Numerical Linear Algebra

- Matrix vector product
- QR factorization.
- Gram-Schmidt.
- Modified Gram-Schmidt.

Matrix Vector product

- Let x be a n-dimensional column vector and A be an $m \times n$ matrix (m rows and n columns).

Matrix Vector product

- Let x be a n-dimensional column vector and A be an $m \times n$ matrix (m rows and n columns).
- The matrix vector product $b = Ax$ is the m-dimensional column vector:

$$b_i = \sum_{j=1}^{n} a_{ij} x_j \quad i = 1, \ldots, m.$$

Matrix Vector product

- Let x be a n-dimensional column vector and A be an $m \times n$ matrix (m rows and n columns).
- The matrix vector product $b = Ax$ is the m-dimensional column vector:

$$b_i = \sum_{j=1}^{n} a_{ij} x_j \quad i = 1, \ldots, m.$$

- Let a_j denote the jth column of A, an m-vector. Then rewriting the above equation:

$$b = Ax = \sum_{j=1}^{n} x_j a_j$$

Matrix Vector product

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \begin{bmatrix} & \vert & \vert & & \vert & \\ a_1 & \vert & a_2 & \vert & \dots & \vert & a_n \\ & \vert & \vert & & \vert & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \cdots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

Matrix Vector product

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \begin{bmatrix} \Bigg| & \Bigg| & & \Bigg| \\ a_1 & a_2 & \dots & a_n \\ \Bigg| & \Bigg| & & \Bigg| \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \cdots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

- b can also be thought of as a linear combination of the columns a_j

Matrix Vector product

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \begin{bmatrix} & \vert & \vert & \vert & \\ a_1 & a_2 & \dots & a_n \\ & \vert & \vert & \vert & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \dots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

- b can also be thought of as a linear combination of the columns a_j
- $Ax = b$ is usually thought of as A acting on x to produce b
- $Ax = b$ can also be thought of as x acting on A to produce b!

Matrix Vector product

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \begin{bmatrix} & \vert & & \vert & & \vert & \\ a_1 & \vert & a_2 & \vert & \dots & \vert & a_n \\ & \vert & & \vert & & \vert & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \cdots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

- b can also be thought of as a linear combination of the columns a_j
- $Ax = b$ is usually thought of as A acting on x to produce b
- $Ax = b$ can also be thought of as x acting on A to produce b!
- From $x = A^{-1}b$, x can be thought of just as the result of application of A^{-1} to b.

Matrix Vector product

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \dots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

- b can also be thought of as a linear combination of the columns a_j
- $Ax = b$ is usually thought of as A acting on x to produce b
- $Ax = b$ can also be thought of as x acting on A to produce b!
- From $x = A^{-1}b$, x can be thought of just as the result of application of A^{-1} to b.
- Alternatively, $A^{-1}b$ is the vector of coefficients of the expansion of b in the basis of columns of A.

QR Factorization

- One algorithmic idea in numerical linear algebra that is more important than all others.

- One algorithmic idea in numerical linear algebra that is more important than all others.
- In many applications, we are interested in column spaces of a matrix A. These are *successive* spaces spanned by the columns $a_1, a_2, \ldots$ of A:

$$\langle a_1 \rangle \subseteq \langle a_1, a_2 \rangle \subseteq \langle a_1, a_2, a_3 \rangle \subseteq \ldots$$

  where $\langle \cdots \rangle$ denotes the subspace spanned by whatever vectors are included in the brackets.

QR Factorization

- One algorithmic idea in numerical linear algebra that is more important than all others.
- In many applications, we are interested in column spaces of a matrix A. These are *successive* spaces spanned by the columns $a_1, a_2, \ldots$ of A:

$$\langle a_1 \rangle \subseteq \langle a_1, a_2 \rangle \subseteq \langle a_1, a_2, a_3 \rangle \subseteq \ldots$$

  where $\langle \cdots \rangle$ denotes the subspace spanned by whatever vectors are included in the brackets.
- The idea of QR factorization is to construct a sequence of *orthonormal* vectors, $q_1, q_2, \ldots$ that span these successive spaces.

QR Factorization

- Assume for the moment that $A \in \mathbb{C}^{m \times n}$ $(m \geq n)$, we want the sequence $q_1, q_2, \ldots$ to have the property:

$$\langle q_1, q_2, \ldots, q_j \rangle = \langle a_1, a_2, \ldots, a_j \rangle \quad j = 1, \ldots, n$$

- Assume for the moment that $A \in \mathbb{C}^{m \times n}$ $(m \geq n)$, we want the sequence $q_1, q_2, \ldots$ to have the property:

$$\langle q_1, q_2, \ldots, q_j \rangle = \langle a_1, a_2, \ldots, a_j \rangle \quad j = 1, \ldots, n$$

- This amounts to :

$$\left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \ldots & a_n \\ & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \ldots & q_n \\ & & & \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}$$

QR Factorization

- Assume for the moment that $A \in \mathbb{C}^{m \times n}$ $(m \geq n)$, we want the sequence $q_1, q_2, \ldots$ to have the property:

$$\langle q_1, q_2, \ldots, q_j \rangle = \langle a_1, a_2, \ldots, a_j \rangle \quad j = 1, \ldots, n$$

- This amounts to :

$$\left[\begin{array}{c|c|c|c} a_1 & a_2 & \ldots & a_n \end{array} \right] = \left[\begin{array}{c|c|c|c} q_1 & q_2 & \ldots & q_n \end{array} \right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}$$

- Then $a_1, \ldots, a_k$ can be expressed as a linear combination of $q_1, \ldots q_k$, and vice versa!

QR Factorization

- Written out the equations are:

$$
\begin{aligned}
a_1 &= r_{11}q_1 \\
a_2 &= r_{12}q_1 + r_{22}q_2 \\
&\vdots \\
a_n &= r_{n1}q_1 + r_{2n}q_2 + \cdots + r_{nn}q_n
\end{aligned}
$$

- Written out the equations are:

$$\begin{aligned} a_1 &= r_{11}q_1 \\ a_2 &= r_{12}q_1 + r_{22}q_2 \\ &\vdots \\ a_n &= r_{n1}q_1 + r_{2n}q_2 + \cdots + r_{nn}q_n \end{aligned}$$

- If $A \in \mathbb{R}^{m\times n}$, vectors $q_1, q_2, \ldots, q_n$ are orthonormal m-vectors:

$$q_i^T q_j = \delta_{ij}$$

- Written out the equations are:

$$\begin{aligned} a_1 &= r_{11}q_1 \\ a_2 &= r_{12}q_1 + r_{22}q_2 \\ &\vdots \\ a_n &= r_{n1}q_1 + r_{2n}q_2 + \cdots + r_{nn}q_n \end{aligned}$$

- If $A \in \mathbb{R}^{m \times n}$, vectors $q_1, q_2, \ldots, q_n$ are orthonormal m-vectors:

$$q_i^T q_j = \delta_{ij}$$

- Diagonal elements r_{ii} are non-zero.

QR Factorization

- Written out the equations are:

$$\begin{aligned} a_1 &= r_{11} q_1 \\ a_2 &= r_{12} q_1 + r_{22} q_2 \\ &\vdots \\ a_n &= r_{n1} q_1 + r_{2n} q_2 + \cdots + r_{nn} q_n \end{aligned}$$

- If $A \in \mathbb{R}^{m \times n}$, vectors $q_1, q_2, \ldots, q_n$ are orthonormal m-vectors:

$$q_i^T q_j = \delta_{ij}$$

- Diagonal elements r_{ii} are non-zero.
- If $r_{ii} < 0$, one can switch the signs of $r_{ii}, \ldots, r_{in}$ and the vector q_i.

QR Factorization

- Written out the equations are:

$$
\begin{aligned}
a_1 &= r_{11}q_1 \\
a_2 &= r_{12}q_1 + r_{22}q_2 \\
&\vdots \\
a_n &= r_{n1}q_1 + r_{2n}q_2 + \cdots + r_{nn}q_n
\end{aligned}
$$

- If $A \in \mathbb{R}^{m \times n}$, vectors $q_1, q_2, \ldots, q_n$ are orthonormal m-vectors:

$$q_i^T q_j = \delta_{ij}$$

- Diagonal elements r_{ii} are non-zero.
- If $r_{ii} < 0$, one can switch the signs of $r_{ii}, \ldots, r_{in}$ and the vector q_i.
- Require $r_{ii} > 0$; this makes Q and R unique.

QR Factorization in matrix notation

- If $A \in \mathbb{R}^{m \times n}$ has linearly independent columns then it can be factored as:

$$A = QR$$

QR Factorization in matrix notation

- If $A \in \mathbb{R}^{m \times n}$ has linearly independent columns then it can be factored as:

$$A = QR$$

- Q factor:

QR Factorization in matrix notation

- If $A \in \mathbb{R}^{m \times n}$ has linearly independent columns then it can be factored as:

$$A = QR$$

- Q factor:
  - Q is $m \times n$ with orthonormal columns ($Q^T Q = I$).

QR Factorization in matrix notation

- If $A \in \mathbb{R}^{m \times n}$ has linearly independent columns then it can be factored as:

$$A = QR$$

- Q factor:
  - Q is $m \times n$ with orthonormal columns ($Q^T Q = I$).
  - if A is square ($m = n$), then Q is orthogonal ($Q^T Q = Q Q^T = I$)

QR Factorization in matrix notation

- If $A \in \mathbb{R}^{m \times n}$ has linearly independent columns then it can be factored as:

$$A = QR$$

- Q factor:
  - Q is $m \times n$ with orthonormal columns ($Q^T Q = I$).
  - if A is square ($m = n$), then Q is orthogonal ($Q^T Q = Q Q^T = I$)
- R factor:

QR Factorization in matrix notation

- If $A \in \mathbb{R}^{m \times n}$ has linearly independent columns then it can be factored as:

$$A = QR$$

- Q factor:
  - Q is $m \times n$ with orthonormal columns ($Q^T Q = I$).
  - if A is square ($m = n$), then Q is orthogonal ($Q^T Q = Q Q^T = I$)
- R factor:
  - R is $n \times n$, upper triangular, with nonzero diagonal elements.

QR Factorization in matrix notation

- If $A \in \mathbb{R}^{m \times n}$ has linearly independent columns then it can be factored as:

$$A = QR$$

- Q factor:
  - Q is $m \times n$ with orthonormal columns ($Q^T Q = I$).
  - if A is square ($m = n$), then Q is orthogonal ($Q^T Q = Q Q^T = I$)
- R factor:
  - R is $n \times n$, upper triangular, with nonzero diagonal elements.
  - R is nonsingular (diagonal elements are nonzero)

Example of QR factorization

$$
\begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix} = \begin{bmatrix} -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 \\ -1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix}
$$

$$
= \left[\begin{array}{c|c|c|c} q_1 & q_2 & \cdots & q_n \end{array} \right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}
$$

$$
= QR
$$

Solution of $Ax = b$ using QR factorization

- Compute the QR factorization of A: $A = QR$.

Solution of $Ax = b$ using QR factorization

- Compute the QR factorization of A: $A = QR$.
- Compute $y = Q^T b$

Solution of $Ax = b$ using QR factorization

- Compute the QR factorization of A: $A = QR$.
- Compute $y = Q^T b$
- Solve $Rx = y$ for x : This is just backward substitution as R is upper triangular.

Algorithms for QR factorization

- Classical Gram-Schmidt

Algorithms for QR factorization

- Classical Gram-Schmidt
  - Complexity is $2mn^2$ flops.

Algorithms for QR factorization

- Classical Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Not recommended in practice (sensitive to rounding errors)

Algorithms for QR factorization

- Classical Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Not recommended in practice (sensitive to rounding errors)
- Modified Gram-Schmidt

Algorithms for QR factorization

- Classical Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Not recommended in practice (sensitive to rounding errors)
- Modified Gram-Schmidt
  - Complexity is $2mn^2$ flops.

Algorithms for QR factorization

- Classical Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Not recommended in practice (sensitive to rounding errors)
- Modified Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Better numerical properties.

Algorithms for QR factorization

- Classical Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Not recommended in practice (sensitive to rounding errors)
- Modified Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Better numerical properties.
- Householder

Algorithms for QR factorization

- Classical Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Not recommended in practice (sensitive to rounding errors)
- Modified Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Better numerical properties.
- Householder
  - Complexity is $2mn^2 - (2/3)n^3$ flops.

Algorithms for QR factorization

- Classical Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Not recommended in practice (sensitive to rounding errors)
- Modified Gram-Schmidt
  - Complexity is $2mn^2$ flops.
  - Better numerical properties.
- Householder
  - Complexity is $2mn^2 - (2/3)n^3$ flops.
  - Represent Q as a product of elementary orthogonal algorithms.

Classical Gram Schmidt

- Given $a_1, a_2, \ldots$, we can construct the vectors $q_1, q_2, \ldots,$ and r_{ij} by a process of successive orthogonalization.

Classical Gram Schmidt

- Given $a_1, a_2, \ldots$, we can construct the vectors $q_1, q_2, \ldots,$ and r_{ij} by a process of successive orthogonalization.
- The process works as follows. At the jth step, we wish to find a unit vector $q_j \in \langle a_1, \ldots, a_j \rangle$ that is orthogonal to $q_1, \ldots, q_{j-1}$.

Classical Gram Schmidt

- Given $a_1, a_2, \ldots$, we can construct the vectors $q_1, q_2, \ldots,$ and r_{ij} by a process of successive orthogonalization.
- The process works as follows. At the jth step, we wish to find a unit vector $q_j \in \langle a_1, \ldots, a_j \rangle$ that is orthogonal to $q_1, \ldots, q_{j-1}$.
- $$v_j = a_j - (q_1^* a_j) q_1 \ldots - (q_{j-1}^* a_j) q_{j-1}$$

  is a vector of kind required except that it is not normalized yet.

Classical Gram Schmidt

- Given $a_1, a_2, \ldots$, we can construct the vectors $q_1, q_2, \ldots,$ and r_{ij} by a process of successive orthogonalization.
- The process works as follows. At the jth step, we wish to find a unit vector $q_j \in \langle a_1, \ldots, a_j \rangle$ that is orthogonal to $q_1, \ldots, q_{j-1}$.
- $$v_j = a_j - (q_1^* a_j)q_1 \ldots - (q_{j-1}^* a_j)q_{j-1}$$
  is a vector of kind required except that it is not normalized yet.
- With this in mind:

$$q_1 = \frac{a_1}{r_{11}}$$

$$q_2 = \frac{a_2 - r_{12}q_1}{r_{22}}$$

$$\vdots$$

$$q_n = \frac{a_n - \sum_{i=1}^{n-1} r_{in} q_i}{r_{nn}}$$

Classical Gram-Schmidt

Algorithm 1 Classical Gram-Schmidt (unstable)

```
for j = 1 to n do
    v_j = a_j
    for i = 1 to j − 1 do
        r_ij = q_i^* a_j
        v_j = v_j − r_ij q_i
    end for
    r_jj = |v_j|_2
    q_j = v_j / r_jj
end for
```

for $j = 1$ to n **do**

  $v_j = a_j$

  for $i = 1$ to $j - 1$ **do**

    $r_{ij} = q_i^* a_j$

    $v_j = v_j - r_{ij} q_i$

  end for

  $r_{jj} = |v_j|_2$

  $q_j = v_j / r_{jj}$

end for

Classical Gram-Schmidt

- In exact arithematic, this algorithm is stable.. but on a computer, we always have roundoff errors.

Classical Gram-Schmidt

- In exact arithematic, this algorithm is stable.. but on a computer, we always have roundoff errors.
- Nevertheless – this decomposes $A = QR$.

- In exact arithematic, this algorithm is stable.. but on a computer, we always have roundoff errors.
- Nevertheless – this decomposes $A = QR$.
- Consider:

$$\left[\begin{array}{c|c|c} a_1 & a_2 & a_3 \end{array}\right] = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}$$

$$= \left[\begin{array}{c|c|c|c} q_1 & q_2 & \cdots & q_n \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}$$

Example

- First column of Q and R:

$$q_1 = a_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad r_{11} = |q_1| = 2 \quad q_1 = \frac{1}{r_{ii}} q_1 = \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix}$$

Example

- First column of Q and R:

$$q_1 = a_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad r_{11} = |q_1| = 2 \quad q_1 = \frac{1}{r_{ii}} q_1 = \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix}$$

- Second column of Q and R:

$$r_{12} = q_1^T a_2 = 4$$

Example

- First column of Q and R:

$$q_1 = a_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad r_{11} = |q_1| = 2 \quad q_1 = \frac{1}{r_{ii}} q_1 = \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix}$$

- Second column of Q and R:

$$r_{12} = q_1^T a_2 = 4$$

- Compute:

$$\tilde{q}_2 = a_2 - r_{12} q_1 = \begin{bmatrix} -1 \\ 3 \\ -1 \\ 3 \end{bmatrix} - 4 \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Example

- $r_{22} = |\tilde{q}_2| = 2$

$$q_2 = \frac{1}{r_{22}}\tilde{q}_2 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

Example

- $r_{22} = |\tilde{q}_2| = 2$

$$q_2 = \frac{1}{r_{22}}\tilde{q}_2 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

Example

- $r_{22} = |\tilde{q}_2| = 2$

$$q_2 = \frac{1}{r_{22}} \tilde{q}_2 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

- Compute $r_{13} = q_1^T a_3 = 3$ and $r_{23} = q_2^T a_3 = 8$

Example

- $r_{22} = |\tilde{q}_2| = 2$

$$q_2 = \frac{1}{r_{22}}\tilde{q}_2 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

- Compute $r_{13} = q_1^T a_3 = 3$ and $r_{23} = q_2^T a_3 = 8$
- Compute

$$\tilde{q}_3 = a_3 - r_{13}q_1 - r_{23}q_2 = \begin{bmatrix} -2 \\ -2 \\ 2 \\ 2 \end{bmatrix}$$

Example

- $r_{22} = |\tilde{q}_2| = 2$

$$q_2 = \frac{1}{r_{22}}\tilde{q}_2 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

- Compute $r_{13} = q_1^T a_3 = 3$ and $r_{23} = q_2^T a_3 = 8$
- Compute

$$\tilde{q}_3 = a_3 - r_{13}q_1 - r_{23}q_2 = \begin{bmatrix} -2 \\ -2 \\ 2 \\ 2 \end{bmatrix}$$

- Normalize:

$$r_{33} = |\tilde{q}_3| = 4 \quad q_3 = \frac{1}{r_{33}}\tilde{q}_3 = \begin{bmatrix} -1/2 \\ -1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

Example

$$\begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix} = \begin{bmatrix} -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 \\ -1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} \\ q_1 & q_2 & \dots & q_n \\ \\ \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix}$$

$$= QR$$

Modified Gram-Schmidt

We define:

$$\mathrm{proj}_{\mathbf{u}}\left(\mathbf{v}\right) = \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle}\mathbf{u}$$

- As mentioned Gram-Schmidt is numerically unstable:

$$\mathbf{u}_k = \mathbf{v}_k - \mathrm{proj}_{\mathbf{u}_1}\left(\mathbf{v}_k\right) - \mathrm{proj}_{\mathbf{u}_2}\left(\mathbf{v}_k\right) - \cdots - \mathrm{proj}_{\mathbf{u}_{k-1}}\left(\mathbf{v}_k\right),$$

Modified Gram-Schmidt

We define:

$$\mathrm{proj}_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{v}, \mathbf{u} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle}\mathbf{u}$$

- As mentioned Gram-Schmidt is numerically unstable:

$$\mathbf{u}_k = \mathbf{v}_k - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_k) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_k) - \cdots - \mathrm{proj}_{\mathbf{u}_{k-1}}(\mathbf{v}_k),$$

- However, it can be stabilized with a small modification:

$$\begin{aligned}
\mathbf{u}_k^{(1)} &= \mathbf{v}_k - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_k), \\
\mathbf{u}_k^{(2)} &= \mathbf{u}_k^{(1)} - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{u}_k^{(1)}), \\
&\vdots \\
\mathbf{u}_k^{(k-2)} &= \mathbf{u}_k^{(k-3)} - \mathrm{proj}_{\mathbf{u}_{k-2}}(\mathbf{u}_k^{(k-3)}), \\
\mathbf{u}_k^{(k-1)} &= \mathbf{u}_k^{(k-2)} - \mathrm{proj}_{\mathbf{u}_{k-1}}(\mathbf{u}_k^{(k-2)}).
\end{aligned}$$

Modified Gram-Schmidt

Algorithm 2 Modified Gram-Schmidt

for $i = 1$ to n **do**
    $v_i = a_i$
end for
for $i = 1$ to n **do**
    $r_{ii} = |v_i|_2$
    $q_i = v_i / r_{ii}$
    for $j = i + 1$ to n **do**
        $r_{ij} = q_i^* v_j$
        $v_j = v_j - r_{ij} q_i$
    end for
end for

Modified Gram Schmidt

- Even though on paper, the modified Gram-Schmidt should give identical results as Classical Gram-Schmidt, in practice it is wildly different.

Modified Gram Schmidt

- Even though on paper, the modified Gram-Schmidt should give identical results as Classical Gram-Schmidt, in practice it is wildly different.
- Modified Gram-Schmidt is stable and is routinely used in various software.