Normal Equations

This approach exploits knowledge of the orthogonality of the residual.

Recall $A \in \mathbb{R}^{n \times k}$ is full rank, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^k$. The vector $x$ minimizes over all vectors in $\mathbb{R}^k$ the two norm of the residual

$$||b - Ax||_2$$

We have

$$r = b - Ax$$
$$r \perp \mathcal{R}(A)$$
$$0 = A^T r$$
$$0 = A^T b - A^T Ax$$

$$A^T Ax = A^T b$$

$$x = (A^T A)^{-1} A^T b$$

$$x = A^{-\dagger} b$$

$A^{-\dagger}$ is called the Moore-Penrose or pseudoinverse. Note $A^{-\dagger} = A^{-1}$ if $n = k$. 
• $A^T A$ is then symmetric positive definite since is $A$ full rank.

• The inverse must exist and therefore we have a unique $x$.

• Even if $A$ is not full rank the normal equations must have a solution (the residual must be orthogonal to any $x$ that gives a minimal residual norm).

• Uniqueness is lost if $A$ is not full rank and the algorithm for solving the singular system must be carefully designed. (Recall we mentioned briefly in class the notion of a low-rank factorization that produces trapezoidal factors.)
Complexity

- Forming $A^T A$:
  - The matrix is symmetric so only half of its elements are needed.
  - $\frac{k^2}{2}$ inner products are needed each of length $n$.
  - $nk^2 + O(k^2)$ operations
  - dominates the algorithm

- Forming $A^T$: $O(nk)$ – a lower order term.

- Cholesky factorization $A^T A = R^T R$: $\frac{k^3}{3} + O(k^2)$ – a lower order term (but the most significant, always check $k^3$ relative to $n$).

- Triangular solves: $O(k^2)$ – a lower order term

- Total operations: $nk^2 + O(k^3)$
• This approach is the fastest that we will consider in terms of operation count.

• If $A$ is square it would not be used since $A$ would be factored via $LU$ directly rather than via $A^T A$ and the Cholesky factorization.

• The backward error involves a perturbation to $A^T A$ and to guarantee that $A^T A$ is positive definite places contraints on the conditioning of $A$ (measured via $\kappa(A) = \|A\|\|A^\dagger\|$)

• The backward error may be projected back to $A$ but the size of the perturbation is related to $\kappa(A)$

• So this approach is worthwhile only for very well-conditioned problems and has earned (a somewhat undeserved) reputation of being a method to avoid in general. It is in fact used successfully in many applications.
**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:

$$b_1 = QQ^T b$$

$$Ax = b_1$$

$$QRx = QQ^T b$$

$$Rx = Q^T b$$

$$Rx = y$$

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 the Housholder reduction can also be used to do it. Here we present a more straightforward method – Gram-Schmidt orthogonalization.

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 $k = 1$ let $\rho = \|a_1\|_2 = \sqrt{a_1^T a_1}$ and let $q_1 = a_1 \rho^{-1}$ then

$$q_1 \rho = a_1$$

which is the desired $QR$ 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. So we need the projection of \( a_{i+1} \) onto \( \mathcal{R}^\perp(Q_i) \) (we have seen how to compute this).

\[
v = (I - Q_iQ_i^T)a_{i+1}
\]

is in the direction that contains new information to be added to the basis. We also know that \( v^T q_j = 0 \) 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

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

which is the desired factorization.
Complexity

The Classical GS algorithm requires

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

\[ \Omega = \sum_{i=1}^k 4ni + 2n + O(1) \]
\[ = (\sum_{i=1}^k 4ni) + 2nk + O(k) \]
\[ = 4n(\sum_{i=1}^k i) + 2nk + O(k) \]
\[ = 2nk^2 + k + 2nk + O(k) \]
\[ = 2nk^2 + 4nk + O(k) \]
\[ = 2nk^2 + O(nk) \]

So, the method consists of

\[ \text{factor} \quad A = QR \]
\[ \text{compute} \quad y = Q^T b \]
\[ \text{solve} \quad Rx = y \]

Which has complexity \(2nk^2 + O(nk)\) when \(n \gg k\).
• 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

Rather than producing \( q_1, \ldots, q_i \) and then removing 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 \( A \):

Given \( a^{(i-1)}_i \), normalize to produce \( q_i \) and remove its contribution from

\[
[a^{(i-1)}_{i+1}, \ldots, a^{(i-1)}_k]
\]

to produce

\[
[a^{(i)}_{i+1}, \ldots, a^{(i)}_k]
\]

\( a^{(i)}_j \) 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.

$$\rho_i = \|a_i^{(i-1)}\|_2$$

$$q_i = a_i^{(i-1)}/\rho_i$$

$$r_i^T = q_i^T [a_{i+1}^{(i-1)}, \ldots, a_k^{(i-1)}]$$

$$\begin{bmatrix} a_{i+1}^{(i)} \ldots, a_k^{(i)} \end{bmatrix} = \begin{bmatrix} a_{i+1}^{(i-1)} \ldots, a_k^{(i-1)} \end{bmatrix} - q_i r_i^T$$

The vector $(\rho_i, r_i^T)$ is the $i$th row of $R$ in $A = QR$. The operation count is the same as CGS.
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 $A$.

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:**

\[ a_{3}^{(2)} = (I - q_1 q_1^T - q_2 q_2^T)a_{3}^{(0)} \]

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 \((1)_1\) can be done at the same time or in any order before operations in \((2)_2\) etc.

\[ a_{3}^{(2)} = (3(2a_{3}^{(0)} - q_1 (1q_1^T a_{3}^{(0)}))_1)_2 - q_2(1q_2^T a_{3}^{(0)})_1)_3 \]

**Modified GS:**

\[ a_{3}^{(2)} = (I - q_2 q_2^T)(I - q_1 q_1^T)a_{3}^{(0)} \]

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 there 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 \( \|Q_2^T Q_2 - I\| \) is small enough (if not do it again). Then

\[
A = Q_2 R_2 R_1
= Q_2 (R_2 R_1)
= Q R
\]

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.