

# 4

# Eigenvalues

*Example 4.0.1* (Analytic solution of homogeneous linear ordinary differential equations). → [52, Remark 5.6.1] Autonomous homogeneous linear ordinary differential equation (ODE):

$$\dot{\mathbf{y}} = \mathbf{A}\mathbf{y} \quad , \quad \mathbf{A} \in \mathbb{C}^{n,n} . \quad (4.0.1)$$

$$\mathbf{A} = \mathbf{S} \underbrace{\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}}_{=: \mathbf{D}} \mathbf{S}^{-1} , \quad \mathbf{S} \in \mathbb{C}^{n,n} \text{ invertible} \implies \left( \dot{\mathbf{y}} = \mathbf{A}\mathbf{y} \underset{\mathbf{z} = \mathbf{S}^{-1}\mathbf{y}}{\longleftrightarrow} \dot{\mathbf{z}} = \mathbf{D}\mathbf{z} \right) .$$

D-ITET,  
D-MATL

## 4.1 Theory of eigenvalue problems

**Definition 4.1.1** (Eigenvalues and eigenvectors). Let  $\mathbf{A} \in \mathbb{C}^{n,n}$ .

- The function  $\chi_{\mathbf{A}} : \mathbb{C} \rightarrow \mathbb{C}$  defined by  $\chi_{\mathbf{A}}(\lambda) := \det(\lambda \mathbf{I} - \mathbf{A})$  for all  $\lambda \in \mathbb{C}$  is called the characteristic polynomial of  $\mathbf{A}$ .
- A complex number  $\lambda \in \mathbb{C}$  is called **eigenvalue** (ger.: *Eigenwert*) of  $\mathbf{A}$  if  $\det(\lambda \mathbf{I} - \mathbf{A}) = 0$  (i.e., if it is a zero of the characteristic polynomial).
- The set  $\sigma(\mathbf{A}) := \{\lambda \in \mathbb{C} : \lambda \text{ eigenvalue of } \mathbf{A}\}$  is called **spectrum** of  $\mathbf{A}$ .
- If  $\lambda \in \sigma(\mathbf{A})$ , then  $\text{Eig}_{\mathbf{A}}(\lambda) := \text{Ker}(\lambda \mathbf{I} - \mathbf{A})$  is called **eigenspace** (ger.: *Eigenraum*) of  $\mathbf{A}$  associated to  $\lambda$ .
- If  $\lambda \in \sigma(\mathbf{A})$  and  $\mathbf{x} \in \text{Eig}_{\mathbf{A}}(\lambda) \setminus \{0\}$ , then  $\mathbf{x}$  is called **eigenvector** of  $\mathbf{A}$  associated to  $\lambda$ .
- If  $\lambda \in \sigma(\mathbf{A})$ , then  $m_{\mathbf{A}}(\lambda) := \dim(\text{Eig}_{\mathbf{A}}(\lambda))$  is called geometric **multiplicity** (ger.: *geometrische Vielfachheit*) of eigenvalue  $\lambda$  of  $\mathbf{A}$ .

**Lemma 4.1.4** (Similarity and spectrum).

The spectrum of a matrix is invariant with respect to **similarity transformations**:

$$\forall \mathbf{A} \in \mathbb{K}^{n,n} : \forall \text{ invertible } \mathbf{S} \in \mathbb{K}^{n,n} : \sigma(\mathbf{S}^{-1} \mathbf{A} \mathbf{S}) = \sigma(\mathbf{A}) .$$

## Eigenvalue

- problems:** ① Given  $\mathbf{A} \in \mathbb{K}^{n,n}$  (EVPs) find **all eigenvalues** (= spectrum of  $\mathbf{A}$ ).  
 ② Given  $\mathbf{A} \in \mathbb{K}^{n,n}$  find  $\sigma(\mathbf{A})$  plus **all eigenvectors**.  
 ③ Given  $\mathbf{A} \in \mathbb{K}^{n,n}$  find **a few** eigenvalues and associated eigenvectors

**Definition 4.1.8** (Generalized eigenvalues and eigenvectors). Let  $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n,n}$ .

- A complex number  $\lambda \in \mathbb{C}$  is called **generalized eigenvalue** (ger.: *Eigenwert*) of  $(\mathbf{A}, \mathbf{B})$  if  $\det(\lambda\mathbf{B} - \mathbf{A}) = 0$ .
- If  $\lambda \in \mathbb{C}$  is a generalized eigenvalue of  $(\mathbf{A}, \mathbf{B})$  and  $\mathbf{x} \in \mathbb{C}^n \setminus \{0\}$  with  $\mathbf{Ax} = \lambda\mathbf{Bx}$ , then  $\mathbf{x}$  is called **generalized eigenvector** of  $(\mathbf{A}, \mathbf{B})$  associated to  $\lambda$ .

In the setting of Definition 4.1.8 it holds that if  $\mathbf{B} \in \mathbb{C}^{n,n}$  is invertible, then the generalized eigenvalue problem is equivalent to a suitable standard eigenvalue problem, that is,

$$\forall \mathbf{x} \in \mathbb{C}^n : \quad \forall \lambda \in \mathbb{C} : \quad \left( \mathbf{Ax} = \lambda\mathbf{Bx} \iff \mathbf{B}^{-1}\mathbf{Ax} = \lambda\mathbf{x} \right).$$

However, usually it is not advisable to use this equivalence for numerical purposes!

*Remark 4.1.1 (Generalized eigenvalue problems and Cholesky factorization).* In the setting of Definition 4.1.8 let  $\mathbf{R} \in \mathbb{C}^{n,n}$  be an upper triangle matrix and let  $\mathbf{B} = \mathbf{B}^H$  be hermitesch and positive definite with Cholesky factorization  $\mathbf{B} = \mathbf{R}^H \mathbf{R}$ , then

$$\forall \mathbf{x} \in \mathbb{C}^n: \quad \forall \lambda \in \mathbb{C}: \quad \left( \mathbf{A}\mathbf{x} = \lambda \mathbf{B}\mathbf{x} \iff \underbrace{\left( \mathbf{R}^{-H} \mathbf{A} \mathbf{R}^{-1} \right)}_{\hat{\mathbf{A}}} \underbrace{\left( \mathbf{R}\mathbf{x} \right)}_{=: \hat{\mathbf{x}}} = \lambda \left( \mathbf{R}\mathbf{x} \right) \right).$$

This transformation can be used for efficient computations.



## 4.2 “Direct” Eigensolvers

Purpose: solution of eigenvalue problems ①, ② for **dense** matrices “up to machine precision”

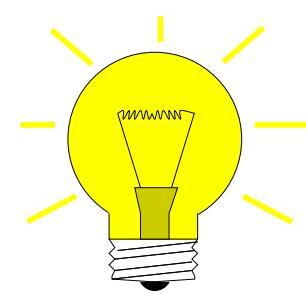
MATLAB-function: `eig`

`d = eig(A)` : computes spectrum  $\sigma(\mathbf{A})$  of  $\mathbf{A} \in \mathbb{C}^{n,n}$

`[V, D] = eig(A)` : computes  $\mathbf{V} \in \mathbb{C}^{n,n}$ , diagonal  $\mathbf{D} \in \mathbb{C}^{n,n}$  such that  $\mathbf{AV} = \mathbf{VD}$

Note:

All “direct” eigensolvers are iterative methods



Idea: Iteration based on successive **unitary** similarity transformations

$A = A^{(0)} \rightarrow A^{(1)} \rightarrow \dots \rightarrow$

:  $A = A^H$ ,  
diagonal matrix  
upper triangular matrix : else.

(superior stability of unitary transformations)



Similar functionality for generalized EVP  $\mathbf{Ax} = \lambda \mathbf{Bx}$ ,  $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n,n}$

D-ITET,  
D-MATL

`d = eig(A, B)` : computes all generalized eigenvalues

`[V, D] = eig(A, B)` : computes  $\mathbf{V} \in \mathbb{C}^{n,n}$ , *diagonal*  $\mathbf{D} \in \mathbb{C}^{n,n}$  such that  $\mathbf{AV} = \mathbf{BVD}$

Remark 4.2.5 (Computational effort for eigenvalue computations). Computational effort (#elementary operations) for `eig()`:

eigenvalues & eigenvectors of  $\mathbf{A} \in \mathbb{C}^{n,n}$ 

$$\sim 25n^3 + O(n^2)$$

only eigenvalues of  $\mathbf{A} \in \mathbb{K}^{n,n}$ 

$$\sim 10n^3 + O(n^2)$$

eigenvalues and eigenvectors of  $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$ 

$$\sim 9n^3 + O(n^2)$$

only eigenvalues of  $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$ 

$$\sim \frac{4}{3}n^3 + O(n^2)$$

only eigenvalues of tridiagonal  $\mathbf{A} = \mathbf{A}^H \in \mathbb{K}^{n,n}$ 

$$\sim 30n^2 + O(n)$$

{ }

 $O(n^3)!$ *Example 4.2.6 (Runtimes of eig).*

## Code 4.1: measuring runtimes of eig

**function** eigtiming

```

A = rand(500,500); B = A'*A;
C = gallery('tridiag',500,1,3,1);
times = [];
for n=5:5:500
    An = A(1:n,1:n); Bn = B(1:n,1:n); Cn = C(1:n,1:n);
    t1 = 1000; for k=1:3, tic; d = eig(An); t1 = min(t1,toc); end
    t2 = 1000; for k=1:3, tic; [V,D] = eig(An); t2 = min(t2,toc); end
    t3 = 1000; for k=1:3, tic; d = eig(Bn); t3 = min(t3,toc); end
    t4 = 1000; for k=1:3, tic; [V,D] = eig(Bn); t4 = min(t4,toc); end
    t5 = 1000; for k=1:3, tic; d = eig(Cn); t5 = min(t5,toc); end
    times = [times; n t1 t2 t3 t4 t5];
end

```

```
figure;
loglog(times(:,1),times(:,2),'r+', times(:,1),times(:,3),'m*',...
        times(:,1),times(:,4),'cp', times(:,1),times(:,5),'b^',...
        times(:,1),times(:,6),'k.');
xlabel('{\bf matrix\_size\_n}', 'fontsize',14);
ylabel('{\bf time_[s]}', 'fontsize',14);
title('eig_runtimes');
legend('d_=eig(A)', '[V,D]=eig(A)', 'd_=eig(B)', '[V,D]=eig(B)', 'd_=eig(C)',...
       'location', 'northwest');
```

```
print -depsc2 '../PICTURES/eigtimingall.eps'
```

```
figure;
loglog(times(:,1),times(:,2),'r+', times(:,1),times(:,3),'m*',...
        times(:,1),(times(:,1).^3)/(times(1,1)^3)*times(1,2),'k-');
xlabel('{\bf matrix\_size\_n}', 'fontsize',14);
ylabel('{\bf time_[s]}', 'fontsize',14);
title('nxn_random_matrix');
legend('d_=eig(A)', '[V,D]=eig(A)', 'O(n^3)', 'location', 'northwest');
```

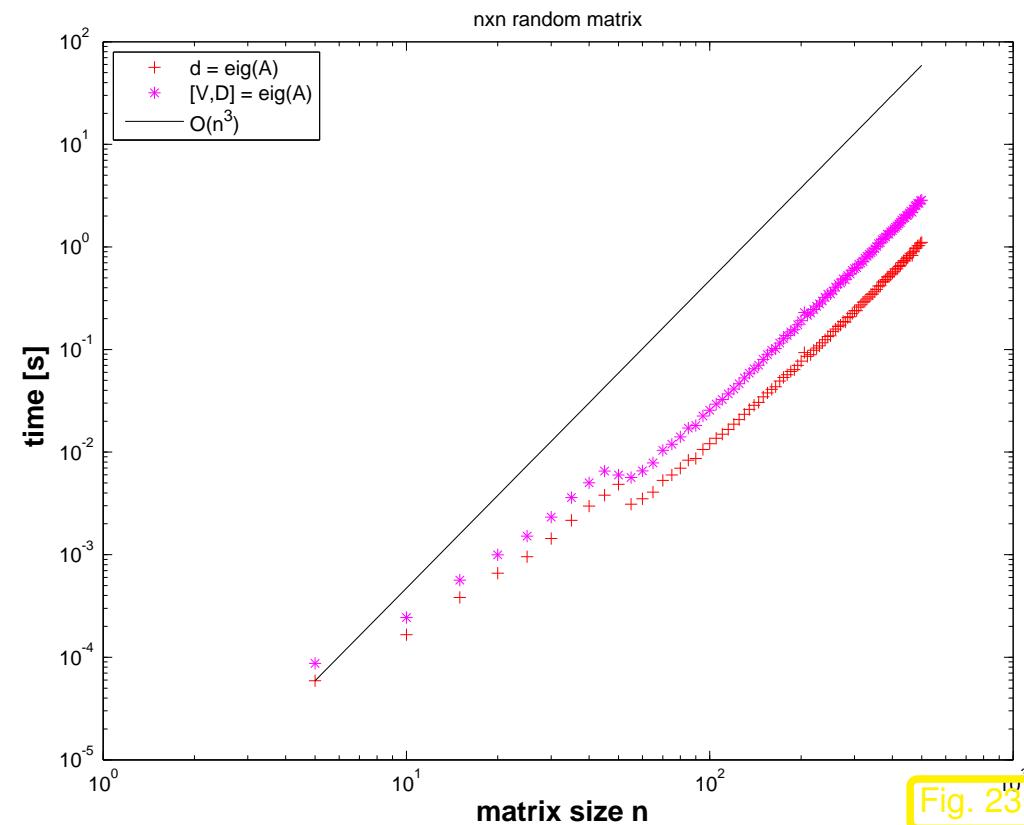
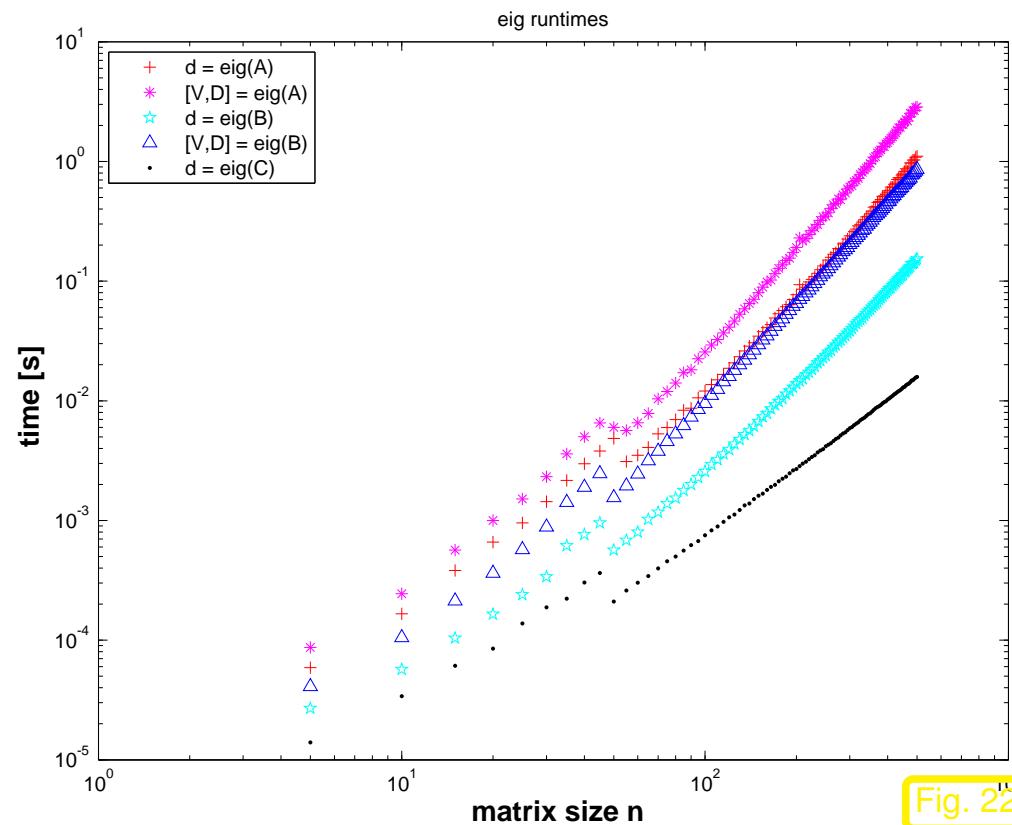
```
print -depsc2 '../PICTURES/eigtimingA.eps'
```

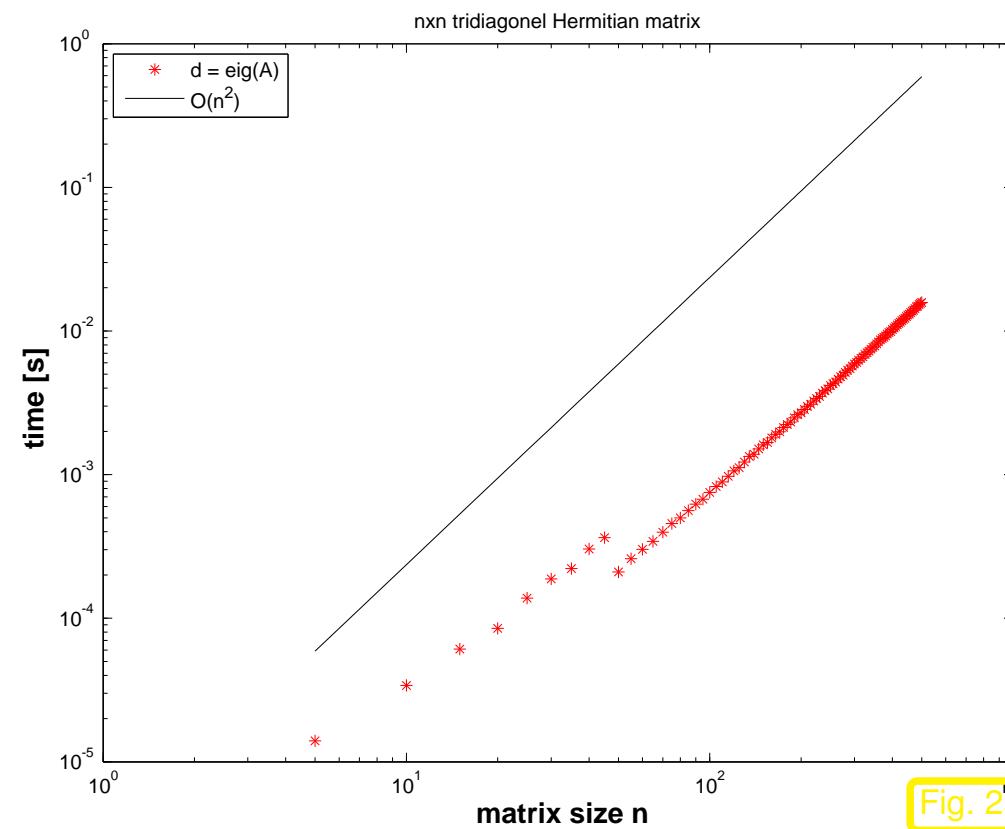
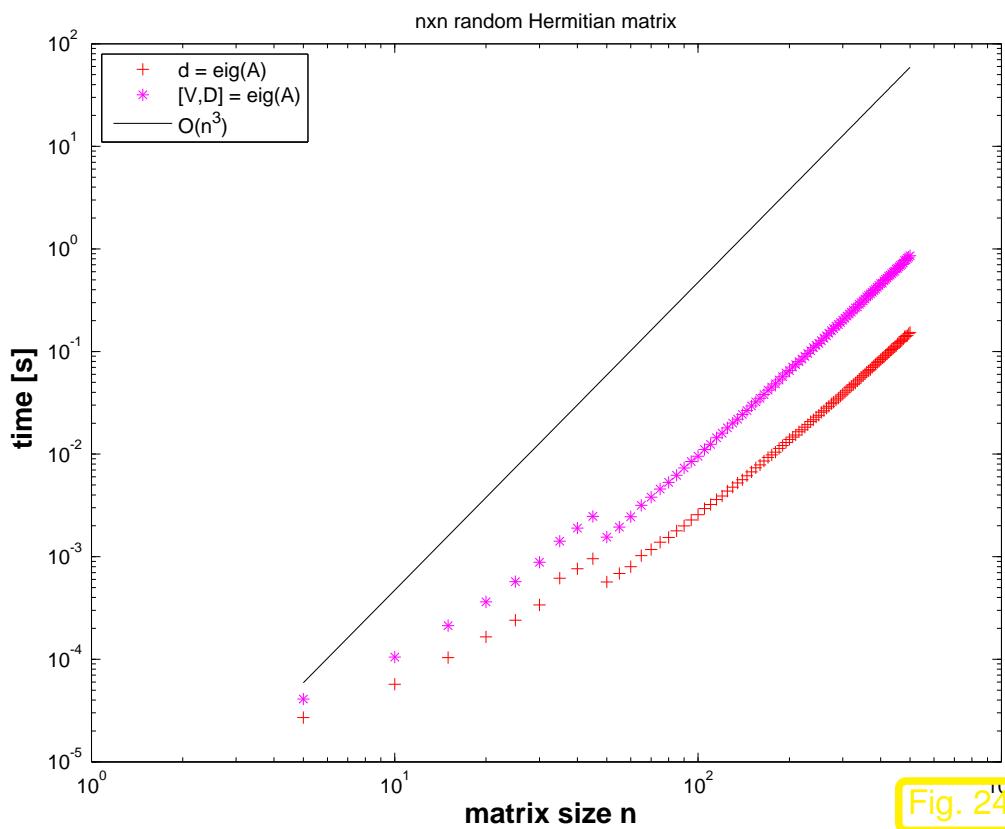
```
figure;
loglog(times(:,1),times(:,4),'r+', times(:,1),times(:,5),'m*',...
        times(:,1),(times(:,1).^3)/(times(1,1)^3)*times(1,2),'k-');
xlabel('{\bf matrix\_size\_n}', 'fontsize',14);
ylabel('{\bf time_[s]}', 'fontsize',14);
title('nxn_random_Hermitian_matrix');
legend('d_=eig(A)', '[V,D]=eig(A)', 'O(n^3)', 'location', 'northwest');

print -depsc2 '../PICTURES/eigtimingB.eps'
```

```
figure;
loglog(times(:,1),times(:,6),'r*',...
        times(:,1),(times(:,1).^2)/(times(1,1)^2)*times(1,2),'k-');
xlabel('{\bf matrix\_size\_n}', 'fontsize',14);
ylabel('{\bf time_[s]}', 'fontsize',14);
title('nxn_tridiagonal_Hermitian_matrix');
legend('d_=eig(A)', 'O(n^2)', 'location', 'northwest');

print -depsc2 '../PICTURES/eigtimingC.eps'
```





D-ITET,  
D-MATL



## 4.3 Krylov Subspace Methods

**Definition 4.3.1** (Orthogonal projection). Let  $\mathbf{U} \subset \mathbb{K}^n$  be a  $\mathbb{K}$ -subvector space of the  $\mathbb{K}^n$ . Then we denote by  $P_{\mathbb{K}^n, \mathbf{U}}: \mathbb{K}^n \rightarrow \mathbb{K}^n$  the unique linear mapping which satisfies

$$P_{\mathbb{K}^n, \mathbf{U}}(\mathbf{v}) \in \mathbf{U} \quad \text{and} \quad \langle \mathbf{v} - P_{\mathbb{K}^n, \mathbf{U}}(\mathbf{v}), \mathbf{u} \rangle = 0 \quad (4.3.1)$$

for all  $\mathbf{v} \in \mathbb{K}^n$  and all  $\mathbf{u} \in \mathbf{U}$ .

**Lemma 4.3.2** (Representation of orthogonal projection). Let  $\mathbf{U} \subset \mathbb{K}^n$  be a  $\mathbb{K}$ -subvector space of the  $\mathbb{K}^n$ , let  $k \in \mathbb{N}$ , let  $\mathbf{u}_1, \dots, \mathbf{u}_k \in \mathbf{U}$  be an orthonormal basis of  $\mathbf{U}$  and let  $\mathbf{V} = [\mathbf{u}_1 \dots \mathbf{u}_k] \in \mathbb{K}^{n,k}$ . Then

$$P_{\mathbb{K}^n, \mathbf{U}}(\mathbf{x}) = \mathbf{V} \mathbf{V}^H \mathbf{x} = \sum_{i=1}^k \mathbf{u}_i \langle \mathbf{u}_i, \mathbf{x} \rangle \quad (4.3.2)$$

for all  $\mathbf{x} \in \mathbb{K}^n$ .

**Definition 4.3.3** (Gram-Schmidt orthogonalization). Let  $\mathbf{a}_1, \dots, \mathbf{a}_m \in \mathbb{K}^n$  be linearly independent vectors. Then we define the **Gram-Schmidt** vectors  $\mathbf{v}_1, \dots, \mathbf{v}_m \in \mathbb{K}^n$  associated to  $\mathbf{a}_1, \dots, \mathbf{a}_m$  (recursively) by  $\mathbf{v}_1 = \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|_2}$  and by

$$\mathbf{v}_{k+1} := \frac{\mathbf{a}_{k+1} - P_{\mathbb{K}^n, \text{Span}\{\mathbf{a}_1, \dots, \mathbf{a}_k\}}(\mathbf{a}_{k+1})}{\left\| \mathbf{a}_{k+1} - P_{\mathbb{K}^n, \text{Span}\{\mathbf{a}_1, \dots, \mathbf{a}_k\}}(\mathbf{a}_{k+1}) \right\|_2} = \frac{\mathbf{a}_{k+1} - \sum_{i=1}^k \mathbf{v}_i \langle \mathbf{v}_i, \mathbf{a}_{k+1} \rangle}{\left\| \mathbf{a}_{k+1} - \sum_{i=1}^k \mathbf{v}_i \langle \mathbf{v}_i, \mathbf{a}_{k+1} \rangle \right\|_2}$$

for all  $k \in \{1, \dots, m-1\}$ .

**Lemma 4.3.4** (Properties of Gram-Schmidt orthogonalization). Let  $\mathbf{a}_1, \dots, \mathbf{a}_m \in \mathbb{K}^n$  be linearly independent vectors and let  $\mathbf{v}_1, \dots, \mathbf{v}_m \in \mathbb{K}^n$  be the **Gram-Schmidt** vectors associated to  $\mathbf{a}_1, \dots, \mathbf{a}_m$ . Then it holds for every  $k \in \{1, \dots, m\}$  that  $\mathbf{v}_1, \dots, \mathbf{v}_k$  is an orthonormal basis of  $\text{Span}\{\mathbf{a}_1, \dots, \mathbf{a}_k\}$ .

A good reference for some of the following notions and results is, e.g.,  
[<http://lp.uni-goettingen.de/get/text/2021>].

**Definition 4.3.5** (Krylov space). Let  $\mathbf{A} \in \mathbb{C}^{n,n}$ , let  $\mathbf{z} \in \mathbb{C}^n \setminus \{0\}$  and let  $k \in \mathbb{N}$ . Then we call the vector space

$$\mathcal{K}_k(\mathbf{A}, \mathbf{z}) := \text{Span} \left\{ \mathbf{z}, \mathbf{A}\mathbf{z}, \dots, \mathbf{A}^{(k-1)}\mathbf{z} \right\} = \left\{ p(\mathbf{A})\mathbf{z} : p: \mathbb{C} \rightarrow \mathbb{C} \text{ polynomial of degree } < k \right\}$$

the  $k$ -th **Krylov subspace** associated to  $\mathbf{A}$  and  $\mathbf{z}$ .

**Definition 4.3.6.** Let  $\mathbf{A} \in \mathbb{C}^{n,n}$  and let  $\mathbf{z} \in \mathbb{C}^n \setminus \{0\}$ . Then we define  $\deg_{\mathbf{A}}(\mathbf{z}) \in \mathbb{N}$  by

$$\begin{aligned} \deg_{\mathbf{A}}(\mathbf{z}) &:= \min \left\{ k \in \mathbb{N} : \mathcal{K}_k(\mathbf{A}, \mathbf{z}) = \mathcal{K}_{k+1}(\mathbf{A}, \mathbf{z}) \right\} \\ &= \min \left\{ k \in \mathbb{N} : (\exists \text{ polynomial } p \neq 0 \text{ with degree } k : p(\mathbf{A})\mathbf{z} = 0) \right\}. \end{aligned}$$

**Lemma 4.3.7.** Let  $\mathbf{A} \in \mathbb{C}^{n,n}$ . Then

$$\deg_{\mathbf{A}}(\mathbf{z}) \leq n, \quad \dim(\mathcal{K}_k(\mathbf{A}, \mathbf{z})) = \min(k, \deg_{\mathbf{A}}(\mathbf{z})), \quad \mathcal{K}_{\deg_{\mathbf{A}}(\mathbf{z})}(\mathbf{A}, \mathbf{z}) = \cup_{l \in \mathbb{N}} \mathcal{K}_l(\mathbf{A}, \mathbf{z})$$

for all  $\mathbf{z} \in \mathbb{C}^n \setminus \{0\}$  and all  $k \in \mathbb{N}$ .

**Definition 4.3.8** (Arnoldi process). Let  $\mathbf{A} \in \mathbb{C}^{n,n}$  and let  $\mathbf{v}_0 \in \mathbb{C}^n \setminus \{0\}$ . Then we define  $\mathbf{v}_1, \dots, \mathbf{v}_{\deg_{\mathbf{A}}(\mathbf{z})} \in \mathbb{C}^n$  recursively by  $\mathbf{v}_1 := \frac{\mathbf{v}_0}{\|\mathbf{v}_0\|_2}$  and by

$$\mathbf{v}_{k+1} := \frac{\mathbf{A}\mathbf{v}_k - P_{\mathbb{C}^n, \text{Span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}}(\mathbf{A}\mathbf{v}_k)}{\left\| \mathbf{A}\mathbf{v}_k - P_{\mathbb{C}^n, \text{Span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}}(\mathbf{A}\mathbf{v}_k) \right\|_2} = \frac{\mathbf{A}\mathbf{v}_k - \sum_{j=1}^k \mathbf{v}_j \langle \mathbf{v}_j, \mathbf{A}\mathbf{v}_k \rangle}{\left\| \mathbf{A}\mathbf{v}_k - \sum_{j=1}^k \mathbf{v}_j \langle \mathbf{v}_j, \mathbf{A}\mathbf{v}_k \rangle \right\|_2}$$

for all  $k \in \{1, \dots, \deg_{\mathbf{A}}(\mathbf{v}_0)\}$ . Moreover, we define  $\mathbf{V}_l := [\mathbf{v}_1 \dots \mathbf{v}_l] \in \mathbb{C}^{n,l}$ ,  $\mathbf{H}_l := \mathbf{V}_l^H \mathbf{A} \mathbf{V}_l \in \mathbb{C}^{l,l}$  and

$$\tilde{\mathbf{H}}_l := \begin{pmatrix} \mathbf{H}_l \\ 0 \dots 0 \left\| \mathbf{A}\mathbf{v}_l - \sum_{j=1}^l \mathbf{v}_j \langle \mathbf{v}_j, \mathbf{A}\mathbf{v}_l \rangle \right\|_2 \end{pmatrix} \in \mathbb{C}^{l+1,l} \quad (4.3.3)$$

for every  $l \in \{1, \dots, \deg_{\mathbf{A}}(\mathbf{v}_0)\}$ . We call  $(\mathbf{V}_l, \mathbf{H}_l, \tilde{\mathbf{H}}_l)$ ,  $l \in \{1, \dots, \deg_{\mathbf{A}}(\mathbf{v}_0)\}$ , Arnoldi process associated to  $\mathbf{A}$  and  $\mathbf{v}_0$ .

## Code 4.2: Arnoldi process

► Arnoldi process

➤ Computational cost

for  $k$  steps, if at most

$m$  non-zero entries

in each row of  $\mathbf{A}$ :

$$\mathcal{O}(nmk^2)$$

```
1 function [V,H] = arnoldi(A,k,v0)
2 V = [v0/norm(v0)];
3 H = zeros(k+1,k);
4 for l=1:k
5     vt = A*V(:,l);
6     for j=1:l
7         H(j,l) = dot(V(:,j),vt);
8         vt = vt - H(j,l)*V(:,j);
9     end
10    H(l+1,l) = norm(vt);
11    if (H(l+1,l) == 0), break; end
12    V = [V, vt/H(l+1,l)];
13 end
```

If it does not stop prematurely, the Arnoldi process of Code 4.2 will yield an *orthonormal basis* (ONB) of  $\mathcal{K}_{k+1}(\mathbf{A}, \mathbf{v}_0)$ . If it does stop prematurely, the Arnoldi process of Code 4.2 will yield an ONB of  $\cup_{v \in \mathbb{N}} \mathcal{K}_v(\mathbf{A}, \mathbf{v}_0)$ .

Algebraic view of the Arnoldi process of Code 4.2, meaning of output H:

$$\mathbf{V}_l = [\mathbf{v}_1, \dots, \mathbf{v}_l], \quad \mathbf{A}\mathbf{V}_l = \mathbf{V}_{l+1}\tilde{\mathbf{H}}_l, \quad \tilde{\mathbf{H}}_l \in \mathbb{C}^{l+1, l}, \quad \tilde{h}_{ij} = \begin{cases} \mathbf{v}_i^H \mathbf{A} \mathbf{v}_j & : i \leq j, \\ \|\mathbf{A} \mathbf{v}_j - \sum_{l=1}^j \mathbf{v}_l \tilde{h}_{l,j}\|_2 & : i = j + 1 \\ 0 & : \text{else.} \end{cases}$$

→  $\tilde{H}_l$  = non-square upper Hessenberg matrices

$$\left( \begin{array}{c} \textbf{A} \\ \hline \end{array} \right) \left( \begin{array}{c} \textbf{v}_1 \\ \vdots \\ \textbf{v}_l \\ \hline \end{array} \right) = \left( \begin{array}{c} \textbf{v}_1 \\ \vdots \\ \textbf{v}_l \\ \hline \textbf{v}_{l+1} \\ \hline \end{array} \right) \left( \begin{array}{c} \tilde{\textbf{H}}_l \\ \hline \end{array} \right)$$

D-ITET,  
D-MATL

Translate Code 4.2 to matrix calculus:

**Lemma 4.3.9** (Theory of Arnoldi process).

Let  $\mathbf{A} \in \mathbb{C}^{n,n}$ , let  $\mathbf{v}_0 \in \mathbb{C}^n \setminus \{0\}$  and let  $l \in \{1, \dots, \deg_{\mathbf{A}}(\mathbf{v}_0)\}$ . Then it holds for the matrices  $\mathbf{V}_l \in \mathbb{C}^{n,l}$ ,  $\mathbf{H}_l \in \mathbb{C}^{l,l}$  and  $\tilde{\mathbf{H}}_l \in \mathbb{C}^{l+1,l}$  arising in the  $l$ -th step of the Arnoldi process that

- (i)  $\mathbf{V}_l^H \mathbf{V}_l = \mathbf{I}$  (orthogonal columns),
- (ii)  $\text{Im}(\mathbf{V}_l) = \mathcal{K}_l(\mathbf{A}, \mathbf{v}_0)$ ,
- (iii)  $\mathbf{A}\mathbf{V}_l = \mathbf{V}_{l+1}\tilde{\mathbf{H}}_l$ ,  $\tilde{\mathbf{H}}_l$  is non-square upper Hessenberg matrix,
- (iv) If  $\mathbf{A} = \mathbf{A}^H$  then  $\mathbf{H}_l$  is tridiagonal ( $\triangleright$  Lanczos process).

► Eigenvalue approximation for general EVP  $\mathbf{Ax} = \lambda \mathbf{x}$  by Arnoldi process:  
 $\sigma(\mathbf{A}) \approx \sigma(\mathbf{H}_l)$  for  $l \in \{1, 2, \dots, \deg_{\mathbf{A}}(\mathbf{v}_0)\}$  sufficiently large and  $\mathbf{v}_0 \in \mathbb{K}^n$  appropriate.

Code 4.3: Arnoldi eigenvalue approximation

```

1 function [dn,V,Ht] = arnoldieig(A,v0,k,tol)
2 n = size(A,1); V = [v0/norm(v0)];
3 H = zeros(1,0); dn = zeros(k,1);
4 for l=1:n
5   d = dn;
6   Ht = [Ht, zeros(l,1); zeros(1,l)];
7   vt = A*V(:,l);
8   for j=1:l
9     Ht(j,l) = dot(V(:,j),vt);

```

```
10    vt = vt - Ht(j,l)*V(:,j);  
11  end  
12  ev = sort(eig(Ht(1:l,1:l)));  
13  dn(1:min(l,k)) = ev(end:-1:end-min(l,k)+1);  
14  if (norm(d-dn) < tol*norm(dn)), break; end;  
15  Ht(l+1,l) = norm(vt);  
16  if (H(l+1,l) == 0), break; end  
17  V = [V, vt/Ht(l+1,l)];  
18 end
```

Arnoldi process for computing the  $k$  largest (in modulus) eigenvalues of  $\mathbf{A} \in \mathbb{C}^{n,n}$

1  $\mathbf{A} \times$ vector per step  
( $\Rightarrow$  attractive for sparse matrices)

However: required storage increases with number of steps

If  $\mathbf{A}^H = \mathbf{A}$ ,  $\mathbf{v}_0 \in \mathbb{C}^n \setminus \{0\}$  and  $k \in \{1, \dots, \deg_{\mathbf{A}}(\mathbf{v}_0)\}$ , then  $\mathbf{V}_k^H \mathbf{A} \mathbf{V}_k$  is a tridiagonal matrix.

$$\mathbf{V}_k^H \mathbf{A} \mathbf{V}_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ \beta_2 & \alpha_3 & \ddots & & \\ \ddots & \ddots & & & \\ & & & \ddots & \beta_{k-1} \\ & & & \ddots & \ddots \\ & & & & \beta_{k-1} & \alpha_k \end{pmatrix} =: \mathbf{T}_k \in \mathbb{K}^{k,k} \quad [\text{tridiagonal matrix}]$$

Algorithm for computing  $\mathbf{V}_k$  and  $\mathbf{T}_k$ :

### Lanczos process

Computational effort/step:

- 1×  $\mathbf{A} \times \text{vector}$
- 2 dot products
- 2 AXPY-operations
- 1 division

Code 4.4: Lanczos process

```

1 function [V, alph , bet] = lanczos (A,k ,z0)
2 V = z0/norm(z0);
3 alph=zeros(k ,1 );
4 bet = zeros(k ,1 );
5 for j=1:k
6   p = A*V(:,j) ; alph(j) = dot(p,V(:,j));
7   w = p - alph(j)*V(:,j);
8   if ( j > 1) , w = w - bet(j-1)*V(:,j-1);
9   end;
10  bet(j) = norm(w) ; V = [V,w/bet(j)];
11 end
12 bet = bet(1:end-1);
```

## Example 4.3.4 (Impact of roundoff on Lanczos process).

$$\mathbf{A} \in \mathbb{R}^{10,10} , \quad a_{ij} = \min\{i, j\} . \quad \mathbf{A} = \text{gallery}('minij', 10);$$

Computed by  $\mathbf{V}, \alpha, \beta, \gamma = \text{lanczos}(\mathbf{A}, n, \text{ones}(n))$ , see Code 4.4:

$$\mathbf{T} = \begin{pmatrix} 38.500000 & 14.813845 & & & & & & \\ 14.813845 & 9.642857 & 2.062955 & & & & & \\ & 2.062955 & 2.720779 & 0.776284 & & & & \\ & & 0.776284 & 1.336364 & 0.385013 & & & \\ & & & 0.385013 & 0.826316 & 0.215431 & & \\ & & & & 0.215431 & 0.582380 & 0.126781 & \\ & & & & & 0.126781 & 0.446860 & 0.074650 \\ & & & & & & 0.074650 & 0.363803 & 0.043121 \\ & & & & & & & 0.043121 & 3.820888 & 11.991094 \\ & & & & & & & & 11.991094 & 41.254286 \end{pmatrix}$$

$$\sigma(\mathbf{A}) = \{0.255680, 0.273787, 0.307979, 0.366209, 0.465233, 0.643104, 1.000000, 1.873023, 5.048917, 44.766069\}$$

$$\sigma(\mathbf{T}) = \{0.263867, 0.303001, 0.365376, 0.465199, 0.643104, 1.000000, 1.873023, 5.048917, 44.765976, 44.766069\}$$

# ► Uncanny cluster of computed eigenvalues of $\mathbf{T}$ (“ghost eigenvalues”, [20, Sect. 9.2.5])

$$\mathbf{V}^H \mathbf{V} = \begin{pmatrix} 1.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000251 & 0.258801 & 0.883711 \\ 0.000000 & 1.000000 & -0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000106 & 0.109470 & 0.373799 \\ 0.000000 & -0.000000 & 1.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000005 & 0.005373 & 0.018347 \\ 0.000000 & 0.000000 & 0.000000 & 1.000000 & -0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000096 & 0.000328 \\ 0.000000 & 0.000000 & 0.000000 & -0.000000 & 1.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000001 & 0.000003 \\ 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 1.000000 & -0.000000 & 0.000000 & 0.000000 & 0.000000 \\ 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 & 0.000000 \\ 0.000251 & 0.000106 & 0.000005 & 0.000000 & 0.000000 & 0.000000 & -0.000000 & 1.000000 & -0.000000 & 0.000000 \\ 0.258801 & 0.109470 & 0.005373 & 0.000096 & 0.000001 & 0.000000 & 0.000000 & -0.000000 & 1.000000 & 0.000000 \\ 0.883711 & 0.373799 & 0.018347 & 0.000328 & 0.000003 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 1.000000 \end{pmatrix},$$

$l$	$\sigma(\mathbf{T}_l)$									D-ITET, D-MATL
1	38.500000									
2	3.392123 44.750734									
3	1.117692 4.979881 44.766064									
4	0.597664 1.788008 5.048259 44.766069									
5	0.415715 0.925441 1.870175 5.048916 44.766069									
6	0.336507 0.588906 0.995299 1.872997 5.048917 44.766069									
7	0.297303 0.431779 0.638542 0.999922 1.873023 5.048917 44.766069									
8	0.276160 0.349724 0.462449 0.643016 1.000000 1.873023 5.048917 44.766069									
9	0.276035 0.349451 0.462320 0.643006 1.000000 1.873023 3.821426 5.048917 44.766069									4.3
10	0.263867 0.303001 0.365376 0.465199 0.643104 1.000000 1.873023 5.048917 44.765976 44.766069	◇	p. 137							

## Example 4.3.5 (Stability of Arnoldi process).

$$\mathbf{A} \in \mathbb{R}^{100,100} , \quad a_{ij} = \min\{i,j\} .$$

```
A = gallery('minij', 100);
```

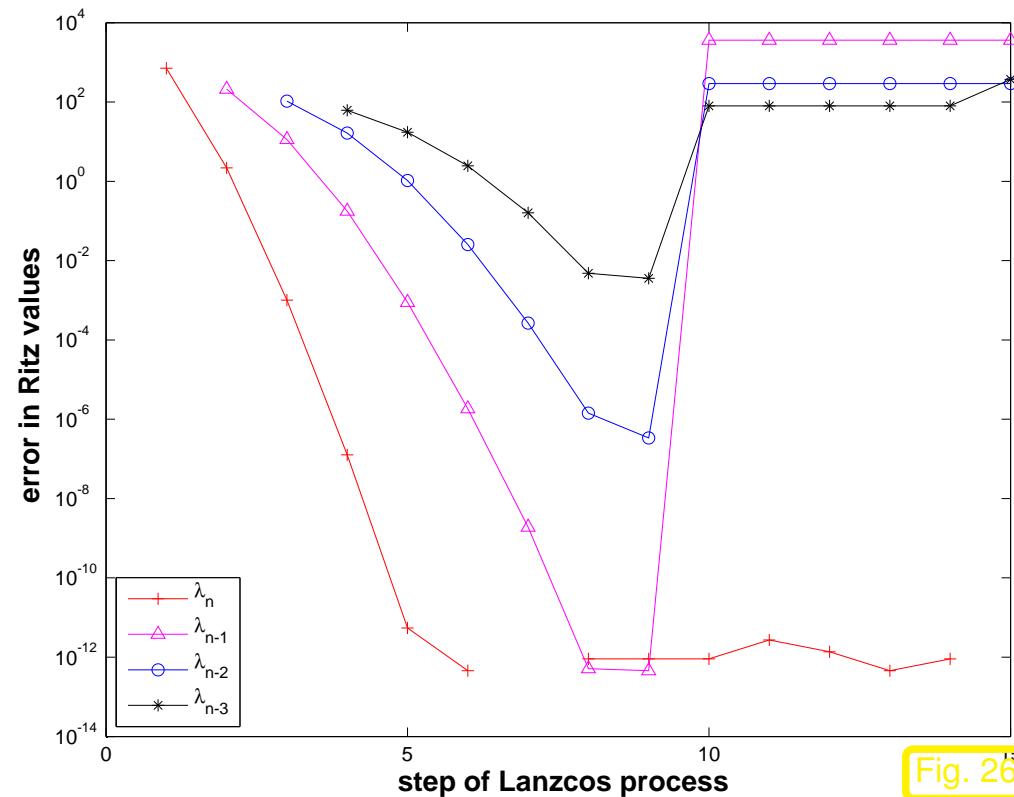


Fig. 26

Lanczos process: Ritz values

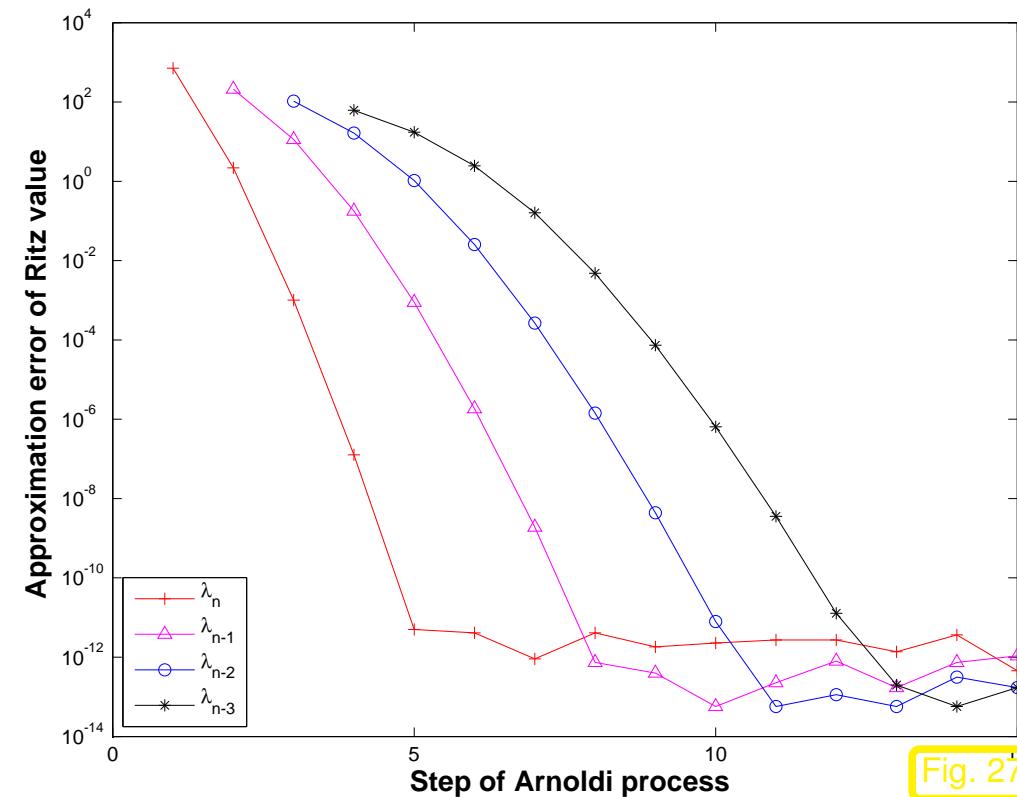


Fig. 27

Arnoldi process: Ritz values

Ritz values during Arnoldi process for  $A = gallery('minij', 10); \leftrightarrow$  Ex. 4.3.4

1									38.500000	
2								3.392123	44.750734	
3							1.117692	4.979881	44.766064	
4						0.597664	1.788008	5.048259	44.766069	
5					0.415715	0.925441	1.870175	5.048916	44.766069	
6				0.336507	0.588906	0.995299	1.872997	5.048917	44.766069	
7			0.297303	0.431779	0.638542	0.999922	1.873023	5.048917	44.766069	
8		0.276159	0.349722	0.462449	0.643016	1.000000	1.873023	5.048917	44.766069	
9	0.263872	0.303009	0.365379	0.465199	0.643104	1.000000	1.873023	5.048917	44.766069	
10	0.255680	0.273787	0.307979	0.366209	0.465233	0.643104	1.000000	1.873023	5.048917	44.766069

Observation: (almost perfect approximation of spectrum of  $\mathbf{A}$ )

D-ITET,  
D-MATL

For the above examples both the Arnoldi process and the Lanczos process are *algebraically equivalent*, because they are applied to a symmetric matrix  $\mathbf{A} = \mathbf{A}^T$ . However, they behave strikingly differently, which indicates that they are *not numerically equivalent*.

The Arnoldi process is much less affected by roundoff than the Lanczos process, because it does not take for granted orthogonality of the “residual vector sequence”. Hence, the Arnoldi process enjoys superior numerical stability compared to the Lanczos process.



4.3

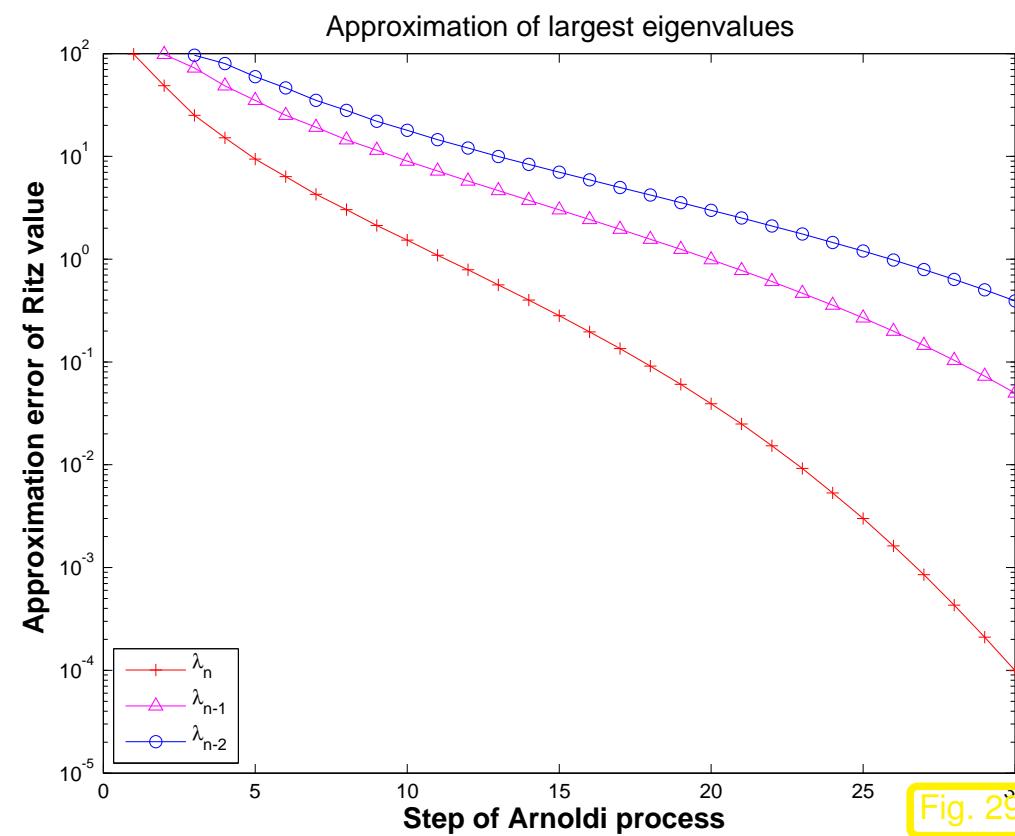
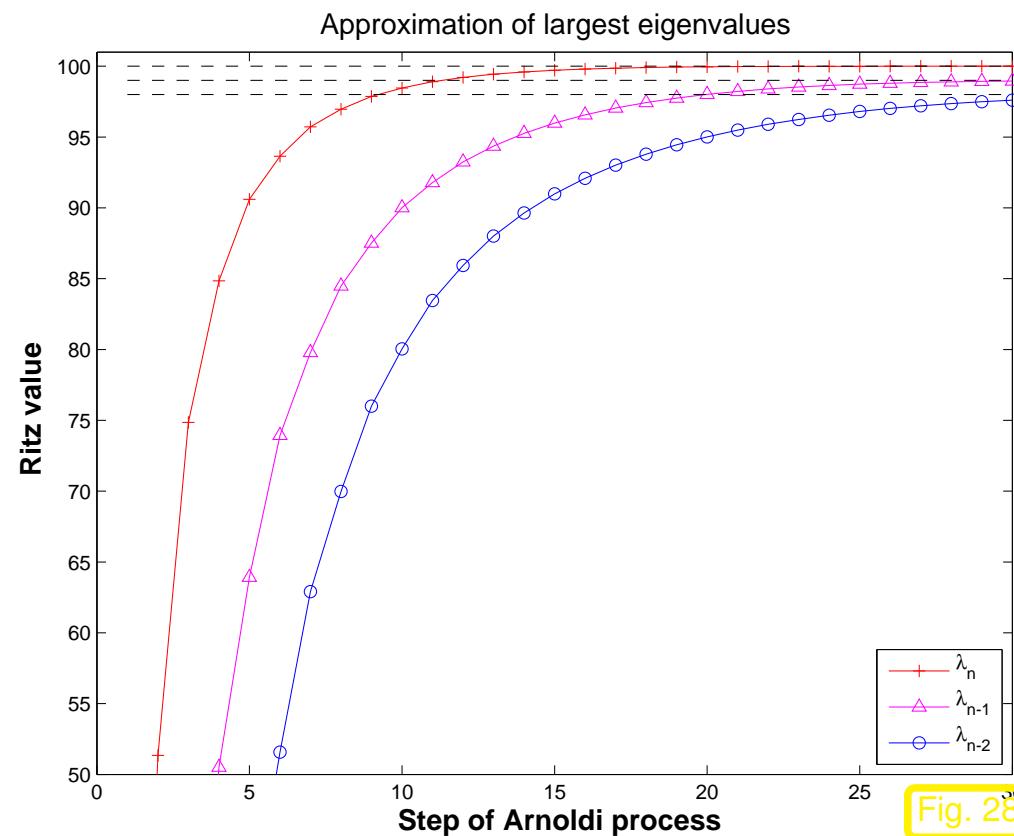
## Example 4.3.6 (Eigenvalue computation with Arnoldi process).

Eigenvalue approximation from Arnoldi process for *non-symmetric A*, initial vector `ones(100)`;

```

1 n=100;
2 M=full(gallery('tridiag', -0.5*ones(n-1,1), 2*ones(n,1), -1.5*ones(n-1,1)));
3 A=M*diag(1:n)*inv(M);

```



Approximation of smallest eigenvalues

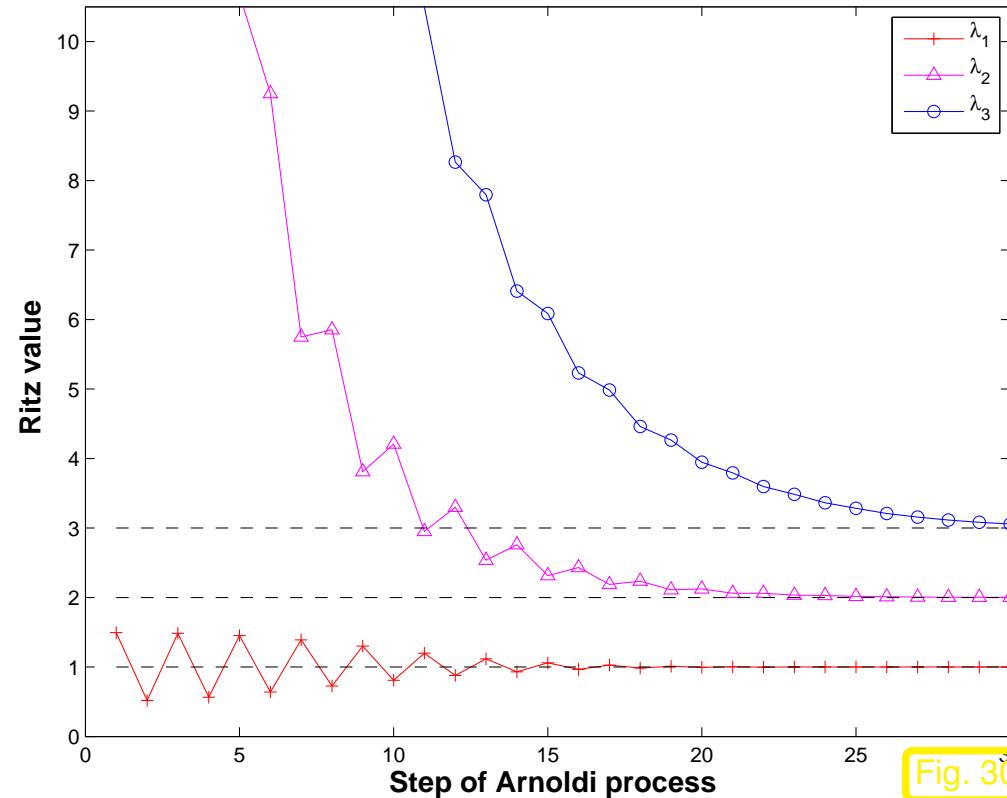


Fig. 30

Approximation of smallest eigenvalues

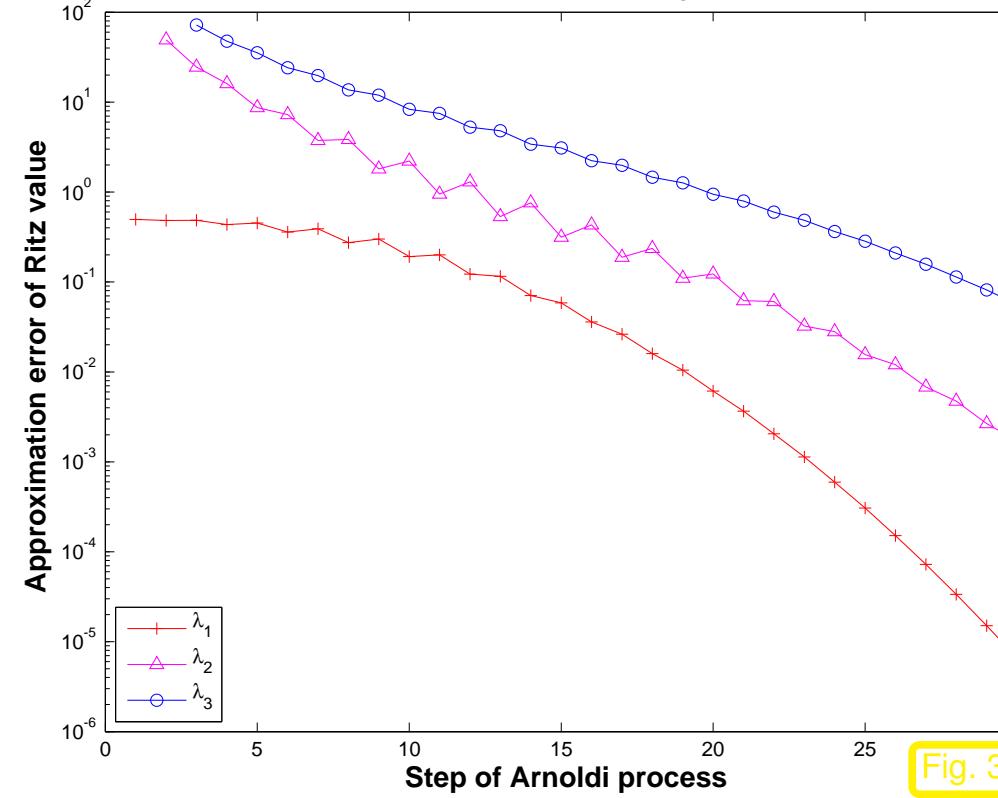


Fig. 31

Observation: “vaguely linear” convergence of largest and smallest eigenvalues



Krylov subspace iteration methods (= Arnoldi process, Lanczos process) attractive for computing *a few* of the largest/smallest eigenvalues and associated eigenvectors of *large sparse matrices*.

- d = eigs(A, k, sigma) :  $k$  largest/smallest eigenvalues of  $\mathbf{A} \in \mathbb{C}^{n,n}$  where sigma may, e.g., be 'LA', 'SA', 'LM', 'SM', ... (see help eigs)
- d = eigs(A, B, k, sigma) :  $k$  largest/smallest eigenvalues for generalized EVP  $\mathbf{Ax} = \lambda \mathbf{Bx}$ ,  $\mathbf{B}$  symmetric positive definite (s.p.d.)
- d = eigs(Afun, n, k) : Afun = handle to function providing matrix×vector for  $\mathbf{A}/\mathbf{A}^{-1}/\mathbf{A} - \alpha \mathbf{I}/(\mathbf{A} - \alpha \mathbf{B})^{-1}$ . (Use flags to tell eigs about special properties of matrix behind Afun.)