

# CHAPTER I

---

Introduction

---

In this lecture, we discuss theory, numerics and application of advanced problems in linear algebra:

- (II) matrix equations (example: solve  $AX + XB = C$ ),
- (III) matrix functions: compute  $f(A)$  or  $f(A)b$ , where  $A \in \mathbb{C}^{n \times n}$ ,  $b \in \mathbb{C}^n$ ,
- (IV) randomized algorithms.

The main focus is on problems defined by real matrices/vectors. In most chapters, we have to make the distinction between problems defined by

- dense matrices of small /moderate dimensions and
- large, sparse matrices, e.g.  $A \in \mathbb{C}^{n \times n}$ ,  $n > 10^4$  or greater, but only  $\mathcal{O}(n)$  nonzero entries, often from PDEs.

We first have to review two important standard problems in numerical linear algebra, namely solving linear systems of equations and eigenvalue problems.

## I.1 Linear systems of equations

We consider the linear system

$$Ax = b, \tag{1.1}$$

with  $A \in \mathbb{C}^{n \times n}$  ( $\mathbb{R}^{n \times n}$ ),  $b \in \mathbb{C}^n$  ( $\mathbb{R}^n$ ). The linear system (1.1) admits a unique solution, if and only if

- there exists an inverse  $A^{-1}$
  - $\det(A) \neq 0$
  - no eigenvalues/ singular values are equal to zero
  - ...
-

**Numerical methods for small and dense**  $A \in \mathbb{C}^{n \times n}$ **Gaussian Elimination (LU-factorization):**

We decompose  $A$  such that

$$A = LU, \quad L = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}, \quad U = \begin{bmatrix} \diagup & & \\ & \ddots & \\ & & \diagdown \end{bmatrix}.$$

We obtain, that

$$(1.1) \Leftrightarrow LUx = b \Leftrightarrow x = U^{-1}(L^{-1}b).$$

Hence, we solve (1.1) in two steps:

1. Solve  $Ly = b$  via backward substitution.
2. Solve  $Ux = y$  via backward substitution.

This procedure is numerically more robust with pivoting  $PAQ = LU$ , where  $P, Q \in \mathbb{C}^{n,n}$  are permutation matrices. This method has a complexity of  $\mathcal{O}(n^3)$  and is, therefore, only feasible for small (moderate) dimensions.

**QR-decomposition:**

We decompose  $A$  into a product of  $Q$  and  $R$  where  $Q$  is an orthogonal matrix and  $R$  is an upper triangular matrix leading to the so-called Gram-Schmidt or the modified Gram-Schmidt algorithm. Numerically this can be done either with Givens rotations or with Householder transformations.

**Methods for large and sparse**  $A \in \mathbb{C}^{n \times n}$ 

Storing and computing dense LU-factors is infeasible for large dimensions  $n$  ( $\mathcal{O}(n^2)$  memory,  $\mathcal{O}(n^3)$  flops). One possibility are *sparse direct solvers*, i.e. find permutation matrices  $P$  and  $Q$ , such that  $PAQ = LU$  has sparse LU-factors (cheap forward/ backward substitution and  $\mathcal{O}(n)$  memory).

**Example:** We consider the LU-factorization of the following matrix

$$A = \begin{bmatrix} * & \dots & * \\ \vdots & \ddots & \vdots \\ * & \dots & * \end{bmatrix} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \begin{bmatrix} \diagup & & \\ & \ddots & \\ & & \diagdown \end{bmatrix}.$$

With the help of permutation matrices  $P$  and  $Q$ , we can factorize

$$PAQ = \begin{bmatrix} * & \dots & * \\ \vdots & \ddots & \vdots \\ * & \dots & * \end{bmatrix} = \begin{bmatrix} * & & \\ & \ddots & \\ * & & * \end{bmatrix} \begin{bmatrix} * & \dots & * \\ & \ddots & \\ & & * \end{bmatrix}.$$

**Algorithm 1** Arnoldi method**Input:**  $A \in \mathbb{C}^{n \times n}$ ,  $b \in \mathbb{C}^n$ **Output:** Orthonormal basis  $Q_k$  of (I.2)

- 1: Set  $q_1 = \frac{b}{\|b\|}$  and  $Q_1 := [q_1]$ .
- 2: **for**  $j = 1, 2, \dots$  **do**
- 3:   Set  $z = Aq_j$ .
- 4:   Set  $w = z - Q_j(Q_j^H z)$ .
- 5:   Set  $q_{j+1} = \frac{w}{\|w\|}$ .
- 6:   Set  $Q_{j+1} = [Q_j, q_{j+1}]$ .
- 7: **end for**

Finding such  $P$  and  $Q$  and still ensuring numerical robustness is difficult and based e.g. on graph theory.

In MATLAB, sparse-direct solvers are found in the "\"-command:  $x = A \setminus b$  or  $\text{lu}(A)$ -routine. (Never use  $\text{inv}(A)$ !)

**Iterative methods**

Often an approximation  $\hat{x} \approx x$  is sufficient. Hence, we generate a sequence  $x_1, x_2, \dots, x_k$  by an iteration, such that

$$\lim_{k \rightarrow \infty} x_k = x = A^{-1}b$$

and each  $x_k$ ,  $k \geq 1$  is generated efficiently (only  $\mathcal{O}(n)$  computations). Of course, we want  $x_k \approx x$  for  $k \ll n$ .

**Idea:** Search approximated solution in a low-dimensional subspace  $\mathcal{Q}_k \subset \mathbb{C}^n$ ,  $\dim(\mathcal{Q}_k) = k$ . Let  $\mathcal{Q}_k$  be given as  $\text{range}(Q_k) = \mathcal{Q}_k$  for a matrix  $Q_k \in \mathbb{C}^{n \times k}$ .

A good choice of the subspace is the Krylow-subspace

$$Q_k = \mathcal{K}_k(A, b) = \text{span}\{b, Ab, \dots, A^{k-1}b\}. \quad (1.2)$$

It holds for  $z \in \mathcal{K}_k(A, b)$ , that  $z = p(A)b$  for a polynomial of degree  $k-1$   $p \in \Pi_{k-1}$ . An orthonormal basis of  $\mathcal{K}_k(A, b)$  can be constructed with the *Arnoldi process* presented in Algorithm 1.

The Arnoldi process requires matrix-vector products  $z = Aq$ . These are cheap for sparse  $A$  and therefore feasible for large dimensions.

We find an approximation  $x_k \in x_0 + \mathcal{Q}_k$  by two common ways:

- Galerkin-approach:  
Impose  $r = b - Ax_k \perp \text{range}(Q_k) \Leftrightarrow (Q_k^H A Q_k) y_k = Q_k^H b$ .

We have to solve a  $k$ -dimensional system  $\Rightarrow$  low costs.

- Minimize the residual:

$$\min_{x_k \in \text{range}(Q_k)} \|b - Ax_k\|$$

in some norm. If  $x_k$  is not good enough, we expand  $Q_k$ .

There are many Krylov-subspace methods for linear systems. (Simplification for  $A = A^H$ : Arnoldi  $\rightsquigarrow$  Lanczos)

In practice: Convergence acceleration by *preconditioning*:

$$Ax = b \Leftrightarrow P^{-1}Ax = P^{-1}b$$

for easily invertible  $P \in \mathbb{C}^{n,n}$  and  $P^{-1}A$  "nicer" than  $A$  ( $\rightsquigarrow$  Literature NLA I).

Another very important building block is the numerical solution of eigenvalue problems.

## I.2 Eigenvalue problems (EVP)

For a matrix  $A \in \mathbb{C}^{n,n}$  we want to find the eigenvectors  $0 \neq x \in \mathbb{C}^n$  and the eigenvalues  $\lambda \in \mathbb{C}$  such that

$$Ax = \lambda x.$$

The set of eigenvalues  $\Lambda(A) = \{\lambda_1, \dots, \lambda_n\}$  is called the *spectrum of A*.

### Small, dense problems:

Computing the Jordan-Normal-Form (JNF)

$$X^{-1}AX = J = \text{diag}(J_{s_1}(\lambda_1), \dots, J_{s_k}(\lambda_k)), \quad J_{s_j}(\lambda_j) := \begin{bmatrix} \lambda_j & 1 & & \\ & \ddots & \ddots & \\ & & \lambda_j & 1 \\ & & & \lambda_j \end{bmatrix}$$

to several eigenvalues and eigenvectors is numerically infeasible, unstable (NLA I).

**Theorem I.1** (Schur): For all  $A \in \mathbb{C}^{n \times n}$  exists a unitary matrix  $Q \in \mathbb{C}^{n,n}$  ( $Q^H Q = I$ ), such that

$$Q^H A Q = R = \underbrace{\begin{bmatrix} \lambda_1 & & * \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}}_{\text{Schur form of } A}$$

with  $\lambda_i \in \Lambda(A)$  in arbitrary order.

The Schur form can be numerically stable computed in  $\mathcal{O}(n^3)$  (NLA I) by the Francis-QR-algorithm. It is this basis for dense eigenvalue computations. In MATLAB we use  $[Q, R] = \text{schur}(A)$ . Additionally, the routine  $\text{eigs}(A)$  uses the Schur form. In general, the columns of  $Q$  are no eigenvectors of  $A$ , but  $Q_k = Q(:, 1 : k)$  spans an  $A$ -invariant subspace for all  $k$ :

$$AQ_k = Q_k R_k, \quad \text{for a matrix } R_k \in \mathbb{C}^{k \times k} \text{ with } \Lambda(R_k) \subseteq \Lambda(A).$$

But because of the  $\mathcal{O}(n^3)$  complexity and  $\mathcal{O}(n^2)$  memory, the Schur form is infeasible for large and sparse matrices  $A$ .

Eigenvalue problems defined by large and sparse matrices  $A$  can again be treated with the Arnoldi-process and projections on the Krylov-subspace  $\mathcal{K}_k(A, b) = \text{range}(Q_k)$ . We obtain the approximated eigenpair  $x_k = Q_k y_k \approx x$ ,  $\mu \approx \lambda$  by using the Galerkin-condition on the residual of the eigenvalue problem:

$$r_k = Ax_k - \mu x_k \perp \text{range}(Q_k) \Leftrightarrow Q_k^H A Q_k y_k = \mu y_k,$$

which means  $(\mu, y_k)$  are the eigenpairs of the  $k \times k$ -dimensional eigenvalue problem for  $Q_k^H A Q_k$ . This small eigenvalue problem is solvable by the Francis-QR-method. This is the basis of the  $\text{eigs}(A)$  routine in MATLAB for computing a few ( $\ll n$ ) eigenpairs of  $A$ .

**Summary:** Solving linear systems and eigenvalue problems is for small or large and sparse matrices  $A$  no problem!

## CHAPTER II

---

### Matrix Equations

---

## II.1 Preliminaries

Up to now we know linear systems of equations

$$Ax = b,$$

where  $A \in \mathbb{R}^{n \times n}$  and  $b \in \mathbb{R}^n$  are given and  $x \in \mathbb{R}^n$  has to be found.

In this course we consider more general equations

$$F(X) = C, \tag{II.1}$$

where  $F : \mathbb{R}^{q \times r} \rightarrow \mathbb{R}^{p \times s}$ ,  $C \in \mathbb{R}^{p \times s}$  is given, and  $X \in \mathbb{R}^{q \times r}$  has to be found. Equations of the form (II.1) are called *algebraic matrix equations*.

### II.1.1 Examples of Algebraic Matrix Equations

- 1)  $F(X) = AXB$ , i. e., (II.1) is

$$AXB = C.$$

- 2) Sylvester equations:

$$AX + XB = C,$$

- 3) algebraic Lyapunov equations:

- a) continuous time:

$$AX + XA^T = -BB^T, \quad X = X^T,$$

- b) discrete time:

$$AXA^T - X = -BB^T, \quad X = X^T,$$

- 4) algebraic Riccati equations:

- a) continuous time:

$$A^T X + XA - XBR^{-1}B^T X + C^T Q C = 0, \quad X = X^T,$$

- b) discrete time:

$$A^T X A - X - (A^T X B)(R + B^T X B)^{-1}(B^T X A) + C^T Q C = 0, \quad X = X^T.$$


---



c) non-symmetric

$$AX + XM - XGX + Q = 0.$$

Examples 1) – 3) are *linear* matrix equations, since the map  $F$  is linear. Equations of the type 4) are called *quadratic* matrix equations. The goal of this lecture is to understand the solution theory as well as numerical algorithms for the above matrix equations. Our focus will be on the equations 2), 3a) and 4a) since these are the equations mainly appearing in the applications.

The term *continuous-/discrete-time* in 3a,b), 4a,b) refers to applications in context of *continuous-time dynamical systems*

$$\dot{x}(t) = Ax(t), \quad t \in \mathbb{R}$$

or *discrete-time dynamical systems*

$$x_{k+1} = Ax_k, \quad k \in \mathbb{N},$$

respectively. More info in courses on *control theory* or *model order reduction*.

There are also variants of the above equations containing  $X^T$  or  $X^H$  – these will not play a prominent role here. Furthermore, there are matrix equations where  $X = X(t)$  is a matrix-valued function and  $F$  contains derivative information of  $X$ . Such equations are called *differential matrix equations*, for example the *differential Lyapunov equation*

$$\dot{X}(t) + A(t)^T X(t) + X(t)A(t) + Q(t) = 0,$$

where  $A, Q \in C([t_0, t_f], \mathbb{R}^{n \times n})$ , and  $X \in C^1([t_0, t_f], \mathbb{R}^{n \times n})$  with  $Q(t) = Q(t)^T \geq 0$  and  $X(t) = X(t)^T$  for all  $t \in [t_0, t_f]$  and the initial condition  $X(t_0) = X_0$ .

## II.2 Linear Matrix Equations

In this chapter we discuss the solution theory and the numerical solution of linear matrix equations as defined precisely below.

**Definition II.1** (linear matrix equation): Let  $A_i \in \mathbb{C}^{p \times q}$ ,  $B_i \in \mathbb{C}^{r \times s}$ , and  $C \in \mathbb{C}^{p \times s}$ ,  $i = 1, \dots, k$  be given. An equation of the form

$$\sum_{i=1}^k A_i X B_i = C \quad (\text{II.2})$$

is called a *linear matrix equation*.

### II.2.1 Solution Theory

To discuss solvability and uniqueness of solutions of (II.2) we need the following concepts.

**Definition II.2** (vectorization operator and Kronecker product): For  $X = \begin{bmatrix} x_{11} & \dots & x_{1m} \\ \vdots & & \vdots \\ x_{n1} & \dots & x_{nm} \end{bmatrix} \in \mathbb{C}^{n \times m}$  and  $Y \in \mathbb{C}^{p \times q}$

a) the vectorization operator  $\text{vec} : \mathbb{C}^{n \times m} \rightarrow \mathbb{C}^{nm}$  is given by

$$\text{vec}(X) := \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix},$$

b) the Kronecker product is given by

$$X \otimes Y = \begin{bmatrix} x_{11}Y & \dots & x_{1m}Y \\ \vdots & & \vdots \\ x_{n1}Y & \dots & x_{nm}Y \end{bmatrix} \in \mathbb{C}^{np \times mq}.$$

**Lemma II.3:** For  $\mathcal{T} \in \mathbb{C}^{n \times m}$ ,  $\mathcal{O} \in \mathbb{C}^{m \times p}$ , and  $\mathcal{R} \in \mathbb{C}^{p \times r}$  it holds

$$\text{vec}(\mathcal{T}\mathcal{O}\mathcal{R}) = (\mathcal{R}^T \otimes \mathcal{T}) \text{vec}(\mathcal{O})$$

(Note that it has to be  $\mathcal{R}^T$  in the above formula, even if all the matrices are complex.)

*Proof.* Exercise. □

By this lemma, and the obvious linearity of  $\text{vec}(\cdot)$ , we see that

$$\sum_{i=1}^k A_i X B_i = C \Leftrightarrow \underbrace{\sum_{i=1}^k (B_i^T \otimes A_i)}_{\mathcal{A}} \underbrace{\text{vec}(X)}_{\mathcal{X}} = \underbrace{\text{vec}(C)}_{\mathcal{B}},$$

and we find that (II.2) has a unique solution if and only if the linear system of equations  $\mathcal{A}\mathcal{X} = \mathcal{B}$  has one. Equivalently,  $\mathcal{A}$  has to be nonsingular.

**Theorem II.4:** The linear matrix equation (II.2) with  $ps = qr$  has a unique solution iff all eigenvalues of the matrix

$$\mathcal{A} = \sum_{i=1}^k (B_i^T \otimes A_i)$$

are non-zero.

In the following we will focus on the case  $k \leq 2$  and  $p = s = q = r$ , since Lyapunov equations ( $k = 2$ ,  $A_1 = A$ ,  $B_1 = A_2 = I_n$ ,  $B_2 = A^T$ ) and Sylvester equations ( $k = 2$ ,  $A_1 = A$ ,  $B_2 = B$ ,  $A_2 = I_n$ ,  $B_1 = I_m$ ) are important special cases of interest in applications.

To check the above condition for unique solvability, we do not want to evaluate the Kronecker products. Therefore, we now derive easily checkable conditions based on the original matrices.

**Lemma II.5:** a) Let  $W, X, Y, Z$  be matrices such that the products  $WX$  and  $YZ$  are defined. Then  $(W \otimes Y)(X \otimes Z) = (WX) \otimes (YZ)$ .

b) Let  $S, G$  be nonsingular matrices. Then  $S \otimes G$  is nonsingular, too, and  $(S \otimes G)^{-1} = S^{-1} \otimes G^{-1}$ .

c) If  $A$  and  $B$ , as well as,  $C$  and  $D$  are similar matrices then  $A \otimes C$  and  $B \otimes D$  are similar ( $A$  similar to  $B$  if  $\exists Q$  nonsingular s.t.  $A = Q^{-1}BQ$ ).

d) Let  $X \in \mathbb{C}^{n \times n}$  and  $Y \in \mathbb{C}^{m \times m}$  be given. Then

$$\Lambda(X \otimes Y) = \{\lambda\mu \mid \lambda \in \Lambda(X), \mu \in \Lambda(Y)\}.$$

*Proof.* Exercise. □

**Theorem II.6** (Theorem of Stephanos): Let  $A \in \mathbb{C}^{n \times n}$  and  $B \in \mathbb{C}^{m \times m}$  with  $\Lambda(A) = \{\lambda_1, \dots, \lambda_n\}$ ,  $\Lambda(B) = \{\mu_1, \dots, \mu_m\}$  be given. For a bivariate polynomial  $p(x, y) = \sum_{i,j=0}^k c_{ij} x^i y^j$  we define by

$$p(A, B) := \sum_{i,j=0}^k c_{ij} (A^i \otimes B^j)$$

a polynomial of the two matrices. Then the spectrum of  $p(A, B)$  is given by

$$\Lambda(p(A, B)) = \{p(\lambda_r, \mu_s) \mid r = 1, \dots, n, s = 1, \dots, m\}.$$

*Proof.* Use JNF or Schurforms of  $A, B$  + Lemma II.5. □

Now we are ready to consider our preferred special cases of (II.2).

a)  $AXB = C$ :

$$\begin{aligned} \mathcal{A} = B^T \otimes A \text{ invertible} &\Leftrightarrow \lambda \cdot \mu \neq 0 \quad \forall \lambda \in \Lambda(A) \text{ and } \mu \in \Lambda(B) \\ &\Leftrightarrow \lambda \neq 0 \text{ and } \mu \neq 0 \quad \forall \lambda \in \Lambda(A) \text{ and } \mu \in \Lambda(B) \\ &\Leftrightarrow \text{both } A \text{ and } B \text{ are nonsingular.} \end{aligned}$$

b) continuous-time Sylvester equation  $AX + XB = C$ , where  $A \in \mathbb{C}^{n \times n}$ ,  $B \in \mathbb{C}^{m \times m}$ ,  $C, X \in \mathbb{C}^{n \times m}$ :

$$\begin{aligned} \mathcal{A} = I_m \otimes A + B^T \otimes I_n \text{ invertible} &\Leftrightarrow \lambda + \mu \neq 0 \quad \forall \lambda \in \Lambda(A) \text{ and } \mu \in \Lambda(B) \\ &\Leftrightarrow \Lambda(A) \cap \Lambda(-B) = \emptyset. \end{aligned}$$

c) continuous-time Lyapunov equation  $AX + XA^H = W$ , where  $A, X \in \mathbb{C}^{n \times n}$ ,  $W = W^H \in \mathbb{C}^{n \times n}$ :

$$\mathcal{A} = I_n \otimes A + \bar{A} \otimes I_n \text{ invertible} \Leftrightarrow \Lambda(A) \cap \Lambda(-A^H) = \emptyset.$$

For example, this is the case when  $A$  is asymptotically stable.

d) discrete-time Lyapunov equations  $\rightarrow$  exercise.

The following result gives some useful results about the solution structure of Sylvester equations.

---

**Theorem II.7:** Let  $A \in \mathbb{C}^{n \times n}$ ,  $B \in \mathbb{C}^{n \times n}$  with  $\Lambda(A) \subset \mathbb{C}_-$ ,  $\Lambda(B) \subset \mathbb{C}_-$ . Then  $AX + XB = W$  has a (unique) solution

$$X = - \int_0^{\infty} e^{At} W e^{Bt} dt$$

*Proof.* Exercise. □

From now on

$$AX + XA^* = W, \quad W = W^*. \quad (\text{II.3})$$

**Definition II.8** (controllability): Let  $A \in \mathbb{C}^{n \times n}$  and  $B \in \mathbb{C}^{n \times m}$ . We say  $(A, B)$  is *controllable* if  $\text{rank}[B, AB, \dots, A^{n-1}B] = n$ .

**Lemma II.9:** The above controllability condition is equivalent to

$$\begin{aligned} \text{rank}[A - \lambda I, B] &= n \text{ for all } \lambda \in \mathbb{C} \\ \iff y^* B \neq 0 \quad \forall y \neq 0 : y^* A &= y^* \lambda \quad (\text{left. eigenvcs of } A) \end{aligned}$$

*Proof.* We first prove that  $\text{rank}[A - \lambda I, B] = n \quad \forall \lambda \in \mathbb{C}$  is equivalent to Definition II.8. Assuming that  $\text{rank}[A - \lambda I, B] < n$  for a  $\lambda \in \mathbb{C}$  then there exists a  $w \neq 0$  such that  $w^T[A - \lambda I, B] = 0$  which means that  $w^T(A - \lambda I) = 0$  and  $w^T B = 0$  and that means that  $w^T[B, AB, \dots, A^{n-1}B] = 0$  which means  $(A, B)$  is not controllable. Assuming  $(A, B)$  is not controllable and therefore  $\text{rank}[B, AB, \dots, A^{n-1}B] < n$  we define a matrix  $M$  contains a basis of the image of  $[B, AB, \dots, A^{n-1}B]$ . Then there is a matrix  $\tilde{M}$  such that  $T = [M, \tilde{M}]$  is invertible and

$$\tilde{A} = T^{-1}AT = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ 0 & \tilde{A}_{22} \end{bmatrix} \quad (\text{II.4})$$

$$\tilde{B} = T^{-1}B = \begin{bmatrix} \tilde{B}_1 \\ 0 \end{bmatrix} \quad (\text{II.5})$$

Let  $\lambda$  be an eigenvalue of  $\tilde{A}_{22}$  and  $w_{22}$  a left eigenvector. Then

$$w := \begin{bmatrix} 0 \\ \tilde{w}_{22} \end{bmatrix} T^{-1} \neq 0.$$


---

It also holds that  $w^T A = \lambda w^T$  and  $w^T B = 0$  and therefore  $\text{rank}[A - \lambda I, B]$  not full. The proof of the equivalence is basically also done within this proof.  $\square$

**Theorem II.10:** Consider Lyapunov equation (II.3) with  $W = W^* = -BB^T \leq 0$ ,  $B \in \mathbb{R}^{n \times m}$ .

- a) For  $\Lambda(A) \subset \mathbb{C}_-$ :  $(A, B)$  controllable  $\Leftrightarrow \exists$  unique sol.  $X = X^* > 0$ .
- b) Let  $(A, B)$  be controllable and assume there  $\exists$  unique sol.  $X = X^* > 0$ . Then  $\Lambda(A) \subset \mathbb{C}_-$ .

*Proof.* a) If the spectrum of  $A$  is in the left half plane and  $W = W^*$  then there exist a unique symmetric solution of the Lyapunov equation. What is left to prove is the equivalence of  $(A, B)$  being controllable and the solution being positive definite. The solution is given by

$$X = \int_0^{\infty} e^{At} B B^T e^{A^* t} dt$$

which is positive if and only if  $(A, B)$  are controllable.

- b) Take an eigenvalue  $\lambda \in \Lambda(A)$  and a corresponding left eigenvector  $y$ . Then

$$0 > -y^* B B^T y = y^* A X y + y^* X A^* y = (\lambda + \bar{\lambda}) y^* X y$$

Since  $X = X^* > 0$  we must have that  $\lambda + \bar{\lambda} = 2\text{Re}\lambda < 0$  and since  $\lambda$  was arbitrary that  $\Lambda(A) \subset \mathbb{C}_-$

$\square$

## II.2.2 Direct Numerical Solution

We have seen that linear matrix equations are equivalent to linear systems. Why do we not just apply a linear solver? Consider a (real) Lyapunov equation where we obtain the system matrix  $\mathcal{A} = I_n \otimes A + A \otimes I_n \in \mathbb{R}^{n^2 \times n^2}$ . For computing an LU-factorization of  $\mathcal{A}$  and a forward/backwards substitution we need approximately  $\frac{2}{3}(n^2)^3 = \frac{2}{3}n^6$  FLOPS and  $n^4$  memory. This is only feasible for small  $n$ . If  $n \gtrsim 50$ , then this is already prohibitively expensive (even if we exploit the structure and symmetry).

Therefore, our first goal is to develop a basic algorithm with complexity  $\mathcal{O}(n^3)$  for moderately sized linear matrix equations.

### The Bartels-Stewart Algorithm

The idea of this method is the transformation of the matrix  $A$  into Schur form.

The Schur form can be computed in a numerically stable fashion by the QR algorithm and it is the backbone of many dense eigenvalue algorithms (MATLAB `schur`).

Consider (II.3) with  $\Lambda(A) \cap \Lambda(-A^H) = \emptyset$  and let  $Q^H A Q = T$  with be the (complex) Schur form of  $A$ .

Premultiplication of (II.3) by  $Q^H$  and postmultiplication by  $Q$  leads to

$$\begin{aligned} Q^H A X Q + Q^H X A^T Q &= Q^H W Q \\ \Leftrightarrow Q^H A Q \underbrace{Q^H X Q}_{=: \tilde{X}} + Q^H X Q Q^H A^T Q &= \underbrace{Q^H W Q}_{=: \tilde{W}} \\ \Leftrightarrow T \tilde{X} + \tilde{X} T^H &= \tilde{W} \end{aligned} \quad (\text{II.6})$$

We partition this in the form

$$\begin{bmatrix} T_1 & T_2 \\ 0 & T_3 \end{bmatrix} \begin{bmatrix} X_1 & X_2 \\ X_2^H & X_3 \end{bmatrix} + \begin{bmatrix} X_1 & X_2 \\ X_2^H & X_3 \end{bmatrix} \begin{bmatrix} T_1^H & 0 \\ T_2^H & T_3^H \end{bmatrix} = \begin{bmatrix} \tilde{W}_1 & \tilde{W}_2 \\ \tilde{W}_2^H & \tilde{W}_3 \end{bmatrix},$$

where  $T_1 \in \mathbb{C}^{(n-1) \times (n-1)}$ ,  $T_2 \in \mathbb{C}^{n-1}$ ,  $T_3 \in \mathbb{C}$ . Thus we get

$$\begin{cases} T_1 X_1 + T_2 X_2^H + X_1 T_1^H + X_2 T_2^H = \tilde{W}_1, \\ T_1 X_2 + T_2 X_3 + X_2 T_3^H = \tilde{W}_2, \\ T_3 X_3 + X_3 T_3^H = \tilde{W}_3, \end{cases}$$

$$\Leftrightarrow \begin{cases} T_1 X_1 + X_1 T_1^H = \tilde{W}_1 - T_2 X_2^H - X_2 T_2^H, & (n-1) \times (n-1) \\ T_1 X_2 + X_2 T_3^H = \tilde{W}_2 - T_2 X_3, & (n-1) \times 1 \\ (T_3 + T_3^H) X_3 = \tilde{W}_3. & 1 \times 1 \end{cases}$$


---

**Algorithm 2** Bartels-Stewart algorithm (complex version)**Input:**  $A, W \in \mathbb{C}^{n \times n}$  with  $W = W^H$ .**Output:**  $X = X^H$  solving (II.3).

- 1: Compute  $T = Q^H A Q$  with the QR algorithm.
- 2: **if**  $\text{diag}(T) \cap \text{diag}(-T^H) \neq \emptyset$  **then**
- 3:   STOP (no unique solution)
- 4: **end if**
- 5: Set  $\tilde{W} := Q^H W Q$ .
- 6: Set  $k := n - 1$ .
- 7: **while**  $k > 1$  **do**
- 8:   Solve (II.7a) with  $\tilde{W}_3 = \tilde{W}(k + 1, k + 1)$  and  $T_3 = T(k + 1, k + 1)$  to obtain  $X(k + 1, k + 1)$ .
- 9:   Solve (II.7b) with  $T_1 = T(1 : k, 1 : k)$ ,  $T_2 = T(1 : k, k + 1)$ ,  $\tilde{W}_2 = \tilde{W}(1 : k, k + 1)$ , and  $X_3 = X(k + 1, k + 1)$  to obtain  $X(1 : k, k + 1)$ .
- 10:   Set  $\tilde{W} = \tilde{W}(1 : k, 1 : k) - T_2 X_3^H - X_3 T_2^H$
- 11:   Set  $k := k - 1$ .
- 12: **end while**
- 13: Solve (II.7c) with  $T_1 = T(1, 1)$  and  $\hat{W}_1 = \tilde{W}(1, 1)$ .
- 14: Set  $X := Q X Q^H$ .

Now we get

$$X_3 = \frac{\tilde{W}_3}{T_3 + \bar{T}_3}, \quad (\text{II.7a})$$

where  $T_3 + \bar{T}_3 \neq 0$  since  $T_3 \in \Lambda(A) \notin i\mathbb{R}$ . Next we obtain

$$T_1 X_2 + X_2 \bar{T}_3 = \tilde{W}_2 - T_2 X_3 =: \hat{W}_2, \quad (\text{II.7b})$$

which is a special Sylvester equation that is equivalent to the linear system

$$(\bar{T}_3 I_{n-1} + T_1) X_2 = \hat{W}_2,$$

and can easily be solved by backward substitution. Its solution always exists since  $\Lambda(T_1) \cap \{-\bar{T}_3\} = \emptyset$ . It remains to solve the smaller  $(n - 1) \times (n - 1)$  sized 'triangular' Lyapunov equation

$$T_1 X_1 + X_1 T_1^H = \tilde{W}_1 - T_2 X_2^H - X_2 T_2^H =: \hat{W}_1, \quad (\text{II.7c})$$

which is also solvable since  $\Lambda(T_1) \cap \Lambda(-T_1^H) = \emptyset$  and  $\hat{W}_1 = \hat{W}_1^H$ . This leads to the complex Bartels-Stewart algorithm, see Algorithm 2. As a convention we use MATLAB notation, i. e., we denote the section of a matrix  $A \in \mathbb{C}^{n \times n}$  consisting only of the rows  $r_1$  to  $r_2$  and the columns  $c_1$  to  $c_2$  by  $A(r_1 : r_2, c_1 : c_2)$ . If for example,  $r_1 = r_2$ , then we shortly write  $A(r_1, c_1 : c_2)$ .



**Remark:** a) In total this algorithm needs approximately

$$32n^3 \approx \underbrace{25n^3}_{\text{Schur}} + \underbrace{3n^3}_{\text{premult.}} + \underbrace{3n^3}_{\text{postmult.}} + \underbrace{n^3}_{\text{while loop}}$$

complex floating point operations.

- b) The algorithm uses only numerically backward stable parts and unitary transformations and thus it can be considered backward stable.
- c) The method is implemented in the MATLAB routine `lyap` and in SLICOT in `SB03MD` (real version only).
- d) The version for Sylvester equations works analogously (see exercise).

**Major drawback:** The algorithm uses complex arithmetic operations even if all data is real. Luckily, it can be reformulated to use real operations only.

**Theorem II.11 (real Schur form):** For every  $A \in \mathbb{R}^{n \times n}$  there exists an orthogonal matrix  $Q \in \mathbb{R}^{n \times n}$  such that  $A$  is transformed to *real Schur form*, i. e.

$$Q^T A Q = T = \begin{bmatrix} T_{11} & \dots & T_{1k} \\ & \ddots & \vdots \\ & & T_{kk} \end{bmatrix}, \quad (II.8)$$

where for  $i = 1, \dots, k$ ,  $T_{ii} \in \mathbb{R}^{1 \times 1}$  (corresponding to a real eigenvalue of  $A$ ) or  $T_{ii} = \begin{bmatrix} \alpha_i & \beta_i \\ -\beta_i & \alpha_i \end{bmatrix} \in \mathbb{R}^{2 \times 2}$  (corresponding to a pair of complex conjugate eigenvalues  $\alpha_i \pm i\beta_i$  of  $A$ ).

*Proof.* See the course on “Numerical Linear Algebra”. □

To this end, we replace the Schur form by the real Schur form (II.8). Then  $T_3$  may be a  $2 \times 2$  block, i. e.,  $T_3 = \begin{bmatrix} t_1 & t_2 \\ t_3 & t_4 \end{bmatrix}$ . We obtain

$$\begin{bmatrix} t_1 & t_2 \\ t_3 & t_4 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix} + \begin{bmatrix} x_1 & x_2 \\ x_2 & x_3 \end{bmatrix} \begin{bmatrix} t_1 & t_3 \\ t_2 & t_4 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 \\ w_2 & w_3 \end{bmatrix}.$$

This is equivalent to

$$\begin{cases} w_1 = t_1 x_1 + t_2 x_2 + t_1 x_1 + t_2 x_2 = 2(t_1 x_1 + t_2 x_2), \\ w_2 = t_1 x_2 + t_2 x_3 + t_3 x_1 + t_4 x_2 = t_3 x_1 + (t_1 + t_4) x_2 + t_2 x_3, \\ w_3 = t_3 x_2 + t_4 x_3 + t_3 x_2 + t_4 x_3 = 2(t_3 x_2 + t_4 x_3). \end{cases}$$


---

We can write this as a linear system of equations

$$\begin{bmatrix} t_1 & t_2 & 0 \\ t_3 & t_1 + t_4 & t_2 \\ 0 & t_3 & t_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{w_1}{2} \\ w_2 \\ \frac{w_3}{2} \end{bmatrix}.$$

Additionally, one can exploit the fact that  $T_3$  corresponds to a pair of complex conjugate eigenvalues  $\lambda_{1,2} = a \pm ib$  and  $T_3 = \begin{bmatrix} a & b \\ -b & a \end{bmatrix}$  which leads to

$$\begin{bmatrix} a & b & 0 \\ -b & 2a & b \\ 0 & -b & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} \frac{w_1}{2} \\ w_2 \\ \frac{w_3}{2} \end{bmatrix}.$$

Now (II.7b) becomes

$$T_1 X_2 + X_2 T_3^T = \hat{W}_2 := \tilde{W}_2 - T_2 X_3 \in \mathbb{R}^{n-2 \times 2}. \quad (\text{II.9})$$

Consider the partitions corresponding to the quasi-triangular structure of  $T_1$ :

$$X_2 = \begin{bmatrix} x_1 \\ \vdots \\ x_{k-1} \end{bmatrix}, \quad \hat{W}_2 = \begin{bmatrix} \hat{w}_1 \\ \vdots \\ \hat{w}_{k-1} \end{bmatrix},$$

In general we have  $x_i, \hat{w}_i \in \mathbb{R}^{n_i \times n_k}$ , where  $n_i, n_k \in \{1, 2\}$  and  $i = 1, \dots, k-1$ .

We now compute  $X_2$  block-wise by progressing upwards from  $x_{k-1}$  to  $x_1$ . It holds

$$T_{jj} x_j + x_j T_3^T = \hat{w}_j - \sum_{h=j+1}^k T_{jh} x_h =: \tilde{w}_j, \quad j = k-1, k-2, \dots, 1.$$

For the solution of this Sylvester equation four cases have to be considered:

- a)  $n_j = n_k = 1$ : We obtain a scalar equation such that  $x_j = \tilde{w}_j / (T_{jj} + T_3)$ .
- b)  $n_j = 2, n_k = 1$ : We obtain a linear system in  $\mathbb{R}^2$  with unique solution given by

$$(T_{jj} + T_3 I_2) x_j = \tilde{w}_j.$$

- c)  $n_j = 1, n_k = 2$ : We obtain a linear system in  $\mathbb{R}^2$  with unique solution given by

$$(T_{jj} I_2 + T_3) x_j^T = \tilde{w}_j^T.$$

- d)  $n_j = 2, n_k = 2$ : We obtain a linear system in  $\mathbb{R}^4$  with unique solution given by

$$((I_2 \otimes T_{jj}) + (T_3 \otimes I_2)) \text{vec}(x_j) = \text{vec}(\tilde{w}_j).$$

Hence, we get  $X_2$  and can set up a Lyap. eqn. for  $X_1$  defined by  $T_1$ . Repeat whole process until  $T_1 \in \mathbb{R}$  or  $T_1 \in \mathbb{R}^{2 \times 2}$ . Then back-transform the solution.

**Remark II.12:** The Sylvester equation (II.9) can be solved alternatively by solving a linear system of the form

$$(T_1^2 + \alpha T_1 + \beta I_{n-2})X_2 = \tilde{W}_2,$$

where  $X_2 = [s, t]$ ,  $\tilde{W}_2 = [y, z] \in \mathbb{R}^{n-2 \times 2}$  and  $\alpha, \beta \in \mathbb{R}$  (see exercise).

### Hammarling's Method

Now we consider (II.3) with  $W = -BB^T$ . By Theorem II.10 we know that  $X = X^T > 0$ , provided that  $\Lambda(A) \subset \mathbb{C}^-$  and the pair  $(A, B)$  is controllable. Sometimes it is desirable to only compute a factor  $U$  of the solution, i.e.,  $X = UU^H$  with some matrix  $U$ . Later we will see that many further algorithms such as projection methods for large scale matrix equations proceed with factors rather than Gramians themselves.

Assume that we have already computed and applied the Schur decomposition of  $A = Q^H T Q$ , analogously to (II.6). So our starting point is

$$T \tilde{X} + \tilde{X} T^H = -\tilde{B} \tilde{B}^H \quad \text{with} \quad \tilde{X} = Q^H X Q, \quad \tilde{B} = Q^H B. \quad (\text{II.10})$$

Since  $X > 0$ , we also have  $\tilde{X} > 0$  by Sylvester's law of inertia. Our goal is to compute upper triangular Cholesky factors  $\tilde{U}$  of  $\tilde{X} = \tilde{U} \tilde{U}^H$ .

Partition

$$\tilde{U} = \left[ \begin{array}{c|c} \square & \\ \hline 0 & \tau \end{array} \right] = \begin{bmatrix} U_1 & u \\ 0 & \tau \end{bmatrix}, \quad U_1 \in \mathbb{C}^{(n-1) \times (n-1)}, \quad u \in \mathbb{C}^{n-1}, \quad 0 < \tau \in \mathbb{R}.$$

Hammarling's method computes (similar to B.S.) first  $\tau$  (scalar equation), then  $u$  (LS of size  $n-1$ ), and finally  $U_1$  as Cholesky factor or a  $n-1 \times n-1$  Lyap. equation defined by  $T_1$ . As in B.S., repeat this until  $T_1 \in \mathbb{C}$ , afterwards back-transform  $U \leftarrow Q U$ . Complexity, stability, real version analog to BS. Details here omitted.

**Remark:** Iterative methods for small, dense Matrix Equations: There are several, iterative methods computing sequences  $X_k, k \geq 0$  converging to the true solution, i.e.,  $\lim_{k \rightarrow \infty} X_k = X$ . For instance:

- Matrix sign function iteration
- Alternating directions implicit (ADI) iteration  $\rightsquigarrow$  later for large problems.

### II.2.3 Iterative Solutions of Large and Sparse Matrix Equations

Now we consider

$$AX + XA^T = -BB^T, \quad (\text{II.11})$$

where  $A \in \mathbb{R}^{n \times n}$  and  $n$  is 'large', but  $A$  is sparse, i. e., only a few entries in  $A$  are non-zero. Therefore, multiplication with  $A$  can be performed in  $\mathcal{O}(n)$  rather than  $\mathcal{O}(n^2)$  FLOPS. Also solves with  $A$  or  $A + pI$  can be performed efficiently.

However,  $X \in \mathbb{R}^{n \times n}$  is usually dense and thus  $X$  cannot be stored for large  $n$  since we would need  $\mathcal{O}(n^2)$  memory.

Thus the question arises whether it is possible to store the solution  $X$  more efficiently.

#### The Low-Rank Phenomenon

In practice we often have  $B \in \mathbb{R}^{n \times m}$ , where  $m \ll n$ , i. e., the right-hand side  $BB^T$  has a low rank. Recall that if  $(A, B)$  is controllable then  $X = X^T \geq 0$  and  $\text{rank}(X) = n$ .

It is a very common observation in practice that the eigenvalues of  $X$  solving (II.11) decay very rapidly towards zero, and fall early below the machine precision.

This gives the concept of the numerical rank of  $X$ :

$$\text{rank}(X, \tau) = \text{argmin}_{j=1, \dots, \text{rank}(X)} \{\sigma_j(X) \geq \tau\}, \quad \text{e.g., } \tau = \epsilon_{\text{mach}} \sigma_1(X).$$

Can we also theoretically explain this eigenvalue decay?

**Theorem II.13:** Let  $A$  be diagonalizable, i. e., there exists an invertible matrix  $V \in \mathbb{C}^{n \times n}$  such that  $A = V\Lambda V^{-1}$ . Then the eigenvalues of  $X$  solving (II.11) with  $B \in \mathbb{C}^{n \times m}$  satisfy

$$\frac{\lambda_{k+1}(X)}{\lambda_1(X)} \leq \|V\|_2^2 \|V^{-1}\|_2^2 \rho(M_k)^2$$

for any choice of shift parameters  $p_k$  used to construct

$$M_k = \prod_{i=1}^k (A - p_i I)(A + p_i I)^{-1}$$

(in particular, the optimal ones).

In the Theorem above the spectral radius  $\rho$  of a matrix is used:

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i(A)|.$$

**Remark II.14:** • If the eigenvalues of  $A$  cluster in the complex plane, only a few  $p_k$  in the clusters suffice to get a small  $\rho(M_k)$  and thus  $\lambda_i(X)$  decay fast.

- If  $A$  is normal, then  $\|V\|_2 \|V^{-1}\|_2 = 1$  and the bound gives a good explanation for the decay. The nonnormal case is much harder to understand.
- This bound (and most others) does not precisely incorporate the eigenvectors of  $A$  as well as the precise influence of  $B$ .

**Consequence:** If there is a fast decay of  $\lambda_i(X)$ , then  $X$  can be well approximated as  $X = X^T \approx ZZ^H$ , where  $Z \in \mathbb{C}^{n \times r}$  with  $r \ll n$  is a *low-rank solution factor*. Hence, only  $nr$  memory is required. Thus, in the next subsection we consider algorithms for computing the factor  $Z$  without explicitly forming  $X$ .

---

### Projection Methods

Now we consider projection-based methods for the solution of large and sparse Lyapunov equations

The main idea consists of representing the solution  $X$  by an approximation extracted from a low-dimensional subspace  $\mathcal{Q}_k = \text{im } Q_k$  with  $Q_k^T Q_k = I_{km}$ , i. e.,  $X \approx X_k = Q_k Y_k Q_k^T$  for some  $Y_k \in \mathbb{R}^{km \times km}$ . Impose a Galerkin condition

$$R(X_k) := AX_k + X_k A^T + BB^T \perp \mathcal{Z}_k,$$

where

$$\mathcal{Z}_k := \left\{ Q_k Z Q_k^T \in \mathbb{R}^{n \times n} \mid Q_k^T Q_k = I_{mk}, \text{im } Q_k = \mathcal{Q}_k, Z \in \mathbb{R}^{km \times km} \right\}$$

and orthogonality is with respect to the trace inner product. Equivalently,  $Y_k$  solves the small-scale Lyapunov equation

$$H_k Y_k + Y_k H_k^T + Q_k^T B B^T Q_k = 0, \quad H_k := Q_k^T A Q_k, \quad (\text{II.12})$$

which can be solved by the Bartels-Stewart or Hammerling's method.

In case that the residual norm  $\|R(X_k)\|$  is not small enough, we increase the dimension of  $\mathcal{Q}_k$  by a clever expansion (orthogonally expand  $Q_k$ ), otherwise we prolongate to obtain  $X_k = Q_k Y_k Q_k^T$  (never formed explicitly).

What are good choices for  $\mathcal{Q}_k$ ?

#### a) Standard block Krylov subspaces

$$\mathcal{Q}_k = \mathcal{K}_k(A, B) := \text{span}\{B, AB, \dots, A^{k-1}B\} :$$

A matrix  $Q_k$  with orthonormal columns spanning  $\mathcal{Q}_k$  can be generated by a block Arnoldi process, i. e. in the  $k$ th iteration we have  $Q_k = [V_1 \ \dots \ V_k]$  fulfilling (assuming there is no breakdown in the process)

$$A Q_k = Q_k H_k + V_{k+1} H_{k+1,k} E_k^T,$$

where

$$H_k = \begin{bmatrix} H_{11} & H_{12} & \dots & \dots & H_{1k} \\ H_{21} & H_{22} & \dots & \dots & \vdots \\ 0 & H_{32} & H_{33} & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & H_{k,k-1} & H_{kk} \end{bmatrix}$$

is a block upper Hessenberg matrix and  $E_k$  is a matrix of the last  $m$  columns of  $I_{km}$ , and

$$H_k = Q_k^T A Q_k.$$

The residual norm computation for this method is cheap as shown by the following theorem.

**Theorem II.15:** Suppose that  $k$  steps of the block Arnoldi process have been taken. Assume that  $\Lambda(H_k) \cap \Lambda(-H_k) = \emptyset$ . Then the following statements are satisfied:

- a) It holds  $Q_k^T R(Q_k Y Q_k^T) Q_k = 0$  if and only if  $Y = Y_k$ , where  $Y_k$  solves the Lyapunov equation (II.12).
- b) The residual norm is given by

$$\|R(Q_k Y_k Q_k^T)\|_F = \sqrt{2} \|H_{k+1,k} E_k^T Y_k\|_F.$$

*Proof.* Exercise. □

Unfortunately, this method often converges only slowly. Therefore, one often chooses modified Krylov subspaces as follows.

b) **Extended block Krylov subspaces**

$$\mathcal{EK}_q(A, B) := \mathcal{K}_q(A, B) \cup \mathcal{K}_q(A^{-1}, A^{-1}B) :$$

The resulting method is also known as EKSM (extended Krylov subspace method) or KPIK (Krylov plus inverted Krylov). We obtain a similar construction formula as for the block Arnoldi method above and also the residual norm formula is similar. However, the approximation quality is often significantly better than with  $\mathcal{K}_q(A, B)$  only. On the other hand, the subspace dimension grows by  $2m$  in each iteration step (until  $n$  is reached).

c) **Rational Krylov subspaces**

$$\begin{aligned} \mathcal{RK}_q(A, B, S) & \qquad \qquad \qquad \text{(II.13)} \\ & := \text{span}\{(s_1 I_n - A)^{-1} B, (s_2 I_n - A)^{-1} B, \dots, (s_q I_n - A)^{-1} B\}, \\ S = \{s_1, \dots, s_q\} & \subset \mathbb{C}^+, \quad s_i \neq s_j, i \neq j : \quad \text{(shifts)} \end{aligned}$$

This choice often gives an even better approximation quality compared to  $\mathcal{EK}_q(A, B)$ , provided that *good* shifts  $S$  are known. Generating the basis requires solving LS  $(s_i I - A)v = q_i$ , but this is usually efficiently possible (cf. Intro).

The shifts  $s_i$  are crucial for a fast convergence, but finding good ones is difficult. For one possible shift selection approach, let  $m = 1$ . One can show

$$\|R_k\| \sim \max |\psi_k(z)| \quad \text{with} \quad \psi_k(z) = \prod_{j=1}^k \frac{z - \lambda_j}{z + s_j}, \quad \lambda_j \in \Lambda(H_k).$$

This leads to the following procedure for getting the next shift

$$s_{k+1} = \operatorname{argmax}_{z \in \partial D} |\psi_k(z)|,$$

where  $\partial D$  is a discrete set of point taken from the convex hull of  $\Lambda(H_k)$  ( $\partial D \subset \operatorname{conv}(\Lambda(H_k))$ ).

For all choices of subspace **a)-c)**: Is the reduced Lyapunov equation (II.12) always uniquely solvable?

For general matrices  $A$  the answer is no. However, for *strictly dissipative matrices*, i. e., matrices  $A$  with  $A + A^T < 0$  we have the following result.

**Theorem II.16:** Let  $A \in \mathbb{R}^{n \times n}$  be strictly dissipative and  $Q_k \in \mathbb{R}^{n \times m_k}$  with  $Q_k^T Q_k = I_{km}$ . Then  $\Lambda(H_k) \subset \mathbb{C}^-$  and the reduced Lyapunov equation (II.12) is always uniquely solvable.

*Proof.* Since  $A + A^T$  is symmetric and negative definite, it holds  $x^H (A + A^T) x < 0$  for all  $x \in \mathbb{C}^n$ . Then we have

$$\begin{aligned} z^H (H_k + H_k^T) z &= z^H (Q_k^T A Q_k + Q_k^T A^T Q_k) z \\ &= y^H (A + A^T) y < 0, \quad y := Q_k z, \quad \forall z \in \mathbb{C}^{km} \\ \Rightarrow H_k + H_k^T &< 0 \end{aligned}$$

Now let  $H_k \hat{x} = \hat{\lambda} \hat{x}$  for  $\hat{x} \in \mathbb{C}^{km} \setminus \{0\}$ . Then we have

$$\hat{x}^H (H_k + H_k^T) \hat{x} = \hat{\lambda} \hat{x}^H \hat{x} + \bar{\hat{\lambda}} \hat{x}^H \hat{x} = 2 \operatorname{Re}(\hat{\lambda}) \hat{x}^H \hat{x} < 0.$$

Thus  $\Lambda(H_k) \subset \mathbb{C}^-$  and the reduced Lyapunov equation (II.12) is uniquely solvable.  $\square$



**Low-rank ADI**

Consider the discrete-time Lyapunov equations

$$X = AXA^T + W, \quad A \in \mathbb{R}^{n \times n}, W = W^T \in \mathbb{R}^{n \times n}. \quad (\text{II.14})$$

The existence of a unique solution is ensured if  $|\lambda| < 1$  for all  $\lambda \in \Lambda(A)$  (see exercise). This motivates the basic iteration

$$X_k = AX_{k-1}A^T + W, \quad k \geq 1, X_0 \in \mathbb{R}^{n \times n}. \quad (\text{II.15})$$

Let  $A$  be diagonalizable, i.e., there exists a nonsingular matrix  $V \in \mathbb{C}^{n \times n}$  such that  $A = V\Lambda V^{-1}$ . Let  $\rho(A) := \max_{\lambda \in \Lambda(A)} |\lambda|$  denote the spectral radius of  $A$ . Since

$$\begin{aligned} \|X_k - X\|_2 &= \|A(X_{k-1} - X)A^T\|_2 = \dots = \|A^k(X_0 - X)(A^T)^k\|_2 \\ &\leq \|A^k\|_2^2 \|X_0 - X\|_2 \leq \|V\|_2^2 \|V^{-1}\|_2^2 \rho(A)^{2k} \|X_0 - X\|_2, \end{aligned} \quad (\text{II.16})$$

this iteration converges because  $\rho(A) < 1$  (fixed point argumentation).

For continuous-time Lyapunov equations, recall the result from the exercise:

**Lemma II.17:** The continuous-times Lyapunov equation

$$AX + XA^T = W, \quad \Lambda(A) \subset \mathbb{C}^-$$

is equivalent to the discrete-time Lyapunov equation

$$\begin{aligned} X &= C(p)XC(p)^H + \tilde{W}(p), \quad C(p) := (A - \bar{p}I_n)(A + pI_n)^{-1}, \\ \tilde{W}(p) &:= -2\operatorname{Re}(p)(A + pI_n)^{-1}W(A + pI_n)^{-H} \end{aligned} \quad (\text{II.17})$$

for  $p \in \mathbb{C}^-$ .

*Proof.* Exercise. □

We call  $C(p)$  a *Cayley transformations* of  $A$  which is the rational function

$$\phi_p(z) = \frac{z - \bar{p}}{z + p}.$$

applied to  $A$ . For  $z, p \in \mathbb{C}_-$  we have  $|\phi_p(z)| < 1$ . It can be easily shown that (special case of *spectral mapping theorem*  $\rightsquigarrow$  later)

$$\Lambda(C(p)) = \{\phi_p(\lambda), \lambda \in \Lambda(A)\}$$

and therefore  $\rho(C(p)) < 1$ . Applying (II.15) to (II.17) gives the *Smith iteration*

$$X_k = C(p)X_{k-1}C(p)^H + \tilde{W}(p), \quad k \geq 1, X_0 \in \mathbb{R}^{n \times n}. \quad (\text{II.18})$$

Similarly as in (II.16), we have

$$\|X_k - X\|_2 \leq \|V\|_2^2 \|V^{-1}\|_2^2 \rho(C(p))^{2k} \|X_0 - X\|_2.$$

This means that we obtain fast convergence by choosing  $p$  such that  $\rho(C(p)) < 1$  is as small as possible. We will discuss this later in more detail.

By varying the shifts  $p$  in (II.18) in every step, we obtain the *ADI iteration for Lyapunov equations*

$$X_k = C(p_k)X_{k-1}C(p_k)^H + \tilde{W}(p_k), \quad k \geq 1, \quad X_0 \in \mathbb{R}^{n \times n}, \quad p_k \in \mathbb{C}^-. \quad (\text{II.19})$$

**Remark:** The name alternating directions implicit comes from a different (historical) derivation of ADI for linear systems. To get the main idea for Lyapunov equations, consider the splitting of the Lyapunov operator

$$\mathcal{L}(X) = AX + XA^T = \mathcal{L}_1(X) + \mathcal{L}_2(X), \quad \mathcal{L}_1(X) = AX, \quad \mathcal{L}_2(X) = XA^T.$$

Obviously,  $\mathcal{L}_1(\cdot)$  and  $\mathcal{L}_2(\cdot)$  are commuting linear operators. It is possible to formulate an iteration working alternately on  $\mathcal{L}_1(\cdot)$  and  $\mathcal{L}_2(\cdot)$ , carrying out “half”-iteration steps for each operator:

$$\begin{aligned} (A + p_i I_n)X_{i-\frac{1}{2}} &= -X_{i-1}(A^T - p_i I_n) + W, \\ (A + p_i I_n)X_i^T &= -X_{i-\frac{1}{2}}^T(A^T - p_i I_n) + W. \end{aligned}$$

Rewriting this into a single step leads to (II.19).

We address two issues of the ADI iteration:

1. ADI requires, similar to the rational Krylov projection method, shift parameters that are crucial for a fast convergence. How to choose the shift parameters  $p_i$ ,  $i \geq 1$ ?
2. The iteration (II.19) is in its given form not feasible for large Lyapunov equations.

**The ADI Shift Parameter Problem** One can show, similarly to (II.16), that

$$\|X_k - X\|_2 \leq \|V\|_2^2 \|V^{-1}\|_2^2 \rho(M_k)^2 \|X_0 - X\|_2, \quad M_k := \prod_{i=1}^k C(p_i), \quad (\text{II.20})$$

where  $V$  is a transformation matrix diagonalizing  $A$  (assuming it is diagonalizable). The eigenvalues of the product of the Cayley transformations  $M_k$  are

$$\Lambda(M_k) = \left\{ \prod_{i=1}^k \frac{\lambda - \bar{p}_i}{\lambda + p_i} \mid \lambda \in \Lambda(A) \right\}.$$

Good shifts  $p_1^*, \dots, p_k^*$  should make  $\rho(M_k) < 1$  as small as possible. This motivates the ADI shift parameter problem

$$[p_1^*, \dots, p_k^*] = \operatorname{argmin}_{p_i \in \mathbb{C}^-} \max_{\lambda \in \Lambda(A)} \left| \prod_{i=1}^k \frac{\lambda - \bar{p}_i}{\lambda + p_i} \right|. \quad (\text{II.21})$$

In general, this is very hard to solve. For instance, in general,  $\rho(C(p))$  is not differentiable and the problem is very expensive, if  $A$  is a large matrix. However, there are some procedures that work well in practice:

- **Wachspress shifts:** Embed  $\Lambda(A)$  in an elliptic function region that depends on the parameters

$$\max_{\lambda \in \Lambda(A)} \operatorname{Re}(\lambda), \quad \min_{\lambda \in \Lambda(A)} \operatorname{Re}(\lambda), \quad \arctan \max_{\lambda \in \Lambda(A)} \left| \frac{\operatorname{Im}(\lambda)}{\operatorname{Re}(\lambda)} \right|$$

(or approximations thereof). Then, (II.21) can be solved by employing an elliptic integral.

- **Heuristic Penzl shifts:** If  $A$  is a large and sparse matrix,  $\Lambda(A)$  is replaced by a small number of approximate eigenvalues (e.g., Ritz values). Then (II.21) is solved heuristically.
- **Self-generating shifts:** If  $A$  is large and sparse, these shifts are based on projections of  $A$  with the data obtained by previous iterations. These shifts also make use of the right-hand side  $W$ .

**The Low-Rank ADI** For a low-rank version of ADI computing low-rank solution factors, consider one step of the dense iteration (II.19) and insert  $X_j = Z_j Z_j^H$ :

$$\begin{aligned} X_j &= C(p_j) X_{j-1} C(p_j)^H + \tilde{W}(p_j) \\ &= (A - \bar{p}_j I_n)(A + p_j I_n)^{-1} Z_{j-1} Z_{j-1}^H (A + p_j I_n)^{-H} (A - \bar{p}_j I_n)^H \\ &\quad - 2 \operatorname{Re}(p_j) (A + p_j I_n)^{-1} B B^T (A + p_j I_n)^{-H}. \\ \Rightarrow X_j &= Z_j Z_j^H, \quad Z_j = \left[ \sqrt{-2 \operatorname{Re}(p_j)} (A + p_j I_n)^{-1} B \quad (A - \bar{p}_j I_n)(A + p_j I_n)^{-1} Z_{j-1} \right]. \end{aligned}$$

With  $Z_0 = 0$  we find a low rank variant the ADI iteration (II.19) forming  $Z_j$  successively (grows by  $m$  columns in each step).

The drawback is that all columns are processed in every step which leads to quickly growing costs (in total  $jm$  linear systems have to be solved to get  $Z_j$ ).

However, there is a remedy to this problem. Obviously,

$$S_i = (A + p_i I_n)^{-1} \text{ and } T_j = (A - \bar{p}_j I_n)$$

commute for all  $i, j$  with each other and themselves (proof it yourself).

Now consider  $Z_j$  being the iterate after iteration step  $j$

$$Z_j = [\alpha_j S_j B \quad (T_j S_j) \alpha_{j-1} S_{j-1} B \quad \dots \quad (T_j S_j) \cdots (T_2 S_2) \alpha_1 S_1 B]$$

with  $\alpha_i = \sqrt{-2 \operatorname{Re}(p_i)}$ . The order of application of the shifts is not important, and we reverse their application to obtain the following alternative iterate

$$\begin{aligned} \tilde{Z}_j &= [\alpha_1 S_1 B \quad \alpha_2 (T_1 S_1) S_2 B \quad \dots \quad \alpha_j (T_1 S_1) \cdots (T_{j-1} S_{j-1}) S_j B] \\ &= [\alpha_1 S_1 B \quad \alpha_2 (T_1 S_2) S_1 B \quad \dots \quad \alpha_j (T_{j-1} S_j) (T_{j-2} S_{j-1}) \cdots (T_1 S_2) S_1 B] \\ &= [\alpha_1 V_1 \quad \alpha_2 V_2 \quad \dots \quad \alpha_j V_j], \\ V_1 &= S_1 B, \quad V_i = T_{i-1} S_i V_{i-1}, \quad i = 1, \dots, j. \end{aligned}$$

We have  $X_j = \tilde{Z}_j \tilde{Z}_j^H$ , but in this formulation only the new columns are processed. Even more structure is revealed by the Lyapunov residual.

**Theorem II.18:** The residual at step  $j$  of (II.19), started with  $X_0 = 0$ , is of rank at most  $m$  and given by

$$\begin{aligned} R_j &:= AZ_j Z_j^H + Z_j Z_j^H A^T + BB^T = W_j W_j^H, \\ W_j &= M_j B = C(p_j) W_{j-1} = W_{j-1} - 2 \operatorname{Re}(p_j) V_j, \quad W_0 := B, \end{aligned}$$

where  $M_j := \prod_{i=1}^j C(p_i)$ . Moreover, it holds  $V_j = (A + p_j I_n)^{-1} W_{j-1}$ .

*Proof.* We have

$$\begin{aligned} R_j &= AX_j + X_j A^T + BB^T = A(X_j - X) + (X_j - X) A^T \quad (\text{by (II.11)}) \\ &= AM_j(X_0 - X) M_j^H + M_j(X_0 - X) M_j^H A^T \\ &= -M_j AX M_j^H - M_j X A^T M_j^H \\ &= -M_j (AX + X A^T) M_j = M_j B B^T M_j. \end{aligned}$$

Moreover, it holds

$$\begin{aligned} V_j &= T_{j-1} S_j V_{j-1} = T_{j-1} S_j T_{j-2} S_{j-1} V_{j-2} = \dots = \\ &= S_j \left( \prod_{k=1}^{j-1} T_k S_k \right) B = S_j M_{j-1} B = (A + p_j I_n)^{-1} W_{j-1}, \end{aligned} \quad (\text{II.22})$$

and

$$W_j = M_j B = S_j T_j W_{j-1} = W_{j-1} - 2 \operatorname{Re}(p_j) S_j W_{j-1} = W_{j-1} - 2 \operatorname{Re}(p_j) V_j.$$

□

**Algorithm 3** Low-rank ADI (LR-ADI) iteration for Lyapunov equations

**Input:**  $A, B$  from (II.11), shifts  $P = \{p_1, \dots, p_{\maxiter}\} \subset \mathbb{C}^-$ , residual tolerance  $\text{tol}$ .

**Output:**  $Z_k$  such that  $X = Z_k Z_k^H$  (approx.) solves (II.11).

- 1: Initialize  $j = 1, W_0 := B, Z_0 := []$ .
- 2: **while**  $\|W_{j-1}\|_2 \geq \text{tol}$  **do**
- 3:   Set  $V_j := (A + p_j I_n)^{-1} W_{j-1}$ .
- 4:   Set  $W_j := W_{j-1} - 2 \operatorname{Re}(p_j) V_j$ .
- 5:   Set  $Z_j := [Z_{j-1} \quad \sqrt{-\operatorname{Re}(p_j)} V_j]$ .
- 6:   Set  $j := j + 1$ .
- 7: **end while**

Thank to the above theorem, the norm of the Lyapunov residual norm can be cheaply computed via  $\|R_j\|_2 = \|W_j W_j^H\|_2 = \|W_j\|_2^2$ . All this leads to Algorithm 3. Again, the major work is solving the LS  $(A + p_j I_n) V_j = W_{j-1}$  in each step, which is efficiently possible for large, sparse  $A$  (cf. Introduction).

Algorithm 3 produces complex low-rank factors, if some of the shifts are complex, which might be required for problems with nonsymmetric  $A$ . Ensuring that  $Z_j \in \mathbb{R}^{n \times n_j}$  and limiting the number of complex operations can be achieved by assuming that for a complex shift  $p_i$  we have  $p_{i+1} = \bar{p}_i$  (see handout)