CHAPTER 1

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

$$Ax = b, \tag{I.1}$$

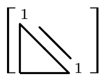
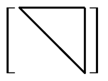
with $A \in \mathbb{C}^{n\times n}\,(\mathbb{R}^{n\times n}),\quad b \in \mathbb{C}^n\,(\mathbb{R}^n)$. The linear system (I.1) admits a unique solution, if and only if

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

Numerical methods for small and dense $A \in \mathbb{C}^{n\times n}$

Gaussian Elimination (LU-factorization):

We decompose A such that

$$A = LU, \quad L =$$  $, \quad U =$  .

We obtain, that

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

Hence, we solve (I.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.

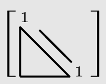
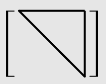
QR-decomposition:

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

$$A = \begin{bmatrix} * & \cdots & * \\ \vdots & \ddots & \\ * & & * \end{bmatrix} =$$   .

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

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

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 $Q_q := [q_1]$.
2: for $j = 1, 2, \dots$ do
3:     Set $z = Aq_j$.
4:     Set $w = z - Q_j(Q_j^{\mathrm{H}} z)$.
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: $x = A\backslash b$ or $\mathrm{lu}(A)$-routine. (Never use $\mathrm{inv}(A)$!)

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

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

and each x_k, $k \geqslant 1$ 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 $\mathrm{range}(Q_k) = \mathcal{Q}_k$ for a matrix $Q_k \in \mathbb{C}^{n\times k}$.

A good choice of the subspace is the Krylow-subspace

$$\mathcal{Q}_k = \mathcal{K}_k(A, b) = \mathrm{span}\{b, Ab, \dots, A^{k-1}b\}. \tag{I.2}$$

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 $r = b - Ax_k \perp \mathrm{range}(Q_k) \quad \Leftrightarrow \quad (Q_k^{\mathrm{H}} A Q_k) y_k = Q_k^{\mathrm{H}} b$.

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

- Minimize the residual:

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

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 $A = A^{\mathrm{H}}$: Arnoldi $\rightsquigarrow$ Lanczos)

In practice: Convergence acceleration by *preconditioning*:

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

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 $\Lambda(A) = \{\lambda_1, \ldots, \lambda_n\}$ is called the *spectrum of* A.

Small, dense problems:

Computing the Jordan-Normal-Form (JNF)

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

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 $Q \in \mathbb{C}^{n,n}$ ($Q^{\mathrm{H}}Q = I$), such that

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

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 $[\mathrm{Q}, \mathrm{R}] = \mathrm{schur}(\mathrm{A})$. Additionally, the routine $\mathrm{eigs}(\mathrm{A})$ 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 treaded with the Arnoldi-process and projections on the Krylov-subspace $\mathcal{K}_k(A, b) = \mathrm{range}(Q_k)$. 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:

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

which means (μ, y_k) are the eigenpairs of the $k \times k$-dimensional eigenvalue problem for $Q_k^{\mathrm{H}} A Q_k$. This small eigenvalue problem is solvable by the Francis-QR-method. This is the basis of the $\mathrm{eigs}(\mathrm{A})$ 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 $F : \mathbb{R}^{q\times r} \to \mathbb{R}^{p\times s}$, $C \in \mathbb{R}^{p\times s}$ 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:

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

   b) discrete time:

$$\begin{aligned} A^TXA - X - (A^TXB)(R + B^TXB)^{-1}(B^TXA) \\ + C^TQC = 0, \quad X = X^T. \end{aligned}$$

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 $A,\ Q \in C([t_0, t_\mathrm{f}], \mathbb{R}^{n\times n})$, and $X \in C^1([t_0, t_\mathrm{f}], \mathbb{R}^{n\times n})$ with $Q(t) = Q(t)^T \geqslant 0$ and $X(t) = X(t)^T$ for all $t \in [t_0, t_\mathrm{f}]$ 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 $C \in \mathbb{C}^{p\times s}$, $i = 1, \ldots, k$ be given. An equation of the form

$$\sum_{i=1}^{k} A_i X B_i = C \tag{II.2}$$

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 $X = \begin{bmatrix} x_1 & \ldots & x_m \end{bmatrix} = \begin{bmatrix} x_{11} & \ldots & x_{1m} \\ \vdots & & \vdots \\ x_{n1} & \ldots & x_{nm} \end{bmatrix} \in \mathbb{C}^{n\times m}$ and $Y \in \mathbb{C}^{p\times q}$

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

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

b) the Kronecker product is given by

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

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

$$\operatorname{vec}(\mathcal{TOR}) = \left(\mathcal{R}^T \otimes \mathcal{T}\right) \operatorname{vec}(\mathcal{O})$$

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

Proof. Exercise. $\square$

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

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

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

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

are non-zero.

In the following we will focus on the case $k \leqslant 2$ 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 $\Lambda(A) = \{\lambda_1, \ldots, \lambda_n\}$, $\Lambda(B) = \{\mu_1, \ldots, \mu_m\}$ be given. For a bivariate polynomial $p(x, y) = \sum\limits_{i,j=0}^{k} c_{ij} x^i y^j$ 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

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

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 $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{m\times m}$, $C, X \in \mathbb{C}^{n\times m}$:

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

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

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

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

d) discrete-time Lyapunov equations $\to$ 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

$$X = -\int_0^\infty \mathrm{e}^{At} W \mathrm{e}^{Bt} \mathrm{d}t$$

Proof. Exercise. □

From now on

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

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 rank$[B, AB, \dots A^{n-1}B] = n$.

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

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

Proof. We first prove that rank$[A - \lambda I, B] = n \quad \forall \lambda \in \mathbb{C}$ is equivalent to Definition II.8. Assuming that 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 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

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

$$\tilde{B} = T^{-1}B = \begin{bmatrix} \tilde{B}_1 \\ 0 \end{bmatrix} \tag{II.5}$$

Let λ 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 rank$[A - \lambda i, B]$ not full. The proof of the equivalence is basically also done within this proof. □

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

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

$$X = \int\limits_0^\infty \mathrm{e}^{At} BB^T \mathrm{e}^{A^*t} \mathrm{d}t$$

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

$$0 > -y^* BB^T y = y^* AXy + y^* XA^* y = (\lambda + \bar{\lambda}) y^* Xy$$

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

□

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 $\Lambda(A) \cap \Lambda(-A^H) = \varnothing$ 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

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

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

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

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

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 $\operatorname{diag}(T) \cap \operatorname{diag}\left(-T^H\right) \neq \varnothing$ 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 $k := k - 1$.
11: end while
12: Solve (II.7c) with $T_1 = T(1,1)$ and $\hat{W}_1 = \tilde{W}(1,1)$.
13: Set $X := Q X Q^H$.
```

Now we get

$$X_3 = \frac{\tilde{W}_3}{T_3 + \overline{T}_3}, \tag{II.7a}$$

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

$$T_1 X_2 + X_2 \overline{T_3} = \tilde{W}_2 - T_2 X_3 =: \hat{W}_2, \tag{II.7b}$$

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

$$(\overline{T_3} I_{n-1} + T_1) X_2 = \hat{W}_2,$$

and can easily be solved by backward substitution. Its solution always exists since $\Lambda(T_1) \cap \left\{-\overline{T_3}\right\} = \varnothing$. It remains to solve the smaller $(n-1) \times (n-1)$ sized 'triangular' Lyapunov equation

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

which is also solvable since $\Lambda(T_1) \cap \Lambda(-T_1^H) = \varnothing$ 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.

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

where for $i = 1, \dots, k$, $T_{ii} \in \mathbb{R}^{1\times 1}$ (corresponding to a real eigenvalue of A) or $T_{ii} = \left[\begin{smallmatrix} \alpha_i & \beta_i \\ -\beta_i & \alpha_i \end{smallmatrix}\right] \in \mathbb{R}^{2\times 2}$ (corresponding to a pair of complex conjugate eigenvalues $\alpha_i \pm \mathrm{i}\beta_i$ 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 2×2 block, i. e., $T_3 = \left[\begin{smallmatrix} t_1 & t_2 \\ t_3 & t_4 \end{smallmatrix}\right]$. 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

$$\begin{cases} w_1 = t_1x_1 + t_2x_2 + t_1x_1 + t_2x_2 = 2(t_1x_1 + t_2x_2), \\ w_2 = t_1x_2 + t_2x_3 + t_3x_1 + t_4x_2 = t_3x_1 + (t_1 + t_4)x_2 + t_2x_3, \\ w_3 = t_3x_2 + t_4x_3 + t_3x_2 + t_4x_3 = 2(t_3x_2 + t_4x_3). \end{cases}$$

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

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

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

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

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 $A = Q^H TQ$, analogously to (II.6). So our starting point is

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

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

$$\tilde{U} = \left[\,\diagdown\!\!\Big|\,\right] = \begin{bmatrix} U_1 & u \\ 0 & \tau \end{bmatrix}, \quad U_1 \in \mathbb{C}^{n-1\times n-1},\ u \in \mathbb{C}^{n-1},\ 0 < \tau \in \mathbb{R}.$$

Hammarlings method computes (similar to B.S.) first τ (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 $U \leftarrow QU$. 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, $k \geqslant 0$ converging to the true solution, i.e., $\lim\limits_{k\to\infty} X_k = X$. For instance:

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