram-Schmidt QR Factorization

Robert A. van de Geijn
Department of Computer Science
The University of Texas
Austin, TX 78712
rvdg@cs.utexas.edu

September 15, 2014

A classic problem in linear algebra is the computation of an orthonormal basis for the space spanned by a given set of linearly independent vectors: Given a linearly independent set of vectors $\{a_0, \ldots, a_{n-1}\} \subset \mathbb{C}^m$ we would like to find a set of mutually orthonormal vectors $\{q_0, \ldots, q_{n-1}\} \subset \mathbb{C}^m$ so that

$$\text{Span}(\{a_0, \ldots, a_{n-1}\}) = \text{Span}(\{q_0, \ldots, q_{n-1}\}).$$

This problem is equivalent to the problem of, given a matrix $A = \left(\begin{array}{c|c|c} a_0 & \cdots & a_{n-1} \end{array} \right)$, computing a matrix $Q = \left(\begin{array}{c|c|c} q_0 & \cdots & q_{n-1} \end{array} \right)$ with $Q^H Q = I$ so that $\mathcal{C}(A) = \mathcal{C}(Q)$, where (A) denotes the column space of A.

A review at the undergraduate level of this topic (with animated illustrations) can be found in Week 11 of

Linear Algebra: Foundations to Frontiers - Notes to LAFF With.

1 Classical Gram-Schmidt process

Given a set of linearly independent vectors $\{a_0, \ldots, a_{n-1}\} \subset \mathbb{C}^m$, the Gram-Schmidt process computes an orthonormal basis $\{q_0, \ldots, q_{n-1}\}$ that span the same subspace, i.e.

$$\text{Span}(\{a_0, \ldots, a_{n-1}\}) = \text{Span}(\{q_0, \ldots, q_{n-1}\}).$$

The process proceeds as described in Figure 1 and in the algorithms in Figure 2.

Exercise 1. What happens in the Gram-Schmidt algorithm if the columns of A are NOT linearly independent? How might one fix this? How can the Gram-Schmidt algorithm be used to identify which columns of A are linearly independent?

Exercise 2. Convince yourself that the relation between the vectors $\{a_j\}$ and $\{q_j\}$ in the algorithms in Figure 2 is given by

$$\left(\begin{array}{c|c|c|c} a_0 & a_1 & \cdots & a_{n-1} \end{array} \right) = \left(\begin{array}{c|c|c|c} q_0 & q_1 & \cdots & q_{n-1} \end{array} \right) \left(\begin{array}{c|c|c|c} \rho_{0,0} & \rho_{0,1} & \cdots & \rho_{0,n-1} \\ \hline 0 & \rho_{1,1} & \cdots & \rho_{1,n-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline 0 & 0 & \cdots & \rho_{n-1,n-1} \end{array} \right),$$

Steps	Comment
$\rho_{0,0} := \|a_0\|_2$ $q_0 =: a_0/\rho_{0,0}$	Compute the length of vector a_0, $\rho_{0,0} := \|a_0\|_2$. Set $q_0 := a_0/\rho_{0,0}$, creating a unit vector in the direction of a_0. Clearly, $\text{Span}(\{a_0\}) = \text{Span}(\{q_0\})$. (Why?)
$\rho_{0,1} = q_0^H a_1$ $a_1^\perp = a_1 - \rho_{0,1} q_0$ $\rho_{1,1} = \|a_1^\perp\|_2$ $q_1 = a_1^\perp/\rho_{1,1}$	 Compute $a_1^\perp$, the component of vector a_1 orthogonal to q_0. Compute $\rho_{1,1}$, the length of $a_1^\perp$. Set $q_1 = a_1^\perp/\rho_{1,1}$, creating a unit vector in the direction of $a_1^\perp$. Now, q_0 and q_1 are mutually orthonormal and $\text{Span}(\{a_0, a_1\}) = \text{Span}(\{q_0, q_1\})$. (Why?)
$\rho_{0,2} = q_0^H a_2$ $\rho_{1,2} = q_1^H a_2$ $a_2^\perp = a_2 - \rho_{0,2} q_0 - \rho_{1,2} q_1$ $\rho_{2,2} = \|a_2^\perp\|_2$ $q_2 = a_2^\perp/\rho_{2,2}$	 Compute $a_2^\perp$, the component of vector a_2 orthogonal to q_0 and q_1. Compute $\rho_{2,2}$, the length of $a_2^\perp$. Set $q_2 = a_2^\perp/\rho_{2,2}$, creating a unit vector in the direction of $a_2^\perp$. Now, $\{q_0, q_1, q_2\}$ is an orthonormal basis and $\text{Span}(\{a_0, a_1, a_2\}) = \text{Span}(\{q_0, q_1, q_2\})$. (Why?)
And so forth.	

Figure 1: Gram-Schmidt orthogonalization.

$$
\begin{array}{l}
\textbf{for } j = 0, \ldots, n-1 \\
\quad a_j^\perp := a_j \\
\quad \textbf{for } k = 0, \ldots, j-1 \\
\quad\quad \rho_{k,j} := q_k^H a_j \\
\quad\quad a_j^\perp := a_j^\perp - \rho_{k,j} q_k \\
\quad \textbf{end} \\
\quad \rho_{j,j} := \|a_j^\perp\|_2 \\
\quad q_j := a_j^\perp / \rho_{j,j} \\
\textbf{end}
\end{array}
$$

$$
\begin{array}{l}
\textbf{for } j = 0, \ldots, n-1 \\
\quad \textbf{for } k = 0, \ldots, j-1 \\
\quad\quad \rho_{k,j} := q_k^H a_j \\
\quad \textbf{end} \\
\quad a_j^\perp := a_j \\
\quad \textbf{for } k = 0, \ldots, j-1 \\
\quad\quad a_j^\perp := a_j^\perp - \rho_{k,j} q_k \\
\quad \textbf{end} \\
\quad \rho_{j,j} := \|a_j^\perp\|_2 \\
\quad q_j := a_j^\perp / \rho_{j,j} \\
\textbf{end}
\end{array}
$$

$$
\begin{array}{l}
\textbf{for } j = 0, \ldots, n-1 \\
\quad \left(\begin{array}{c} \rho_{0,j} \\ \hline \vdots \\ \hline \rho_{j-1,j} \end{array} \right) := \left(\begin{array}{c} q_0^H a_j \\ \hline \vdots \\ \hline q_{j-1}^H a_j \end{array} \right) = \left(\begin{array}{c|c|c} q_0 & \cdots & q_{j-1} \end{array} \right)^H a_j \\
\quad a_j^\perp := a_j - \left(\begin{array}{c|c|c} q_0 & \cdots & q_{j-1} \end{array} \right) \left(\begin{array}{c} \rho_{0,j} \\ \hline \vdots \\ \hline \rho_{j-1,j} \end{array} \right) \\
\quad \rho_{j,j} := \|a_j^\perp\|_2 \\
\quad q_j := a_j^\perp / \rho_{j,j} \\
\textbf{end}
\end{array}
$$

Figure 2: Three equivalent (Classical) Gram-Schmidt algorithms.

where

$$q_i^H q_j = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \rho_{i,j} = \begin{cases} q_i^H a_j & \text{for } i < j \\ \|a_j - \sum_{i=0}^{j-1} \rho_{i,j} q_i\|_2 & \text{for } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Thus, this relationship between the linearly independent vectors $\{a_j\}$ and the orthonormal vectors $\{q_j\}$ can be concisely stated as

$$A = QR,$$

where A and Q are $m \times n$ matrices ($m \geq n$), $Q^H Q = I$, and R is an $n \times n$ upper triangular matrix.

Theorem 3. Let A have linearly independent columns, $A = QR$ where $A, Q \in \mathbb{C}^{m \times n}$ with $n \leq m$, $R \in \mathbb{C}^{n \times n}$, $Q^H Q = I$, and R is an upper triangular matrix with nonzero diagonal entries. Then, for $0 < k < n$, the first k columns of A span the same space as the first k columns of Q.

Proof: Partition

$$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array} \right), \quad Q \rightarrow \left(\begin{array}{c|c} Q_L & Q_R \end{array} \right), \quad \text{and} \quad R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right),$$

where $A_L, Q_L \in \mathbb{C}^{m \times k}$ and $R_{TL} \in \mathbb{C}^{k \times k}$. Then R_{TL} is nonsingular (since it is upper triangular and has no zero on its diagonal), $Q_L^H Q_L = I$, and $A_L = Q_L R_{TL}$. We want to show that $\mathcal{C}(A_L) = \mathcal{C}(Q_L)$:

- We first show that $\mathcal{C}(A_L) \subseteq \mathcal{C}(Q_L)$. Let $y \in \mathcal{C}(A_L)$. Then there exists $x \in \mathbb{C}^k$ such that $y = A_L x$. But then $y = Q_L z$, where $z = R_{TL} x \neq 0$, which means that $y \in \mathcal{C}(Q_L)$. Hence $\mathcal{C}(A_L) \subseteq \mathcal{C}(Q_L)$.
- We next show that $\mathcal{C}(Q_L) \subseteq \mathcal{C}(A_L)$. Let $y \in \mathcal{C}(Q_L)$. Then there exists $z \in \mathbb{C}^k$ such that $y = Q_L z$. But then $y = A_L x$, where $x = R_{TL}^{-1} z$, from which we conclude that $y \in \mathcal{C}(A_L)$. Hence $\mathcal{C}(Q_L) \subseteq \mathcal{C}(A_L)$.

Since $\mathcal{C}(A_L) \subseteq \mathcal{C}(Q_L)$ and $\mathcal{C}(Q_L) \subseteq \mathcal{C}(A_L)$, we conclude that $\mathcal{C}(Q_L) = \mathcal{C}(A_L)$. $\square$

Theorem 4. Let $A \in \mathbb{C}^{m \times n}$ have linearly independent columns. Then there exist $Q \in \mathbb{C}^{m \times n}$ with $Q^H Q = I$ and upper triangular R with no zeroes on the diagonal such that $A = QR$. **This is known as the QR factorization.** If the diagonal elements of R are chosen to be real and positive, th QR factorization is unique.

Proof: (By induction). Note that $n \leq m$ since A has linearly independent columns.

- **Base case:** $n = 1$. In this case $A = \left(a_0 \right)$ where a_0 is its only column. Since A has linearly independent columns, $a_0 \neq 0$. Then

  $$A = \left(a_0 \right) = (q_0)(\rho_{00}),$$

  where $\rho_{00} = \|a_0\|_2$ and $q_0 = a_0 / \rho_{00}$, so that $Q = (q_0)$ and $R = (\rho_{00})$.

Algorithm: $[Q, R] := \mathrm{QR}(A)$

Partition $A \rightarrow \left(A_L \,\|\, A_R \right)$,
$Q \rightarrow \left(Q_L \,\|\, Q_R \right)$,
$R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right)$

where A_L and Q_L has 0 columns and R_{TL} is 0×0

while $n(A_L) \neq n(A)$ **do**

Repartition

$$\left(A_L \,\|\, A_R \right) \rightarrow \left(A_0 \,\|\, a_1 \mid A_2 \right), \quad \left(Q_L \,\|\, Q_R \right) \rightarrow \left(Q_0 \,\|\, q_1 \mid Q_2 \right),$$

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

where a_1 and q_1 are columns, ρ_{11} is a scalar

$r_{01} := Q_0^T a_1$
$a_1^\perp := a_1 - Q_0 r_{01}$
$\rho_{11} := \|a_1^\perp\|_2$
$q_1 := a_1^\perp / \rho_{11}$

Continue with

$$\left(A_L \,\|\, A_R \right) \leftarrow \left(A_0 \mid a_1 \,\|\, A_2 \right), \quad \left(Q_L \,\|\, Q_R \right) \leftarrow \left(Q_0 \mid q_1 \,\|\, Q_2 \right),$$

$$\left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

endwhile

for $j = 0, \ldots, n-1$

$$\underbrace{\begin{pmatrix} \rho_{0,j} \\ \vdots \\ \rho_{j-1,j} \end{pmatrix}}_{r_{01}} := \underbrace{\begin{pmatrix} q_0 & \cdots & q_{j-1} \end{pmatrix}^H}_{Q_0^H} \underbrace{a_j}_{a_1}$$

$$\underbrace{a_j^\perp}_{a_1^\perp} := \underbrace{a_j}_{a_1} - \underbrace{\begin{pmatrix} q_0 & \cdots & q_{j-1} \end{pmatrix}}_{Q_0} \underbrace{\begin{pmatrix} \rho_{0,j} \\ \vdots \\ \rho_{j-1,j} \end{pmatrix}}_{r_{01}}$$

$\rho_{j,j} := \|a_j^\perp\|_2 \qquad (\rho_{11} := \|a_1^\perp\|_2)$
$q_j := a_j^\perp / \rho_{j,j} \qquad (q_1 := a_1^\perp / \rho_{11})$

end

Figure 3: (Classical) Gram-Schmidt algorithm for computing the QR factorization of a matrix A.

- **Inductive step:** Assume that the result is true for all A with $n-1$ linearly independent columns. We will show it is true for $A \in \mathbb{C}^{m \times n}$ with linearly independent columns.

  Let $A \in \mathbb{C}^{m \times n}$. Partition $A \rightarrow \left(A_0 \mid a_1 \right)$. By the induction hypothesis, there exist Q_0 and R_{00} such that $Q_0^H Q_0 = I$, R_{00} is upper triangular with nonzero diagonal entries and $A_0 = Q_0 R_{00}$. Now, compute $r_{01} = Q_0^H a_1$ and $a_1^\perp = a_1 - Q_0 r_{01}$, the component of a_1 orthogonal to $\mathcal{C}(Q_0)$. Because the columns of A are linearly independent, $a_1^\perp \neq 0$. Let $\rho_{11} = \|a_1^\perp\|_2$ and $q_1 = a_1^\perp / \rho_{11}$. Then

$$\begin{aligned} \left(Q_0 \mid q_1 \right) \left(\begin{array}{c|c} R_{00} & r_{01} \\ \hline 0 & \rho_{11} \end{array} \right) &= \left(Q_0 R_{00} \mid Q_0 r_{01} + q_1 \rho_{11} \right) \\ &= \left(A_0 \mid Q_0 r_{01} + a_1^\perp \right) = \left(A_0 \mid a_1 \right) = A. \end{aligned}$$

  Hence $Q = \left(Q_0 \mid q_1 \right)$ and $R = \left(\begin{array}{c|c} R_{00} & r_{01} \\ \hline 0 & \rho_{11} \end{array} \right)$.

- **By the Principle of Mathematical Induction** the result holds for all matrices $A \in \mathbb{C}^{m \times n}$ with $m \geq n$.

□

The proof motivates the algorithm in Figure 3 (left) in FLAME notation[1].

An alternative for motivating that algorithm is as follows: Consider $A = QR$. Partition A, Q, and R to yield

$$\left(\begin{array}{c|c|c} A_0 & a_1 & A_2 \end{array} \right) = \left(\begin{array}{c|c|c} Q_0 & q_1 & Q_2 \end{array} \right) \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right).$$

Assume that Q_0 and R_{00} have already been computed. Since corresponding columns of both sides must be equal, we find that

$$a_1 = Q_0 r_{01} + q_1 \rho_{11}. \quad (1)$$

Also, $Q_0^H Q_0 = I$ and $Q_0^H q_1 = 0$, since the columns of Q are mutually orthonormal. Hence $Q_0^H a_1 = Q_0^H Q_0 r_{01} + Q_0^H q_1 \rho_{11} = r_{01}$. This shows how r_{01} can be computed from Q_0 and a_1, which are already known. Next, $a_1^\perp = a_1 - Q_0 r_{01}$ is computed from (1). This is the component of a_1 that is perpendicular to the columns of Q_0. We know it is nonzero since the columns of A are linearly independent. Since $\rho_{11} q_1 = a_1^\perp$ and we know that q_1 has unit length, we now compute $\rho_{11} = \|a_1^\perp\|_2$ and $q_1 = a_1^\perp / \rho_{11}$, which completes a derivation of the algorithm in Figure 3.

Exercise 5. Let A have linearly independent columns and let $A = QR$ be a QR factorization of A. Partition

$$A \rightarrow \left(\begin{array}{c|c} A_L & A_R \end{array} \right), \quad Q \rightarrow \left(\begin{array}{c|c} Q_L & Q_R \end{array} \right), \quad \text{and} \quad R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right),$$

where A_L and Q_L have k columns and R_{TL} is $k \times k$. Show that

1. $A_L = Q_L R_{TL}$: $Q_L R_{TL}$ equals the QR factorization of A_L,
2. $\mathcal{C}(A_L) = \mathcal{C}(Q_L)$: the first k columns of Q form an orthonormal basis for the space spanned by the first k columns of A.
3. $R_{TR} = Q_L^H A_R$,
4. $(A_R - Q_L R_{TR})^H Q_L = 0$,
5. $A_R - Q_L R_{TR} = Q_R R_{BR}$, and
6. $\mathcal{C}(A_R - Q_L R_{TR}) = \mathcal{C}(Q_R)$.

2 Modified Gram-Schmidt process

We start by considering the following problem: Given $y \in \mathbb{C}^m$ and $Q \in \mathbb{C}^{m \times k}$ with orthonormal columns, compute $y^\perp$, the component of y orthogonal to the columns of Q. This is a key step in the Gram-Schmidt process in Figure 3.

Recall that if A has linearly independent columns, then $A(A^H A)^{-1} A^H y$ equals the projection of y onto the columns space of A (i.e., the component of y in $\mathcal{C}(A)$) and $y - A(A^H A)^{-1} A^H y = (I - A(A^H A)^{-1} A^H) y$ equals the component of y orthogonal to $\mathcal{C}(A)$. If Q has orthonormal columns, then $Q^H Q = I$ and hence

[1] The FLAME notation should be intuitively obvious. If it is not, you may want to review the earlier weeks in Linear Algebra: Foundations to Frontiers - Notes to LAFF With.

$[y^\perp, r] = \text{Proj\_orthog\_to\_Q}_{\text{CGS}}(Q, y)$ (used by classical Gram-Schmidt)	$[y^\perp, r] = \text{Proj\_orthog\_to\_Q}_{\text{MGS}}(Q, y)$ (used by modified Gram-Schmidt)

$y^\perp = y$
for $i = 0, \ldots, k-1$
  $\rho_i := q_i^H y$
  $y^\perp := y^\perp - \rho_i q_i$
endfor
	$y^\perp = y$
	for $i = 0, \ldots, k-1$
	  $\rho_i := q_i^H y^\perp$
	  $y^\perp := y^\perp - \rho_i q_i$
	endfor

Figure 4: Two different ways of computing $y^\perp = (I - QQ^H)y$, the component of y orthogonal to $\mathcal{C}(Q)$, where Q has k orthonormal columns.

Algorithm: $[AR] := \text{Gram-Schmidt}(A)$ (overwrites A with Q)

Partition $A \rightarrow \left(A_L \,\middle|\, A_R \right)$, $R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right)$

where A_L has 0 columns and R_{TL} is 0×0

while $n(A_L) \neq n(A)$ **do**

Repartition

$$\left(A_L \,\middle|\, A_R \right) \rightarrow \left(A_0 \,\middle|\, a_1 \,\middle|\, A_2 \right), \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

where a_1 and q_1 are columns, ρ_{11} is a scalar

CGS	MGS	MGS (alternative)
$r_{01} := A_0^H a_1$		
$a_1 := a_1 - A_0 r_{01}$	$[a_1, r_{01}] = \text{Proj\_orthog\_to\_Q}_{\text{MGS}}(A_0, a_1)$	
$\rho_{11} := \Vert a_1 \Vert_2$	$\rho_{11} := \Vert a_1 \Vert_2$	$\rho_{11} := \Vert a_1 \Vert_2$
$a_1 := a_1 / \rho_{11}$	$q_1 := a_1 / \rho_{11}$	$a_1 := a_1 / \rho_{11}$
		$r_{12}^T := a_1^H A_2$
		$A_2 := A_2 - a_1 r_{12}^T$

Continue with

$$\left(A_L \,\middle|\, A_R \right) \leftarrow \left(A_0 \,\middle|\, a_1 \,\middle|\, A_2 \right), \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

endwhile

Figure 5: Left: Classical Gram-Schmidt algorithm. Middle: Modified Gram-Schmidt algorithm. Right: Modified Gram-Schmidt algorithm where every time a new column of Q, q_1 is computed the component of all future columns in the direction of this new vector are subtracted out.

$QQ^H y$ equals the projection of y onto the columns space of Q (i.e., the component of y in $\mathcal{C}(Q)$) and $y - QQ^H y = (I - QQ^H)y$ equals the component of y orthogonal to $\mathcal{C}(A)$.

Thus, mathematically, the solution to the stated problem is given by

$$\begin{aligned} y^\perp &= (I - QQ^H)y = y - QQ^H y \\ &= y - \left(q_0 \,\middle|\, \cdots \,\middle|\, q_{k-1} \right) \left(q_0 \,\middle|\, \cdots \,\middle|\, q_{k-1} \right)^H y \end{aligned}$$

$$
\begin{aligned}
&= y - \left(q_0 \mid \cdots \mid q_{k-1} \right) \begin{pmatrix} q_0^H \\ \hline \vdots \\ \hline q_{k-1}^H \end{pmatrix} y \\
&= y - \left(q_0 \mid \cdots \mid q_{k-1} \right) \begin{pmatrix} q_0^H y \\ \hline \vdots \\ \hline q_{k-1}^H y \end{pmatrix} \\
&= y - \left[(q_0^H y) q_0 + \cdots + (q_{k-1}^H y) q_{k-1} \right] \\
&= y - (q_0^H y) q_0 - \cdots - (q_{k-1}^H y) q_{k-1}.
\end{aligned}
$$

This can be computed by the algorithm in Figure 4 (left) and is used by what is often called the *Classical* Gram-Schmidt (CGS) algorithm given in Figure 3.

An alternative algorithm for computing $y^\perp$ is given in Figure 4 (right) and is used by the *Modified* Gram-Schmidt (MGS) algorithm also given in Figure 5. This approach is mathematically equivalent to the algorithm to its left for the following reason:

The algorithm on the left in Figure 4 computes

$$
y^\perp := y - (q_0^H y) q_0 - \cdots - (q_{k-1}^H y) q_{k-1}
$$

by in the ith step computing the component of y in the direction of q_i, $(q_i^H y) q_i$, and then subtracting this off the vector $y^\perp$ that already contains

$$
y^\perp = y - (q_0^H y) q_0 - \cdots - (q_{i-1}^H y) q_{i-1},
$$

leaving us with

$$
y^\perp = y - (q_0^H y) q_0 - \cdots - (q_{i-1}^H y) q_{i-1} - (q_i^H y) q_i.
$$

Now, notice that

$$
\begin{aligned}
q_i^H \left[y - (q_0^H y) q_0 - \cdots - (q_{i-1}^H y) q_{i-1} \right] &= q_i^H y - q_i^H (q_0^H y) q_0 - \cdots - q_i^H (q_{i-1}^H y) q_{i-1} \\
&= q_i^H y - (q_0^H y) \underbrace{q_i^H q_0}_{0} - \cdots - (q_{i-1}^H y) \underbrace{q_i^H q_{i-1}}_{0} \\
&= q_i^H y.
\end{aligned}
$$

What this means is that we can use $y^\perp$ in our computation of ρ_i instead:

$$
\rho_i := q_i^H y^\perp = q_i^H y,
$$

an insight that justifies the equivalent algorithm in Figure 4 (right).

Next, we massage the MGS algorithm into the third (right-most) algorithm given in Figure 5. For this, consider the equivalent algorithms in Figure 6 and 7.

3 In Practice, MGS is More Accurate

In theory, all Gram-Schmidt algorithms discussed in the previous sections are equivalent: they compute the exact same QR factorizations. In practice, in the presense of round-off error, MGS is more accurate than CGS. We will (hopefully) get into detail about this later, but for now we will illustrate it with a classic example.

When storing real (or complex for that matter) valued numbers in a computer, a limited accuracy can be maintained, leading to round-off error when a number is stored and/or when computation with numbers

for $j = 0, \ldots, n-1$
  $a_j^\perp := a_j$
  for $k = 0, \ldots, j-1$
    $\rho_{k,j} := q_k^H a_j^\perp$
    $a_j^\perp := a_j^\perp - \rho_{k,j} q_k$
  end
  $\rho_{j,j} := \|a_j^\perp\|_2$
  $q_j := a_j^\perp / \rho_{j,j}$
end

(a) MGS algorithm that computes Q and R from A.

for $j = 0, \ldots, n-1$

  for $k = 0, \ldots, j-1$
    $\rho_{k,j} := a_k^H a_j$
    $a_j := a_j - \rho_{k,j} a_k$
  end
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
end

(b) MGS algorithm that computes Q and R from A, overwriting A with Q.

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
  for $k = j+1, \ldots, n-1$
    $\rho_{j,k} := a_j^H a_k$

    $a_k := a_k - \rho_{j,j} a_j$
  end
end

(c) MGS algorithm that normalizes the jth column to have unit length to compute q_j (overwriting a_j with the result) and then subtracts the component in the direction of q_j off the rest of the columns $(a_{j+1}, \ldots, a_{n-1})$.

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
  for $k = j+1, \ldots, n-1$
    $\rho_{j,k} := a_j^H a_k$
  end
  for $k = j+1, \ldots, n-1$
    $a_k := a_k - \rho_{j,k} a_j$
  end
end

(d) Slight modification of the algorithm in (c) that computes $\rho_{j,k}$ in a separate loop.

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
  $\left(\rho_{j,j+1} \mid \cdots \mid \rho_{j,n-1} \right) :=$
    $\left(a_j^H a_{j+1} \mid \cdots \mid a_j^H a_{n-1} \right)$
  $\left(a_{j+1} \mid \cdots \mid a_{n-1} \right) :=$
    $\left(a_{j+1} - \rho_{j,j+1} a_j \mid \cdots \mid a_{n-1} - \rho_{j,n-1} a_j \right)$
end

(e) Algorithm in (d) rewritten without loops.

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2$
  $a_j := a_j / \rho_{j,j}$
  $\left(\rho_{j,j+1} \mid \cdots \mid \rho_{j,n-1} \right) :=$
    $a_j^H \left(a_{j+1} \mid \cdots \mid a_{n-1} \right)$
  $\left(a_{j+1} \mid \cdots \mid a_{n-1} \right) :=$
    $\left(a_{j+1} \mid \cdots \mid a_{n-1} \right) - a_j \left(\rho_{j,j+1} \mid \cdots \mid \rho_{j,n-1} \right)$
end

(f) Algorithm in (e) rewritten to expose the row-vector-times matrix multiplication $a_j^H \left(a_{j+1} \mid \cdots \mid a_{n-1} \right)$ and rank-1 update $\left(a_{j+1} \mid \cdots \mid a_{n-1} \right) - a_j \left(\rho_{j,j+1} \mid \cdots \mid \rho_{j,n-1} \right)$.

Figure 6: Various equivalent MGS algorithms.

Algorithm: $[A, R] := \mathrm{QR}(A)$

Partition $A \rightarrow \left(A_L \,\big|\, A_R \right)$,
$R \rightarrow \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right)$
  where A_L and Q_L has 0 columns and R_{TL} is 0×0

while $n(A_L) \neq n(A)$ **do**

  Repartition

$$\left(A_L \,\big|\, A_R \right) \rightarrow \left(A_0 \,\big|\, a_1 \,|\, A_2 \right), \quad \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \rightarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

  where a_1 is a column, ρ_{11} is a scalar

  $\rho_{11} := \|a_1\|_2$
  $a_1 := a_1 / \rho_{11}$
  $r_{12}^T := a_1^H A_2$
  $A_2 := A_2 - a_1 r_{12}^T$

  Continue with

$$\left(A_L \,\big|\, A_R \right) \leftarrow \left(A_0 \,|\, a_1 \,\big|\, A_2 \right), \quad \left(\begin{array}{c|c} R_{TL} & R_{TR} \\ \hline 0 & R_{BR} \end{array} \right) \leftarrow \left(\begin{array}{c|c|c} R_{00} & r_{01} & R_{02} \\ \hline 0 & \rho_{11} & r_{12}^T \\ \hline 0 & 0 & R_{22} \end{array} \right)$$

endwhile

for $j = 0, \ldots, n-1$
  $\rho_{j,j} := \|a_j\|_2 \qquad (\rho_{11} := \|a_1^\perp\|_2)$
  $a_j := a_j / \rho_{j,j} \qquad (a_1 := a_1/\rho_{11})$

$$\overbrace{\left(\rho_{j,j+1} \,|\, \cdots \,|\, \rho_{j,n-1} \right)}^{r_{12}^T} := \underbrace{a_j^H}_{a_1^H} \underbrace{\left(a_{j+1} \,|\, \cdots \,|\, a_{n-1} \right)}_{A_2}$$

$$\overbrace{\left(a_{j+1} \,|\, \cdots \,|\, a_{n-1} \right)}^{A_2} := \overbrace{\left(a_{j+1} \,|\, \cdots \,|\, a_{n-1} \right)}^{A_2} - \underbrace{a_j}_{a_1} \underbrace{\left(\rho_{j,j+1} \,|\, \cdots \,|\, \rho_{j,n-1} \right)}_{r_{12}^T}$$

end

Figure 7: Modified Gram-Schmidt algorithm for computing the QR factorization of a matrix A.

are performed. The *machine epsilon* or *unit roundoff error* is defined as the largest positive number ϵ_{mach} such that the stored value of $1 + \epsilon_{\mathrm{mach}}$ is rounded to 1. Now, let us consider a computer where the **only** error that is ever incurred is when $1 + \epsilon_{\mathrm{mach}}$ is computed and rounded to 1. Let $\epsilon = \sqrt{\epsilon_{\mathrm{mach}}}$ and consider the matrix

$$A = \left(\begin{array}{c|c|c} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{array} \right) = \left(a_0 \,|\, a_1 \,|\, a_2 \right) \tag{2}$$

In Figure 8 (left) we execute the CGS algorithm. It yields the approximate matrix

$$Q \approx \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{array} \right)$$

If we now ask the question "Are the columns of Q orthonormal?" we can check this by computing $Q^H Q$, First iteration

$\rho_{0,0} = \|a_0\|_2 = \sqrt{1+\epsilon^2} = \sqrt{1+\epsilon_{\rm mach}}$

which is rounded to 1.

$$q_0 = a_0/\rho_{0,0} = \begin{pmatrix} 1 \\ \epsilon \\ 0 \\ 0 \end{pmatrix} /1 = \begin{pmatrix} 1 \\ \epsilon \\ 0 \\ 0 \end{pmatrix}$$

Second iteration

$\rho_{0,1} = q_0^H a_1 = 1$

$$a_1^\perp = a_1 - \rho_{0,1} q_0 = \begin{pmatrix} 0 \\ -\epsilon \\ \epsilon \\ 0 \end{pmatrix}$$

$\rho_{1,1} = \|a_1^\perp\|_2 = \sqrt{2\epsilon^2} = \sqrt{2}\epsilon$

$$q_1 = a_1^\perp/\rho_{1,1} = \begin{pmatrix} 0 \\ -\epsilon \\ \epsilon \\ 0 \end{pmatrix} /(\sqrt{2}\epsilon) = \begin{pmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{pmatrix}$$

Third iteration

$\rho_{0,2} = q_0^H a_2 = 1$

$\rho_{1,2} = q_1^H a_2 = 0$

$$a_2^\perp = a_2 - \rho_{0,2} q_0 - \rho_{1,2} q_1 = \begin{pmatrix} 0 \\ -\epsilon \\ 0 \\ \epsilon \end{pmatrix}$$

$\rho_{2,2} = \|a_2^\perp\|_2 = \sqrt{2\epsilon^2} = \sqrt{2}\epsilon$

$$q_2 = a_2^\perp/\rho_{2,2} = \begin{pmatrix} 0 \\ -\epsilon \\ 0 \\ \epsilon \end{pmatrix} /(\sqrt{2}\epsilon) = \begin{pmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ 0 \\ \frac{\sqrt{2}}{2} \end{pmatrix}$$

First iteration

$\rho_{0,0} = \|a_0\|_2 = \sqrt{1+\epsilon^2} = \sqrt{1+\epsilon_{\rm mach}}$

which is rounded to 1.

$$q_0 = a_0/\rho_{0,0} = \begin{pmatrix} 1 \\ \epsilon \\ 0 \\ 0 \end{pmatrix} /1 = \begin{pmatrix} 1 \\ \epsilon \\ 0 \\ 0 \end{pmatrix}$$

Second iteration

$\rho_{0,1} = q_0^H a_1 = 1$

$$a_1^\perp = a_1 - \rho_{0,1} q_0 = \begin{pmatrix} 0 \\ -\epsilon \\ \epsilon \\ 0 \end{pmatrix}$$

$\rho_{1,1} = \|a_1^\perp\|_2 = \sqrt{2\epsilon^2} = \sqrt{2}\epsilon$

$$q_1 = a_1^\perp/\rho_{1,1} = \begin{pmatrix} 0 \\ -\epsilon \\ \epsilon \\ 0 \end{pmatrix} /(\sqrt{2}\epsilon) = \begin{pmatrix} 0 \\ -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \\ 0 \end{pmatrix}$$

Third iteration

$\rho_{0,2} = q_0^H a_2 = 1$

$$a_2^\perp = a_2 - \rho_{0,2} q_0 = \begin{pmatrix} 0 \\ -\epsilon \\ 0 \\ \epsilon \end{pmatrix}$$

$\rho_{1,2} = q_1^H a_2^\perp = (\sqrt{2}/2)\epsilon$

$$a_2^\perp = a_2^\perp - \rho_{1,2} q_1 = \begin{pmatrix} 0 \\ -\epsilon/2 \\ -\epsilon/2 \\ \epsilon \end{pmatrix}$$

$\rho_{2,2} = \|a_2^\perp\|_2 = \sqrt{(6/4)\epsilon^2} = (\sqrt{6}/2)\epsilon$

$$q_2 = a_2^\perp/\rho_{2,2} = \begin{pmatrix} 0 \\ -\frac{\epsilon}{2} \\ -\frac{\epsilon}{2} \\ \epsilon \end{pmatrix} /(\frac{\sqrt{6}}{2}\epsilon) = \begin{pmatrix} 0 \\ \frac{\sqrt{6}}{6} \\ -\frac{\sqrt{6}}{6} \\ -\frac{2\sqrt{6}}{6} \end{pmatrix}$$

Figure 8: Execution of the CGS algorith (left) and MGS algorithm (right) on the example in Eqn. (2).

which should equal I, the identity. But

$$Q^HQ = \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{array}\right)^H \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{array}\right) = \left(\begin{array}{ccc} 1+\epsilon_{\rm mach} & -\frac{\sqrt{2}}{2}\epsilon & -\frac{\sqrt{2}}{2}\epsilon \\ -\frac{\sqrt{2}}{2}\epsilon & 1 & \frac{1}{2} \\ -\frac{\sqrt{2}}{2}\epsilon & \frac{1}{2} & 1 \end{array}\right).$$

Clearly, the computed columns of Q are **not** mutually orthogonal.

Similarly, in Figure 8 (right) we execute the MGS algorithm. It yields the approximate matrix

$$Q \approx \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & \frac{\sqrt{6}}{6} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & 0 & \frac{2\sqrt{6}}{6} \end{array}\right).$$

If we now ask the question "Are the columns of Q orthonormal?" we can check if $Q^HQ = I$. The answer:

$$Q^HQ = \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & 0 & \frac{2\sqrt{6}}{6} \end{array}\right)^H \left(\begin{array}{c|c|c} 1 & 0 & 0 \\ \epsilon & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{6}}{6} \\ 0 & 0 & \frac{2\sqrt{6}}{6} \end{array}\right) = \left(\begin{array}{ccc} 1+\epsilon_{\rm mach} & -\frac{\sqrt{2}}{2}\epsilon & -\frac{\sqrt{6}}{6}\epsilon \\ -\frac{\sqrt{2}}{2}\epsilon & 1 & 0 \\ -\frac{\sqrt{6}}{6}\epsilon & 0 & 1 \end{array}\right),$$

which shows that for this example MGS yields better orthogonality than does CGS. What is going on? The answer lies with how $a_2^\perp$ is computed in the last step of each of the algorithms.

- In the CGS algorithm, we find that

$$a_2^\perp := a_2 - (q_0^H a_2)q_0 - (q_1^H a_2)q_1.$$

  Now, q_0 has a relatively small error in it and hence $q_0^H a_2 q_0$ has a relatively) small error in it. It is likely that a part of that error is in the direction of q_1. Relative to $q_0^H a_2 q_0$, that error in the direction of q_1 is small, but relative to $a_2 - q_0^H a_2 q_0$ it is not. The point is that then $a_2 - q_0^H a_2 q_0$ has a relatively large error in it in the direction of q_1. Subtracting $q_1^H a_2 q_1$ does not fix this and since in the end $a_2^\perp$ is small, it has a relatively large error in the direction of q_1. This error is amplified when q_2 is computed by normalizing $a_2^\perp$.

- In the MGS algorithm, we find that

$$a_2^\perp := a_2 - (q_0^H a_2)q_0$$

  after which

$$a_2^\perp := a_2^\perp - q_1^H a_2^\perp q_1 = [a_2 - (q_0^H a_2)q_0] - (q_1^H [a_2 - (q_0^H a_2)q_0])q_1.$$

  This time, if $a_2 - q_1^H a_2^\perp q_1$ has an error in the direction of q_1, this error is subtracted out when $(q_1^H a_2^\perp)q_1$ is subtracted from $a_2^\perp$. This explains the better orthogonality between the computed vectors q_1 and q_2.

Obviously, we have argued via an example that MGS is more accurage than CGS. A more thorough analysis is needed to explain why this is generally so. This is beyond the scope of this note.

4 Modified Gram-Schmidt process

Let us examine the cost of computing the QR factorization of an $m \times n$ matrix A. We will count multiplies and an adds as each as one floating point operation.

We start by reviewing the cost, in floating point operations (flops), of various vector-vector and matrix-vector operations:

Name	Operation	Approximate cost (in flops)
Vector-vector operations ($x, y \in \mathbb{C}^n, \alpha \in \mathbb{C}$)		
Dot	$\alpha := x^H y$	$2n$
Axpy	$y := \alpha x + y$	$2n$
Scal	$x := \alpha x$	n
Nrm2	$\alpha := \|a_1\|_2$	$2n$
Matrix-vector operations ($A \in \mathbb{C}^{m \times n}, \alpha, \beta \in \mathbb{C}$, with x and y vectors of appropriate size)		
Matrix-vector multiplication (Gemv)	$y := \alpha A x + \beta y$ $y := \alpha A^H x + \beta y$	$2mn$ $2mn$
Rank-1 update (Ger)	$A := \alpha y x^H + A$	$2mn$

Now, consider the algorithms in Figure 5. Notice that the columns of A are of size m. During the kth iteration ($0 \leq k < n$), A_0 has k columns and A_2 has $n-k-1$ columns.

4.1 Cost of CGS

Operation	Approximate cost (in flops)
$r_{01} := A_0^H a_1$	$2mk$
$a_1 := a_1 - A_0 r_{01}$	$2mk$
$\rho_{11} := \|a_1\|_2$	$2m$
$a_1 := a_1 / \rho_{11}$	m

Thus, the total cost is (approximately)

$$\begin{aligned}
&\sum_{k=0}^{n-1} [2mk + 2mk + 2m + m] \\
&= \sum_{k=0}^{n-1} [3m + 4mk] \\
&= 3mn + 4m \sum_{k=0}^{n-1} k \\
&\approx 3mn + 4m \frac{n^2}{2} \qquad (\textstyle\sum_{k=0}^{n-1} k = n(n-1)/2 \approx n^2/2 \\
&= 3mn + 2mn^2 \\
&\approx 2mn^2 \qquad (3mn \text{ is of lower order}).
\end{aligned}$$

4.2 Cost of MGS

Operation	Approximate cost (in flops)
$\rho_{11} := \|a_1\|_2$	$2m$
$a_1 := a_1 / \rho_{11}$	m
$r_{12}^T := a_1^H A_2$	$2m(n-k-1)$
$A_2 := A_2 - a_1 r_{12}^T$	$2m(n-k-1)$

Thus, the total cost is (approximately)

$$
\begin{aligned}
\sum_{k=0}^{n-1}\,[2m+m+2m(n-k-1)+2m(n-k-1)] \\
&= \sum_{k=0}^{n-1}\,[3m+4m(n-k-1)] \\
&= 3mn+4m\sum_{k=0}^{n-1}(n-k-1) \\
&= 3mn+4m\sum_{i=0}^{n-1} i && \text{(Change of variable: } i=n-k-1\text{)} \\
&\approx 3mn+4m\frac{n^2}{2} && \textstyle(\sum_{i=0}^{n-1} i = n(n-1)/2 \approx n^2/2 \\
&= 3mn+2mn^2 \\
&\approx 2mn^2 && (3mn \text{ is of lower order}).
\end{aligned}
$$