QR-method lecture 1

SF2524 - Matrix Computations for Large-scale Systems

So far we have in the course learned about...

Methods suitable for large sparse matrices

- Power method
  - Computes largest eigenvalue

So far we have in the course learned about...

Methods suitable for large sparse matrices

- Power method
  - Computes largest eigenvalue
- Inverse iteration
  - Computes eigenvalue closest to a target

So far we have in the course learned about...

Methods suitable for large sparse matrices

- Power method
  - Computes largest eigenvalue
- Inverse iteration
  - Computes eigenvalue closest to a target
- Rayleigh Quotient Iteration
  - Computes one eigenvalue with a starting guess

So far we have in the course learned about...

Methods suitable for large sparse matrices

- Power method
  - Computes largest eigenvalue
- Inverse iteration
  - Computes eigenvalue closest to a target
- Rayleigh Quotient Iteration
  - Computes one eigenvalue with a starting guess
- Arnoldi method for eigenvalue problems
  - Computes extreme eigenvalue

So far we have in the course learned about...

Methods suitable for large sparse matrices

- Power method
  - Computes largest eigenvalue
- Inverse iteration
  - Computes eigenvalue closest to a target
- Rayleigh Quotient Iteration
  - Computes one eigenvalue with a starting guess
- Arnoldi method for eigenvalue problems
  - Computes extreme eigenvalue

Now: QR-method

- Compute all eigenvalues
- Suitable for dense problems
- Small matrices in relation to previous algorithms

Agenda QR-method

1. Decompositions
   - Jordan form
   - Schur decomposition
   - QR-factorization
2. Basic QR-method
3. Improvement 1: Two-phase approach
   - Hessenberg reduction
   - Hessenberg QR-method
4. Improvement 2: Acceleration with shifts
5. Convergence theory

Agenda QR-method

1. Decompositions
   - Jordan form
   - Schur decomposition
   - QR-factorization
2. Basic QR-method
3. Improvement 1: Two-phase approach
   - Hessenberg reduction
   - Hessenberg QR-method
4. Improvement 2: Acceleration with shifts
5. Convergence theory

Reading instructions

Point 1: TB Lecture 24

Points 2-4: Lecture notes "QR-method" on course web page

Point 5: TB Chapter 28

(Extra reading: TB Chapter 25-26, 28-29)

Similarity transformation

Suppose $A \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{m\times m}$ is an invertible matrix. Then

$$A$$

and

$$B = VAV^{-1}$$

have the same eigenvalues.

Similarity transformation

Suppose $A \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{m\times m}$ is an invertible matrix. Then

$$A$$

and

$$B = VAV^{-1}$$

have the same eigenvalues.

Numerical methods based on similarity transformations

- If B is triangular we can read-off the eigenvalues from the diagonal.
- Idea of numerical method: Compute V such that B is triangular.

First idea: compute the Jordan canonical form

Jordan canonical form

Suppose $A \in \mathbb{C}^{m\times m}$. There exists an invertible matrix $V \in \mathbb{C}^{m\times m}$ and a block diagonal matrix such that

$$A = V\Lambda V^{-1}$$

First idea: compute the Jordan canonical form

Jordan canonical form

Suppose $A \in \mathbb{C}^{m\times m}$. There exists an invertible matrix $V \in \mathbb{C}^{m\times m}$ and a block diagonal matrix such that

$$A = V\Lambda V^{-1}$$

where

$$\Lambda = \begin{pmatrix} J_1 & & \\ & \ddots & \\ & & J_k \end{pmatrix},$$

where

$$J_i = \begin{pmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix}, \quad i = 1, \ldots, k$$

Common case: distinct eigenvalues

Suppose $\lambda_i \neq \lambda_j$, $i = 1, \ldots, m$. Then,

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{pmatrix}.$$

Common case: distinct eigenvalues

Suppose $\lambda_i \neq \lambda_j$, $i = 1, \ldots, m$. Then,

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{pmatrix}.$$

Common case: symmetric matrix

Suppose $A = A^T \in \mathbb{R}^{m \times m}$. Then,

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_m \end{pmatrix}.$$

Example - numerical stability of Jordan form

Consider

$$A = \begin{pmatrix} 2 & 1 & \\ & 2 & 1 \\ \varepsilon & & 2 \end{pmatrix}$$

Example - numerical stability of Jordan form

Consider

$$A = \begin{pmatrix} 2 & 1 & \\ & 2 & 1 \\ \varepsilon & & 2 \end{pmatrix}$$

If $\varepsilon = 0$. Then, the Jordan canonical form is

$$\Lambda = \begin{pmatrix} 2 & 1 & \\ & 2 & 1 \\ & & 2 \end{pmatrix}.$$

Example - numerical stability of Jordan form

Consider

$$A = \begin{pmatrix} 2 & 1 & \\ & 2 & 1 \\ \varepsilon & & 2 \end{pmatrix}$$

If $\varepsilon = 0$. Then, the Jordan canonical form is

$$\Lambda = \begin{pmatrix} 2 & 1 & \\ & 2 & 1 \\ & & 2 \end{pmatrix}.$$

If $\varepsilon > 0$. Then, the eigenvalues are distinct and

$$\Lambda = \begin{pmatrix} 2 + O(\varepsilon^{1/3}) & & \\ & 2 + O(\varepsilon^{1/3}) & \\ & & 2 + O(\varepsilon^{1/3}) \end{pmatrix}.$$

Example - numerical stability of Jordan form

Consider

$$A = \begin{pmatrix} 2 & 1 & \\ & 2 & 1 \\ \varepsilon & & 2 \end{pmatrix}$$

If $\varepsilon = 0$. Then, the Jordan canonical form is

$$\Lambda = \begin{pmatrix} 2 & 1 & \\ & 2 & 1 \\ & & 2 \end{pmatrix}.$$

If $\varepsilon > 0$. Then, the eigenvalues are distinct and

$$\Lambda = \begin{pmatrix} 2 + O(\varepsilon^{1/3}) & & \\ & 2 + O(\varepsilon^{1/3}) & \\ & & 2 + O(\varepsilon^{1/3}) \end{pmatrix}.$$

$\Rightarrow$ Not continuous with respect to ε

$\Rightarrow$ The Jordan form is often not numerically stable

Schur decomposition (essentially TB Theorem 24.9)

Suppose $A \in \mathbb{C}^{m \times m}$. There exists an unitary matrix P

$$P^{-1} = P^*$$

and a triangular matrix T such that

$$A = PTP^*.$$

Schur decomposition (essentially TB Theorem 24.9)

Suppose $A \in \mathbb{C}^{m\times m}$. There exists an unitary matrix P

$$P^{-1} = P^*$$

and a triangular matrix T such that

$$A = PTP^*.$$

The Schur decomposition is numerically stable.
Goal with QR-method: Numercally compute a Schur factorization

Outline:

1. Decompositions
   - Jordan form
   - Schur decomposition
   - QR-factorization
2. **Basic QR-method**
3. Improvement 1: Two-phase approach
   - Hessenberg reduction
   - Hessenberg QR-method
4. Improvement 2: Acceleration with shifts
5. Convergence theory

QR-factorization

Suppose $A \in \mathbb{C}^{m \times m}$. There exists a unitary matrix Q and an upper triangular matrix R such that

$$A = QR$$

QR-factorization

Suppose $A \in \mathbb{C}^{m\times m}$. There exists a unitary matrix Q and an upper triangular matrix R such that

$$A = QR$$

Note: Very different from Schur factorization

$$A = QTQ^*$$

- QR-factorization can be computed with a finite number of iterations
- Schur decomposition directly gives us the eigenvalues

Basic QR-method

Didactic simplifying assumption: All eigenvalues are real

Basic QR-method

Didactic simplifying assumption: All eigenvalues are real

Basic QR-method = basic QR-algorithm

Simple basic idea: Let $A_0 = A$ and iterate:

- Compute QR-factorization of $A_k = QR$
- Set $A_{k+1} = RQ$.

Basic QR-method

Didactic simplifying assumption: All eigenvalues are real

Basic QR-method = basic QR-algorithm

Simple basic idea: Let $A_0 = A$ and iterate:

- Compute QR-factorization of $A_k = QR$
- Set $A_{k+1} = RQ$.

Note:

- $A_1 = RQ = Q^* A_0 Q \Rightarrow A_0, A_1, \ldots$ have the same eigenvalues

Basic QR-method

Didactic simplifying assumption: All eigenvalues are real

Basic QR-method = basic QR-algorithm

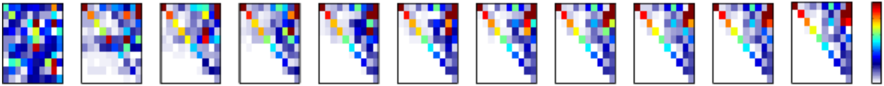
Simple basic idea: Let $A_0 = A$ and iterate:

- Compute QR-factorization of $A_k = QR$
- Set $A_{k+1} = RQ$.

Note:

- $A_1 = RQ = Q^* A_0 Q \Rightarrow A_0, A_1, \ldots$ have the same eigenvalues
- More remarkable: $A_k \rightarrow$ triangular matrix (except special cases)

$A_k \to$ triangular matrix:



$A_k \rightarrow$ triangular matrix:

* Time for matlab demo *

Elegant and robust but not very efficient:

Elegant and robust but not very efficient:

Disadvantages

- Computing a QR-factorization is quite expensive. One iteration of the basic QR-method

$$\mathcal{O}(m^3).$$

Elegant and robust but not very efficient:

Disadvantages

- Computing a QR-factorization is quite expensive. One iteration of the basic QR-method

$$\mathcal{O}(m^3).$$

- The method often requires many iterations.

Improvement demo:

`http://www.youtube.com/watch?v=qmgxzsWWsNc`

Outline:

1. Decompositions
   - Jordan form
   - Schur decomposition
   - QR-factorization
2. Basic QR-method
3. **Improvement 1: Two-phase approach**
   - Hessenberg reduction
   - Hessenberg QR-method
4. Improvement 2: Acceleration with shifts
5. Convergence theory

Improvement 1: Two-phase approach

We will separate the computation into two phases:

$$
\begin{bmatrix}
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times
\end{bmatrix}
\underset{\text{Phase 1}}{\rightarrow}
\begin{bmatrix}
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
 & \times & \times & \times & \times \\
 & & \times & \times & \times \\
 & & & \times & \times
\end{bmatrix}
\underset{\text{Phase 2}}{\rightarrow}
\begin{bmatrix}
\times & \times & \times & \times & \times \\
 & \times & \times & \times & \times \\
 & & \times & \times & \times \\
 & & & \times & \times \\
 & & & & \times
\end{bmatrix}
$$

Improvement 1: Two-phase approach

We will separate the computation into two phases:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \underset{\text{Phase 1}}{\rightarrow} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} \underset{\text{Phase 2}}{\rightarrow} \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}$$

Phases:

- Phase 1: Reduce the matrix to a Hessenberg with similarity transformations (Section 2.2.1 in lecture notes)

Improvement 1: Two-phase approach

We will separate the computation into two phases:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \underset{\text{Phase 1}}{\rightarrow} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix} \underset{\text{Phase 2}}{\rightarrow} \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}$$

Phases:

- Phase 1: Reduce the matrix to a Hessenberg with similarity transformations (Section 2.2.1 in lecture notes)
- Phase 2: Specialize the QR-method to Hessenberg matrices (Section 2.2.2 in lecture notes)

Phase 1: Hessenberg reduction

Idea for Hessenberg reduction

Compute unitary P and Hessenberg matrix H such that

$$A = PHP^*$$

Phase 1: Hessenberg reduction

Idea for Hessenberg reduction

Compute unitary P and Hessenberg matrix H such that

$$A = PHP^*$$

Unlike the Schur factorization, this can be computed with a finite number of operations.

Phase 1: Hessenberg reduction

Idea for Hessenberg reduction

Compute unitary P and Hessenberg matrix H such that

$$A = PHP^*$$

Unlike the Schur factorization, this can be computed with a finite number of operations.

Key method: Householder reflectors

Phase 1: Hessenberg reduction

Definition

A matrix $P \in \mathbb{C}^{m\times m}$ of the form

$$P = I - 2uu^* \quad \text{where } u \in \mathbb{C}^m \text{ and } \|u\| = 1$$

is called a *Householder reflector*.

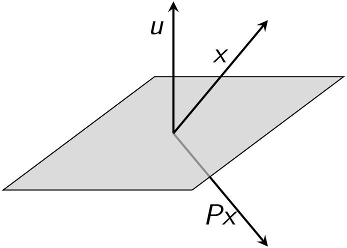
Phase 1: Hessenberg reduction

Definition

A matrix $P \in \mathbb{C}^{m \times m}$ of the form

$$P = I - 2uu^* \quad \text{where } u \in \mathbb{C}^m \text{ and } \|u\| = 1$$

is called a *Householder reflector*.



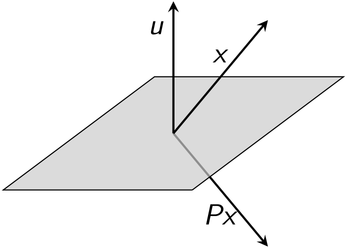
Phase 1: Hessenberg reduction

Definition

A matrix $P \in \mathbb{C}^{m \times m}$ of the form

$$P = I - 2uu^* \quad \text{where } u \in \mathbb{C}^m \text{ and } \|u\| = 1$$

is called a *Householder reflector*.



Properties

- $P^* = P^{-1} = P$

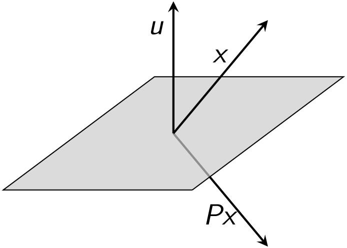
Phase 1: Hessenberg reduction

Definition

A matrix $P \in \mathbb{C}^{m \times m}$ of the form

$$P = I - 2uu^* \quad \text{where } u \in \mathbb{C}^m \text{ and } \|u\| = 1$$

is called a *Householder reflector*.



Properties

- $P^* = P^{-1} = P$
- $Pz = z - 2u(u^*z)$ can be computed with $O(m)$ operations.

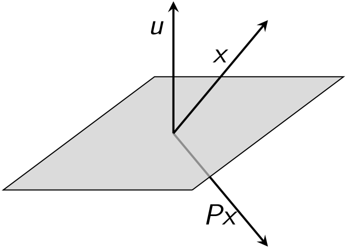
Phase 1: Hessenberg reduction

Definition

A matrix $P \in \mathbb{C}^{m\times m}$ of the form

$$P = I - 2uu^* \quad \text{where } u \in \mathbb{C}^m \text{ and } \|u\| = 1$$

is called a *Householder reflector*.



Properties

- $P^* = P^{-1} = P$
- $Pz = z - 2u(u^*z)$ can be computed with $O(m)$ operations.
- $\dots$

Householder reflectors satisfying $Px = \alpha e_1$

Problem

Given a vector x compute a Householder reflector such that

$$Px = \alpha e_1.$$

Householder reflectors satisfying $Px = \alpha e_1$

Problem

Given a vector x compute a Householder reflector such that

$$Px = \alpha e_1.$$

Solution (Lemma 2.2.3)

Let $\rho = \text{sign}(x_1)$,

$$z := x - \rho\|x\|e_1 = \begin{bmatrix} x_1 - \rho\|x\| \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

and

$$u = z/\|z\|.$$

Householder reflectors satisfying $Px = \alpha e_1$

Problem

Given a vector x compute a Householder reflector such that

$$Px = \alpha e_1.$$

Solution (Lemma 2.2.3)

Let $\rho = \text{sign}(x_1)$,

$$z := x - \rho\|x\|e_1 = \begin{bmatrix} x_1 - \rho\|x\| \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

and

$$u = z/\|z\|.$$

Then, $P = I - 2uu^*$ is a Householder reflector that satisfies $Px = \alpha e_1$.

Householder reflectors satisfying $Px = \alpha e_1$

Problem

Given a vector x compute a Householder reflector such that

$$Px = \alpha e_1.$$

Solution (Lemma 2.2.3)

Let $\rho = \text{sign}(x_1)$,

$$z := x - \rho\|x\|e_1 = \begin{bmatrix} x_1 - \rho\|x\| \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

and

$$u = z/\|z\|.$$

Then, $P = I - 2uu^*$ is a Householder reflector that satisfies $Px = \alpha e_1$.

* Matlab demo showing Householder reflectors *

We will be able to construct $m-2$ householder reflectors that bring the matrix to Hessenberg form.

We will be able to construct $m-2$ householder reflectors that bring the matrix to Hessenberg form.

Elimination for first column

$$P_1 := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & 0^T \\ 0 & I - 2u_1u_1^T \end{bmatrix}.$$

We will be able to construct $m-2$ householder reflectors that bring the matrix to Hessenberg form.

Elimination for first column

$$P_1 := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & 0^T \\ 0 & I - 2u_1 u_1^T \end{bmatrix}.$$

Use Lemma 2.2.1 with $x^T = [a_{21}, \dots, a_{n1}]$ to select u_1 such that

$$P_1 A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}$$

We will be able to construct $m-2$ householder reflectors that bring the matrix to Hessenberg form.

Elimination for first column

$$P_1 := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & 0^T \\ 0 & I - 2u_1u_1^T \end{bmatrix}.$$

Use Lemma 2.2.1 with $x^T = [a_{21}, \ldots, a_{n1}]$ to select u_1 such that

$$P_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}$$

In order to have a similarity transformation mult from right:

$$P_1AP_1^{-1} = P_1AP_1 = \text{same structure as } P_1A.$$

We will be able to construct $m-2$ householder reflectors that bring the matrix to Hessenberg form.

Elimination for first column

$$P_1 := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & 0^T \\ 0 & I - 2u_1u_1^T \end{bmatrix}.$$

Use Lemma 2.2.1 with $x^T = [a_{21}, \ldots, a_{n1}]$ to select u_1 such that

$$P_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}$$

In order to have a similarity transformation mult from right:

$$P_1AP_1^{-1} = P_1AP_1 = \text{same structure as } P_1A.$$

Elimination for second column

Repeat the process with:

$$P_2 = \begin{bmatrix} 1 & 0 & 0^T \\ 0 & 1 & 0^T \\ 0 & 0 & I - 2u_2u_2^T \end{bmatrix}$$

* Matlab demo of the first two steps of the Hessenberg reduction *

The iteration can be implemented without explicit use of the P matrices.

Algorithm 2 Reduction to Hessenberg form

Input: A matrix $A \in \mathbb{C}^{n\times n}$

Output: A Hessenberg matrix H such that $H = U^*AU$.

for $k = 1, \dots, n-2$ **do**

Compute u_k using (2.4) where $x^T = [a_{k+1,k}, \cdots, a_{n,k}]$

Compute P_kA: $A_{k+1:n,k:n} := A_{k+1:n,k:n} - 2u_k(u_k^* A_{k+1:n,k:n})$

Compute $P_kAP_k^*$: $A_{1:n,k+1:n} := A_{1:n,k+1:n} - 2(A_{1:n,k+1:n}u_k)u_k^*$

end for

Let H be the Hessenberg part of A.

* show it in matlab *