Gram-Schmidt Projections

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

\[ q_1 = \frac{P_1 a_1}{\|P_1 a_1\|}, \quad q_2 = \frac{P_2 a_2}{\|P_2 a_2\|}, \quad \ldots, \quad q_n = \frac{P_n a_n}{\|P_n a_n\|} \]

where

\[ P_j = I - \hat{Q}_{j-1} \hat{Q}_{j-1}^* \text{ with } \hat{Q}_{j-1} = \left[ \begin{array}{c|c|c} q_1 & q_2 & \cdots & q_{j-1} \end{array} \right] \]

- \( P_j \) projects orthogonally onto the space orthogonal to \( \langle q_1, \ldots, q_{j-1} \rangle \),

  and \( \text{rank}(P_j) = m - (j - 1) \)
The Modified Gram-Schmidt Algorithm

- The projection \( P_j \) can equivalently be written as
  \[
  P_j = P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1}
  \]
  where (last lecture)
  \[
  P_{\perp q} = I - qq^*
  \]
- \( P_{\perp q} \) projects orthogonally onto the space orthogonal to \( q \), and
  \[
  \text{rank}(P_{\perp q}) = m - 1
  \]
- The Classical Gram-Schmidt algorithm computes an orthogonal vector by
  \[
  v_j = P_j a_j
  \]
  while the Modified Gram-Schmidt algorithm uses
  \[
  v_j = P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1} a_j
  \]

Classical vs. Modified Gram-Schmidt

- Small modification of classical G-S gives modified G-S (but see next slide)
- Modified G-S is numerically stable (less sensitive to rounding errors)

Classical/Modified Gram-Schmidt

\[
\text{for } j = 1 \text{ to } n \\
v_j = a_j \\
\text{for } i = 1 \text{ to } j - 1 \\
\begin{cases} 
  r_{ij} = q_i^* a_j & \text{(CGS)} \\
  r_{ij} = q_i^* v_j & \text{(MGS)} \\
  v_j = v_j - r_{ij} q_i \\
  r_{jj} = \|v_j\|_2 \\
  q_j = v_j / r_{jj}
\end{cases}
\]
Implementation of Modified Gram-Schmidt

- In modified G-S, $P_{\perp q_i}$ can be applied to all $v_j$ as soon as $q_i$ is known
- Makes the inner loop iterations independent (like in classical G-S)

**Classical Gram-Schmidt**

for $j = 1$ to $n$
  $v_j = a_j$
  for $i = 1$ to $j - 1$
    $r_{ij} = q_i^* a_j$
    $v_j = v_j - r_{ij} q_i$
    $r_{jj} = \|v_j\|_2$
  $q_j = v_j / r_{jj}$

**Modified Gram-Schmidt**

for $i = 1$ to $n$
  $v_i = a_i$
  for $i = 1$ to $n$
    $r_{ii} = \|v_i\|$
    $q_i = v_i / r_{ii}$
    for $j = i + 1$ to $n$
      $r_{ij} = q_i^* v_j$
      $v_j = v_j - r_{ij} q_i$

**Example: Classical vs. Modified Gram-Schmidt**

- Compare classical and modified G-S for the vectors

  $a_1 = (1, \epsilon, 0, 0)^T$, $a_2 = (1, 0, \epsilon, 0)^T$, $a_3 = (1, 0, 0, \epsilon)^T$

  making the approximation $1 + \epsilon^2 \approx 1$

- Classical:

  $v_1 \leftarrow (1, \epsilon, 0, 0)^T$, $r_{11} = \sqrt{1 + \epsilon^2} \approx 1$, $q_1 = v_1 / 1 = (1, \epsilon, 0, 0)^T$
  $v_2 \leftarrow (1, 0, \epsilon, 0)^T$, $r_{12} = q_1^T a_2 = 1$, $v_2 \leftarrow v_2 - 1q_1 = (0, -\epsilon, \epsilon, 0)^T$
  $r_{22} = \sqrt{2} \epsilon$, $q_2 = v_2 / r_{22} = (0, -1, 1, 0)^T / \sqrt{2}$
  $v_3 \leftarrow (1, 0, 0, \epsilon)^T$, $r_{13} = q_1^T a_3 = 1$, $v_3 \leftarrow v_3 - 1q_1 = (0, -\epsilon, 0, \epsilon)^T$
  $r_{23} = q_2^T a_3 = 0$, $v_3 \leftarrow v_3 - 0q_2 = (0, -\epsilon, 0, \epsilon)^T$
  $r_{33} = \sqrt{2} \epsilon$, $q_3 = v_3 / r_{33} = (0, -1, 0, 1)^T / \sqrt{2}$
Example: Classical vs. Modified Gram-Schmidt

- Modified:

\[ v_1 \leftarrow (1, \epsilon, 0, 0)^T, \quad r_{11} = \sqrt{1 + \epsilon^2} \approx 1, \quad q_1 = v_1 / 1 = (1, \epsilon, 0, 0)^T \]
\[ v_2 \leftarrow (1, 0, \epsilon, 0)^T, \quad r_{12} = q_1^T v_2 = 1, \quad v_2 \leftarrow v_2 - 1q_1 = (0, -\epsilon, \epsilon, 0)^T \]
\[ r_{22} = \sqrt{2} \epsilon, \quad q_2 = v_2 / r_{22} = (0, -1, 1, 0)^T / \sqrt{2} \]
\[ v_3 \leftarrow (1, 0, 0, \epsilon)^T, \quad r_{13} = q_1^T v_3 = 1, \quad v_3 \leftarrow v_3 - 1q_1 = (0, -\epsilon, 0, \epsilon)^T \]
\[ r_{23} = q_2^T v_3 = \epsilon / \sqrt{2}, \quad v_3 \leftarrow v_3 - r_{23} q_2 = (0, -\epsilon/2, -\epsilon/2, \epsilon)^T \]
\[ r_{33} = \sqrt{6} \epsilon / 2, \quad q_3 = v_3 / r_{33} = (0, -1, -1, 2)^T / \sqrt{6} \]

- Check Orthogonality:
  
  - Classical: \( q_2^T q_3 = (0, -1, 1, 0)(0, -1, 0, 1)^T / 2 = 1/2 \)
  
  - Modified: \( q_2^T q_3 = (0, -1, 1, 0)(0, -1, -1, 2)^T / \sqrt{12} = 0 \)

Operation Count

- Count number of floating points operations – “flops” – in an algorithm

- Each +, −, *, /, or \( \sqrt{\text{ }} \) counts as one flop

- No distinction between real and complex

- No consideration of memory accesses or other performance aspects
Operation Count - Modified G-S

- Example: Count all $+, -, \times, \div$ in the Modified Gram-Schmidt algorithm (not just the leading term)

1. for $i = 1$ to $n$
2. $v_i = a_i$
3. for $i = 1$ to $n$
4. $r_{ii} = \|v_i\|$ \hspace{1cm} $m$ multiplications, $m - 1$ additions
5. $q_i = v_i/r_{ii}$ \hspace{1cm} $m$ divisions
6. for $j = i + 1$ to $n$
7. $r_{ij} = q_i^*v_j$ \hspace{1cm} $m$ multiplications, $m - 1$ additions
8. $v_j = v_j - r_{ij}q_i$ \hspace{1cm} $m$ multiplications, $m$ subtractions

The total for each operation is

\[
\#A = \sum_{i=1}^{n} \left( m - 1 + \sum_{j=i+1}^{n} m - 1 \right) = n(m - 1) + \sum_{i=1}^{n} (m - 1)(n - i) = \\
= n(m - 1) + \frac{n(n-1)(m-1)}{2} = \frac{1}{2}n(n+1)(m-1)
\]

\[
\#S = \sum_{i=1}^{n} \sum_{j=i+1}^{n} m = \sum_{i=1}^{n} m(n-i) = \frac{1}{2}mn(n-1)
\]

\[
\#M = \sum_{i=1}^{n} \left( m + \sum_{j=i+1}^{n} 2m \right) = mn + \sum_{i=1}^{n} 2m(n-i) = \\
= mn + \frac{2mn(n-1)}{2} = mn^2
\]

\[
\#D = \sum_{i=1}^{n} m = mn
\]
and the total flop count is
\[
\frac{1}{2}n(n + 1)(m - 1) + \frac{1}{2}mn(n - 1) + mn^2 + mn = \\
2mn^2 + mn - \frac{1}{2}n^2 - \frac{1}{2}n \sim 2mn^2
\]

- The symbol \( \sim \) indicates asymptotic value as \( m, n \to \infty \) (leading term)

- Easier to find just the leading term:
  - Most work done in lines (7) and (8), with \( 4m \) flops per iteration
  - Including the loops, the total becomes

\[
\sum_{i=1}^{n} \sum_{j=i+1}^{n} 4m = 4m \sum_{i=1}^{n} (n - i) \sim 4m \sum_{i=1}^{n} i = 2mn^2
\]