ECEn 671: Mathematics of Signals and Systems Moon: Chapter 5.

Randal W. Beard

Brigham Young University

October 23, 2020

Table of Contents

Section 1

LU Factorization

LU Factorization

- Suppose that $A \in \mathbb{C}^{n \times n}$ is full rank. What is a numerically efficient method for computing the solution to $Ax = b$, i.e. $x = A^{-1}b$?
- An explicit formula is:

$$x = \frac{adj(A)b}{det(A)}$$

  but this requires numerical computation of determinants.
- LU factorization is more efficient.

LU Factorization: Basic Idea

- Find a permutation matrix P, a lower diagonal matrix with ones on the diagonal L, and an upper diagonal matrix U such that
$$PA = LU.$$
- How? Will illustrate by example:

LU Factorization: cont.

Let

$$A = \begin{pmatrix} 1 & -2 & 3 \\ -4 & 5 & -6 \\ 7 & -8 & 9 \end{pmatrix}$$

The idea is to perform row reductions to get a triangular matrix.
Key Idea: Reduce the row with the largest element.

LU Factorization: cont.

First, permute A to get the third row on top:

$$P_{13}A = \underbrace{\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}}_{P_{13}} \begin{pmatrix} 1 & -2 & 3 \\ -4 & 5 & -6 \\ 7 & -8 & 9 \end{pmatrix}$$

$$= \begin{pmatrix} 7 & -8 & 9 \\ -4 & 5 & -6 \\ 1 & -2 & 3 \end{pmatrix}$$

The idea is that you always want to divide by the largest element (in absolute value) in the row to avoid numerical problems.

LU Factorization: cont.

Now zero out the -4 and 1 by multiplying the first row by $+\frac{4}{7}$ and adding to the second row and multiplying the first row by $-\frac{1}{7}$ and adding to the third row:

$$E_1 P_{13} A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ \frac{4}{7} & 1 & 0 \\ \frac{-1}{7} & 0 & 1 \end{pmatrix}}_{E_1} \begin{pmatrix} 7 & -8 & 9 \\ -4 & 5 & -6 \\ 1 & -2 & 3 \end{pmatrix}$$

$$= \begin{pmatrix} 7 & -8 & 9 \\ 0 & 0.4286 & -5.4286 \\ 0 & -0.8571 & 2.8571 \end{pmatrix}$$

LU Factorization: cont.

Now permute (or "pivot") to get the largest (in absolute value) number in the second column in the second row:

$$P_{23}E_1P_{13}A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}_{P_{23}} \begin{pmatrix} 7 & -8 & 9 \\ 0 & 0.4286 & -5.4286 \\ 0 & -0.8571 & 2.8571 \end{pmatrix}$$

$$= \begin{pmatrix} 7 & -8 & 9 \\ 0 & -0.8571 & 2.8571 \\ 0 & 0.4286 & -5.4286 \end{pmatrix}$$

LU Factorization: cont.

Zero out the 0.4286 by multiplying the second row by $\frac{0.4286}{0.8571}$ and adding to the third row:

$$E_2 P_{23} E_1 P_{13} A = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{0.4286}{0.8571} & 1 \end{pmatrix}}_{E_2} \begin{pmatrix} 7 & -8 & 9 \\ 0 & -0.8571 & 2.8571 \\ 0 & 0.4286 & -5.4286 \end{pmatrix}$$

$$= \begin{pmatrix} 7 & -8 & 9 \\ 0 & -0.8571 & 2.8571 \\ 0 & 0 & -4 \end{pmatrix}$$

$$= U$$

Therefore

$$A = (E_2 P_{23} E_1 P_{13})^{-1} U$$

$$= P_{13}^{-1} E_1^{-1} P_{23}^{-1} E_2^{-1} U$$

LU Factorization: cont.

Note that if $E_1 = \begin{pmatrix} 1 & 0 & 0 \\ \frac{4}{7} & 1 & 0 \\ -\frac{1}{7} & 0 & 1 \end{pmatrix}$, then $E_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{4}{7} & 1 & 0 \\ \frac{1}{7} & 0 & 1 \end{pmatrix}$

since

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{4}{7} & 1 & 0 \\ -\frac{1}{7} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -\frac{4}{7} & 1 & 0 \\ \frac{1}{7} & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

So the inverse of any lower diagonal matrix formed by multiplying and adding rows is found by negating the off-diagonal terms.

Therefore E_1^{-1} and E_2^{-1} are easy to compute.

LU Factorization: cont.

Also note that for permutation matrices

$$P_{ij}^{-1} = P_{ji}$$

since

$$\underbrace{P_{ij}}_{\text{switch } ij \text{ rows}} \quad \underbrace{P_{ij}^{-1}}_{\text{switch back}} = I.$$

For example

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

LU Factorization: cont.

So we have $A = VU$ where
$$V = P_{13}E_1^{-1}P_{23}E_2^{-1} = \begin{pmatrix} 0.1429 & 1 & 0 \\ -0.5714 & -0.5 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Note that V is not lower triangular but

$$L = P_{23}P_{13}V = P_{23}\begin{pmatrix} 1 & 0 & 0 \\ -0.5714 & -0.5 & 1 \\ 0.1429 & 1 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 & 0 \\ 0.1429 & 1 & 0 \\ -0.5714 & -0.5 & 1 \end{pmatrix}$$

is, so $P_{23}P_{13}A = P_{23}P_{13}VU$. Therefore

$$PA = LU$$

where $P = P_{23}P_{13}$.

LU Factorization: cont.

For our example we have

$$\underbrace{\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}}_{P} \underbrace{\begin{pmatrix} 1 & -2 & 3 \\ -4 & 5 & -6 \\ 7 & -8 & 9 \end{pmatrix}}_{A} = \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0.1429 & 1 & 0 \\ -0.5714 & -0.5 & 1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} 7 & -8 & 9 \\ 0 & -0.8571 & 2.8571 \\ 0 & 0 & -4 \end{pmatrix}}_{U}$$

How do we solve the equation $Ax = b$?

Suppose $b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

Note that

$$PAx = Pb = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

So that

$$LUx = Pb.$$

LU Factorization: cont.

Let $y = Ux$ then

$$
\begin{aligned}
& Ly = Pb \\
\Longrightarrow \quad & \begin{pmatrix} 1 & 0 & 0 \\ 0.1429 & 1 & 0 \\ -0.5714 & -0.5 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix} \\
\Longrightarrow \quad & \begin{cases} y_1 & = 3 \\ y_2 & = 1 - 0.1429 y_1 \\ y_3 & = 2 + 0.5714 y_1 + 0.5 y_2 \end{cases} \qquad \text{(easy to solve)} \\
\Longrightarrow \quad & \begin{cases} y_1 & = 3 \\ y_2 & = 0.5741 \\ y_3 & = 4 \end{cases} \qquad \text{(easy to solve)}
\end{aligned}
$$

LU Factorization: cont.

Next solve $Ux = y$ for x:

$$
\begin{aligned}
& Ux = y \\
\Longrightarrow \quad & \begin{pmatrix} 7 & -8 & 9 \\ 0 & -0.8571 & 2.8571 \\ 0 & 0 & -4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ 0.5714 \\ 4 \end{pmatrix} \\
\Longrightarrow \quad & \begin{cases} -4x_3 & = 4 \\ -0.8571x_2 & = 0.5714 - 2.8571x_2 \\ 7x_1 & = 3 + 8x_2 - x_3 \end{cases} \qquad \text{(easy to solve)} \\
\Longrightarrow \quad & \begin{cases} x_1 & = -4 \\ x_2 & = -4 \\ x_3 & = -1 \end{cases}
\end{aligned}
$$

LU Factorization: cont.

In Matlab:

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 0];
[L, U, P] = lu(A)
```

In Python:

```
import numpy as np
import scipy.linalg as linalg

A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
P, L, U = linalg.lu(A)
```

Homework problem: Write your own custom `lu` function and compare to the built in `lu` function on 100 randomly generated matrices.

Section 2

Cholesky Factorization

Square Root of a Matrix

- If $B = B^H > 0$ then we can compute the "square root" of B as $B = QQ^H$ where $Q = B^{\frac{1}{2}}$ is the square root of B.
- In general, the square root of a matrix is not unique!

Example

Let $B = \begin{pmatrix} 9 & 0 \\ 0 & 0 \end{pmatrix}$

We can write

$$B = \begin{pmatrix} 3 \\ 0 \end{pmatrix} \begin{pmatrix} 3 & 0 \end{pmatrix}$$

and

$$B = \begin{pmatrix} 3 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 0 \end{pmatrix}$$

So both $Q = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$ and $Q = \begin{pmatrix} 3 & 0 \\ 0 & 0 \end{pmatrix}$ are square roots of B.

Cholesky Factorization

Definition

The Cholesky factorization of B is a square lower triangular square root $L \in \mathbb{C}^{n \times n}$ of B, where

$$B = LL^H.$$

Note that this can also be written as

$$B = U^H U$$

where $U = L^H$ is upper triangular.

Cholesky Factorization: Numerical Algorithm

Let $B = \begin{pmatrix} \alpha & \mathbf{v}^H \\ \mathbf{v} & B_1 \end{pmatrix}$. Then factor B as

$$\begin{aligned} B &= \begin{pmatrix} \alpha & \mathbf{v}^H \\ \mathbf{v} & B_1 \end{pmatrix} \\ &= \begin{pmatrix} \sqrt{\alpha} & 0 \\ \frac{\mathbf{v}}{\sqrt{\alpha}} & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & B_1 - \frac{\mathbf{v}\mathbf{v}^H}{\alpha} \end{pmatrix} \begin{pmatrix} \sqrt{\alpha} & \frac{\mathbf{v}^H}{\sqrt{\alpha}} \\ 0 & I_{n-1} \end{pmatrix} \end{aligned}$$

Cholesky Factorization: Numerical Algorithm, cont.

(RECURSIVE ALGORITHM)

Now find the Cholesky factorization of $B_1 - \frac{\mathbf{v}\mathbf{v}^H}{\alpha} \triangleq G_1 G_1^H$, so that

$$B = \begin{pmatrix} \sqrt{\alpha} & 0 \\ \frac{\mathbf{v}}{\sqrt{\alpha}} & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & G_1 G_1^H \end{pmatrix} \begin{pmatrix} \sqrt{\alpha} & \frac{\mathbf{v}^H}{\sqrt{\alpha}} \\ 0 & I_{n-1} \end{pmatrix}$$
$$= \begin{pmatrix} \sqrt{\alpha} & 0 \\ \frac{\mathbf{v}}{\sqrt{\alpha}} & G_1 \end{pmatrix} \begin{pmatrix} \sqrt{\alpha} & \frac{\mathbf{v}^H}{\sqrt{\alpha}} \\ 0 & G_1^H \end{pmatrix}$$

which implies that the Cholesky factor is

$$L = \begin{pmatrix} \sqrt{\alpha} & 0 \\ \frac{\mathbf{v}}{\sqrt{\alpha}} & G_1 \end{pmatrix}.$$

Cholesky Factorization: Example

Let $B = \begin{pmatrix} 1 & 2 & 4 & 1 \\ 2 & 13 & 17 & 8 \\ 4 & 17 & 29 & 16 \\ 1 & 8 & 16 & 30 \end{pmatrix}$. Then

$$B = \begin{pmatrix} \alpha_1 & \mathbf{v}_1^\top \\ \mathbf{v}_1 & B_1 \end{pmatrix},$$

where

$$\begin{aligned} \alpha_1 &= 1 \\ \mathbf{v}_1 &= \begin{pmatrix} 2 & 4 & 1 \end{pmatrix}^\top \\ B_1 &= \begin{pmatrix} 13 & 17 & 8 \\ 17 & 29 & 16 \\ 8 & 16 & 30 \end{pmatrix}. \end{aligned}$$

Cholesky Factorization: Example, cont.

Therefore

$$B = \begin{pmatrix} \sqrt{\alpha_1} & 0^\top \\ \frac{\mathbf{v}_1}{\sqrt{\alpha_1}} & G_1 \end{pmatrix} \begin{pmatrix} \sqrt{\alpha_1} & \frac{\mathbf{v}_1^\top}{\sqrt{\alpha_1}} \\ 0 & G_1^\top \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & & & \\ 4 & & G_1 & \\ 1 & & & \end{pmatrix} \begin{pmatrix} 1 & 2 & 4 & 1 \\ 0 & & & \\ 0 & & G_1^\top & \\ 0 & & & \end{pmatrix}$$

where

$$G_1 G_1^\top = B_1 - \frac{\mathbf{v}_1 \mathbf{v}_1^\top}{\alpha_1}$$

$$= \begin{pmatrix} 13 & 17 & 8 \\ 17 & 29 & 16 \\ 8 & 16 & 30 \end{pmatrix} - \frac{1}{1} \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix} \begin{pmatrix} 2 & 4 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 9 & 9 & 6 \\ 9 & 13 & 12 \\ 6 & 12 & 29 \end{pmatrix}.$$

Cholesky Factorization: Example, cont.

Therefore

$$G_1 G_1^\top = \begin{pmatrix} \sqrt{\alpha_2} & \mathbf{0}^\top \\ \frac{\mathbf{v}_2}{\sqrt{\alpha_2}} & G_2 \end{pmatrix} \begin{pmatrix} \sqrt{\alpha_2} & \frac{\mathbf{v}_2^\top}{\sqrt{\alpha_2}} \\ \mathbf{0} & G_2^\top \end{pmatrix}$$

$$= \begin{pmatrix} 3 & 0 & 0 \\ 3 & & \\ 2 & & G_2 \end{pmatrix} \begin{pmatrix} 3 & 3 & 2 \\ 0 & & \\ 0 & & G_2^\top \end{pmatrix}$$

where

$$G_2 G_2^\top = B_2 - \frac{\mathbf{v}_2 \mathbf{v}_2^\top}{\alpha_2}$$

$$= \begin{pmatrix} 13 & 12 \\ 12 & 29 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 9 \\ 6 \end{pmatrix} \begin{pmatrix} 9 & 6 \end{pmatrix}$$

$$= \begin{pmatrix} 4 & 6 \\ 6 & 25 \end{pmatrix}.$$

Cholesky Factorization: Example, cont.

Therefore

$$
\begin{aligned}
G_2 G_2^\top &= \begin{pmatrix} \sqrt{\alpha_3} & 0^\top \\ \frac{\mathbf{v}_3}{\sqrt{\alpha_3}} & G_3 \end{pmatrix} \begin{pmatrix} \sqrt{\alpha_3} & \frac{\mathbf{v}_3^\top}{\sqrt{\alpha_3}} \\ 0 & G_3^\top \end{pmatrix} \\
&= \begin{pmatrix} 2 & 0 \\ 3 & G_3 \end{pmatrix} \begin{pmatrix} 2 & 3 \\ 0 & G_3^\top \end{pmatrix}
\end{aligned}
$$

where

$$
\begin{aligned}
G_3 G_3^\top &= B_3 - \frac{\mathbf{v}_3 \mathbf{v}_3^\top}{\alpha_3} \\
&= 25 - \frac{1}{4} 3 \cdot 3 \\
&= 16
\end{aligned}
$$

Therefore $G_3 = 4$.

Cholesky Factorization: Example, cont.

Combining gives

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & & & \\ 4 & & G_1 & \\ 1 & & & \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 4 & 3 & & \\ 1 & 2 & & G_2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 4 & 3 & 2 & 0 \\ 1 & 2 & 3 & G_3 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 4 & 3 & 2 & 0 \\ 1 & 2 & 3 & 4 \end{pmatrix}.$$

Applications of Cholesky Factorization: Quadratic Forms

The quadratic form
$$x^H Qx = \|x\|_Q^2$$
where $Q = Q^H$, can be written as
$$x^H Qx = x^H U^H Ux = \|Ux\|_2^2$$
where $Q = U^H U = LL^H$

In other words, work with the regular 2-norm as opposed to the Q norm.

Applications of Cholesky Factorization: Simulating a random vector

Suppose you want to generate in Matlab/Simulink/Python/etc. a Gaussian random vector with covariance $R = R^T > 0$.

The Matlab `randn([m,1])` command returns an $m \times 1$ random vector which is normally distributed with zero mean and co-variance I ($\mathcal{N}(0, I)$).

To generate $\mathcal{N}(0, R)$ let $R = LL^T$ and let $z = Lx$ where $x \sim \mathcal{N}(0, I)$.

Then

$$E\{zz^T\} = E\{Lxx^TL^T\} = LE\{xx^T\}L^T = LL^T = R$$
$$\Rightarrow z \sim \mathcal{N}(0, R).$$

Applications of Cholesky Factorization: Solving normal equations

Normal equations are given by

$$R\mathbf{c} = \mathbf{b}$$

where $R = R^H$ is the Grammian and full rank if the data vectors are linearly independent.

Let $R = LL^H$, then $LL^H\mathbf{c} = \mathbf{b}$

First solve

$$L\mathbf{y} = \mathbf{b}$$

by forward substitution, and then solve

$$L^H\mathbf{c} = \mathbf{y}$$

by backward substitution.

Applications of Cholesky Factorization: Kalman filtering

In Kalman filtering we propagate two items; The estimate $\hat{x}(k)$ and the error covariance $P(k)$ where $P(k) = P^T(k) > 0$.

If implemented directly, numerical error can cause $P(k)$ to become indefinite introducing large errors into the estimate $\hat{x}(k)$.

To avoid this problem a "square root" Kalman filter is usually implemented where $P(k) = L(k)L^T(k)$ and $L(k)$ is propagated instead of $P(k)$. Then even with numerical errors in $L(k)$, $P(k)$ is still symmetric positive definite.

Cholesky Factorization: cont.

In Matlab:

```
L1 = [2, 0, 0; 3, 4, 0; 5, 6, 7];
A = L1 * L1';
L = chol(A)'
```

In Python:

```
import numpy as np
import scipy.linalg as linalg

L1 = np.array([[2, 0, 0], [3, 4, 0], [5, 6, 7]])
A = L1 @ L1.T
L = linalg.cholesky(A)
```

L should equal L_1.

Note that both Matlab and Python return an upper triangular matrix.

Homework problem: Write your own custom `cholesky` function and compare to the built in `cholesky` function on 100 randomly generated symmetric matrices.

Section 3

QR Factorization

Unitary and Orthogonal Matrices

Definition

$Q \in \mathbb{C}^{m \times m}$ is unitary if

$$Q^H Q = QQ^H = I$$

Equivalently $Q^{-1} = Q^H$.
Equivalently, the rows of Q form an orthonormal set.
Equivalently, the columns of Q form an orthonormal set.

Definition

$Q \in \mathbb{R}^{m \times m}$ is orthogonal if

$$Q^T Q = QQ^T = I.$$

Rotation matrices are examples of orthogonal matrices.

Hermitian Matrices

Definition

$Q \in \mathbb{C}^{m \times m}$ is <u>Hermitian</u> if $Q^H = Q$

Hermitian matrices are like real numbers, i.e., $\bar{z} = z$.
Unitary matrices correspond to the unit circle

$$|z|^2 = \bar{z}z = 1$$

Bilinear transformation

$$z = \frac{1 + jr}{1 - jr} \text{ maps the real line to the unit circle}$$

For matrices this becomes Cayley's formula

$$U = (I + jR)(I - jR)^{-1}$$

which maps Hermitian (analagous to real $\#$'s) to unitary matrices (analagous to complex unit circle).

Unitary Matrices, cont

Lemma (Moon Lemma 5.1)

Let $Q \in \mathbb{C}^{m \times m}$ then $\|Qx\|_2 = \|x\|_2, \forall x \in \mathbb{C}^m$ iff Q is unitary.

Proof.

If Q is unitary then

$$\|Qx\|_2 = \langle Qx, Qx \rangle^{\frac{1}{2}} = (x^H Q^H Q x)^{\frac{1}{2}} = (x^H x)^{\frac{1}{2}} = \|x\|_2$$

Conversely if $\|Qx\|_2 = \|x\|_2, \quad \forall x \in \mathbb{C}^m$ then

$$\begin{aligned} & x^H Q^H Q x = x^H x \qquad \forall x \in \mathbb{C}^m \\ \iff & x^H (Q^H Q - I) x = 0 \qquad \forall x \in \mathbb{C}^m \\ \iff & Q^H Q = I. \end{aligned}$$

Therefore Q is unitary. □

Unitary Matrices, cont

Lemma (Moon Lemma 5.2)

If $Y = QX$ *where* Q*-unitary then*

$$\|Y\|_F = \|X\|_F$$

Unitary Matrices, cont.

Lemma

If Q_1 and Q_2 are unitary then $Q_2 Q_1$ is unitary.

Proof.

$$(Q_2 Q_1)^H (Q_2 Q_1) = Q_1^H Q_2^H Q_2 Q_1 = Q_1^H Q_1 = I.$$

$\square$

QR - Factorization

Definition

Let $A \in \mathbb{C}^{m \times n}$. The QR factorization of A is given by

$$A = QR$$

where $Q \in \mathbb{C}^{m \times m}$ is unitary and $R \in \mathbb{C}^{m \times n}$ is upper triangular.

Lemma

Every matrix $A \in \mathbb{C}^{m \times n}$ has a QR factorization.

QR - Factorization, cont.

Example

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} -0.1961 & -0.9806 \\ -0.9806 & 0.1961 \end{pmatrix}}_{Q} \underbrace{\begin{pmatrix} -5.0090 & -6.2757 & -7.4524 & -8.6291 \\ 0 & -0.7845 & -1.5689 & -2.3534 \end{pmatrix}}_{R}$$

In Matlab:

```
[Q,R] = qr(A)
```

In Python:

```
Q, R = scipy.linalg.qr(A)
```

QR - Factorization, cont.

Example

$$A = \begin{pmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{pmatrix}$$

$$= \underbrace{\begin{pmatrix} -0.1826 & -0.8165 & -0.4001 & -0.3741 \\ -0.3651 & -0.4082 & 0.2546 & 0.797 \\ -0.5477 & 0 & 0.6910 & -0.4717 \\ -0.7303 & 0.4082 & -0.5455 & 0.0488 \end{pmatrix}}_{Q} \underbrace{\begin{pmatrix} -5.4772 & -12.7 \\ 0 & -3.266 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}}_{R}$$

Application: Full rank least squares

If $A \in \mathbb{C}^{m \times n}$ is full rank and $m > n$, find

$$\hat{x} = \arg\min \|Ax - b\|_2$$

Recall that the solution is $\hat{x} = (A^H A)^{-1} A^H b$ but

$$\mathcal{K}(A^H A) = (\mathcal{K}(A))^2$$

Therefore computing the inverse of $A^H A$ with LU or Cholesky factorization may be ill-advised.

Use QR factorization instead.

Application: Full rank least squares, cont.

Let $A = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$ where $Q \in \mathbb{C}^{m \times m}$ and $R_1 \in \mathbb{C}^{n \times n}$ is upper triangular.

Let $Q^H \mathbf{b} = \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix}$ where $\mathbf{c} \in \mathbb{C}^n$ and $\mathbf{d} \in \mathbb{C}^{m-n}$.

Then

$$\begin{aligned}
\|A\mathbf{x} - \mathbf{b}\|_2^2 &= \|QR\mathbf{x} - \mathbf{b}\|_2^2 \\
&= \left\| Q(R\mathbf{x} - Q^H\mathbf{b}) \right\|_2^2 \qquad (\text{ since } QQ^H = I) \\
&= \left\| \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix} \right\|_2^2 \qquad (\text{by lemma 5.1}) \\
&= \|R_1\mathbf{x} - \mathbf{c}\|_2^2 + \|\mathbf{d}\|_2^2 \qquad (\text{by definition of 2-norm})
\end{aligned}$$

so $\hat{\mathbf{x}} = \arg\min \|A\mathbf{x} - \mathbf{b}\|_2^2$ satisfies $R_1\hat{\mathbf{x}} = \mathbf{c}$ where $\hat{\mathbf{x}}$ is easily found by forward-substitution.

Application: Full rank least squares, cont.

Note that we don't actually need to compute all of Q since $Q^H\mathbf{b} = \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix}$.

Let

$$Q = \begin{pmatrix} \underset{m\times n}{Q_1} & \underset{m\times(m-n)}{Q_2} \end{pmatrix}$$

then

$$Q^H b = \begin{pmatrix} Q_1^H \\ Q_2^H \end{pmatrix} \mathbf{b} = \begin{pmatrix} Q_1^H\mathbf{b} \\ Q_2^H\mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{c} \\ \mathbf{d} \end{pmatrix}$$

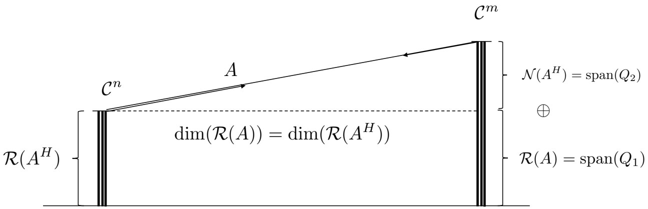
so $\mathbf{c} = Q_1^H\mathbf{b}$.

Therefore, we only need the first n columns of Q.

QR Factorization and Fundamental Subspaces

If A is tall then

$$A = QR$$
$$= \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_1 \\ \mathbf{0} \end{pmatrix}$$



Computational Methods for QR Factorization

We will discuss two methods for computing the QR Factorization:

- Given rotation.
- Householder transformation.

QR Factorization using Given Rotation

The basic idea is to diagonalize A one element at at time: So find

Q_1 such that $Q_1 A = \begin{pmatrix} x & x \\ 0 & x \\ x & x \end{pmatrix}$

Then find Q_2 such that $Q_2 Q_1 A = \begin{pmatrix} x & x \\ 0 & x \\ 0 & x \end{pmatrix}$

Then find Q_3 such that $Q_3 Q_2 Q_1 A = \begin{pmatrix} x & x \\ 0 & x \\ 0 & 0 \end{pmatrix}$

Then

$$\begin{aligned} A &= (Q_3 Q_2 Q_1)^{-1} R \\ &= (Q_3 Q_2 Q_1)^H R \qquad \text{(since } (Q_3 Q_2 Q_1) \text{ is unitary)} \\ &= \underbrace{Q_1^H Q_2^H Q_3^H}_{\hat{=} Q} R \\ &= QR. \end{aligned}$$

QR Factorization using Givens Rotations

Consider the 2×2 rotation matrix

$$G(\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

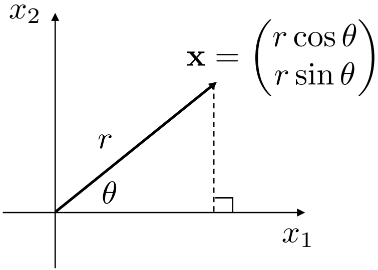
Note that

$$G^{-1}(\theta) = G^T(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ +\sin\theta & \cos\theta \end{pmatrix}$$

Therefore, $G(\theta)$ is orthogonal and hence unitary.

QR Factorization using Givens Rotations

Let $x = \begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix} \in \mathbb{R}^2$:



Then

$$
\begin{aligned}
G(\theta)x &= \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix} \\
&= \begin{pmatrix} r\cos^2(\theta) + r\sin^2(\theta) \\ -r\sin\theta\cos\theta + r\cos\theta\sin\theta \end{pmatrix} \\
&= \begin{pmatrix} r \\ 0 \end{pmatrix}.
\end{aligned}
$$

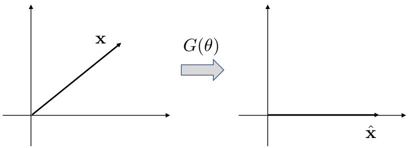
QR Factorization using Givens Rotations

Therefore $G(\theta)$ rotated

$$x = \begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix}$$

to

$$\hat{x} = \begin{pmatrix} r \\ 0 \end{pmatrix}.$$



QR Factorization using Givens Rotations

Note that if $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ then $\theta = \tan^{-1}\left(\frac{x_2}{x_1}\right)$ and

$$\cos\theta = \cos\left(\tan^{-1}\left(\frac{x_2}{x_1}\right)\right) = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}$$

$$\sin\theta = \sin\left(\tan^{-1}\left(\frac{x_2}{x_1}\right)\right) = \frac{x_2}{\sqrt{x_1^2 + x_2^2}}$$

Therefore

$$G_x(\theta) = \begin{pmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} \\ -\frac{x_2}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} \end{pmatrix}.$$

Note that each term in $G_x(\theta)$ decreases as a result of dividing by $\frac{1}{\sqrt{x_1^2+x_2^2}}$ so even if x_1 and x_2 are small, this is numerically stable.

QR Factorization using Givens Rotations: Example

Let

$$A = \begin{pmatrix} 1 & 6 & 7 & 12 \\ 2 & 5 & 8 & 11 \\ 13 & 4 & 9 & 10 \end{pmatrix}$$

Letting $x_1 = 1$ and $x_2 = 2$ and

$$Q_1 = \begin{pmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} & 0 \\ -\frac{x_2}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.4472 & 0.8944 & 0. \\ -0.8944 & 0.4472 & 0. \\ 0 & 0 & 1 \end{pmatrix}$$

gives

$$Q_1 A = \begin{pmatrix} 2.2360 & 7.1554 & 10.2859 & 15.2052 \\ 0 & -3.1304 & -2.6832 & -5.8137 \\ 13 & 4 & 9 & 10 \end{pmatrix}.$$

QR Factorization using Givens Rotations: Example

$$Q_1 A = \begin{pmatrix} 2.2360 & 7.1554 & 10.2859 & 15.2052 \\ 0 & -3.1304 & -2.6832 & -5.8137 \\ 13 & 4 & 9 & 10 \end{pmatrix}.$$

Letting $x_1 = 2.2360$ and $x_2 = 13$ and

$$Q_2 = \begin{pmatrix} \frac{x_1}{\sqrt{x_1^2+x_2^2}} & 0 & \frac{x_2}{\sqrt{x_1^2+x_2^2}} \\ 0 & 1 & 0 \\ -\frac{x_2}{\sqrt{x_1^2+x_2^2}} & 0 & \frac{x_1}{\sqrt{x_1^2+x_2^2}} \end{pmatrix} = \begin{pmatrix} 0.1695 & 0 & 0.9855 \\ 0 & 1 & 0 \\ -0.9855 & 0 & 0.1695 \end{pmatrix}$$

gives

$$Q_2 Q_1 A = \begin{pmatrix} 13.1909 & 5.1550 & 10.6133 & 12.4328 \\ 0 & -3.1304 & -2.6832 & -5.8137 \\ 0 & -6.3737 & -8.6114 & -13.2900 \end{pmatrix}.$$

QR Factorization using Givens Rotations: Example

$$Q_2 Q_1 A = \begin{pmatrix} 13.1909 & 5.1550 & 10.6133 & 12.4328 \\ 0 & -3.1304 & -2.6832 & -5.8137 \\ 0 & -6.3737 & -8.6114 & -13.2900 \end{pmatrix}.$$

Letting $x_1 = -3.1304$ and $x_2 = -6.3737$ and

$$Q_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{x_1}{\sqrt{x_1^2+x_2^2}} & \frac{x_2}{\sqrt{x_1^2+x_2^2}} \\ 0 & -\frac{x_2}{\sqrt{x_1^2+x_2^2}} & \frac{x_1}{\sqrt{x_1^2+x_2^2}} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.44084797 & -0.8975 \\ 0 & 0.89758179 & -0.4408 \end{pmatrix}$$

gives

$$Q_3 Q_2 Q_1 A = \begin{pmatrix} 13.1909 & 5.1550 & 10.6133 & 12.4328 \\ 0 & 7.101 & 8.912 & 14.4918 \\ 0 & 0 & 1.3878 & 0.6405 \end{pmatrix}.$$

QR Factorization using Givens Rotations: Example

Therefore

$$\underbrace{\begin{pmatrix} 1 & 6 & 7 & 12 \\ 2 & 5 & 8 & 11 \\ 13 & 4 & 9 & 10 \end{pmatrix}}_{A}$$

$$= \underbrace{\begin{pmatrix} 0.0967 & 0.9077 & -0.4082 \\ 0.4834 & 0.3157 & 0.816 \\ 0.8701 & -0.2763 & 0.4082 \end{pmatrix}}_{Q} \underbrace{\begin{pmatrix} 13.1909 & 5.1550 & 10.6133 & 12.4328 \\ 0 & 7.101 & 8.912 & 14.4918 \\ 0 & 0 & 1.3878 & 0.6405 \end{pmatrix}}_{R}$$

where

$$\begin{aligned} Q &= Q_1^H Q_2^H Q_3^H \\ &= \begin{pmatrix} 0.0758 & 0.7899 & -0.6085 \\ 0.1516 & 0.5940 & 0.7900 \\ 0.9855 & -0.1521 & -0.0747 \end{pmatrix}. \end{aligned}$$

QR Factorization using Givens Rotations: Example

```
import numpy as np

def Q_givens(x1, x2, size, m, n):
    Q = np.eye(size)
    cos_theta = x1 / np.sqrt(x1**2 + x2**2)
    sin_theta = x2 / np.sqrt(x1**2 + x2**2)
    Q[n-1, n-1] = cos_theta
    Q[m-1, m-1] = cos_theta
    Q[m-1, n-1] = -sin_theta
    Q[n-1, m-1] = sin_theta
    return Q

A = np.array([[1, 6, 7, 12],
              [2, 5, 8, 11],
              [13, 4, 9, 10]])
```

QR Factorization using Givens Rotations: Example

```
Q1 = Q_givens(x1=1, x2=2, size=3, m=2, n=1)
R1 = Q1 @ A

Q2 = Q_givens(x1=R1[0,0], x2=R1[2,0], size=3,
              m=3, n=1)
R2 = Q2 @ Q1 @ A

Q3 = Q_givens(x1=R2[1,1], x2=R2[2,1], size=3,
              m=3, n=2)
R3 = Q3 @ Q2 @ Q1 @ A

Q = Q1.conj().T @ Q2.conj().T @ Q3.conj().T
print("Q=", Q)
print("R=", R3)
```

QR Factorization using Householder Transformation

The basic idea is to diagonalize A one column at a time using unitary matrices.

Lemma

If Q_1 and Q_2 are unitary then $Q_2 Q_1$ is unitary.

Proof.

$$(Q_2 Q_1)^H (Q_2 Q_1) = Q_1^H Q_2^H Q_2 Q_1 = Q_1^H Q_1 = I.$$

$\square$

QR Factorization using Householder Transformation

So find Q_1 such that $Q_1 A = \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & x & x \\ 0 & x & x \end{pmatrix}$

Then find Q_2 such that $Q_2 Q_1 A = \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & x \end{pmatrix}$

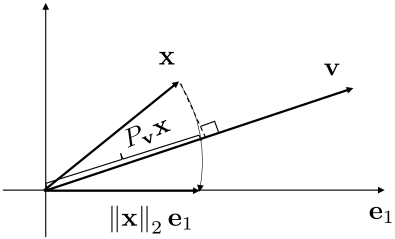
Then find Q_3 such that $Q_3 Q_2 Q_1 A = \begin{pmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \end{pmatrix}$

Then

$$\begin{aligned} A &= (Q_3 Q_2 Q_1)^{-1} R \\ &= (Q_3 Q_2 Q_1)^H R \qquad \text{(since } (Q_3 Q_2 Q_1) \text{ is unitary)} \\ &= \underbrace{Q_1^H Q_2^H Q_3^H}_{\hat{=} Q} R \end{aligned}$$

QR Factorization using Householder Transformation

Geometrically what do we want?



We would like to rotate x down to e_1. This can be thought of as a reflection of x about some vector v

We need an operator that transforms x to $y = \|x\|_2\, e_1$

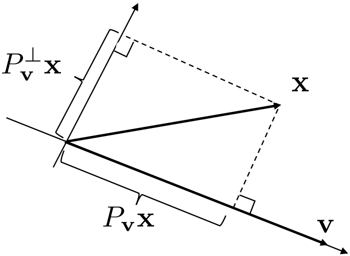
QR Factorization using Householder Transformation

Let

$$P_v = \frac{vv^H}{v^H v}$$

be the projection matrix that projects onto the vector v and let

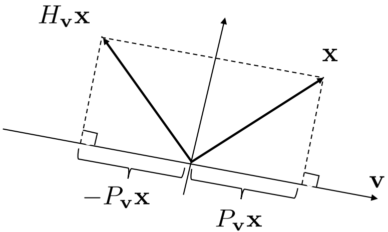
$$P_v^{\perp} = I - P_v$$



QR Factorization using Householder Transformation

The Householder transformation is

$$H_v = I - 2P_v$$



$H_{\mathbf{v}}\mathbf{x}$ reflects $\mathbf{x}$ about the vector that is orthogonal to $\mathbf{v}$, and in the same hyperplane as both $\mathbf{x}$ and $\mathbf{v}$.

QR Factorization using Householder Transformation

Lemma

H_v is unitary.

Proof.

$$
\begin{aligned}
H_v^H H_v &= (I - 2P_v^H)^H (I - 2P_v) \\
&= I - 2P_v - 2P_v + 4P_v^2 \\
&= I - 4P_v + 4P_v \qquad \text{(since } P_v^2 = P_v\text{)} \\
&= I
\end{aligned}
$$

$\square$

QR Factorization using Householder Transformation

Lemma

$H_v v = -v$

Proof.

$$H_v v = v - 2P_v v = v - 2v = -v$$

$\square$

Lemma

If $z \perp v$ *then* $H_v z = z$.

Proof.

$$H_v z = z - zP_v z = z$$

$\square$

QR Factorization using Householder Transformation

Find v so that

$$H_{\mathbf{v}}\mathbf{x} = \begin{pmatrix} \pm \|\mathbf{x}\|_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \pm \|\mathbf{x}\|_2 \, \mathbf{e}_1$$

i.e. the Householder transformation compresses all of the energy in $\mathbf{x}$ into the first component.

$$H_{\mathbf{v}}\mathbf{x} = \mathbf{x} - \frac{2\mathbf{v}\mathbf{v}^H}{\mathbf{v}^H\mathbf{v}}\mathbf{x} = \mathbf{x} - 2\frac{\mathbf{v}^H\mathbf{x}}{\mathbf{v}^H\mathbf{v}}\mathbf{v} = \pm \|\mathbf{x}\|_2 \, e_1$$

Therefore

$$\left(2\frac{\mathbf{v}^H\mathbf{x}}{\mathbf{v}^H\mathbf{v}}\right)\mathbf{v} = \mathbf{x} \pm \|\mathbf{x}\|_2 \, e_1$$

which implies that $\mathbf{v}$ is a scalar multiple of $\mathbf{x} \pm \|\mathbf{x}\|_2 \, e_1$.

QR Factorization using Householder Transformation

- Let $\mathbf{v} = \mathbf{x} \pm \|\mathbf{x}\|_2 \, e_1$.
- Numerically we would like $\mathbf{v}$ to be large so that dividing by $\frac{1}{\mathbf{v}^H \mathbf{v}}$ does not cause problems.
- Selecting $\mathbf{v} = \mathbf{x} + sign(x_1) \|\mathbf{x}\|_2 \, e_1$ implies that

$$\|\mathbf{v}\| = \|\mathbf{x} + sign(x_1) \|\mathbf{x}\|_2 \, e_1\| \geq \|\mathbf{x}\|$$

  (Since we only change the first element and the magnitude of that element always <u>increases</u> we can use $\geq$).
- Therefore, if $\mathbf{v} = \mathbf{x} + sign(x_1) \|\mathbf{x}\|_2 \, e_1$ then $H_{\mathbf{v}}\mathbf{x} = \begin{pmatrix} \|\mathbf{x}\|_2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ and $H_{\mathbf{v}}$ is numerically well conditioned, i.e. we are not dividing by small numbers.

QR Factorization using Householder Transformation

Suppose that $A = (a_1 \cdots a_n)$.

Letting $Q_1 = H_{v_1}$ where $v_1 = a_1 + sign(a_{11}) \|a_1\|_2 e_1$ implies that

$$Q_1 A = \begin{pmatrix} \|a_1\|_2 & * & \cdots & * \\ 0 & & & \\ \vdots & \tilde{a}_2 & \cdots & \tilde{a}_n \\ 0 & & & \end{pmatrix}.$$

Lemma

If S is unitary then

$$Q = \begin{pmatrix} I & 0 \\ 0 & S \end{pmatrix}$$

is unitary.

Proof.

$$QQ^H = \begin{pmatrix} I & 0 \\ 0 & S^H \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & S \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & S^H S \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} = I.$$

QR Factorization using Householder Transformation

Let $Q_2 = \begin{pmatrix} I & 0 \\ 0 & H_{v_2} \end{pmatrix}$ where $v_2 = \tilde{a}_2 + sign(\tilde{a}_{21}) \left\| \tilde{a}_2 \right\|_2 e_2$
Could also write as:

$$Q_2 = I - 2\frac{\tilde{v}_2 \tilde{v}_2^H}{\tilde{v}_2^H \tilde{v}_2} \qquad \text{where } \tilde{v}_2 = \begin{pmatrix} 0 \\ v_2 \end{pmatrix}$$

Then

$$Q_2 Q_1 A = \begin{pmatrix} \|a_1\|_2 & * & * & \cdots & * \\ 0 & \|\tilde{a}_2\| & * & \cdots & * \\ 0 & 0 & & & \\ \vdots & \vdots & \tilde{\tilde{a}}_3 & \cdots & \tilde{\tilde{a}}_n \\ 0 & 0 & & & \end{pmatrix}$$

The process is repeated until an upper triangular matrix is obtained on the right.

QR Factorization using Householder Transformation: Example

Let $A = \begin{pmatrix} 1 & -2 & 13 \\ -6 & 5 & -4 \\ 7 & -8 & 9 \\ -12 & 11 & -10 \end{pmatrix}$

Let $v_1 = \begin{pmatrix} 1 \\ -6 \\ 7 \\ -12 \end{pmatrix} + sign(1) \left\| \begin{pmatrix} 1 \\ -6 \\ 7 \\ -12 \end{pmatrix} \right\| e_1 = \begin{pmatrix} 6.1657 \\ -6 \\ 7 \\ -12 \end{pmatrix}$ and $Q_1 = I - 2\frac{v_1 v_1^H}{v_1^H v_1}$. Then

$$Q_1 A = \begin{pmatrix} -15.1657 & 14.5063 & -14.5063 \\ 0 & -1.1264 & 6.2091 \\ 0 & -0.8525 & -2.9106 \\ 0 & -1.2528 & 10.4182 \end{pmatrix}.$$

QR Factorization using Householder Transformation: Example

$$Q_1 A = \begin{pmatrix} -15.1657 & 14.5063 & -14.5063 \\ 0 & -1.1264 & 6.2091 \\ 0 & -0.8525 & -2.9106 \\ 0 & -1.2528 & 10.4182 \end{pmatrix}.$$

Let

$$v_2 = \begin{pmatrix} 0 \\ -1.1264 \\ -0.8525 \\ -1.2528 \end{pmatrix} + sign(-1.1264) \left\| \begin{pmatrix} 0 \\ -1.1264 \\ -0.8525 \\ -1.2528 \end{pmatrix} \right\| e_2 = \begin{pmatrix} 0 \\ -3.0146 \\ -0.8525 \\ -1.2528 \end{pmatrix}$$

which implies that

$$Q_2 = I - 2\frac{v_2 v_2^H}{v_2^H v_2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.5965 & -0.4514 & -0.6635 \\ 0 & -0.4514 & 0.8723 & -0.1876 \\ 0 & -0.6635 & -0.1876 & 0.72424585 \end{pmatrix}.$$

QR Factorization using Householder Transformation: Example

Then

$$Q_2 Q_1 A = \begin{pmatrix} -15.1657 & 14.5063 & -14.5063 \\ 0 & 1.88810 & -9.30270 \\ 0 & 0 & -7.29720 \\ 0 & 0 & 3.97160 \end{pmatrix}.$$

$$v_3 = \begin{pmatrix} 0 \\ 0 \\ -7.29720 \\ 3.97160 \end{pmatrix} + sign(-7.29720) \left\| \begin{pmatrix} 0 \\ -7.29720 \\ 3.97160 \end{pmatrix} \right\| e_3 = \begin{pmatrix} 0 \\ 0 \\ -15.6053 \\ 3.971 \end{pmatrix}$$

which implies that

$$Q_3 = I - 2\frac{v_3 v_3^H}{v_3^H v_3} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.8783 & 0.47804 \\ 0 & 0 & 0.4780 & 0.8783 \end{pmatrix}.$$

QR Factorization using Householder Transformation: Example

Then

$$Q_3 Q_2 Q_1 A = \begin{pmatrix} -15.1657 & 14.5063 & -14.5063 \\ 0 & 1.888 & -9.3027 \\ 0 & 0 & 8.3080 \\ 0 & 0 & 0 \end{pmatrix}.$$

$$\begin{aligned} Q &= Q_1^H Q_2^H Q_3^H \\ &= \begin{pmatrix} -0.0659 & -0.5526 & 0.8308 & 0 \\ 0.3956 & -0.3914 & -0.2289 & -0.79862957 \\ -0.4615 & -0.6907 & -0.4961 & 0.25219881 \\ 0.7912 & -0.2532 & -0.1056 & 0.54643076 \end{pmatrix} \end{aligned}$$

QR Factorization using Householder Transformation: Example

```
import numpy as np

def Q_householder(A, column):
    (m,n) = A.shape
    x = A[column-1:m, column-1:column]
    e = np.zeros(x.shape)
    e[0,0]=1
    v = x + np.sign(x[0, 0])
         * np.linalg.norm(x) * e
    H = np.eye(m)
    H[(column-1):m, (column-1):m]
         = np.eye(m-(column-1))
            - 2 * v @ v.T / (v.T @ v)
    return H
```

QR Factorization using Householder Transformation: Example

```
A = np.array([[1, -2, 13],
              [-6, 5, -4],
              [7, -8, 9],
              [-12, 11, -10]])
Q1 = Q_householder(A, column=1)
R1 = Q1 @ A
Q2 = Q_householder(R1, column=2)
R2 = Q2 @ Q1 @ A
Q3 = Q_householder(R2, column=3)
R3 = Q3 @ Q2 @ Q1 @ A
Q = Q1.conj().T @ Q2.conj().T @ Q3.conj().T
print("Q=", Q)
print("R=", R3)
```