QR Factorization or Subspace-based Methods

Suppose we have the factorization $A = QR$ where $Q \in \mathbb{R}^{n \times k}$ and $R \in \mathbb{R}^{k \times k}$ described earlier.

We know that the solution to $Ax = b$ that minimizes is given by the solution to

$$Ax = b_1$$

where $b = b_1 + b_2$, $b_1 \in \mathcal{R}(A)$ and $b_2^T b_1 = 0$ uniquely.

We have discussed how to compute these projections:

$$b_1 = QQ^T b$$
$$Ax = b_1$$
Substitution yields

\[ QRx = QQ^T b \]
\[ Rx = Q^T b \]
\[ Rx = y \]

So, **once the factorization is known**, the method consists of

- **compute** \( y = Q^T b \)
- **solve** \( Rx = y \)
How do we compute $QR$ from $A$?

There are many ways. In fact, as we have seen the normal equations can be adapted to do it, and we will see that Householder transformations can also be used to do it. Here we present a more straightforward method – Gram-Schmidt orthogonalization.

**NOTE:** Gram-Schmidt in various disguises and forms is a fundamental algorithmic technique in numerical linear algebra and many other areas.
Assume we have

\[ A = [a_1, \ldots, a_k] \]

We want to produce an orthogonal matrix

\[ Q = [q_1, \ldots, q_k] \]

such that \( A = QR \) where \( R \) is upper triangular.
Suppose \( k = 1 \) and let \( \rho = \|a_1\|_2 = \sqrt{a_1^T a_1} \) 

Define \( q_1 = a_1 \rho^{-1} \) and we have 

\[
q_1 \rho = a_1
\]

which is the desired \( n \times 1 \) \( QR \) factorization.
Now suppose we have \( q_1, \ldots, q_i \), and \( R_i \in \mathbb{R}^{i \times i} \) such that

\[
Q_i R_i = [q_1, \ldots, q_i] R_i = [a_1, \ldots, a_i]
\]

and we want to create \( q_{i+1} \) and \( R_{i+1} \) from \( a_{i+1} \).

The part of \( a_{i+1} \) that is in \( \mathcal{R}(Q_i) \) is of no interest since it is already in the space that has been orthogonalized.
We need the projection of $a_{i+1}$ onto $\mathcal{R}^\perp(Q_i)$

$$v = (I - Q_iQ_i^T)a_{i+1}$$

is in the direction that contains new information to be added to the basis.

We also know that $v^Tq_j = 0$ for $j = 1, \ldots, i$ therefore we only have to normalize its length to 1 and take

$$q_{i+1} = v(\|v\|_2)^{-1}$$
This yields the Classical Gram-Schmidt orthogonalization algorithm.

\[
\begin{align*}
    r &= Q_i^T a_{i+1} \\
    v &= a_{i+1} - Q_i r \\
    \rho &= ||v||_2 \\
    q_{i+1} &= v(||v||_2)^{-1}
\end{align*}
\]

We have then

\[
\begin{pmatrix}
    Q_i & q_{i+1}
\end{pmatrix}
\begin{pmatrix}
    R_i & r \\
    0^T & \rho
\end{pmatrix} = \begin{pmatrix} a_1, \ldots, a_i, a_{i+1} \end{pmatrix}
\]

which is the desired factorization.
Complexity

\[ r = Q_i^T a_{i+1} \quad \text{costs} \quad 2ni + O(1) \]
\[ v = a_{i+1} - Q_i r \quad \text{costs} \quad 2ni + O(1) \]
\[ \rho = \|v\|_2 \quad \text{costs} \quad 2n + O(1) \]
\[ q_{i+1} = v(\|v\|_2)^{-1} \quad \text{costs} \quad 2n + O(1) \]

\[ \Omega = \sum_{i=1}^{k} 4ni + 2n + O(1) \]
\[ = (\sum_{i=1}^{k} 4ni) + 2nk + O(k) \]
\[ = 4n(\sum_{i=1}^{k} i) + 2nk + O(k) \]
\[ = 2nk^2 + k + 2nk + O(k) \]
\[ = 2nk^2 + 4nk + O(k) \]
\[ = 2nk^2 + O(nk) \]
The method to solve the least squares problem is therefore

\[
\begin{align*}
\text{factor} & \quad A = QR \\
\text{compute} & \quad y = Q^T b \\
\text{solve} & \quad Rx = y
\end{align*}
\]

Which has complexity $2nk^2 + O(nk)$ when $n >> k$. 
• Twice as many operations as the normal equations.

• Classical GS can produce in finite precision a $Q$ which is far from orthogonal.

• As given, this method tends to be unreliable for the least squares problem and for the task of computing an orthogonal basis given $A$.

• The Modified GS algorithm fixes part of this.
Modified Gram-Schmidt

CGS produces $q_1, \ldots, q_i$ and then removes their contributions from $a_{i+1}$ to produce $q_{i+1}$.

The idea behind modified GS is an immediate update of the active part of $A$.

**NOTE:** This is just a motivating idea. As we will see it is not exactly the characterizing difference.
Given $a_i^{(i-1)}$, normalize to produce $q_i$ and remove its contribution from

$$[a_i^{(i-1)}, \ldots, a_k^{(i-1)}]$$

to produce

$$[a_i^{(i)}, \ldots, a_k^{(i)}]$$

**NOTE:** $a_j^{(i)}$ denotes $a_j$ with the contribution of $q_1, \ldots, q_i$ removed.
The $i$th step assumes
\[ q_1, \ldots, q_{i-1} \]
and
\[ a_i^{(i-1)}, \ldots, a_k^{(i-1)} \]
are known.

\[
\begin{align*}
\rho_i &= \|a_i^{(i-1)}\|_2 \\
q_i &= a_i^{(i-1)}/\rho_i \\
r_i^T &= q_i^T [a_{i+1}^{(i-1)}, \ldots, a_k^{(i-1)}] \\
[a_i^{(i)}, \ldots, a_k^{(i)}] &= [a_{i+1}^{(i-1)}, \ldots, a_k^{(i-1)}] - q_i r_i^T
\end{align*}
\]
The vector \( \begin{pmatrix} \rho_i & r_i^T \end{pmatrix} \) is the \( i \)th row of \( R \) in \( A = QR \). The operation count is the same as CGS.

This produces the \( i \)-th column of \( Q \) and the \( i \)-th row of \( R \) on the \( i \)-th step.

CGS produces the \( i \)-th column of \( Q \) and the \( i \)-th column of \( R \) on the \( i \)-th step.

MGS has as its main primitive a rank-1 update.

CSG has as its main primitive a matrix-vector multiplication.
Be Careful about the Essential Difference!!

There is a tendency to think the difference between CGS and MGS comes from delayed versus immediate update of the active part of $A$ since that is the easiest way to derive the algorithm.

Delay is not the essential difference. Product of sums versus Sum of products is the essential difference.

Exactly what does that mean??
Consider the case where we assume that $q_1, \ldots, q_i$ are available and $a_{i+1}, \ldots, a_k$ have not been touched, i.e., both CGS and MGS are delayed update versions.

Now look at the order of two different basic BLAS-1 operations inner product to compute the coefficient and triad to remove the component of $q_j$ from $a_{i+1}$ that is being placed in its final form before scaling, $a_{i+1}^{(i)}$. 
We set $i = 2$ to illustrate the difference.

**Classical GS:**

$$a_3^{(2)} = (I - q_1q_1^T - q_2q_2^T)a_3^{(0)}$$

is to be computed but we must be specific about ordering. We give the grouping and put a subscript around pairs of parentheses at the same level, i.e. all operations inside $(1\cdots)_1$ can be done at the same time or in any order before operations in $(2\cdots)_2$ etc.

$$a_3^{(2)} = (3(2a_3^{(0)} - q_1(1q_1^Ta_3^{(0)})_1)_2 - q_2(1q_2^Ta_3^{(0)})_1)_3$$
Modified GS:

\[ a_3^{(2)} = (I - q_2q_2^T)(I - q_1q_1^T)a_3^{(0)} \]

is to be computed but we must be specific about ordering. We first compute \( \alpha_1 = q_1^T a_3^{(0)} \)

\[ a_3^{(2)} = (5(2a_3^{(0)} - q_1\alpha_1)_2 \\
- 4q_2(3q_2^T(2a_3^{(0)} - q_1\alpha_1)_2)_3)_4)_5 \]
So what? The difference between Sum of products in CGS and Product of sums in MGS is their sensitivity to certain quantities actually being 0 or not under finite precision.

We have

\[(I - q_1q_1^T - q_2q_2^T) = (I - q_2q_2^T)(I - q_1q_1^T)\]

in exact arithmetic because then \(q_2^T q_1 = 0\).

The left form (Sum of products) essentially sets the inner product to 0 and proceeds in finite precision.

The right form (Product of sums) in fact calculates \(q_2^T q_1\) approximately and uses it implicitly and is therefore less sensitive to the actual computed orthogonality of \(q_1\) and \(q_2\).
This is not a rigorous proof but it is the starting point for a rigorous and relatively understandable proof due to Philippe and Jalby. The standard analysis due to Bjorck is much harder to understand intuitively.
• MGS is more stable than CGS and tends to produce a $Q$ which is closer to orthogonal in finite precision.

• Both can produce a $Q$ which is very far from orthogonal.

• CGS and MGS as given are not reliable in computing $Q$.

• CGS is not reliable for least squares problems.

• **BUT** for MGS the deviation from orthogonality in $Q$ is compensated for in errors in $R$ and MGS is as stable as any of the methods for the least squares problem.
The deviation from orthogonality problem in CGS and MGS can also be fixed via reorthogonalization.

Suppose you compute

\[ A = Q_1 R_1 \]

but \( \|Q_1^T Q_1 - I\| \) is not small enough. Treat \( Q_1 \) as a nonorthogonal matrix and apply CGS or MGS to it.

\[ Q_1 = Q_2 R_2 \]
Usually \( \|Q_2^TQ_2 - I\| \) is small enough (if not do it again). Then

\[
A = Q_2R_2R_1 \\
= Q_2(R_2R_1) \\
= QR
\]

where \( Q \) is orthogonal to working precision.

There are other ways to produce \( Q \) based on Householder reflectors so the number of reorthogonalizations is crucial in determining whether it is more or less efficient that the other methods.
Next we must consider:

- a robust transformation-based approach to least squares and producing an orthogonal basis,

- the stability of all methods,

- the conditioning of the least squares problem,

- summarizing the relative worth of the methods.