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

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \dots & a_n \\ & & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \cdots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

Matrix Vector product

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

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

Matrix Vector product

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \dots & a_n \\ & & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \cdots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

- 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!

Matrix Vector product

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \left[\begin{array}{c|c|c|c} \\ a_1 & a_2 & \dots & a_n \\ \\ \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \cdots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

- 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

$$\begin{bmatrix} \\ b \\ \\ \end{bmatrix} = \left[\begin{array}{c|c|c|c} & & & \\ a_1 & a_2 & \dots & a_n \\ & & & \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \begin{bmatrix} \\ a_1 \\ \\ \end{bmatrix} + x_2 \begin{bmatrix} \\ a_2 \\ \\ \end{bmatrix} + \dots + x_n \begin{bmatrix} \\ a_n \\ \\ \end{bmatrix}$$

- 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.

- 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!

- Written out the equations are:

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

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:

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

- 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 ($Q^T Q = QQ^T = I$)
- 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 ($Q^T Q = QQ^T = I$)
- 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

$$
\begin{aligned}
\begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}
&= \begin{bmatrix} -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 \\ -1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix}
\begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix} \\
&= \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \cdots & q_n \\ & & & \end{array}\right]
\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix} \\
&= QR
\end{aligned}
$$

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.

- 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 $q_1, \ldots, q_{j-1}$.

- 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:

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

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

Example

- First column of Q and R:

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

- 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

$$\begin{aligned}
\begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix} &= \begin{bmatrix} -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 \\ -1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix} \\
&= \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \cdots & q_n \\ & & & \end{array}\right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix} \\
&= QR
\end{aligned}$$

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:

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

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.

- Modified Gram-Schmidt as triangular orthogonalization.
- Householder Triangularization.

Gram-Schmidt

- Consider classical Gram-Schmidt as a sequence of formulas:
$$q_1 = \frac{P_1 a_1}{||P_1 a_1||}, \quad q_2 = \frac{P_2 a_2}{||P_2 a_2||}, \quad \ldots, \quad q_n = \frac{P_n a_n}{||P_n a_n||}$$
where P_j denotes an orthogonal projector

Gram-Schmidt

- Consider classical Gram-Schmidt as a sequence of formulas:

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

  where P_j denotes an orthogonal projector
- This projector can be represented explicitly. Let Q_{j-1} denote the $m \times (j-1)$ matrix containing the first $(j-1)$ columns of Q:

$$Q_{j-1} = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \ldots & q_{j-1} \\ & & & \end{array} \right]$$

Gram-Schmidt

- Consider classical Gram-Schmidt as a sequence of formulas:

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

  where P_j denotes an orthogonal projector
- This projector can be represented explicitly. Let Q_{j-1} denote the $m \times (j-1)$ matrix containing the first $(j-1)$ columns of Q:

$$Q_{j-1} = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & q_2 & \ldots & q_{j-1} \\ & & & \end{array}\right]$$

- Then P_j is given by:

$$P_j = I - Q_{j-1} Q_{j-1}^*$$

Modified Gram-Schmidt

- In contrast, if one thinks of the modified Gram-Schmidt algorithm, then computes the same result by a sequence of of $(j-1)$ projections.

Modified Gram-Schmidt

- In contrast, if one thinks of the modified Gram-Schmidt algorithm, then computes the same result by a sequence of of $(j-1)$ projections.
- If $P_{\perp q} = I - qq*$, then in modified Gram-Schmidt,

$$P_j = P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1}$$

- In contrast, if one thinks of the modified Gram-Schmidt algorithm, then computes the same result by a sequence of of $(j-1)$ projections.
- If $P_{\perp q} = I - qq*$, then in modified Gram-Schmidt,

$$P_j = P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1}$$

- Each outer step of the algorithm can be interpreted as a right multiplication by a square upper-triangular matrix.

- In contrast, if one thinks of the modified Gram-Schmidt algorithm, then computes the same result by a sequence of of $(j-1)$ projections.
- If $P_{\perp q} = I - qq*$, then in modified Gram-Schmidt,

$$P_j = P_{\perp q_{j-1}} \cdots P_{\perp q_2} P_{\perp q_1}$$

- Each outer step of the algorithm can be interpreted as a right multiplication by a square upper-triangular matrix.
- For example, beginning with A, the first iteration multiplies the first column a_1 with $\frac{1}{r_{11}}$ and then subtracts r_{1j} times the result from each of the remaining columns a_j.

Triangular Orthogonalization

- This is equivalent to right-multiplication by a matrix R_1:

$$\left[\begin{array}{c|c|c|c} & & & \\ v_1 & v_2 & \dots & v_n \\ & & & \end{array}\right] \begin{bmatrix} \frac{1}{r_{11}} & \frac{-r_{12}}{r_{11}} & \dots & \frac{-r_{1n}}{r_{11}} \\ & 1 & & \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix} = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & v_2^{(2)} & \dots & v_n^{(2)} \\ & & & \end{array}\right]$$

Triangular Orthogonalization

- This is equivalent to right-multiplication by a matrix R_1:

$$\left[\begin{array}{c|c|c|c} & & & \\ v_1 & v_2 & \dots & v_n \\ & & & \end{array}\right] \begin{bmatrix} \frac{1}{r_{11}} & \frac{-r_{12}}{r_{11}} & \dots & \frac{-r_{1n}}{r_{11}} \\ & 1 & & \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix} = \left[\begin{array}{c|c|c|c} & & & \\ q_1 & v_2^{(2)} & \dots & v_n^{(2)} \\ & & & \end{array}\right]$$

- In general, step i subtracts r_{ij}/r_{ii} times column i of the current A from columns, $j > i$ and replaces column i by $1/r_{ii}$ times itself. This corresponds to multiplication by upper triangular matrix R_i:

$$R_2 = \begin{bmatrix} 1 & & & \\ & \frac{1}{r_{22}} & \frac{-r_{23}}{r_{22}} & \dots \\ & & 1 & \\ & & & \ddots \end{bmatrix} R_3 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \frac{1}{r_{33}} & \dots \\ & & & \ddots \end{bmatrix}, \dots$$

- At the end of the iteration:

$$A \underbrace{R_1 R_2 \cdots R_n}_{R^{-1}} = Q$$

Triangular Orthogonalization

- At the end of the iteration:

$$A \underbrace{R_1 R_2 \cdots R_n}_{R^{-1}} = Q$$

- This shows that Gram-Schmidt is a method of triangular orthogonalization: It applies triangular operations on the right of a matrix to reduce it to a matrix of orthonormal columns.

Householder triangularization

- Householder method applies a succession of elementary unitary matrices Q_k on the left of A so that the resulting matrix:

$$\underbrace{Q_n \cdots Q_2 Q_1}_{Q*} A = R$$

is upper triangular.

- Householder method applies a succession of elementary unitary matrices Q_k on the left of A so that the resulting matrix:

$$\underbrace{Q_n \cdots Q_2 Q_1}_{Q*} A = R$$

is upper triangular.

- The product, $Q = Q_1^* Q_2^* \cdots Q_n^*$ is unitary too and therefore $A = QR$ is the QR factorization of A.

Householder triangularization

- Householder method applies a succession of elementary unitary matrices Q_k on the left of A so that the resulting matrix:

$$\underbrace{Q_n \cdots Q_2 Q_1}_{Q*} A = R$$

  is upper triangular.
- The product, $Q = Q_1^* Q_2^* \cdots Q_n^*$ is unitary too and therefore $A = QR$ is the QR factorization of A.
-
    - Gram-Schmidt: Triangular Orthogonalization

Householder triangularization

- Householder method applies a succession of elementary unitary matrices Q_k on the left of A so that the resulting matrix:

$$\underbrace{Q_n \cdots Q_2 Q_1}_{Q*} A = R$$

  is upper triangular.
- The product, $Q = Q_1^* Q_2^* \cdots Q_n^*$ is unitary too and therefore $A = QR$ is the QR factorization of A.
-
  - Gram-Schmidt: Triangular Orthogonalization
  - Householder : Orthogonal triangularization

Triangularization by introducing zeroes

- The matrix Q_k are chosen such that it introduces zeros below the diagonal in the kth column, while preserving all the zeroes previously introduced.

Triangularization by introducing zeroes

- The matrix Q_k are chosen such that it introduces zeros below the diagonal in the kth column, while preserving all the zeroes previously introduced.
- For example, in the 5×3 case, the zeroes are introduced in the following way:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{Q_1} \begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix} \xrightarrow{Q_2} \begin{bmatrix} \times & \times & \times \\ & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} \end{bmatrix} \xrightarrow{Q_3} \begin{bmatrix} \times & \times & \times \\ & \times & \times \\ & & \boldsymbol{\times} \\ & & \mathbf{0} \\ & & \mathbf{0} \end{bmatrix}$$

Householder Reflectors

- How do we construct the reflectors that will do this?

Householder Reflectors

- How do we construct the reflectors that will do this?
- In general, each Q_k is chosen to be a unitary matrix:

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}$$

where I is a $k \times k$ identity matrix and F is an $(m-k+1) \times (m-k+1)$ unitary matrix.

- How do we construct the reflectors that will do this?
- In general, each Q_k is chosen to be a unitary matrix:

$$Q_k = \begin{bmatrix} I & 0 \\ 0 & F \end{bmatrix}$$

  where I is a $k \times k$ identity matrix and F is an $(m-k+1) \times (m-k+1)$ unitary matrix.
- Multiplication by F should introduce zeroes in the kth column.

- Suppose at the beginning of step k, the entries $k, \ldots, m$ of the kth column are given by vector $x \in \mathbb{C}^{m-k+1}$

Householder reflector

- Suppose at the beginning of step k, the entries $k, \ldots, m$ of the kth column are given by vector $x \in \mathbb{C}^{m-k+1}$
- To introduce zeroes the Householder reflector, F should have the following effect:

$$x = \begin{bmatrix} \times \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \xrightarrow{F} Fx = \begin{bmatrix} ||x|| \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Householder reflector

- Suppose at the beginning of step k, the entries $k, \ldots, m$ of the kth column are given by vector $x \in \mathbb{C}^{m-k+1}$
- To introduce zeroes the Householder reflector, F should have the following effect:

$$x = \begin{bmatrix} \times \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \xrightarrow{F} Fx = \begin{bmatrix} ||x|| \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- The reflector F will reflect the space $\mathbb{C}^{m-k+1}$ across the hyperplane H orthogonal to $v = ||x||e_1 - x$

Householder reflector

- Suppose at the beginning of step k, the entries $k, \ldots, m$ of the kth column are given by vector $x \in \mathbb{C}^{m-k+1}$
- To introduce zeroes the Householder reflector, F should have the following effect:

$$x = \begin{bmatrix} \times \\ \times \\ \times \\ \vdots \\ \times \end{bmatrix} \xrightarrow{F} Fx = \begin{bmatrix} ||x|| \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

- The reflector F will reflect the space $\mathbb{C}^{m-k+1}$ across the hyperplane H orthogonal to $v = ||x||e_1 - x$
- The matrix F is:

$$F = I - 2\frac{vv^*}{v^*v}$$

Householder Reflector

- Given a non-zero p-vector $y = (y_1, y_2, \ldots, y_p)$ define:

$$w = \begin{bmatrix} y_1 + \mathrm{sign}(y_1)||y|| \\ y_2 \\ \vdots \\ y_p \end{bmatrix}, \qquad v = \frac{1}{||w||} w$$

- Given a non-zero p-vector $y = (y_1, y_2, \ldots, y_p)$ define:

$$w = \begin{bmatrix} y_1 + \mathsf{sign}(y_1)||y|| \\ y_2 \\ \vdots \\ y_p \end{bmatrix}, \qquad v = \frac{1}{||w||} w$$

- Vector w satisfies $||w||^2 = 2(w^* y) = 2||y||(||y|| + |y_1|)$

Householder Reflector

- Given a non-zero p-vector $y = (y_1, y_2, \ldots, y_p)$ define:

$$w = \begin{bmatrix} y_1 + \mathsf{sign}(y_1)||y|| \\ y_2 \\ \vdots \\ y_p \end{bmatrix}, \qquad v = \frac{1}{||w||} w$$

- Vector w satisfies $||w||^2 = 2(w^* y) = 2||y||(||y|| + |y_1|)$
- The reflector $F = I - 2vv^*$ maps y to multiple of $e_1 = (1, 0, \ldots, 0)$:

$$Fy = y - \frac{2(w^* y)}{||w||^2} w = y - w = -\mathsf{sign}(y_1)||y||e_1$$

Householder QR factorization

The following algorithm computes the factor R of a QR factorization of a $m \times n$ matrix A ($m \geq n$), leaving the result in place of A. n reflection vectors, $v_1, v_2, \ldots, v_n$ are stored for later use:

Algorithm 1 Householder QR Factorization

```
for k = 1 to n do
    x = A_{k:m,k}
    v_k = sign(x_1)||x||_2 e_1 + x
    v_k = v_k / ||v_k||_2
    A_{k:m,k:n} = A_{k:m,k:n} − 2 v_k (v_k^* A_{k:m,k:n})
end for
```

Applying or forming Q

- Upon completion A has been reduced to upper triangular matrix, but the Q matrix has not been formed.

Applying or forming Q

- Upon completion A has been reduced to upper triangular matrix, but the Q matrix has not been formed.
- This is because explicit construction of Q requires additional work – however we often don't need it explicitly.

Applying or forming Q

- Upon completion A has been reduced to upper triangular matrix, but the Q matrix has not been formed.
- This is because explicit construction of Q requires additional work – however we often don't need it explicitly.
- For solving $Ax = b$, we need to evaluate Q^*b which we do as follows:

Algorithm 6 Implicit calculation of Q^*b

for $k = 1$ to n **do**
    $b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$
end for

Applying or forming Q

- Upon completion A has been reduced to upper triangular matrix, but the Q matrix has not been formed.
- This is because explicit construction of Q requires additional work – however we often don't need it explicitly.
- For solving $Ax = b$, we need to evaluate Q^*b which we do as follows:

Algorithm 8 Implicit calculation of Q^*b

for $k = 1$ to n **do**
  $b_{k:m} = b_{k:m} - 2v_k(v_k^* b_{k:m})$
end for

- Similarly Qx can also be evaluated:

Algorithm 9 Implicit calculation of Qx

for $k = n$ down to 1 **do**
  $x_{k:m} = x_{k:m} - 2v_k(v_k^* x_{k:m})$
end for

Householder triangularization

- Work for Householder orthogonalization $\sim 2mn^2 - \frac{2}{3}n^3$

Householder triangularization

- Work for Householder orthogonalization $\sim 2mn^2 - \frac{2}{3}n^3$
- Work for (modified) Gram-Schmidt: $\sim 2mn^2$

Householder triangularization

- Work for Householder orthogonalization $\sim 2mn^2 - \frac{2}{3}n^3$
- Work for (modified) Gram-Schmidt: $\sim 2mn^2$
- Householder triangularization is numerically more stable than Gram-Schmidt and hence is used for QR factorization.

Example

$$A = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix} = Q_1 Q_2 Q_3 \begin{bmatrix} R \\ 0 \end{bmatrix}$$

Example

$$A = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix} = Q_1 Q_2 Q_3 \begin{bmatrix} R \\ 0 \end{bmatrix}$$

We compute reflectors Q_1, Q_2, Q_3 that trangularize A:

$$Q_3 Q_2 Q_1 A = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \\ 0 & 0 & 0 \end{bmatrix}$$

Example

Compute the reflector that maps first column of A to multiple of e_1

$$y = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix},\ w = y - ||y||e_1 = \begin{bmatrix} -3 \\ 1 \\ -1 \\ 1 \end{bmatrix},\ v_1 = \frac{1}{||w||} w = \frac{1}{2\sqrt{3}} \begin{bmatrix} -3 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

Example

Compute the reflector that maps first column of A to multiple of e_1

$$y = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix},\ w = y - ||y||e_1 = \begin{bmatrix} -3 \\ 1 \\ -1 \\ 1 \end{bmatrix},\ v_1 = \frac{1}{||w||}w = \frac{1}{2\sqrt{3}} \begin{bmatrix} -3 \\ 1 \\ -1 \\ 1 \end{bmatrix}$$

Overwrite A with $I - 2v_1v_1^*$

$$A := (I - 2v_1v_1^*)A = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 4/3 & 8/3 \\ 0 & 2/3 & 16/3 \\ 0 & 4/3 & 20/3 \end{bmatrix}$$

Example

Compute the reflector that maps $A_{2:4,2}$ to multiple of e_1

$$y = \begin{bmatrix} 4/3 \\ 2/3 \\ 4/3 \end{bmatrix},\ w = y + ||y||e_1 = \begin{bmatrix} 10/3 \\ 2/3 \\ 4/3 \end{bmatrix},\ v_2 = \frac{1}{||w||} w = \frac{1}{\sqrt{30}} \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}$$

Example

Compute the reflector that maps $A_{2:4,2}$ to multiple of e_1

$$y = \begin{bmatrix} 4/3 \\ 2/3 \\ 4/3 \end{bmatrix}, \; w = y + ||y||e_1 = \begin{bmatrix} 10/3 \\ 2/3 \\ 4/3 \end{bmatrix}, \; v_2 = \frac{1}{||w||} w = \frac{1}{\sqrt{30}} \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}$$

Overwrite $A_{2:4,2:3}$ with $I - 2v_2v_2^*$

$$A := \begin{bmatrix} 1 & 0 \\ 0 & I - 2v_2v_2^* \end{bmatrix} A = \begin{bmatrix} 2 & 4 & 2 \\ 0 & -2 & -8 \\ 0 & 0 & 16/5 \\ 0 & 0 & 12/5 \end{bmatrix}$$

Example

Compute the reflector that maps $A_{3:4,3}$ to multiple of e_1

$$y = \begin{bmatrix} 16/5 \\ 12/5 \end{bmatrix}, \; w = y + ||y||e_1 = \begin{bmatrix} 36/5 \\ 12/5 \end{bmatrix}, \; v_3 = \frac{1}{||w||} w = \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Example

Compute the reflector that maps $A_{3:4,3}$ to multiple of e_1

$$y = \begin{bmatrix} 16/5 \\ 12/5 \end{bmatrix}, \; w = y + ||y||e_1 = \begin{bmatrix} 36/5 \\ 12/5 \end{bmatrix}, \; v_3 = \frac{1}{||w||} w = \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

Overwrite $A_{3:4,3}$ with $I - 2v_3v_3^*$

$$A := \begin{bmatrix} I & 0 \\ 0 & I - 2v_3v_3^* \end{bmatrix} A = \begin{bmatrix} 2 & 4 & 2 \\ 0 & -2 & -8 \\ 0 & 0 & -4 \\ 0 & 0 & 0 \end{bmatrix}$$