OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

FACULTY OF MATHEMATICS

Eigenvalue Problems

Celine Reddig

19. August 2025

Outline

Introduction

Problem definition

Given a square matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$. Find scalars $\lambda \in \mathbb{C}$ and vectors $\mathbf{v} \in \mathbb{C}^n$, $\mathbf{v} \neq 0$ such that

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v},$$

i.e. $(\mathbf{A} - \lambda I)\mathbf{v} = 0$. We call λ an eigenvalue of $\mathbf{A}$ and $\mathbf{v}$ an eigenvector of $\mathbf{A}$.

Introduction

Problem definition

Given a square matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$. Find scalars $\lambda \in \mathbb{C}$ and vectors $\mathbf{v} \in \mathbb{C}^n$, $\mathbf{v} \neq 0$ such that

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v},$$

i.e. $(\mathbf{A} - \lambda I)\mathbf{v} = 0$. We call λ an eigenvalue of $\mathbf{A}$ and $\mathbf{v}$ an eigenvector of $\mathbf{A}$.

Eigenvalue problems arise in many applications:

Recall Properties I

- $\sigma(\mathbf{A})$ is the set of all eigenvalues and **called spectrum of $\mathbf{A}$**.
- $E_\lambda = \{\mathbf{v} : (\mathbf{A} - \lambda I)\mathbf{v} = 0\}$ is called the **eigenspace of $\boldsymbol{\lambda}$** .
  - $\dim(E_\lambda)$ is called **geometric multiplicity**.
- Since $x \neq 0$, $(\mathbf{A} - \lambda I)x = 0$, has a non-trivial solution if an eigenvalue is a root of the **characteristic polynomial**

$$\chi(\lambda) := \det(\mathbf{A} - \lambda I) = \lambda^n + a_{n-1}\lambda^{n-1} + ... + a_0.$$

- multiplicity of λ is called **algebraic multiplicity**.

Recall Properties II

- $\mathbf{A} \to \mathbf{S}^{-1}\mathbf{A}\mathbf{S} =: \mathbf{C}$, where $\mathbf{S}$ is non-singular, is called **similarity transformation**
  - $\sigma(\mathbf{A}) = \sigma(\mathbf{C})$, and if (λ, x) is an eigenpair of $\mathbf{A}$, $(\lambda, \mathbf{S}^{-1})$ is an eigenpair of $\mathbf{C}$.

Transform $\mathbf{A}$ into matrices which eigenvalues can be easily read:

- **Diagonalization:** $D_{\mathbf{A}} = \mathbf{S}^{-1}\mathbf{A}\mathbf{S}$, where $D_{\mathbf{A}} = \operatorname{diag}\left(\sigma(\mathbf{A})\right)$ and $\mathbf{S}$ is non-singular.
  - $\mathbf{A}$ is diagonalizable if $\mathbf{A}$ has n distinct eigenvalues and for every λ_i the $g(\lambda_i) = m(\lambda_i)$.

Schur decomposition

If $\mathbf{A} \in C^{n\times n}$, $\exists \mathbf{U} \in C^{n\times n}$ unitary, such that $\mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{T}$ is upper triangular.

Some algorithms

For small dense matrices

- QR algorithm
  - computes all eigenvalues and eigenvectors
  - more modern: implicit QR algorithm
  - employed in algorithms for large and sparse matrices to small "internal" auxiliary eigenvalue problems

For symmetric and sparse matrices

- Power method
  - the largest eigenvalue and eigenvector
- Lanczos(hermitian)/Arnoldi(non-hermitian)
  - based on Krylov-Subspaces
  - only a few eigenvalues
  - very efficient
- ...

Some algorithms

For small dense matrices

- QR algorithm
  - computes all eigenvalues and eigenvectors
  - more modern: implicit QR algorithm
  - employed in algorithms for large and sparse matrices to small "internal" auxiliary eigenvalue problems

For symmetric and sparse matrices

- Power method
  - the largest eigenvalue and eigenvector
- Lanczos(hermitian)/Arnoldi(non-hermitian)
  - based on Krylov-Subspaces
  - only a few eigenvalues
  - very efficient
- ...

Assumptions

- $\mathbf{A} \in \mathbb{C}^{n \times n}$ with multually different and ordered eigenvalues $|\lambda_1| > |\lambda_2| > ... > |\lambda_n|$
- For the ease of explanation: $\mathbf{A}$ is simple, i.e. $\mathbf{A}$ has n linearly independent eigenvectors $\mathbf{v}_1, ..., \mathbf{v}_n$
- $\mathbf{v}_i$ denotes the eigenvector associated with eigenvalue λ_i.
- $\mathbf{A}$ is dense

Motivating the QR Algorithm

Basic Power Method

Method: Choose a vector $\mathbf{v}$ and compute $\mathbf{v}, \mathbf{A}\mathbf{v}, \mathbf{A}^2\mathbf{v}, \mathbf{A}^3\mathbf{v}, \ldots$. This converges usually to an eigenvector $\mathbf{v}_1$ corresponding to the largest eigenvalue.

Why? Assume: $|\lambda_1| > |\lambda_2|$. The choosen vector $\mathbf{v}$ can be expressed as:

$$\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_n\mathbf{v}_n.$$

Subspace Iteration/Simultaneous Iteration

Power method applied to a k-dim subspace S and form $S, \mathbf{A}S, \mathbf{A}^2S, \ldots$

Let $T = \langle \mathbf{v}_1, \ldots, \mathbf{v}_k \rangle$ and $U = \langle \mathbf{v}_{k+1}, \ldots \mathbf{v}_n \rangle$. Assume $|\lambda_k| > |\lambda_{k+1}|$ and let S be a k-dim subspace of $\mathbb{C}^n$ such that $S \cap U = \{0\}$). Then, $\mathbf{A}^m S \to T$ linearly with ratio $|\frac{\lambda_{k+1}}{\lambda_k}|$. (T and U are invariant subspaces)

In theory: Iterate over a basis for S given by by $\mathbf{q}_1^0, \ldots \mathbf{q}_k^0$. Then, $\langle \mathbf{A}^m q_1^0, \ldots \mathbf{A}^m q_k^0 \rangle = \mathbf{A}^m S$ for $m = 2, 3, 4$. We get bases for $\mathbf{A}S, \mathbf{A}^2 S, \ldots$

In practice:

1. Rescaling is required to avoid overflow/underflow
2. Each sequence $\mathbf{q}_i^0, \mathbf{A}\mathbf{q}_i^0, \mathbf{A}^2\mathbf{q}_i^0$ for $i = 1, \ldots k$ converges independently to $\langle \mathbf{v}_1 \rangle$

Basis is ill-conditioned $\rightarrow$ orthonormalize

Simultaneous Iteration

For $m = 1, \ldots$ do

1. $\mathbf{Q}_m = [\mathbf{q}_1^m, \ldots, \mathbf{q}_k^m]$ is orthonormal basis of $\mathbf{A}^m S$. Compute $\mathbf{A}\mathbf{Q}_m$.
2. Orthonormalize $\mathbf{A}\mathbf{Q}_m$ to get $\mathbf{Q}_{m+1} = \left[\mathbf{q}_1^{m+1}, \ldots, \mathbf{q}_k^{m+1}\right]$, a basis for $\mathbf{A}^{m+1} S$

Remark. For a complete set, i.e. $k = n$, we converge to a unitary basis $\mathbf{q}_1, \mathbf{q}_2, \ldots \mathbf{q}_n$.

But where are the eigenvalues?

For each i, $\mathbf{A}^m S_i \to T_i$ as $m \to \infty$ the first i columns of $\mathbf{Q}_m$ are a unitary basis for the $\mathbf{A}$ -invariant subspace T_i, i.e. $\forall \mathbf{x} \in T_i, \mathbf{A}\mathbf{x} \in T_i$. The subspaces are preserved under orthonormalization.

For each i, $\mathbf{A}^m S_i \to T_i$ as $m \to \infty$ the first i columns of $\mathbf{Q}_m$ are a unitary basis for the $\mathbf{A}$-invariant subspace T_i, i.e. $\forall \mathbf{x} \in T_i, \mathbf{A}\mathbf{x} \in T_i$. The subspaces are preserved under orthonormalization.

Let $\mathbf{Q} = [\mathbf{Q}_1 \mathbf{Q}_2]$ be a **unitary matrix** where its first k columns form a basis for the $\mathbf{A}$-invariant subspace T_k and define $\mathbf{B} = \mathbf{Q}^* \mathbf{A} \mathbf{Q}$. Then,

$$\mathbf{B} = \begin{pmatrix} \mathbf{Q}_1^* \mathbf{A} \mathbf{Q}_1 & \mathbf{Q}_1^* \mathbf{A} \mathbf{Q}_2 \\ \mathbf{Q}_2^* \mathbf{A} \mathbf{Q}_1 & \mathbf{Q}_2^* \mathbf{A} \mathbf{Q}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{0} & \mathbf{B}_{22} \end{pmatrix},$$

where $\mathbf{Q}_2^* \mathbf{A} \mathbf{Q}_1 = \mathbf{0}$ since T is $\mathbf{A}$-invariant, see [1]. Moreover, $\sigma(\mathbf{A}) = \sigma(\mathbf{B}_{11}) \cup \sigma(\mathbf{B}_{22})$.

The previous result requires invariant subspaces, however we only converge to an invariant subspace:

Let $[\mathbf{q}_1^m, ...\mathbf{q}_n^m] = \mathbf{Q}_m$. Then, one can show that

$$\mathbf{A}_m = \mathbf{Q}_m^* \mathbf{A} \mathbf{Q}_m \to \begin{pmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{0} & \mathbf{B}_{22} \end{pmatrix}$$

as $m \to \infty$.

Since $\mathbf{Q}_n$ contains bases also for $i = 1, ..., n-1$ spanning T_i, $\mathbf{A}_m \to \begin{pmatrix} \lambda_1 & * & ... & * \\ 0 & \lambda_2 & ... & * \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \lambda_n \end{pmatrix}$

Remarks on the Simultaneous Iteration

Algorithm:

$$\text{Let } \mathbf{Q}_0 = \mathbf{I}.$$

for $m = 1, \dots$ do

$$\mathbf{D}_{m+1} := \mathbf{A}\mathbf{Q}_m.$$

$$\mathbf{D}_{m+1} = \mathbf{Q}_{m+1}\mathbf{R}_{m+1}.$$

$$\mathbf{A}_m = \mathbf{Q}_m^*\mathbf{A}\mathbf{Q}_m.$$

- Convergence rate to T_k is $|\frac{\lambda_{k+1}}{\lambda_k}|$. Very slow. Speed up with shifts, allows for fast convergence in theory but is numerically unstable.
- To employ shifts we need the QR iteration.
- see [2] for more details.

The QR Algorithm

QR Iteration

- instead of working on $\mathbf{A}$ the QR iteration performs equivalent operations on $\mathbf{A}_m$

Simultaneous Iteration:

$$\text{Let } \underline{\mathbf{Q}}_0 = I.$$

for $m = 1, \ldots$ do

$$\mathbf{D}_{m+1} := \mathbf{A}\underline{\mathbf{Q}}_m.$$

$$\mathbf{D}_{m+1} = \underline{\mathbf{Q}}_{m+1}\mathbf{R}_{m+1}.$$

$$\mathbf{A}_m := \underline{\mathbf{Q}}_m^* \mathbf{A}\underline{\mathbf{Q}}_m.$$

QR Iteration:

$$\text{Let } \mathbf{A}_0 = \mathbf{A}.$$

for $m = 1, \ldots$ do

$$\mathbf{A}_{m-1} = \mathbf{Q}_m\underline{\mathbf{R}}_m.$$

$$\mathbf{A}_m := \underline{\mathbf{R}}_m\mathbf{Q}_m.$$

$$\underline{\mathbf{Q}}_m := \mathbf{Q}_1\mathbf{Q}_2 \cdots \mathbf{Q}_m$$

Simultaneous Iteration $\Leftrightarrow$ QR Iteration

Both schemes generate the QR decomposition $\mathbf{A}^m = \underline{\mathbf{Q}}_m \underline{\mathbf{R}}_m$ and the projection $\mathbf{A}_m = \underline{\mathbf{Q}}_m^* \mathbf{A} \underline{\mathbf{Q}}_m$

Algorithm 4.1 Basic QR algorithm

```
Let $A \in \mathbb{C}^{n \times n}$. This algorithm computes an upper triangular matrix $T$ and a unitary
    matrix $U$ such that $A = UTU^*$ is the Schur decomposition of $A$.
Set $A_0 := A$ and $U_0 = I$.
for $k = 1, 2, \ldots$ do
    $A_{k-1} =: Q_k R_k$; /* QR factorization */
    $A_k := R_k Q_k$;
    $U_k := U_{k-1} Q_k$; /* Update transformation matrix */
end for
Set $T := A_\infty$ and $U := U_\infty$.
```

Figure 1: from [3]

Remarks on the basic QR Iteration

- The basic QR iteration has the same slow convergence rate but is numerically stable.
- It is expensive since each iteration step requires the QR decomposition of a full $n \times n$ matrix, that is already $\mathcal{O}(n^3)$.

Improvements

- A preliminary reduction to a Hessenberg matirx decrease the cost of each QR step.
- The use of shifts reduces the total number of steps to attain convergence.

Improvements: Hessenberg form

Def. A matrix $\mathbf{H}$ is a Hessenberg matrix if its elements below the lower off-diagonal are zero, $h_{ij} = 0$ for $i > j + 1$.

Theorem. The Hessenberg form is preserved by the QR algorithm, i.e given $\mathbf{H} = \mathbf{QR}$, $\overline{\mathbf{H}} = \mathbf{RQ}$ is again a Hessenberg matrix.

- using Givens rotations, the QR decomposition gets very cheap $\mathcal{O}(n^2)$

Givens rotation

Givens rotation is a rotation in the plane spanned by two coordinates axes,

$$G(i,j,\theta) = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix},$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$. Pre-multiplication corresponds to a counter-clockwise rotation by θ in (i,j) plane, i.e. only the rows i and j are effected.

Givens rotation

If $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} = G(i,j,\theta)^*\mathbf{x}$,

$$\begin{pmatrix} y_i \\ y_j \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix}, \text{ and } \; y_k = x_k \; \text{ for } \; k \neq i,j.$$

By setting $c = \frac{x_i}{\sqrt{|x_i|^2 + |x_j|^2}}$ and $s = -\frac{x_j}{\sqrt{|x_i|^2 + |x_j|^2}}$, we can force $\boldsymbol{y_j}$ **to zero**, i.e Givens rotations allow zeroing a specific entry.

Givens rotations

A small example:

$$\mathbf{H} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{G(1,2,\theta_1)^*} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

$$\xrightarrow{G(2,3,\theta_2)^*} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix} \xrightarrow{G(3,4,\theta_3)^*} \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} = \mathbf{R}$$

With $\mathbf{G}_k = G(k, k+1, \theta_k)$, we obtain the QR decomposition $\underbrace{\mathbf{G}_3^*\mathbf{G}_2^*\mathbf{G}_1^*}_{\mathbf{Q}^*}\mathbf{H} = \mathbf{R}$

Givens rotations

To sketch that the second step of the QR iteration, i.e. computing $\mathbf{RQ}$ is a Hessenberg matrix, we check if $\mathbf{R}\mathbf{G}_1\mathbf{G}_2\mathbf{G}_3$ is Hessenberg:

$$\mathbf{R} = \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{\cdot G(1,2,\theta_1)} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{\cdot G(2,3,\theta_2)} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \xrightarrow{\cdot G(3,4,\theta_3)} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \end{bmatrix}$$

Algorithm 4.2 A Hessenberg QR step

Let $H \in \mathbb{C}^{n\times n}$ be an upper Hessenberg matrix. This algorithm overwrites H with $\overline{H} = RQ$ where $H = QR$ is a QR factorization of H.

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

  /* Generate G_k and then apply it: $H = G(k, k+1, \vartheta_k)^* H$ */

  $[c_k, s_k] :=$ **givens**$(H_{k,k}, H_{k+1,k})$;

  $H_{k:k+1,k:n} = \begin{bmatrix} c_k & -s_k \\ s_k & c_k \end{bmatrix} H_{k:k+1,k:n}$;

end for

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

  /* Apply the rotations G_k from the right */

  $H_{1:k+1,k:k+1} = H_{1:k+1,k:k+1} \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}$;

end for

Figure 2: from [3]

Householder reflector

Householder reflectors can zero a number of elements of a vector at once.

Def. A matrix of the form $\mathbf{P} = \mathbf{I} - 2\mathbf{u}\mathbf{u}^*$, $\|u\| = 1$ is called a Householder reflector.

- $\mathbf{P}$ is Hermitian and $\mathbf{P}^2 = \mathbf{I}$, so $\mathbf{P}$ is unitary.
- only store $\mathbf{u}$ for $\mathbf{P}\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{u}(2\boldsymbol{u}^*\boldsymbol{x})$
- one can determine $\mathbf{u}$ such that $\mathbf{P}\boldsymbol{x} = \alpha\boldsymbol{e}_1$, where $\alpha = \rho\|\boldsymbol{x}\|$, $\rho \in \mathbb{C}$ with $|\rho| = 1$, since $\mathbf{P}$ is unitary:

$$\mathbf{u} = \frac{\boldsymbol{x} - \rho\|\boldsymbol{x}\|\boldsymbol{e}_1}{\|\boldsymbol{x} - \rho\|\boldsymbol{x}\|\boldsymbol{e}_1\|} = \frac{1}{\|\boldsymbol{x} - \rho\|\boldsymbol{x}\|\boldsymbol{e}_1\|}\begin{pmatrix} x_1 - \rho\|\boldsymbol{x}\| \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

One choice for ρ is $-e^{i\varphi}$, where $x_1 = |x_1|e^{i\varphi}$ or in the real case, $\rho = -\mathrm{sign}\ (x_1)$.

Reduction to Hessenberg form

- Reduction is obtained by **similarity transformations** (preserve eigenvalues) using Householder reflectors

$$\mathbf{A} = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{P_1 \cdot} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{\cdot P_1} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} = \mathbf{P_1 A P_1}$$

$$P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \times & \times \\ 0 & \times & \times \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{I}_2 - 2\mathbf{u}_1\mathbf{u}_1^* \end{pmatrix}$$

- In general $\mathbf{H} = \mathbf{P}_{n-2} \cdot \ldots \cdot \mathbf{P}_1 A \mathbf{P}_1 \cdot \ldots \cdot \mathbf{P}_{n-2}$. The kth Householder reflector is generated by

$$(\mathbf{I} - 2\mathbf{u}_k\mathbf{u}_k^*)\begin{pmatrix} \alpha_{k+1,k} \\ \alpha_{k+2,k} \\ \vdots \\ \alpha_{n,k} \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{with } |\alpha| = \|x\|$$

- complexity is $\mathcal{O}(n^3)$
- if eigenvectors are desired store $\mathbf{U} = \mathbf{P}_1 \cdot \ldots \cdot \mathbf{P}_{n-2}$

Improvements: Shifts

Shifts

Improve convergence of the QR iteration with shifts. We assume to work on a matrix $\mathbf{H}$ in Hessenberg form that is similar to $\mathbf{A}$.

Lemma. Let $\mathbf{H}$ be an unreduced hessenberg matrix, i.e $h_{i+1,i} \neq 0$ for all $i = 1, ..., n-1$, and $\mathbf{H} = \mathbf{QR}$ the QR decomposition of $\mathbf{H}$. Then,

$$|r_{kk}| > 0, \quad \text{for all} \quad k < n.$$

So if $\mathbf{H}$ is singular, $r_{nn} = 0$

Shifts

Consider an eigenvalue λ of an unreduced Hessenberg matrix. What happens if we perform:

1. $\mathbf{H} - \lambda\mathbf{I} = \mathbf{QR}$
2. $\overline{\mathbf{H}} = \mathbf{RQ} + \lambda\mathbf{I}$?

- $\mathbf{H} \sim \overline{\mathbf{H}}$:
- using the previous Lemma:

- A perfect shift drops out the eigenvalue. We could **deflate** and proceed with a smaller matrix.

QR algorithm with shifts

- no perfect shifts available $\rightarrow$ estimate a shift heuristically

Rayleigh quotient shift: in the k-th step, set the shift σ_k equal to the last diagonal element:

$$\sigma_k := h_{nn}^{(k-1)}$$

Algorithm 4.4 The Hessenberg QR algorithm with Rayleigh quotient shift

Let $H_0 = H \in \mathbb{C}^{n\times n}$ be an upper Hessenberg matrix. This algorithm computes its Schur normal form $H = UTU^*$.

$k := 0$;

for m=n,n-1,...,2 **do**

    repeat

        $k := k + 1$;

        $\sigma_k := h_{m,m}^{(k-1)}$;

        $H_{k-1} - \sigma_k I =: Q_k R_k$;

        $H_k := R_k Q_k + \sigma_k I$;

        $U_k := U_{k-1} Q_k$;

    until $|h_{m,m-1}^{(k)}|$ is sufficiently small

end for

$T := H_k$;

Figure 3: from [3]

What happens if h_{nn} is a good approximation to an eigenvalue?

$$\begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \varepsilon & h_{nn} \end{pmatrix}$$

- Assume ε is small and perform shifted QR-Hessenberg step, i.e. compute $\mathbf{QR} = \mathbf{H} - h_{nn}\mathbf{I}$
- After $n-2$ Givens rotations, $\mathbf{R}$ is almost upper triangular:

$$\begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & \varepsilon & 0 \end{pmatrix}$$

To zero ε, we have a non-trivial Givens rotation:

$$c_{n-1} = \frac{\alpha}{\sqrt{|\alpha|^2 + |\varepsilon^2|}} \quad s_{n-1} = -\frac{\varepsilon}{\sqrt{|\alpha|^2 + |\varepsilon^2|}}.$$

Applying Givens rotation from the right to compute $\overline{H} = RQ + h_{nn}I$:

- quadratic convergence until α is also tiny

Other shift variants

- Wilkinson Shift (for real symmetric matrices):

$$\sigma_k := \text{ eigenvalue of } \begin{pmatrix} h_{n-1,n-1}^{(k-1)} & h_{n-1,n}^{(k-1)} \\ h_{n,n-1}^{(k-1)} & h_{nn}^{(k-1)} \end{pmatrix} \text{ that is closer to } h_{nn}$$

- it can be shown that $h_{n,n-1}$ converges cubically to zero
- double shift algorithm (Francis algorithm)
  - resolves the case if α is not tiny
  - real matrices with complex eigenvalues (requires complex shifts) $\rightarrow$ estimate a pair of complex conjugated eigenvalues

Summary

Summary

- The QR algorithm can be very powerful tool
  - reduce the matrix to Hessenberg form via Householder reduction
  - Givens rotation for the QR Iteration reduce the complexity of the QR decomposition
  - shifts allow for quadratic converge (or better)
  - deflate: division into smaller subproblems
  - complexity is $10n^3$ or $25n^3$ if Schurvectors are desired

Outlook

- implicit QR iteration for handling multiple shifts efficiently
- special versions for e.g. symmetric matrices with a reduction to a tridiagonal matrix

Bibliography

Bibliography

[1] D. S. Watkins, *Fundamentals of Matrix Computations*, 1st ed. Wiley, 2002. doi: 10.1002/0471249718.

[2] D. S. Watkins, "Understanding the QR Algorithm," *SIAM Review*, vol. 24, no. 4, pp. 427–440, Oct. 1982, doi: 10.1137/1024100.

[3] D. P. Arbenz, "Lecture Notes on Solving Large Scale Eigenvalue Problems," 2016.

[4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Fourth edition. in Johns Hopkins Studies in the Mathematical Sciences. Baltimore: The Johns Hopkins University Press, 2013.

[5] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, 0th ed. Society for Industrial, Applied Mathematics, 2011. doi: 10.1137/1.9781611970739.