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

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

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
- ...

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

Gaussian Elimination (LU-factorization):

We decompose A such that

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

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

$$A = \begin{bmatrix} * & \cdots & * \\ \vdots & \ddots & \\ * & & * \end{bmatrix} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \begin{bmatrix} \ddots & & \\ & \ddots & \\ & & \ddots \end{bmatrix}.$$

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** $j = 1, 2, \ldots$ **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 $\widehat{x} \approx x$ is sufficient. Hence, we generate a sequence $x_1, x_2, \ldots, x_k$ 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

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

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:

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

  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)

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

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:

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

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 $A,\ Q \in C([t_0, t_\mathrm{f}], \mathbb{R}^{n\times n})$, 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

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

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

Proof. Exercise. □

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

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

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 $p(x,y) = \sum_{i,j=0}^{k} c_{ij}x^i y^j$ 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 → 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

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

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

$$X = \int_0^\infty \mathrm{e}^{At} BB^T \mathrm{e}^{A^* t} \mathrm{d}t$$

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 $T = Q^H AQ$ with the QR algorithm.
2: **if** $\operatorname{diag}(T) \cap \operatorname{diag}(-T^H) \neq \varnothing$ **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 $\tilde{W} = \tilde{W}(1:k, 1:k) - T_2 X_2^H - X_2 T_2^H$
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 $X := QXQ^H$.

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 $\Lambda(T_1) \cap \{-\overline{T_3}\} = \varnothing$. 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 $\lambda_{1,2} = a \pm \mathrm{i}b$ 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 $i = 1, \ldots, k-1$.

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

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

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

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

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

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

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

$$\tilde{U} = \begin{bmatrix} \ddots \end{bmatrix} = \begin{bmatrix} U_1 & u \\ 0 & \tau \end{bmatrix}, \quad U_1 \in \mathbb{C}^{n-1\times n-1},\ u \in \mathbb{C}^{n-1},\ 0 < \tau \in \mathbb{R}.$$

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., $\lim_{k\to\infty} X_k = X$. 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

$$AX + XA^T = -BB^T, \tag{II.11}$$

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 $X = X^T \geqslant 0$ and $\operatorname{rank}(X) = n$.

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:

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

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

$$\frac{\lambda_{km+1}(X)}{\lambda_1(X)} \leqslant \|V\|_2^2 \left\|V^{-1}\right\|_2^2 \rho(M_k)^2$$

for any choice of shift parameters p_k used to construct

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

(in particular, the optimal ones).

In the Theorem above the spectral radius ρ of a matrix is used:

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

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 $\|V\|_2 \left\|V^{-1}\right\|_2 = 1$ 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 $\mathcal{Q}_k = \operatorname{im} Q_k$ with $Q_k^T Q_k = I_{km}$, i. e., $X \approx X_k = Q_k Y_k Q_k^T$ for some $Y_k \in \mathbb{R}^{mk \times mk}$. Impose a Galerkin condition

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

where

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

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

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

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**

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

A matrix Q_k with orthonormal columns spanning $\mathcal{Q}_k$ can be generated by a block Arnoldi process, i. e. in the kth iteration we have $Q_k = \begin{bmatrix} V_1 & \ldots & V_k \end{bmatrix}$ fulfilling (assuming there is no breakdown in the process)

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

where

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

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 $\Lambda(H_k) \cap \Lambda(-H_k) = \varnothing$. 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

$$\left\| R(Q_k Y_k Q_k^T) \right\|_{\mathrm{F}} = \sqrt{2} \left\| H_{k+1,k} E_k^T Y_k \right\|_{\mathrm{F}}.$$

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**

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

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

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

This leads to the following procedure for getting the next shift

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

where ∂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 $Q_k \in \mathbb{R}^{n\times mk}$ 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 $x^H\big(A + A^T\big)x < 0$ for all $x \in \mathbb{C}^n$. Then we have

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

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

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

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

Low-rank ADI

Consider the discrete-time Lyapunov equations

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

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

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

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

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

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

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

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

Proof. Exercise. □

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

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

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*)

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

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

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

Similarly as in (II.16), we have

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

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*

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

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

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

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, $i \geqslant 1$?
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

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

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

$$\Lambda(M_k) = \left\{ \prod_{i=1}^{k} \frac{\lambda - \overline{p_i}}{\lambda + p_i} \;\middle|\; \lambda \in \Lambda(A) \right\}.$$

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

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

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

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

  (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 $X_j = Z_j Z_j^H$:

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

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,

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

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

Now consider Z_j being the iterate after iteration step j

$$Z_j = \begin{bmatrix} \alpha_j S_j B & (T_j S_j)\alpha_{j-1} S_{j-1} B & \dots & (T_j S_j)\cdots(T_2 S_2)\alpha_1 S_1 B \end{bmatrix}$$

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

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

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

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

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

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

Moreover, it holds

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

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 $P = \{p_1, \ldots, p_{\text{maxiter}}\} \subset \mathbb{C}^-$, residual tolerance 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** $\|W_{j-1}\|_2 \geqslant \text{tol}$ **do**
3: Set $V_j := (A + p_j I_n)^{-1} W_{j-1}$.
4: Set $W_j := W_{j-1} - 2\,\text{Re}(p_j)\, V_j$.
5: Set $Z_j := \begin{bmatrix} Z_{j-1} & \sqrt{-\text{Re}(p_j)} V_j \end{bmatrix}$.
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 $p_{i+1} = \overline{p_i}$ (see handout)

II.3 Algebraic Riccati Equations

In this section we concentrate on the continuous time algebraic Riccati equation (ARE) briefly introduced in Chapter II.1 as one important representative of nonlinear matrix equations:

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

Here, we simplify the representation to

$$F + A^TX + XA - XGX = 0, \quad G \geqslant 0, \quad X = X^T, \tag{II.23}$$

where A, $F = F^T$, $G = G^T \in \mathbb{R}^{n\times n}$.

II.3.1 Hamiltonian Matrices and the ARE

Define the matrix

$$H = \begin{bmatrix} A & -G \\ -F & -A^T \end{bmatrix} \in \mathbb{R}^{2n\times 2n}.$$

If X solves the ARE, then we have

$$\begin{bmatrix} A & -G \\ -F & -A^T \end{bmatrix}\begin{bmatrix} I_n & 0 \\ X & I_n \end{bmatrix} = \begin{bmatrix} I_n & 0 \\ X & I_n \end{bmatrix}\begin{bmatrix} A-GX & -G \\ 0 & -A^T+XG \end{bmatrix},$$

which means that

$$H\begin{bmatrix} I_n \\ X \end{bmatrix} = \begin{bmatrix} I_n \\ X \end{bmatrix}(A-GX),$$

which means that $\operatorname{span}\left\{\left[\begin{smallmatrix} I_n \\ X \end{smallmatrix}\right]\right\}$ is an H-invariant subspace and $\Lambda(A-GX) \subset \Lambda(H)$.

Assume on the other hand, that

$$H\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix}M \quad \text{for} \quad U, V, M \in \mathbb{R}^{n\times n},$$

in particular, $\Lambda(M) \subset \Lambda(H)$. Then $\operatorname{span}\left\{\left[\begin{smallmatrix} U \\ V \end{smallmatrix}\right]\right\}$ is an H-invariant subspace. Now assume that U is invertible. Then we find

$$AU - GV = UM \overset{U^{-1}\ \exists}{\Leftrightarrow} U^{-1}AU - U^{-1}GV = M.$$

Moreover, we have

$$-FU - A^TV = VM = VU^{-1}AU - VU^{-1}GV.$$

A right-multiplication by U^{-1} then yields

$$-F - A^T V U^{-1} = V U^{-1} A - V U^{-1} G V U^{-1}.$$

With $X := VU^{-1}$ this finally results in

$$0 = F + A^T X + XA - XGX.$$

If we can ensure that U is invertible, then computing an invariant subspace for H provides a solution for the ARE. However, recall that in contrast to linear matrix equations, solutions are (except in some special cases) not unique. In practice one is interested in a stabilizing solution $X_* := VU^{-1}$, i. e., $\Lambda(A - GX_*) \subset \mathbb{C}^-$.

Remark: Why a stabilizing solution? Consider the *linear quadratic regulator* problem:

$$\min \mathcal{J}(u(t)) = \frac{1}{2} \int_0^\infty \|u(t)\|^2 + \|x(t)\|^2 dt$$
$$\text{subject to} \quad \dot{x}(t) = Ax(t) + Bu(t), x(t_0) = x_0 \in \mathbb{R}^n.$$

for given $A \in \mathbb{R}^{n\times n}$ possibly unstable, $B \in \mathbb{R}^{n\times m}$. Such problems are an important topic for control theory and widely used in practice to stabilize technical systems.

One can show that, under certain conditions, a solution of this *optimal control problem* is given by $u_*(t) = -BB^T X_* x(t)$, where X_* is the stabilizing solution of an ARE similar to (II.23).

So the question arises, which choice of the invariant subspace results in a symmetric and stabilizing solution. For this we analyze the matrix H in more detail, which turns out to be a Hamiltonian matrix.

Definition II.19 (Hamiltonian matrix)**:** Define

$$J := \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} \in \mathbb{R}^{2n\times 2n}. \tag{II.24}$$

A matrix $H \in \mathbb{R}^{2n\times 2n}$ is called *Hamiltonian* if

$$(HJ)^T = HJ.$$

We denote the set of all real Hamiltonian $2n \times 2n$ matrices by $\mathbb{H}_n$.

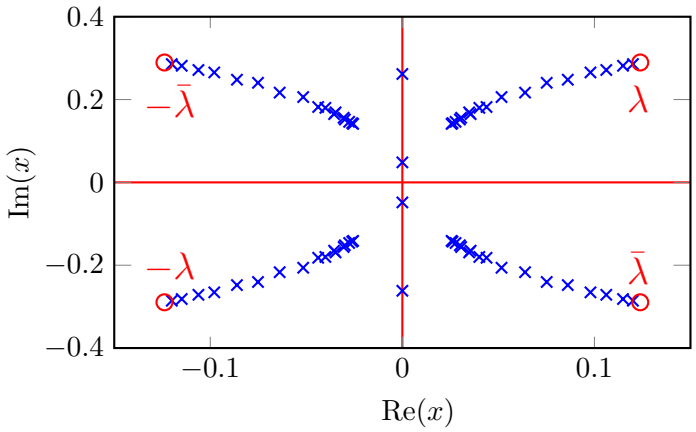


Figure II.1: Eigenvalues of a real Hamiltonian matrix

Proposition II.20: The following statements are equivalent:

a) H is Hamiltonian.

b) $H = JS$ for some matrix $S = S^T \in \mathbb{R}^{2n\times 2n}$.

c) It holds $(JH)^T = JH$.

d) H has the block structure

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & -H_{11}^T \end{bmatrix} \tag{II.25}$$

for $H_{11} \in \mathbb{R}^{n\times n}$, $H_{12} = H_{12}^T \in \mathbb{R}^{n\times n}$, and $H_{21} = H_{21}^T \in \mathbb{R}^{n\times n}$.

Proof. Exercise. □

Proposition II.21 (Hamiltonian spectrum)**:** Let $H \in \mathbb{H}_n$ and p_H the characteristic polynomial of H. Then the following statements are satisfied:

a) It holds $p_H(\lambda) = p_H(-\lambda)$ for all $\lambda \in \mathbb{C}$.

b) If $p_H(\lambda) = 0$, then $p_H(-\lambda) = p_H(-\overline{\lambda}) = p_H(\overline{\lambda}) = 0$ for $\lambda \in \mathbb{C}$.

Proof. Exercise. □

Proposition II.21 states that the spectrum of every real Hamiltonian matrix is symmetric with respect to the real and imaginary axis, see also Figure II.1.

II.3.2 Characterization of Stabilizing Solutions

Recall that we have started at

$$\begin{bmatrix} A & -G \\ -F & -A^T \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix} M, \quad \Lambda(M) \subset \Lambda(H),$$

where U is assumed to be invertible. From the first row we see

$$AU - GV = UM.$$

Assuming that U is invertible and a multiplication with U^{-1} from the right gives $A - GX = UMU^{-1}$, where $X := VU^{-1}$. Thus, we have $\Lambda(A - GX) = \Lambda(M)$. In particular, $A - GX$ is asymptotically stable if and only if $\Lambda(M) \subset \mathbb{C}^-$. This means that $\operatorname{span}\left\{\left[\begin{smallmatrix} U \\ V \end{smallmatrix}\right]\right\}$ is the H-invariant subspace corresponding to $\Lambda(H) \cap \mathbb{C}^-$.

First we show that stabilizing solutions (in case they exist) are unique.

Lemma II.22: The ARE (II.23) has at most one stabilizing solution.

Proof. If X_* is a stabilizing solution of (II.23) then $X_* = VU^{-1}$, where

$$\operatorname{span}\left\{\begin{bmatrix} U \\ V \end{bmatrix}\right\} = \operatorname{span}\left\{\begin{bmatrix} I_n \\ X_* \end{bmatrix}\right\}$$

is the invariant subspace of H associated with its eigenvalues in $\mathbb{C}^-$. If there exists a second stabilizing solution $\tilde{X}_*$, then

$$\operatorname{span}\left\{\begin{bmatrix} I_n \\ X_* \end{bmatrix}\right\} = \operatorname{span}\left\{\begin{bmatrix} I_n \\ \tilde{X}_* \end{bmatrix}\right\},$$

implying that $X_* = \tilde{X}_*$. □

We still don't know when a stabilizing solution exists. For this recall a weaker concept of controllability (Def. II.8) is useful.

Definition II.23: We call (A, B) stabilizable if $\operatorname{rank}\begin{bmatrix} \lambda I_n - A & B \end{bmatrix} = n \quad \forall \lambda \in \overline{\mathbb{C}^+} := \{\lambda \in \mathbb{C} : \operatorname{Re}(\lambda) \geqslant 0\}$

The dual concept is detectability: (A, C) detectable if (A^T, C^T) stabilizable.

The following theorem gives an equivalent characterization for the existence of stabilizing solutions.

Theorem II.24: The ARE (II.23) has a stabilizing solution X_* if and only if (A, G) is stabilizable and the matrix H does not have imaginary eigenvalues.

It remains to check under which conditions there are no eigenvalues of H on the imaginary axis. A sufficient condition is the following.

Theorem II.25: Let (A, G) be stabilizable and (A, F) be detectable with $F,\ G \geqslant 0$. Then the Hamiltonian matrix H does not have imaginary eigenvalues.

Combining the above findings we can conclude the following theorem.

Theorem II.26: Consider the ARE (II.23) with $F \geqslant 0$. Let (A, G) be stabilizable and (A, F) be detectable. Further let $\operatorname{span}\left\{\left[\begin{smallmatrix} U \\ V \end{smallmatrix}\right]\right\}$ with $U,\ V \in \mathbb{R}^{n\times n}$ be an H-invariant subspace corresponding to the eigenvalues of H in the open left half-plane. Then $X_* = X_*^T = VU^{-1}$ is the unique stabilizing solution of (II.23).

We analyze the structure of the stabilizing solution in more detail. First we show that for AREs with $F \geqslant 0$ the stabilizing solution is positive semi-definite.

Proposition II.27: If $F \geqslant 0$, then the stabilizing solution X_* of the ARE (II.23) (if it exists) is positive semi-definite. Furthermore, if (A^T, F) is controllable, then $X_* > 0$.

Proof. If X is any symmetric solution of the ARE, we obtain

$$(A - GX)^T X + X(A - GX) = -XGX - F.$$

With $\hat{A} := A - GX$ and $\hat{F} := -XGX - F$ it holds

$$\hat{A}^T X + X\hat{A} = \hat{F}.$$

If $X = X_*$ is stabilizing, then $\Lambda(\hat{A}) \subset \mathbb{C}^-$. Since $F \geqslant 0$, we have $\hat{F} \leqslant 0$ and thus $X_* \geqslant 0$.

If (A^T, F) is controllable, then so is$(\hat{A}, \hat{F})$: If $\hat{A}v = \lambda v$ and $\hat{F}v = 0$ for $v \neq 0$, then we get $v^H \hat{F} v = 0$ and therefore, $GX_* v = 0$ and $Fx = 0$. The former implies $Av = \lambda v$. This yields $v = 0$, since (A^T, F) is controllable. This implies $X_* > 0$ by Theorem II.10**a)**. □

Algorithm 4 Schur vector method for solving the ARE

Input: $H = \begin{bmatrix} A & -G \\ -F & -A^T \end{bmatrix}$ corresponding to (II.23).
Output: the stabilizing solution X_* of (II.23).

1: Apply the standard QR iteration to H to compute a Schur decomposition.
2: Sort the eigenvalues according to (II.26) via orthogonal similarity transformations.
3: Solve the n linear systems $X_* Q_{11} = Q_{21}$.

II.3.3 Direct Numerical Solution Methods

Now we discuss direct numerical solution algorithms for the ARE (II.23). We assume that all assumptions of Theorem II.26 are satisfied, such that a unique stabilizing and positive semi-definite solution X_* exists. We are interested in computing this solution.

The Schur Vector Method

From Theorem II.26 we know that the Hamiltonian matrix $H = \begin{bmatrix} A & -G \\ -F & -A^T \end{bmatrix}$ has exactly n eigenvalues in $\mathbb{C}^-$ and exactly n eigenvalues in $\mathbb{C}^+$.

The simplest idea consists of using the real Schur decomposition to compute the H-invariant subspace via

$$Q^T H Q = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} =: T \tag{II.26}$$

where T_{11}, T_{22} are in real Schur form and $\Lambda(T_{11}) \subset \mathbb{C}^-$. By partitioning

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}$$

as T in (II.26), we find that $\operatorname{span}\left\{ \begin{bmatrix} Q_{11} \\ Q_{21} \end{bmatrix} \right\}$ is the desired subspace. The computation of the stabilizing solution X_* is summarized in Algorithm 4.

This method is very simple to implement and all steps numerically backward stable. On the other hand, the Hamiltonian structure not exploited. This means that the double symmetry of the Hamiltonian spectrum may be lost in T due to round-off errors. In particular, the eigenvalues close to the imaginary axis may move to the wrong half-plane. In this case the computation of X_* may break down. Therefore, we are interested in algorithms, that exploit and preserve the Hamiltonian structure during the computation.

Hamiltonian Schur Methods

Now we discuss structure-preserving methods for the Hamiltonian eigenvalue problem. For this we need to define the class of structure-preserving transformations for which we need symplectic matrices.

Definition II.28 (Symplectic matrix)**:** A matrix $S \in \mathbb{R}^{2n\times 2n}$ is called *symplectic* if

$$S^T J S = J,$$

where J is as in (II.24).

It can be shown that symplectic similarity transformations preserve the Hamiltonian structure. This is stated in the next lemma.

Lemma II.29: If $H \in \mathbb{R}^{2n\times 2n}$ is Hamiltonian and $S \in \mathbb{R}^{2n\times 2n}$ is symplectic, then $\tilde{H} := S^{-1}HS \in \mathbb{R}^{2n\times 2n}$ is Hamiltonian.

Proof. Ex. □

In order to have transformations that do not increase the condition number of the problem we aim at symplectic similarity transformations that are additionally orthogonal. Orthogonal symplectic matrices have a certain block structure given in the next lemma.

Lemma II.30: Every orthogonal symplectic matrix $U \in \mathbb{R}^{2n\times 2n}$ is given as

$$U = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix} \quad \text{for } U_1, U_2 \in \mathbb{R}^{n\times n}.$$

Proof. Exercise. □

Using orthogonal symplectic transformations we can now formulate the following result which gives us a Hamiltonian Schur form.

Theorem II.31 (Hamiltonian Schur form)**:** Let $H \in \mathbb{R}^{2n\times 2n}$ be a Hamiltonian matrix with $\Lambda(H) \cap \mathrm{i}\mathbb{R} = \varnothing$. Then there exist an orthogonal symplectic $U \in \mathbb{R}^{2n\times 2n}$ and a Hamiltonian matrix $T \in \mathbb{R}^{2n\times 2n}$ such that

$$U^T H U = T = \begin{bmatrix} T_1 & T_2 \\ 0 & -T_1^T \end{bmatrix}, \tag{II.27}$$

where T_1 is in real Schur form and $T_2 = T_2^T \in \mathbb{R}^{n\times n}$.

The goal now is to devise an algorithm for computing (II.27). This is not easy! For preserving Ham. structure by unitary symplectic trafos, all major steps in the standard QR-Algorithm for the normal Schur form have to be modified accordingly (Hessenberg-reductions, QR-factorization, ...). Only in the recent years this was achieved completely $\rightsquigarrow$ literature.

II.3.4 Iterative Solution of the ARE – The Newton-Kleinman Iteration

Now we consider the ARE

$$\mathcal{R}(X) = F + A^T X + XA - XGX = 0. \tag{II.28}$$

Assume that (A, G) is stabilizable, (A, F) is detectable, and $F, G \geqslant 0$ such that there exists a unique stabilizing solution X_* of the ARE. We now consider (II.28) as a nonlinear system of equations and apply Newton's method. For this, we need to evaluate the (Fréchet) derivative of $\mathcal{R}(X)$ with respect to X.

Definition II.32 (Fréchet differentiability, Fréchet derivative)**:** Let $(\mathcal{X}, \|\cdot\|_{\mathcal{X}})$ and $(\mathcal{Y}, \|\cdot\|_{\mathcal{Y}})$ be two normed linear spaces and let $\mathcal{U} \subset \mathcal{X}$ be an open subset. A linear operator $\mathcal{F} : \mathcal{U} \to \mathcal{Y}$ is called *Fréchet differentiable* at $X \in \mathcal{U}$ if there exists a bounded linear operator $\mathcal{F}'(X) : \mathcal{X} \to \mathcal{Y}$ such that

$$\lim_{\|N\|_{\mathcal{X}} \to 0} \frac{1}{\|N\|_{\mathcal{X}}} \left\| \mathcal{F}(X+N) - \mathcal{F}(X) - (\mathcal{F}'(X))(N) \right\|_{\mathcal{Y}} = 0.$$

The operator $\mathcal{F}'(X)$ is called *Fréchet derivative* of $\mathcal{F}$ at X. The map $\mathcal{F}' : \mathcal{U} \to \mathcal{L}(\mathcal{X}, \mathcal{Y})$ with $X \mapsto \mathcal{F}'(X)$ is called *Fréchet derivative* of $\mathcal{F}$ on $\mathcal{U}$.

Let us see whether $\mathcal{R}(\cdot)$ is Fréchet differentiable and (if yes) determine its Fréchet derivative. If the Fréchet derivative exists it is given by

$$\begin{aligned}
(\mathcal{R}'(X))(N) &= \lim_{h\to 0} \frac{1}{h}(\mathcal{R}(X+hN) - \mathcal{R}(X)) \\
&= \lim_{h\to 0} \frac{1}{h} \big(F + A^T(X+hN) + (X+hN)A \\
&\qquad -(X+hN)G(X+hN) - (F + A^T X + XA - XGX)\big) \\
&= \lim_{h\to 0} \frac{1}{h}(hA^T N + hNA - hXGN - hNGX - h^2 NGN) \\
&= \lim_{h\to 0} (A^T N + NA - XGN - NGX - hNGN) \\
&= A^T N + NA - XGN - NGX \\
&= (A - GX)^T N + N(A - GX).
\end{aligned}$$

Algorithm 5 Newton's method for the algebraic Riccati equation

Input: A, F, G as in (II.28) and initial value X_0 such that $\Lambda(A - GX_0) \subset \mathbb{C}^-$.
Output: Stabilizing solution X_* solving (II.28).

1: **for** $j = 1, 2 \ldots,$ **do**
2: Set $A_j := A - GX_{j-1}$.
3: Solve $A_j^T N_{j-1} + N_{j-1} A_j = -\mathcal{R}(X_{j-1})$ for N_{j-1}.
4: Set $X_j := X_{j-1} + N_{j-1}$.
5: **end for**

Algorithm 6 Newton-Kleinman iteration for the algebraic Riccati equation

Input: A, F, G as in (II.28) and initial value X_0 such that $\Lambda(A - GX_0) \subset \mathbb{C}^-$.
Output: Stabilizing solution X_* solving (II.28).

1: **for** $j = 1, 2, \ldots$ **do**
2: Set $A_j := A - GX_{j-1}$ and $F_j := -F - X_{j-1} G X_{j-1}$.
3: Solve $A_j^T X_j + X_j A_j = -F_j$.
4: **end for**

In other words, the Fréchet derivative of a Riccati operator is a Lyapunov operator. Now the Newton iteration is given by

$$(\mathcal{R}'(X_{j-1}))(N_{j-1}) = -\mathcal{R}(X_{j-1}), \quad X_j = X_{j-1} + N_{j-1}, \quad j = 1, 2, \ldots$$

and the iteration is summarized in Algorithm 5. This formulation of the algorithm has the disadvantage that $\mathcal{R}(X_{j-1})$ is evaluated in every iteration. Therefore, let us revisit the computation of the update N_{j-1}. We know that

$$\begin{aligned}(A - GX_{j-1})^T N_{j-1} + N_{j-1}(A - GX_{j-1}) \\ = -F - A^T X_{j-1} - X_{j-1} A + X_{j-1} G X_{j-1}.\end{aligned} \tag{II.29}$$

Plugging in $N_{j-1} = X_j - X_{j-1}$ then gives

$$\begin{aligned}(A - GX_{j-1})^T (X_j - X_{j-1}) + (X_j - X_{j-1})(A - GX_{j-1}) \\ = -F - A^T X_{j-1} - X_{j-1} A + X_{j-1} G X_{j-1}.\end{aligned}$$

Some manipulations and rearrangements of the terms finally lead to

$$(A - GX_{j-1})^T X_j + X_j (A - GX_{j-1}) = -F - X_{j-1} G X_{j-1}. \tag{II.30}$$

This leads to Kleinman's formulation of the Newton iteration which is given in Algorithm 6. The question arises whether Algorithm 6 and Algorithm 5 converges to the right solution.

Theorem II.33: Consider the ARE (II.28) with stabilizable (A, G), detectable (A, F), and $F, G \geqslant 0$. Let X_* be its unique stabilizing solution. Let further $X_0 \in \mathbb{R}^{n\times n}$ be stabilizing, i. e., $\Lambda(A - GX_0) \subset \mathbb{C}^-$. Then the iterates X_j, $j = 1, 2, \ldots$ fulfill the following statements:

a) The matrix X_j is stabilizing.

b) It holds $X_* \leqslant \cdots \leqslant X_{j+1} \leqslant X_j \leqslant \cdots \leqslant X_1$.

c) It holds $\lim_{j\to\infty} X_j = X_*$.

d) The convergence is globally quadratic, i. e., there exists a constant $\gamma > 0$ such that

$$\|X_* - X_j\| \leqslant \gamma \|X_* - X_{j-1}\|^2, \quad j = 1, 2, \ldots.$$

Proof. a) Let X_* be the stabilizing solution of the ARE (II.28) which exists and is unique due to the assumptions. Now consider (II.30) and the Riccati equation of the solution [$F + A^T X_* + X_* A - X_* G X_* = 0$]:

$$A_{j-1}^T((X_j) - X_*) + ((X_j) - X_*)A_{j-1} = -(X_{j-1} - X_*)G(X_{j-1} - X_*). \quad \text{(II.31)}$$

Assume that X_{j-1} is stabilizing, i.e. $\Lambda(A_{j-1} = A - GX_{j-1}) \subset \mathbb{C}_-$ With $G \geqslant 0$ it follows

$$(X_j) - X_* \geqslant 0 \quad \text{(II.32)}$$

from Lemma II.10 a). Now

$$\begin{aligned} \text{(II.31)} \quad - \quad & [N_{j-1}G((X_j) - X_*) + ((X_j) - X_*)GN_{j-1}] \\ \Rightarrow \quad & (A - GX_j)^T((X_j) - X_*) + ((X_j) - X_*)(A - GX_j) \\ & = -((X_j) - X_*)G((X_j) - X_*) - N_{j-1}GN_{j-1} =: W. \end{aligned} \quad \text{(II.33)}$$

The matrix W is negative semi-definite. Assume that $A - GX_j$ has an eigenvalue $\lambda \in \overline{\mathbb{C}^+}$ with an associated eigenvector $v \neq 0$. Then it holds

$$\begin{aligned} & (A - GX_j)v = \lambda v, \quad v^*(A - GX_j)^* = \overline{\lambda} v^H. \quad \text{(II.34)} \\ & v^*\text{(II.33)}v \quad \Rightarrow \quad 2\,\mathrm{Re}(\lambda)\, v^H((X_j) - X_*)v = v^H W v. \end{aligned}$$

The left-hand side is non-negative, since $\mathrm{Re}(\lambda) \geqslant 0$ and $(X_j) - X_* \geqslant 0$. On the other hand, the right-hand side gives $v^H W v \leqslant 0$. Therefore, $v^H W v = 0$ and moreover,

$$v^H((X_j) - X_*)G(X_j - X_*)v = 0.$$

Since $G \geqslant 0$, it holds $G(X_j - X_*)v = 0$, i. e., we have

$$GX_j v = GX_* v.$$

Thus, together with (II.34) we obtain

$$\lambda v = (A - GX_j)v = (A - GX_*)v,$$

i. e., λ is an eigenvalue of $A - GX_*$ and thus a contradiction to the asymptotic stability of $A - GX_*$.

b) From (II.32) we directly have that $X_* \leqslant X_j$ for $j \geqslant 1$. On the other hand, by (II.30) and fixed $j \geqslant 1$ we have

$$(A - GX_{j-1})^T X_j + X_j(A - GX_{j-1}) = -F - X_{j-1}GX_{j-1}, \tag{II.35}$$
$$(A - GX_j)^T X_{j+1} + X_{j+1}(A - GX_j) = -F - X_j GX_j. \tag{II.36}$$

By subtracting (II.35) from (II.36) and some manipulations we obtain

$$\begin{aligned}&(A - GX_j)^T(X_{j+1} - X_j) + (X_{j+1} - X_j)(A - GX_j)\\ &= -X_jGX_j + X_{j-1}GX_{j-1} + N_{j-1}GX_j + X_jGN_{j-1} = N_{j-1}GN_{j-1}.\end{aligned}$$

Since $A - GX_j$ is asymptotically stable and $G \geqslant 0$, it holds $X_{j+1} - X_j \leqslant 0$ by Lemma II.7 a). Therefore, it holds $X_{j+1} \leqslant X_j$ for all $j \geqslant 1$.

c) From b) we know that $\{X_j\}_{j=1}^{\infty}$ is a monotonically decreasing and bounded sequence. Therefore, the limit $\hat{X} := \lim_{j\to\infty} X_j$ exists. Since $A - GX_j$ is asymptotically stable for all $j \geqslant 1$ and the eigenvalues of a matrix are continuous with respect to the matrix entries, it holds $\Lambda(A - G\hat{X}) \subset \overline{\mathbb{C}^-}$. By taking the limit in (II.30), we see that $\hat{X}$ solves the ARE (II.28). Thus $\begin{bmatrix} I_n \\ \hat{X} \end{bmatrix}$ spans an invariant subspace corresponding to the eigenvalues in $\overline{\mathbb{C}^-}$ of the matrix $H = \begin{bmatrix} A & -G \\ -F & -A^T \end{bmatrix}$. Since by Theorem II.25, H does not have imaginary eigenvalues we obtain $\Lambda(A - G\hat{X}) \subset \mathbb{C}^-$. Therefore, $\hat{X}$ is a stabilizing solution of the ARE. Since by Lemma II.22, the stabilizing solution is unique, we have $X_* = \hat{X}$.

d) From (II.31) we obtain

$$(\mathcal{R}'(X_{j-1}))(X_j - X_*) = -(X_{j-1} - X_*)G(X_{j-1} - X_*).$$

Note that $(\mathcal{R}'(X_{j-1}))(\cdot)$ is invertible since $A - GX_{j-1}$ is asymptotically stable. This gives

$$\|X_* - X_j\| \leqslant \left\|(\mathcal{R}'(X_{j-1}))^{-1}\right\| \|G\| \|X_* - X_{j-1}\|^2,$$

where $\|\cdot\|$ denotes any consistent norms. Since $\{X_j\}_{j=0}^{\infty}$ converges, the limit $\lim_{j\to\infty} \mathcal{R}'(X_j) = \mathcal{R}'(X_*)$ exists. Since $A - GX_*$ is asymptotically stable, also the limit $\lim_{j\to\infty}(\mathcal{R}'(X_j))^{-1} = (\mathcal{R}'(X_*))^{-1}$ exists. Denote $\delta_j := \left\|(\mathcal{R}'(X_j))^{-1}\right\|$ and $\delta_* := \lim_{j\to\infty} \delta_j$. Since the sequence $\{\delta_j\}_{j=0}^{\infty}$ converges, it has a supremum, denoted by $\hat{\delta}$. Therefore, we get

$$\|X_* - X_j\| \leqslant \gamma \|X_* - X_{j-1}\|^2, \quad j = 1, 2, \ldots$$

with $\gamma := \hat{\delta}\,\|G\|$ and we have quadratic convergence. The statement for arbitrary matrix norms follows by equivalence of matrix norms.

□

Remark II.34: a) Computing X by the Schur vector method costs as much as $6-7$ iterations of Algorithm 6. Often, Algorithm 6 requires more iterations. However, it is often very useful in refining solutions obtained by other methods.

b) If A is not asymptotically stable (otherwise $X_0 = 0$ is stabilizing), then the computation of a stabilizing X_0 usually costs as much as another iteration step since this requires the solution of one additional Lyapunov equation (Homework 3, Problem 2).

c) The convergence theory also holds for $X_j := X_{j-1} + tN_{j-1}$ where $t \in [0,2]$. There exist *line search strategies* to optimize the step length after computation of the direction N_{j-1} in Algorithm 5. That is, we use a step length

$$t = \operatorname{argmin}_{\tau\in[0,2]} \|\mathcal{R}(X_{j-1} + \tau N_{j-1})\|_{\mathrm{F}}.$$

The computation of t is usually much cheaper than the actual Newton step which can drastically accelerate the iteration.

II.3.5 Solving large-scale AREs

We now consider large-scale AREs

$$\mathcal{R}(X) = C^TC + A^TX + XA - XBB^TX = 0, \quad X = X^T,$$

where $A \in \mathbb{R}^{n\times n}$ is large and sparse, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, and $m, p \ll n$. We again assume that the assumptions of Theorem II.26 hold, i.e., (A,B) stabilizable and (A,C) detectable. The constant and the quadratic term are of low rank, a setting often arising in optimal control problems. Similar to the linear (Lyapunov) case, this motivates to numerically compute an approximate solution of low-rank $X_* \approx ZDZ^T$, $Z \in \mathbb{R}^{n\times r}$, $D = D^T \in \mathbb{R}^{r\times r}$ with $r \ll n$.

The Low-Rank Newton-Kleinman Method

Inserting the low-rank matrices $G = BB^T$, $F = C^TC$ into the NK iteration scheme (II.30) gives

$$\begin{aligned}&\big(A - BB^TX_{j-1}\big)^TX_j + X_j\big(A - BB^TX_{j-1}\big)\\ &\quad = -\tilde{C}^T\tilde{C} - X_{j-1}BB^TX_{j-1} = -\underbrace{\big[\tilde{C}^T \quad X_{j-1}B\big]}_{\in\mathbb{R}^{n\times(p+m)}}\big[\tilde{C}^T \quad X_{j-1}B\big]^T\end{aligned} \tag{II.37}$$

Thus the right-hand side of the Lyapunov eqn. is of low rank and we can apply any of the low-rank methods from Section II.2.3 (projection methods, low-rank ADI) to obtain a low-rank approx. of X_{j-1}. This results in the low-rank Newton-Kleinman method for AREs. One problem remains for extended, rational Krylov and the LR-ADI method: even if A is sparse and B is thin, the *closed loop matrix*

$$A_j := A - BB^TX_{j-1} \tag{II.38}$$

at Newton step j is usually dense $\rightsquigarrow$ never explicitly form (II.38).

There are several ways to solve linear systems with the system matrix $A_j + p_jI_n$ efficiently in the low-rank ADI method or rational Krylov subspace methods:

a) **Application of an iterative solver:** This option only requires multiplications with A_j. Since $K_j := X_{j-1}B$ and B have only a few columns and rows, respectively, these can be carried out efficiently. On the other hand, the convergence of iterative solvers is often slow, as long as no good preconditioner is available.

b) **Application of the Sherman-Morrison-Woodbury identity:** It holds for $S_j := A + p_jI_n$

$$\big(A + p_jI_n - BK_j^T\big)^{-1} = S_j^{-1} + S_j^{-1}B\big(I_m - K_j^TS_j^{-1}B\big)^{-1}K_j^TS_j^{-1}.$$

Then a linear system solve with $S_j := A_j + p_jI_n$ only requires two sparse solves with S_j and one small dense solve with the matrix $I_m - K_j^TS_jB$.

Projection Approaches

In complete analogy to the Lyapunov case, we can use the Galerkin projection approach directly onto the large ARE: build subspace $\mathcal{U}$ w.r.t. A^T, C^T, e.g., range$(Q_k) = \mathcal{EK}(A^T, C^T)$ or range$(Q_k) = \mathcal{RK}(A^T, C^T, s)$ and solve small, projected ARE

$$H_kY_k + Y_kH_k^T - Y_kQ_k^HBB^TQ_k + Q_kC^TCQ_k = 0, \quad H_j = Q_k^T(A^TQ_k)$$

for stabilizing $Y_k \in \mathbb{R}^{k\times k}$ in each step (e.g., by (Hamiltonian) Schur vector method). Approximate stab. solution is $X_* \approx Q_kY_kQ_k^T$.

CHAPTER III

Matrix Functions

III.1 Introduction

By "Matrix Functions" we mean the following: take a scalar function f and $A \in \mathbb{C}^{n\times n}$ and specify $f(A) \in \mathbb{C}^{n\times n}$ such that a useful generalization of $f(z)$, $z \in \mathbb{C}$ is obtained.

Other meanings of $f(A)$ which are not part of this lecture are:

- element-wise operations, i.e. $f(A) = [f(a_{ij})]_{i,j=1}^{n}$,
- scalar valued functions: $f : \mathbb{C}^{n\times n} \to \mathbb{C}$, i.e. $\mathrm{trace},\ \det,\ \kappa(A),\ \ldots$,
- mappings $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{m\times m}$, with $m \neq n$, which do not come from a scalar function, e.g. $A^{\mathrm{T}},\ A^{\mathrm{H}},\ \mathrm{adj}(A),\ A(1:m,\, 1:m),\ \ldots$,
- function mappings $f : \mathbb{C} \to \mathbb{C}^{n\times n}$, e.g. transfer function $f(s) = C(sI - A)^{-1}B$, with $C \in \mathbb{C}^{n\times m}$, $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times m}$.

Let $f(t)$ be a scalar polynomial or rational function. We substitute A for t to define $f(A)$, replacing t^{-1} by A^{-1}, 1 by I_n.

Example: We consider the following two examples:

- $f(t) = \sum \alpha_i t^i \ \Rightarrow\ f(A) = \sum \alpha_i A^i$,
- $f(t) = \frac{1+t^3}{1-t} \Rightarrow f(A) = (I - A)^{-1}(I + A^3) = (I + A^3)(I - A)^{-1}$.

This easily generalizes to functions, having a convergent power series:

$$f(t) = \sum_{i=0}^{\infty} \alpha_i t^i.$$

For example, we can write a representation of the logarithm $\log(1 + t)$

$$\begin{aligned} \log(1+t) &= t - \frac{t^2}{2} + \frac{t^3}{3} - \frac{t^4}{4} + \ldots, \quad |t| < 1, \\ \Rightarrow\ \log(I+A) &= A - \frac{A^2}{2} + \frac{A^3}{3} - \frac{A^4}{4} + \ldots, \quad \rho(A) < 1. \end{aligned}$$

Next, we want to define $f(A)$ for general f. Moreover, some focus is given to multivalued functions like $\sqrt{t},\ \log(t)$, where we want to classify all possible $f(A)$.

III.2 Definitions of $f(A)$

III.2.1 Using the Jordan Canonical Form

We recall the Jordan normal form of an arbitrary matrix $A \in \mathbb{C}^{n\times n}$:

$$Z^{-1}AZ = J = \operatorname{diag}(J_1, \ldots, J_s), \tag{III.1}$$

$$J_k = \operatorname{diag}(J_k^1, \ldots, J_k^{p_k}), \tag{III.2}$$

$$J_k^i = \begin{bmatrix} \lambda_k & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{bmatrix} \in \mathbb{C}^{m_k^i \times m_k^i},$$

with Z nonsingular and $\sum_{k=1}^{s}\sum_{i=1}^{p_k} m_k^i = n$ and $\lambda_1, \ldots, \lambda_s$ are the distinct eigenvalues of A. The *index* of λ_k is the dimension of the largest Jordan block in which λ_k appears and denoted by $n_k = \max_i m_k^i$.

Definition III.1: A function f is said to be *defined on* $\{\lambda_i, n_i\}_{i=1}^s$ if the values

$$f^{(j)}(\lambda_i), \quad j = 0, \ldots, n_i - 1, \quad i = 1, \ldots, s$$

exist, where $f^{(j)}$ denotes the jth derivative. We call all this evaluations the values of f at $\{\lambda_i, n_i\}_{i=1}^s$.

Definition III.2 (Matrix functions via Jordan normal form)**:** Let f be defined on $\{\lambda_i, n_i\}_{i=1}^s$, where λ_i in the spectrum of $A \in \mathbb{C}^{n\times n}$ and n_i the index of λ_i, and $A = ZJZ^{-1}$ its Jordan normal form (III.1). Then

$$f(A) := Zf(J)Z^{-1} = Z\operatorname{diag}(f(J_k^i))Z^{-1}, \tag{III.3}$$

$$f(J_k^i) = \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \ldots & \frac{f^{(m_k^i-1)}(\lambda_k)}{(m_k^i-1)!} \\ & \ddots & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{bmatrix} \in \mathbb{C}^{m_k^i \times m_k^i}. \tag{III.4}$$

Remark III.3:

1. Since every square matrix$A$ has a Jordan form, $f(A)$ is defined as long as the function evaluations of f and its derivatives are defined at the eigenvalues. In particular is $f(A)$ always defined for a function f that is smooth on all of $\mathbb{C}$.
2. If A is diagonalizable it follows:

- $f(A) = Z\mathrm{diag}(f(\lambda_i))Z^{-1}$,
- A and $f(A)$ have the same eigenvectors.

3. In case of multivalued functions such as $\sqrt{t},\ \log(t)$, we have to pick a branch. But more on that later.

Example: We consider the Jordan normal form $J = \begin{bmatrix} \frac{1}{2} & 1 \\ 0 & \frac{1}{2} \end{bmatrix}$ and the matrix function $f(X) = X^3$. Applying the Definition III.2 leads to

$$f(J) = \begin{bmatrix} f(\frac{1}{2}) & f'(\frac{1}{2}) \\ 0 & f(\frac{1}{2}) \end{bmatrix} = \begin{bmatrix} \frac{1}{8} & \frac{3}{4} \\ 0 & \frac{1}{8} \end{bmatrix}.$$

III.2.2 Polynomial Interpolation

Recall: The minimal polynomial of $A \in \mathbb{C}^{n\times n}$ is the unique monic polynomial ψ of lowest degree such that $\psi(A) = 0$. It results from the Jordan normal form that

$$\psi(t) = \Pi_{i=1}^{s}(t - \lambda_i)^{n_i}$$

and ψ divides any polynomial p for which $p(A) = 0$ holds.

Theorem III.4: Let p and q be two polynomials of $A \in \mathbb{C}^{n\times n}$. It holds $p(A) = q(A)$ if and only if p and q take the same values on $\{\lambda_i, n_i\}_{i=1}^{s}$ where λ_i in the spectrum of $A \in \mathbb{C}^{n\times n}$ and n_i the index of λ_i. (Here we do not use the definition III.2 but the polynomial in the matrix as A^m is a defined quanitity for matrices.)

Proof. Exercise □

Definition III.5 (Matrix functions via Hermite interpolation)**:** Let f be defined on $\{\lambda_i, n_i\}_{i=1}^{s}$ where λ_i in the spectrum of $A \in \mathbb{C}^{n\times n}$ and n_i the index of λ_i and ψ be the minimal polynomial of A. Then $f(A) := p(A)$, where $\deg(p) < \deg(\psi) = \sum_{i=1}^{s} n_i$ and p satisfies the Hermite interpolation condition

$$p^{(j)}(\lambda_i) = f^{(j)}(\lambda_i),\ \ j = 0, \ldots, n_i - 1,\ \ i = 1, \ldots, s. \tag{III.5}$$

p is unique and called *Hermite interpolation polynomial*.

Example: We consider the matrix $A = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}$ and the function $f(t) = \sqrt{t}$ (principal branch $t^{1/2}$ of square root function), $\Lambda(A) = \{1, 4\}$, $s = 2$, $n_1 = n_2 = 1$. The interpolation has to satisfy $p(1) = f(1) = 1$, $p(4) = f(4) = 2$.

$$\Rightarrow\ p(t) = f(1)\tfrac{t-4}{1-4} + f(4)\tfrac{t-1}{4-1} = \tfrac{1}{3}(t+2)$$

$$\Rightarrow\ f(A) = p(A) = \tfrac{1}{3}(A + 2I) = \tfrac{1}{3}\begin{bmatrix} 4 & 2 \\ 1 & 5 \end{bmatrix}$$

and obviously $f(A)^2 = A$.

Attention: We do not have to use the same branch of square root for each eigenvalue:

$$f(1) = 1,\ f(4) = -2 \ \Rightarrow\ p(t) = 2 - t \text{ and } f(A) = \begin{bmatrix} 0 & -2 \\ -1 & -1 \end{bmatrix}.$$

The following properties result immediately from the Definition III.5:

- $f(A) = p(A)$ with polynomial depending on A,
- $f(A)A = Af(A)$,
- $f(A^{\mathrm{T}}) = f(A)^{\mathrm{T}}$.

Since $\psi(t)$ divides the characteristic polynomial $q(t) = \det(tI - A)$ it follows

- $q(A) = 0$ (Cayley-Hamilton),
- any power series collapses to polynomial in A:

$$\sum_{k=0}^{\infty} \alpha_k A^k = \sum_{k=0}^{n-1} d_k(A) A^k,$$

  d_k dependent on A.

III.2.3 Cauchy integral definition

Excursus: Useful concepts from complex analysis

A function $f : D \to \mathbb{C}$, $D \subset \mathbb{C}$ is called *analytic* in an open set $\mathcal{U}$, if it is complex differentiable for all $z_0 \in \mathcal{U}$, i.e.

$$f'(z_0) = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}$$

exists. An other name for this property is *holomorphic*. Analytic functions can be expressed as convergent power series.

The following properties hold for analytic functions:

- It holds $\oint_\gamma f(z)dz = 0,$ where γ is a closed curve in U (Cauchy integral theorem).
- Cauchy integral formula: We consider $f : U \to \mathbb{C}$ analytic, $D = \{z : |z - z_0| \leqslant r\} \subset U$, γ circle around ∂D. For all a in the interior of D it holds

$$f(a) = \frac{1}{2\pi i} \int_\gamma \frac{f(z)}{z-a} \mathrm{d}z. \tag{III.6}$$

  The equation (III.6) follows from elementary integral calculus and limit considerations.
- A function f that is holomorphic on a disc is completely determined by its values on the boundary of the disc.

This inspires the following definition of a matrix function for analytic functions f.

Definition III.6 (Matrix functions via Cauchy integral)**:** For a matrix $A \in \mathbb{C}^{n\times n}$ we define the matrix function

$$f(A) := \frac{1}{2\pi i} \int_\Gamma f(z)(zI - A)^{-1} \mathrm{d}z, \tag{III.7}$$

where f analytic on and inside the closed contour Γ that encloses $\Lambda(A)$.

III.2.4 Equivalence of definitions

Theorem III.7: Definition III.2 and III.5 are equivalent. If f is analytic then also Definition III.6 is equivalent to Definition III.2 and III.5.

Proof (part). Definition III.5 $\Leftrightarrow f(A) = p(A)$ for a Hermite interpolation polynomial that satisfies the condition III.5.

Assume A has a Jordan normal form (III.1).

$$\Rightarrow f(A) = p(A) = p(ZJZ^{-1}) = Zp(J)Z^{-1} = Z\mathrm{diag}(p(J_k))Z^{-1}$$
$$\Rightarrow \quad \text{exersice} \quad \Rightarrow \text{(III.4)}$$

□

III.2.5 Non-primary matrix functions

The three equivalent definitions lead to *"primary matrix functions"*, which is what we are mostly interested. However there is something called *non-primary matrix functions*. Consider the nonlinear matrix equation

$$X^2 = A$$

and the solutions X of this equation. If this where a scalar equation the solutions of this equation for X is given by the square root of A.

But for some matrices A , some solutions of $X^2 = A$ are not obtained as primary matrix functions (i.e. in the sense of Definition III.2, III.5 and III.6 with $f(x) = \pm\sqrt{x}$). We consider $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and we want to solve $X^2 = A$. We take $f(t) = \sqrt{t}$ as in Definition III.5 with $p(1) = \sqrt{1} = \pm 1$. It follows $p(1) = 1$ or $p(1) = -1$. Both I and $-I$ are square roots of I. Definition III.2 leads to the same results.

If we ignore the demand that the same branches of $\sqrt{t}$ are used for different Jordan blocks associated to $\lambda = 1$, we find $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ as extra square roots (the double eigenvalue 1 was sent to different square roots).

Are there more?

Yes! It holds $A = I = ZIZ^{-1}$ for all nonsingular Z. It follows that $Z \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} Z^{-1}$ and $Z \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} Z^{-1}$ give an infinite number of square roots. This includes the matrices $\begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix}$ (Housholder reflectors).

All these are examples of *non-primary matrix functions*. They occur when f is multivalued, A is derogatory (a distinct evaluation occurs in more than one Jordan block), and when equal eigenvalues in different branches are mapped to different branches of f in Definition III.2. Nonprimary matrix functions are not expressible as polynomials of A. Nor all nonprimary matrix functions come from Jordan normal forms: $A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Rightarrow X = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ is square root of f. Luckily, virtually all algorithms and applications need primary matrix functions.

III.3 Applications

Differential equations

The differential equation

$$\dot{x}(t) = Ax(t), \quad x_0 = x(0)$$

with $A \in \mathbb{R}^{n\times n}$, $x(t) \in \mathbb{R}^n$ has the solution $x(t) = \mathrm{e}^{At}x_0$.

The general differential equation

$$\dot{x}(t) = Ax(t) + f(t,x), \quad x_0 = x(0)$$

has the solution $x(t) = \mathrm{e}^{At}x_0 + \int_0^t \mathrm{e}^{A(t-\tau)}f(\tau,x)\mathrm{d}\tau$. For the case of second order time derivatives

$$\ddot{x}(t) + Ax(t) = 0, \quad x(0) = x_0, \quad \dot{x}(0) = \dot{x}_0$$

we obtain the solution $x(t) = \cos(\sqrt{A}t)x_0 + (\sqrt{A})^{-1}\sin(\sqrt{A}t)\dot{x}_0$. We can define $\sqrt{A}$ by the matrix function coming from each of the two branches. We can also use different branches on different eigenvalues or we can pick an arbitrary matrix satisfying that its square is equal to A. The solution $x(t)$ is independent of that. ⇝Exercise

We can also rewrite the differential equation as

$$\frac{d}{dt}\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & I \\ -A & 0 \end{bmatrix}\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix}.$$

Matrix equations in control theory

We consider the dynamical system

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{III.8a}$$
$$y(t) = Cx(t) \tag{III.8b}$$

with $A \in \mathbb{R}^{n\times n}$, $\Lambda(A) \in \mathbb{C}^-$. We know Lyapunov equations are important for working with (III.8). The Lyapunov equation

$$AX + XA^{\mathrm{T}} = -BB^{\mathrm{T}}$$

has the solution

$$\begin{aligned} X &= \int_0^\infty \mathrm{e}^{At}BB^{\mathrm{T}}\mathrm{e}^{A^{\mathrm{T}}t}\mathrm{d}t \\ &= \int_{-\infty}^\infty (\mathrm{i}wI - A)^{-1}BB^{\mathrm{T}}(\mathrm{i}wI - A)^{-\mathrm{T}}\mathrm{d}w. \end{aligned}$$

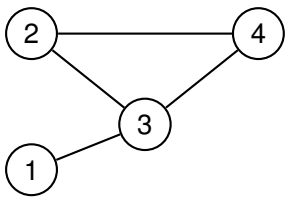


Figure III.1: Undirected network graph

Exponential integrators

Define the *phi-function* $\phi_k(z)$ recursive via

$$\phi_{k+1}(z) = \frac{\phi_k(z) - 1/k!}{z}, \quad \phi_0(z) = \mathrm{e}^z.$$

It holds that

$$\begin{aligned}
\dot{x}(t) &= Ax(t), & x(0) &= x_0 \quad \Rightarrow \quad x(t) &= \phi_0(At)x_0 \\
\dot{x}(t) &= Ax(t) + b, & x(0) &= x_0 \quad \Rightarrow \quad x(t) &= \phi_0(At)x_0 + t\phi_1(tA)b \\
\dot{x}(t) &= Ax(t) + ct, & x(0) &= x_0 \quad \Rightarrow \quad x(t) &= \phi_0(At)x_0 + t^2\phi_2(tA)c
\end{aligned}$$

what extends to general polynomial inhomogenities

$$\dot{x}(t) = Ax(t) + p(t).$$

Note: $\exp \begin{bmatrix} A & b \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathrm{e}^A & \phi_1(A)b \\ 0 & 1 \end{bmatrix}$.

Complex networks

Let $A \in \mathbb{R}^{n,n}$ be the adjacency matrix of the undirected network graph presented in Figure III.3, $A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$.

Important network measures are

- Centrality $(\mathrm{e}^A)_{ii}$ (how important is note i)
- Commutativity $(\mathrm{e}^A)_{ij}$ (how well is information from note i to j).

Other measures involve $\cosh(A), \ldots$.

Statistics

We want to sample a multivariate normal distribution $Y \sim N(\mu, C), \quad \mu \in \mathbb{R}^n$ (mean), $C = C^{\mathrm{T}} = LL^{\mathrm{T}} > 0$ (covariance matrix). Let $X \sim N(0, I)$ which is easy to simulate ,

$$Y = \mu + LX \sim N(\mu, C) \quad \text{if} \quad C = LL^{\mathrm{T}}.$$

But

$$Y = \mu + C^{1/2}X \sim N(\mu, C)$$

preferred because computing $C^{1/2}X$ is easier, especially if n is very large.

The next sections will discuss the numerical algebra for $f(A)$. We have to distinguish two problems:

1. $A \in \mathbb{C}^{n\times n}$ is small moderately sized and computing matrix-factorizations (e.g. Schur form) is possible.
2. $A \in \mathbb{C}^{n\times n}$ large, sparse (Schur form not possible), but matrix vector linear system solves with A possible and only $f(A)b$ for $b \in \mathbb{C}^n$ desired.

III.4 Methods for computing $f(A)$

We start with approaches for general f and A.

III.4.1 Taylor series

If f has a Taylor expansion

$$f(z) = \sum_{k=0}^{\infty} a_k (z-\alpha)^k$$

with convergence radius r, it can be used as basic tool for $f(A)$:

$$f(A) = \sum_{k=0}^{\infty} a_k (A-\alpha I)^k, \quad \text{if } |\lambda_i - \alpha| < r \ \ \forall \lambda_i \in \Lambda(A). \tag{III.9}$$

In practice, we truncate the sum after m terms. Important Taylor series are:

$$\begin{aligned}
\mathrm{e}^A &= I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \dots, \\
\cos(A) &= I - \frac{A^2}{2!} + \frac{A^4}{4!} - \frac{A^6}{6!} + \dots, \\
\sin(A) &= A - \frac{A^3}{3!} + \frac{A^5}{5!} - \dots, \\
\log(I+A) &= A - \frac{A^2}{2!} + \frac{A^3}{3!} - \dots, \quad \rho(A) > 1.
\end{aligned}$$

III.4.2 Rational and Padé approximations

The Taylor series (III.9) provides a polynomial approximation of f, $f \approx p$ with $\deg(p) = m$.

A generalization is given by

$$f(z) \approx r_{km}(z) = \frac{p_{km}(z)}{q_{km}(z)} \tag{III.10}$$

with numerator and denominator polynomials p_{km}, q_{km} of degree k and m, respectively. We hope to achieve a better approximation of f with lower degrees k, m compared to (III.9). We call (III.10) a $[k/m]$ *Padé approximant* of f if $q_{km}(0) = 1$ and $f(z) - r_{km}(z) = \mathcal{O}(z^{k+m+1})$. For many important functions ($\exp(z)$, $\log(z)$,...), Padé approximents are explicitly known (book on (rational) approximation theory). The evaluation of $r_{km}(A)$ for an example can be done via:

- $r_{km}(A) = q_{km}(A)^{-1}p_{km}(A)$ and $p_{km}(A)$, $q_{km}(A)$ evaluated appropriate (Horner scheme,...),
- continued fraction form $r_{mm}(z) = b_0 + \frac{a_2 z}{b_1 + \frac{a_2 z}{b_2 + \frac{a_3 z}{b_3 + \dots}}}$,
- partial function expansion.

III.4.3 Methods based on matrix functions

Easy start: Let A be diagonalizable $A = XDX^{-1},\ D = \mathrm{diag}(\lambda_i)$ such that

$$f(A) = Xf(D)X^{-1} = X\mathrm{diag}(f(\lambda_i))X^{-1}.$$

From the point of numerical stability, this computation is not recommended because of the error application by $\kappa(X) = \|X\|\|X^{-1}\| \geqslant 1$.

Better: Use a unitary factorization, e.g. our beloved Schur form (Theorem I.1) $A = QRQ^{\mathrm{H}},\ Q^{\mathrm{H}}Q = I,\ R =$  , such that

$$f(A) = Qf(R)Q^{\mathrm{H}}.$$

If $\lambda_i \neq \lambda_j$ for all $i, j \in \Lambda(A)$, we can compute first the diagonal of $F = f(R)$ and then the strict upper triangular part. This is problematic if $\lambda_i = \lambda_j$ or $\lambda_i \approx \lambda_j$.

Better: reorder and partition the Schurform

$$R = \begin{bmatrix} R_{11} & \dots & R_{1q} \\ & \ddots & \vdots \\ & & R_{qq} \end{bmatrix},\quad R_{ii} \in \mathbb{C}^{n_i \times n_i},\quad \sum_{i=1}^{q} n_i = n \tag{III.11}$$

with $\Lambda(R_{ii}) \cap \Lambda(R_{jj}) = \varnothing$ for all $i \neq j \in \{1, \dots, q\}$ and $|\lambda_i - \lambda_k| < \delta,\ \lambda_k \in \Lambda(R_{ii})$. The spectra $\Lambda(R_{ii})$ are well separated from $\Lambda(R_{jj}),\ j \in \{1, \dots, q\}\backslash\{i\}$.

Partition $F := f(R)$ in the same way

$$F = \begin{bmatrix} F_{11} & \dots & F_{1q} \\ & \ddots & \vdots \\ & & F_{qq} \end{bmatrix}.$$

At first we compute the "atomic block" $F_{ii} = f(R_{ii}),\ i = 1, \dots, q$. The eigenvalues of R_{ii} are supposed to be close to each other. Let $T \in \mathbb{C}^{m,m}$ be one of the diagonal blocks of R in (III.11). We write $T = \sigma I + M,\ \sigma = \frac{1}{m}\sum_{k=1}^{n} \lambda_k = \mathrm{trace}(T)/m$ (= mean of the eigenvalues). It holds

$$f(\sigma + z) = \sum_{k=0}^{\infty} \frac{f^{(k)}(\sigma)}{k!} z^k \tag{III.12}$$

and therefore with Section III.4.1

$$f(T) = \sum_{k=0}^{\infty} \frac{f^{(k)}(\sigma)}{k!} M^k.$$

If T has only one eigenvalue σ, M is a strictly upper triangular matrix and hence, $M^n = 0$, where n is the dimension of the matrix. Otherwise we truncate (III.12), e.g. when the difference of suczessive terms is small.

Sketch of algorithm:

1: Set $\sigma = \operatorname{trace}(T)/m,\ M = T - \sigma I,\ \epsilon =$ machine precision.
2: Set $F_0 = f(\sigma)I,\ P = M$.
3: **for** $s = 1, 2 \ldots,$ **do**
4: Set $F_s = F_{s-1} + f^{(s)}(\sigma)P$ (Taylor series).
5: Set $P = PM/(s+1)$ (power of M).
6: **if** $\|F_s - F_{s-1}\|_{\mathrm{F}} \leqslant \epsilon \|F_s\|_{\mathrm{F}}$ **then**
7: STOP
8: **end if**
9: **end for**

Assume we now have the "atomic blocks" $F_{ii} = f(R_{ii}),\ i = 1, \ldots, q$, but still require the strictly upper triangular block of F. We know $FR = RF$ because $F = f(R)$. We compute the "atomic blocks" $F_{ij} = f(R_{ij})$ by Taylor, Padé explicit formula

$$\begin{bmatrix} F_{11} & \ldots & F_{1q} \\ & \ddots & \vdots \\ & & F_{qq} \end{bmatrix} \begin{bmatrix} R_{11} & \ldots & R_{1q} \\ & \ddots & \vdots \\ & & R_{qq} \end{bmatrix} = \begin{bmatrix} R_{11} & \ldots & R_{1q} \\ & \ddots & \vdots \\ & & R_{qq} \end{bmatrix} \begin{bmatrix} F_{11} & \ldots & F_{1q} \\ & \ddots & \vdots \\ & & F_{qq} \end{bmatrix}.$$

We want to find F_{12}:

$$\begin{aligned} F_{11}R_{12} + F_{12}R_{22} &= R_{11}F_{12} - R_{12}F_{22} \\ \Leftrightarrow R_{11}F_{12} - F_{12}R_{22} &= F_{11}R_{12} - R_{22}F_{22} \end{aligned}$$

which is a Sylvester equation for F_{12}. In general

$$R_{ii}F_{ij} - F_{ij}R_{jj} = F_{ii}R_{ij} - R_{ij}F_{jj} + \sum_{k=i+1}^{j-1} (F_{ik}R_{kj} - R_{ik}F_{kj}) \qquad \text{(III.13)}$$

(Sylvester equation for F_{ij}) is uniquely solvable because (III.11) and Section II.2, e.g. by Sylvester variant of Bartels-Steward method II.2.2.

After computing F, we undo the unitary transformation Q: $f(A) = QFQ^{\mathrm{H}}$. This method is called *Schur-Parlett-Algorithm* (Handout).

We now proceed towards methods for special f, especially $f = \mathrm{e}^A$

III.4.4 Scaling and squaring for e^A

The exponential of $A \in \mathbb{C}^{n,n}$ is omnipresent in applied mathematics. There is a zoo of approaches for computing $f(A) = \mathrm{e}^A$.

We will only discuss one of the best methods for e^A, the *scaling and squaring* method which is considered as state of art. We will use the property:

$$\mathrm{e}^A = \left(\mathrm{e}^{\frac{A}{v}}\right)^v, \quad v \in \mathbb{N}.$$

The idea is, to choose $v = 2^s$ such that $\mathrm{e}^{A/v}$ can be reliably computed. By enforcing $\|A\|/v \leqslant 1$. The expression $\mathrm{e}^{A/v}$ can be computed by Taylor or Padé approximations. Here we use the diagonal Padé approximation ([m/m] Padé)

$$r_m(z) = \frac{p_m(z)}{q_m(z)}$$

for e^z, where $r_m(z) - f(z) = \mathcal{O}(z^{2m+1})$ and $q_m(0) = 1$. That is scaling and squaring which can be roughly summarized as

$$\mathrm{e}^A \approx \left(r_m\left(\frac{A}{v}\right)\right)^v,$$

and with $r_m(z) \approx \mathrm{e}^z$ for $|z| < 1$ or $|z| \approx 1$.

For e^A such [m/m] Padé approximations are known explicitly

$$p_m(z) = \sum_{j=0}^{m} \frac{(2m-j)!m!}{(2m)!(m-j)!j!} z^j, \quad q_m(z) = \sum_{j=0}^{m} \frac{(2m-j)!m!(-z)^j}{(2m)!(m-j)!j!}. \quad \text{(III.14)}$$

Scaling and Squaring in 3 Steps:

1. Find s such that $\|A/2^s\| \approx 1$ and replace $A \leftarrow A/2^s$.
2. Evaluate [m/m] Padé approximation: $r_m(A) = p_m(A)q_m(A)^{-1}$.
3. Set $X = (r_m(A))^{2^s} \approx \mathrm{e}^A$.

How to choose the scaling order $s \in \mathbb{N}$ and Padé order $m \in \mathbb{N}$?

In early developments m was fixed and s chosen so that $\|A/2^s\| \leqslant 1$ or $\leqslant 0.5$. Current versions of scaling and squaring select s, m adaptively on the basis of $\|A\|$. The core idea is that if $\mathrm{e}^{-A}\mathrm{e}^A = I$ we have $\mathrm{e}^{-A}r_m(A) = I + G$. If we assume $\|G\| \leqslant 1$ we obtain

$$\|H\| := \|\log(I+G)\| \leqslant \sum_{j=0}^{\infty} \frac{\|G\|^j}{j} = -\log(1-\|G\|)$$

and consequently $r_m(A) = \mathrm{e}^A \mathrm{e}^H = \mathrm{e}^{A+H}$ (A and H commute). If we replace A by $A/2^S$ we get

$$r_m\left(\frac{A}{2^s}\right)^{2^s} = \mathrm{e}^{A+E}, \quad E = 2^s H \text{ with } \|E\| \leqslant -2^s \log(1 - \|G\|).$$

The matrices H and G so depend on s here and the above proves the following theorem.

Theorem III.8: Let $\mathrm{e}^{-2^{-s}A} r_m(2^{-s}A) = I + G$ with $\|G\| < 1$. It holds

$$r_m(2^{-s}A)^{2^s} = \mathrm{e}^{A+E} \quad \text{and} \quad \frac{\|E\|}{\|A\|} \leqslant \frac{-\log(1 - \|G\|)}{\|2^{-s}A\|}.$$

We need to bound $\|G\|$. Let $w_m(z) = \mathrm{e}^{-z} r_m(z) - 1 = \sum_{i=2m+1}^{\infty} c_i x^i$ (from Padé approximation). We define $\Theta := \|2^{-s}A\|$ and obtain

$$\|G\| = \|w_m(2^{-s}A)\| \leqslant \sum_{i=2m+1}^{\infty} |c_i| \Theta^i =: f(\Theta).$$

With Theorem III.8 we obtain $\frac{\|E\|}{\|A\|} \leqslant -\frac{\log(1-f(\Theta))}{\Theta}$. Lets define $\beta := -log(1 - f(\Theta))$, the c_i is known explicitly. We use symbolic math to find the largest Θ_m for each m such that $\beta \leqslant \epsilon = 2^{-53}$.

Often $m = 13$ and $\Theta = 5.4$ are chosen for double precision. With $\Theta_m = \|2^{-s}A\| = 2^{-s}\|A\|$ we find $s = \lceil \log_2 \|A\| / \Theta_m \rceil$ and set $A \leftarrow \frac{A}{2^s}$.

Now s, m are chosen , we need to evaluate $r_m(A) = p_m(A) q_m(A)^{-1}$, $q_m(A) = p_m(-A)$. We observe for even degrees $m = 2d$:

$$\begin{aligned} p_m(A) = p_{2d}(A) &= \sum_{k=0}^{2d} \beta_k z^k = \beta_0 I + \beta_1 A + \cdots + \beta_{2d} A^{2d} \\ &= \underbrace{\beta_{2d} A^{2d} + \cdots + \beta_2 A^2 + \beta_0 I}_{\text{even powers}} + \underbrace{\beta_1 A + \cdots + \beta_{2d-1} A^{2d-1}}_{\text{odd powers}} \\ &= \underbrace{\beta_{2d} A^{2d} + \cdots + \beta_2 A^2}_{=:U} + \underbrace{\beta_0 I + A(\beta_1 I + \cdots + \beta_{2d-1} A^{2d-2})}_{=:V} \\ &= U + V. \end{aligned}$$

what requires $d+1$ matrix multiplications for A^2, $A^4, \ldots, A^{2d}$. We have exactly the same costs for odd degrees $2d+1$, so we always use odd degrees.

Likewise it holds $q(A) = U - V$ because $q(A) = p(-A)$ and therefore

$$r_m(A) = (U - V)^{-1}(U + V) =: X$$

Last step is then X^{2^s}.

III.4.5 Inverse Scaling and Squaring for $\log(A)$

We consider a matrix $A \in \mathbb{C}^{n,n}$ with $\Lambda(A) \cap \mathbb{R}_- = \varnothing$. We know $\log(z) + \log(y) = \log(zy)$ for $z, y \in \mathbb{R}_+$ and $\log(z) = k\log(z^{1/k})$. Hence, it holds $\log(z) = 2^s \log(z^{1/2^s})$ and for $|z|$ small $\log(1+z)$ can be approximated well by a Padé approximation. We use the property

$$\log(A) = 2^s \log(A^{\frac{1}{2^s}}) = 2^s \log(I + A^{\frac{1}{2^s}} - I).$$

Again, we want to use a Padé approximation for $\log(1+z)$, so $A^{1/2^s}$ is required. Since we need successively square roots of A, transforming it first to Schur form is beneficial

$$Q^{\mathrm{H}} A Q = R = \quad \ddots .$$

The Schur-Parlett algorithm can be specially modified to compute $R^{1/2}$. Assuming we have computed the successive square roots $Z = R^{1/2^s}$. We use Padé of $\log(1+z) \approx r_m(z)$ given in partial form

$$r_m(z) = \sum_{j=1}^{m} \frac{\alpha_j z}{1 + \beta_j z}, \tag{III.15}$$

where α_j and β_j are given explicitly. We have to evaluate $r_m(R^{1/2^s} - I)$ in an appropriate way directly via (III.15)

$$X = r_m(Z - I) = r_m(Y) = (I + \beta_1 Y)^{-1} \alpha_1 Y + \dots .$$

The degrees s and m can be chosen adaptively since for $Y \in \mathbb{C}^{n,n}$ with $\|Y\| < 1$ it holds

$$\|r_m(Y) - \log(I + Y)\| \leqslant |r_m(-\|Y\|) - \log(1 - \|Y\|)|. \tag{III.16}$$

We wand to find the smallest m such that the right-hand side is below a desired tolerance $\epsilon > 0$. The number s of square roots can be chosen until $\|Z - I\| < 1$. Only take extra square roots (costly) if this decreases the Padé degree m by more than 1 in (III.16).

Afterwards, we scale and back transform

$$\log(A) \approx F = 2^s Q X Q^{\mathrm{H}}.$$

See handout.

III.4.6 Methods for matrix square roots

We consider $A \in \mathbb{C}^{n,n}$ with $\Lambda(A) \cap \mathbb{R}_- = \varnothing$. We want $X = \sqrt{A}$ such that $X^2 = A$. We already mentioned the Schur-Parlett variant.

Another popular approach is based on the Newtons method for $X^2 - A$:

With $\mathcal{F}(X) = X^2 - A$ we have to solve the following matrix equation in every step to get the increment

$$\mathcal{F}'(X)E = \mathcal{F}(X)$$

where we use the Fréchet derivative. This leads to Newtons-method for $X^2 = A$:

1. Choose X_0.
2. Solve $E_k X_k + X_k E_k = A - X_k^2$ for E_k.
3. Set $X_{k+1} = X_k + E_k$.

Solving the Sylvester-equation in Step 2 is expensive compared to the Schur-Parlett method.

Theorem III.9: If X commutes with A then Newtons method for $X^2 = A$ can be reformulated to

$$X_{k+1} = \frac{1}{2}\left(X_k + AX_k^{-1}\right). \tag{III.17}$$

Moreover, (III.17) is related to the sign-iteration

$$S_{k+1} = \frac{1}{2}\left(S_k + S_k^{-1}\right), \quad S_0 = A^{-\frac{1}{2}}X_0$$

by $X_k = A^{1/2}S_k$.

III.5 Methods for $f(A)b$

Now let $A \in \mathbb{C}^{n,n}$ be large and sparse, $b \in \mathbb{C}^n$ and f defined on $\Lambda(A)$. Because of the large size of A, methods working with $\mathcal{O}(n^3)$ operations are infeasible. But in practice often only the action of $f(A)$ to b is desired:

$$y = f(A)b \in \mathbb{C}^n.$$

In this section we consider a method that does not work directly with $f(A)$. We recall Definition III.5: $f(A) = p(A)$ where p is the Hermite interpolation

polynomial determined by values of f on $\Lambda(A)$ such that the degree of p is the degree of the minimal polynomial of A. The minimal polynomial of A is the unique monic polynomial ψ_A of lowest degree such that $\psi_A(A) = 0$:

$$\psi_A(t) = \prod_{i=1}^{s} (t - \lambda_i)^{l_i}, \quad 0 \leqslant l_i \leqslant n_i$$

where s is the number of distinct eigenvalues and n_i the size of the largest Jordan block.

Theorem III.10: Let f be defined on $\{\lambda_i, l_i\}$ and ψ_A the minimal polynomial of A. It holds $f(A)b = q(A)b$ for the unique Hermite interpolation polynomial q with degree

$$\deg(q) < \sum_{i=1}^{s} l_i = \deg \psi_{A,b}$$

that satisfies the interpolation condition

$$q^{(j)}(\lambda_i) = f^{(j)}(\lambda_i), \quad j = 0, \ldots, l_i - 1, \ i = 1, \ldots, s.$$

Recall the Krylov subspace

$$\mathcal{K}_k(A, b) = \operatorname{span}\{b, bA, \ldots, A^{k-1}b\} = \operatorname{span}\{q(A)b, \ ; \deg(q) \leqslant k - 1\}.$$

With Theorem III.10 we get $f(A)b \in \mathcal{K}_d(A, b)$, $d = \deg(\psi_{A,b})$.

Krylov subspace method

We generate $Q_k \in \mathbb{C}^{n,k}$ with orthonormal columns such that $\operatorname{range}(Q_k) = \mathcal{K}_k(A, b)$ via an Arnoldi process (recall Section II.2.3)

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^{\mathrm{T}}, \quad Q_k^{\mathrm{H}} A Q_k = \begin{bmatrix} \ddots \\ \ddots \end{bmatrix}.$$

We always set $q_1 = b/\|b\|$. We look for $f(A)b \approx y \in \mathcal{K}_k(A, b)$. The Arnoldi approximation is given by

$$f_k = Q_k f(H_k) Q^{\mathrm{H}} b. \tag{III.18}$$

f can be effectively computed on $H_k \in \mathbb{C}^{k,k}$ by any of the methods discussed before. f_k is exact when (clearly) $k = n$ or $k = \deg(\psi_{A,b})$.

Theorem III.11: Let Q_k, H_k be given from the Arnoldi process w.r.t A and b. Then

$$f_k = Q_k f((H_k)Q_k^{\mathrm{H}}b = \widetilde{p}_{k-1}(A)b$$

where $\widetilde{p}_{k-1}$ is the unique polynomial of degree $\leqslant k-1$ that interpolates f on $\Lambda(H_k)$.

This procedure works well if f is in some sense well approximable by polynomials. In practice, this converges often slow (many Arnoldi steps) for several $f(A)$. We essentially get a polynomial approximation.

When to stop the Arnoldi process?

- Relative change $\frac{\|f_k - f_{k-1}\|}{\|f_k\|} < \epsilon$
- Special for $\mathrm{e}^A b$: $\|\mathrm{e}b - f_k\|_2 \approx \|b\|_2 h_{k+1,k} |e_k^{\mathrm{T}} \mathrm{e}^{H_k} e_1| < \epsilon$

Extension: rational Krylov subspaces (of Section II.2.3)

We consider

$$\begin{aligned}\mathrm{range}(Q_k) &= \mathcal{RK}_k(A, b, s)\\ &= \mathrm{span}\Big\{(s_1 I - A)^{-1}b, (s_1 I - A)^{-1}(s_2 I - A)^{-1}b, \ldots, \prod_{i=1}^{k}(s_i I - A)^{-1}b\Big\}\end{aligned}$$

with $s_i \in \mathbb{C}$. Let $T_k = Q_k^{\mathrm{T}} A Q_k \neq$  . We generate Q_k by rational Arnoldi-process

$$f(A) \approx f_k^{RA} = Q_k f(T_k) Q_k^{\mathrm{H}} b.$$

Theorem III.12: It holds

$$f_k^{RA} = r_m^{RA}(A)b,$$

where the rational function r_k interpolates f at $\Lambda(T_k)$.

Often roots of $r_k(A) = \Lambda(T_k)$, poles $\{s_1, \ldots, s_k\}$ make faster convergence compared to normal Arnoldi approximation. The selection of the poles $\{s_1, \ldots, s_k\}$, however, is not trivial (cf. comments in Section II.3.5) and is typically connected to rational approximation of f. For the special case $s_{2j} = 0$, $s_{2j+1} = \infty$ we use the extended Krylov subspace

$$\mathcal{EK}_k(A, b) = \mathcal{K}(A, b) \cup \mathcal{K}_k(A^{-1}, A^{-1}b).$$

It can be shown that this works well for the "Markov" function

$$f(z) = \int_{-\infty}^{0} \frac{\mathrm{d}\gamma(x)}{z - x},$$

where γ is a measure such that the integral converges for $z \in \mathbb{C}\backslash(-\infty, 0]$, for example

$$f(z) = z^{-\frac{1}{2}} = \int_{-\infty}^{0} \frac{1}{z - x} \frac{\mathrm{d}x}{\Pi\sqrt{-x}}.$$

CHAPTER IV

Randomized Numerical Linear Algebra

Goal: fast and accurate approximation of matrix factorizations and "important" subspaces, when traditional approaches are no longer feasible or become computationally too demanding (CPU time, memory).

Motivation often comes from big data/ data mining applications:

- enormous amount of data and consequently, large matrices which may not fit into fast memory,
- dense data/ unstructured sparsity (contrast to discretization matrices with structured sparsity)
- NLA algorithms often aim at best possible accuracy $\sim$ machine precision. This is unnecessary if data are uncertain/ highly inaccurate!

Randomized algorithms do not manipulate the large data matrices. They usually require matrix-matrix or matrix-vector multiplications, similar to Krylov-subspace methods, but come with the claim of being more robust and better parallelizable.

RNLA exploits that matrices are often of low numerical rank: Let $A \in \mathbb{R}^{n\times m}$, then $A \approx BC,\ B \in \mathbb{R}^{n\times k},\ C \in \mathbb{R}^{k\times m},\ k \ll m, n$. In other words,

$$A = BC + E, \quad \|E\| \leqslant \epsilon \tag{IV.1}$$

for a small ϵ. This leads to two computational problems:

1. The fixed precision low-rank approximation problem:
   Find B, C of minimal rank, such that

   $$\|A - BC\| \leqslant \epsilon. \tag{IV.2}$$

   Hence, $k = k(\epsilon)$, i.e. the rank k depends of ϵ.

2. The fixed rank (low-rank) approximation problem:

   $$\min_{\substack{B\in\mathbb{R}^{n\times k}\\ C\in\mathbb{R}^{k\times m}}} \|A - BC\| =: \epsilon. \tag{IV.3}$$

   Hence, $\epsilon = \epsilon(k)$, i.e. the precision ϵ is dependent on the rank k.

Recall: if $\|\cdot\| = \|\cdot\|_2,\ \|\cdot\|_{\mathrm{F}}$, the SVD yields solutions to both problems: for given $\mathbb{A} \in \mathbb{R}^{n\times m}$, let its SVD be given by

$$A = U\Sigma V^{\mathrm{T}} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \begin{bmatrix} V_1^{\mathrm{T}} \\ V_2^{\mathrm{T}} \end{bmatrix}, \tag{IV.4}$$

where $\sigma_1 \geqslant \cdots \geqslant \sigma_r > 0$ and $\Sigma_1 := \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$. The problem (IV.2) is (for $\|\cdot\|_2$) solved by

$$A = U_1 \Sigma_1 V_1^{\mathrm{T}} =: U_1 W_1^{\mathrm{T}} \quad (W_1 := V_1 \Sigma_1 \in \mathbb{R}^{m \times r})$$

for $\epsilon = 0$. For $\epsilon > 0$ it is solved by

$$B = \widetilde{U}_1 = \begin{bmatrix} u_1 & \ldots & u_k \end{bmatrix} \quad \text{and} \quad \widetilde{W}_1 = \begin{bmatrix} w_1 & \ldots & w_k \end{bmatrix}$$

where

$$k = \min_{j \in \{1,\ldots,r\}} \{j \,:\, \sigma_{j+1} \leqslant \epsilon\}$$

(for $\|\cdot\|_{\mathrm{F}}$ we set $k = \min_{j \in \{1,\ldots,r\}} \{j \,:\, \sqrt{\sigma_{j+1}^2 + \cdots + \sigma_r^2} \leqslant \epsilon\}$). Also, problem (IV.3) is solved by $\epsilon = \sigma_{k+1}$ for $\|\cdot\|_2$. This is true since the truncated SVD $\widetilde{U}_1 \widetilde{W}_1^{\mathrm{T}}$ yields the best rank-k approximation to A w.r.t. $\|\cdot\|_2, \|\cdot\|_{\mathrm{F}}$ with the error bound σ_k, $\sqrt{\sigma_{k+1}^2 + \cdots + \sigma_r^2}$ respectively.

Now, recall that $\mathrm{range}(U_1) = \mathrm{range}(A)$ and $\ker(A) = \mathrm{range}(V_2)$ and the columns of U_1 and V_2 are orthonormal: $U_1^{\mathrm{T}} U_1 = I_r$, $V_2^{\mathrm{T}} V_2 = I_{m-r}$. So let us look at low-rank factorizations $A \approx BC$, where B is orthonormal (note: $\mathrm{range}(B) \approx \mathrm{range}(A)$). The idea is now to compute an SVD (or QR-decomposition, Eigenvalue Decomposition(EVD)) by a 2-step procedure:

Step1: Compute an orthonormal approximate basis for $\mathrm{range}(A)$, i.e. compute $Q \in \mathbb{R}^{n \times k}$ orthonormal, such that $\mathrm{range}(A) \approx \mathrm{range}(Q)$ which implies that $A \approx QQ^{\mathrm{T}} A$. (Remark: if Q were the factor of an exact QR factorization of A, i.e. $A = QR$, then $Q^{\mathrm{T}} A = R$ and we would recover the QR factorization.)

Step 2: Given Q from Step 1, use it to compute the desired matrix factorization.

Example (SVD - Step 2)**:** Compute $B := Q^{\mathrm{T}} A$ (cheap, since Q has few columns) which yields directly a low-rank factorization $A \approx QB$. Then compute an SVD of B: $B = U_B \Sigma_B V_B^{\mathrm{T}}$ (fast, $B \in \mathbb{R}^{k \times m}$) and set $U := QU_B$, $\Sigma = \Sigma_B$, $V = V_B$.

$$\Rightarrow \quad A \approx U \Sigma V^{\mathrm{T}}$$

Example (EVD - Step 2)**:** Let $A = A^{\mathrm{T}}$ and let Q be as above. For the EVD, compute $B := Q^{\mathrm{T}} A Q$ and it follows $B = B^T$, then the eigenvalue decomposition of B is cheap as the matrix is small: $B = U \Delta U^{\mathrm{T}}$, Δ diagonal,

$$\Rightarrow A \approx QQ^{\mathrm{T}} AQQ^{\mathrm{T}} = QBQ^{\mathrm{T}} = QU\Delta U^{\mathrm{T}} Q^{\mathrm{T}} = (QU)\Delta(QU)^{\mathrm{T}},$$

which yields the dominant eigenvalues and -vectors of A. Assume now that $Y = A\Omega$ and $\mathrm{range}(Q) = \mathrm{range}(Y)$. Hence

$$BQ^{\mathrm{T}}\Omega = Q^{\mathrm{T}} \underbrace{AQQ^{\mathrm{T}}}_{\approx A} \Omega$$
$$\Rightarrow BQ^{\mathrm{T}}\Omega \approx Q^{\mathrm{T}}A\Omega = Q^{\mathrm{T}}Y.$$

So the best approximation to B can be obtained by solving the least-square problem

$$\min_{\tilde{B}} \|\tilde{B}(Q^{\mathrm{T}}\Omega) - Q^{\mathrm{T}}Y\|_{\mathrm{F}}. \tag{IV.5}$$

Call the least-square solution to (IV.5) $\hat{B}$ and replace B by $\hat{B}$ in the EVD.

Remark IV.1: In the eigenvalue decomposition example we show a way to compute the necessary matrix B without touching A again. Such an algorithm is called a *single-pass algorithm*, as the data matrix A is only used once (in Step 1). It is also possible to devise single-pass algorithms for nonsymmetric A.

Before now turning to the "randomized" aspect, observe that our fixed precision/ rank approximation problems can be formulated using orthonomality requirements as follows:

- fixed precision approximation (given $\epsilon > 0$)

$$\begin{cases} \text{find } Q \text{ with } k = k(\epsilon) \text{ orthonormal columuns such that} \\ \|A - QQ^{\mathrm{T}}A\| \leqslant \epsilon \end{cases} \tag{IV.6}$$

- fixed rank approximation (given $0 < k < \min\{m, n\}$)

$$\begin{cases} \text{find } Q \text{ with orthonormal columuns such that} \\ \|A - QQ^{\mathrm{T}}A\| \approx \min_{\mathrm{rank}(X) \leqslant k} \|A - X\| \end{cases} \tag{IV.7}$$

So, now how to address Step 1?

This is the key step in RNLA: choose a suitable random $\Omega \in \mathbb{R}^{m \times k+p}$ ($p \geqslant 0$ is an *oversampling parameter*). Then with $Y := A\Omega$, the $\mathrm{range}(Y)$ is a good approximation to the $\mathrm{range}(A)$. Q can then be obtained from Y using a standard QR factorization, or modified Gram-Schmidt.

Algorithm 7 *Step 1*

Input: $A \in \mathbb{R}^{m\times n}$, target rank $k < \min\{m,\, n\}$, oversampling parameter p.
Output: $Q \in \mathbb{R}^{m\times k+p}$ such that $Q^{\mathrm{T}}Q = I_{k+p}$ and $\mathrm{range}(Q) \approx \mathrm{range}(A)$.

1: Draw random $n \times (k+p)$ test matrix Ω.
2: Compute $Y := A\Omega$.
3: Set $Q := \mathrm{orth}(Y)$.

Why does is work?

Assume $\mathrm{rank}(A) = k$, and let w be a random vector. Then $y = Aw$ is a sample from $\mathrm{range}(A)$. Repeat this k times:

$$y^{(j)} := Aw^{(j)}, \quad j = 1, \ldots, k, \quad w^{(j)} \text{ randomly chosen.}$$

Very likely, $\{y^{(1)}, \ldots, y^{(k)}\} \subset \mathrm{range}(A)$ is linearly independent. Hence, for $Y = [y^{(1)}, \ldots, y^{(k)}]$, it holds likely that $\mathrm{range}(Y) = \mathrm{range}(A)$.

Now assume A is not low rank exactly, but approximately, $A = B + E$, where $\mathrm{rank}(B) = k$, $\|E\|$ is small. We can expect that with $p = 0$,

$$y^{(j)} := Aw^{(j)} = Bw^{(j)} + Ew^{(j)}, \quad j = 1, \ldots, k+p, \tag{IV.8}$$

leads to $Y = [y^{(1)}, \ldots, y^{(k)}]$ with $\mathrm{range}(Y) = \mathrm{range}(B) \approx \mathrm{range}(A)$ as $Ew^{(j)}$ likely moves $y^{(j)}$ slightly outside the $\mathrm{range}(B)$. Hence, an enriched space with $p > 0$ has better chance to recover $\mathrm{range}(B)$.

Note: usually $5 \leqslant p \leqslant 10$ suffices! With this, we obtain a prototype algorithm for Step 1, given in Algorithm 7. The matrix Ω is often a Gaussian, i.e. w_{ij} draw independently from normal distribution. Caution: Y may be ill-conditioned, so care must be taken when orthonormalizing its columns.

How good is the random SVD (first example and Algorithm 7)?

Theorem IV.2: Let $A \in \mathbb{R}^{m\times n}$. Run Algorithm 7 with $k \geqslant 2,\ p \geqslant 2$, and $k+p \leqslant \min\{m,\, n\}$, choose Ω as standard Gaussian test matrix ($w_{ij} \sim \mathcal{N}(0,1)$). Then

$$\mathcal{E}(\|A - QQ^{\mathrm{T}}A\|_2) \leqslant \left[1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{\min\{m,\, n\}}\right]\sigma_{k+1}, \tag{IV.9}$$

where σ_{k+1} is the $(k+1)$-th singular value of A, is the best approximation error.

How likely is this approach to fail?

Theorem IV.3: Under the assumptions of Theorem IV.2 and $p \geqslant 4$,

$$\mathcal{P}\left(\|A - QQ^{\mathrm{T}}A\|_2 \leqslant \left[1 + 9\sqrt{k+p}\sqrt{\min\{m,\, n\}}\right]\sigma_{k+1}\right) \geqslant 1 - 3p^{-p}$$

($p = 5: \ p^{-p} = 3.2 \cdot 10^{-4}, \ \ p = 10: \ p^{-p} = 10^{-10} \ \Rightarrow$ choose oversampling parameter $5 \leqslant p \leqslant 10$ justified!).

A posteriori error estimation: The exact error is $\|(I - QQ^{\mathrm{T}})A\|$. Can we get information by computing $\|(I - QQ^{\mathrm{T}})Aw^{(j)}\|, \ j = 1, \ldots, R$?

Theorem IV.4: Let Q be computed with Algorithm 7, Ω standard Gaussian test matrix, and let $w^{(j)}, \ j = 1, \ldots, l$ standard Gaussian test vectors. Then

$$\|(I - QQ^{\mathrm{T}})A\|_2 \leqslant 10\sqrt{\frac{2}{\pi}} \max_{j=1,\ldots,l} \|(I - QQ^{\mathrm{T}})Aw^{(j)}\|_2 \tag{IV.10}$$

with probability $1 - 10^{-l}$.

So for $l = 10$ we got a very reliable error estimate! If the error is too large, increase p.

Now, as a final step we derive a prototype algorithm (Algorithm 8) for the randomized SVD, where we add a few $q = 1, 2, \ldots$ steps of power iteration to Step 1 to derive Y more into the direction of the dominant left singular vectors of A(= eigenvectors of $AA^{\mathrm{T}} = U\Sigma V^{\mathrm{T}} V\Sigma^{\mathrm{T}} U^{\mathrm{T}} = U\widehat{\Sigma}V^{\mathrm{T}}$). This requires $(2q + 2)km_A + \mathcal{O}(k^2 m)$ flops, where $m_A =$ cost for applying A, A^{T} once to a vector.

How does it perform?

Theorem IV.5: Let $2 \leqslant k \leqslant \frac{1}{2}\min\{m, n\}$. Then the result of Algorithm 8 satisfies

$$\mathcal{E}(\|A - U\Sigma V^{\mathrm{T}}\|) \leqslant \underbrace{\left[1 + 4\sqrt{\frac{2\min\{m, n\}}{k - 1}}\right]^{(2q+1)^{-1}}}_{=:\delta_k} \sigma_{k+1}. \tag{IV.11}$$

Truncating to a rank k factorization yields

$$\mathcal{E}(\|A - U\Sigma V^{\mathrm{T}}\|) \leqslant \sigma_{k+1} + \delta_k. \tag{IV.12}$$

Algorithm 8 Randomized SVD

Input: $A \in \mathbb{R}^{n\times m}$, target number k of singular values, small integer q ($q = 1, 2$).

Output: Rank $2k$ factorization $A \approx U\Sigma V^{\mathrm{T}}$, U and V have orthonormal columns, $\Sigma = \mathrm{diag}(\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_{2k})$, $\widetilde{\sigma}_j \geqslant 0$.

Step 1:

1: Generate a Gaussian test matrix $\Omega \in \mathbb{R}^{m\times 2k}$.
2: Form $Y = (AA^{\mathrm{T}})^q A\Omega$.
3: Set $Q = \mathrm{orth}(Y)$.

Step 2:

4: Form $G := Q^{\mathrm{T}}A$.
5: Compute SVD $G = \widetilde{U}\Sigma V^{\mathrm{T}}$.
6: Set $U := Q\widetilde{U}$.

CHAPTER V

Outlook to Multilinear (Numerical) Linear Algebra

So far we considered 2D arrays $A \in \mathbb{R}^{n\times n}$ (matrices). In numerical methods, matrix-factorizations are a very important ingredient. For instance, recall the singular value decomposition (SVD)

$$A = U\Sigma V^{\mathrm{T}},\ U \in \mathbb{R}^{n\times n},\ \Sigma = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{n\times n},\ V \in \mathbb{R}^{n\times n},$$

with $U^{\mathrm{T}}U = I,\ V^{\mathrm{T}}V = I,\ \Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)\,,\ \sigma_1 \geqslant \cdots \geqslant \sigma_r > 0,$ $r = \mathrm{rank}(A)$. The SVD can be used to set a low-rank approximation

$$A \approx U(:,\, 1:k)\Sigma(1:k,\, 1:k)V(:,\, 1:k)^{\mathrm{T}},\ \ k \leqslant r.$$

In practice, d-order *tensors* (d-dimensional arrays)

$$A \in \mathbb{R}^{n_1\times\cdots\times n_d},\ \ d \geqslant 2$$

are more and more important. Our goal is to get an idea of SVD-type factorizations for tensors. Recall the vectorization operator for $A \in \mathbb{R}^{n\times m}$:

$$\mathrm{vec}(A) = \begin{bmatrix} A(:,\, 1) \\ \vdots \\ A(:,\, m) \end{bmatrix} \in \mathbb{R}^{mn\times 1}.$$

This operator transforms matrices into vectors (tensors of order 2 to order 1). The "matricization" or "unfolding" of a d-order tensor A generalizes this concept:

Definition V.1: Let $A \in \mathbb{R}^{n_1\times\cdots\times n_d}$. The *k-mode matrix matricization/ unfolding*

$$A_{(k)} \in \mathbb{R}^{n_k\times(n_1\ldots n_{k-1}n_{k+1}\ldots n_d)},\ \ k = 1, \ldots, d$$

has the element $a_{n_1,\ldots,n_d}$ at the position $n_k, 1 + \sum_{\substack{j=1\\ j\neq k}}^{d} \left((n_j - 1) \prod_{\substack{m=1\\ m\neq k}}^{j-1} n_m \right)$.

Applications:

- $A \in \mathbb{R}^{n\times m}$ represents an $n \times m$ image, so $A \in \mathbb{R}^{n\times m\times T}$ represents a sequence (video) of T images.
- Physical problems in several dimensions:
  - multibody problem,
  - particle physics,
  - quantum physics,

  each part has 3 spatial dimensions + time + "quantum-numbers" (spin).

Lets consider definition V.1 for order 3 tensors $A \in \mathbb{R}^{n_1 \times n_2 \times n_2}$.

Then the 1-mode matricization is

$$A_{(1)} = \boxed{A_1} \; \boxed{A_2} \; \cdots \; \boxed{A_{n_3}} \in \mathbb{R}^{n_1 \times n_2 n_3}$$

the 2-mode matricization is

$$A_{(2)} = \boxed{A_1^{\mathrm{T}}} \; \boxed{A_2^{\mathrm{T}}} \; \cdots \; \boxed{A_{n_3}^{\mathrm{T}}} \in \mathbb{R}^{n_2 \times n_1 n_3}$$

and the 3-mode matricization is

$$A_{(3)} = \begin{bmatrix} \mathrm{vec}(A_1)^{\mathrm{T}} \\ \mathrm{vec}(A_2)^{\mathrm{T}} \\ \vdots \\ \mathrm{vec}(A_{n_3})^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{n_3 \times n_1 n_2}.$$

Example: Let $\begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}$ and $\begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$ be the frontal slices of $A \in \mathbb{R}^{3 \times 4 \times 2}$. Then the matricizations are

$$A_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix},$$

$$A_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix} \text{ and}$$

$$A_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \end{bmatrix}.$$

Definition V.2: The *multilinear rank* of $A \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is the tuple

$$(r_1, \ldots, r_d) := [\mathrm{rank}(A_{(1)}), \ldots, \mathrm{rank}(A_{(d)})].$$

The example has the multilinear rank $(2, 2, 2)$.

Let $M \in \mathbb{R}^{m \times n_k}$, $k = 1, \ldots, d$ and let $B_{(k)} = MA_{(k)} \in \mathbb{R}^{m \times (n_1 \ldots n_{k-1} n_{k+1} \ldots n_d)}$.

$B_{(k)}$ can be folded back into a tensor $B \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times m \times n_{k+1} \times \cdots \times n_d}$. This is called *k-mode matrix multiplication*:

$$B = M \circ_k A \Leftrightarrow B_{(k)} = MA_{(k)}, \quad M \in \mathbb{R}^{m \times n_k}.$$

Relation to vectorization:

$$\begin{aligned} \mathrm{vec}(M \circ_1 A) &= \mathrm{vec}(MA_{(1)}) \\ &= (I_{n_2 \ldots n_d} \otimes M)\mathrm{vec}(A_{(1)}) \\ &= (I_{nd} \otimes \cdots \otimes I_{n_2} \otimes M)\mathrm{vec}(A_{(1)}). \end{aligned}$$

In general it holds

$$\begin{aligned} \mathrm{vec}(M \circ_k A) &= (I_{n_1 \ldots n_{k-1} n_{k+1} \ldots n_d} \otimes M)\mathrm{vec}(A_{(k)}) \\ &= (I \otimes M) \prod_{\sigma_k} \mathrm{vec}(A), \end{aligned}$$

where σ_k are the correct permutations.

Towards a higher order SVD

We recall the SVD $A = U\Sigma V^{\mathrm{T}}$, $U \in \mathbb{R}^{n_1 \times r}$, $V \in \mathbb{R}^{n_2 \times r}$, $\Sigma \in \mathbb{R}^{r \times r}$. We obtain

$$\mathrm{vec}(A) = (V \otimes U)\mathrm{vec}(\Sigma).$$

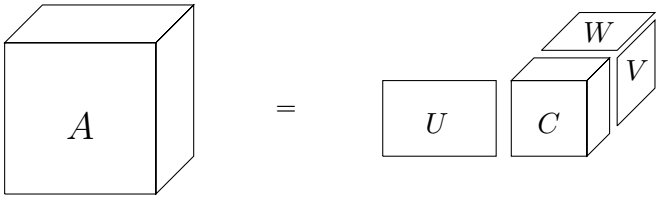
The *Tucker decomposition* of a three dimensional tensor is defined as

$$A = U \circ_1 V \circ_2 W \circ_3 C =: (U, V, W) \circ C$$

for $U \in \mathbb{R}^{n_1 \times r_1}$, $V \in \mathbb{R}^{n_2 \times r_2}$, $W \in \mathbb{R}^{n_3 \times r_3}$ orthogonal and the "core-tensor" $C \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, and leads to

$$\mathrm{vec}(A) = (W \otimes V \otimes U)\mathrm{vec}(C).$$

f



It follows

$$\begin{aligned} A_{(1)} &= UC_{(1)}(W \otimes V)^{\mathrm{T}} \\ A_{(2)} &= VC_{(2)}(W \otimes U)^{\mathrm{T}} \\ A_{(3)} &= WC_{(3)}(V \otimes U)^{\mathrm{T}}. \end{aligned}$$

We derive the **higher order SVD (HOSVD)** for the truncation to prescribed multirank (r_1, r_2, r_3):

1. Compute the SVDs of the k-matricizations

$$A_{(k)} = \widehat{U}_k \widehat{\Sigma}_k \widehat{V}_k^{\mathrm{T}}, \quad k = 1, 2, 3.$$

2. Truncate $U_k := \widehat{U}_k$, $V_k := \widehat{V}_k$, $\Sigma := \widehat{\Sigma}_k$.

3. Form the core tensor $\mathrm{vec}(C) = (U_3 \otimes U_2 \otimes U_1)\mathrm{vec}(A)$.

We obtain a truncated version of $A := (U_3 \otimes U_2 \otimes U_1)\mathrm{vec}(C)$. Note that $C \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ has no special (e.g.diagonal) structure. The generalization to order-d tensors is straight forward. This is conceptually simple, but the memory is intensive for C ($\sim r^d$, $r = \max_{i=1,\ldots,d} r_i$). There exist different tensor formats (hierachical Tucker, canonocal polyadic decomposition, tensor trains, ...) to reduce the memory demands.