Matrix Analysis and Applications
Chapter 8: QR Decomposition

Instructor: Kai Lu
(http://seit.sysu.edu.cn/teacher/1801)

School of Electronics and Information Technology
Sun Yat-sen University

December 13, 2020

Table of Contents

Table of Contents

QR Decomposition

QR decomposition: any $\boldsymbol{A} \in \mathbb{C}^{m \times n}$ can be decomposed as

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}, \tag{1}$$

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

- efficient to compute, a.k.a. poor man's SVD.
- can be used to compute
  - LS solutions;
  - bases of $\mathbb{R}(\boldsymbol{A})$ and $\mathbb{R}(\boldsymbol{A})_{\perp}$;
  - decompositions such as SVD.

QR Decomposition

QR decomposition: any $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ can be decomposed as

$$\boldsymbol{A} = \boldsymbol{QR}, \tag{1}$$

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

- efficient to compute, a.k.a. poor man's SVD.
- can be used to compute
  - LS solutions;
  - bases of $\mathbb{R}(\boldsymbol{A})$ and $\mathbb{R}(\boldsymbol{A})_\perp$;
  - decompositions such as SVD.

Question 1

1) *How do we know that QR decomposition exists?*
2) *How to compute the factors $\boldsymbol{Q}$ and $\boldsymbol{R}$ algorithmically?*

An Illustrative Application: solving LS problem

Consider the LS problem for a full column-rank $\boldsymbol{A} \in \mathbb{C}^{m\times n}$:

$$\min_{\boldsymbol{x}\in\mathbb{C}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2. \tag{2}$$

Let

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R}_1, \tag{3}$$

where $\boldsymbol{Q}_1 \in \mathbb{C}^{m\times n}, \boldsymbol{Q}_2 \in \mathbb{C}^{m\times(m-n)}, \boldsymbol{R}_1 \in \mathbb{C}^{n\times n}$ (upper triangular). Since

$$\begin{aligned} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 &= \left\|\boldsymbol{Q}^H(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y})\right\|_2^2 \\ &= \left\|\begin{bmatrix} \boldsymbol{Q}_1^H(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) \\ \boldsymbol{Q}_2^H(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) \end{bmatrix}\right\|_2^2 \qquad (4) \\ &= \left\|\boldsymbol{R}_1\boldsymbol{x} - \boldsymbol{Q}_1^H\boldsymbol{y}\right\|_2^2 + \left\|\boldsymbol{Q}_2^H\boldsymbol{y}\right\|_2^2, \qquad (5) \end{aligned}$$

the LS problem is equivalent to solving the upper triangular system

$$\boldsymbol{R}_1\boldsymbol{x} = \boldsymbol{Q}_1^H\boldsymbol{y}. \tag{6}$$

QR for Full Column-Rank Matrices

Theorem 1 (QR Decompostion)

Let $\boldsymbol{A} \in \mathbb{C}^{m\times n}$, *and suppose that* $\boldsymbol{A}$ *has full column rank. Then,* $\boldsymbol{A}$ *admits a decomposition*

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R}_1, \tag{7}$$

where $\boldsymbol{Q}_1 \in \mathbb{C}^{m\times n}$ *is semi-unitary;* $\boldsymbol{R}_1 \in \mathbb{C}^{n\times n}$ *is upper triangular and nonsingular.*

QR for Full Column-Rank Matrices

Theorem 1 (QR Decompostion)

Let $\boldsymbol{A} \in \mathbb{C}^{m\times n}$, and suppose that $\boldsymbol{A}$ has full column rank. Then, $\boldsymbol{A}$ admits a decomposition

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{0} \end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R}_1, \tag{7}$$

where $\boldsymbol{Q}_1 \in \mathbb{C}^{m\times n}$ is semi-unitary; $\boldsymbol{R}_1 \in \mathbb{C}^{n\times n}$ is upper triangular and nonsingular.

- We call $\boldsymbol{A} = \boldsymbol{Q}_1\boldsymbol{R}_1$ a thin QR of $\boldsymbol{A}$, and $(\boldsymbol{Q}_1, \boldsymbol{R}_1)$ a thin QR pair of $\boldsymbol{A}$.
- It follows directly that

Property 1

For the thin QR in Theorem 1, we have $\mathbb{R}(\boldsymbol{A}) = \mathrm{span}(\boldsymbol{Q}_1)$ and $\mathbb{R}(\boldsymbol{A})_\perp = \mathrm{span}(\boldsymbol{Q}_2)$. Moreover, $\boldsymbol{Q}_1\boldsymbol{Q}_1^H$ is orthogonal projector onto $\mathbb{R}(\boldsymbol{A})$.

QR for Full Column-Rank Matrices (cont'd)

Proof of **Theorem 1**:

1) $\boldsymbol{A}^H\boldsymbol{A}$ is positive definite.
2) Let $\boldsymbol{R}_1$ be the (upper triangular and nonsingular) Cholesky factor of $\boldsymbol{A}^H\boldsymbol{A}$ (Ch6, P41), i.e.,

$$\boldsymbol{A}^H\boldsymbol{A} = \boldsymbol{R}_1^H\boldsymbol{R}_1.$$

   Also, let $\boldsymbol{Q}_1 = \boldsymbol{A}\boldsymbol{R}_1^{-1}$.

3) It can be verified that $\boldsymbol{Q}_1^H\boldsymbol{Q}_1 = \boldsymbol{I}$ and $\boldsymbol{Q}_1\boldsymbol{R}_1 = \boldsymbol{A}$.

Application: the Geometrical Interpretation of Determinant

Question 2

When $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ has linearly independent columns, explain why the volume of the n-dimensional parallelepiped generated by the columns of $\boldsymbol{X}$ is

$$V_n = \left[\det\left(\boldsymbol{X}^T \boldsymbol{X}\right)\right]^{1/2}. \tag{8}$$

In particular, if $\boldsymbol{X}$ is square, then

$$V_n = \left|\det\left(\boldsymbol{X}\right)\right|. \tag{9}$$

Application: the Geometrical Interpretation of Determinant

Question 2

When $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ has linearly independent columns, explain why the volume of the n-dimensional parallelepiped generated by the columns of $\boldsymbol{X}$ is

$$V_n = \left[\det\left(\boldsymbol{X}^T\boldsymbol{X}\right)\right]^{1/2}. \tag{8}$$

In particular, if $\boldsymbol{X}$ is square, then

$$V_n = \left|\det\left(\boldsymbol{X}\right)\right|. \tag{9}$$

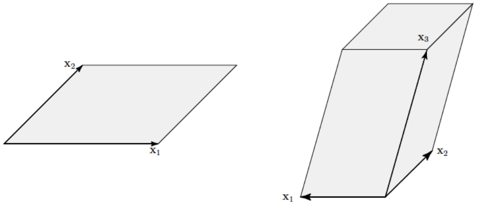


Figure 1: Geometric interpretation of determinant.

Application: Geometrical Interpretation of Determinant

Solution: Recalling the application 'Determine the Volume of n-Dimensional Parallelepiped' in Chapter 7, we know that if $\boldsymbol{X} = \boldsymbol{QR}$ is the QR factorization of $\boldsymbol{X}$, then the volume of the n-dimensional parallelepiped generated by the columns of $\boldsymbol{X}$ is

$$V_n = v_1 v_2 \cdots v_n = \det(\boldsymbol{R}), \tag{10}$$

where the v_k's are the diagonal elements of the upper-triangular matrix $\boldsymbol{R}$.

Application: Geometrical Interpretation of Determinant

Solution: Recalling the application 'Determine the Volume of n-Dimensional Parallelepiped' in Chapter 7, we know that if $\boldsymbol{X} = \boldsymbol{QR}$ is the QR factorization of $\boldsymbol{X}$, then the volume of the n-dimensional parallelepiped generated by the columns of $\boldsymbol{X}$ is

$$V_n = v_1 v_2 \cdots v_n = \det(\boldsymbol{R}), \tag{10}$$

where the v_k's are the diagonal elements of the upper-triangular matrix $\boldsymbol{R}$.

On the other hand, it is not hard to prove that

$$\det\left(\boldsymbol{X}^T\boldsymbol{X}\right) = \det\left(\boldsymbol{R}^T\boldsymbol{Q}^T\boldsymbol{Q}\boldsymbol{R}\right) = \left(\det\left(\boldsymbol{R}\right)\right)^2 = \left(v_1 v_2 \cdots v_n\right)^2 = V_n^2, \tag{11}$$

which leads to (8). If $\boldsymbol{X}$ is square, we have

$$\det\left(\boldsymbol{X}^T\boldsymbol{X}\right) = \det\left(\boldsymbol{X}^T\right)\det\left(\boldsymbol{X}\right) = \left(\det\left(\boldsymbol{X}\right)\right)^2 = \left(\det\left(\boldsymbol{R}\right)\right)^2 = V_n^2, \tag{12}$$

which implies (9).

Table of Contents

Classical Gram-Schmidt for Thin QR

Question 3

Given a full column-rank $\boldsymbol{A}$, how to compute its thin QR pair $(\boldsymbol{Q}_1, \boldsymbol{R}_1)$?

- The proof of Theorem 1 already suggests a way to compute $(\boldsymbol{Q}_1, \boldsymbol{R}_1)$, but there are more efficient ways.

Classical Gram-Schmidt for Thin QR

Question 3

Given a full column-rank $\boldsymbol{A}$, how to compute its thin QR pair $(\boldsymbol{Q}_1, \boldsymbol{R}_1)$?

- The proof of Theorem 1 already suggests a way to compute $(\boldsymbol{Q}_1, \boldsymbol{R}_1)$, but there are more efficient ways.

Suppose that $\boldsymbol{A} = \boldsymbol{Q}_1\boldsymbol{R}_1$ exists. Then, the k^{th} column of $\boldsymbol{A} = \boldsymbol{Q}_1\boldsymbol{R}_1$ equals

$$\boldsymbol{a}_k = r_{1k}\boldsymbol{q}_1 + r_{2k}\boldsymbol{q}_2 + \cdots + r_{kk}\boldsymbol{q}_k. \tag{13}$$

Stage 1: since $\boldsymbol{a}_1 = r_{11}\boldsymbol{q}_1$, we can obtain r_{11} and $\boldsymbol{q}_1$ via

$$r_{11} = \|\boldsymbol{a}_1\|_2,\ \boldsymbol{q}_1 = \boldsymbol{a}_1/\|\boldsymbol{a}_1\|_2. \tag{14}$$

Classical Gram-Schmidt for Thin QR (cont'd)

Stage 2: since $\boldsymbol{a}_2 = r_{12}\boldsymbol{q}_1 + r_{22}\boldsymbol{q}_2$ and $\boldsymbol{q}_1^H\boldsymbol{q}_2 = 0$, we can obtain r_{12} via

$$r_{12} = \boldsymbol{q}_1^H\boldsymbol{a}_2. \tag{15}$$

Also, we can obtain r_{22} and $\boldsymbol{q}_2$ via a cancellation trick:

$$r_{22} = \|\boldsymbol{y}_2\|_2,\ \boldsymbol{q}_2 = \boldsymbol{y}_2/\|\boldsymbol{y}_2\|_2, \tag{16}$$

where $\boldsymbol{y}_2 = \boldsymbol{a}_2 - r_{12}\boldsymbol{q}_1$.

Classical Gram-Schmidt for Thin QR (cont'd)

Stage 2: since $\boldsymbol{a}_2 = r_{12}\boldsymbol{q}_1 + r_{22}\boldsymbol{q}_2$ and $\boldsymbol{q}_1^H \boldsymbol{q}_2 = 0$, we can obtain r_{12} via

$$r_{12} = \boldsymbol{q}_1^H \boldsymbol{a}_2. \tag{15}$$

Also, we can obtain r_{22} and $\boldsymbol{q}_2$ via a cancellation trick:

$$r_{22} = \|\boldsymbol{y}_2\|_2, \ \boldsymbol{q}_2 = \boldsymbol{y}_2/\|\boldsymbol{y}_2\|_2, \tag{16}$$

where $\boldsymbol{y}_2 = \boldsymbol{a}_2 - r_{12}\boldsymbol{q}_1$.

Stage 3: we have $\boldsymbol{a}_3 = r_{13}\boldsymbol{q}_1 + r_{23}\boldsymbol{q}_2 + r_{33}\boldsymbol{q}_3$, Using the same trick as above yields

$$r_{13} = \boldsymbol{q}_1^H \boldsymbol{a}_3, \ r_{23} = \boldsymbol{q}_2^H \boldsymbol{a}_3, \tag{17}$$

and

$$\boldsymbol{q}_3 = \boldsymbol{y}_3/\|\boldsymbol{y}_3\|_2, \ r_{33} = \|\boldsymbol{y}_3\|_2, \tag{18}$$

where $\boldsymbol{y}_3 = \boldsymbol{a}_3 - r_{13}\boldsymbol{q}_1 - r_{23}\boldsymbol{q}_2$.

The same idea applies to the subsequent stages.

Classical Gram-Schmidt for Thin QR

Gram-Schmidt procedure: for $k = 1, \cdots, n$, compute

$$r_{ik} = \boldsymbol{q}_i^H \boldsymbol{a}_k, \quad i = 1, \cdots, k-1, \tag{19a}$$

$$\boldsymbol{y}_k = \boldsymbol{a}_k - \sum_{i=1}^{k-1} r_{ik} \boldsymbol{q}_i, \tag{19b}$$

$$\boldsymbol{q}_k = \frac{\boldsymbol{y}_k}{\|\boldsymbol{y}_k\|_2}, \tag{19c}$$

$$r_{kk} = \|\boldsymbol{y}_k\|_2. \tag{19d}$$

- There are variants with the implementations of Gram-Schmidt (numerical stability is usually the concern).

Classical Gram-Schmidt for Thin QR (cont'd)

1) Classical Gram-Schmidt procedure often suffers loss of orthogonality in finite-precision.

2) Also, separate storage is required for $\boldsymbol{A}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$, since original $\boldsymbol{a}_k$ are needed in inner loop, so $\boldsymbol{q}_k$ cannot overwrite columns of $\boldsymbol{A}$.

3) Both deficiencies are improved by modified Gram-Schmidt procedure, with each vector orthogonalized in turn against all subsequent vectors, so $\boldsymbol{q}_k$ can overwrite $\boldsymbol{a}_k$.

Modified Gram-Schmidt

Let $\boldsymbol{r}_i^T$ denote the i^{th} row of $\boldsymbol{R}_1$ (with a slight abuse of notations). The following illustrates a **modified** Gram-Schmidt, which is numerically more stable.

Modified Gram-Schmidt procedure: for $k = 1, \cdots, n$, compute

$$\boldsymbol{A} = \boldsymbol{Q}_1 \boldsymbol{R}_1 = \sum_{i=1}^{n} \boldsymbol{q}_i \boldsymbol{r}_i^T. \tag{20}$$

Modified Gram-Schmidt

Let $\boldsymbol{r}_i^T$ denote the i^{th} row of $\boldsymbol{R}_1$ (with a slight abuse of notations). The following illustrates a **modified** Gram-Schmidt, which is numerically more stable.

Modified Gram-Schmidt procedure: for $k = 1, \cdots, n$, compute

$$\boldsymbol{A} = \boldsymbol{Q}_1 \boldsymbol{R}_1 = \sum_{i=1}^{n} \boldsymbol{q}_i \boldsymbol{r}_i^T. \tag{20}$$

Suppose that $\boldsymbol{q}_1, \cdots, \boldsymbol{q}_{k-1}, \boldsymbol{r}_1, \cdots, \boldsymbol{r}_{k-1}$ are determined. By observing

$$\boldsymbol{A} - \sum_{i=1}^{k-1} \boldsymbol{q}_i \boldsymbol{r}_i^T = \sum_{i=k}^{n} \boldsymbol{q}_i \boldsymbol{r}_i^T = [\underbrace{\boldsymbol{0}, \cdots, \boldsymbol{0}}_{k-1 \text{ zeros}}, r_{kk} \boldsymbol{q}_k, \cdots], \tag{21}$$

we can obtain $\boldsymbol{q}_k$ and $\boldsymbol{r}_k$ via

$$\boldsymbol{y}_k = \left(\boldsymbol{A} - \sum_{i=1}^{k-1} \boldsymbol{q}_i \boldsymbol{r}_i^T \right) \boldsymbol{e}_k, \tag{22a}$$

$$\boldsymbol{q}_k = \boldsymbol{y}_k / \|\boldsymbol{y}_k\|_2, \tag{22b}$$

$$\boldsymbol{r}_i^T = \boldsymbol{q}_k^T \left(\boldsymbol{A} - \sum_{i=1}^{k-1} \boldsymbol{q}_i \boldsymbol{r}_i^T \right). \tag{22c}$$

Table of Contents

Givens Rotations

Consider yet another way for QR; again assume the real-valued case.

Example 2

Given $\boldsymbol{x} \in \mathbb{R}^2$, consider

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \underbrace{\begin{bmatrix} c & s \\ -s & c \end{bmatrix}}_{=\boldsymbol{J}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} cx_1 + sx_2 \\ -sx_1 + cx_2 \end{bmatrix}, \tag{23}$$

where $c \triangleq \cos(\theta)$, $s \triangleq \sin(\theta)$, for some θ. It can be verified that

1) $\boldsymbol{J}$ is orthogonal.
2) $y_2 = 0$ if $\theta = \tan^{-1}(x_2/x_1)$, or if

$$c = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}, \quad s = \frac{x_2}{\sqrt{x_1^2 + x_2^2}}. \tag{24}$$

Givens Rotations (cont'd)

Givens rotations:

$$\boldsymbol{J}(i,k,\theta) \triangleq \begin{bmatrix} \boldsymbol{I} & & & & \\ & c & & s & \\ & & \boldsymbol{I} & & \\ & -s & & c & \\ & & & & \boldsymbol{I} \end{bmatrix}, \tag{25}$$

where $c \triangleq \cos(\theta)$, $s \triangleq \sin(\theta)$; c lies in the $(k,k)^{\rm th}$ and $(i,i)^{\rm th}$ entries; $-s$ the $(i,k)^{\rm th}$ entry; s the $(k,i)^{\rm th}$ entry.

1) $\boldsymbol{J}(i,k,\theta)$ is orthogonal;
2) Let $\boldsymbol{y} = \boldsymbol{J}(i,k,\theta)\boldsymbol{x}$. We have

$$y_j = \begin{cases} cx_i + sx_k, & j = i \\ -sx_i + cx_k, & j = k \\ x_j, & j \neq i,k \end{cases}. \tag{26}$$

Also, y_k can be forced to zero by choosing $\theta = \tan^{-1}(x_k/x_i)$.

Givens Rotations (cont'd)

Example 3

Consider a 4×3 matrix.

$$\boldsymbol{A} = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{\boldsymbol{J}_{2,1}} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{\boldsymbol{J}_{3,1}} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{\boldsymbol{J}_{4,1}} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

$$\xrightarrow{\boldsymbol{J}_{3,2}} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{\boldsymbol{J}_{4,2}} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{\boldsymbol{J}_{4,3}} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix} = \boldsymbol{R},$$

where $\boldsymbol{B} \xrightarrow{\boldsymbol{J}} \boldsymbol{C}$ means $\boldsymbol{C} = \boldsymbol{J}\boldsymbol{B}$; $\boldsymbol{J}_{i,k} = \boldsymbol{J}(i, k, \theta)$, with θ that is chosen to zero out the $(i, k)^{\text{th}}$ entry of the matrix transformed by $\boldsymbol{J}_{i,k}$.

Givens Rotations (cont'd)

Givens QR: perform a sequence of Givens rotations to annihilate the lower triangular parts of $\boldsymbol{A}$ to obtain

$$\underbrace{\boldsymbol{J}_{n,n-1} \ldots (\boldsymbol{J}_{n,2} \cdots \boldsymbol{J}_{3,2})(\boldsymbol{J}_{n,1} \cdots \boldsymbol{J}_{2,1})}_{=\boldsymbol{Q}^T} \boldsymbol{A} = \boldsymbol{R}, \tag{27}$$

where $\boldsymbol{R}$ is upper triangular, and $\boldsymbol{Q}$ is orthogonal.

Example: Givens Rotation

1) Let $\boldsymbol{x} = \begin{bmatrix} 4 & 3 \end{bmatrix}^T$.

2) To annihilate second entry we compute cosine and sine

$$c = \frac{x_1}{\sqrt{x_1^2 + x_2^2}} = 0.8 \text{ and } s = \frac{x_2}{\sqrt{x_1^2 + x_2^2}} = 0.6.$$

3) Rotation is then given by

$$\boldsymbol{J} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix}.$$

4) Rotation is then given by

$$\boldsymbol{J}\boldsymbol{x} = \begin{bmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}.$$

Characteristics of Givens QR Factorization

Remark 1

1) *Straightforward implementation of Givens method requires about 50% more work than Householder method, and also requires more storage, since each rotation requires two numbers, c and s, to define it.*

2) *These disadvantages can be overcome, but requires more complicated implementation.*

3) *Givens can be advantageous for computing QR factorization when many entries of matrix are already zero, since those annihilations can then be skipped.*

Interactive example (offline with Java)

Table of Contents

Householder Transformations

Consider an alternative QR via the Householder transformations (to be introduced). Assume the real-valued case for convenience.

Reflection matrix: a matrix of the form

$$\boldsymbol{H} \triangleq \boldsymbol{I} - 2\boldsymbol{P}, \tag{28}$$

where $\boldsymbol{P} \in \mathbb{R}^{m\times m}$ is an orthogonal projector, is called a reflection matrix.

Householder Transformations

Consider an alternative QR via the Householder transformations (to be introduced). Assume the real-valued case for convenience.

Reflection matrix: a matrix of the form

$$\boldsymbol{H} \triangleq \boldsymbol{I} - 2\boldsymbol{P}, \tag{28}$$

where $\boldsymbol{P} \in \mathbb{R}^{m \times m}$ is an orthogonal projector, is called a reflection matrix.

- By letting $\boldsymbol{a} = \boldsymbol{P}\boldsymbol{a} + \boldsymbol{P}_\perp \boldsymbol{a}$, $\boldsymbol{P}_\perp = \boldsymbol{I} - \boldsymbol{P}$, it is seen that

  $$\boldsymbol{H}\boldsymbol{a} = (\boldsymbol{I} - 2\boldsymbol{P})\boldsymbol{a} = -\boldsymbol{P}\boldsymbol{a} + (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{a} = -\boldsymbol{P}\boldsymbol{a} + \boldsymbol{P}_\perp \boldsymbol{a}, \tag{29}$$

  where $\boldsymbol{P}\boldsymbol{a}$ is 'reflected'.

- A reflection matrix is both orthogonal and symmetric:

  $$\boldsymbol{H} = \boldsymbol{H}^T = \boldsymbol{H}^{-1}. \tag{30}$$

Householder Transformations

Consider an alternative QR via the Householder transformations (to be introduced). Assume the real-valued case for convenience.

Reflection matrix: a matrix of the form

$$\boldsymbol{H} \triangleq \boldsymbol{I} - 2\boldsymbol{P}, \tag{28}$$

where $\boldsymbol{P} \in \mathbb{R}^{m\times m}$ is an orthogonal projector, is called a reflection matrix.

- By letting $\boldsymbol{a} = \boldsymbol{P}\boldsymbol{a} + \boldsymbol{P}_\perp \boldsymbol{a}$, $\boldsymbol{P}_\perp = \boldsymbol{I} - \boldsymbol{P}$, it is seen that

$$\boldsymbol{H}\boldsymbol{a} = (\boldsymbol{I} - 2\boldsymbol{P})\boldsymbol{a} = -\boldsymbol{P}\boldsymbol{a} + (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{a} = -\boldsymbol{P}\boldsymbol{a} + \boldsymbol{P}_\perp \boldsymbol{a}, \tag{29}$$

  where $\boldsymbol{P}\boldsymbol{a}$ is 'reflected'.

- A reflection matrix is both orthogonal and symmetric:

$$\boldsymbol{H} = \boldsymbol{H}^T = \boldsymbol{H}^{-1}. \tag{30}$$

  Proof:

$$\boldsymbol{H}^T\boldsymbol{H} = \boldsymbol{I} - 2\boldsymbol{P} - 2\underbrace{\boldsymbol{P}^T}_{=\boldsymbol{P}} + 4\underbrace{\boldsymbol{P}\boldsymbol{P}}_{=\boldsymbol{P}} = \boldsymbol{I}. \tag{31}$$

Householder Transformations (cont'd)

Question 4

Given $\boldsymbol{a} \in \mathbb{R}^m$, *find a reflection matrix* $\boldsymbol{H} \in \mathbb{R}^{m \times m}$ *such that*

$$\boldsymbol{H}\boldsymbol{a} = \beta \boldsymbol{e}_1, \text{ for some } \beta \in \mathbb{R}. \tag{32}$$

Householder Transformations (cont'd)

Question 4

Given $\boldsymbol{a} \in \mathbb{R}^m$, *find a reflection matrix* $\boldsymbol{H} \in \mathbb{R}^{m\times m}$ *such that*

$$\boldsymbol{H}\boldsymbol{a} = \beta \boldsymbol{e}_1, \text{ for some } \beta \in \mathbb{R}. \tag{32}$$

Householder transformations: let

$$\boldsymbol{H} \triangleq \boldsymbol{I} - \frac{2}{\|\boldsymbol{v}\|_2^2}\boldsymbol{v}\boldsymbol{v}^T, \tag{33}$$

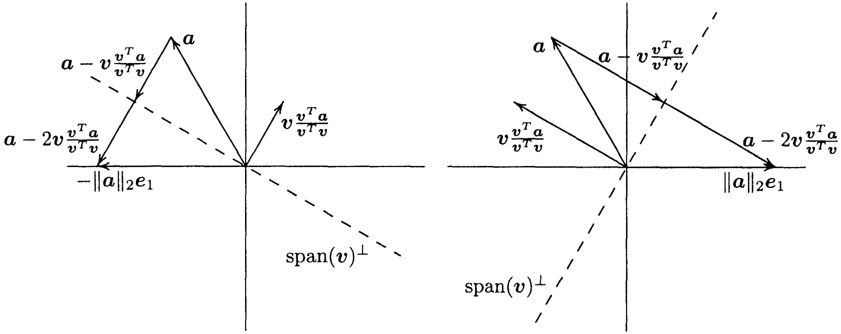
for some $\boldsymbol{v} \in \mathbb{R}^m$ (compared with (28), $\boldsymbol{P} = \frac{\boldsymbol{v}\boldsymbol{v}^T}{\|\boldsymbol{v}\|_2^2}$ is indeed an orthogonal projector onto $\mathbb{R}(\boldsymbol{v})$). It can be verified that if

$$\boldsymbol{v} = \boldsymbol{a} + \operatorname{sign}(a_1)\|\boldsymbol{a}\|_2\,\boldsymbol{e}_1, \tag{34}$$

where $\operatorname{sign}(a_1)$ was introduced to avoid cancellation, then, we obtain

$$\boldsymbol{H}\boldsymbol{a} = -\operatorname{sign}(a_1)\|\boldsymbol{a}\|_2\,\boldsymbol{e}_1. \tag{35}$$

Householder Transformations (cont'd)



$\boldsymbol{a}$
$\boldsymbol{a} - \boldsymbol{v}\frac{\boldsymbol{v}^T\boldsymbol{a}}{\boldsymbol{v}^T\boldsymbol{v}}$
$\boldsymbol{a} - 2\boldsymbol{v}\frac{\boldsymbol{v}^T\boldsymbol{a}}{\boldsymbol{v}^T\boldsymbol{v}}$
$\boldsymbol{v}\frac{\boldsymbol{v}^T\boldsymbol{a}}{\boldsymbol{v}^T\boldsymbol{v}}$
$-\|\boldsymbol{a}\|_2\boldsymbol{e}_1$
$\mathrm{span}(\boldsymbol{v})^\perp$
$\boldsymbol{a}$
$\boldsymbol{a} - \boldsymbol{v}\frac{\boldsymbol{v}^T\boldsymbol{a}}{\boldsymbol{v}^T\boldsymbol{v}}$
$\boldsymbol{v}\frac{\boldsymbol{v}^T\boldsymbol{a}}{\boldsymbol{v}^T\boldsymbol{v}}$
$\boldsymbol{a} - 2\boldsymbol{v}\frac{\boldsymbol{v}^T\boldsymbol{a}}{\boldsymbol{v}^T\boldsymbol{v}}$
$\|\boldsymbol{a}\|_2\boldsymbol{e}_1$
$\mathrm{span}(\boldsymbol{v})^\perp$

Figure 2: Geometric interpretation of Householder transformation as reflection.

Example: Householder Transformation

1) If $\boldsymbol{a} = \begin{bmatrix} 2 & 1 & 2 \end{bmatrix}^T$, then we take

$$\boldsymbol{v} = \boldsymbol{a} + \operatorname{sign}(a_1)\|\boldsymbol{a}\|_2\, \boldsymbol{e}_1 = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} + \|\boldsymbol{a}\|_2 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix}.$$

2) To confirm that transformation works,

$$\boldsymbol{H}\boldsymbol{a} = \boldsymbol{a} - 2\frac{\boldsymbol{v}^T\boldsymbol{a}}{\|\boldsymbol{v}\|^2}\boldsymbol{v} = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} - 2 \times \frac{15}{30} \times \begin{bmatrix} 5 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} -3 \\ 0 \\ 0 \end{bmatrix} = -\operatorname{sign}(a_1)\|\boldsymbol{a}\|_2\, \boldsymbol{e}_1.$$

Interactive example

Householder QR Factorization

Stage 1: let $\boldsymbol{H}_1 \in \mathbb{R}^{m\times m}$ be the Householder transformation w.r.t. $\boldsymbol{a}_1$, i.e., the 1^{st} column of $\boldsymbol{A}$. Then,

$$\boldsymbol{A}^{(1)} = \boldsymbol{H}_1\boldsymbol{A} = \begin{bmatrix} \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}. \tag{36}$$

Householder QR Factorization

Stage 1: let $\boldsymbol{H}_1 \in \mathbb{R}^{m\times m}$ be the Householder transformation w.r.t. $\boldsymbol{a}_1$, i.e., the $1^{\rm st}$ column of $\boldsymbol{A}$. Then,

$$\boldsymbol{A}^{(1)} = \boldsymbol{H}_1\boldsymbol{A} = \begin{bmatrix} \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \cdots & \times \end{bmatrix}. \tag{36}$$

Stage 2: let $\tilde{\boldsymbol{H}}_2 \in \mathbb{R}^{(m-1)\times(m-1)}$ be the Householder transformation w.r.t. the $1^{\rm st}$ column of $\boldsymbol{A}^{(1)}_{2:m,\,2:n}$. Then,

$$\boldsymbol{A}^{(2)} = \underbrace{\begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{H}}_2 \end{bmatrix}}_{=\boldsymbol{H}_2} \boldsymbol{H}_1\boldsymbol{A} = \begin{bmatrix} \times & \times \\ \mathbf{0} & \tilde{\boldsymbol{H}}_2\boldsymbol{A}^{(1)}_{2:m\,2:n} \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \times & \cdots & \times \end{bmatrix}. \tag{37}$$

The same idea applies to stage k, $1 \le k \le n-1$.

Householder QR Factorization (cont'd)

Householder QR procedure: for $k = 1, \cdots, n-1$, compute

$$\boldsymbol{A}^{(k)} = \boldsymbol{H}_k \boldsymbol{A}^{(k-1)}, \tag{38}$$

where

$$\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{I}_{k-1} & \mathbf{0} \\ \mathbf{0} & \tilde{\boldsymbol{H}}_k \end{bmatrix}, \tag{39}$$

in which $\boldsymbol{I}_k$ is the $k \times k$ identity matrix, and $\tilde{\boldsymbol{H}}_k \in \mathbb{R}^{(m-k+1)\times(n-k+1)}$ is the Householder transformation w.r.t $\boldsymbol{A}^{(k-1)}_{k:m,\, k:n}$.

Householder QR Factorization (cont'd)

Householder QR procedure: for $k = 1, \cdots, n-1$, compute

$$\boldsymbol{A}^{(k)} = \boldsymbol{H}_k \boldsymbol{A}^{(k-1)}, \tag{38}$$

where

$$\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{I}_{k-1} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{H}}_k \end{bmatrix}, \tag{39}$$

in which $\boldsymbol{I}_k$ is the $k \times k$ identity matrix, and $\tilde{\boldsymbol{H}}_k \in \mathbb{R}^{(m-k+1)\times(n-k+1)}$ is the Householder transformation w.r.t $\boldsymbol{A}^{(k-1)}_{k:m,\,k:n}$.

The above procedure results in

$$\boldsymbol{A}^{(n-1)} = \boldsymbol{H}_{n-1} \cdots \boldsymbol{H}_2 \boldsymbol{H}_1 \boldsymbol{A} = \boldsymbol{R}. \tag{40}$$

- Eq. (40) implies $\boldsymbol{Q} = \boldsymbol{H}_1^T \boldsymbol{H}_2^T \cdots \boldsymbol{H}_{n-1}^T$;
- The above QR procedure already implies the existence of full QR; also, $\boldsymbol{A}$ can be arbitrary, rather than being of full column rank as in Gram-Schmidt.

Example: Householder QR Factorization

1) For polynomial data-fitting example given previously, with

$$\boldsymbol{A} = \begin{bmatrix} 1 & -1.0 & 1.0 \\ 1 & -0.5 & 0.25 \\ 1 & 0.0 & 0.0 \\ 1 & 0.5 & 0.25 \\ 1 & 1.0 & 1.0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1.0 \\ 0.5 \\ 0.0 \\ 0.5 \\ 2.0 \end{bmatrix}.$$

2) Householder vector $\boldsymbol{v}_1$ for annihilating subdiagonal entries of first column of $\boldsymbol{A}$ is

$$\boldsymbol{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} -2.236 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 3.236 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Example (cont'd)

3) Applying resulting Householder transformation $\boldsymbol{H}_1$ yields transformed matrix and right-hand side

$$\boldsymbol{H}_1\boldsymbol{A} = \begin{bmatrix} -2.236 & 0 & -1.118 \\ 0 & -0.191 & -0.405 \\ 0 & 0.309 & -0.655 \\ 0 & 0.809 & -0.405 \\ 0 & 1.309 & 0.345 \end{bmatrix}, \quad \boldsymbol{H}_1\boldsymbol{b} = \begin{bmatrix} -1.789 \\ -0.362 \\ -0.862 \\ -0.362 \\ 1.138 \end{bmatrix}.$$

4) Householder vector $\boldsymbol{v}_2$ for annihilating subdiagonal entries of second column of $\boldsymbol{H}_1\boldsymbol{A}$ is

$$\boldsymbol{v}_2 = \begin{bmatrix} 0 \\ -0.191 \\ 0.309 \\ 0.809 \\ 1.309 \end{bmatrix} - \begin{bmatrix} 0 \\ 1.581 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -1.772 \\ 0.309 \\ 0.809 \\ 1.309 \end{bmatrix}.$$

Example (cont'd)

5) Applying resulting Householder transformation $\boldsymbol{H}_2$ yields

$$\boldsymbol{H}_2\boldsymbol{H}_1\boldsymbol{A} = \begin{bmatrix} -2.236 & 0 & -1.118 \\ 0 & 1.581 & 0 \\ 0 & 0 & -0.725 \\ 0 & 0 & -0.589 \\ 0 & 0 & 0.047 \end{bmatrix}, \quad \boldsymbol{H}_2\boldsymbol{H}_1\boldsymbol{b} = \begin{bmatrix} -1.789 \\ 0.632 \\ -1.035 \\ 0.816 \\ 0.404 \end{bmatrix}.$$

6) Householder vector $\boldsymbol{v}_3$ for annihilating subdiagonal entries of third column of $\boldsymbol{H}_2\boldsymbol{H}_1\boldsymbol{A}$ is

$$\boldsymbol{v}_3 = \begin{bmatrix} 0 \\ 0 \\ -0.725 \\ -0.589 \\ 0.047 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0.935 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1.660 \\ -0.589 \\ 0.047 \end{bmatrix}.$$

Example (cont'd)

7) Applying resulting Householder transformation $\boldsymbol{H}_3$ yields

$$\boldsymbol{H}_3\boldsymbol{H}_2\boldsymbol{H}_1\boldsymbol{A} = \begin{bmatrix} -2.236 & 0 & -1.118 \\ 0 & 1.581 & 0 \\ 0 & 0 & 0.935 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{H}_3\boldsymbol{H}_2\boldsymbol{H}_1\boldsymbol{b} = \begin{bmatrix} -1.789 \\ 0.632 \\ 1.336 \\ 0.026 \\ 0.337 \end{bmatrix}.$$

8) Now solve upper triangular system $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{c}_1$ by back-substitution to obtain $\boldsymbol{x} = \begin{bmatrix} 0.086 & 0.400 & 1.429 \end{bmatrix}^T$.

Interactive example

Further Study: Hilbert Transform

Question: Householder transform is able to change the amplitudes of elements of a vector. But how to change their phases?

Further Study: Hilbert Transform

Question: Householder transform is able to change the amplitudes of elements of a vector. But how to change their phases?

$$\hat{x}(t) = \mathcal{H}\left\{x(t)\right\} = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau}\, \mathrm{d}\tau \tag{41}$$

$$x(t) = \mathcal{H}^{-1}\left\{\hat{x}(t)\right\} = -\frac{1}{\pi}\int_{-\infty}^{\infty} \frac{\hat{x}(\tau)}{t-\tau}\, \mathrm{d}\tau \tag{42}$$

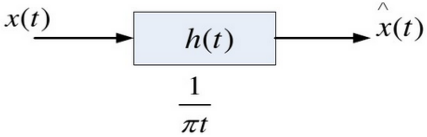


Figure 3: The Hilbert transform in time domain, which is in essence a $\pi/2$ phase shifter in frequency domain.

Further Study: Hilbert Transform

Question: Householder transform is able to change the amplitudes of elements of a vector. But how to change their phases?

$$\hat{x}(t) = \mathcal{H}\left\{x(t)\right\} = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{x(\tau)}{t-\tau}\,\mathrm{d}\tau \tag{41}$$

$$x(t) = \mathcal{H}^{-1}\left\{\hat{x}(t)\right\} = -\frac{1}{\pi}\int_{-\infty}^{\infty}\frac{\hat{x}(\tau)}{t-\tau}\,\mathrm{d}\tau \tag{42}$$

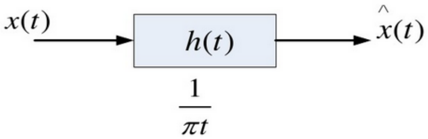


Figure 3: The Hilbert transform in time domain, which is in essence a $\pi/2$ phase shifter in frequency domain.

$$H(\omega) = \mathcal{F}\left\{h(t)\right\} = \begin{cases} -j, & \text{if } \omega \geq 0; \\ +j, & \text{if } \omega < 0. \end{cases} \tag{43}$$

An **application** in signal processing is to construct analytical signals.

Table of Contents

Table of Contents

Power Iteration

1) Simplest method for computing one eigenvalue-eigenvector pair is power iteration, which repeatedly multiplies matrix times initial starting vector.

2) Assume $\boldsymbol{A}$ has unique eigenvalue of maximum modulus, say λ_1, with corresponding eigenvector $\boldsymbol{v}_1$.

3) Then, starting from nonzero vector $\boldsymbol{x}_0$, iteration scheme

$$\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{x}_{k-1} \tag{44}$$

converges to multiple of eigenvector $\boldsymbol{v}_1$ corresponding to dominant eigenvalue λ_1.

Convergence of Power Iteration

1) To see why power iteration converges to dominant eigenvector, express starting vector $\boldsymbol{x}_0$ as linear combination

$$\boldsymbol{x}_0 = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i, \tag{45}$$

where $\boldsymbol{v}_i$ are eigenvectors of $\boldsymbol{A}$.

2) Then

$$\begin{aligned}\boldsymbol{x}_k &= \boldsymbol{A}\boldsymbol{x}_{k-1} = \boldsymbol{A}^2\boldsymbol{x}_{k-2} = \cdots = \boldsymbol{A}^k\boldsymbol{x}_0 \\ &= \sum_{i=1}^{n} \lambda_i^k \alpha_i \boldsymbol{v}_i = \lambda_1^k \left(\alpha_1 \boldsymbol{v}_1 + \sum_{i=2}^{n} (\lambda_i/\lambda_1)^k \alpha_i \boldsymbol{v}_i\right) \\ &= \lambda_1^k \alpha_1 \boldsymbol{v}_1, \text{ as } k \to \infty.\end{aligned} \tag{46}$$

3) Since $|\lambda_i/\lambda_1| < 1$ for $i > 1$, successively higher powers go to zero, leaving only component corresponding to $\boldsymbol{v}_1$.

Example: Power Iteration

1) Ratio of values of given component of $\boldsymbol{x}_k$ from one iteration to next converges to dominant eigenvalue λ_1.

2) For example, if $\boldsymbol{A} = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$ and $\boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, we obtain

k	$\boldsymbol{x}_k^T$		ratio
0	0.0	1.0	
1	0.5	1.5	1.500
2	1.5	2.5	1.667
3	3.5	4.5	1.800
4	7.5	8.5	1.889
5	15.5	16.5	1.941
6	31.5	32.5	1.970
7	63.5	64.5	1.985
8	127.5	128.5	1.992

3) Ratio is converging to dominant eigenvalue, which is 2.

Limitations of Power Iteration

Power iteration can fail for various reasons:

1) Starting vector may have no component in dominant eigenvector $\boldsymbol{v}_1$, i.e., $\alpha_1 = 0$. Nevertheless, this problem does not occur in practice because rounding error usually introduces such component in any case.

2) There may be more than one eigenvalue having same (maximum) modulus, in which case iteration may converge to linear combination of corresponding eigenvectors.

3) for real matrix and starting vector, iteration can never converge to complex vector.

Normalized Power Iteration

1) Geometric growth of components at each iteration risks eventual overflow (or underflow if $\lambda_1 < 1$).

2) Approximate eigenvector should be normalized at each iteration, say, by requiring its largest component to be unity in modulus, giving iteration scheme

$$\boldsymbol{y}_k = \boldsymbol{A}\boldsymbol{x}_{k-1}, \tag{47}$$

$$\boldsymbol{x}_k = \boldsymbol{y}_k / \|\boldsymbol{y}_k\|_\infty. \tag{48}$$

3) With normalization, $\|\boldsymbol{y}_k\|_\infty \to |\lambda_1|$ and $\boldsymbol{x}_k \to \boldsymbol{v}_1 / \|\boldsymbol{v}_1\|_\infty$.

Example: Normalized Power Iteration

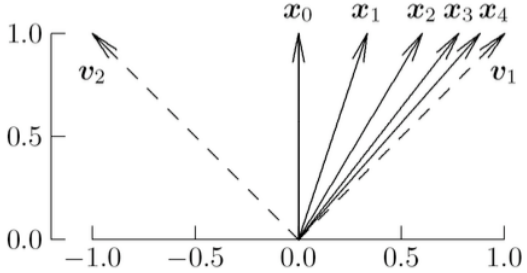
- Repeating previous example with normalized scheme,

k	$\boldsymbol{x}_k^T$		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	0.333	1.0	1.500
2	0.600	1.0	1.667
3	0.778	1.0	1.800
4	0.882	1.0	1.889
5	0.939	1.0	1.941
6	0.969	1.0	1.970
7	0.984	1.0	1.985
8	0.992	1.0	1.992

Interactive example

Geometric Interpretation

1) Behavior of power iteration depicted geometrically:



2) Initial vector $\boldsymbol{x}_0 = \boldsymbol{v}_1 + \boldsymbol{v}_2$ contains equal components in eigenvectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ (dashed arrows).
3) Repeated multiplication by $\boldsymbol{A}$ causes component in $\boldsymbol{v}_1$ (corresponding to larger eigenvalue, 2) to dominate, so sequence of vectors $\boldsymbol{x}_k$ converges to $\boldsymbol{v}_1$.

Power Iteration with Shift

1) Convergence rate of power iteration depends on ratio $|\lambda_2/\lambda_1|$, where λ_2 is eigenvalue having second largest modulus.
2) May be possible to choose shift, $\boldsymbol{A} - \sigma \boldsymbol{I}$ such that

$$\left|\frac{\lambda_2 - \sigma}{\lambda_1 - \sigma}\right| < \left|\frac{\lambda_2}{\lambda_1}\right|, \tag{49}$$

   so convergence is accelerated.
3) Shift must then be added to result to obtain eigenvalue of original matrix.
4) In earlier example, for instance, if we pick shift of $\sigma = 1$, (which is equal to other eigenvalue) then ratio becomes zero and method converges in one iteration.
5) In general, we would not be able to make such fortuitous choice, but shifts can still be extremely useful in some contexts, as we will see later.

Inverse Iteration

1) If smallest eigenvalue of matrix required rather than largest, can make use of fact that eigenvalues of $\boldsymbol{A}^{-1}$ are reciprocals of those of $\boldsymbol{A}$, so smallest eigenvalue of $\boldsymbol{A}$ is reciprocal of largest eigenvalue of $\boldsymbol{A}^{-1}$.
2) This leads to inverse iteration scheme

$$\boldsymbol{A}\boldsymbol{y}_k = \boldsymbol{x}_{k-1}, \tag{50}$$

$$\boldsymbol{x}_k = \boldsymbol{y}_k / \|\boldsymbol{y}_k\|_\infty, \tag{51}$$

which is equivalent to power iteration applied to $\boldsymbol{A}^{-1}$.
3) Inverse of $\boldsymbol{A}$ not computed explicitly, but factorization of $\boldsymbol{A}$ used to solve system of linear equations at each iteration.
4) Inverse iteration converges to eigenvector corresponding to smallest eigenvalue of $\boldsymbol{A}$.
5) Eigenvalue obtained is dominant eigenvalue of $\boldsymbol{A}^{-1}$, and hence its reciprocal is smallest eigenvalue of $\boldsymbol{A}$ in modulus.

Example: Inverse Iteration

- Applying inverse iteration to previous example to compute smallest eigenvalue yields sequence

k	$\boldsymbol{x}_k^T$		$\|\boldsymbol{y}_k\|_\infty$
0	0.000	1.0	
1	-0.333	1.0	0.750
2	-0.600	1.0	0.833
3	-0.778	1.0	0.900
4	-0.882	1.0	0.944
5	-0.939	1.0	0.971
6	-0.969	1.0	0.985

which is indeed converging to 1 (which is its own reciprocal in this case).

Interactive example

Inverse Iteration with Shift

1) As before, shifting strategy, working with $\boldsymbol{A} - \sigma\boldsymbol{I}$ for some scalar σ, can greatly improve convergence.

2) Inverse iteration is particularly useful for computing eigenvector corresponding to approximate eigenvalue, since it converges rapidly when applied to shifted matrix $\boldsymbol{A} - \lambda\boldsymbol{I}$, where λ is approximate eigenvalue.

3) Inverse iteration is also useful for computing eigenvalue closest to given value β, since if β is used as shift, then desired eigenvalue corresponds to smallest eigenvalue of shifted matrix.

Rayleigh Quotient

1) Given approximate eigenvector $\boldsymbol{x}$ for real matrix $\boldsymbol{A}$, determining best estimate for corresponding eigenvalue λ can be considered as $n \times 1$ linear least squares approximation problem

$$\boldsymbol{x}\lambda \cong \boldsymbol{A}\boldsymbol{x}. \tag{52}$$

2) From normal equation $\boldsymbol{x}^T\boldsymbol{x}\lambda = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}$, least squares solution is given by

$$\lambda = \frac{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}. \tag{53}$$

3) This quantity, known as Rayleigh quotient, has many useful properties.

Example: Rayleigh Quotient

1) Rayleigh quotient can accelerate convergence of iterative methods such as power iteration, since Rayleigh quotient $\boldsymbol{x}_k^T \boldsymbol{A} \boldsymbol{x}_k / \boldsymbol{x}_k^T \boldsymbol{x}_k$ gives better approximation to eigenvalue at iteration k than does basic method alone.
2) For previous example using power iteration, value of Rayleigh quotient at each iteration is shown below.

k	$\boldsymbol{x}_k^T$		$\lVert \boldsymbol{y}_k \rVert_\infty$	$\boldsymbol{x}_k^T \boldsymbol{A} \boldsymbol{x} / \boldsymbol{x}_k^T \boldsymbol{x}_k$
0	0.000	1.0		
1	-0.333	1.0	1.500	1.500
2	-0.600	1.0	1.667	1.800
3	-0.778	1.0	1.800	1.941
4	-0.882	1.0	1.889	1.985
5	-0.939	1.0	1.941	1.996
6	-0.969	1.0	1.970	1.999

Rayleigh Quotient Iteration

1) Given approximate eigenvector, Rayleigh quotient yields good estimate for corresponding eigenvalue.
2) Conversely, inverse iteration converges rapidly to eigenvector if approximate eigenvalue is used as shift, with one iteration often sufficing.
3) These two ideas combined in Rayleigh quotient iteration

$$\sigma_k = \boldsymbol{x}_k^T \boldsymbol{A} \boldsymbol{x}_k / \boldsymbol{x}_k^T \boldsymbol{x}_k, \tag{54a}$$

$$(\boldsymbol{A} - \sigma_k \boldsymbol{I}) \boldsymbol{y}_{k+1} = \boldsymbol{x}_k, \tag{54b}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} / \|\boldsymbol{y}_{k+1}\|_\infty, \tag{54c}$$

starting from given nonzero vector $\boldsymbol{x}_0$.

Rayleigh Quotient Iteration (cont'd)

4) Rayleigh quotient iteration is especially effective for symmetric matrices and usually converges very rapidly.
5) Using different shift at each iteration means matrix must be refactored each time to solve linear system, so cost per iteration is high unless matrix has special form that makes factorization easy.
6) Same idea also works for complex matrices, for which transpose is replaced by conjugate transpose, so Rayleigh quotient becomes $\boldsymbol{x}^H \boldsymbol{A}\boldsymbol{x}/\boldsymbol{x}^H\boldsymbol{x}$.
7) Using same matrix as previous examples and randomly chosen starting vector $\boldsymbol{x}_0$, Rayleigh quotient iteration converges in two iterations.

k	$\boldsymbol{x}_k^T$		σ_k
0	0.807	0.397	1.896
1	0.924	1.000	1.998
2	1.000	1.000	2.000

Deflation

1) After eigenvalue λ_1 and corresponding eigenvector $\boldsymbol{x}_1$ have been computed, then additional eigenvalues $\lambda_2, \cdots, \lambda_n$ of $\boldsymbol{A}$ can be computed by deflation, which effectively removes known eigenvalue.

2) Let $\boldsymbol{H}$ be any nonsingular matrix such that $\boldsymbol{H}\boldsymbol{x}_1 = \alpha\boldsymbol{e}_1$, scalar multiple of first column of identity matrix (Householder transformation is a good choice for $\boldsymbol{H}$).

3) Then similarity transformation determined by $\boldsymbol{H}$ transforms $\boldsymbol{A}$ into form

$$\boldsymbol{H}\boldsymbol{A}\boldsymbol{H}^{-1} = \begin{bmatrix} \lambda_1 & \boldsymbol{b}^T \\ \boldsymbol{0} & \boldsymbol{B} \end{bmatrix}, \tag{55}$$

where $\boldsymbol{B}$ is matrix of order $n-1$ having eigenvalues $\lambda_2, \cdots, \lambda_n$.

Deflation (cont'd)

4) Thus, we can work with $\boldsymbol{B}$ to compute next eigenvalue λ_2.

5) Moreover, if $\boldsymbol{y}_2$ is eigenvector of $\boldsymbol{B}$ corresponding to λ_2, then

$$\boldsymbol{x}_2 = \boldsymbol{H}^{-1} \begin{bmatrix} \alpha \\ \boldsymbol{y}_2 \end{bmatrix}, \tag{56}$$

where $\alpha = \frac{\boldsymbol{b}^T \boldsymbol{y}_2}{\lambda_2 - \lambda_1}$, is eigenvector corresponding to λ_2 for original matrix $\boldsymbol{A}$, provided $\lambda_1 \neq \lambda_2$.

6) Process can be repeated to find additional eigenvalues and eigenvectors.

Deflation (cont'd)

7) Alternative approach lets $\boldsymbol{u}_1$ be any vector such that $\boldsymbol{u}_1^T \boldsymbol{x}_1 = \lambda_1$.

8) Then $\boldsymbol{A} - \boldsymbol{x}_1 \boldsymbol{u}_1^T$ has eigenvalues $0, \lambda_2, \cdots, \lambda_n$.

9) Possible choices for $\boldsymbol{u}_1$ include

- $\boldsymbol{u}_1 = \lambda_1 \boldsymbol{x}_1$, if $\boldsymbol{A}$ is symmetric and $\boldsymbol{x}_1$ is normalized so that $\|\boldsymbol{x}_1\|_2 = 1$;
- $\boldsymbol{u}_1 = \lambda_1 \boldsymbol{y}_1$, where $\boldsymbol{y}_1$ is corresponding left eigenvector (i.e., $\boldsymbol{A}^T \boldsymbol{y}_1 = \lambda_1 \boldsymbol{y}_1$) normalized so that $\boldsymbol{y}_1^T \boldsymbol{x}_1 = 1$;
- $\boldsymbol{u}_1 = \boldsymbol{A}^T \boldsymbol{e}_k$, if $\boldsymbol{x}_1$ is normalized so that $\|\boldsymbol{x}_1\|_\infty = 1$ and k^{th} component of $\boldsymbol{x}_1$ is 1.

Table of Contents

Simultaneous Iteration

1) Simplest method for computing many eigenvalue-eigenvector pairs is simultaneous iteration, which repeatedly multiplies matrix times matrix of initial starting vectors.

2) Starting from $n \times p$ matrix $\boldsymbol{X}_0$ of rank p, iteration scheme is

$$\boldsymbol{X}_k = \boldsymbol{A}\boldsymbol{X}_{k-1}. \tag{57}$$

3) $\operatorname{span}(\boldsymbol{X}_k)$ converges to invariant subspace determined by p largest eigenvalues of $\boldsymbol{A}$, provided $|\lambda_p| > |\lambda_{p+1}|$.

4) Also called subspace iteration.

Orthogonal Iteration

1) As with power iteration, normalization is needed with simultaneous iteration.

2) Each column of $\boldsymbol{X}_k$ converges to dominant eigenvector, so columns of $\boldsymbol{X}_k$ become increasingly ill-conditioned basis for span$(\boldsymbol{X}_k)$.

3) Both issues can be addressed by computing QR factorization at each iteration

$$\hat{\boldsymbol{Q}}_k \boldsymbol{R}_k = \boldsymbol{X}_{k-1} \tag{58a}$$

$$\boldsymbol{X}_k = \boldsymbol{A}\hat{\boldsymbol{Q}}_k \tag{58b}$$

where $\hat{\boldsymbol{Q}}_k \boldsymbol{R}_k$ is reduced QR factorization of $\boldsymbol{X}_{k-1}$.

4) This orthogonal iteration converges to block triangular form, and leading block is triangular if moduli of consecutive eigenvalues are distinct.

QR Iteration (proposed by Heinz Rutishauser in 1958 and refined by J. F. G. Francis of Ferranti Ltd., London in 1961-1962)

1) For $p = n$ and $\boldsymbol{X}_0 = \boldsymbol{I}$, matrices

$$\boldsymbol{A}_k = \hat{\boldsymbol{Q}}_k^H \boldsymbol{A} \hat{\boldsymbol{Q}}_k \tag{59}$$

generated by orthogonal iteration converge to triangular or block triangular form, yielding all eigenvalues of $\boldsymbol{A}$.

2) QR iteration computes successive matrices $\boldsymbol{A}_k$ without forming above product explicitly.

3) Starting with $\boldsymbol{A}_0 = \boldsymbol{A}$, at iteration k compute QR factorization

$$\boldsymbol{Q}_k \boldsymbol{R}_k = \boldsymbol{A}_{k-1} \tag{60}$$

and form reverse product

$$\boldsymbol{A}_k = \boldsymbol{R}_k \boldsymbol{Q}_k. \tag{61}$$

QR Iteration (cont'd)

4) Successive matrices $\boldsymbol{A}_k$ are unitarily similar to each other

$$\boldsymbol{A}_k = \boldsymbol{R}_k \boldsymbol{Q}_k = \boldsymbol{Q}_k^H \boldsymbol{A}_{k-1} \boldsymbol{Q}_k. \tag{62}$$

5) Diagonal entries (or eigenvalues of diagonal blocks) of $\boldsymbol{A}_k$ converge to eigenvalues of $\boldsymbol{A}$.

6) Product of orthogonal matrices $\boldsymbol{Q}_k$ converges to matrix of corresponding eigenvectors.

7) If $\boldsymbol{A}$ is symmetric, then symmetry is preserved by QR iteration, so $\boldsymbol{A}_k$ converge to matrix that is both triangular and symmetric, hence diagonal.

Example: QR Iteration

1) Let $\boldsymbol{A}_0 = \begin{bmatrix} 7 & 2 \\ 2 & 4 \end{bmatrix}$.
2) Compute QR factorization

$$\boldsymbol{A}_0 = \boldsymbol{Q}_1 \boldsymbol{R}_1 = \begin{bmatrix} 0.962 & -0.275 \\ 0.275 & 0.962 \end{bmatrix} \begin{bmatrix} 7.28 & 3.02 \\ 0 & 3.30 \end{bmatrix},$$

   and form reverse product

$$\boldsymbol{A}_1 = \boldsymbol{R}_1 \boldsymbol{Q}_1 = \begin{bmatrix} 7.83 & 0.906 \\ 0.906 & 3.17 \end{bmatrix}.$$

3) Off-diagonal entries are now smaller, and diagonal entries closer to eigenvalues, 8 and 3.
4) Process continues until matrix is within tolerance of being diagonal, and diagonal entries then closely approximate eigenvalues.

QR Iteration with Shifts

1) Convergence rate of QR iteration can be accelerated by incorporating shifts

$$\begin{aligned} \boldsymbol{Q}_k \boldsymbol{R}_k &= \boldsymbol{A}_{k-1} - \sigma_k \boldsymbol{I}, \\ \boldsymbol{A}_k &= \boldsymbol{R}_k \boldsymbol{Q}_k + \sigma_k \boldsymbol{I}, \end{aligned}$$

where σ_k is rough approximation to eigenvalue.

2) Good shift can be determined by computing eigenvalues of 2×2 submatrix in lower right corner of matrix.

Example: QR Iteration with Shifts

1) Repeat previous example, but with shift of $\sigma_1 = 4$, which is lower right corner entry of matrix.

2) We compute QR factorization

$$\boldsymbol{A}_0 - \sigma_1 \boldsymbol{I} = \boldsymbol{Q}_1 \boldsymbol{R}_1 = \begin{bmatrix} 0.832 & 0.555 \\ 0.555 & -0.832 \end{bmatrix} \begin{bmatrix} 3.61 & 1.66 \\ 0 & 1.11 \end{bmatrix},$$

and form reverse product, adding back shift to obtain

$$\boldsymbol{A}_1 = \boldsymbol{R}_1 \boldsymbol{Q}_1 + \sigma_1 \boldsymbol{I} = \begin{bmatrix} 7.92 & 0.615 \\ 0.615 & 3.08 \end{bmatrix}.$$

3) After one iteration, off-diagonal entries smaller compared with unshifted algorithm, and eigenvalues closer approximations to eigenvalues.

Interactive example

Preliminary Reduction

1) Efficiency of QR iteration can be enhanced by first transforming matrix as close to triangular form as possible before beginning iterations.

2) Hessenberg matrix is triangular except for one additional nonzero diagonal immediately adjacent to main diagonal.

3) Any matrix can be reduced to Hessenberg form in finite number of steps by orthogonal similarity transformation, for example using Householder transformations.

4) Symmetric Hessenberg matrix is tridiagonal.

5) Hessenberg or tridiagonal form is preserved during successive QR iterations.

Preliminary Reduction (cont'd)

Advantages of initial reduction to upper Hessenberg or tridiagonal form

6) Work per QR iteration is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ for general matrix or $\mathcal{O}(n)$ for symmetric matrix.

7) Fewer QR iterations are required because matrix nearly triangular (or diagonal) already.

8) If any zero entries on first subdiagonal, then matrix is block triangular and problem can be broken into two or more smaller subproblems.

Preliminary Reduction (cont'd)

9) QR iteration is implemented in two-stages:

$$\begin{array}{llcccc} \textbf{Case 1}: & \text{symmetric matrix} & \longrightarrow & \text{tridiagonal} & \longrightarrow & \text{diagonal} \\ & & & \text{or} & & \\ \textbf{Case 2}: & \text{general matrix} & \longrightarrow & \text{Hessenberg} & \longrightarrow & \text{triangular} \end{array}$$

10) Preliminary reduction requires definite number of steps, whereas subsequent iterative stage continues until convergence.

11) In practice only modest number of iterations usually required, so much of work is in preliminary reduction.

12) Cost of accumulating eigenvectors, if needed, dominates total cost.

Cost of QR Iteration

Approximate overall cost of preliminary reduction and QR iteration, counting both additions and multiplications:

1) Symmetric matrices
   - $\frac{4}{3}n^3$ for eigenvalues only;
   - $9n^3$ for eigenvalues and eigenvectors.

2) General matrices
   - $10n^3$ for eigenvalues only;
   - $25n^3$ for eigenvalues and eigenvectors.

Table of Contents

Krylov Subspace Methods (Russian applied mathematician, 1863–1945)

1) Krylov subspace methods reduce matrix to Hessenberg (or tridiagonal) form using only matrix-vector multiplication, developed by Krylov in 1931.

2) For arbitrary starting vector $\boldsymbol{x}_0$, if

$$\boldsymbol{K}_k = \begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{A}\boldsymbol{x}_0 & \cdots & \boldsymbol{A}^{k-1}\boldsymbol{x}_0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_{k-1} \end{bmatrix}, \tag{64}$$

then, for $k = n$, we have

$$\boldsymbol{K}_n^{-1}\boldsymbol{A}\boldsymbol{K}_n = \boldsymbol{C}_n, \tag{65}$$

where $\boldsymbol{C}_n = [\boldsymbol{e}_2, \cdots, \boldsymbol{e}_n, \boldsymbol{a}]$, with $\boldsymbol{a} = \boldsymbol{K}_n^{-1}\boldsymbol{x}_n$, is upper Hessenberg.

3) To obtain better conditioned basis for span($\boldsymbol{K}_n$), compute QR factorization

$$\boldsymbol{Q}_n\boldsymbol{R}_n = \boldsymbol{K}_n, \tag{66}$$

so that

$$\boldsymbol{Q}_n^H\boldsymbol{A}\boldsymbol{Q}_n = \boldsymbol{R}_n\boldsymbol{C}_n\boldsymbol{R}_n^{-1} \equiv \boldsymbol{H}, \tag{67}$$

with $\boldsymbol{H}$ upper Hessenberg.

Krylov Subspace Methods (cont'd)

4) Equating k^{th} columns on each side of equation $\boldsymbol{A}\boldsymbol{Q}_n = \boldsymbol{Q}_n\boldsymbol{H}$ yields recurrence

$$\boldsymbol{A}\boldsymbol{q}_k = h_{1k}\boldsymbol{q}_1 + \cdots + h_{kk}\boldsymbol{q}_k + h_{k+1,k}\boldsymbol{q}_{k+1}, \tag{68}$$

relating $\boldsymbol{q}_{k+1}$ to preceding vectors $\boldsymbol{q}_1, \cdots, \boldsymbol{q}_k$.

5) Premultiplying by $\boldsymbol{q}_j^H$ and using orthonormality,

$$\boldsymbol{q}_j^H\boldsymbol{A}\boldsymbol{q}_k = h_{jk},\ j = 1, \cdots, k. \tag{69}$$

6) These relationships yield Arnoldi iteration, which produces unitary matrix $\boldsymbol{Q}_n$ and upper Hessenberg matrix $\boldsymbol{H}_n$ column by column using only matrix-vector multiplication by $\boldsymbol{A}$ and inner products of vectors.

Arnoldi Iteration (American engineer, 1917–1995)

1) The Arnoldi iteration algorithm, developed by Arnoldi in 1951.

```
1: $\boldsymbol{x}_0$ = arbitrary nonzero starting vector
2: $\boldsymbol{q}_1 = \boldsymbol{x}_0 / \|\boldsymbol{x}_0\|_2$
3: for $k = 1, 2, \cdots$ do
4:     $\boldsymbol{u}_k = \boldsymbol{A}\boldsymbol{q}_k$
5:     for $j = 1$ to $k$ do
6:         $h_{jk} = \boldsymbol{q}_j^H \boldsymbol{u}_k$
7:         $\boldsymbol{u}_k = \boldsymbol{u}_k - h_{jk}\boldsymbol{q}_j$
8:     end for
9:     $h_{k+1,k} = \|\boldsymbol{u}_k\|_2$
10:    if $h_{k+1,k} = 0$ then
11:        stop
12:    end if
13:    $\boldsymbol{q}_{k+1} = \boldsymbol{u}_k / h_{k+1,k}$
14: end for
```

Arnoldi Iteration (cont'd)

2) If
$$\boldsymbol{Q}_k = \begin{bmatrix} \boldsymbol{q}_1 & \cdots & \boldsymbol{q}_k \end{bmatrix}, \tag{70}$$
then
$$\boldsymbol{H}_k = \boldsymbol{Q}_k^H \boldsymbol{A} \boldsymbol{Q}_k. \tag{71}$$
is upper Hessenberg matrix.

3) Eigenvalues of $\boldsymbol{H}_k$, called Ritz values, are approximate eigenvalues of $\boldsymbol{A}$, and Ritz vectors given by $\boldsymbol{Q}_k \boldsymbol{y}$, where $\boldsymbol{y}$ is eigenvector of $\boldsymbol{H}_k$, are corresponding approximate eigenvectors of $\boldsymbol{A}$.

4) Eigenvalues of $\boldsymbol{H}_k$ must be computed by another method, such as QR iteration, but this is easier problem if $k \ll n$.

Arnoldi Iteration (cont'd)

5) Arnoldi iteration fairly expensive in work and storage because each new vector $\boldsymbol{q}_k$ must be orthogonalized against all previous columns of $\boldsymbol{Q}_k$, and all must be stored for that purpose.

6) So Arnoldi process usually restarted periodically with carefully chosen starting vector.

7) Ritz values and vectors produced are often good approximations to eigenvalues and eigenvectors of $\boldsymbol{A}$ after relatively few iterations.

Lanczos Iteration (Hungary, 1893–1974, Einstein's assistant in Berlin in 1928)

1) Work and storage costs drop dramatically if matrix is symmetric or Hermitian, since recurrence then has only three terms and $\boldsymbol{H}_k$ is tridiagonal (so usually denoted $\boldsymbol{T}_k$)

```
1: $\boldsymbol{q}_0 = \boldsymbol{0}$, $\beta_0 = 0$
2: $\boldsymbol{x}_0 =$ arbitrary nonzero starting vector
3: $\boldsymbol{q}_1 = \boldsymbol{x}_0 / \|\boldsymbol{x}_0\|_2$
4: for $k = 1, 2, \cdots$ do
5:     $\boldsymbol{u}_k = \boldsymbol{A}\boldsymbol{q}_k$
6:     $\alpha_k = \boldsymbol{q}_k^H \boldsymbol{u}_k$
7:     $\boldsymbol{u}_k = \boldsymbol{u}_k - \beta_{k-1}\boldsymbol{q}_{k-1} - \alpha_k \boldsymbol{q}_k$
8:     $\beta_k = \|\boldsymbol{u}_k\|_2$
9:     if $\beta_k = 0$ then
10:        stop
11:    end if
12:    $\boldsymbol{q}_{k+1} = \boldsymbol{u}_k / \beta_k$
13: end for
```

Lanczos Iteration (cont'd)

2) α_k and β_k are diagonal and subdiagonal entries of symmetric tridiagonal matrix $\boldsymbol{T}_k$.

3) As with Arnoldi, Lanczos iteration does not produce eigenvalues and eigenvectors directly, but only tridiagonal matrix $\boldsymbol{T}_k$, whose eigenvalues and eigenvectors must be computed by another method to obtain Ritz values and vectors.

4) If $\beta_k = 0$, then algorithm appears to break down, but in that case invariant subspace has already been identified (i.e., Ritz values and vectors are already exact at that point).

Lanczos Iteration (cont'd)

5) In principle, if Lanczos algorithm were run until $k = n$, resulting tridiagonal matrix would be orthogonally similar to $\boldsymbol{A}$.

6) In practice, rounding error causes loss of orthogonality, invalidating this expectation.

7) Problem can be overcome by reorthogonalizing vectors as needed, but expense can be substantial.

8) Alternatively, can ignore problem, in which case algorithm still produces good eigenvalue approximations, but multiple copies of some eigenvalues may be generated.

Lanczos Iteration (cont'd)

9) Great virtue of Arnoldi and Lanczos iterations is their ability to produce good approximations to extreme eigenvalues for $k \ll n$.

10) Moreover, they require only one matrix-vector multiplication by $\boldsymbol{A}$ per step and little auxiliary storage, so are ideally suited to large sparse matrices.

11) If eigenvalues are needed in middle of spectrum, say near σ, then algorithm can be applied to matrix $(\boldsymbol{A} - \sigma \boldsymbol{I})^{-1}$, assuming it is practical to solve systems of form $(\boldsymbol{A} - \sigma \boldsymbol{I})\boldsymbol{x} = \boldsymbol{y}$.

Example: Lanczos Iteration

For a 29×29 symmetric matrix with eigenvalues $1, \cdots, 29$, behavior of Lanczos iteration is shown below.

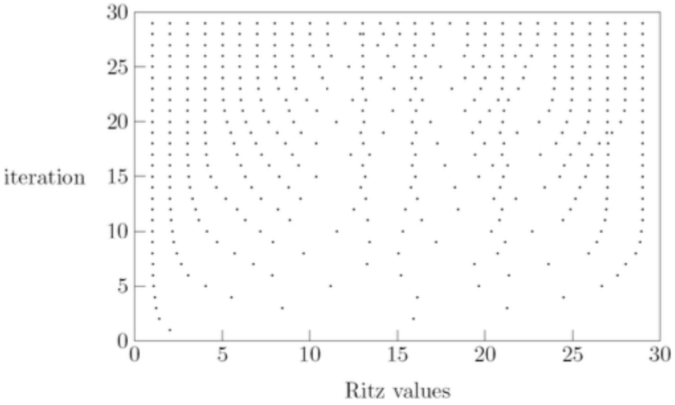


Table of Contents

Jacobi Method

1) One of oldest methods for computing eigenvalues is Jacobi method, which uses similarity transformation based on plane rotations.

2) Sequence of plane rotations chosen to annihilate symmetric pairs of matrix entries, eventually converging to diagonal form.

3) Choice of plane rotation slightly more complicated than in Givens method for QR factorization.

4) To annihilate given off-diagonal pair, choose c and s so that

$$\boldsymbol{J}^H \boldsymbol{A} \boldsymbol{J} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} a & b \\ b & d \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{72}$$

$$= \begin{bmatrix} c^2a - 2csb + s^2d & c^2b + cs(a-d) - s^2b \\ c^2b + cs(a-d) - s^2b & c^2d + 2csb + s^2a \end{bmatrix} \tag{73}$$

is diagonal.

Jacobi Method (cont'd)

5) Transformed matrix diagonal if

$$c^2b + cs(a-d) - s^2b = 0. \tag{74}$$

6) Dividing both sides by c^2b, we obtain

$$1 + \frac{s}{c}\frac{a-d}{b} - \frac{s^2}{c^2} = 0. \tag{75}$$

7) Making substitution $t = s/c$, we get quadratic equation

$$1 + t\,\frac{a-d}{b} - t^2 = 0. \tag{76}$$

for tangent t of angle of rotation, from which we can recover $c = 1/\sqrt{1+t^2}$ and $s = c \cdot t$.

8) Advantageous numerically to use root of smaller magnitude.

Example: Plane Rotation

- Consider 2×2 matrix
$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}.$$
- Quadratic equation for tangent reduces to $t^2 = 1$, so $t = \pm 1$.
- Two roots of same magnitude, so we arbitrarily choose $t = -1$, which yields $c = 1/\sqrt{2}$ and $s = -1/\sqrt{2}$.
- Resulting plane rotation $\boldsymbol{J}$ gives
$$\boldsymbol{J}^T \boldsymbol{A} \boldsymbol{J} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & -1 \end{bmatrix}.$$

Jacobi Method (cont'd)

9) Starting with symmetric matrix $\boldsymbol{A}_0 = \boldsymbol{A}$, each iteration has form

$$\boldsymbol{A}_{k+1} = \boldsymbol{J}_k^T \boldsymbol{A}_k \boldsymbol{J}_k, \tag{77}$$

where $\boldsymbol{J}_k$ is plane rotation chosen to annihilate a symmetric pair of entries in $\boldsymbol{A}_k$.

10) Plane rotations repeatedly applied from both sides in systematic sweeps through matrix until off-diagonal mass of matrix is reduced to within some tolerance of zero.

11) Resulting diagonal matrix orthogonally similar to original matrix, so diagonal entries are eigenvalues, and eigenvectors are given by product of plane rotations.

Jacobi Method (cont'd)

12) Jacobi method is reliable, simple to program, and capable of high accuracy, but converges rather slowly and difficult to generalize beyond symmetric matrices.

13) Except for small problems, more modern methods usually require 5 to 10 times less work than Jacobi.

14) One source of inefficiency is that previously annihilated entries can subsequently become nonzero again, thereby requiring repeated annihilation.

15) Newer methods such as QR iteration preserve zero entries introduced into matrix.

Example: Jacobi Method

1) Let $\boldsymbol{A}_0 = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 2 & 1 \\ 2 & 1 & 1 \end{bmatrix}$.

2) First annihilate (1,3) and (3,1) entries using rotation

$$\boldsymbol{J}_0 = \begin{bmatrix} 0.707 & 0 & -0.707 \\ 0 & 1 & 0 \\ 0.707 & 0 & 0.707 \end{bmatrix}$$

to obtain

$$\boldsymbol{A}_1 = \boldsymbol{J}_0^T \boldsymbol{A}_0 \boldsymbol{J}_0 = \begin{bmatrix} 3 & 0.707 & 0 \\ 0.707 & 2 & 0.707 \\ 0 & 0.707 & -1 \end{bmatrix}.$$

Example (cont'd)

3) Next annihilate (1,2) and (2,1) entries using rotation

$$\boldsymbol{J}_1 = \begin{bmatrix} 0.888 & -0.460 & 0 \\ 0.460 & 0.888 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

to obtain

$$\boldsymbol{A}_2 = \boldsymbol{J}_1^T \boldsymbol{A}_1 \boldsymbol{J}_1 = \begin{bmatrix} 3.366 & 0 & 0.325 \\ 0 & 1.634 & 0.628 \\ 0.325 & 0.628 & -1 \end{bmatrix}.$$

Example (cont'd)

4) Next annihilate (2,3) and (3,2) entries using rotation

$$\boldsymbol{J}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.975 & -0.221 \\ 0 & 0.221 & 0.975 \end{bmatrix}$$

to obtain

$$\boldsymbol{A}_3 = \boldsymbol{J}_2^T \boldsymbol{A}_2 \boldsymbol{J}_2 = \begin{bmatrix} 3.366 & 0.072 & 0.317 \\ 0.072 & 1.776 & 0 \\ 0.317 & 0 & -1.142 \end{bmatrix}.$$

Example (cont'd)

5) Beginning new sweep, again annihilate (1,3) and (3,1) entries using rotation

$$\boldsymbol{J}_3 = \begin{bmatrix} 0.998 & 0 & -0.070 \\ 0 & 1 & 0 \\ 0.070 & 0 & 0.998 \end{bmatrix}$$

to obtain

$$\boldsymbol{A}_4 = \boldsymbol{J}_3^T \boldsymbol{A}_3 \boldsymbol{J}_3 = \begin{bmatrix} 3.388 & 0.072 & 0 \\ 0.072 & 1.776 & -0.005 \\ 0 & -0.005 & -1.164 \end{bmatrix}.$$

Example (cont'd)

6) Process continues until off-diagonal entries reduced to as small as desired.

7) Result is diagonal matrix orthogonally similar to original matrix, with the orthogonal similarity transformation given by product of plane rotations.

Interactive example

Generalized Eigenvalue Problems

1) Generalized eigenvalue problem has form

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}, \tag{78}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are given $n \times n$ matrices.

2) If either $\boldsymbol{A}$ or $\boldsymbol{B}$ is nonsingular, then generalized eigenvalue problem can be converted to standard eigenvalue problem, either

$$(\boldsymbol{B}^{-1}\boldsymbol{A})\boldsymbol{x} = \lambda \boldsymbol{x} \ \text{ or } \ (\boldsymbol{A}^{-1}\boldsymbol{B})\boldsymbol{x} = (1/\lambda)\boldsymbol{x}. \tag{79}$$

3) This is not recommended, since it may cause
   - loss of accuracy due to rounding error;
   - loss of symmetry if $\boldsymbol{A}$ and $\boldsymbol{B}$ are symmetric.

4) Better alternative for generalized eigenvalue problems is QZ algorithm.

Generalized Eigenvalue Problems: QZ Algorithm

1) If $\boldsymbol{A}$ and $\boldsymbol{B}$ are triangular, then eigenvalues are given by $\lambda_i = a_{ii}/b_{ii}$, for $b_{ii} \neq 0$.
2) QZ algorithm reduces $\boldsymbol{A}$ and $\boldsymbol{B}$ simultaneously to upper triangular form by orthogonal transformations.
3) First, $\boldsymbol{B}$ is reduced to upper triangular form by orthogonal transformation from left, which is also applied to $\boldsymbol{A}$.
4) Next, transformed $\boldsymbol{A}$ is reduced to upper Hessenberg form by orthogonal transformation from left, while maintaining triangular form of $\boldsymbol{B}$, which requires additional transformations from right.
5) Finally, analogous to QR iteration, $\boldsymbol{A}$ is reduced to triangular form while still maintaining triangular form of $\boldsymbol{B}$, which again requires transformations from both sides.
6) Eigenvalues can now be determined from mutually triangular form, and eigenvectors can be recovered from products of left and right transformations, denoted by $\boldsymbol{Q}$ and $\boldsymbol{Z}$.

Generalized Eigenvalue Problems: Matlab Application

In Matlab, the built-in function

$$[\boldsymbol{V}, \boldsymbol{D}] = \mathrm{eig}(\boldsymbol{A}, \boldsymbol{B}, \mathrm{algorithm}) \tag{80}$$

solves the generalized eigenvalue problem $\boldsymbol{AV} = \boldsymbol{BVD}$, where $\boldsymbol{B}$ must be the same size as $\boldsymbol{A}$.

- If the algorithm is 'chol', uses the Cholesky factorization of $\boldsymbol{B}$ to compute the generalized eigenvalues.
- The default for algorithm depends on the properties of $\boldsymbol{A}$ and $\boldsymbol{B}$, but is generally 'qz', which uses the QZ algorithm.
- If $\boldsymbol{A}$ is Hermitian and $\boldsymbol{B}$ is Hermitian positive definite, then the default for algorithm is 'chol'.

Computing SVD

1) Singular values of $\boldsymbol{A}$ are nonnegative square roots of eigenvalues of $\boldsymbol{A}^T\boldsymbol{A}$, and columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthonormal eigenvectors of $\boldsymbol{A}\boldsymbol{A}^T$ and $\boldsymbol{A}^T\boldsymbol{A}$, respectively.

2) Algorithms for computing SVD work directly with $\boldsymbol{A}$, without forming $\boldsymbol{A}\boldsymbol{A}^T$ or $\boldsymbol{A}^T\boldsymbol{A}$, to avoid loss of information associated with forming these matrix products explicitly.

3) SVD is usually computed by variant of QR iteration, with $\boldsymbol{A}$ first reduced to bidiagonal form by orthogonal transformations, then remaining off-diagonal entries are annihilated iteratively.

4) SVD can also be computed by variant of Jacobi method, which can be useful on parallel computers or if matrix has special structure.

Thank you
for your attention!

Kai Lu
E-mail: lukai86@mail.sysu.edu.cn
Web: http://seit.sysu.edu.cn/teacher/1801