6. QR factorization

- triangular matrices
- QR factorization
- Gram–Schmidt algorithm
- modified Gram–Schmidt algorithm
- QR factorization with column pivoting

Triangular matrix

a square matrix A is **lower triangular** if $A_{ij} = 0$ for $j > i$

$$A = \begin{bmatrix} A_{11} & 0 & \cdots & 0 & 0 \\ A_{21} & A_{22} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ A_{n-1,1} & A_{n-1,2} & \cdots & A_{n-1,n-1} & 0 \\ A_{n1} & A_{n2} & \cdots & A_{n,n-1} & A_{nn} \end{bmatrix}$$

A is **upper triangular** if $A_{ij} = 0$ for $j < i$ (the transpose A^T is lower triangular)

a triangular matrix is **unit** upper/lower triangular if $A_{ii} = 1$ for all i

Forward substitution

solve $Ax = b$ when A is lower triangular with nonzero diagonal elements

Algorithm

$$
\begin{aligned}
x_1 &= b_1/A_{11} \\
x_2 &= (b_2 - A_{21}x_1)/A_{22} \\
x_3 &= (b_3 - A_{31}x_1 - A_{32}x_2)/A_{33} \\
&\vdots \\
x_n &= (b_n - A_{n1}x_1 - A_{n2}x_2 - \cdots - A_{n,n-1}x_{n-1})/A_{nn}
\end{aligned}
$$

Complexity: $1 + 3 + 5 + \cdots + (2n-1) = n^2$ flops

Back substitution

solve $Ax = b$ when A is upper triangular with nonzero diagonal elements

Algorithm

$$\begin{aligned}
x_n &= b_n / A_{nn} \\
x_{n-1} &= (b_{n-1} - A_{n-1,n} x_n) / A_{n-1,n-1} \\
x_{n-2} &= (b_{n-2} - A_{n-2,n-1} x_{n-1} - A_{n-2,n} x_n) / A_{n-2,n-2} \\
&\vdots \\
x_1 &= (b_1 - A_{12} x_2 - A_{13} x_3 - \cdots - A_{1n} x_n) / A_{11}
\end{aligned}$$

Complexity: n^2 flops

Inverse of triangular matrix

a triangular matrix A with nonzero diagonal elements is nonsingular:

$$Ax = 0 \quad \Longrightarrow \quad x = 0$$

this follows from forward or back substitution applied to the equation $Ax = 0$

- inverse of A can be computed by solving $AX = I$ column by column

$$A \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} = \begin{bmatrix} e_1 & e_2 & \cdots & e_n \end{bmatrix} \qquad (x_i \text{ is column } i \text{ of } X)$$

- inverse of lower triangular matrix is lower triangular
- inverse of upper triangular matrix is upper triangular
- complexity of computing inverse of $n \times n$ triangular matrix is

$$n^2 + (n-1)^2 + \cdots + 1 \approx \frac{1}{3}n^3 \text{ flops}$$

Outline

- triangular matrices
- **QR factorization**
- Gram–Schmidt algorithm
- modified Gram–Schmidt algorithm
- QR factorization with column pivoting

QR factorization

if $A \in \mathbf{R}^{m \times n}$ has linearly independent columns then it can be factored as

$$A = \left[\begin{array}{cccc} q_1 & q_2 & \cdots & q_n \end{array}\right] \left[\begin{array}{cccc} R_{11} & R_{12} & \cdots & R_{1n} \\ 0 & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{nn} \end{array}\right]$$

- vectors $q_1, \ldots, q_n$ are orthonormal m-vectors:

$$\|q_i\| = 1, \qquad q_i^T q_j = 0 \quad \text{if } i \neq j$$

- diagonal elements R_{ii} are nonzero
- if $R_{ii} < 0$, we can switch the signs of $R_{ii}, \ldots, R_{in}$, and the vector q_i
- most definitions require $R_{ii} > 0$; this makes Q and R unique

QR factorization in matrix notation

if $A \in \mathbf{R}^{m \times n}$ 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
- R is nonsingular (diagonal elements are nonzero)

Example

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

$$
= \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}
\begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}
$$

$$
= QR
$$

Full QR factorization

the QR factorization is often defined as a factorization

$$A = \left[\begin{array}{cc} Q & \tilde{Q} \end{array} \right] \left[\begin{array}{c} R \\ 0 \end{array} \right]$$

- $A = QR$ is the QR factorization as defined earlier (page 6.7)
- $\tilde{Q}$ has size $m \times (m-n)$, the zero block has size $(m-n) \times n$
- the matrix $\left[\begin{array}{cc} Q & \tilde{Q} \end{array} \right]$ is $m \times m$ and orthogonal
- MATLAB's function `qr` returns this factorization
- this is also known as the *full QR factorization* or *QR decomposition*

in this course we use the definition of page 6.7

Applications

in the following lectures, we will use the QR factorization to solve

- linear equations
- least squares problems
- constrained least squares problems

here, we show that it gives useful simple formulas for

- the pseudo-inverse of a matrix with linearly independent columns
- the inverse of a nonsingular matrix
- projection on the range of a matrix with linearly independent columns

QR factorization and (pseudo-)inverse

pseudo-inverse of a matrix A with linearly independent columns (page 4.22)

$$A^\dagger = (A^T A)^{-1} A^T$$

- pseudo-inverse in terms of QR factors of A:

$$\begin{aligned}
A^\dagger &= ((QR)^T (QR))^{-1} (QR)^T \\
&= (R^T Q^T Q R)^{-1} R^T Q^T \\
&= (R^T R)^{-1} R^T Q^T && (Q^T Q = I) \\
&= R^{-1} R^{-T} R^T Q^T && (R \text{ is nonsingular}) \\
&= R^{-1} Q^T
\end{aligned}$$

- for square nonsingular A this is the inverse:

$$A^{-1} = (QR)^{-1} = R^{-1} Q^T$$

Range

recall definition of range of a matrix $A \in \mathbf{R}^{m\times n}$ (page 4.27):

$$\text{range}(A) = \{Ax \mid x \in \mathbf{R}^n\}$$

suppose A has linearly independent columns with QR factors Q, R

- Q has the same range as A:

$$\begin{aligned}
y \in \text{range}(A) \quad &\Longleftrightarrow \quad y = Ax \ \text{ for some } x \\
&\Longleftrightarrow \quad y = QRx \ \text{ for some } x \\
&\Longleftrightarrow \quad y = Qz \ \text{ for some } z \\
&\Longleftrightarrow \quad y \in \text{range}(Q)
\end{aligned}$$

- columns of Q are an orthonormal basis for $\text{range}(A)$

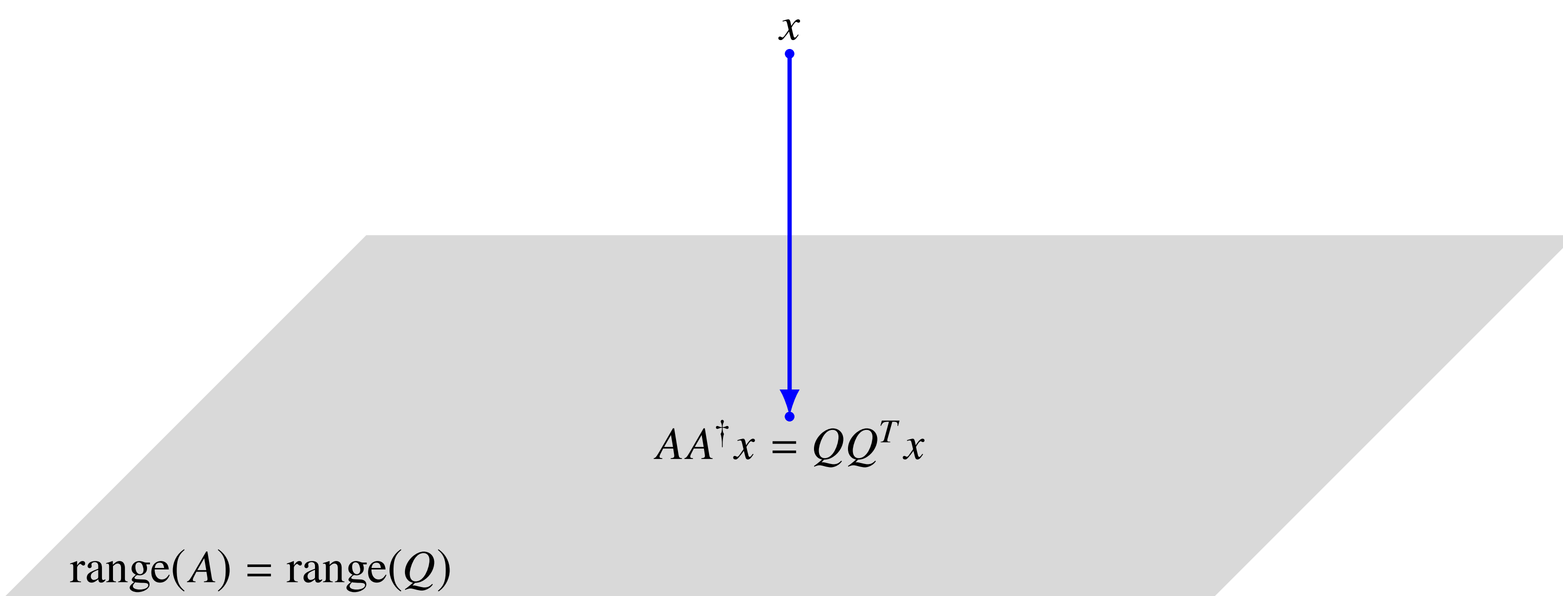
Projection on range

- combining $A = QR$ and $A^\dagger = R^{-1}Q^T$ (from page 6.11) gives

$$AA^\dagger = QRR^{-1}Q^T = QQ^T$$

note the order of the product in $AA^\dagger$ and the difference with $A^\dagger A = I$

- recall (from page 5.16) that QQ^Tx is the projection of x on the range of Q



QR factorization of complex matrices

if $A \in \mathbf{C}^{m\times n}$ has linearly independent columns then it can be factored as

$$A = QR$$

- $Q \in \mathbf{C}^{m\times n}$ has orthonormal columns ($Q^H Q = I$)
- $R \in \mathbf{C}^{n\times n}$ is upper triangular with real nonzero diagonal elements
- most definitions choose diagonal elements R_{ii} to be positive
- in the rest of the lecture we assume A is real

Algorithms for QR factorization

Gram–Schmidt algorithm (section 5.4 in textbook and page 6.16)

- complexity is $2mn^2$ flops
- not recommended in practice (sensitive to rounding errors)

Modified Gram–Schmidt algorithm

- complexity is $2mn^2$ flops
- better numerical properties

Householder algorithm

- complexity is $2mn^2 - (2/3)n^3$ flops
- represents Q as a product of elementary orthogonal matrices
- the most widely used algorithm (used by the function `qr` in MATLAB and Julia)

in the rest of the course we will take $2mn^2$ for the complexity of QR factorization

Outline

- triangular matrices
- QR factorization
- **Gram–Schmidt algorithm**
- modified Gram–Schmidt algorithm
- QR factorization with column pivoting

Gram–Schmidt algorithm

Gram–Schmidt QR algorithm computes Q and R column by column

- after k steps we have a partial QR factorization

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_k \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_k \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1k} \\ 0 & R_{22} & \cdots & R_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{kk} \end{bmatrix}$$

  this is the QR factorization for the first k columns of A

- columns $q_1, \ldots, q_k$ are orthonormal
- diagonal elements $R_{11}, R_{22}, \ldots, R_{kk}$ are positive
- columns $q_1, \ldots, q_k$ have the same span as $a_1, \ldots, a_k$ (see page 6.12)
- in step k of the algorithm we compute $q_k, R_{1k}, \ldots, R_{kk}$

Computing the kth columns of Q and R

suppose we have the partial factorization for the first $k-1$ columns of Q and R

- column k of the equation $A = QR$ reads

$$a_k = R_{1k}q_1 + R_{2k}q_2 + \cdots + R_{k-1,k}q_{k-1} + R_{kk}q_k$$

- regardless of how we choose $R_{1k}, \ldots, R_{k-1,k}$, the vector

$$\tilde{q}_k = a_k - R_{1k}q_1 - R_{2k}q_2 - \cdots - R_{k-1,k}q_{k-1}$$

  will be nonzero: $a_1, a_2, \ldots, a_k$ are linearly independent and therefore

$$a_k \notin \operatorname{span}\{a_1, \ldots, a_{k-1}\} = \operatorname{span}\{q_1, \ldots, q_{k-1}\}$$

- q_k is $\tilde{q}_k$ normalized: choose $R_{kk} = \|\tilde{q}_k\|$ and $q_k = (1/R_{kk})\tilde{q}_k$
- $\tilde{q}_k$ and q_k are orthogonal to $q_1, \ldots, q_{k-1}$ if we choose $R_{1k}, \ldots, R_{k-1,k}$ as

$$R_{1k} = q_1^T a_k, \qquad R_{2k} = q_2^T a_k, \qquad \ldots, \qquad R_{k-1,k} = q_{k-1}^T a_k$$

Gram–Schmidt algorithm

Given: $m \times n$ matrix A with linearly independent columns $a_1, \ldots, a_n$

Algorithm

for $k = 1$ to n

$$\begin{aligned}
R_{1k} &= q_1^T a_k \\
R_{2k} &= q_2^T a_k \\
&\vdots \\
R_{k-1,k} &= q_{k-1}^T a_k \\
\tilde{q}_k &= a_k - (R_{1k} q_1 + R_{2k} q_2 + \cdots + R_{k-1,k} q_{k-1}) \\
R_{kk} &= \|\tilde{q}_k\| \\
q_k &= \frac{1}{R_{kk}} \tilde{q}_k
\end{aligned}$$

Example

example on page 6.8:

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}$$

$$= \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

First column of Q and R

$$\tilde{q}_1 = a_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \qquad R_{11} = \|\tilde{q}_1\| = 2, \qquad q_1 = \frac{1}{R_{11}}\tilde{q}_1 = \begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix}$$

Example

Second column of Q and R

- compute $R_{12} = q_1^T a_2 = 4$
- compute

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

- normalize to get

$$R_{22} = \|\tilde{q}_2\| = 2, \qquad q_2 = \frac{1}{R_{22}} \tilde{q}_2 = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$$

Example

Third column of Q and R

- compute $R_{13} = q_1^T a_3 = 2$ and $R_{23} = q_2^T a_3 = 8$
- compute

$$\tilde{q}_3 = a_3 - R_{13}q_1 - R_{23}q_2 = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \end{bmatrix} - 2\begin{bmatrix} -1/2 \\ 1/2 \\ -1/2 \\ 1/2 \end{bmatrix} - 8\begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ 2 \\ 2 \end{bmatrix}$$

- normalize to get

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

Example

Final result

$$
\begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}
= \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}
$$

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

Complexity

Complexity of cycle k (of algorithm on page 6.18)

- $k-1$ inner products with a_k: $(k-1)(2m-1)$ flops
- computation of $\tilde{q}_k$: $2(k-1)m$ flops
- computing R_{kk} and q_k: $3m$ flops

total for cycle k: $(4m-1)(k-1)+3m$ flops

Complexity for $m \times n$ factorization:

$$\begin{aligned} \sum_{k=1}^{n}((4m-1)(k-1)+3m) &= (4m-1)\frac{n(n-1)}{2}+3mn \\ &\approx 2mn^2 \text{ flops} \end{aligned}$$

Numerical experiment

- we use the following MATLAB implementation of the algorithm on page 6.18:

```
[m, n] = size(A);
Q = zeros(m,n);
R = zeros(n,n);
for k = 1:n
    R(1:k-1,k) = Q(:,1:k-1)' * A(:,k);
    v = A(:,k) - Q(:,1:k-1) * R(1:k-1,k);
    R(k,k) = norm(v);
    Q(:,k) = v / R(k,k);
end;
```

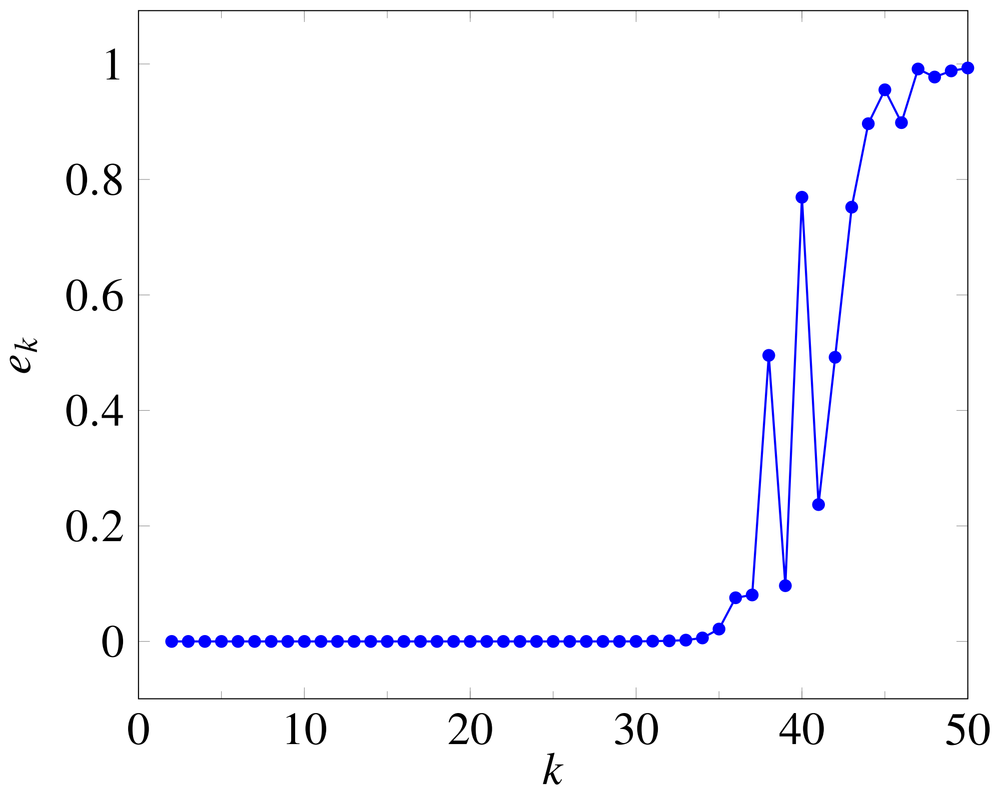
- we apply this to a square matrix A of size $m = n = 50$
- A is constructed as $A = USV$ with U, V orthogonal, S diagonal with

$$S_{ii} = 10^{-10(i-1)/(n-1)}, \quad i = 1, \ldots, n$$

Numerical experiment

plot shows deviation from orthogonality between q_k and previous columns

$$e_k = \max_{1 \le i < k} |q_i^T q_k|, \quad k = 2, \ldots, n$$



loss of orthogonality is due to rounding error

Outline

- triangular matrices
- QR factorization
- Gram–Schmidt algorithm
- **modified Gram–Schmidt algorithm**
- QR factorization with column pivoting

Modified Gram–Schmidt algorithm

a variation of the Gram–Schmidt algorithm for the QR factorization

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ 0 & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{nn} \end{bmatrix}$$

- has better numerical properties than the Gram–Schmidt algorithm
- computes Q column by column, R row by row
- main difference lies in how the vectors

$$\tilde{q}_k = R_{kk}q_k = a_k - (R_{1k}q_1 + \cdots + R_{k-1,k}q_{k-1})$$

are computed

Modified Gram–Schmidt algorithm

after k steps, the algorithm has computed a partial factorization

$$
\begin{aligned}
A &= \left[\begin{array}{c|c} a_1 \cdots a_k & a_{k+1} \cdots a_n \end{array}\right] \\
&= \left[\begin{array}{c|c} q_1 \cdots q_k & B^{(k)} \end{array}\right]
\left[\begin{array}{ccc|ccc}
R_{11} & \cdots & R_{1k} & R_{1,k+1} & \cdots & R_{1n} \\
\vdots & \ddots & \vdots & \vdots & & \vdots \\
0 & \cdots & R_{kk} & R_{k,k+1} & \cdots & R_{kn} \\
\hline
 & 0 & & & I &
\end{array}\right]
\end{aligned}
$$

- $B^{(k)}$ has size $m \times (n-k)$ with columns orthogonal to $q_1, \ldots, q_k$
- we start with $k = 0$ and $B^{(0)} = A$
- the factorization is complete when $k = n$
- in step k, we compute

$$q_k, \quad R_{kk}, \quad R_{k,k+1}, \quad \ldots, \quad R_{kn}, \quad B^{(k)}$$

Modified Gram–Schmidt update

careful inspection of the update at step k shows that

$$B^{(k-1)} = \begin{bmatrix} q_k & B^{(k)} \end{bmatrix} \begin{bmatrix} R_{kk} & R_{k,(k+1):n} \\ 0 & I \end{bmatrix}$$

partition $B^{(k-1)}$ as $B^{(k-1)} = \begin{bmatrix} b & \hat{B} \end{bmatrix}$ with b the first column, $\hat{B}$ of size $m \times (n-k)$:

$$b = q_k R_{kk}, \qquad \hat{B} = q_k R_{k,(k+1):n} + B^{(k)}$$

- from the first equation, and the required properties $\|q_k\| = 1$ and $R_{kk} > 0$:

$$R_{kk} = \|b\|, \qquad q_k = \frac{1}{R_{kk}} b$$

- from the second equation, and the requirement that $q_k^T B^{(k)} = 0$:

$$R_{k,(k+1):n} = q_k^T \hat{B}, \qquad B^{(k)} = \hat{B} - q_k R_{k,(k+1):n}$$

Summary: modified Gram–Schmidt algorithm

Algorithm (A is $m \times n$ with linearly independent columns)

define $B^{(0)} = A$; for $k = 1$ to n,

- compute $R_{kk} = \|b\|$ and $q_k = (1/R_{kk})b$ where b is the first column of $B^{(k-1)}$
- compute

$$\begin{bmatrix} R_{k,k+1} & \cdots & R_{kn} \end{bmatrix} = q_k^T \hat{B}, \qquad B^{(k)} = \hat{B} - q_k \begin{bmatrix} R_{k,k+1} & \cdots & R_{kn} \end{bmatrix}$$

  where $\hat{B}$ is $B^{(k-1)}$ with first column removed

MATLAB code (`Q(:,k+1:n)` is used to store $B^{(k)}$)

```
Q = A;  R = zeros(n,n);
for k = 1:n
    R(k,k) = norm(Q(:,k));
    Q(:,k) = Q(:,k) / R(k,k);
    R(k,k+1:n) = Q(:,k)' * Q(:,k+1:n);
    Q(:,k+1:n) = Q(:,k+1:n)  - Q(:,k) * R(k,k+1:n);
end;
```

Example

example on page 6.8

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}$$

Step 1: first column of Q, first row of R

$$\begin{aligned} \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} &= \left[\begin{array}{r|rr} -1/2 & 1 & 2 \\ 1/2 & 1 & 2 \\ -1/2 & 1 & 6 \\ 1/2 & 1 & 6 \end{array}\right] \left[\begin{array}{c|cc} 2 & 4 & 2 \\ \hline 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{c|c} q_1 & B^{(1)} \end{array}\right] \left[\begin{array}{c|c} R_{11} & R_{1,2:3} \\ \hline 0 & I \end{array}\right] \end{aligned}$$

Example

Step 2: second column of Q, second row of R

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} = \left[\begin{array}{cc|c} -1/2 & 1/2 & -2 \\ 1/2 & 1/2 & -2 \\ -1/2 & 1/2 & 2 \\ 1/2 & 1/2 & 2 \end{array}\right] \left[\begin{array}{cc|c} 2 & 4 & 2 \\ 0 & 2 & 8 \\ \hline 0 & 0 & 1 \end{array}\right]$$

$$= \left[\begin{array}{cc|c} q_1 & q_2 & B^{(2)} \end{array}\right] \left[\begin{array}{cc|c} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ \hline 0 & 0 & 1 \end{array}\right]$$

Step 3: third column of Q, third row of R

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} = \begin{bmatrix} -1/2 & 1/2 & -1/2 \\ 1/2 & 1/2 & -1/2 \\ -1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{bmatrix}$$

Complexity

Complexity of cycle k (of algorithm on page 6.29)

- computing R_{kk} and q_k: $3m$ flops
- computing $R_{k,k+1}, \ldots, R_{kn}$: $(n-k)(2m-1)$ flops
- computing $B^{(k)}$: $2(n-k)m$ flops

total for cycle k: $(4m-1)(n-k)+3m$ flops

Complexity for $m \times n$ factorization:

$$\begin{aligned}\sum_{k=1}^{n}((4m-1)(n-k)+3m) &= (4m-1)\frac{n(n-1)}{2}+3mn \\ &\approx 2mn^2 \text{ flops}\end{aligned}$$

Exercise

we use the notation of page 6.29

1. show that

$$B^{(k)} = \begin{bmatrix} b & \hat{B} \end{bmatrix} \begin{bmatrix} y^T \\ I \end{bmatrix} \quad \text{where } y = -\frac{1}{R_{kk}} \begin{bmatrix} R_{k,k+1} \\ \vdots \\ R_{kn} \end{bmatrix}$$

2. use this to show that $B^{(k)}$ has linearly independent columns if

$$B^{(k-1)} = \begin{bmatrix} b & \hat{B} \end{bmatrix}$$

   has linearly independent columns

since $B^{(0)} = A$, this proves that $B^{(k)}$ has linearly independent columns for all k

Exercise

in the notation of page 6.29,

$$\begin{bmatrix} R_{k,k+1} & \cdots & R_{kn} \end{bmatrix} = q_k^T \hat{B}, \qquad B^{(k)} = (I - q_k q_k^T)\hat{B}$$

- denote column i of $\hat{B}$ by $\hat{b}_i$:

$$B^{(k-1)} = \begin{bmatrix} b & \hat{B} \end{bmatrix} = \begin{bmatrix} b & \hat{b}_1 & \cdots & \hat{b}_{n-k} \end{bmatrix}$$

- denote column i of $B^{(k)}$ by $b_i^{(k)}$

$$B^{(k)} = \begin{bmatrix} b_1^{(k)} & \cdots & b_{n-k}^{(k)} \end{bmatrix}$$

show that

$$\|b_i^{(k)}\|^2 = \|\hat{b}_i\|^2 - R_{k,k+i}^2$$

Exercise

in the Gram–Schmidt algorithm (page 6.18) the vector $\tilde{q}_k = R_{kk}q_k$ is computed as

$$\begin{aligned} \tilde{q}_k &= a_k - q_1(q_1^T a_k) - q_2(q_2^T a_k) - \cdots - q_{k-1}(q_{k-1}^T a_k) \\ &= \left(I - q_1 q_1^T - q_2 q_2^T - \cdots - q_{k-1} q_{k-1}^T\right) a_k \end{aligned}$$

- show that

$$\begin{aligned} & I - q_1 q_1^T - q_2 q_2^T - \cdots - q_{k-1} q_{k-1}^T \\ & \quad = \quad (I - q_{k-1} q_{k-1}^T)(I - q_{k-2} q_{k-2}^T) \cdots (I - q_2 q_2^T)(I - q_1 q_1^T) \end{aligned}$$

- in the modified Gram–Schmidt algorithm (page 6.29) $\tilde{q}_k$ is column 1 of $B^{(k-1)}$

  verify that the modified GS algorithm obtains $\tilde{q}_k$ by evaluating

$$\tilde{q}_k = (I - q_{k-1} q_{k-1}^T)(I - q_{k-2} q_{k-2}^T) \cdots (I - q_2 q_2^T)(I - q_1 q_1^T) a_k$$

  from right to left

Outline

- triangular matrices
- QR factorization
- Gram–Schmidt algorithm
- modified Gram–Schmidt algorithm
- **QR factorization with column pivoting**

QR factorization with column pivoting

A is an $m \times n$ matrix with rank r (may be wide or have linearly dependent columns)

QR factorization with column pivoting (column reordering)

$$A = QRP$$

- Q is $m \times r$ with orthonormal columns
- R is $r \times n$, leading $r \times r$ submatrix is upper triangular with positive diagonal:

$$R = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1r} & R_{1,r+1} & \cdots & R_{1n} \\ 0 & R_{22} & \cdots & R_{2r} & R_{2,r+1} & \cdots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & R_{rr} & R_{r,r+1} & \cdots & R_{rn} \end{bmatrix}$$

- can be chosen to satisfy $R_{11} \geq R_{22} \geq \cdots \geq R_{rr} > 0$
- P is an $n \times n$ permutation matrix
- this is a full-rank factorization (page 4.34) with $B = Q$, $C = RP$

Interpretation

- columns of $AP^T = QR$ are the columns of A in a different order
- the columns are divided in two groups:

$$AP^T = \begin{bmatrix} \hat{A}_1 & \hat{A}_2 \end{bmatrix} = Q \begin{bmatrix} R_1 & R_2 \end{bmatrix} \qquad \hat{A}_1 \text{ is } m \times r,\ R_1 \text{ is } r \times r$$

- $\hat{A}_1$ is $m \times r$ with linearly independent columns and QR factorization $\hat{A}_1 = QR_1$
- $\hat{A}_2$ is $m \times (n-r)$ with columns that are linear combinations of columns of $\hat{A}_1$:

$$\hat{A}_2 = QR_2 = \hat{A}_1 R_1^{-1} R_2$$

the QR factorization with column pivoting provides two useful bases for $\mathrm{range}(A)$

- columns of Q are an orthonormal basis
- columns of $\hat{A}_1$ are a basis selected from the columns of A

Modified Gram–Schmidt algorithm with pivoting

with minor changes the modified GS algorithm computes the pivoted factorization

$$AP^T = \left[\begin{array}{cccc} q_1 & q_2 & \cdots & q_r \end{array}\right] \left[\begin{array}{ccccccc} R_{11} & R_{12} & \cdots & R_{1r} & R_{1,r+1} & \cdots & R_{1n} \\ 0 & R_{22} & \cdots & R_{1r} & R_{1,r+1} & \cdots & R_{1n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & R_{rr} & R_{r,r+1} & \cdots & R_{rn} \end{array}\right]$$

- partial factorization after k steps

$$AP_k^T = \left[\begin{array}{c|c} q_1 \cdots q_k & B^{(k)} \end{array}\right] \left[\begin{array}{ccc|ccc} R_{11} & \cdots & R_{1k} & R_{1,k+1} & \cdots & R_{1n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & R_{kk} & R_{k,k+1} & \cdots & R_{kn} \\ \hline & 0 & & & I & \end{array}\right]$$

- algorithm starts with $P_0 = I$ and $B^{(0)} = A$
- if $B^{(k)} = 0$, the factorization is complete ($r = k$, $P = P_k$)
- before step k, we reorder columns of $B^{(k-1)}$ to place its largest column first
- this requires reordering columns $k, \ldots, n$ of R, and modifying P_{k-1}

Example

$$A = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & -1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & -1 & -1 \end{bmatrix}$$

Step 1

- a_2 and a_4 have the largest norms; we move a_2 to the first position
- find first column of Q, first row of R

$$\begin{aligned} \begin{bmatrix} a_2 & a_1 & a_3 & a_4 \end{bmatrix} &= \left[\begin{array}{c|ccc} 1/2 & 1/2 & 0 & 1 \\ 1/2 & -1/2 & 1 & -1 \\ 1/2 & 1/2 & 0 & 1 \\ 1/2 & -1/2 & -1 & -1 \end{array}\right] \left[\begin{array}{c|ccc} 2 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{c|c} q_1 & B^{(1)} \end{array}\right] \left[\begin{array}{c|c} R_{11} & R_{1,2:4} \\ \hline 0 & I \end{array}\right] \end{aligned}$$

Example

Step 2

- move column 3 of $B^{(1)}$ to first position in $B^{(1)}$

$$
\left[\begin{array}{cccc} a_2 & a_4 & a_1 & a_3 \end{array}\right] = \left[\begin{array}{c|ccc} 1/2 & 1 & 1/2 & 0 \\ 1/2 & -1 & -1/2 & 1 \\ 1/2 & 1 & 1/2 & 0 \\ 1/2 & -1 & -1/2 & -1 \end{array}\right] \left[\begin{array}{c|ccc} 2 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]
$$

- find second column of Q, second row or R

$$
\begin{aligned}
\left[\begin{array}{cccc} a_2 & a_4 & a_1 & a_3 \end{array}\right] &= \left[\begin{array}{cc|cc} 1/2 & 1/2 & 0 & 0 \\ 1/2 & -1/2 & 0 & 1 \\ 1/2 & 1/2 & 0 & 0 \\ 1/2 & -1/2 & 0 & -1 \end{array}\right] \left[\begin{array}{cc|cc} 2 & 0 & 1 & 0 \\ 0 & 2 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right] \\
&= \left[\begin{array}{cc|c} q_1 & q_2 & B^{(2)} \end{array}\right] \left[\begin{array}{cc|c} R_{11} & R_{12} & R_{1,3:4} \\ 0 & R_{22} & R_{2,3:4} \\ \hline 0 & 0 & I \end{array}\right]
\end{aligned}
$$

Example

Step 3

- move column 2 of $B^{(2)}$ to first position in $B^{(2)}$

$$\left[\begin{array}{cccc} a_2 & a_4 & a_3 & a_1 \end{array}\right] = \left[\begin{array}{cc|cc} 1/2 & 1/2 & 0 & 0 \\ 1/2 & -1/2 & 1 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1/2 & -1/2 & -1 & 0 \end{array}\right] \left[\begin{array}{cc|cc} 2 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ \hline 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

- find third column of Q, third row of R

$$\begin{aligned} \left[\begin{array}{cccc} a_2 & a_4 & a_3 & a_1 \end{array}\right] &= \left[\begin{array}{ccc|c} 1/2 & 1/2 & 0 & 0 \\ 1/2 & -1/2 & 1/\sqrt{2} & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1/2 & -1/2 & -1/\sqrt{2} & 0 \end{array}\right] \left[\begin{array}{ccc|c} 2 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & \sqrt{2} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{ccc|c} q_1 & q_2 & q_3 & B^{(3)} \end{array}\right] \left[\begin{array}{ccc|c} R_{11} & R_{12} & R_{13} & R_{14} \\ 0 & R_{22} & R_{23} & R_{24} \\ 0 & 0 & R_{33} & R_{34} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \end{aligned}$$

Example

Result: since $B^{(3)}$ is zero, the algorithm terminates with the factorization

$$
\begin{bmatrix} a_2 & a_4 & a_3 & a_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & -1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & -1 & -1 & 0 \end{bmatrix}
$$

$$
= \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & -1/2 & 1/\sqrt{2} \\ 1/2 & 1/2 & 0 \\ 1/2 & -1/2 & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 & 1 \\ 0 & 2 & 0 & 1 \\ 0 & 0 & \sqrt{2} & 0 \end{bmatrix}
$$

Exercise

use the result on page 6.34 to show that

$$R_{11} \geq R_{22} \geq \cdots \geq R_{rr}$$