Metodi di Approssimazione

Course notes

federico.poloni@unipi.it

May 24, 2023

This document contains work-in-progress notes on the topics treated in the course. They are converted semi-automatically from the Beamer slides that I used in the past years, so the wording is still "slides-like" in many parts, but I am trying to expand e.g. on the proofs.

They are meant first of all for me to use as notes while I am teaching, but they could be useful for students and independent learners, too.

Chapter 1

Sylvester equations and vectorization

Sylvester equation

$$AX - XB = C$$

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

This is, in disguise, a $mn \times mn$ linear system. *Vectorization* gives us an explicit way to construct it.

Vectorization: definition

$$\operatorname{vec} X = \operatorname{vec} \left[\begin{array}{c|c|c|c} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{array}\right] := \left[\begin{array}{c} x_{11} \\ x_{21} \\ \vdots \\ \hline x_{m1} \\ x_{12} \\ x_{22} \\ \vdots \\ \hline x_{m2} \\ \vdots \\ \hline \\ x_{1n} \\ x_{2n} \\ \vdots \\ x_{mn} \end{array}\right].$$

Vectorization: comments

Column-major order: leftmost index 'changes more often'. Matches Fortran, Matlab standard (C/C++ prefer row-major instead).

Converting indices in the matrix into indices in the vector:

$$\begin{aligned} (X)_{ij} &= (\operatorname{vec} X)_{i+mj} && \text{0-based,} \\ (X)_{ij} &= (\operatorname{vec} X)_{i+m(j-1)} && \text{1-based.} \end{aligned}$$

Using vectorization, we can work out the representation of a simple linear map, $X \mapsto AXB$ (for fixed matrices A, B of compatible dimensions).

If $X \in \mathbb{R}^{m\times n}$, $AXB \in \mathbb{R}^{p\times q}$, we need the $pq \times mn$ matrix that maps $\operatorname{vec} X$ to $\operatorname{vec}(AXB)$.

$$(AXB)_{hl} = \sum_j (AX)_{hj}(B)_{jl} = \sum_j \sum_i A_{hi} X_{ij} B_{jl}$$

$$= \left[\begin{array}{cccc|cccc|c} A_{h1}B_{1l} & A_{h2}B_{1l} & \dots & A_{hm}B_{1l} & A_{h1}B_{2l} & A_{h2}B_{2l} & \dots & A_{hm}B_{2l} & \dots \\ & & & & & & & & \\ & & & & & & & & \end{array} \right.$$

$$\left. \begin{array}{cccc} & & & \\ & & & \\ A_{h1}B_{nl} & A_{h2}B_{nl} & A_{hm}B_{nl} & \end{array} \right] \operatorname{vec} X$$

Kronecker product: definition

$$\operatorname{vec}(AXB) = \begin{bmatrix} b_{11}A & b_{21}A & \dots & b_{n1}A \\ b_{12}A & b_{22}A & \dots & b_{n2}A \\ \vdots & \vdots & \ddots & \vdots \\ b_{1q}A & b_{2q}A & \dots & b_{nq}A \end{bmatrix} \operatorname{vec} X$$

Each block is a multiple of A, with coefficient given by the corresponding entry of $B^\top$.

Definition

$$X \otimes Y := \begin{bmatrix} x_{11}Y & x_{12}Y & \dots & x_{1n}Y \\ x_{21}Y & x_{22}Y & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ x_{m1}Y & x_{m2}Y & \dots & x_{mn}Y \end{bmatrix}.$$

so the matrix above is $B^\top \otimes A$.

Properties of Kronecker products

$$X \otimes Y = \begin{bmatrix} x_{11}Y & x_{12}Y & \dots & x_{1n}Y \\ x_{21}Y & x_{22}Y & \dots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ x_{m1}Y & x_{m2}Y & \dots & x_{mn}Y \end{bmatrix}.$$

- $\operatorname{vec} AXB = (B^\top \otimes A) \operatorname{vec} X$. (*Warning*: not B^*, if complex).

- $(A \otimes B)(C \otimes D) = (AC \otimes BD)$, when dimensions are compatible. *Proof*: $B(DXC^\top)A^\top = (BD)X(AC)^\top$.
- $(A \otimes B)^\top = A^\top \otimes B^\top$.
- orthogonal $\otimes$ orthogonal = orthogonal.
- upper triangular $\otimes$ upper triangular = upper triangular.
- One can "factor out" several decompositions, e.g.,

$$A \otimes B = (U_1 S_1 V_1^*) \otimes (U_2 S_2 V_2^*) = (U_1 \otimes U_2)(S_1 \otimes S_2)(V_1 \otimes V_2)^*.$$

- In particular, $\|A \otimes B\| = \|A\|\|B\|$.

Solvability criterion

Theorem
The Sylvester equation is solvable for all C iff $\Lambda(A) \cap \Lambda(B) = \emptyset$.

$AX - XB = C \iff$

$$(I_n \otimes A - B^\top \otimes I_m)\,\mathrm{vec}(X) = \mathrm{vec}(C).$$

Take Schur decompositions of $A = Q_A U_A Q_A^*$, $B^\top = Q_B U_B Q_B^*$.

Then,

$$M = I_n \otimes A - B^\top \otimes I_m = (Q_B \otimes Q_A)(I_n \otimes U_A - U_B \otimes I_m)(Q_B \otimes Q_A)^*.$$

is a Schur decomposition.

What is on the diagonal of $I_n \otimes U_A - U_B \otimes I_m$?

If $\Lambda(A) = \{\lambda_1, \ldots, \lambda_m\}$, $\Lambda(B) = \{\mu_1, \ldots, \mu_n\}$, then on the diagonal we get $\Lambda(I_n \otimes A - B^\top \otimes I_m) = \{\lambda_i - \mu_j : i, j\}$.

Solution algorithms

Solving the linear system $M \,\mathrm{vec}\, X = \mathrm{vec}\, C$, with $M = I_n \otimes A - B^\top \otimes I_m$, costs $O((mn)^3)$.

A much better algorithm is the *Bartels–Stewart algorithm* (1972), which solves the problem $O(m^3 + n^3)$.

Idea: invert factor by factor the decomposition

$$(Q_B \otimes Q_A)(I_n \otimes U_A - U_B \otimes I_m)(Q_B \otimes Q_A)^*.$$

- Solving orthogonal systems $\iff$ multiplying by their transpose, $O(m^3 + n^3)$ using the $\otimes$ structure.
- Solving upper triangular system $\iff$ back-substitution; costs $O(\mathrm{nnz}) = O(m^3 + n^3)$.

Bartels–Stewart algorithm

A more operational description...

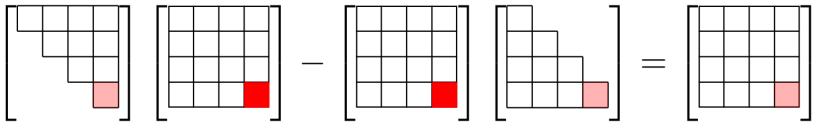
Step 1: reduce to a triangular equation.

$$Q_A U_A Q_A^* X - X\overline{Q_B} U_B^\top Q_B^\top = C$$

$$U_A Y - Y U_B^\top = D, \quad Y = Q_A^* X \overline{Q_B}, E = Q_A^* C \overline{Q_B}.$$

Let us set $L_B := U_B^\top$ (mnemonic: U / L for upper / lower triangular).

Step 2: We can compute each entry Y_{ij}, by using the (i,j)th equation, as long as we have computed all the entries *below* and *to the right* of Y_{ij}.



Step 3: $X = Q_A Y Q_B^\top$.

Matlab source

Idea We wish to store (eventually) in E_{ij} the numerator of the expression that gives Y_{ij} (with denominator $(U_A)_{ii} - (L_B)_{jj}$).

As soon as we compute an entry Y_{ij}, we update all the numerators E_{ij} that have terms including it.

```
function Y = sylv_triangular(UA, LB, E)
% solve UA*Y - Y*LB = E with UA upper tr., LB lower tr.

[m, n] = size(E);
Y = zeros(m,n);

for j = n:-1:1
    for i = m:-1:1
        Y(i,j) = E(i,j) / (UA(i,i)-LB(j,j));
        E(1:i-1,j) = E(1:i-1,j) - UA(1:i-1,i) * Y(i,j);
        E(i,1:j-1) = E(i,1:j-1) + Y(i,j) * LB(j,1:j-1);
    end
end
```

Exercise: show that one can solve by substitution in a similar fashion also equations of the form $U_A Y - Y U_B = E$, with U_A, U_B upper triangular, or $L_A Y - Y L_B = E$, with L_A, L_B lower triangular. In which order do we need to compute the entries of Y in each case?

Comments

- The idea works also with real Schur forms, i.e., block triangular forms with blocks of size 1 and 2: back-substitution gives block equations which are tiny 1×1, 1×2, 2×1 or 2×2 Sylvesters.

- A similar idea works also for $X + AXB = C$ and $AXB + CXD = E$ (with some complications); in the latter case one needs to start with the QZ decomposition of the pairs (A, C) and $(B^\top, D^\top)$.
- The idea does *not* work for three-term equations, $AXB+CXD+EXF = G$. For those, there is no (known) way to beat $O(m^3n^3)$.

Conditioning of Sylvester equations

Condition number: ratio between

$$\sigma_{\max}(I \otimes A - B^\top \otimes I) = \|I \otimes A - B^\top \otimes I\| \le \|I \otimes A\| + \|B^\top \otimes I\| \le \|A\| + \|B\|$$

and

$$\operatorname{sep}(A, B) := \sigma_{\min}(I \otimes A - B^\top \otimes I) = \min_Z \frac{\|AZ - ZB\|_F}{\|Z\|_F}.$$

(Note that $\|\operatorname{vec}(X)\| = \|X\|_F$.)

We have seen $\lambda_{\min}(I \otimes A - B^\top \otimes I) = \min_{\lambda\in\Lambda(A),\mu\in\Lambda(B)}|\lambda - \mu|$
(*minimum difference* between their eigenvalues).

If A, B are both normal, this difference is also $\operatorname{sep}(A, B) = \sigma_{\min}(I \otimes A - B^\top \otimes I)$. Otherwise, $\sigma_{\min}$ can be arbitrary smaller than $\lambda_{\min}$; there is no simple bound or expression for it.

Backward stability

The Bartels-Stewart method is backward stable (as a system of mn linear equations); i.e., the computed $\tilde{X}$ is the exact solution of a nearby linear system $(M + \Delta_M)\operatorname{vec}(\tilde{X}) = \operatorname{vec} C + \Delta_C$, with $\|\Delta_C\|/\|C\|, \|\Delta_M\|/\|M\| = O(\mathsf{u})$.

This follows from the fact that back-substitution is backward stable, combined with orthogonal transformations that do not change norms.

In particular, this implies a bound on the residual,

$$\|r\| = \|M \operatorname{vec}\tilde{X} - \operatorname{vec} C\| \le O(\mathsf{u})(\|M\|\|X\| + \|C\|) \le O(\mathsf{u})((\|A\| + \|B\|)\|X\| + \|C\|).$$

The catch, however, is that $\|X\|$ may be arbitrarily larger than $\|A\|, \|B\|, \|C\|$, when M is ill-conditioned and the separation is small.

In particular, we cannot conclude that $\widetilde{X}$ solves a nearby matrix equation

$$(A + \Delta_A)\widetilde{X} - \widetilde{X}(B + \Delta_B) = C + \Delta_C \qquad (\tilde{S})$$

To see why, note that the backward stability relation $(\tilde{S})$ is equivalent to the underdetermined linear system

$$\underbrace{\begin{bmatrix} -\widetilde{X}^\top \otimes I_n & I_m \otimes \widetilde{X} & I_{mn} \end{bmatrix}}_{=:S} \begin{bmatrix} \operatorname{vec}\Delta_A \\ \operatorname{vec}\Delta_B \\ \operatorname{vec}\Delta_C \end{bmatrix} = \underbrace{\operatorname{vec}(A\widetilde{X} - \widetilde{X}B - C)}_{=:r}.$$

Any solution of this linear system provides a perturbed matrix equation whose solution is $\tilde{X}$.

From standard results on underdetermined linear systems, the smallest-norm solution of the system is given by $S^+ r$, where S^+ is the Moore-Penrose pseudoinverse $S^+ = S^\top (SS^\top)^{-1}$. We have then

$$\left\| \begin{bmatrix} \operatorname{vec} \Delta_A \\ \operatorname{vec} \Delta_B \\ \operatorname{vec} \Delta_C \end{bmatrix} \right\| \leq \|S^+\| \|r\| = \frac{\|r\|}{\sigma_{\min}(S)}.$$

We would like this quantity to be $O(\mathsf{u})$, when $\|A\|, \|B\|, \|C\| = O(1)$. We have argued before that $r \sim \mathsf{u}\|X\|$ can be arbitrarily large than 1. We must now study the singular values of S.

For simplicity, we shall restrict to the simpler case when $m = n$ and $X = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is diagonal (which we can obtain by changing bases in the original equation).

Then, we can compute explicitly $SS^\top = \tilde{X}^\top \tilde{X} \otimes I_n + I_n \otimes \tilde{X}\tilde{X}^\top + I_{n^2}$, which is the diagonal matrix with entries $\sigma_i^2 + \sigma_j^2 + 1$ for $i, j = 1, 2, \ldots, n$. Hence, $\sigma_{\min}(S) = \sqrt{2\sigma_n^2 + 1}$, which equals 0 if $\tilde{X}$ is singular (and $\tilde{X}$ may well be large *and* singular).

This analysis comes from [Higham '93], which contains also an explicit computed 2×2 example in which the backward error is large.

Decoupling eigenvalues

If X solves the Sylvester equation $AX - XB = C$, then

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \begin{bmatrix} A & -C \\ 0 & B \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}.$$

In a sense, a large X indicates that it takes an ill-conditioned transformation to transform a block triangular decomposition into a block diagonal one.

Compare also with the scalar case: we know from the theory of the Jordan form that

$$\begin{bmatrix} \lambda & 1 \\ 0 & \mu \end{bmatrix} \text{ is similar to } \begin{bmatrix} \lambda & 0 \\ 0 & \mu \end{bmatrix},$$

if $\lambda \neq \mu$, but

$$\begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} \text{ is } \textit{not} \text{ similar to } \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}.$$

The similarity transformation used in the first case contains $\frac{1}{\lambda - \mu}$, so it breaks down when $\lambda = \mu$, and it becomes ill-conditioned when they are close.

Applications of Sylvester equations

- Compute invariant subspaces by reordering Schur forms.
- Compute matrix functions.
- Stability of linear dynamical systems, via Lyapunov equations $AX + XA^* = B$, B symmetric.

- As a step to solve certain more complicated matrix equations (Newton's method $\rightarrow$ linearization).

We see the first one now; but we will re-encounter these equations later in the course.

Invariant subspaces

Invariant subspace of a matrix M: any subspace $\mathcal{U}$ such that $M\mathcal{U} \subseteq \mathcal{U}$.

Given a basis matrix $U_1 \in \mathbb{C}^{n\times k}$ of $\mathcal{U}$ (i.e., a matrix whose columns are a basis for $\mathcal{U}$), we can find a matrix A associated to the linear operator $M|_{\mathcal{U}}$; this means finding A such that $MU_1 = U_1A$.

If $Av = \lambda v$, then $M(U_1v) = \lambda U_1 v$, i.e., $\Lambda(A) \subseteq \Lambda(M)$.

Completing a basis U_1 to one $U = [\,U_1\ U_2\,]$ of $\mathbb{C}^m$, we get

$$U^{-1}MU = \begin{bmatrix} A & C \\ 0 & B \end{bmatrix}.$$

Conversely, given matrices such that

$$M = U \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} U^{-1},$$

it is easy to verify that $\mathrm{span}(U_1)$ is an invariant subspace for M. *Example*: given a Schur form $M = QTQ^*$, and any $k = 1, 2, \ldots, n$, the span of the first k columns of Q is an invariant subspace for M.

Invariant subspaces $\iff$ block triangular decomposition $\iff$ part of the spectrum/eigenvectors of a matrix.

Examples (stable invariant subspaces)

Idea: invariant subspaces are 'the span of some eigenvectors' (usually) or Jordan chains (more generally).

Example 1 $\mathrm{span}(v_1, v_2, \ldots, v_k)$ (eigenvectors).

Example 2 Invariant subspaces of a Jordan block $\begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{bmatrix}$: $\mathrm{span}(e_1)$ and $\mathrm{span}(e_1, e_2)$.

Example 4: stable invariant subspace: vectors x_0 s.t. $\lim_{k\to\infty} A^k x_0 = 0$

Characterization of invariant subspaces

Theorem (characterization of invariant subspaces)
Let $\mathcal{U}$ be an invariant subspace of M, i.e., $MU_1 = U_1A$ for a basis matrix U_1 of $\mathcal{U}$. Then,

- $\mathcal{U}$ = span of initial parts of some Jordan chains of M.

- $\Lambda(A) \subseteq \Lambda(M)$ (with multiplicities).

Proof Take a full set of Jordan chains for A, i.e., a family of vectors v_{ij} such that $(A - \lambda_i I)v_{i1} = 0$, and $(A - \lambda_i I)v_{i,j+1} = v_{ij}$. Here $i = 1, 2, \ldots, \ell$, $j = 1, 2, \ldots, k_i$. Since this is a full set, the v_{ij}'s are the columns of an invertible matrix V (which puts A into Jordan form).

Define $w_{ij} = U_1 v_{ij}$. Then, the w_{ij} span U_1 and are Jordan chains for M:

$$(M - \lambda_i I)w_{ij} = (M - \lambda_i I)U_1 v_{ij} = U_1(A - \lambda_i I)v_{ij} = \begin{cases} 0 & j = 1, \\ U v_{i,j-1} & j > 1. \end{cases}$$

Note that the ith chain for M may be longer than k_i.

How to compute invariant subspaces?

Given a matrix M and its Schur form $M = QTQ^*$, suppose we wish to find the invariant subspace associated to a subset of its eigenvalues (example: the stable invariant subspace). We suppose, for simplicity, that this subspace is well-separated from its complement, i.e., if $\Lambda(M)$ has multiple eigenvalues, $\Lambda(A)$ contains either all of them or none.

In a (complex) Schur form $M = QTQ^*$, the T_{ii} are the eigenvalues of A. If the eigenvalues we seek are in leading position, $T_{11}, T_{22}, \ldots, T_{kk}$, then we can simply take the first k columns of Q, which span the required invariant subspace.

Reordering Schur forms

Problem
Given a Schur form $M = QTQ^*$, compute another Schur form $M = \hat{Q}\hat{T}\hat{Q}^*$ that has the eigenvalues in another (different) order.

This can be solved with the help of Sylvester equations.

It is enough to have a method to swap two blocks on the diagonal, i.e., $Q^* \begin{bmatrix} A & * \\ 0 & B \end{bmatrix} Q = \begin{bmatrix} B & * \\ 0 & A \end{bmatrix}$; then we can apply it repeatedly.

Note that if $M = \begin{bmatrix} A & * \\ 0 & * \end{bmatrix}$, then $M \operatorname{Im} \begin{bmatrix} I \\ 0 \end{bmatrix} \subseteq \operatorname{Im} \begin{bmatrix} I \\ 0 \end{bmatrix}$.

Finding a matrix Q s.t. $Q^*MQ = \begin{bmatrix} A & * \\ 0 & * \end{bmatrix}$ (with $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$) means finding a Q_1 such that $MQ_1 \subseteq Q_1$.

Reordering Schur forms

Let X solve the Sylvester equation $AX - XB = C$.

Since

$$\begin{bmatrix} 0 & I \\ I & -X \end{bmatrix} \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} \begin{bmatrix} X & I \\ I & 0 \end{bmatrix} = \begin{bmatrix} B & 0 \\ 0 & A \end{bmatrix},$$

the matrix $\left[\begin{smallmatrix} X & I \\ I & 0 \end{smallmatrix}\right]$ does the job, but it is *not unitary*. However, what matters is that Im $\begin{bmatrix} X \\ I \end{bmatrix}$ is the inv. subspace associated to $\Lambda(B)$.

We can replace $\left[\begin{smallmatrix} X & I \\ I & 0 \end{smallmatrix}\right]$ with its QR factor:

$Q = \mathrm{qr}(\left[\begin{smallmatrix} X & I \\ I & 0 \end{smallmatrix}\right])$ is such that $Q^* \begin{bmatrix} A & C \\ 0 & B \end{bmatrix} Q = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}$ with $\Lambda(T_{11}) = \Lambda(B)$, $\Lambda(T_{22}) = \Lambda(A)$.

Matlab example: computing the stable invariant subspace with `ordschur`.

Remark In a "true" Schur form, A, B are already upper triangular, so when solving the Sylvester equation we only need the back-substitution part of the Bartels–Stewart algorithm.

Sensitivity of invariant subspaces

If we perturb M to $M + \delta_M$, how much does an invariant subspace U_1 change?

The answer is once again given by the $\mathrm{sep}(\cdot, \cdot)$ function.

We can assume $U = I$ for simplicity (orthogonal change of basis): $\begin{bmatrix} I \\ 0 \end{bmatrix}$ spans an invariant subspace of $M = \begin{bmatrix} A & C \\ 0 & B \end{bmatrix}$.

Theorem [Stewart Sun book, Section V.2.2]
Let $M = \begin{bmatrix} A & C \\ 0 & B \end{bmatrix}$, $\delta_M = \begin{bmatrix} \delta_A & \delta_C \\ \delta_D & \delta_B, \end{bmatrix}$, $a = \|\delta_A\|_F$ and analogously for b, c, d.

If $(\mathrm{sep}(A, B) - a - b)^2 - 4d(\|C\|_F + c) \geq 0$, then there is a (unique) X with $\|X\|_F \leq 2\frac{d}{\mathrm{sep}(A,B)-a-b}$ such that $\begin{bmatrix} I \\ X \end{bmatrix}$ spans an invariant subspace of $M + \delta_M$.

In particular, $\frac{1}{\mathrm{sep}(A,B)}$ is the absolute condition number of an invariant subspace with respect to perturbation of the matrix.

Proof (sketch)

Note that $M + \delta M = \left[\begin{smallmatrix} A+\delta_A & C+\delta_C \\ \delta_D & B+\delta_B \end{smallmatrix}\right]$, with a small $(2, 1)$ block.

We look for a similarity transformation which zeroes out this block, with the form

$$\begin{bmatrix} I & 0 \\ -X & I \end{bmatrix} (M + \delta_M) \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} = \begin{bmatrix} * & * \\ 0 & * \end{bmatrix}.$$

Expanding out, this corresponds to the matrix equation $X(A+\delta_A)-(B+\delta_B)X = \delta_D - X(C + \delta C)X$ for X. Note that this equation is more complicated than a Sylvester equation, because it has a degree-2 term. It is called algebraic Riccati equation, and we will encounter it again in the course.

We shall prove that this algebraic Riccati equation has a sufficiently small solution X.

Consider the "Sylvester operator" $T = I \otimes A - B^\top \otimes I$, and its perturbed analogous $\hat{T} = I \otimes (A + \delta_A) - X(B + \delta_B)^\top \otimes I$. Using its inverse, we can rewrite the equation for X as a fixed-point equation.

$$\operatorname{vec} X = \underbrace{\hat{T}^{-1} \operatorname{vec}(\delta_D - X(C + \delta C)X)}_{\Phi(X)}.$$

Note that $\|\hat{T}^{-1}\|^{-1} = \sigma_{\min}(\hat{T}) \geq \sigma_{\min}(T) - a - b$, from SVD perturbation results: $|\sigma_{\min}(T + E) - \sigma_{\min}(T)| \leq \|E\|$. Recall that $\sigma_{\min}(T) = \operatorname{sep}(A, B)$.

Take norms,

$$\|X_{k+1}\|_F \leq \|\hat{T}^{-1}\|(d + \|X_k\|_F^2(\|C\|_F + c)).$$

In particular, for each $r > 0$, if $\|X\|_F \leq r$ then we have

$$\|\Phi(X)\|_F \leq \frac{1}{\beta}(\alpha + \gamma r^2)$$

for suitable positive coefficients $\alpha = d, \beta = \operatorname{sep}(A, B) - a - b = \frac{1}{\|\hat{T}^{-1}\|}, \gamma = \|C\|_F + c$.

If we can find r such that $\frac{1}{\beta}(\alpha + \gamma r^2) = r$, then we have proved that the continuous map Φ sends the ball $B(0, r)$ into itself, hence it must have a fixed point. This gives us the X that we need. To conclude the proof, we study the (scalar) quadratic equation $\alpha - \beta r + \gamma r^2 = 0$ to determine its smallest positive root $r_{\min}$.

We switch to the reverse equation $\alpha s^2 - \beta s + \gamma = 0$, whose roots are the reciprocals $s_\pm = \frac{1}{r_\pm}$. This gives

$$\frac{1}{r_\pm} = \frac{\beta \pm \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha}.$$

In particular, if $\beta^2 - 4\alpha\gamma \geq 0$ then there is a positive solution, and the smallest one satisfies $r_+ \leq \frac{2\alpha}{\beta}$. These are the two conditions that appear in the text of the theorem.

Chapter 2

Matrix functions

Polynomials of matrices

Definition 2.1. Given a scalar polynomial $p(x) = c_0 + c_1 x + \cdots + c_d x^d$, and a square matrix $A \in \mathbb{C}^{n\times n}$, we set

$$p(A) := c_0 I + c_1 A + \cdots + c_d A^d.$$

We want to give an explicit formula for $p(A)$ in terms of a Jordan decomposition $A = V \operatorname{blkdiag}(J_1, J_2, \ldots, J_s)V^{-1}$.

Jordan block J_0 If

$$J_0 = \begin{bmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{bmatrix} \in \mathbb{C}^{k\times k},$$

then

$$p(J_0) = \begin{bmatrix} c_0 & c_1 & \ldots & c_{k-1} \\ & c_0 & \ddots & \vdots \\ & & \ddots & c_1 \\ & & & c_0 \end{bmatrix}.$$

This follows from evaluating the powers of J_0. If needed, we take $c_{d+1} = c_{d+2} = \cdots = 0$.

Jordan block J_λ If

$$J_\lambda = \begin{bmatrix} \lambda & 1 & & \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{bmatrix} \in \mathbb{C}^{k\times k},$$

then

$$p(J_\lambda) = \begin{bmatrix} p(\lambda) & p'(\lambda) & \dots & \frac{p^{(k-1)}(\lambda)}{(k-1)!} \\ & p(\lambda) & \ddots & \vdots \\ & & \ddots & p'(\lambda) \\ & & & p(\lambda) \end{bmatrix}.$$

This follows from writing the polynomial in its Taylor expansion

$$p(x) = p(\lambda) + p'(\lambda)(x-\lambda) + \frac{p''(\lambda)}{2!}(x-\lambda^2) + \cdots + \frac{p^{(d)}(\lambda)}{d!}(x-\lambda)^d,$$

which reduces us to the previous case. Note that the equality holds also if we evaluate it in a matrix argument instead of x, since it follows from algebraic manipulations with powers of A (which commute with each other).

Proposition 2.2. *If $A = VJV^{-1}$ is a Jordan form, and $J = \operatorname{blkdiag}(J_1, J_2, \dots, J_s)$ with each block $J_i = J_{\lambda_i}$ of size $k_i \times k_i$, then*

$$p(A) = V \operatorname{blkdiag}(p(J_1), p(J_2), \dots, p(J_s))V^{-1}, \quad p(J_i) = \begin{bmatrix} p(\lambda_i) & p'(\lambda_i) & \dots & \frac{p^{(k_i-1)}(\lambda_i)}{(k_i-1)!} \\ & p(\lambda_i) & \ddots & \vdots \\ & & \ddots & p'(\lambda_i) \\ & & & p(\lambda_i) \end{bmatrix}. \tag{2.1}$$

Indeed, we have

$$p(A) = \sum c_i (VJV)^{-1} = V\left(\sum c_i J^i\right) V^{-1} = V \operatorname{blkdiag}(p(J_1), p(J_2), \dots, p(J_s))V^{-1},$$

and we can conclude using the previous results.

Functions of matrices [Higham book, '08, Ch. 1]

We can extend the same definition (2.1) to arbitrary scalar functions:

Definition 2.3. Given a function $f : U \subseteq \mathbb{C} \to \mathbb{C}$, and a matrix A with Jordan decomposition as above, we say that f is *defined on A* if f is defined and differentiable at least $k_i - 1$ times on each eigenvalue λ_i of A, and its value is

$$f(A) := V \operatorname{blkdiag}(f(J_1), f(J_2), \dots, f(J_s))V^{-1},$$
$$f(J_i) := \begin{bmatrix} f(\lambda_i) & f'(\lambda_i) & \dots & \frac{f^{(k_i-1)}(\lambda_i)}{(k_i-1)!} \\ & f(\lambda_i) & \ddots & \vdots \\ & & \ddots & f'(\lambda_i) \\ & & & f(\lambda_i) \end{bmatrix}.$$

Note that in the Jordan decomposition J is unique, but V is not; is this definition independent of the choice of V? To prove this, we rely on another equivalent definition.

Alternate definition: via Hermite interpolation

Definition 2.4. Given $f : U \subseteq \mathbb{C} \to \mathbb{C}$ and $A \in \mathbb{C}^{n\times n}$ as above, we define $f(A) := p(A)$, where $p(x)$ is any polynomial such that $f(\lambda_i) = p(\lambda_i), f'(\lambda_i) = p'(\lambda_i), \ldots, f^{(k_i-1)}(\lambda_i) = p^{(k_i-1)}(\lambda_i)$ for each i.

Remarks:

- We shall prove soon that there always exists a polynomial $p(x)$ that satisfies these conditions.
- Note that this definition does not depend on the choice of $p(x)$ among all the polynomials that satisfy the given conditions, since (2.1) shows that the value of $p(A)$ depends only on the prescribed values and conditions.
- This definition coincides with the previous Definition 2.3, again in view of (2.1), but it is independent of V because $p(A)$ is so.
- If A has more than one Jordan block with the same eigenvalue, some of these conditions are repeated. This is fine.
- One may think "all matrix functions are polynomials", but $p(x)$ here depends on A, so this is not too deep; it is like saying that "all scalar functions are polynomials" because given any function f and $z \in \mathbb{C}$ there is always a polynomial such that $f(z) = p(z)$ (for instance, a constant one).

Example: square root

$$A = \begin{bmatrix} 4 & 1 & & \\ & 4 & 1 & \\ & & 4 & \\ & & & 0 \end{bmatrix}, \quad f(x) = \sqrt{x}$$

We look for an interpolating polynomial with

$$p(0) = 0,\ p(4) = 2,\ p'(4) = f'(4) = \frac{1}{4},\ p''(4) = f''(4) = -\frac{1}{32}.$$

I.e.,

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 4^3 & 4^2 & 4 & 1 \\ 3\cdot 4^2 & 2\cdot 4 & 1 & 0 \\ 6\cdot 4 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ \frac{1}{4} \\ -\frac{1}{32} \end{bmatrix},$$

$$p(x) = \frac{3}{256}x^3 - \frac{5}{32}x^2 + \frac{15}{16}x.$$

$$p(A) = \frac{3}{256}A^3 - \frac{5}{32}A^2 + \frac{15}{16}A = \begin{bmatrix} 2 & \frac{1}{4} & -\frac{1}{64} & \\ & 2 & \frac{1}{4} & \\ & & 2 & \\ & & & 0 \end{bmatrix}.$$

(One can check that $f(A)^2 = A$.)

Hermite interpolation

A suitable polynomial always exists:

Theorem 2.5. *Given distinct points $x_1, x_2, \ldots, x_n$, multiplicities $m_1, m_2, \ldots, m_n$, there exists a unique polynomial of degree $d < m_1 + m_2 + \cdots + m_n$ such that (for all $i = 1, \ldots, n$)*

$$p(x_i) = y_{i,0},\ p'(x_i) = y_{i,1},\ \ldots,\ p^{(m_i-1)}(x_i) = y_{i,m_i-1},$$

for each choice of the y_{ij}.

Proof (sketch)

- Interpolation conditions $\iff$ square linear system $Vp = y$, where p is the vector of polynomial coefficients.
- We prove that V has no kernel. If $Vz = 0$ for a vector z, then the associated polynomial $z(x)$ has roots at x_i of multiplicity m_i. By degree reasons it must be the zero polynomial.

Non-Example: square root

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad f(x) = \sqrt{x}$$

does not exist (because $f'(0)$ is not defined).

(Indeed, there is no matrix such that $X^2 = A$: every 2×2 nilpotent matrix X has Jordan form equal to A, thus $X^2 = 0$.)

Example: matrix exponential

$$A = S \begin{bmatrix} -1 & & & \\ & 1 & & \\ & & 1 & 1 \\ & & & 1 \end{bmatrix} S^{-1}, \quad f(x) = \exp(x).$$

$$\exp(A) = S \begin{bmatrix} e^{-1} & & & \\ & e & & \\ & & e & e \\ & & & e \end{bmatrix} S^{-1}$$

Since there are 2 blocks with the same eigenvalue 1, there are only 3 interpolation conditions rather than 4:

$$p(-1) = e^{-1}, \quad p(1) = e, \quad p'(1) = e.$$

This is not a problem, though; we can still find a polynomial $p(x)$ (of degree at most 2 rather than 3) that satisfies them.

Note that the matrix $B = \exp(A)$ obtained in this way coincides with the limit of the matrix-valued power series $I + A + \frac{1}{2}A^2 + \frac{1}{3!}A^3 + \dots$. It is simple to see this for the diagonal terms, since the diagonal entries of this power series are just the scalar series for e^{-1} and e, but more complicated to see for $B_{3,4}$. We will prove later that matrix functions can also be computed as the limit of the Taylor series, when one has a Taylor series for $f(x)$ and $\Lambda(A)$ is included in its domain of convergence.

Example: matrix sign

$$A = V \begin{bmatrix} -3 & & & \\ & -2 & & \\ & & 1 & 1 \\ & & & 1 \end{bmatrix} V^{-1}, \quad f(x) = \operatorname{sign}(x) = \begin{cases} 1 & \operatorname{Re} x > 0, \\ -1 & \operatorname{Re} x < 0. \end{cases}$$

$$f(A) = V \begin{bmatrix} -1 & & & \\ & -1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} V^{-1}.$$

This matrix function is not constant (for general V), unlike its scalar counterpart.

Instead, we can recover stable / unstable invariant subspaces of A as $\ker(f(A) \pm I)$.

If we found a way to compute $f(A)$ without diagonalizing, we could use it to compute eigenvalues via bisection...

Example: complex square root

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad f(x) = \sqrt{x}.$$

We can choose different branches: for instance $f(i) = \frac{1}{\sqrt{2}}(1 + i)$, $f(-i) = \frac{1}{\sqrt{2}}(1 - i)$.

Polynomial: $p(x) = \frac{1}{\sqrt{2}}(1 + x)$.

$$p(A) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}.$$

This is the so-called *principal* square root, i.e., the one with eigenvalues in the right half-plane — other choices are possible.

Exercise 2.6. Compute $g(A)$, where $g(x)$ is another branch of the square root function with $g(\pm i) = \frac{1}{\sqrt{2}}(\pm 1 + i)$. Note that, unlike the previous one, $g(A)$ does not have real coefficients. (Indeed, it can't be real because its eigenvalues are not complex conjugate!)

Non-example: nonprimary square root

With our definition, if we have

$$A = S \begin{bmatrix} 1 & & \\ & 1 & \\ & & 2 \end{bmatrix} S^{-1}, \quad f(x) = \sqrt{x}$$

we cannot get

$$f(A) = S \begin{bmatrix} 1 & & \\ & -1 & \\ & & \sqrt{2} \end{bmatrix} S^{-1} :$$

in our definition, we have to set either $f(1) = 1$, or $f(1) = -1$, and we cannot use both on different Jordan blocks.

This matrix is a solution of $X^2 = A$, though.

In general, if a matrix A has multiple eigenvalues, one can find more solutions of $X^2 = A$ by choosing different signs on Jordan blocks with the same eigenvalue. As a more extreme example, for $A = I_2$, we can take $X = V \begin{bmatrix} 1 & \\ & -1 \end{bmatrix} V^{-1}$ for any invertible V....

'Pseudo-matrix-functions' defined in this way are called *nonprimary matrix functions*; they are *not* matrix functions with our definition. We will not deal with them in the course, but we just mention their existence.

Note that nonprimary matrix functions are *not* polynomials in A, unlike matrix functions.

2.1 Properties of matrix functions

- $f(MAM^{-1}) = Mf(A)M^{-1}$, since they are polynomials in A.
- $f(\left[\begin{smallmatrix} A & 0 \\ 0 & B \end{smallmatrix}\right]) = \begin{bmatrix} f(A) & 0 \\ 0 & f(B) \end{bmatrix}$, for the same reason.
- If the eigenvalues of A are $\lambda_1, \ldots, \lambda_s$, the eigenvalues of $f(A)$ are $f(\lambda_1), \ldots, f(\lambda_s)$. Their algebraic multiplcities stay the same, but geometric multiplicities may increase (when $f'(\lambda_i) = 0$).
- If $h(x) = f(x)g(x)$ for three scalar functions f, g, h, then $f(A)g(A) = h(A)$. Analogously for sums and compositions. Proof: replace $f(A), g(A)$ with polynomials $p_f(A), p_g(A)$. Then, set $p_h(x) := p_f(x)p_g(x)$; $p_f(A)p_g(A) = p_h(A)$ is true (by expanding), and $p_h(x)$ is an Hermite interpolant for $h(x)$.
- In particular, if $f(x)$ satisfies a certain scalar identity, for instance $f(x)^2 - x = 0$ for the square root, then $f(A)$ does too.
- If $f_n \to f$ together with 'enough derivatives' (pointwise; for instance because they are analytic and the convergence is uniform), then $f_n(A) \to f(A)$.

- If a sequence of matrices $A_n \to A$, then $f(A_n) \to f(A)$. Proof: we will see it soon; it is slightly more complicated because each A_n corresponds to a different polynomial p_{f,A_n}.

Matrix functions as Cauchy integrals

The following result generalizes Cauchy's integral formula from complex analysis.

Proposition 2.7. *If f is holomorphic (analytic) on and inside a contour Γ that encloses the eigenvalues of A,*

$$f(A) = \frac{1}{2\pi i} \int_\Gamma f(z)(zI - A)^{-1} dz. \tag{2.2}$$

Proof. Using a Jordan form $A = VJV^{-1} \in \mathbb{C}^{m\times m}$, we can reduce to the case of a single Jordan block. Then,

$$\frac{1}{2\pi i} \int_\Gamma f(z)(zI - J)^{-1} dz = \frac{1}{2\pi i} \int_\Gamma f(z) \begin{bmatrix} z-\lambda & -1 & & \\ & z-\lambda & -1 & \\ & & \ddots & \ddots \\ & & & z-\lambda \end{bmatrix}^{-1} dz$$

$$= \begin{bmatrix} \frac{1}{2\pi i}\int_\Gamma \frac{f(z)}{z-\lambda} dz & \frac{1}{2\pi i}\int_\Gamma \frac{f(z)}{(z-\lambda)^2} dz & \cdots & \frac{1}{2\pi i}\int_\Gamma \frac{f(z)}{(z-\lambda)^{n-1}} dz \\ & \ddots & \ddots & \ddots \\ & & \ddots & \frac{1}{2\pi i}\int_\Gamma \frac{f(z)}{(z-\lambda)^2} dz \\ & & & \frac{1}{2\pi i}\int_\Gamma \frac{f(z)}{z-\lambda} dz \end{bmatrix}$$

$$= \begin{bmatrix} f(\lambda) & f'(\lambda) & \cdots & \frac{f^{(n-1)}}{(n-1)!} \\ & \ddots & \ddots & \ddots \\ & & \ddots & f'(\lambda) \\ & & & f(\lambda) \end{bmatrix}$$

by the scalar version of Cauchy's integral formula (including the version that computes derivatives).

□

Note that (2.2) can be considered an alternative definition of matrix functions. It is surprisingly general, as it works also for infinite-dimensional operators that do not have a spectrum in the classical sense, and indeed it is popular in that setting.

Corollary $f(A)$ is continuous in A, since the integrand $f(z)(zI - A)^{-1}$ is continuous, and hence uniformly continuous on a compact set $K \subset \mathbb{C} \setminus \Lambda(A)$.

This proof works for *holomorphic* f.

One can prove that $f(A)$ is continuous in A also in more general settings, e.g., A with real eigenvalues and f with enough continuous real derivatives. The proof is more complicated.

Sketch:

- Given a sequence $A_n \to A$, take for each n an interpolating polynomial $p_n(x)$ of f in the spectrum of A_n.
- The coefficients of these interpolating polynomials $p_n(x)$ are continuous in the nodes, even in a setting where some of the eigenvalues converge to a common limit: this is not at all obvious from our proof, but it follows from other techniques from interpolation like divided differences.
- $\|f(A)-f(A_n)\| = \|p_n(A_n)-p(A)\| \leq \|p_n(A_n)-p_n(A)\|+\|p_n(A)-p(A)\|$, and both terms are bounded.

Chapter 3

Sensitivity of matrix functions

Conditioning of computing matrix functions

Recall: the absolute condition number of a differentiable $f : \mathbb{R}^m \to \mathbb{R}^n$ is the norm of its Jacobian.

$f(\tilde{x}) = f(x+h) = f(x) + \nabla_x f \cdot h + o(h)$ implies

$$\kappa_{abs}(f,x) = \lim_{\varepsilon\to 0} \sup_{\|\tilde{x}-x\|\le\varepsilon} \frac{\|f(\tilde{x})-f(x)\|}{\|\tilde{x}-x\|} = \|\nabla f\|$$

$$\kappa_{rel}(f,x) = \lim_{\varepsilon\to 0} \sup_{\frac{\|\tilde{x}-x\|}{\|x\|}\le\varepsilon} \frac{\frac{\|f(\tilde{x})-f(x)\|}{\|f(x)\|}}{\frac{\|\tilde{x}-x\|}{\|x\|}} = \kappa_{abs}(f,x)\frac{\|x\|}{\|f(x)\|}.$$

Fréchet derivative

The Fréchet derivative is an "operator version" of the Jacobian.

Definition 3.1. The *Fréchet derivative* of a matrix function f is the *linear* operator $L_{f,X} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ (when it exists) such that

$$f(X+E) = f(X) + L_{f,X}(E) + o(\|E\|).$$

I.e., in a neighbourhood of X, f behaves like a linear function.

Example

$f(x) = x^2$, $f(X) = X^2$.

$$(X+E)^2 = X^2 + XE + EX + E^2 = X^2 + \underbrace{XE+EX}_{L_{f,X}(E)} + \underbrace{E^2}_{o(\|E\|)}.$$

$L_{f,X}$ is a linear operator that maps matrices to matrices; we can consider its vectorized version:

$$\widehat{L} : \operatorname{vec} E \mapsto \operatorname{vec} L_{f,X}(E).$$

In this case,

$$\widehat{L} = X^T \otimes I + I \otimes X.$$

The matrix $\widehat{L}$ (an $n^2 \times n^2$ matrix) is the "usual" Jacobian of the map $\operatorname{vec} X \mapsto \operatorname{vec} f(X)$. So this is nothing new with respect to the usual setting in multivariate analysis.

Example

$f(x) = x^{-1}$, $f(X) = X^{-1}$.

$$\begin{aligned}(X+E)^{-1} &= \big((I + EX^{-1})X\big)^{-1} \\ &= X^{-1} \underbrace{-X^{-1}EX^{-1}}_{L_{f,X}(E)} + \underbrace{X^{-1}EX^{-1}EX^{-1} - \dots}_{o(\|E\|)}\end{aligned}$$

$$\widehat{L} = -X^{-T} \otimes X^{-1}.$$

Properties

Follow from those of Jacobians:

- $L_{f+g,X} = L_{f,X} + L_{g,X}$.
- $L_{f\circ g,X} = L_{f,g(X)} \circ L_{g,X}$.
- $L_{f^{-1},f(X)} = L_{f,X}^{-1}$.

Example Let $g(y) = \sqrt{y}$ (principal branch: we take the root in the right half-plane), Y with no real nonpositive eigenvalue.

Then $g(y)$ is the inverse of $f(x) = x^2$, and its Fréchet derivative $F = L_{g,Y}(E)$ is the matrix such that $L_{f,X}(F) = E$, i.e.,

$$XF + FX = E, \quad X = g(Y) = Y^{1/2}.$$

(solution of a Sylvester equation). X has eigenvalues in the right half-plane, so the Sylvester equation is always solvable: $\Lambda(X) \cap \Lambda(-X) = \emptyset$.

Trick to compute $L_{f,X}(E)$

The following formula generalizes $f(\left[\begin{smallmatrix}\lambda & 1\\ 0 & \lambda\end{smallmatrix}\right]) = \left[\begin{smallmatrix}f(\lambda) & f'(\lambda)\\ 0 & f(\lambda)\end{smallmatrix}\right]$, and lets us evaluate $L_{f,X}(E)$ "as fast as" $f(X)$ for $X \in \mathbb{C}^{2n\times 2n}$.

Theorem 3.2. *Let matrices $X, E \in \mathbb{C}^{n\times n}$ and a function f Fréchet differentiable in X be given. Then,*

$$f\left(\begin{bmatrix} X & E \\ 0 & X \end{bmatrix}\right) = \begin{bmatrix} f(X) & L_{f,X}(E) \\ 0 & f(X) \end{bmatrix}. \tag{3.1}$$

Proof. Set $\varepsilon > 0$. We wish to evaluate $f\left(\begin{bmatrix} X+\varepsilon E & E \\ 0 & X \end{bmatrix}\right)$ by block-diagonalizing.

Take a solution of the Sylvester equation $(X+\varepsilon E)Z - ZX = -E$. Then,

$$\begin{bmatrix} I & -Z \\ 0 & I \end{bmatrix} \begin{bmatrix} X+\varepsilon E & E \\ 0 & X \end{bmatrix} \begin{bmatrix} I & Z \\ 0 & I \end{bmatrix} = \begin{bmatrix} X+\varepsilon E & 0 \\ 0 & X \end{bmatrix}.$$

By direct verification, one sees that $Z = -\frac{1}{\varepsilon} I$ is a solution. Note that we do not need to prove that the Sylvester equation has a unique solution; even if it were not unique, taking that particular solution Z always work for block-diagonalization.

Hence

$$\begin{aligned} f\left(\begin{bmatrix} X+\varepsilon E & E \\ 0 & X \end{bmatrix}\right) &= f\left(\begin{bmatrix} I & Z \\ 0 & I \end{bmatrix} \begin{bmatrix} X+\varepsilon E & 0 \\ 0 & X \end{bmatrix} \begin{bmatrix} I & -Z \\ 0 & I \end{bmatrix}\right) \\ &= \begin{bmatrix} I & Z \\ 0 & I \end{bmatrix} \begin{bmatrix} f(X+\varepsilon E) & 0 \\ 0 & f(X) \end{bmatrix} \begin{bmatrix} I & -Z \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} f(X+\varepsilon E) & \frac{f(X+\varepsilon E)-f(X)}{\varepsilon} \\ 0 & f(X) \end{bmatrix}. \end{aligned}$$

By the definition of Fréchet derivative,

$$f(X+\varepsilon E) = f(X) + \varepsilon L_{f,X}(E) + o(\varepsilon \|E\|)$$

when $\varepsilon \to 0$. □

Exercise 3.3. Can you find an example of X, E such that the Sylvester equation $(X+\varepsilon E)Z - ZX = -E$ is singular for each $\varepsilon > 0$?

Theorem
Let $X \in \mathbb{C}^{n\times n}$, with eigenvalues λ_i and algebraic multiplicities m_i. If f is of class $\mathcal{C}^{2m_i-1}$ in (a neighborhood of) each of the λ_i, then $f(X)$ is Fréchet differentiable in X.

Note that matrices $\tilde{X}$ inside a sufficiently small neighborhood $B(X,\varepsilon)$ of X have Jordan block sizes in each λ_i of size at most m_i, and $\widehat{X} = \left[\begin{smallmatrix} \tilde{X} & E \\ 0 & \tilde{X} \end{smallmatrix}\right]$ has Jordan block sizes at most $2m_i$. So $f(\widehat{X})$ exists and is continuous in $\tilde{X} = X$. Hence, by the formula in the above proof, all directional derivatives exist and they are continuous in X. By a standard result in multivariate analysis (known as "teorema del differenziale totale" in Italian), then f is differentiable.

Fréchet derivative and condition number

Hence, $\kappa_{abs}(f,X) = \|L_{f,X}\|$.

... with some attention to what 'norm' means here.

The norm used for $\|\widetilde{X} - X\|$ is any matrix norm on $n \times n$ matrices, and $\|L_{f,X}\|$ is the 'operator norm' (on $n^2 \times n^2$ matrices) induced by it.

Easy case If we take $\|\widetilde{X} - X\|_F$, it corresponds to $\|\operatorname{vec} X\|_2$, so $\kappa_{abs}(f, X) = \|\widehat{L}_{f,X}\|_2$.

Harder cases For all other norms ($\|\widetilde{X} - X\|_2$ in particular), there is no simple expression for the 'induced operator norm'.

Even with the "easy norm", computing $\|\widehat{L}_{f,X}\|_2$ isn't immediate. Its *eigenvalues* $\Lambda(\widehat{L}_{f,X})$ are simpler, and can give us at least some partial information on when the norm is large.

Eigenvalues of Fréchet derivatives [Higham book '08, Ch. 3]

Theorem
Let $X \in \mathbb{C}^{n\times n}$ have eigenvalues $\lambda_1, \ldots, \lambda_m$ (with their algebraic multiplicity). Then, the n^2 eigenvalues of $L_{f,X}$ (with multiplicity) are

$$f[\lambda_i, \lambda_j] := \begin{cases} \frac{f(\lambda_i) - f(\lambda_j)}{\lambda_i - \lambda_j} & \lambda_i \neq \lambda_j, \\ f'(\lambda_i) & \lambda_i = \lambda_j. \end{cases}$$

for all $i, j = 1, 2, \ldots, n$.

Proof. First of all, replace $f(x)$ with a polynomial that interpolates X with sufficiently high multiplicities, so that for each E

$$f\left(\begin{bmatrix} X & E \\ 0 & X \end{bmatrix}\right) = p\left(\begin{bmatrix} X & E \\ 0 & X \end{bmatrix}\right)$$

and hence $L_{f,X}(E) = L_{p,X}(E)$.

$$\begin{aligned} p(X+E) &= c_0 + (X+E) + c_1(X+E)^2 + c_2(X+E)^3 + \ldots \\ &= c_0 + c_1(X+E) + c_2(X^2 + EX + XE + E^2) + c_3(X^3 + \ldots) \\ &= p(X) + c_1 E + c_2(EX + XE) + c_3(X^2E + XEX + X^2E) \\ &\quad + \cdots + O(\|E\|^2) \end{aligned}$$

Vectorizing,

$$\widehat{L}_{f,X} = c_1 I + c_2(I \otimes X + X^\top \otimes I) + c_3(I \otimes X^2 + X^\top \otimes X + (X^2)^\top \otimes I) + \ldots$$

i.e.,

$$\hat{L}_{f,X} = \sum_{k=1}^{d} c_k \sum_{h=1}^{k} (X^{k-h})^\top \otimes X^{h-1}$$

Now take Schur forms $X = Q_1 U_1 Q_1^*$, $X^T = Q_2 U_2 Q_2^*$.

$$\hat{L}_{f,X} = (Q_2 \otimes Q_1) \underbrace{\left(\sum_{k=0}^{d} p_k \sum_{h=1}^{k} U_2^{k-h} \otimes U_1^{h-1} \right)}_{:=U} (Q_2 \otimes Q_1)^*.$$

This is a Schur decomposition (orthogonal-triangular-orthogonal), so we can read off the eigenvalues on the diagonal: if $i \neq j$ we have

$$U_{i+n(j-1),i+n(j-1)} = \sum_{k=0}^{d} p_k (\sum_{h=1}^{k} \lambda_i^{k-h} \lambda_j^{h-1}) = \sum_{k=0}^{d} p_k \frac{\lambda_i^k - \lambda_j^k}{\lambda_i - \lambda_j}$$
$$= \frac{p(\lambda_i) - p(\lambda_j)}{\lambda_i - \lambda_j} = \frac{f(\lambda_i) - f(\lambda_j)}{\lambda_i - \lambda_j}.$$

Similarly for $i = j$ we get $f'(\lambda_i)$. □

Unfortunately, similar tricks don't work with an SVD, because X^2 and X do not have the same singular vectors and values. So there is no simple formula for the singular values of $\hat{L}$.

Condition number bound

If X is diagonalizable, we can replace the Schur form with an eigendecomposition, and obtain a bound.

Proposition 3.4. *Let $X = V\Lambda V^{-1}$ be diagonalizable. Then, for the Frobenius norm,*

$$\kappa_{abs}(f, X) = \|\hat{L}_{f,X}\| \leq \kappa_2(V)^2 \max_{i,j} |f[\lambda_i, \lambda_j]|.$$

Then, as usual, $\kappa_{rel}(f, X) = \kappa_{abs}(f, X) \frac{\|X\|}{\|f(X)\|}$.

This bound displays two possible causes of ill-conditioning:

- $|f[\lambda_i, \lambda_j]|$ is large, or
- $\kappa_2(V)$ is large, i.e., X is very non-normal.

Example

Example $f(x) = \sqrt{x}$ (principal square root): for which choices of $\Lambda(X)$ do we encounter a large $|f[\lambda_i, \lambda_j]|$?

- λ_i's close to 0, and
- Pairs of close-by eigenvalues on opposite sides of the branch cut (negative real axis).

More generally, a large $f[\lambda_i, \lambda_j]$ may come from

- Large f',
- pairs of eigenvalues close to a discontinuity in f.

Chapter 4

Methods for general matrix functions

Matrix functions arise in several areas: solving ODEs (e.g., $\exp A$), matrix analysis (e.g., square roots), physics, ...

Next topic: *how to compute them?* Many methods:

- Factorizations: eigendecompositions, Schur...
- Interpolation / approximation: replace f with a polynomial or a rational function.
- Complex integrals + quadrature,
- Matrix versions of scalar iterations (e.g., Newton on $x^2 = a$),
- Function-specific tricks (e.g., $\exp(2a) = \exp(a)^2$),
- Arnoldi.

Methods for general matrix functions

We start from methods for matrix functions in general, not restricting to specific choices of f. [Higham book, Ch. 4]

Simplest strategy If A diagonalizable, then

$$f(A) = f(V\Lambda V^{-1}) = V f(\Lambda) V^{-1} = V \begin{bmatrix} f(\lambda_1) & & \\ & \ddots & \\ & & f(\lambda_m) \end{bmatrix} V^{-1}.$$

This works fine if A is symmetric/Hermitian/normal, because we take V orthogonal so everything works in a stable way.

Otherwise, errors (forward and/or backward) in the computation of $f(\lambda_i)$ are multiplied by a factor $\kappa(V)$, possibly much higher than the conditioning of the problem.

Even if we ignore the error in the eigendecomposition, the approximate computation of the diagonal values $|f(\lambda_i) - \tilde{f}(\lambda_i)| < \varepsilon$, causes an error

$$\|f(A) - \tilde{f}(A)\| = \|V(f(\Lambda) - \tilde{f}(\Lambda))V^{-1}\| \leq \kappa(V)\varepsilon,$$

so you may expect trouble if A is non-diagonalizable (again!) or close to it.

We need better algorithms for *non-normal* A.

Matlab example

```
>> A = [3 -1; 1 1+1e-15]
A =
    3.0000 -1.0000
    1.0000 1.0000
>> [V, D] = eig(A);
>> cond(V)
ans =
   6.7109e+07
>> B = V * sqrt(D) / V;
>> norm(B^2 - A)
ans =
   1.5500e-08
```

Why? A is very close to a non-diagonalizable matrix: $\left[\begin{smallmatrix} 3 & -1 \\ 1 & 1 \end{smallmatrix}\right]$ has a Jordan block of size 2 with eigenvalue 2. Knowing this information, we do much better if we replace $f(A) \approx p(A)$, where $p(A)$ is the polynomial that interpolates $\sqrt{x}$ in $\lambda = 2$ with multiplicity 2.

Matlab example, continued

```
>> C = 1/sqrt(8) * A + 1/sqrt(2) * eye(2);
>> norm(C^2 - A)
ans =
   4.4409e-16
```

Note that $p(A) \approx f(A)$ is not an exact inequality though! This $p(x)$ is not the polynomial that appears in one of the definitions of matrix functions, since the spectrum of A is not exactly $\{2, 2\}$.

The correct thing to do would have been considering a full Taylor series for f, which contains higher-order terms. Indeed, if all the eigenvalues of A are sufficiently close to α, then a Taylor series in α converges to $f(A)$; we shall prove it soon.

Observation A good computable choice for α is the arithmetic mean of the eigenvalues,

$$\alpha = \frac{\alpha_1 + \alpha_2 + \cdots + \alpha_n}{n} = \frac{1}{n}\operatorname{trace}(A).$$

```
>> alpha = trace(A) / 2;
>> I = eye(2);
>> D = alpha^(1/2)*I + 1/2*alpha^(-1/2)*(A-alpha*I) ...
   - 1/4*alpha^(-3/2)*(A-alpha*I)^2
>> norm(D^2 - A)
ans =
   4.4409e-16
```

In the case of our matrix, the series converges spectacularly fast because A is approximately a Jordan block and hence $(A - \alpha I)^2 \approx 0$.

More generally, the more clustered the eigenvalues, the faster the convergence.

Taylor series (and variants, such as rational approximants $f(x) = \frac{p(x)}{q(x)} + \mathcal{O}(x^{\deg p + \deg q + 1})$, which we will see in more detail) work best when the eigenvalues of A are in a small region, but in general it is better to combine them with other methods.

Convergence of Taylor series

Theorem [Higham book Thm. 4.7]
Suppose $f = \sum_{k=0}^{\infty} f_k (x - \alpha)^k$, with $f_k = \frac{f^{(k)}(\alpha)}{k!}$, is a Taylor series with convergence radius r.

Then,

$$\lim_{d \to \infty} \sum_{k=0}^{d} f_k (A - \alpha I)^k = f(A)$$

for each A whose eigenvalues satisfy $|\lambda_i - \alpha| < r$.

Proof:

- We can reduce to the case when A is a Jordan block.
- The partial sum with d terms has $p_d(\lambda)$ on the diagonal (truncated Taylor polynomial), and its derivatives $\frac{1}{k!} p_d^{(k)}(\lambda)$ on superdiagonals.
- $p_d^{(k)}(x)$ is the Taylor polynomial of $f^{(k)}$ of degree $d - k$, and it has the same radius of convergence $r^{-1} = \limsup(f_k)^{1/k}$, since power series can be differentiated term-by-term.

Issues with Taylor series Taylor series have two problems:

- The cost to evaluate a polynomial of degree d is $O(n^3 d)$, with the Horner rule. There are more efficient methods that reduce the cost to $O(n^3 \sqrt{d})$, but still this can be high if convergence is poor and many terms are required.

- Convergence is poor when the eigenvalues of A are not clustered around a certain α.

Example: exponential of a 2×2 matrix.

$$A = \begin{bmatrix} 0 & \alpha \\ -\alpha & 0 \end{bmatrix}, \quad \exp(A) = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}.$$

For $\alpha = 30$, even summing a lot of terms gives poor precision, because the intermediate terms of the series grow a lot (the "hump phenomenon") with respect to the final result: *cancellation*.

This is a problem encountered also in the scalar case; the classical example is computing $\exp(-30)$. In that case, we can avoid the issue by switching to the alternative formula $\exp(-30) = 1/\exp(30)$, but in this matrix case there is not a simple solution.

Extras: Polynomial evaluation

How to evaluate polynomials in a matrix argument?

- *Direct evaluation*: compute powers of X by successive products, take a linear combination of them).
- *Horner method*: $(\dots(((a_dX + a_{d-1})X + a_{d-2})X + \dots)X + a_0I$

Bulk of the cost: $d-1$ matrix products, in both cases. Unlike the scalar case, the two methods are essentially equivalent in terms of cost.

Cheaper: divide the terms into 'chunks' of size approx. $\sqrt{d}$, e.g.,

$$(p_8A^2 + p_7A + p_6)(A^3)^2 + (p_5A^2 + p_4A + p_3)A^3 + (p_2A^2 + p_1A_1 + p_0).$$

This is known as *Paterson-Stockmayer* method. Fewer multiplications, but requires more storage.

Stability of polynomial evaluation methods

All these polynomial evaluation methods are stable only with respect to the 'absolute value' polynomial.

Theorem

The value $\tilde{Y}$ computed by any of these methods satisfies

$$|\tilde{Y} - p(X)| \leq O(d\mathbf{u})(|p_0| + |p_1||X| + |p_2||X|^2 + \dots + |p_d||X|^d).$$

All OK if p and X only contain nonnegative values, but in all cases in which there is cancellation this could be troublesome (an example later).

Parlett recurrence

When Jordan is unstable, use Schur. Can one compute matrix functions using the Schur form of A?

If $A = VTV^{-1}$, $f(A) = Vf(T)V^{-1}$, so we reduce to the triangular case.

Example

$$T = \begin{bmatrix} t_{11} & t_{12} \\ 0 & t_{22} \end{bmatrix}, \quad f(T) = S = \begin{bmatrix} s_{11} & s_{12} \\ 0 & s_{22} \end{bmatrix}.$$

Clearly, $s_{11} = f(t_{11})$, $s_{22} = f(t_{22})$. But how to compute the remaining term s_{12}?

Trick: expanding $Af(A) = f(A)A$ gives an equation for s_{12}:

$$t_{11}s_{12} + t_{12}s_{22} = s_{11}t_{12} + s_{12}t_{22} \implies s_{12} = t_{12}\frac{s_{11} - s_{22}}{t_{11} - t_{22}}.$$

If $t_{11} = t_{22}$, the equation is not solvable and we already know (at least when $t_{12} = 1$) that the finite difference should be replaced by a derivative.

Parlett recurrence — II

The same idea works for larger blocks (provided we compute things in the correct order):

$$T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ & t_{22} & t_{23} \\ & & t_{33} \end{bmatrix}, \quad f(T) = S = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ & s_{22} & s_{23} \\ & & s_{33} \end{bmatrix},$$

$$t_{11}s_{13} + t_{12}s_{23} + t_{13}s_{33} = s_{11}t_{13} + s_{12}t_{23} + s_{13}t_{33}.$$

This formula requires elements from the first subdiagonal s_{12} and s_{23}, which we have already computed. More generally,

$$t_{ii}s_{ij} - s_{ij}t_{jj} = \sum_{i \le k < j} s_{ik}t_{kj} - \sum_{i < k \le j} t_{ik}s_{kj},$$

and the RHS includes only elements from lower subdiagonals which have already been computed.

Hence we can set up a back-substitution very similar to the Bartels–Stewart algorithm.

(In essence, we are solving the (singular) Sylvester equation $TX - XT = 0$ with the back-substitution technique that we have already seen, after computing the diagonal elements by hand to obtain the unique solution with $X_{ii} = f(A_{ii})$.)

To turn this into Matlab code, we need a few more observations. Each term s_{ij} depends only on terms on *below it* on the same column and terms *to its left* on the same row. Hence we can solve *column-by-column* starting from the first rather than superdiagonal-by-superdiagonal; this is easier to write and more cache-friendly.

```
function S = funm_parlett(f, T)
% computes f(T) for upper triangular T
% with the Parlett recurrence
n = size(T, 1);
S = zeros(n, n);
for j = 1:n
    S(j,j) = f(T(j,j));
    for i = j-1:-1:1
        S(i,j) = S(i, i:j-1) * T(i:j-1, j) ...
               - T(i, i+1:j) * S(i+1:j, j);
        S(i,j) = S(i,j) / (T(i,i) - T(j,j));
    end
end
```

Problem: due to the $T(i,i) - T(j,j)$ denominator, this formula becomes very unstable when there are equal, or close-by, eigenvalues.

```
>> T = triu(ones(8) + 1e-5*randn(8));
>> S = funm_parlett(@sqrt, T);
>> norm(S^2 - T)
ans =
   1.7648e+21
```

Solution: the previous formulas also work *blockwise*, and they become Sylvester equations. If we can partition the eigenvalues into *well-separated clusters*, then we can use Taylor expansion on each cluster.

Parlett recurrence — III

Algorithm (Schur–Parlett method)

1. Compute Schur form $A = QTQ^*$;
2. Reorder T so that it can be partitioned into blocks with 'well-separated eigenvalues' (with a configurable threshold);
3. Compute $f(T_{ii})$ for each block (e.g., with a Taylor series centered in the average of the cluster);
4. Use recurrences to compute off-diagonal blocks of $f(T)$;
5. Return $f(A) = Qf(T)Q^*$.

This algorithm tries to get the best of both worlds: we use Taylor expansion when the eigenvalues are close, and recurrences when they are distant.

Matlab's `funm` does this:

```
>> T = triu(ones(8) + 1e-5*randn(8));
>> [S, ~, details] = funm(T, @sin);
>> details
  struct with fields:

    terms: 10
      ind: {[1 2 3 4 5 6 7 8]}
      ord: [1 1 1 1 1 1 1 1]
        T: [8x8 double]
>> T = triu(randn(10));
>> [S, ~, details] = funm(T, @sin);
>> details
  struct with fields:

    terms: [1 1 8 1 1 1 1 1]
      ind: {8x1 cell}
      ord: [6 8 6 7 5 4 3 2 1 6]
        T: [10x10 double]
```

Problems with Schur–Parlett

This is more or less the state-of-the-art method for generic functions, and performs well in most cases, but it is not free of problems.

- What happens if the eigenvalues are not naturally divided into clusters? E.g., a single big chunk, or a long string. Applying a degree-d Taylor polynomial to a $k \times k$ block (with k up to n) costs $O(n^3 d)$ (or $O(n^3\sqrt{d})$ with better algorithms), so if $d \sim n$ the cost is more than cubic.
- The correct metric to use to predict accuracy is not the difference between eigenvalues, but $\operatorname{sep}(T_{ii}, T_{jj})$, which is more complicated to handle.
- Derivatives must be known (or computable). Note that `funm` cheats: it has hard-coded derivatives for a small number of standard functions like `@exp` or `@sin`, and for all other functions one must provide explicit derivatives. (Indeed, computing derivatives automatically is more complicated.)

Extras: Parlett recurrence and block diagonalization

The Parlett recurrence is very similar to computation via *block diagonalization*.

Consider the case of 2 blocks for simplicity. T can be block-diagonalized via

$$W^{-1}TW = \begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} T_{11} & \\ & T_{22} \end{bmatrix}$$

where X solves $T_{11}X - XT_{22} + T_{12} = 0$ (Sylvester equation). Then

$$f(T) = W \begin{bmatrix} f(T_{11}) & \\ & f(T_{22}) \end{bmatrix} W^{-1} = \begin{bmatrix} f(T_{11}) & Xf(T_{22}) - f(T_{11})X \\ & f(T_{22}) \end{bmatrix}.$$

(Note indeed that $S = Xf(T_{22}) - f(T_{11})X$ solves the Sylvester equation appearing in the Parlett recurrence.)

So both methods solve a Sylvester equation with operator $Z \mapsto T_{11}Z - ZT_{22}$.

Chapter 5

Automatic differentiation

Intermezzo: some words on *automatic differentiation*: a method that allows one to compute accurately derivatives of arbitrary functions on a computer. This is a topic that has become very popular in the recent years because of interest in machine learning and because of better support from many programming languages (Julia in the first place).

Problem
Given code `function y = f(x)` to compute a function $f : \mathbb{R} \to \mathbb{R}$, how does one compute (or approximate) $f'(x)$ in a given point?

```
function y = f(x)
z = x * x;
w = x + 5;
y = z * w;
```

5.1 Numerical differentiation

First attempt: *numerical differentiation*: compute $g = \frac{f(x+h)-f(x)}{h}$, with a fixed $h > 0$.

Problem: Two sources of error:

- analytical error: $g - f'(x) = \frac{1}{2}f''(\xi)h$ for a nearby point ξ (Taylor expansion).
- numerical error: because of machine arithmetic, even with perfect code we can compute only $f(x)(1+\delta_1)$ and $f(x+h)(1+\delta_2)$ with $|\delta_i| < \mathbf{u}$. So the computed value $\tilde{g}$ of $g = \frac{f(x+h)-f(x)}{h}$ is affected by an error that we can bound with $\mathbf{u}\frac{|f(x)|+|f(x+h)|}{h}$

So for the total error we have

$$|\tilde{g} - f'(x)| \leq |\tfrac{1}{2}f''(\xi)|h + \mathbf{u}\frac{|f(x)| + |f(x+h)|}{h}.$$

Assuming $|\frac{1}{2}f''(\xi)|, |f(x)|, |f(x+h)| = \mathcal{O}(1)$, this error bound is minimized when $h \approx \mathbf{u}^{1/2}$ and is $\mathcal{O}(\mathbf{u}^{1/2})$.

Hence when computing derivatives numerically with the *forward difference*

$$f'(x) \approx \frac{f(x+h) - f(x)}{h},$$

the best accuracy is attained when $h \approx \mathbf{u}^{1/2}$, and the error is $\mathcal{O}(\mathbf{u}^{1/2})$. Importantly, this means that this method is not able to compute derivatives to full precision.

Exercise 5.1. Prove an analogous estimate for the *centered difference*

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}.$$

The final result should be that the error is minimized for $h = \mathcal{O}(u^{1/3})$, leading to an error of $\mathcal{O}(u^{2/3})$.

Example

```
>> x = 5
x =
     5
>> h = 1e-4; g = (f(x+h) - f(x)) / h
g =
     1.250020000097152e+02
% error ≈ 10^-4
>> h = 1e-8; g = (f(x+h) - f(x)) / h
g =
     1.250000025265763e+02
% error ≈ 10^-8
>> h = 1e-12; g = (f(x+h) - f(x)) / h
g =
     1.249986780749168e+02
% error ≈ 10^-4
```

Complex step differentiation

A similar trick: if f is holomorphic, and your code to compute it works also for complex inputs, then for $x \in \mathbb{R}$ one can write

$$f(x + ih) = f(x) + f'(x)ih - \frac{f''(x)}{2}h^2 + O(h^3),$$

so $g = \frac{\operatorname{Im} f(x+ih)}{h}$ is an approximation of the derivative $f'(x)$ with error $g - f'(x) = O(h^2)$.

Typically, the numerical error on $\operatorname{Im} f(x+ih)$ is $\sim |\operatorname{Im} f(x+ih)|\mathbf{u} = \mathcal{O}(h)\mathbf{u}$ (but if real/imaginary parts are 'mixed' in computation, it may get as large as $\sim |f(x)|\mathbf{u} = \mathcal{O}(\mathbf{u})$). Hence

$$|\tilde{g} - f'(x)| \leq \mathcal{O}(h^2) + \frac{\mathcal{O}(h)}{h}\mathbf{u}.$$

The total error is $\mathcal{O}(\mathbf{u})$ as long as $h \leq \mathcal{O}(\mathbf{u}^{1/2})$.

Example

```
>> x = 5
x =
     5
>> h = 1e-4; g = imag(f(x+1i*h)) / h
g =
     1.249999999900000e+02
>> h = 1e-8; g = imag(f(x+1i*h)) / h
g =
     1.250000000000000e+02
>> h = 1e-12; g = imag(f(x+1i*h)) / h
g =
     1.250000000000000e+02
```

Key idea
We obtained a better bound by exploiting the fact that our code runs also for a *more general type* (complex numbers).

5.2 Automatic differentiation

Automatic differentiation via matrix functions

Suppose our code works also for *matrix* arguments x, which we can achieve in Matlab with some changes:

```
function y = f(x)
n = size(x, 1);
z = x * x;
w = x + 5*eye(n);
y = z * w;
```

Then,

$$f\left(\begin{bmatrix} \lambda & 1 & \\ & \lambda & 1 \\ & & \lambda \end{bmatrix}\right) = \begin{bmatrix} f(\lambda) & f'(\lambda) & \frac{f''(\lambda)}{2} \\ & f(\lambda) & f'(\lambda) \\ & & f(\lambda) \end{bmatrix}.$$

No "small h" and subtractions are needed this time $\implies$ the derivative can be computed with error $\mathcal{O}(\mathbf{u})$.

Automatic differentiation

This is a form of *automatic differentiation*.

- It typically computes derivatives up to *machine precision* error $\mathcal{O}(\mathbf{u})$, by running the code with *more general types*.
- It can compute also higher derivatives.
- It works also for $x \in \mathbb{C}$, unlike the complex step trick.
- It is something fundamentally different from numerical differentiation; it is more similar to *symbolic differentiation* with a computer algebra system, but easier to do algorithmically.
- It works also if your code includes *loops, conditionals, and more complicated functions*.

```
function y = somefunction(x)
a = x*x + 1;
z = 2 / a;
while z < 5
  z = z^2;
end
y = exp(z);
```

This function is not continuous at "decision points" (when $z = 5$ at some iteration of the `while`). However, in all other points it is C^∞, and we can compute its derivatives with the same method.

```
function y = somefunction(x)
n = size(x, 1);
a = x*x + eye(n);
z = 2 * inv(a);
while z(1,1) < 5
  z = z^2;
end
y = expm(z);
```

What is going on

Actually, we do not need matrices here: all operations are on triangular Toeplitz matrices, i.e., polynomials in

$$\varepsilon := \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

So

$$\begin{bmatrix} a & b & c \\ 0 & a & b \\ 0 & 0 & a \end{bmatrix} = aI + b\varepsilon + c\varepsilon^2.$$

We can just work in the polynomial algebra $\mathbb{C}[\varepsilon]$, with the relation $\varepsilon^3 = 0$.

This interpretation reveals that what we are really doing is *propagating Taylor expansions*: instead of the input x, for instance $x = 5$, we start from $5 + \varepsilon$, and whenever we compute a variable we compute the first n coefficients of its Taylor expansion in ε alongside it; for instance given code

```
function y = f(x) % input: x=5
z = x * x; % z is 25
w = x + 5; % w is 10
y = z * w; % y is 250
```

we can use it to compute two derivatives ($n = 3$) alongside it:

```
function y = f(x) % input: x = 5 + ε = 5 + 1ε + 0ε^2 + O(ε^3)
z = x * x; % z is (5 + 1ε + 0ε^2 + O(ε^3))(5 + 1ε + 0ε^2 + O(ε^3))
% z = 25 + 10ε + 1ε^2 + O(ε^3)
w = x + 5; % w is (5 + 1ε + 0ε^2 + O(ε^3)) + 5
% w = 10 + 1ε + 0ε^2 + O(ε^3)
y = z * w; % y is (25 + 10ε + 1ε^2 + O(ε^3))(10 + 1ε + 0ε^2 + O(ε^3))
% y = 250 + 125ε + 20ε^2 + O(ε^3)
```

From this Taylor expansion we can read off the first two derivatives of $y = f(x)$ in $x = 5$.

How do we get the computer to do all this automatically without writing ad-hoc code? With *object oriented programming*.

- Define a *class* `Taylor` that encapsulates a length-3 vector.
- Write special code that is run when one writes `a+b` or `a*b` with `Taylor` arguments (*operator overloading*). This is done in Matlab by defining *methods* `plus` and `mtimes`.

```
function y = f(x) % input: x = Taylor[5 1 0]
z = x * x; % z = Taylor[5 1 0] * Taylor[5 1 0]
% z = Taylor[25 10 1]
w = x + 5; % w = Taylor[5 1 0] + Taylor[5 0 0]
% w = Taylor[10 1 0]
y = z * w; % y = Taylor[25 10 1] * Taylor[10 1 0]
% y = Taylor[250 125 20]
```

Rules for operations:

- A real `a` is converted to `Taylor[a 0 0]`
- `Taylor[a0, a1, a2] + Taylor[b0, b1, b2] = Taylor[a0+b0, a1+b1, a2+b2]`
- `Taylor[a0, a1, a2] * Taylor[b0, b1, b2] = Taylor[a0*b0, a1*b0+a0*b1, a2*b0+a1*b1+a0*b2]`

Matlab is not the best language in the world, but it can do OOP, too.

In Matlab

```
classdef Taylor
   properties
      coeffs %length-3 vector
   end
   methods
      function obj = Taylor(v) %constructor
         obj.coeffs = v;
      end
      function c = plus(a, b)
         if isa(b, 'double'), b = Taylor([b 0 0]); end
         c = Taylor(a.coeffs + b.coeffs);
      end
      function c = mtimes(a, b)
         c = Taylor([a.coeffs(1)*b.coeffs(1), ...
           a.coeffs(1)*b.coeffs(2) + a.coeffs(2)*b.coeffs(1),
           a.coeffs(1)*b.coeffs(3) + a.coeffs(2)*b.coeffs(2) + a.coeffs(3)*b.coeffs(1)]);
      end
   end
end
```

Automatic differentiation, generically

For any elementary operation $z = f(a, b, \dots)$, we can update derivatives alongside according to composite-function differentiation rules:

$$\begin{aligned} z' &= \frac{\partial f}{\partial a} a' + \frac{\partial f}{\partial b} b' + \dots \\ z'' &= \frac{\partial^2 f}{\partial a^2} (a')^2 + \frac{\partial f}{\partial a} a'' + \frac{\partial^2 f}{\partial b^2} (b')^2 + \frac{\partial f}{\partial b} b'' + \dots \\ &\vdots \quad \vdots \end{aligned}$$

The formulas get lengthy for higher derivatives.

Global derivatives updated according to local rules for each line of code. Each operation that we use needs to be "augmented" to specify how it acts on derivatives. If we define for our type `Taylor` each operation appearing in our code $(ab, a/b, \exp(a), \dots)$, we can effectively compute derivatives algorithmically.

Again, the key is having code that supports different *types* and operator overloading.

Special case: dual numbers

The most common case is when one only needs one derivative.

An algebraic structure that reflects the formalism for this case: the ring of *dual numbers*, i.e., $\mathbb{R}[\varepsilon]/(\varepsilon^2)$.

- Each member of the ring can be written as $a + \varepsilon b$, for $a, b \in \mathbb{R}$ (or any other base ring).
- Operations are performed with usual algebraic rules plus $\varepsilon^2 = 0$; for instance, `a * b` when its argument are dual numbers becomes $(a + \varepsilon a')(b + \varepsilon b') = ab + (a'b + ab')\varepsilon$.
- $f'(x)$ is equal to the "epsilon part" of $f(x + \varepsilon)$.

5.3 Reverse mode

This is called *forward mode* of automatic differentiation. There is also a *reverse mode* which is more popular in some contexts; most notably machine learning, where it is known as *back-propagation*.

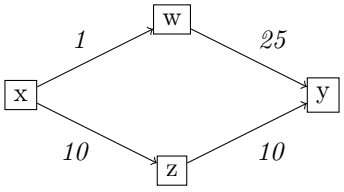
We give an idea of how it works, for $n = 1$ derivatives.

General idea: After having computed $y = f(x)$, revisit your code backwards line-by-line and for each intermediate variable a appearing in it determine iteratively $\frac{dy}{da}$.

This manipulation requires more complicated source code transformations to the code than forward-mode; introducing new types is not sufficient.

Typically it is performed via a *computational graph representation* of the operations in the code.

Reverse mode: example




```
function y = f(x) % input: x=5
z = x * x; % ∂z/∂x = 2x = 10
w = x + 5; % ∂w/∂x = 1
y = z * w; % ∂y/∂w = z = 25, ∂y/∂z = w = 10
```

Using these *edge derivatives*, we work our way right-to-left and compute starting from $\frac{\partial y}{\partial y} = 1$:

$$\frac{dy}{dw} = \frac{dy}{dy}\frac{\partial y}{\partial w} = 25, \quad \frac{dy}{dz} = \frac{dy}{dy}\frac{\partial y}{\partial z} = 10,$$

$$\frac{dy}{dx} = \frac{dy}{dw}\frac{\partial w}{\partial x} + \frac{dy}{dz}\frac{\partial z}{\partial x} = 25 \cdot 1 + 10 \cdot 10 = 125.$$

The multivariate case

The same ideas (both forward and reverse mode) work in the multivariate case, with *Jacobian matrices* in place of scalar derivatives. Composition $\frac{\partial y}{\partial w}\frac{\partial w}{\partial x}$ $\rightsquigarrow$ matrix multiplication.

For a function $f : \mathbb{R}^n \to \mathbb{R}^m$:

Forward mode: for each intermediate variable $w \in \mathbb{R}^p$, store its Jacobian $\frac{dw}{dx} \in \mathbb{R}^{p\times n}$. Whenever an instruction computes z from w, multiply on the *left*: $\frac{dz}{dx} = \frac{\partial z}{\partial w}\frac{dw}{dx}$.

Reverse mode: for each intermediate variable $w \in \mathbb{R}^p$, store its Jacobian $\frac{dy}{dw} \in \mathbb{R}^{m\times p}$. During the reverse part, for each z on which w depends, multiply on the *right*: $\frac{dy}{dz} = \frac{dy}{dw}\frac{\partial w}{\partial z}$.

Forward mode can also compute only a directional derivative $\frac{dy}{dv}$ in a direction v, i.e., the matrix-vector product $\frac{dy}{dx}v$: just start from $\frac{dx}{dx}v = v$ instead of $\frac{dx}{dx} = I$.

Similarly, reverse mode can be used to compute only $w^T\frac{dy}{dx}$, but this is typically less useful.

Complexity

Which one to use?

When one among n and m is much larger than the other, the cheapest mode is the one that works with *smaller matrices*.

For a function $f : \mathbb{R}^n \to \mathbb{R}^m$ with $n \ll m$ that is the composition of many steps, the *forward mode* is faster because all intermediate Jacobians are tall-thin.

For a function $f : \mathbb{R}^n \to \mathbb{R}^m$ with $n \gg m$ that is the composition of many steps, the *reverse mode* is faster, because all intermediate Jacobians are short-fat.

Machine learning = fitting functions with a large number of parameters $\beta \in \mathbb{R}^n$, by minimizing a certain scalar error function $E(\beta) \in \mathbb{R}$. And that's why *reverse mode* (backpropagation) is used there.

Neural network training happens via gradient descent, so reverse-mode automatic differentiation (known in the field as back-propagation) is one of the main ingredients.

Chapter 6

The matrix exponential

The matrix exponential

We will now discuss some specific important matrix functions. First one:

$$\exp(A) = I + A + \frac{1}{2}A^2 + \frac{1}{3!}A^3 + \dots.$$

Note: in Matlab, `exp(A)` does entrywise exponentiation; one must use `expm` to compute the matrix function. This non-feature is often annoying, exactly like the fact that `A + 1` does not return $A + I$.

Useful to recall it: the solution of the ODE initial value problem

$$\frac{d}{dt}v(t) = Av(t), \quad v(0) = v_0 \tag{6.1}$$

is $v(t) = \exp(At)v_0$. *Proof*: we can differentiate term-by-term

$$v(t) = v_0 + tAv_0 + \frac{t^2}{2}A^2v_0 + \frac{t^3}{3}A^3v_0 + \dots.$$

For this reason, often we are concerned with computing $\exp(At)$ for several values of $t \in \mathbb{R}$.

Nice fact: applying the explicit Euler method to the IVP (6.1) produces the approximation $\exp(A) \approx (I + \frac{1}{n}A)^n$.

Note that $\exp(A + E) \neq \exp(A)\exp(E)$, in general; this equality holds only if A and E commute!

How to compute $\exp(A)$**?**

It is easy to come up with ways that turn out to be unstable. [Moler, Van Loan, "Nineteen dubious ways to compute the exponential of a matrix", '78 & '03].

Example truncated Taylor series, $I + A + \frac{1}{2}A^2 + \frac{1}{6}A^3 \cdots + \frac{1}{k!}A^k$.

(See example in the previous slide set.)

Example $\exp(A) \approx (I + \frac{1}{n}A)^n$.

Growth in matrix powers

The main problem: on non-normal matrices, the coefficients exhibit intermediate growth.

Example Even on a nilpotent matrix, entries may grow.

$$A = \begin{bmatrix} 0 & 10 & & \\ & 0 & 10 & \\ & & 0 & 10 \\ & & & 0 \end{bmatrix}, \; A^2 = \begin{bmatrix} 0 & 0 & 100 & \\ & 0 & 0 & 100 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix}, \; A^3 = \begin{bmatrix} 0 & 0 & 0 & 1000 \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix}.$$

Typical behavior for non-normal matrices. In general, we can bound

$$\|\exp(A)\| = \left\| I + A + \frac{1}{2!}A^2 + \frac{1}{3!}A^3 + \ldots \right\| \leq I + \|A\| + \frac{1}{2!}\|A\|^2 + \frac{1}{3!}\|A\|^3 + \cdots = e^{\|A\|},$$

but this bound may be very loose, as there may be important numerical cancellation between the factors: growth + cancellation = trouble.

Exception: normal matrices are nice! Since a normal matrix can be diagonalized orthogonally, $\|A\|_2^k = \|A\|_2^k$, and power series work well.

"Humps"

Similarly, even for a matrix with $\Lambda(A) \subset LHP$, $\exp(tA)$ may grow for small values of t before settling down and converging to 0.

Example [Higham book, Ch. 10]

```
>> A = [-0.97 25; 0 -0.3];
>> t = linspace(0,20,100);
>> for i = 1:length(t); y(i) = norm(expm(t(i)*A)); end
>> plot(t, y)
```

In particular, you can expect intermediate growth and then cancellation that causes instability if you try to compute matrix exponentials $\exp(A)$ with any method that uses power series or "goes through" other values $\exp(At)$ with $t < 1$, for instance by solving the ODE problem

$$X'(t) = AX(t), \quad X(0) = I.$$

Exception: again, everything works well for normal matrices.

$$\|\exp(A)\|_2 = \max_{\lambda \in \Lambda(A)} |e^\lambda| = e^{\max \operatorname{Re}(\lambda)}.$$

In particular, $\|\exp(tA)\| = \|\exp(A)\|^t$, so plotting the norm produces an exponential decrease (or increase) without "humps".

Fréchet derivative of the matrix exponential

Using the same argument that we have used to compute the eigenvalues of Fréchet derivatives, we can obtain

$$L_{\exp,A}[E] = E + \frac{1}{2!}(EA + AE) + \frac{1}{3!}(EA^2 + AEA + A^2E) + \ldots$$

This formula is typically not very useful; a more useful one is the following integral form.

Proposition 6.1.

$$L_{\exp,A}[E] = \int_0^1 \exp(A(1-t))E\exp(At)\,dt.$$

Proof. Expand the integral as

$$\begin{aligned} RHS &= \int_0^1 \sum_{i=0}^{\infty} \frac{A^i(1-t)^i}{i!} E \sum_{j=0}^{\infty} \frac{A^j t^j}{j!}\,dt \\ &= \sum_{i,j=0}^{\infty} A^i E A^j \frac{1}{i!j!} \int_0^1 (1-t)^i t^j\,dt \\ &= \sum_{i,j=0}^{\infty} A^i E A^j \frac{1}{(i+j+1)!} = LHS. \end{aligned}$$

To get to the last line, we have used a classical scalar integral (the one that defines the Beta function). You can have fun computing that by induction, if that's your thing. □

Using the integral formula, we can bound the norm of $L_{\exp,A}[E]$ and hence the condition number of the matrix exponential.

$$\|L_{\exp,A}[E]\| \le \int_0^1 e^{\|A\|(1-t)}\|E\|e^{\|A\|t}\,dt = e^{\|A\|}\|E\|.$$

This is not good news, because we have shown that $e^{\|A\|}$ may be much larger than $\|\exp(A)\|$.

Exception: for a normal matrix, $\|\exp(tA)\| = \|\exp(A)\|^t$, and hence we get the tighter bound

$$\|L_{\exp,A}[E]\| \le \|\exp(A)\|\|E\|,$$

which leads to

$$\kappa_{rel}(\exp, A) \le \|A\|.$$

This bound on the condition number is tight: indeed, when $E = I$ the formula gives $L_{\exp,A}[I] = \|\exp(A)\|$, hence $\kappa_{rel}(\exp, A) \ge \|A\|$ for each matrix A.

6.1 Scaling and squaring

We now describe the method used in Matlab's `expm`, which is the state-of-the-art one.

Padé approximants

Padé approximants (in $x = 0$ here for simplicity) are rational Taylor-like approximations: $f(x) = \frac{N(x)}{D(x)} + \mathcal{O}(x^{p+q+1})$, with $\deg N = p, \deg D = q$.

$p+q+1$ coefficients (up to scaling) to be determined from $p+q+1$ equations.

Padé approximants for the exponential are known explicitly.

Padé approximants of degrees (p, q) to $\exp(x)$

$$N_{pq}(x) = \sum_{j=0}^{p} \frac{(p+q-j)!p!}{(p+q)!j!(p-j)!} x^j,$$
$$D_{pq}(x) = N_{pq}(-x).$$

Since $D(x) = N(-x)$, Padé approximants satisfy the same property as the exponential $\exp(-x) = \frac{1}{\exp(x)}$

Padé approximations of the exponential

We can use this rational approximation to compute matrix functions:

$$\exp(A) \approx (D_{pq}(A))^{-1} N_{pq}(A).$$

Question 1 Is $D_{pq}(A)$ going to be well-conditioned, though?

For $p = q \to \infty$, the jth coefficient of N_{pp} tends to $\frac{1}{2^j j!}$ (direct verification), hence $N_{pp}(x) \approx \exp(-\frac{1}{2}x)$.

Thus $\kappa(D_{pp}(A)) \approx \frac{e^{-\frac{1}{2}\operatorname{Re}\lambda_{\min}}}{e^{-\frac{1}{2}\operatorname{Re}\lambda_{\max}}}$.

$D_{pp}(A)$ is ill-conditioned if $\operatorname{Re}(\lambda_{\max} - \lambda_{\min})$ is large.

Question 2 Can we bound the error of this approximation?

Backward error of Padé approximants

Are Padé approximants reliable when $\|A\|$ is small, at least?

Let $H = f(A)$, where $f(x) = \log(e^{-x}\frac{N_{pq}(x)}{D_{pq}(x)})$. Note that $e^{-x}\frac{N_{pq}(x)}{D_{pq}(x)} = 1 + O(x^{p+q+1})$, so the log exists for x sufficiently small.

H is a matrix function, so it commutes with A, an we can expand as if they were scalars: $\exp(H) = \exp(-A)(D_{pq}(A))^{-1}N_{pq}(A)$, so

$$(D_{pq}(A))^{-1}N_{pq}(A) = \exp(A)\exp(H) = \exp(A + H)$$

(since A and H commute).

We can regard H as a sort of 'backward error': the Padé approximant $(D_{pq}(A))^{-1}N_{pq}(A)$ is the exact exponential of a certain perturbed matrix $A+H$.

Can one bound $\frac{\|H\|}{\|A\|}$?

Bounding $\|H\|$

$H = f(A)$, where $f(x) = \log(e^{-x}\frac{N_{pq}(x)}{D_{pq}(x)})$.

f is analytic, so $f(x) = c_1x^{p+q+1} + c_2x^{p+q+2} + c_3x^{p+q+3} + \dots$.

$$H = f(A) = c_1A^{p+q+1} + c_2A^{p+q+2} + c_3x^{p+q+3} + \dots$$
$$\|H\| \leq |c_1|\|A\|^{p+q+1} + |c_2|\|A\|^{p+q+2} + |c_3|\|A\|^{p+q+3} + \dots$$

All these coefficients can be computed, by hand or with Mathematica (but it's a lot of work).

Luckily, someone did it for us. For instance:

[Higham book '08, p. 244]
If $p = q = 13$ and $\|A\| \leq 5.4$, then $\frac{\|H\|}{\|A\|} \leq \mathbf{u}$ (machine precision).

Degree 13 achieves a good ratio between accuracy and number of required operations (with Paterson–Stockmayer + noting that numerator and denominator are of the form $p(x^2) \pm xq(x^2)$.) Evaluating $N_{13,13}$ and $D_{13,13}$ requires 6 matmuls.

6.2 Some Matlab computations

Computing Padé approximants

```
>> syms x a b c d e
>> T = taylor(exp(x), x, 0, 'Order', 5)
T =
x^4/24 + x^3/6 + x^2/2 + x + 1
>> D = x^2+a*x+b;
>> N = c*x^2 + d*x + e;
>> collect(expand(T*D-N))
ans =
x^6/24 + (a/24 + 1/6)*x^5 + (a/6 + b/24 + 1/2)*x^4 + (a/2 + b/6 + 1)*x^3 ...
  + (a + b/2 - c + 1)*x^2 + (a + b - d)*x + b - e
>> C = coeffs(collect(expand(T*D-N)),x)
C =
[ b - e, a + b - d, a + b/2 - c + 1, a/2 + b/6 + 1, a/6 + b/24 + 1/2, a/24 + 1/6, 1/24]
>> S = solve(C(1:5),[a,b,c,d,e]);
>> [S.a, S.b, S.c, S.d, S.e]
ans =
[ -6, 12, 1, 6, 12]
```

Accuracy of Padé approximants

```
>> ezplot(exp(x), -5, 5);
>> hold on;
>> ezplot(pade(exp(x), x, 'Order', [2,2]), -5, 5);
```

We saw that $D(A)^{-1}N(A) = \exp(A+H)$, where $H = f(A)$ corresponds to the matrix function $f(x) = \log(\exp(-x)\frac{N(x)}{D(x)})$

```
>> P = pade(exp(x), x, 'Order', [2,2]);
>> T = taylor(log(exp(-x)*P), 'Order', 20)
T =
- x^19/98035826688 - x^17/7309688832 + x^13/38817792 + x^11/2737152 ...
   - x^7/12096 - x^5/720
>> AbsC = abs(coeffs(T,'All'))
AbsC =
[ 1/98035826688, 0, 1/7309688832, 0, 0, 0, 1/38817792, 0, 1/2737152, ...
   0, 0, 0, 1/12096, 0, 1/720, 0, 0, 0, 0, 0]
>> % Solve |C|(x) = 2e-16
>> double(solve(C * x.^transpose(19:-1:0) - 2e-16, x, 'Real', true))
ans =
   3.1037e-03
```

This shows that the "backward error" of the Padé approximation is bounded by 2×10^{-16} when $\|A\| \leq 3.11 \times 10^{-3}$ — more or less; we have neglected terms past x^{20} in the Taylor expansion; these have very small coefficients but they do count. There is a full proof that bounds rigorously the sum of the series in the interesting paper [Moler, Van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix"].

Remark

The same technique (both to construct Padé approximants and to evaluate their backward stability) can be applied to other functions as well; the exponential is just a nice example.

In general, rational approximation methods work well only when the eigenvalues are in a sufficiently small region.

6.3 Scaling and squaring

What if $\|A\| > 5.4$? Idea: let us use the identity $\exp(A) = (\exp(\frac{1}{s}A))^s$.

Algorithm (scaling and squaring)

1. Find the smallest $s = 2^k$ such that $\|\frac{1}{s}A\| \leq 5.4$.
2. Compute $F = D_{13,13}(B)^{-1}N_{13,13}(B)$, where $D_{13,13}$ and $N_{13,13}$ are given polynomials and $B = \frac{1}{s}A$.
3. Compute F^{2^k} by repeated squaring.

This algorithm is used in Matlab's `expm`, currently (more or less — approximants of degree smaller than 13 are used in some cases).

Note Very recently, better techniques to evaluate matrix polynomials have been found, making Taylor expansions more competitive.

Is scaling and squaring provably stable?

Note that 'humps' may still give problems: $\exp(B)$ may be much larger than $\exp(A) = \exp(B)^{2^k}$, leading to cancellation when one computes the squares.

So scaling and squaring does not avoid the intermediate growth problem either, but it's still the best algorithm available and is stable experimentally.

6.4 Argument reduction

Argument reduction

Suppose $A = B + \tau I$, for a certain $B \in \mathbb{C}^{n\times n}$ and $\tau \in \mathbb{C}$. Then,

$$\exp(A) = \exp(B + \tau I) = \exp(B)\exp(\tau I) = \exp(B)e^{-\tau},$$

since B and τI commute. This formula has two interesting applications.

The first is computational savings: if $\|B\| \ll \|A\|$, it will take fewer scaling and squaring steps to compute $\exp(B)$ rather than $\exp(A)$.

The second is this result that has applications to Markov chains.

Essentially non-negative (or Metzler, or -Z) matrices

Suppose $A_{ij} \geq 0$ for all $i \neq j$ (the elements on the diagonal can be anything). Then $\exp(A)_{ij} \geq 0$ for all i, j.

Proof For a suitable τ, $B = A + \tau I \geq 0$, and hence

$$\exp(A) = \exp(-\tau I + B) = \exp(-\tau I)\exp(B) = e^{-\tau}\sum_{k=0}^{\infty}\frac{1}{k!}B^k \geq 0.$$

This formula uses only sums and products of non-negative numbers, hence there is no numerical cancellation and all computations are accurate. We do not see the details, but the matrix exponential can be shown to be very well-conditioned, even in the componentwise sense. [Shao, Gao, Xue, '14]

Chapter 7

The matrix sign function

The matrix sign function

$$\operatorname{sign}(x) = \begin{cases} 1 & \operatorname{Re} x > 0, \\ -1 & \operatorname{Re} x < 0, \\ \text{undefined} & \operatorname{Re} x = 0. \end{cases}$$

Suppose the Jordan form of A is reblocked as

$$A = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} J_1 & \\ & J_2 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^{-1},$$

where J_1 contains all eigenvalues in the LHP (left half-plane) and J_2 in the RHP. Then,

$$\operatorname{sign}(A) = \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} -I & \\ & I \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^{-1}.$$

$\operatorname{sign}(A)$ is always diagonalizable with eigenvalues ± 1.

Application: computing eigenvalues by bisection

Given $M \in \mathbb{C}^{n\times n}$, set $A := \alpha M + \beta I$ for suitable $\alpha, \beta \in \mathbb{C}$, and compute $S = \operatorname{sign}(A)$. Then, with the notation in the previous slide,

$$\operatorname{Im}(S - I) = \operatorname{Im} \begin{bmatrix} V_1 & V_2 \end{bmatrix} \begin{bmatrix} -2I & \\ & 0 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^{-1} = \operatorname{Im} V_1.$$

Here $\operatorname{Im} V_1$ is the invariant subspace associated to the Jordan chains of A with negative real part (*Hurwitz stable* invariant subspace), i.e., $\{\mu \in \Lambda(M) : \operatorname{Re}(\alpha\mu + \beta) \le 0\}$. In particular, if we let Q be the orthogonal factor in $\texttt{qr}(S - I)$, then

$$Q^* M Q = \begin{bmatrix} M_1 & * \\ 0 & M_2 \end{bmatrix},$$

as A and M have the same Jordan chains and invariant subspaces. In particular M_1 contains the eigenvalues inside the half-plane $\operatorname{Re}(\alpha z + \beta) \le 0$.

This can be seen as the first step of a *bisection procedure* to compute the eigenvalues and eigenvectors of M: we have split the spectrum into two (hopefully equal) subsets; now we can repeat the procedure on M_1 and M_2 with new values of α, β.

7.1 The Schur-Parlett method

Schur–Parlett method

A first algorithm to compute $\operatorname{sign}(M)$ comes from the Schur–Parlett strategy. Compute a Schur decomposition of M, reordered so that eigenvalues in the LHP come first, to obtain

$$Q^* M Q = \begin{bmatrix} A & C \\ 0 & B \end{bmatrix}, \quad \Lambda(A) \subset LHP, \Lambda(B) \subset RHP$$

hence

$$Q^* f(M) Q = \begin{bmatrix} -I & Z \\ 0 & I \end{bmatrix}$$

where Z solves

$$AZ - ZB = f(A)C - Cf(B) = -2C. \tag{7.1}$$

We can then summarize the Schur–Parlett algorithm for the matrix sign as follows.

1. Compute a Schur decomposition $M = QTQ^*$.
2. Reorder it so that eigenvalues in the LHP come first.
3. Compute Z by solving the Sylvester equation (7.1) (which has triangular coefficients).
4. $\operatorname{sign}(M) = Q \left[\begin{smallmatrix} -I & Z \\ 0 & I \end{smallmatrix}\right] Q^*$.

```
function S = sign_schurparlett(M)
% Computes the matrix sign function with the Schur-Parlett method

n = size(M, 1);
[Q, U] = schur(M, 'complex');
% overwrite Q, U with their reordered version
[Q, U] = ordschur(Q, U, "lhp");

% count the number of eigenvalues in the LHP
p = sum(real(diag(U)) < 0);

A = U(1:p, 1:p);
B = U(p+1:n, p+1:n
```

```
C = U(1:p, p+1:n);

% Matlab function to solve a Sylvester equation
% Note that Matlab's syntax has different signs
Z = lyap(A, -B, 2*C);

S = zeros(n, n);
S(1:p, 1:p) = -eye(p);
S(1:p, p+1:end) = Z;
S(p+1:end, p+1:end) = eye(n-p);

S = Q*S*Q';

% If M is real, S=sign(M) is supposed to be real,
% but the computed one will have a tiny nonzero imaginary part,
% due to complex arithmetic in the Schur form.
% We remove it if that is the case.

if isreal(M)
    S = real(S);
end
```

Note that this algorithm is worthless for our computing-eigenvalues-via-bisection application, because it requires the Schur form, which reveals the eigenvalues already. Before seeing a different algorithm, we expand on perturbation theory.

7.2 Perturbation theory

The general results that we have obtained for perturbation of functions of matrices show that for a diagonalizable matrix $M = VDV^{-1}$ the condition number of the matrix sign satisfies

$$\kappa_{abs}(\text{sign}, M) \leq \kappa(V)^2 \max_{\substack{\lambda \in \Lambda(M) \cap LHP \\ \mu \in \Lambda(M) \cap RHP}} \frac{1}{|\lambda - \mu|}.$$

This result is slightly misleading, though, because of two reasons:

- The factor $\kappa(V)^2$ may be an overestimate, because if we think in terms of invariant subspaces we see that the difficulty is separating the two invariant subspaces relative to the LHP and RHP, which in general is a more well-conditioned task than a full diagonalization.
- For matrices with small separation, another source of error amplification comes from the fact that $\|\text{sign}(M)\|$ is itself large. Indeed, taking norms

in the vectorized form of (7.1) gives

$$\|Z\| \leq \|(I \otimes A - B^T \otimes I)^{-1}\| \|2C\| = \frac{2\|C\|}{\operatorname{sep}(A,B)}.$$

A more accurate result is the following.

Theorem 7.1 ([Byers Mehrmann He '97]). *Let $M = Q\left[\begin{smallmatrix} A & C \\ 0 & B \end{smallmatrix}\right]Q^*$ as above, with $\operatorname{sep}(A,B) = \delta$, and let $\|E\|_F = \varepsilon$ be sufficiently small. Then,*

1. $$\frac{\|\operatorname{sign}(M+E) - \operatorname{sign}(M)\|_F}{\|\operatorname{sign}(M)\|_F} = \mathcal{O}\left(\frac{\epsilon}{\delta^2}\right).$$

2. *However, the Hurwitz stable invariant subspace of $M+E$ is $Q\left[\begin{smallmatrix} I \\ X \end{smallmatrix}\right]$, where $\|X\|_F = \mathcal{O}(\frac{\epsilon}{\delta})$.*

So the sign is highly ill-conditioned in presence of a small separation δ, but the Hurwitz stable invariant subspace that we can compute with it is better conditioned. This is good news for our application.

Proof. Assume $Q = I$, up to a change of basis. Part 2 follows from the perturbation result for invariant subspaces: if $M + E = \left[\begin{smallmatrix} \tilde{A} & \tilde{C} \\ \tilde{D} & \tilde{B} \end{smallmatrix}\right]$, there exists X with $\|X\|_F = \mathcal{O}(\frac{\varepsilon}{\delta})$ such that

$$\begin{bmatrix} I & 0 \\ -X & I \end{bmatrix} (M+E) \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} = \underbrace{\begin{bmatrix} \tilde{A} + \tilde{C}X & \tilde{C} \\ 0 & \tilde{B} - X\tilde{C} \end{bmatrix}}_{:=\tilde{T}}.$$

We can continue from here to compute the sign.

If ε is sufficiently small, $\Lambda(\tilde{A} + \tilde{C}X) \subset LHP$, $\Lambda(\tilde{B} - X\tilde{C}) \subset RHP$, hence $\operatorname{sign}(\tilde{M}) = \begin{bmatrix} I & \tilde{Z} \\ 0 & I \end{bmatrix}$ for a certain matrix $\tilde{Z}$. Arguing as in the Schur-Parlett method, we must have $\operatorname{sign}(\tilde{M})\tilde{M} = \tilde{M}\operatorname{sign}(\tilde{M})$, hence $\tilde{Z}$ solves

$$(\tilde{A} + \tilde{C}X)\tilde{Z} - \tilde{Z}(\tilde{B} - X\tilde{C}) = 2\tilde{C}.$$

Then,

$$\operatorname{sign}(M+E) = \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} \begin{bmatrix} I & \tilde{Z} \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ -X & I \end{bmatrix}. \tag{7.2}$$

The coefficients of the Sylvester equation for $\tilde{Z}$ are a perturbation of magnitude $\mathcal{O}(\frac{\varepsilon}{\delta})$ of those of $AZ - ZB = 2C$. We can apply the following classical result for perturbation of linear systems.

Lemma 7.2. *Let x be the solution to $Tx = c$, and $\tilde{x}$ be the solution to $(T + \delta_T)\tilde{x} = c$. Then,*

$$\frac{\|\tilde{x} - x\|}{\|x\|} \dot{\leq} \kappa(T) \frac{\|\delta_T\|}{\|T\|}.$$

Note that, unlike its counterpart for a perturbation of the RHS vector c, this result for a perturbation of the matrix T holds only up to terms of order $\left(\frac{\|\delta_T\|}{\|T\|}\right)^2$: indeed, the symbol $\dot{\leq}$ stands for a first-order inequality.

Applying this result to the vectorization of $AZ - ZB = C$, we get

$$\|\tilde{Z} - Z\|_F \dot{\leq} \underbrace{\frac{1}{\operatorname{sep}(A,B)}}_{=\frac{1}{\delta}} \mathcal{O}(\frac{\varepsilon}{\delta})\|Z\|_F = \mathcal{O}(\frac{\varepsilon}{\delta^2})\|Z\|_F.$$

Plugging this into (7.2) shows that $\|\operatorname{sign}(M+E) - \operatorname{sign}(M)\| = \mathcal{O}(\frac{\varepsilon}{\delta^2})\|Z\|_F = \mathcal{O}(\frac{\varepsilon}{\delta^2})\|\operatorname{sign}(M)\|_F$. □

7.3 Newton for the matrix sign

We wish to see that the iteration

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \quad X_0 = M \tag{7.3}$$

satisfies $\lim_{k\to\infty} X_k = \operatorname{sign}(M)$.

To study the behavior of this iteration, let us start from the case when $M = V \operatorname{diag}(\lambda_1, \ldots, \lambda_n)V^{-1}$ is diagonalizable. Then it is easy to see that

$$X_1 = V \operatorname{diag}(f(\lambda_1), \ldots, f(\lambda_n))V^{-1},$$

where $f(x) = \frac{1}{2}\left(x + \frac{1}{x}\right)$, the map that corresponds to the scalar version of the iteration (7.3). Similarly, by induction,

$$X_k = V \operatorname{diag}(f^{\circ k}(\lambda_1), \ldots, f^{\circ k}(\lambda_n))V^{-1}$$

(we use the symbol $f^{\circ k}$ to denote the composition of f with itself k times).

The map $f(x)$ is the one obtained by applying Newton's method to solve the equation $x^2 - 1 = 0$; this explains the name of the method. The map $f(x)$ has two fixed points ± 1, with (locally) quadratic convergence.

Convergence analysis of the scalar iteration

Theorem
The limit of $x_{k+1} = \frac{1}{2}\left(x_k + \frac{1}{x_k}\right)$ is $\operatorname{sign}(x_0)$ (for $\operatorname{Re}(x_0) \neq 0$).

Trick: change of variables (*Cayley transform*)

$$y = \frac{x-1}{x+1}, \text{ with inverse } x = \frac{1+y}{1-y}.$$

If $x \in$ RHP, then $|x+1| > |x-1| \implies y$ inside the unit disk.

If $x \in$ LHP, then $|x-1| > |x+1| \implies y$ outside the unit disk.

(It's a Padé approximant of $\exp(-2x)$, with the same property.)

If $y_k = \frac{x_k - 1}{x_k + 1}$ for each k, then $y_{k+1} = y_k^2$ (check).

$$x_0 \in \text{RHP} \implies |y_0| < 1 \implies \lim_{k\to\infty} y_k = 0 \implies \lim_{k\to\infty} x_k = 1;$$
$$x_0 \in \text{LHP} \implies |y_0| > 1 \implies \lim_{k\to\infty} y_k = \infty \implies \lim_{k\to\infty} x_k = -1.$$

Rational approximations of the step function

Let $g(x) = \frac{1}{2}(x + 1/x)$; then its iterates $g^{\circ k}$ are rational approximations of the step function $\operatorname{sign}(x)$ around -1 and 1.

```
>> syms x
>> g = 1/2*(x + 1/x);
>> g2 = simplify(subs(g, x, g))
>> g3 = subs(g2, x, g)
>> fplot(g, [-2,2])
>> axis([-2 2 -2 2]);
>> hold on
>> fplot(g2, [-2,2])
>> fplot(g3, [-2,2])
```

(They diverge badly around 0, though.)

From the coefficients of `g2, g3`, one can infer that the following general formula holds.

Proposition 7.3.

$$g^{\circ k}(x) = \frac{((1+x)^{2^k})_{\text{even}}}{((1+x)^{2^k})_{\text{odd}}} = \frac{(1+x)^{2^k} + (1-x)^{2^k}}{(1+x)^{2^k} - (1-x)^{2^k}}.$$

Proof. Induction. □

This rational approximant can be obtained by imposing approximation properties in two different points, unlike one for Padé approximant: $g^{\circ k}(x)$ is the only degree-$(k, k-1)$ rational function that satisfies $g^{\circ k}(x) - \operatorname{sign}(x) = \mathcal{O}(x^{2^k})$ for both $x \to \pm 1$.

Alternatively, it can be obtained if one starts from

$$\operatorname{sign}(x) = \frac{(x^2)^{1/2}}{x}$$

and replaces the principal square root $x^{1/2}$ with a Padé approximant in 1.

Convergence analysis of the matrix iteration

A modification of the convergence proof for the scalar case works in the matrix case.

Theorem 7.4. *Let $X_0 = M$ have no purely imaginary eigenvalues. Then, the sequence $X_{k+1} = \frac{1}{2}\left(X_k + X_k^{-1}\right)$ converges to* $\operatorname{sign}(M)$.

Proof. Set $S = \operatorname{sign}(M)$. Note that all the X_k are rational functions of M, so they commute with it and with S. We can assume (up to a change of basis) that M is upper triangular, in Schur form. Then S and X_k are upper triangular, too.

Set

$$Y_k = (X_k - S)(X_k + S)^{-1}.$$

Analyzing eigenvalues: the inverse $(X_k + S)^{-1}$ exists, and we have $\rho(Y_k) < 1$.

$$Y_{k+1} = (X_k^{-1}(X_k^2 + I - 2SX_k))X_k(X_k^2 + I + 2SX_k)^{-1} = Y_k^2.$$

It is clear in this form that we have $Y_k \to 0$.

We can now express X_k as a function of Y_k: since everything commutes,

$$Y_k X_k + Y_k S = X_k - S \implies X_k = S(I + Y_k)(I - Y_k)^{-1}.$$

hence $X_k \to S$. □

The algorithm

1. $X_0 = M$.
2. Repeat $X_{k+1} = \frac{1}{2}(X_k + X_k^{-1})$, until convergence.

We really need to compute a full matrix inverse here; this is unusual in numerical linear algebra.

Scaling

Unfortunately, this iterative method requires a large number of iterations if the starting matrix M has a norm that is particularly large or small. Indeed, if $x_k \gg 1$, then

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{1}{x_k}\right) \approx \frac{1}{2}x_k,$$

and "the iteration is an expensive way to divide by 2" [Higham].

Same if $x_k \ll 1$: the iteration just multiplies by 2.

Solution: we can replace M with μM for any scalar $\mu > 0$, since $\operatorname{sign}(M) = \operatorname{sign}(\mu M)$. A good choice of μ will avoid this initial phase in which the method spends iterations just to get the eigenvalues close to 1.

Choices of scaling

Ideally, we want to choose μ so that the eigenvalues of M are "as close to 1 as possible".

Possibility 1: (determinantal scaling): choose $\mu = (\det M)^{-1/n}$, so that $\det M = 1$. This choice reduces the "mean distance" from 1. This determinant is cheap to compute, since we already need to invert M, and methods to do it (e.g., PLU factorization) typically produce the determinant as a byproduct.

Possibility 2: (spectral scaling): choose μ so that $|\lambda_{\min}(\mu M)\lambda_{\max}(\mu M)| = 1$. We can use a few steps of the power method on M and M^{-1} to obtain cheaply estimate of these two extremal eigenvalues.

Possibility 3: (norm scaling): choose μ so that $\sigma_{\min}(\mu A)\sigma_{\max}(\mu A) = 1$. Again we can use the power method to get cheap estimates.

All these methods work reasonably well in practice. It is important to use one, at least at the first iteration, but which one does not matter much.

(Matlab examples)

Stability of the Newton iteration

The analysis of (floating point) stability of the Newton iteration is complicated. [Bai Demmel '98 and Byers Mehrmann He '97]

Even though the algorithm contains only sums and inversions, it is difficult to assess and propagate the impact of numerical errors in the first steps, which are the most ill-conditioned ones.

TL;DR The stability analysis reflects the results of our conditioning analysis: while the sign in itself is *unstable*, it produces invariant subspaces as good (numerically) as those computed via a reordered Schur decomposition.

Extras: Inversion-free sign

Suppose that we are given M, N such that $A = M^{-1}N$. Can we compute $\operatorname{sign}(A)$ without inverting M? *Yes.*

Idea: suppose that we can find $\hat{M}, \hat{N}$ such that $MN^{-1} = \hat{M}^{-1}\hat{N}$. Then we can write

$$\begin{aligned} X_1 = \frac{1}{2}(A + A^{-1}) &= \frac{1}{2}(M^{-1}N + N^{-1}M) \\ &= \frac{1}{2}M^{-1}(N + MN^{-1}M) \\ &= \frac{1}{2}M^{-1}(N + \hat{M}^{-1}\hat{N}M) \\ &= \frac{1}{2}M^{-1}\hat{M}^{-1}(\hat{M}N + \hat{N}M) \\ &= (\hat{M}M)\,\frac{1}{2}(\hat{M}N + \hat{N}M) =: M_1^{-1}\,N_1. \end{aligned}$$

Similarly one can produce $M_2, N_2, M_3, N_3, \ldots$

How do we actually find $\hat{M}, \hat{N}$ such that $MN^{-1} = \hat{M}^{-1}\hat{N}$?

We can rewrite this relation as $\hat{M}M = \hat{N}N$, or $\begin{bmatrix}\hat{M} & \hat{N}\end{bmatrix}\begin{bmatrix} M \\ -N \end{bmatrix} = 0$. The rightmost matrix has full column rank, if M is invertible; hence we can obtain $\hat{M}, \hat{N}$ from any basis of $\ker \begin{bmatrix} M \\ -N \end{bmatrix}$.

Computing this kernel can be a more well-conditioned task than inverting M and/or N, e.g.,

$$\begin{bmatrix} M \\ -N \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \varepsilon \\ \varepsilon & 0 \\ 0 & 1 \end{bmatrix}.$$

All this is a sort of 'linear algebra on matrix pencils': we map the matrix pencil $N - xM$ to $N_1 - xM_1$. There is one final project on this topic.

Chapter 8

The matrix square root

Next (and last, for us) matrix function: square root. Here and in the following, we focus on the principal square root, i.e., the only square root in the right half-plane:

$$f : \rho e^{i\theta} \mapsto \rho^{1/2} e^{i\frac{\theta}{2}}, \quad \theta \in (-\pi, \pi), \rho \geq 0.$$

We leave $x^{1/2}$ undefined when x is a negative real number. We denote this function with $x^{1/2}$, and the corresponding matrix function with $A^{1/2}$.

$A^{1/2}$ is well-defined unless A has:

- Real eigenvalues $\lambda_i < 0$, or
- Non-trivial Jordan blocks at $\lambda_i = 0$ (because $f(x) = x^{1/2}$ is not differentiable at 0).

Condition number / sensitivity

We have already computed the Fréchet derivative of this function in an earlier example. Let us recall the argument: the Fréchet derivative of $g(Y) = Y^2$ is

$$L_{g,Y}(E) = YE + EY, \quad \widehat{L} = I \otimes Y + Y^T \otimes I.$$

The Fréchet derivative of $f(X) = X^{1/2}$ is its inverse,

$$\widehat{L}_{f,X} = (I \otimes X^{1/2} + (X^{1/2})^T \otimes I)^{-1}.$$

In particular,

$$\|\widehat{L}_{f,X}\| = \operatorname{sep}(X^{1/2}, -X^{1/2}) = \operatorname{sep}(f(X), -f(X)).$$

As we noted earlier, $L_{f,X}$ has eigenvalues with $\frac{1}{\lambda_i^{1/2} + \lambda_j^{1/2}}$, $i, j = 1, \ldots, n$. This shows that f is necessarily ill-conditioned for matrices that either:

- have a small eigenvalue (taking $i = j$), or
- have two complex conjugate eigenvalues close to the negative real axis (because then $\lambda_i^{1/2} \approx ai$, $\lambda_j^{1/2} \approx -ai$).

8.1 The modified Schur method

Let us recall the Schur-Parlett method to compute matrix functions:

1. Reduce to a triangular $U = Q^*AQ$ using a Schur form;
2. Compute the diagonal of $S = f(U)$;
3. Compute the off-diagonal entries of S from $SU = US$. The resulting formula involves a denominator $u_{ii} - u_{jj}$; if this quantity is small or 0, trouble ensues, and to avoid it we must work blockwise.
4. Return $f(A) = QSQ^*$.

In the case of $A^{1/2}$, we have another option: rather than $SU = US$, we can use $S^2 = U$ to get the off-diagonal entries of S:

$$s_{ii}s_{ij} + s_{i,i+1}s_{i+1,j} + \cdots + s_{ij}s_{jj} = u_{ij}. \tag{8.1}$$

The advantage is that now the denominator $s_{ii} + s_{jj}$, which is guaranteed to be nonzero because $s_{ii} + s_{jj} \in RHP$.

The method:

1. Reduce to a triangular $U = Q^*AQ$ using a Schur form;
2. Compute the diagonal of $S = f(U)$;
3. For each $j = 1, 2, \ldots, n$ and $i = j-1, j-2, \ldots, 1$, compute the off-diagonal entry s_{ij} of S by solving the equation (8.1).
4. Return $f(A) = QSQ^*$.

Matlab does something similar in `sqrtm`, but in a divide-and-conquer way: it splits T into two blocks of the same size, computes $S_{11} = f(T_{11})$ and $S_{12} = f(T_{22})$ recursively, and then solves the Sylvester equation $S_{11}S_{12} + S_{12}S_{22} = T_{12}$ to obtain the missing block S_{12}.

Stability of the modified Schur method

In general, not much can be said about the stability of the Schur-Parlett method; for a generic function one cannot easily obtain backward stability results. However, in this case we can prove a stability result.

Theorem 8.1. *Let $U \in \mathbb{C}^{n\times n}$ be an upper triangular matrix. Then, the matrix $\tilde{S}$ computed with machine precision u using the Schur-Parlett variant described above satisfies*

$$\tilde{S}^2 = U + \delta_U, \quad |\delta_U| \le |S|^2\mathcal{O}(n\mathsf{u}).$$

Here, $|M|$ is componentwise absolute value.

Combining this bound with a Schur form and converting it into a normwise bound, we get for a generic matrix A

$$\|\tilde{X}^2 - A\|_F \leq \|X\|^2 \mathcal{O}(n^3 \mathsf{u}).$$

Note that this bound is weaker than backward stability, because in the RHS we have $\|X\|_F^2$ instead of $\|A\|_F = \|X^2\|_F$: and, due to cancellation, the former may be significantly larger.

Proof. In machine arithmetic, we have

$$\tilde{s}_{ij} = (u_{ij} \ominus \tilde{s}_{i,i+1} \odot \tilde{s}_{i+1,j} \ominus \cdots \ominus \tilde{s}_{i,j-1} \odot \tilde{s}_{j-1,j}) \oslash (\tilde{s}_{ii} \oplus \tilde{s}_{jj}).$$

Using multiple times the relation $a \circledast b = (a * b)(1 + \varepsilon)$ and rearranging, we arrive to

$$\begin{aligned} &|u_{ij} - \tilde{s}_{ii}\tilde{s}_{ij} - \tilde{s}_{i,i+1}\tilde{s}_{i+1,j} - \cdots - \tilde{s}_{ij}\tilde{s}_{jj}| \\ &\qquad \leq n\mathsf{u}\,(|u_{ij}| + |\tilde{s}_{ii}||\tilde{s}_{ij}| + |\tilde{s}_{i,i+1}||\tilde{s}_{i+1,j}| + \cdots + |\tilde{s}_{ij}||\tilde{s}_{jj}|) + \mathcal{O}(\mathsf{u}^2). \end{aligned}$$

We can replace $\tilde{s}_{ij}$ with s_{ij}, as this is a second-order change in u, and use

$$|u_{ij}| = |s_{ii}s_{ij} + \cdots + s_{ij}s_{jj}| \leq |s_{ii}||s_{ij}| + \cdots + |s_{ij}||s_{jj}|$$

to get the (i, j) entry of the sought result. □

8.2 Relation to the sign function and matrix iterations

The following result relates the sign function and the matrix square root, showing that we can use one to compute the other and viceversa.

Proposition 8.2. *1. For $A \in \mathbb{C}^{n\times n}$ with no real negative eigenvalues,* $\operatorname{sign}(A) = A(A^2)^{-1/2}$*, where $A^{-1/2}$ is the inverse of $A^{1/2}$.*

2. For $A, B \in \mathbb{C}^{n\times n}$ such that AB has no real negative eigenvalues, (and hence neither does BA),

$$\operatorname{sign}\begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} = \begin{bmatrix} 0 & C \\ C^{-1} & 0 \end{bmatrix}, \quad C = A(BA)^{-1/2}.$$

Proof. 1. It is enough to prove the corresponding scalar identity, $\operatorname{sign}(x) = \frac{x}{(x^2)^{1/2}}$, since algebraic identities between scalar functions extend to the corresponding matrix functions. The quantity x^2 has two square roots, $+x$ and $-x$; the principal square root is x if $x \in RHP$ and $-x \in LHP$, and from here we conclude easily.

2. Use $\operatorname{sign}(A) = A(A^2)^{-1/2}$, and then use the relation $\operatorname{sign}(A)^2 = I$ to show that the $(2, 1)$ block does indeed contain C^{-1}.

□

In particular,

$$\operatorname{sign}\begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix} = \begin{bmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{bmatrix}.$$

This relationship suggests using similar iterations to those used for the matrix sign.

In fact, the first method we start from is another classical method, which (we will see) can be transformed into a variant of the matrix sign iteration.

Newton method on $X^2 - A$

We can run the Newton method on the map $G(X) = X^2 - A$, $G : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$.

Its Jacobian is the Fréchet derivative $L_{G,X}[E] = EX + XE$, hence we have

$$X_{k+1} = X_k - L^{-1}_{G,X_k}[G(X_k)],$$

i.e.,

$$X_{k+1} = X_k - E, \quad \text{where } E \text{ solves } EX_k + X_kE = X_k^2 - A. \tag{8.2}$$

On paper, using this iteration is much more expensive than the Schur method: we must solve a Sylvster equation at each step, and in turn this requires a Schur factorization. Clearly a method that requires computing one Schur factorization per step cannot be better than a method that requires only one Schur factorization plus a (cheap) back-substitution step. However, we can find a cheap closed-form solution to those Sylvester equations.

Lemma 8.3. *Suppose the method* (8.2) *is run with an initial matrix X_0 that commutes with A, for instance $X_0 = \alpha I$ or $X_0 = \alpha A$, for $\alpha > 0$. Then,*

1. *A and X_k commute;*
2. *we can take $E = (2X_k)^{-1}(X_k^2 - A)$ at each step.*

Proof. We prove both points at the same time by induction. Point 1 is obvious for $k = 0$. Once point 1 is established, $(2X_k)^{-1}(X_k^2 - A)$ commutes with A, and we can plug it into the Sylvester equation to check that it satisfies it. This not only proves 2, but shows that E commutes with A and hence also $X_{k+1} = X_k - E$ does. □

After plugging in this formula for E, we obtain the following simpler algorithm.

(Modified) Newton iteration (MN)

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}A), \quad X_0 = \alpha I \text{ or } X_0 = \alpha A.$$

We still have to prove that the Newton method converges to the principal square root rather than to another solution of $X^2 = A$.

Theorem 8.4. *Assume A has no eigenvalues in $\mathbb{R}_-$. Then, the MN and TN iterations converge to the principal square root $A^{1/2}$ for each starting point of the form $X_0 = \alpha I$ or $X_0 = \alpha A$, with $\alpha > 0$.*

Proof. We start from MN. Pre-multiply by $A^{-1/2}$, and use commutativity:

$$A^{-1/2}X_{k+1} = \frac{1}{2}\left(A^{-1/2}X_k + (A^{-1/2}X_k)^{-1}\right).$$

This is the Newton iteration for the matrix sign! Hence $A^{-1/2}X_k \to \operatorname{sign}(A^{-1/2}X_0) = I$.

As the two formulas produce the same sequence of matrices X_n, the same property holds for TN. □

Theory and practice

Problem All of this holds in *exact arithmetic*, but the method often doesn't work in practice in machine arithmetic!

```
format short e; % for better error display
rng(0); M = randn(10); M = M*M';
X = eye(size(M));
Y = eye(size(M));
T = table();
for k = 1:15
    X = X - lyap(X, X, M-X^2); % TN
    Y = 1/2*(Y + Y\M); % MN
    T.TNres(k) = norm(X^2-M)/norm(M);
    T.MNres(k) = norm(Y^2-M)/norm(M);
    T.difference(k) = norm(X-Y)/norm(Y);
    T.TNcommute(k) = norm(X*M-M*X)/norm(M)/norm(X);
    T.MNcommute(k) = norm(Y*M-M*Y)/norm(M)/norm(Y);
end
T
```

```
TNres MNres difference TNcommute MNcommute
---------- ---------- ---------- ---------- ----------

1.2716e+01 1.2716e+01 0.0000e+00 4.5914e-17 4.5914e-17
2.9472e+00 2.9472e+00 9.4733e-16 2.8536e-16 9.2812e-17
5.5013e-01 5.5013e-01 1.9541e-15 6.8206e-16 4.4933e-16
4.8810e-02 4.8810e-02 3.3700e-15 3.5178e-16 3.1431e-15
5.6788e-04 5.6788e-04 2.4031e-14 9.0480e-17 2.3496e-14
8.0577e-08 8.0577e-08 1.8106e-13 1.0374e-16 1.7888e-13
1.5991e-15 1.4569e-12 1.3761e-12 5.7152e-17 1.3656e-12
8.2069e-17 1.1128e-11 1.0497e-11 8.2657e-17 1.0438e-11
9.9611e-17 8.5061e-11 8.0189e-11 1.1024e-16 7.9815e-11
9.1823e-17 6.5044e-10 6.1301e-10 6.9810e-17 6.1043e-10
```

```
8.7288e-17 4.9746e-09 4.6877e-09 8.7521e-17 4.6690e-09
1.0434e-16 3.8049e-08 3.5853e-08 6.9442e-17 3.5713e-08
5.8770e-17 2.9104e-07 2.7423e-07 1.0662e-16 2.7318e-07
8.5101e-17 2.2262e-06 2.0976e-06 6.9301e-17 2.0896e-06
8.5801e-17 1.7029e-05 1.6045e-05 6.1899e-17 1.5984e-05
```

There is nothing apparently wrong with our matrix M, apart with moderate ill-conditioning $\kappa(M) \approx 266$, but we see that already in this simple example MN reaches residual 10^{-12} at best, but then starts to diverge. On the other hand, TN gives good results.

The final two columns `TNcommute` and `MNcommute` hint to a possible reason for this notable discrepancy with the results in exact arithmetic: the matrices computed by MN no longer commute with A, as they were supposed to.

The geometric picture TN, MN coincide on the manifold of matrices that commute with A, $\{X \in \mathbb{C}^{n\times n} \colon AX = XA\}$, but not on the rest of $\mathbb{C}^{n\times n}$. Numerical perturbations take us outside of the manifold, where the two do not coincide anymore.

Being a Newton method, TN is quadratically convergent. However, MN does not even have a *stable fixed point* in $A^{1/2}$: there are starting points arbitrarily close to $A^{1/2}$ for which the sequence diverges.

To prove this formally, we need to recall a few facts from the theory of (discrete-time) dynamical systems.

Dynamical systems

Discrete-time, nonlinear dynamical system
Consider the dynamical system induced by a sufficiently regular map F.

$$x_{k+1} = F(x_k), \quad F : \mathbb{C}^n \to \mathbb{C}^n.$$

Starting from $x_0 = x_* + e$ closed to a *fixed point* $x_* = F(x_*)$,

$$\begin{aligned} x_1 &= F(x_* + e) = x_* + F'_{x_*} e + \mathcal{O}(\|e\|^2), \\ x_k &= x_* + (F'_{x_*})^k e + \mathcal{O}(\|e\|^2). \end{aligned}$$

If $\rho(F'_{x_*}) < 1$, there is exponential convergence to x_*; we call x_* a *stable fixed point*. If $\rho(F'_{x_*}) > 1$, the iterates (for almost all starting points) diverge away from x_*; we call it an *unstable fixed point*.

Note that in some other contexts this convergence speed is called *linear*:

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x_*\|}{\|x_k - x_*\|} \to \rho(F'_{x_*}).$$

In our case, this Jacobian F'_{x_*} is the Fréchet derivative of the map $X_k \mapsto X_{k+1}$.

From standard multivariate Newton results, we have that $L_{TN,A^{1/2}} = 0 \implies$ doubly exponential ("quadratic") convergence.

Local stability

We can study the local stability of the map $h(X) = \frac{1}{2}(X + X^{-1}A)$. Its Fréchet derivative is

$$L_{h,X}(E) = \frac{1}{2}(E - X^{-1}EX^{-1}A).$$

Hence $L_{h,A^{1/2}} = \frac{1}{2}(E - A^{-1/2}EA^{1/2})$, or $\hat{L}_{h,A^{1/2}} = \frac{1}{2}\left(I - (A^{1/2})^T \otimes A^{-1/2}\right)$.

It has eigenvalues $\frac{1}{2} - \frac{1}{2}\lambda_i^{1/2}\lambda_j^{-1/2}$, where λ_i are the eigenvalues of A.

Whenever $\kappa(A)$ is large, $\rho(L_{h,A^{1/2}}) > 1$, hence $A^{1/2}$ is an *unstable fixed point* of $h(X)$.

Denman–Beavers iteration

However, the stability properties are significantly different for slight variations of the modified Newton's method.

Set $Y_k = A^{-1}X_k$ to get the coupled iteration

Denman–Beavers iteration [Denman–Beavers, '76]

$$X_{k+1} = \frac{1}{2}(X_k + Y_k^{-1}),$$
$$Y_{k+1} = \frac{1}{2}(Y_k + X_k^{-1}),$$

Remark The same iteration can be obtained by expanding blocks in the Newton iteration for sign $\left(\left[\begin{smallmatrix} 0 & A \\ I & 0 \end{smallmatrix}\right]\right)$.

Local stability of the DB iteration

Theorem
The DB iteration satisfies $\lim(X_k, Y_k) = (A^{1/2}, A^{-1/2})$, and it is locally stable.

We have

$$L_{DB,(X,Y)}\left(\begin{bmatrix} E \\ F \end{bmatrix}\right) = \frac{1}{2}\begin{bmatrix} E - Y^{-1}FY^{-1} \\ F - X^{-1}EX^{-1} \end{bmatrix}$$

Using the fact that $X_*Y_* = I$, one can verify that the Jacobian is idempotent, i.e., $(K_{DB,(B,B^{-1})})^2 = K_{DB,(B,B^{-1})}$. This shows immediately that it has bounded powers $\implies$ "weak" stability: the error coming from machine arithmetic stays bounded (at least in first-order).

Exercise 8.5. Perform a local stability analysis of the Newton method for the matrix sign. You should be able to conclude that it has the same stability properties as the DB iteration.

Other variants of the MN method are studied on [Higham book, Ch. 6], if you are interested.

Chapter 9

Functions of large-scale matrices

Functions of large-scale matrices

How do we compute $f(A)$ if A is large and sparse? This is a topic of recent research. We can consider it as an extension of methods to solve large-scale linear systems, which is the case $f(x) = x^{-1}$.

Most of the time, one wants $f(A)b$ rather than $f(A)$, because $f(A)$ is full. Some of the main techniques:

1. Replace f with an *approximating polynomial/rational function* on a region U that includes the spectrum of A (how?).

2. *Contour integration*:

$$\frac{1}{2\pi i}\int_\Gamma f(z)(zI-A)^{-1}\mathsf{d}z \approx \sum_{k=1}^{n} w_k f(x_k)(x_k I - A)^{-1}.$$

3. Ad-hoc methods, involving e.g. discretization of *differential equations*: for instance, $\exp(A)b = v(1)$ where $\dot{v}(t) = Av(t)$, $v(0) = b$.

Actually it is not complicated to see that 2. and 3. are special cases of 1., so ultimately the problem is finding good rational approximations.

9.1 Arnoldi for matrix functions

A different possibility, which contains a way to construct a well-suited approximation function, is using the "Swiss-army knife" algorithm for large matrices: Arnoldi.

Let us recap Arnoldi (with matrix functions in mind). Let $A \in \mathbb{C}^{m\times m}$, and $n \leq m$.

Krylov subspaces

$$\begin{aligned} K_n(A,b) &= \text{span}(b, Ab, A^2b, \dots, A^{n-1}b) \\ &= \{p(A)b\colon p \text{ polynomial of degree } < n\}. \end{aligned}$$

The interesting feature of Krylov spaces is that in a problem with a matrix A and a vector b, such as a linear system, most of the "important stuff" happens in the subspace $K_n(A,b)$, so we can replace the problem with its projection on the space $K_n(A,b)$, and in many cases the solution of the projected problem converges quickly to that of the original problem.

Suppose that we have an orthonormal basis V_n of $K_n(A,b)$. We can then define an orthogonal projection matrix $P = V_n V_n^*$. The projection of b is $Pb = b = V_n e_1 \beta$, with $\beta = \|b\|$, while the projected version of A is

$$PAP = V_n \underbrace{(V_n^* A V_n)}_{A_n \in \mathbb{C}^{n\times n}} V_n^*,$$

where $A_n = V_n^* A V_n$. For instance, to solve a linear system $Ax = b$ we can search for $x = V_n y$ that solves the projected problem $V_n^*(Ax - b) = 0$.

Even if there is no vector b involved, Krylov spaces work remarkably well also to compute approximations of eigenvalues and eigenvectors: in particular, for most choices of A and b, one sees that $\Lambda(A_n)$ approximates well the outer eigenvalues of A, i.e., those with larger absolute value. A very nice visual example is on `https://en.wikipedia.org/wiki/Arnoldi_iteration#/media/File:Arnoldi_Iteration.gif`. We will not elaborate on why this is true (also because it does not hold in all cases), but the general idea is that if (λ_i, v_i) are the eigenpairs of a diagonalizable A, and b is written in the eigenvector basis as

$$b = v_1\alpha_1 + \cdots + v_n\alpha_n,$$

then

$$A^k b = v_1\alpha_1\lambda_1^k + \cdots + v_n\alpha_n\lambda_n^k,$$

and the largest components here are those with large $|\lambda_i|$, which shows that $A^k b$ lies approximately in the span of the leading eigenvectors.

To use Krylov subspaces efficiently, one must compute an orthogonal basis V_n. Moreover, as Krylov spaces are nested one into the other

$$K_1(A,b) \subset K_2(A,b) \subset K_3(A,b) \subset \dots$$

$$\text{span}(b) \subset \text{span}(b, Ab) \subset \text{span}(b, Ab, A^2b), \dots,$$

we would like to compute a set of nested orthonormal bases: given an orthonormal basis $(v_1, v_2, \dots, v_j)$ of $K_j(A,b)$, can we find one additional vector v_{j+1} so that $(v_1, v_2, \dots, v_j, v_{j+1})$ is an orthonormal of $K_{j+1}(A,b)$?

The first idea is computing V_j as the Q factor of

$$\text{qr}([b, Ab, \dots, A^{j-1}b]).$$

Unfortunately this idea is doomed to fail, since the columns of this matrix tend to be aligned with each other and then its condition number becomes very high: indeed, by the power method, when k grows $A^k b$ converges (after a suitable normalization) to the maximum-modulus eigenvector of A.

Arnoldi iteration

Idea: it is sufficient to take any vector $w \in K_{j+1}(A,b) \setminus K_j(A,b)$ and orthogonalize it against all the previous vectors, using the Gram-Schmidt process.

```
w = A*V(:,j); % the continuation vector
for i = 1:j
    alpha(i,j) = V(:,i)' * w;
    w = w - V(:,i) * alpha(i,j);
end
alpha(j+1,j) = norm(w);
V(:,j+1) = w / alpha(j+1,j);
```

Starting point: $v_1 = \frac{b}{\beta}$.

Remark: the variant above is called *modified Gram-Schmidt* (MGS): we compute coordinates α_{ij} one by one and we subtract the component $v_i \alpha_{ij}$ from w *immediately*.

Remarks on Arnoldi

This algorithm computes nested bases for the Krylov subspaces:

$$K_j(A,b) = \operatorname{Im} \begin{bmatrix} v_1 & v_2 & \dots & v_j \end{bmatrix}, \quad j = 1, 2, \dots, n.$$

Important detail At each step, we need to generate a vector in $K_{j+1}(A,b)$ that we orthogonalize to all previous vectors. Why did we choose exactly $w = Av_j$ here? Because with this choice we can prove that $\alpha_{j+1,j} \neq 0$.

Lemma 9.1. *Suppose V_j has full column rank. Then,*

1. $v_j \in K_j(A,b) \setminus K_{j-1}(A,b)$
2. $\alpha_{j+1,j} \neq 0$.

Proof. We prove the two statements together by induction on j. We can start from $j = 1$, as long as we set $K_0(A,b) = \{0\}$, and then the first statement is obvious. Note that 1. means that the relation $v_j = p(A)b$ holds with a polynomial p of degree *exactly* $j-1$. Note that this polynomial is uniquely determined because V_j is full-rank.

Hence, before the `for` cycle, we have $w = Av_j = q(A)b$ with $q(x) = xp(x)$ of degree exactly j, i.e., $w \in K_{j+1}(A,b) \setminus K_j(A,b)$. The same property holds for the value of the variable w after the `for` cycle, since at each step we subtract from it an element of $K_j(A,b)$, i.e., a vector of the form $q(A)b$, where $\deg(q) < j$. □

Arnoldi: the associated matrix

Gathering all the relations involving the Av_j in a matrix, we get

$$A\underbrace{\begin{bmatrix} v_1 & \dots & v_n \end{bmatrix}}_{V_n} = \underbrace{\begin{bmatrix} v_1 & \dots & v_{n+1} \end{bmatrix}}_{V_{n+1}} \underbrace{\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \dots & \alpha_{1,n} \\ \alpha_{1,2} & \alpha_{2,2} & \alpha_{2,3} & \dots & \alpha_{2,n} \\ 0 & \alpha_{3,2} & \alpha_{3,3} & \dots & \alpha_{3,n} \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \alpha_{n,n} \\ 0 & \dots & \dots & 0 & \alpha_{n+1,n} \end{bmatrix}}_{\underline{H}_n}.$$

When $\alpha_{n+1,n} = 0$ (breakdown), $AV_n = V_n H_n$, where H_n is $\underline{H}_n$ without the last row. This is an invariant subspace relation.

Note that $H_n = V_n^* A V_n = A_n$. The matrix H_n plays a *double role* here: it gives the action of $A : K_j \to K_{j+1}$ on the Krylov subspace, and it is a projected version of A.

Formula for $p(A)b$

Lemma 9.2. *For all polynomials with* $\deg p < n$,

$$p(A)b = V_n p(A_n) V_n^* b = V_n p(A_n) e_1 \beta \qquad (\beta = \|b\|).$$

Proof. By linearity, it is sufficient to show that $A^j b = V_n H_n^j V_n^* b$ for $j < n$.

$$V_n H_n^j V_n^* = V_n V_n^* A V_n V_n^* A \cdots V_n V_n^* A V_n V_n^* A V_n V_n^* b$$

Let us start from the right. $V_n V_n^*$ is the orthogonal projection matrix onto the Krylov space. Since $b \in K_n(A, b)$, $V_n V_n^* b = b$.

Now the rightmost part reads $V_n V_n^* A b$; but this equals Ab because $Ab \in K_n(A, b)$, and so on.

□

This result suggests an idea: we can take $c = V_n f(A_n) e_1 \beta$ as an approximation of $f(A)$, even for an arbitrary f that is not a polynomial.

Arnoldi, matrix functions, and polynomial approximations

Note that

$$c = V_n f(A_n) e_1 \|b\| = V_n \tilde{p}(A_n) e_1 \|b\| = \tilde{p}(A) b,$$

where $\tilde{p}$ is the interpolating polynomial to f on $\Lambda(A_n)$ (not $\Lambda(A)$!)

Known behavior from Arnoldi theory: for many matrices, the eigenvalues of H_n (*Ritz values*) approximate the *outermost eigenvalues* of A.

What is going on: for a diagonalizable $A = W \Lambda W^{-1}$, we are computing $c = W \tilde{p}(\lambda) W^{-1} b$ instead of $f(A)b = W f(\lambda) W^{-1} b$;

- for eigenvalues "on the outside", $f(\lambda) \approx \tilde{p}(\lambda)$ because $f(\mu) = \tilde{p}(\mu)$ for a nearby Ritz value $\mu \in \Lambda(A_n)$.
- for eigenvalues "on the inside", they may be different, but *hopefully* $|f(\lambda)|$ is smaller and does not contribute too much.

A more precise error bound, for Hermitian A

Theorem
Let A be *Hermitian*, I real interval s.t. $\Lambda(A) = [\lambda_{\min}, \lambda_{\max}] \subset I$. Let $p(x)$ be the best-approximation polynomial to f on I, i.e., the one that attains the minimum of $\delta = \max_{x \in I} |f(x) - p(x)|$. Then,

$$\|f(A)b - c\| \leq 2\delta \|b\|.$$

(And, magically, Arnoldi does all this without knowing p!)

Proof. The eigenvalues of $H_n = H_n^*$ are in I, too: indeed, $Hx = \mu x \implies \mu = \frac{x^* H x}{x^* x} = \frac{x^* V_n^* A V_n x}{x^* V_n^* V_n x}$ is a Rayleigh quotient for A, and Rayleigh quotients can be written as convex combinations of eigenvalues.

Since the Arnoldi approximation is exact on p,

$$\begin{aligned}
\|f(A)b - c\| &= \|f(A)b - V_n f(H_n) V_n^* b\| \\
&= \|(f-p)(A)b - V_n (f-p)(H_n) V_n^* b\| \\
&\leq \|(f-p)(A)b\| + \|V_n (f-p)(H_n) V_n^* b\| \\
&\leq \delta\|b\| + \delta\|b\|.
\end{aligned}$$

□

Hence the Arnoldi approximation is *almost optimal*: it loses only a factor 2 from the best polynomial approximation $\|f(A)b - p(A)b\| \leq \delta\|b\|$.

To prove a similar bound also for non-normal A, we need a few additional results. Define the *field of values* or *numerical range*

$$\mathbb{W}(A) = \left\{ \frac{x^* A x}{x^* x} : x \in \mathbb{C}^n \setminus \{0\} \right\} = \{\text{set of Rayleigh quotients of } A\}.$$

Clearly $\Lambda(A) \subseteq \mathbb{W}(A)$, but this region is not simple to describe in general.

Exercise 9.3. 1. If A is a normal matrix, show that $\mathbb{W}(A) = \text{hull}(\Lambda(A))$. (Idea: start from A diagonal.)

2. If $A = \left[\begin{smallmatrix} 0 & 1 \\ 0 & 0 \end{smallmatrix}\right]$, show that $\mathbb{W}(A) = B(0, 1/2)$, the disc in the complex plane with center 0 and radius 1/2.

It can be proved that for a non-normal matrix $\mathbb{W}(A)$ is a convex set that contains $\text{hull}(\Lambda(A))$. The following technical result, whose proof is far from easy, relates matrix functions and numerical range.

Theorem 9.4 (Crouzeix-Palencia theorem)**.** *Let $\gamma = 1 + \sqrt{2}$. Then, for any matrix A and any function f that is holomorphic on $\mathbb{W}(A)$ we have the inequality*

$$\|f(A)\| \leq \gamma \max_{x \in \mathbb{W}(A)} |f(x)|.$$

Crouzeix's conjecture, which is still an open problem, states that the theorem still holds with the smaller constant $\gamma = 2$.

An error bound for non-normal matrices

Theorem
Let $A \in \mathbb{C}^{n \times n}$, and let $p(x)$ be the best-approximation polynomial to f in $\mathbb{W}(A)$, i.e., the one that attains the minimum of

$$\delta = \max_{z \in \mathbb{W}(A)} |f(z) - p(z)|.$$

Then,

$$\|f(A)b - c\| \leq 2\gamma\delta\|b\|,$$

where γ is the constant in the Crouzeix-Palencia theorem.

Proof As above, $\mathbb{W}(H_n) \subseteq \mathbb{W}(A)$, so $\|(f-p)(H_n)\| \leq \gamma\delta$.

$$\begin{aligned}
\|f(A)b - c\| &= \|f(A)b - V_n f(H_n) V_n^* b\| \\
&= \|(f-p)(A)b - V_n (f-p)(H_n) V_n^* b\| \\
&\leq \|(f-p)(A)b\| + \|V_n (f-p)(H_n) V_n^* b\| \\
&\leq \gamma\delta\|b\| + \gamma\delta\|b\|.
\end{aligned}$$

Arnoldi variants [Güttel '13]

What if f takes its larger values at some internal point of the spectrum of A, e.g., $f(x) = \frac{1}{x}$ and A has eigenvalues "around" 0?

Idea: we can use variants of Arnoldi to compute bases for other spaces.

- *Extended Krylov space*: it is the space of Laurent polynomials

$$\begin{aligned}
K_{n_1,n_2}(A,b) &= A^{-n_1} K_{n_1+n_2}(A,b) = K_{n_1+n_2}(A, A^{-n_1}b) \\
&= \{p(A)b\colon\, p = \alpha_{-n_1} x^{-n_1} + \alpha_{-n_1+1} x^{-n_1+1} + \cdots + \alpha_{n_2-1} x^{n_2-1}\}
\end{aligned}$$

- *Rational Krylov space*: it is the space of rational functions with fixed denominator $q(z)$ of degree $< n$,

$$\begin{aligned}
K_{q,n}(A,b) &= q(A)^{-1} K_n(A,b) = K_n(A, q(A)^{-1}b) \\
&= \{r(A)b\colon\, r(z) = p(z)/q(z),\ p \text{ any polynomial of degree} < n\}
\end{aligned}$$

One can compute bases for these spaces with a similar strategy.

Extended Arnoldi

Idea: to expand an extended Arnoldi space adding a negative power of A, take a suitable continuation vector

$$v = (\alpha_{-n_1} A^{-n_1} + \cdots + \alpha_{n_2-1} A^{n_2-1})b \in K_{n_1,n_2}(A, b)$$

and multiply it by A^{-1} to obtain

$$w = A^{-1}v = (\alpha_{-n_1} A^{-n_1-1} + \cdots + \alpha_{n_2-1} A^{n_2-2})b \in K_{n_1+1,n_2}(A, b).$$

We can now orthogonalize w with respect to the other vectors of the basis. We should ensure that the continuation vector has $\alpha_{-n_1} \neq 0$ to avoid breakdown.

Similarly, to expand it adding a positive power of A, multiply by A as usual: $w = Av$.

A working choice: at each step take $v = v_k$, where k is the last iteration in which you expanded the space from the same side.

Rational Arnoldi

A similar idea works for rational Arnoldi. Let us start from a generic vector v in the rational Arnoldi space $K_{q,j}(A, b)$, with $q(z) = (z-\xi_1)(z-\xi_2)\dots(z-\xi_{j-1})$, i.e.,

$$v = q(A)^{-1}p(A)b$$

for some p with $\deg(p) < j$. Then, for a given $\xi_j \in \mathbb{C}$ we can compute

$$w = (A - \xi_j I)^{-1}v \in K_{q(z)(z-\xi_j),j+1}(A, b).$$

We need to choose a vector v with $z - \xi_j \nmid p(z)$ to avoid breakdown. We won't go into details on how this is done.

One can also include steps of traditional Arnoldi, i.e., take $w = Av$ at certain iterations. This has the effect of raising by 1 the allowed degree of the numerator j, while not raising the degree of the denominator $q(z)$; it can be interpreted as adding a "pole at infinity", projectively.

Putting together all orthogonalization relations yields an equality of the form

$$AV_{n+1}\underline{K}_n = V_{n+1}\underline{H}_n.$$

Again, we can obtain an expression for $A_n = V_n^* A V_n$ from $\underline{K}_n$ and $\underline{H}_n$; we will not go into details on its exact form.

For these two variants, many of the results that we have proved for Arnoldi continue to hold. In particular, we can consider the approximation

$$f(A)b \approx V_n f(A_n)V_n^* b = V_n f(A_n)e_1\beta, \quad A_n = V_n^* A V_n. \tag{9.1}$$

Analogous versions of the approximation results that we have proved for Arnoldi hold for these variants, for instance the following.

Lemma 9.5. *Let V_n be the basis matrix produced by extended (resp. rational) Arnoldi, which we suppose to have full rank. If f is a Laurent polynomial with terms of degrees $-n_1$ to $n_2 - 1$ (resp. a rational function with denominator $q(z)$), then*

$$f(A)b = V_n f(A_n) e_1 \beta.$$

Proofs: use $K_{n_1,n_2}(A,b) = K_{n_1+n_2}(A, A^{-n_1}b)$ or $K_{q,n} = K_n(A, q(A)^{-1}b)$ to reduce to the previous case.

Costs and benefits

Computational cost:

- *Extended Arnoldi*: one needs to solve several linear systems with A^{-1}, one for every negative power of x in the approximation space. These can be computed with a single (sparse) LU factorization of A.
- *Rational Arnoldi*: one needs to solve several linear systems with $(A - \xi_i I)^{-1}$, so we need one mew sparse LU is needed for each new pole.

Both variants are significantly more expensive than Arnoldi, but they compensate by having additional degrees of freedom in the choice of the poles.

Key issue: how much more effective is *rational interpolation* (for your f and A) than *polynomial interpolation*, so that the trade-off is convenient? How to choose good poles ξ_j?

There are many classical and current research results on these aspects. No details here; I am not an expert myself!

More detail in the review paper [Güttel '13].

Matlab examples

```
rng(0); n = 10; A = randn(n,n); b = randn(n,1);
k = 3; % rational Arnoldi with poles 1, 2, \infty
K = zeros(k+1,k); H = zeros(k+1,k); V = zeros(n, k+1);
beta = norm(b); V(:,1) = b / beta;
lambda1 = 1; % step 1
w = (A - lambda1*eye(n)) \ V(:,1);
alpha11 = V(:,1)'*w; w = w - V(:,1)*alpha11;
alpha21 = norm(w); V(:,2) = w / alpha21;
K(1:2,1) = [alpha11; alpha21];
H(1:2,1) = [1+alpha11*lambda1; alpha21*lambda1];
lambda2 = 2; % step 2
w = (A-lambda2*eye(n)) \ V(:,1);
alpha12 = V(:,1)' * w; w = w - V(:,1) * alpha12;
alpha22 = V(:,2)' * w; w = w - V(:,2) * alpha22;
alpha32 = norm(w); V(:,3) = w / alpha32;
K(1:3,2) = [alpha12; alpha22; alpha32];
H(1:3,2) = [1+alpha12*lambda2; alpha22*lambda2; alpha32*lambda2];
```

Matlab examples

```
% lambda3 = inf % step 3
w = A*V(:,1);
alpha13 = V(:,1)'*w; w = w - V(:,1)*alpha13;
alpha23 = V(:,2)'*w; w = w - V(:,2)*alpha23;
alpha33 = V(:,3)'*w; w = w - V(:,3)*alpha33;
alpha43 = norm(w); V(:,4) = w / alpha43;
K(:,3) = [1;0;0;0];
H(:,3) = [alpha13; alpha23; alpha33; alpha43];
norm(A*V*M-V*N) / norm(N) %accuracy check
%formula for the projected matrix when lambda(end)=inf
An = H(1:end-1,:) / K(1:end-1,:);
norm(V(:,1:end-1)'*A*V(:,1:end-1) - An) / norm(An)
```

Usually one takes the *last pole* ξ_n to be ∞ (a traditional Arnoldi step), so the last row of $\underline{K}_n$ is 0 and $A_n = H_n K_n^{-1}$.

Matlab examples

Using Rktoolbox by S. Güttel `http://guettel.com/rktoolbox/`.

```
>> rng(0); A = randn(100) + 10*eye(100);
>> v = eig(A); plot(real(v), imag(v), 'x');
>> b = randn(size(A,1), 1);
>> poles = [-20:-1, inf]; % inf as last pole
>> [V, K, H] = rat_krylov(A, b, poles);
>> An = H(1:end-1,:) / K(1:end-1,:);
>> v = eig(A); w = eig(An);
>> plot(real(v), imag(v), 'x', real(w), imag(w), 'o');
>> c = V(:, 1:end-1)*expm(An) * V(:, 1:end-1)'*b;
>> norm(expm(A)*b - c) / norm(c)
```

Try again with `poles = [21:40, inf]`, or `inf*ones(1,21)` (classical Arnoldi), `[0*ones(1,10), inf*ones(1,10)]` (extended Arnoldi), ...

The choice of poles directs which eigenvalues are best approximated and influences performance greatly.

Chapter 10

Lyapunov equations

Given $A, Q \in \mathbb{C}^{n\times n}$ with $Q = Q^* \succeq 0$, the Lyapunov equation is the matrix equation

$$A^*W + WA + Q = 0 \tag{10.1}$$

in the unknown $W \in \mathbb{C}^{n\times n}$.

It is a special case of the Sylvester equation: so we already know that there is a unique solution if and only if $\Lambda(A) \cup \Lambda(-A^*) = \emptyset$.

Important case: when $\Lambda(A) \subset LHP$ (open left half-plane).

Lemma 10.1. *Suppose* (L) *has a unique solution W; then W is Hermitian.*

Proof. Transpose everything; W^* is another solution. □

Lyapunov equation: positivity

Lemma 10.2. *Suppose $\Lambda(A) \subset LHP$ (open). Then, $Q \succeq 0$ implies $W \succeq 0$, and $Q \succ 0$ implies $W \succ 0$.*

Proof. We prove the result by giving the following closed formula for the solution:

$$W = \int_0^\infty e^{A^*t} Q e^{At}\, \mathsf{d}t.$$

Note that the integral converges: since $\Lambda(A) \subset LHP$, $\rho(\exp(A)) < 1$, and hence e^{At} decreases exponentially.

To prove this formula, compute $\frac{\mathsf{d}}{\mathsf{d}t} e^{A^*t} Q e^{At} = A^* e^{A^*t} Q e^{At} + e^{A^*t} Q e^{At} A$, then integrate both sides. □

Lemma
Suppose $Q \succ 0$ and $W \succ 0$. Then, $\Lambda(A) \subset LHP$.

Proof Let $Av = \lambda v$; then $0 < v^*Qv = \ldots$

Remark We cannot replace the $\succ$ symbols in this lemma with $\succeq$: $Q, W \succeq 0$ do *not* imply $\Lambda(A) \subset \overline{LHP}$. This is easy to see by considering an extreme case: obviously from $0 \cdot A + A \cdot 0 = 0$ we cannot deduce anything on A!

Relation to linear dynamical systems

Continuous-time linear dynamical system

$$\begin{cases} \dot{x}(t) = Ax(t), \quad x : [0, \infty] \to \mathbb{C}^n \\ x(0) = x_0. \end{cases}$$

We know that the solution to this ODE is $x(t) = \exp(At)x_0$. This system is *asymptotically stable*, i.e.,

$$\lim_{t \to \infty} x(t) = 0 \quad \text{for all choices of } x_0 \in \mathbb{C}^n,$$

iff $\Lambda(A) \subset LHP$.

In view of the lemmas above, it is sufficient to exhibit $W \succ 0$ such that $A^*W + WA \prec 0$ to prove that A has all its eigenvalues in the LHP.

At the time of Lyapunov (1857–1918), doing this (together with factorizations to show that $W, Q \succ 0$) was often easier than computing the full spectrum $\Lambda(A)$ (without a computer!).

Remark The matrix W that solves (L) is an *energy function* for the system: if $V(x) = x^*Wx$, then $\frac{d}{dt}V(x(t)) < 0$ (direct verification).

Discrete-time version

These results (and many of the following ones) also come in a discrete-time variant.

Discrete-time linear dynamical system

$$\begin{cases} x_0 \in \mathbb{C}^n \\ x_{k+1} = Ax_k, \quad k = 0, 1, 2, \ldots \end{cases}$$

The system is *asymptotically stable*, i.e., $\lim_{k \to \infty} x_k = 0$ for each x_0, *iff* A has its eigenvalues in the *open unit disc* $\mathbb{D}$.

Stein's equation

$$W - A^*WA = Q, \quad Q \succ 0 \tag{S}$$

If $W \succ 0$ solves (S), then $V(x) = x^*Wx$ is an *energy function*, i.e., $V(x_{k+1}) < V(x_k)$.

Lemma

$\Lambda(A) \subset \mathbb{D}$ *iff* (S) holds with $W, Q \succ 0$.

Proof Analogous to the continuous-time one. Closed formula:

$$W = \sum_{k=0}^{\infty} (A^*)^k Q A^k.$$

Proof Vectorizing, (S) becomes $(I - A^T \otimes A^*) \operatorname{vec}(W) = \operatorname{vec}(Q)$. Then use the Neumann series $(I - M)^{-1} = I + M + M^2 + \ldots$.

Remark (S) can be solved with a Bartels-Stewart-like method.

Remark More generally, Bartels-Stewart-type methods can be obtained for all equations of the form $AXB + CXD = E$, using QZ factorizations of (A, C) and (D^T, B^T).

Chapter 11

Introduction to control theory

11.1 Examples of control systems

Control theory [Datta, Ch. 5] is the study of dynamical systems with controllers; it is an important topic in engineering.

Example can we keep an 'inverted pendulum' of length 1 in the unstable upright position (12 o' clock) by applying a steering force?

We suppose that the pendulum is a massless stiff bar of length 1 with with weight at the end, so that there is only one degree of freedom, the angle θ that the bar makes with the vertical (12 o' clock $\leftrightarrow \theta = 0$).

The equation of motion is $\ddot{\theta} = g \sin\theta \approx g\theta$. We can rewrite it in terms of the *state* $x(t) = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$, to obtain the matrix version

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} x_2 \\ g x_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ g & 0 \end{bmatrix} x.$$

The system is not stable: $A = \begin{bmatrix} 0 & 1 \\ g & 0 \end{bmatrix}$ has one positive and one negative eigenvalue.

Example: controlling an inverted pendulum

Now we apply an additional steering force u (*control*): we have $\ddot{\theta} = g\theta + u$, or in matrix form

$$\dot{x} = Ax + Bu, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Can we choose $u(t)$ so that the system is stable? Yes: we can even choose one of the form $u(t) = Fx(t)$, $F \in \mathbb{R}^{1\times 2}$

We can literally build a contraption (engine + camera) that sets the appropriate force according to the current state only (*feedback control*). $u = \begin{bmatrix} f_1 & f_2 \end{bmatrix} x$ gives the *closed-loop system*

$$\dot{x} = (A + BF)x = \begin{bmatrix} 0 & 1 \\ f_1 + g & f_2 \end{bmatrix} x.$$

Choosing f_1, f_2, we can move the eigenvalues of $A + BF$ arbitrarily.

Remark: A (linear) 'controller' that observes only the position and not the velocity corresponds to $f_2 = 0$. It is easy to see that this is not enough to stabilize the system: if $f_2 = 0$, there is no choice of f_1 for which $\Lambda(A + BF) \subset LHP$.

Example: heating a long corridor with a window

Heat equation: in a bar of uniform material (the segment $[0, 1]$), one endpoint 1 is kept at constant temperature $0°\mathsf{C}$, and we apply a variable temperature (amount of 'heat') $u(t)$ at the other endpoint 0.

The temperature $x(y, t)$ at position y and time t follows

$$\frac{\partial}{\partial t} x(y,t) = \alpha \frac{\partial^2}{\partial y^2} x(y,t), \quad x(0,t) = u(t),\ x(1,t) = 0.$$

We discretize in space: $x(t)$ is a vector of temperatures at equi-spaced points $h, 2h, \ldots, (n-1)h$ (those at 0 and $(n+1)h = 1$ are prescribed).

$$\frac{d}{dt} x(t) = Ax(t) + Bu(t),$$

$A = \alpha h^2 \operatorname{tridiag}(1, -2, 1)$, $B = \alpha h^2 e_1$.

Other examples in [Datta, Ch. 5], e.g. electrical circuits.

Another impressive example of a control system is the triple pendulum on a cart; see e.g. the video `youtu.be/cyN-CRNrb3E`. This is a system with 3 degrees of freedom.

11.2 Controllability

$$\dot{x} = Ax + Bu, \quad A \in \mathbb{C}^{n \times n},\ B \in \mathbb{C}^{n \times m}.$$

Q1 Can we *stabilize* the system around 0, i.e., choose $u(t) = Fx(t)$ so that the system is asymptotically stable?

Q2 Can we *control* the system, i.e., choose $u(t)$ to reach a given value of $x(t_F)$?

Not always: counterexample:

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}. \tag{11.1}$$

No matter what $u(t)$ we choose, we cannot change the dynamics of the second block $x_2(t)$. If A_{22} has eigenvalues outside the LHP, the system will always be unstable.

We shall see that this is essentially the only case when a system is not controllable, but this structure may be hidden behind a change of basis for the state $A \leftarrow MAM^{-1}, B \leftarrow MB$, so it might be more difficult to identify. The key to analyze it is invariant subspaces: in the situation (11.1), the columns of B_1 belong to a nontrivial invariant subspace $\operatorname{Im}[\begin{smallmatrix} I \\ 0 \end{smallmatrix}]$ of A. The concept of invariant subspaces does not depend on the basis.

Hence, to identify a structure like (11.1) we can construct the smallest invariant subspace for A that contains the columns of B. We can give a formula for this subspace.

Lemma 11.1. *Let $A \in \mathbb{C}^{n\times n}, B \in \mathbb{C}^{n\times m}$. The smallest A-invariant subspace that contains the columns of B is*

$$K(A,B) := \operatorname{Im}[B, AB, A^2B, \dots].$$

Proof. It is simple to check that $K(A,B)$ is in fact invariant, and that every invariant subspace must contains the columns of $B, AB, A^2B, \dots$. □

Note the connection with Krylov subspaces: if B is a single vector, this is the union of all Krylov subspaces $K_n(A,B)$.

(In fact, this K does not stand for Krylov but for Kalman, another key figure in control theory.)

Definition 11.2. The space $K(A,B)$ is called the controllability space of (A,B). A matrix pair $(A,B) \in \mathbb{C}^{n\times n} \times \mathbb{C}^{n\times m}$ is called controllable when $K(A,B) = \mathbb{C}^n$.

Properties

- Controllability depends only on $\operatorname{Im} B$, hence (A,B) controllable $\iff$ (A,BK) controllable, for any invertible K.
- Similarly, (A,B) controllable $\iff$ $(A, BR^{-1}B^*)$ controllable for any invertible $R \in \mathbb{C}^{m\times m}$; we will use this property in future.
- (A,B) controllable $\iff$ $(A-\alpha I, B)$ controllable, since the powers of $A-\alpha I$ are linear combinations of the powers of A.

Controllability [Datta, Ch. 6, with more streamlined proofs]

We shall show that indeed the controllability space reveals the structure we were interested in.

Lemma 11.3 (Kalman decomposition)**.** *For each pair (A, B), there exists a nonsingular $M \in \mathbb{C}^{n\times n}$ such that the following block decomposition holds, and (A_{11}, B_1) is controllable.*

$$M^{-1}AM = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad M^{-1}B = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}.$$

The blocks must be of the same size, i.e., $B_1 \in \mathbb{C}^{n_1\times m}$ if $A_{11} \in \mathbb{C}^{n_1\times n_1}$.

Moreover, $n_1 = \dim K(A, B)$, and in particular we have $n_1 = n$ if and only if (A, B) is controllable.

Proof. It is sufficient to take $M = \begin{bmatrix} M_1 & M_2 \end{bmatrix}$ such that M_1 is a basis of $K(A, B)$; then, the blocks B_2 and A_{21} must be zero, because $K(A, B)$ contains the columns of B and is A-invariant.

If (A_{11}, B_1) were not controllable, then $K(A_{11}, B_{11})$ (extended with zeros) would be a smaller invariant subspace of $M^{-1}AM$ that contains the columns of $M^{-1}B$, contradicting minimality. □

We now need to prove that this concept that we dubbed controllability of a matrix pair is indeed related to the controllability of the dynamical system $\dot{x} = Ax + Bu$.

Theorem 11.4. *The following are equivalent.*

1. *The system $\dot{x} = Ax + Bu$, $x(0) = x_0$ is controllable, i.e., given any target state x_F and time t_F we can choose a control function $u(t)$ such that $x(t_F) = x_F$.*
2. *The pair (A, B) is controllable.*
3. *The matrix*
$$W_t = \int_0^t \exp(A\tau)BB^* \exp(A^*\tau)d\tau$$
is invertible (for a specific $t > 0$, or, equivalently, for all of them).

Before starting the proof, we remark that the expression of W_t resembles the integral formula for the solution of the Lyapunov equation that we have proved earlier. Indeed, we see that $W = \lim_{t\to\infty} W_t$. In particular, this equivalence shows that $W \succ 0$ if and only if (A, B) is controllable.

Proof. $1 \implies 2$ Suppose, by contradiction, that $K(A, B)$ is not the whole space.

Recall the ugly closed formula for the solution of a linear differential equation $\dot{x}(t) = Ax(t) + f(t)$, where in our case $f(t) = Bu(t)$. We have

$$x(t) = \exp(At)x_0 + \int_0^t \exp(A(t-\tau))Bu(t)dt. \tag{11.2}$$

Since $\exp(A(t-\tau))$ is a matrix function and hence a polynomial in A, one sees that the integral always takes values in $K(A,B)$. Hence, independently of $u(t)$, we cannot obtain all possible values of $x(t)$; it is sufficient to take x_F such that

$$x_F - \exp(At_F)x_0 \notin K(A,B)$$

to get a vector that cannot be reached.

2 $\implies$ 3 Suppose $W_t v = 0$ for a certain $t > 0$. Then, $0 = v^* W_t v = \int_0^t \|v^* \exp(A\tau)B\|^2 dt$. This must mean that the (continuous) function $\phi(\tau) = v^* \exp(A\tau)B = 0$. Hence, in particular.

$$\begin{aligned} 0 = \phi(0) &= v^* B, \\ 0 = \phi'(0) &= v^* AB, \\ 0 = \phi''(0) &= v^* A^2 B, \\ \vdots \quad &\quad \vdots \end{aligned}$$

and this shows that $v^*[B, AB, A^2B, \dots] = 0$, so the controllable space is not the whole $\mathbb{C}^n$. 3 $\implies$ 1 Take a control $u(t)$ of the form

$$u(t) = B^* \exp(A^*(t_F - t))y,$$

with $y \in \mathbb{C}^n$. Plugging it into (11.2), and recognizing the matrix W_t inside the expression (up to a change of variables $\tau = t_F - t$), we obtain

$$x(t_F) = \exp(At_F)x_0 + W_{t_F} y.$$

Since W_{t_F} is invertible, with a suitable choice of y we can obtain any value of $x(t_F) \in \mathbb{C}^n$. □

Other controllability criteria

Popov (or Hautus) criterion
(A,B) controllable $\iff$ $\operatorname{rank}[A - \lambda I,\ B] = n$ for all $\lambda \in \Lambda(A)$ $\iff$ $\operatorname{rank}[A - \lambda I, B] = n$ for all $\lambda \in \mathbb{C}$.

It is sufficient to test the condition on $\lambda \in \Lambda(A)$, because for all other λs we already have $\operatorname{rank}(A - \lambda I) = n$.

Proof

$\Leftarrow$ We can assume (up to a change of basis) that (A,B) is in a Kalman decomposition, with a non-trivial block A_{22}. Take a left eigenpair $v^* A_{22} = \lambda v^*$: then, $[0,\ v^*][A - \lambda I,\ B] = 0$.

$\Rightarrow$ If $v^*[A - \lambda I, B] = 0$ for some $\lambda \in \Lambda(A)$, then we get $0 = v^*B = v^*AB = v^*A^2B = \dots$, and hence $K(A,B) \neq \mathbb{C}^n$ as in the proof of the previous theorem.

How to test controllability numerically?

Numerically, *almost any* (A, B) is controllable: things are rarely zero. Anyway, various options:

- Compute $\text{rank}[B, AB, A^2B, \ldots, A^{n-1}B]$. We can stop at $n-1$, because A^n is a linear combination of $I, A, A^2, \ldots, A^{n-1}$ by the Cayley-Hamilton theorem.
- If B is a single vector, you can also run a Krylov algorithm until the final iteration, and check for breakdown.
- Compute $\Lambda(A)$ and check that $\text{rank}[A - zI, B] = n$ for each $z \in \Lambda(A)$.
- Assume (up to replacing it with $A - \alpha I$) that $\Lambda(A) \subset LHP$. Solve the Lyapunov equation $AW + WA^* + BB^* = 0$ and check if $W \succ 0$.

All these methods rely on a *rank decision*: are certain computed values positive?

Remark There are methods to compute the distance of a certain matrix pair (A, B) to the nearest uncontrollable pair, exactly like the condition number of a matrix M is a measure of the distance of M to the nearest singular matrix. They are somewhat more complex, and research is still active on the best one.

Remark The criterion with the Lyapunov equation actually corresponds to a physical quantity: $x_0^* W^{-1} x_0$ is the minimal amount of *energy* $\int_0^{t_F} u(\tau)^* u(\tau) d\tau$ that we need to reach $x(t_F) = 0$ starting from $x(0) = x_0$. We won't prove it here. Hence, the closer to uncontrollable a system is, the more energy you need to put in to actually control it.

(Matlab examples: construct a non-controllable (A, B) from a Kalman decomposition, and apply the various methods.)

11.3 Stabilizability

If a system is controllable, then it is also stabilizable: we can find F such that $\Lambda(A + BF) \subset LHP$. The following result gives us a practical way to compute one such choice of F.

Theorem (Bass algorithm)
Let (A, B) be controllable, $\alpha > \rho(A)$, and W the solution of

$$(-A - \alpha I)W + W(-A - \alpha I)^* + 2BB^* = 0. \qquad \text{(L')}$$

Then, $W \succ 0$ and $F = -B^* W^{-1}$ is a stabilizing feedback.

Note that $(-A - \alpha I, B)$ is controllable because (A, B) is so, that $\Lambda(-A - \alpha I) \subset LHP$, and that $Q := 2BB^* \succeq 0$. By Lyapunov eq. results, this implies $W \succ 0$. Rearranging (L') gives

$$(A + BF)W + W(A + BF)^* + 2\alpha W = 0.$$

By Lyapunov eq. results, $W \succ 0$, $Q := 2\alpha W \succ 0$ implies $\Lambda(A + BF) \subset LHP$.

Remark We can actually find F such that $A+BF$ has any chosen spectrum. (We won't prove it here.) [Datta, Ch. 11]

Stabilizability

Sometimes, even if a system is not controllable, we can still ensure that the solution converges to 0. Example: take a system already in Kalman decomposition

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}, \quad x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

with $\Lambda(A_{22}) \subset LHP$. Then, the control does not act on $x_2(t)$, but $x_2(t) \to 0$ already by itself!

To stabilize this system with a feedback control, just take $F = [F_1\, 0]$, where F_1 is chosen so that $\Lambda(A_{11} + B_1 F_1) \subset LHP$ (it exists because (A_{11}, B_1) is controllable by definition of Kalman decomposition).

Stabilizability conditions

Theorem

The following conditions are equivalent; if they hold, (A, B) is called *stabilizable*.

1. $\Lambda(A_{22}) \subset LHP$ in the Kalman decomposition;
2. $\mathrm{rk}[A - \alpha I, B] = n$ for all $\alpha \notin LHP$;
3. We can find $u(t)$ such that $\lim_{t\to\infty} x(t) = 0$;
4. We can find F such that $\Lambda(A + BF) \subset LHP$ (hence we can take $u(t) = Fx(t)$ to satisfy the previous point).

We won't see a full proof, but it mostly follows from things we have already stated.

Matlab example: inverted pendulum

```
% (dubious example because the true sys is nonlinear)
A = [0 1; 1 0]; B = [0;1]; x0 = [0.1; -0.05];

% open-loop system
[t, x] = ode45(@(t,x) A*x, [0,5], x0);
plot(t,x);

% feedback that observes the position and tries to push it back
F = [-1.5 0];
[t, x] = ode45(@(t,x) (A+B*F)*x, [0,5], x0);
plot(t,x);

% random feedback (try several)
F = randn(1,2);
[t, x] = ode45(@(t,x) (A+B*F)*x, [0,5], x0);
plot(t,x);
```

Matlab example: heat equation

```
% heat equation on a steel bar
n = 10; y = linspace(h,n*h,n);
h = 1/(n+1); x0 = rand(n, 1);

A = h^2*(-2*eye(n) + diag(ones(n-1,1),1) ...
    + diag(ones(n-1,1),-1));
B = h^2*eye(n,1);

% open-loop system
[t, x] = ode45(@(t,x) A*x, [0,1000], x0);
surf(y, t, x);

% constant stream of heating
[t, x] = ode45(@(t,x) A*x + B*1, [0,1000], x0);
surf(y, t, x);
```

```
% feedback
F = rand(1,n);
[t, x] = ode45(@(t,x) A*x + B*F*x, [0,1000], x0);
surf(y, t, x);

% thermal sensor midway along the bar
F = zeros(1,n); F(end/2) = -1000*h^2;
[t, x] = ode45(@(t,x) A*x + B*F*x, [0,1000], x0);
surf(y, t, x);
```

```
% thermal sensor with wrong sign
F = zeros(1,n); F(end/2) = 1000*h^2;
[t, x] = ode45(@(t,x) A*x + B*F*x, [0,1000], x0);
surf(y, t, x);
```

```
% controlling to a specified position
tf = 1000; xf = rand(n,1);

Winf = lyap(A, B*B');
W = integral(@(t) expm(A*(tf-t))*B*B'*expm(A'*(tf-t)), ...
    0, tf, 'ArrayValued', true);
eig(Winf), eig(W) % system barely controllable

y = W \ (xf - expm(A*tf)*x0);
[t, x] = ode45(@(t,x) A*x + B*B'*expm(A'*(tf-t))*y, ...
    [0,1000], x0);
surf(y, t, x); % hard-to-control system

[x(end,:); xf'] % not too accurate!

[t, x] = ode45(@(t,x) A*x + B*B'*expm(A'*(tf-t))*y, ...
   [0,1000], x0, odeset('RelTol', 1e-8, 'AbsTol', 1e-10));
[x(end,:); xf'] % but it was just an ode45 accuracy issue
```

```
% Bass's algorithm
alpha = 1.1*max(abs(eig(A)))
W = lyap(-A-alpha*eye(n), 2*B*B')
F = -B'/W; eig(A+B*F) %all in LHP!

[t, x] = ode45(@(t,x) A*x + B*F*x, [0,1000], x0);
surf(y, t, x);
```

Chapter 12

Optimal control

Optimal control

Several choices available for stabilizing feedback F: for instance, you can choose different α's in Bass algorithm.

Is there an 'optimal' one? One possible way to formalize this: the control that uses the minimum *energy*, defined by a quadratic form ($R \succeq 0$, $Q \succeq 0$).

Linear-quadratic optimal control

Find $u : [0, \infty) \to \mathbb{C}^m$ (piecewise C^0, let's say) that minimizes

$$V(u) = \int_0^\infty x^* Q x + u^* R u \, \mathsf{d}t$$
$$\text{s.t. } \dot{x} = Ax + Bu,\ x(0) = x_0,\ \lim_{t\to\infty} x(t) = 0.$$

We assume here that $R \succ 0$: control is never free. Optimal control becomes a trickier problem otherwise.

Linear-quadratic regulator theorem [Datta, Thm 10.5.1]

A solution follows from calculus of variations principles; here is a self-contained version.

Theorem

Let $Q \succeq 0$, $R \succ 0$, (A, B) controllable. Set $G = BR^{-1}B^* \succeq 0$.

There exists a unique $X = X^* \in \mathbb{C}^{n\times n}$ such that

1. $A^*X + XA + Q - XGX = 0$,
2. $\Lambda(A - GX) \subset LHP$.

The optimal value of the minimum problem

$$\min \int_0^\infty x(t)^* Q x(t) + u(t)^* R u(t) \, \mathsf{d}t,$$
$$\text{s.t. } \dot{x}(t) = Ax(t) + Bu(t), \quad \lim_{t\to\infty} x(t) = 0$$

is $x_0^* X x_0$, attained with the feedback control $u(t) = Fx(t)$ obtained with $F = -R^{-1}B^*X$.

Note that indeed $A + BF = A - GX$ is stable by the conditions we imposed on X. The equation

$$A^*X + XA + Q - XGX = 0$$

is called continuous-time algebraic Riccati equation, and X that satisfies 1-2 is called its stabilizing solution.

Proof. Proving the existence of X with those properties will be long, and it is the topic of the rest of this chapter. We shall now conclude assuming it exists.

Note that $\Lambda(A - GX) \subset LHP$ implies $\lim_{t\to\infty} x(t) = 0$, so this u is admissible. Take a generic stabilizing control u, and compute

$$\begin{aligned}
\frac{\mathsf{d}}{\mathsf{d}t} x^*Xx &= \dot{x}^*Xx + x^*X\dot{x} \\
&= (Ax + Bu)^*Xx + x^*X(Ax + Bu) \\
&= x^*(A^*X + XA)x + u^*B^*Xx + x^*XBu \\
&= x^*(XBR^{-1}B^*X - Q)x + u^*B^*Xx + x^*XBu \\
&= \underbrace{(u + R^{-1}B^*Xx)^*R(u + R^{-1}B^*Xx)}_{\geq 0} - x^*Qx - u^*Ru.
\end{aligned}$$

Integrating from 0 to ∞,

$$\int_0^\infty x^*Qx + u^*Ru \,\mathsf{d}t \geq x_0^*Xx_0 - \underbrace{x(\infty)^*Xx(\infty)}_{=0},$$

with equality if $u + R^{-1}B^*Xx \equiv 0$. □

Riccati equation and subspaces

The equation

$$A^*X + XA + Q - XGX = 0, \quad Q \succeq 0, G \succeq 0$$

is called *algebraic Riccati equation* (ARE). It is an *invariant subspace problem* in disguise, because if X satisfies the equation then

$$\begin{bmatrix} A & -G \\ -Q & -A^* \end{bmatrix} \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} (A - GX).$$

Hence the problem of finding X can be recast as finding a suitable invariant subspace of $\mathcal{H}$. The road to proving the existence of our solution X passes through studying the properties of the matrix $\mathcal{H}$.

Hamiltonian matrices

A matrix of the form

$$\mathcal{H} = \begin{bmatrix} A & -G \\ -Q & -A^* \end{bmatrix}, \quad Q = Q^*,\, G = G^*$$

is called Hamiltonian matrix.

Lemma 12.1. *Let $\mathcal{H}$ be Hamiltonian, and $\lambda \in \Lambda(\mathcal{H})$. Then, $-\overline{\lambda} \in \Lambda(\mathcal{H})$, too, and the two have the same multiplicity.*

It is easy to see that $-\overline{\lambda}$ is the symmetric of λ with respect to the imaginary axis.

Proof. Let $J = \begin{bmatrix} & I \\ -I & \end{bmatrix}$. One can verify directly the equality $J^{-1}\mathcal{H}J = -\mathcal{H}^*$. Hence, $\mathcal{H}$ and $-\mathcal{H}^*$ are similar, and they have the same spectrum (also counted with multiplicities). □

We can say more.

Theorem 12.2. *Assume $Q \succeq 0$, $G = BR^{-1}B^* \succeq 0$, (A, B) (or, equivalently, (A, G)) stabilizable, and (A^*, Q) stabilizable. Then, $\mathcal{H}$ has no eigenvalues with* $\operatorname{Re}\lambda = 0$.

Proof. Suppose instead $\mathcal{H}\left[\begin{smallmatrix} z_1 \\ z_2 \end{smallmatrix}\right] = \imath\omega\left[\begin{smallmatrix} z_1 \\ z_2 \end{smallmatrix}\right]$. Writing the blocks out explicitly, we get

$$\begin{aligned} Az_1 - Gz_2 &= \imath\omega z_1, \\ -Qz_1 - A^*z_2 &= \imath\omega z_2. \end{aligned}$$

We can eliminate $A - \imath\omega$ from these equations: multiply the first equation by z_2^*, transpose the second, and multiply it by z_1. Then we are left with

$$z_1^*Qz_1 + z_2^*Gz_2 = 0.$$

Since Q and G are positive semidefinite, it must be the case that $Qz_1 = Gz_2 = 0$. Substituting it above, we get $(A - \imath\omega I)z_1 = 0$, $(A - \imath\omega)^*z_2 = 0$. We then obtain

$$z_1^*[A^* + \imath\omega \quad Q] = 0, \quad z_2^*[A - \imath\omega \quad G] = 0.$$

Since at least one of z_1 and z_2 is nonzero, we contradict one of the two stabilizability conditions (Popov test). □

Hence, $\mathcal{H}$ has n eigenvalues in the LHP and n in the RHP, counted with multiplicity. In particular, it has a (unique) n-dimensional invariant subspace associated to its eigenvalues in the LHP, i.e., there is $U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \in \mathbb{C}^{2n\times n}$ such that

$$\begin{bmatrix} A & -G \\ -Q & -A^* \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \mathcal{S}, \quad \Lambda(\mathcal{S}) \subset LHP. \tag{12.1}$$

We call this invariant subspace the stable invariant subspace. We can prove a particular property.

Lemma 12.3. *Let $\mathcal{H}$ be Hamiltonian, and $\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$ be a basis matrix for its stable invariant subspace. Then,*

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^* J \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} = U_2^* U_1 - U_1^* U_2 = 0.$$

Proof. Consider a Jordan basis for $\mathcal{H}$, partitioned into a set of Jordan chains in the LHP and one in the RHP.

$$\mathcal{H} = V \begin{bmatrix} J_{LHP} & 0 \\ 0 & J_{RHP} \end{bmatrix} V^{-1}. \tag{12.2}$$

Note that the first n columns of V are a basis for the stable invariant subspace of $\mathcal{H}$. By transposing and negating (12.2), one sees that the last n rows of V^{-1} are a basis for the stable invariant subspace of $-\mathcal{H}^*$. In particular, these two subspaces must be orthogonal, because $V^{-1}V = I$.

We know that U is a basis for the stable invariant subspace of $\mathcal{H}$, and, thanks to the relation $J^{-1}\mathcal{H}J = -\mathcal{H}^*$, we see that JU is a basis for the stable invariant subspace of $\mathcal{H}^*$. Hence, in particular, U and JU are orthogonal. □

A subspace such that U and JU is orthogonal is called Lagrangian subspace.

Existence of X

We are now very close to proving the existence of X. If we can prove that U_1 is invertible, then we can take a different basis

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix} U_1^{-1} = \begin{bmatrix} I \\ U_2 U_1^{-1} \end{bmatrix}$$

for that invariant subspace, and get (with $X = U_2 U_1^{-1}$)

$$\begin{bmatrix} A & -G \\ -Q & -A^* \end{bmatrix} \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} \widehat{\mathcal{S}}, \quad \widehat{\mathcal{S}} = U_1 \mathcal{S} U_1^{-1}. \tag{12.3}$$

Expanding out the blocks we get

$$-Q - A^* X = X\widehat{\mathcal{S}} = X(A - GX),$$

which is the Riccati equation, and $\Lambda(A - GX) = \Lambda(\widehat{\mathcal{S}}) \subset LHP$.

Theorem 12.4. *Suppose (A, B) and (A^*, Q) stabilizable, $Q \succeq 0$, $G \succeq 0$. Then, U_1 is invertible.*

Proof. The key is proving that $\ker U_1$ is an invariant subspace for $\mathcal{S}$. Let $v \in \ker U_1$,

$$-v^* U_2^* G U_2 v = \begin{bmatrix} v^* U_2^* & 0 \end{bmatrix} \mathcal{H} \begin{bmatrix} 0 \\ U_2 v \end{bmatrix} = v^* \underbrace{\begin{bmatrix} U_2^* & -U_1^* \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}}_{=0} \mathcal{S} v = 0$$

implies $GU_2v = 0$. Then looking at the first block row of

$$\begin{bmatrix} A & -G \\ -Q & -A^* \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} v = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \mathcal{S}v$$

we get $U_1\mathcal{S}v = 0$ as needed.

If $\ker U_1$ is nontrivial, we can find v, $\lambda \in LHP$ such that $U_1v = 0$, $\mathcal{R}v = \lambda v$. Now the second block row gives $-A^*U_2v = \lambda U_2v$. This (together with $GU_2v = 0$ from above) contradicts stabilizability. □

Symmetry of the solution

$$X^* - X = U_1^{-*}U_2^* - U_2U_1^{-1} = U_1^{-*}(U_2^*U_1 - U_1^*U_2)U_1^{-1} = 0.$$

Positive definiteness of the solution

Note that

$$ARE \iff (A - GX)^*X + X(A - GX) + Q + XGX = 0.$$

So X solves the Lyapunov equation

$$\hat{A}^*X + X\hat{A} + \hat{Q} = 0, \quad \hat{A} = A - GX,\ \hat{Q} = Q + XGX.$$

And we know that $\Lambda(\hat{A}) \subset LHP, \hat{Q} \succeq 0 \implies X \succeq 0$.

Under slightly stronger assumptions one can also show that $(\hat{A}^*, \hat{Q})$ controllable $\implies X \succ 0$.

Factorization

Once we know X exists, we can write the factorization

$$\begin{bmatrix} I & 0 \\ -X & I \end{bmatrix} \mathcal{H} \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} = \begin{bmatrix} A - GX & -G \\ 0 & -(A - GX)^* \end{bmatrix},$$

which displays clearly the eigenvalue pairing $\lambda, -\overline{\lambda}$.

How to solve Riccati equations

- Newton's method (historically the first option).
- Invariant subspace computation: via unstructured methods (QR), 'semi-structured' methods (Laub trick), or fully structured methods (URV).
- Sign iteration (and variants).

Chapter 13

Newton's method for AREs

Newton's method for CARE

Each iterate of Newton's method is a Lyapunov e)quation:

$$F(X) = A^*X + XA + Q - XGX$$

$$L_{F,X}(H) = A^*H + HA - HGX - XGH = H(A - GX) + (A - GX)^*H.$$

$$\widehat{L}_{F,X} = (A - GX)^T \otimes I + I \otimes (A - GX)^*.$$

If X_* is the stabilizing solution then $\Lambda(A - GX_*) \subset LHP \implies L_{F,X_*}$ is nonsingular.

Newton's method

For $k = 0, 1, 2, \ldots$

1. Solve $H(A - GX_k) + (A - GX_k)^*H = F(X_k)$ for H;
2. Set $X_{k+1} = X_k - H$.

Newton's method

Note that $H(A - GX_k) + (A - GX_k)^*H = F(X_k)$ is equivalent to

$$X_{k+1}(A - GX_k) + (A - GX_k)^*X_{k+1} = -Q - X_kGX_k \preceq 0.$$

If $\Lambda(A - GX_k) \subset LHP$, then $X_{k+1} \succeq 0$, by the results on Lyapunov equations.

Actually, something stronger holds.

Theorem

Suppose X_0 is chosen such that $\Lambda(A - GX_0) \subset LHP$. Then, $X_1 \succeq X_2 \succeq X_3 \succeq \cdots \succeq X_* \succeq 0$. Moreover, $X_k \to X_*$ quadratically.

Remark The thesis does not include $X_0 \succeq X_1$: anything could happen in the first iteration!

Monotonicity of Newton's method

Proof (sketch) Coupled induction. Set $A_k := A - GX_k$. Some algebra gives

$$(X_k - X_{k+1})A_k + A_k^*(X_k - X_{k+1}) = -(X_k - X_{k-1})G(X_k - X_{k-1})$$
$$(X_* - X_{k+1})A_k + A_k^*(X_* - X_{k+1}) = -(X_* - X_k)G(X_* - X_k)$$

hence A_k stable $\implies X_k \succeq X_{k+1} \succeq X_*$.

$$(X_{k+1} - X_*)A_{k+1} + A_{k+1}^*(X_{k+1} - X_*)$$
$$= -(X_{k+1} - X_k)G(X_{k+1} - X_k) - (X_{k+1} - X_*)G(X_{k+1} - X_*)$$

This does not prove immediately that A_{k+1} is stable (because the RHS is not $\prec 0$), but plugging in $A_{k+1}v = \lambda v$ with $\operatorname{Re}\lambda \geq 0$ we get $B(X_{k+1} - X_k)v = 0$, hence also $A_k v = \lambda v$.

Matlab experiments

Newton: wrap-up

Algorithm

- Use Bass's algorithm to find X_0 such that $A - GX_0$ is stable
- Run Newton iterations until convergence.

Expensive: each iteration requires a Schur form.

Convergence: standard quadratic convergence of Newton's method holds: if the solution is simple (which is the case whenever the Hamiltonian has no imaginary eigenvalues $\iff$ $L_{F,X}$ is invertible), then $\|X_* - X_{k+1}\| \sim \|X_* - X_k\|^2$.

Defect correction / iterative refinement: One final step of Newton can be used to *improve the accuracy* of a computed solution from another algorithm.

Chapter 14

Invariant subspace methods for CAREs

Invariant subspace methods for CAREs

X solves CARE $A^*X + XA + Q = XGX$ iff

$$\begin{bmatrix} A & -G \\ -Q & -A^* \end{bmatrix} \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} \mathcal{R}, \quad \mathcal{R} = A - GX.$$

One can find X through an invariant subspace of the Hamiltonian.

```
>> [A,G,Q] = carex(4) %if test suite is installed
>> n = length(A);
>> H = [A -G; -Q -A'];
>> [U, T] = schur(H);
>> [U, T] =ordschur(U, T, 'lhp');
>> X = U(n+1:2*n, 1:n) / U(1:n, 1:n);
```

Recall: backward stability

QR-like algorithms based on successive orthogonal transformations are *backward stable*: at each step instead of $\mathcal{H}_{i+1} = Q_i^* \mathcal{H} Q_i$ one computes $\tilde{H}_i = Q_i^* \mathcal{H} Q_i + F_i$, with forward error $\|F_i\|/\|\mathcal{H}_i\| = O(\mathtt{u})$. This is mapped back to a *backward error* $\Delta\mathcal{H}^{(i)} = Q_1^* Q_2^* \dots Q_i^* F_i Q_i \dots Q_2 Q_1$ with the same norm.

In particular, the Schur method computes a true invariant subspace of $\mathcal{H} + \Delta\mathcal{H}$, with $\|\Delta\mathcal{H}\|$ small.

However, this method is not *structured* backward stable: the error $\Delta\mathcal{H}$ is not Hamiltonian.

Among the consequences, eigenvalues close to the imaginary axis can be 'mixed up'. Try `carex(14)` for instance: the Schur method produces an invariant subspace $\mathcal{U}$ that does *not* give a symmetric X, because it is the wrong invariant subspace.

To improve accuracy on ill-conditioned problem, it would be ideal to have a *structurally backward stable* method.

Indefinite scalar products and symplectic transformations

Ultimately, the reason why the previous algorithm fails to preserve the eigenvalue pairing is that orthogonal transformations do not preserve the matrix structure of the Hamiltonian.

This structure is intimately related with indefinite scalar products. Let us consider the indefinite scalar product (bilinear form) defined by the matrix J, i.e.,

$$\langle u, v\rangle = u^* J v = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}^* \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = u_1^* v_2 - u_2^* v_1.$$

The matrix $\mathcal{H}$ is skew-self-adjoint with respect to this scalar product, i.e., $\langle u, \mathcal{H}v\rangle = \langle -\mathcal{H}^* u, v\rangle$; indeed, this is equivalent to Lemma TODO. Indeed, any matrix H with $H_{11} = -H_{22}^*$, $H_{21} = H_{21}^*$, $H_{12} = H_{12}^*$ is so, even without the positive semidefiniteness constraint. These are called Hamiltonian matrices, and they all satisfy the eigenvalue pairing lemma.

So we must look for orthogonal transformations with respect to this scalar product.

Definition 14.1. A matrix $S \in \mathbb{C}^{2n\times 2n}$ is called *symplectic* if it is orthogonal w.r.t the scalar product J, that is, if $S^* J S = J$.

Lemma

If $\mathcal{H}$ is Hamiltonian and S is symplectic, then $S^{-1}\mathcal{H}S$ is Hamiltonian.

Proof: $(S^{-1}\mathcal{H}S)^* J = J(S^{-1}\mathcal{H}S) \iff (S^{-1}\mathcal{H}S)^* S^* J S = S^* J S (S^{-1}\mathcal{H}S) \iff S^*\mathcal{H}^* J S = S^* J \mathcal{H} S$.

Remark: unlike orthogonal transformations, symplectic ones do not automatically ensure stability: $\|v\|$ small does not imply $\|Sv\|$ small: for instance, any matrix of the form $S = \left[\begin{smallmatrix} A & 0 \\ 0 & A^{-T} \end{smallmatrix}\right]$ is symplectic.

Orthosymplectic transformations

Ideal setting: construct successive changes of bases $\mathcal{H} \mapsto S^{-1}\mathcal{H}S$ where S is *both* orthogonal (for stability reasons) and symplectic (for structure preservation reasons). These are called orthosymplectic matrices.

Examples of orthosymplectic matrices:

- If $Q \in \mathbb{C}^{n\times n}$ is any orthogonal matrix, then $\mathrm{blkdiag}(Q, Q)$ is orthosymplectic.
- A Givens matrix that acts on entries k and $n+k$ (i.e., that generated with the Matlab commands

  ```
  G = eye(2*n); G([k,n+k], [k,n+k]) = [c s; -s c];
  ```

  is orthosymplectic.

The Laub trick

There is a certain *orthogonal and symplectic* matrix that reduces $\mathcal{H}$ to a special form.

Theorem

Let $U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}$ be unitary s.t. $\begin{bmatrix} U_{11} \\ U_{21} \end{bmatrix}$ spans the stable invariant subspace. Then,

1. $V = \begin{bmatrix} U_{11} & -U_{21} \\ U_{21} & U_{11} \end{bmatrix}$ is orthosymplectic;
2. $V^*\mathcal{H}V = \begin{bmatrix} T_{11} & T_{12} \\ 0 & -T_{11}^* \end{bmatrix}$, with T_{11} upper triangular and T_{12} symmetric (*Hamiltonian Schur form*).

Proof (1) follows from the fact that $\left[\begin{smallmatrix} U_{11} \\ U_{21} \end{smallmatrix}\right]$ has orthonormal columns, and we showed earlier that $U_{21}^*U_{11} - U_{11}^*U_{21} = 0$.

(2) follows from the facts that $\left[\begin{smallmatrix} U_{11} \\ U_{21} \end{smallmatrix}\right]$ spans an invariant subspace and that $V^*\mathcal{H}V$ is Hamiltonian.

An orthogonal symplectic algorithm

Numerically, the Laub trick is no more effective than the Schur method, because they compute the same invariant subspace.

But the existence of this structured factorization suggests that there may be a structure-preserving method to compute it.

Problem ("curse of Van Loan")

Is there a *structure-preserving QR method* that produces the Hamiltonian Schur form via a sequence of orthosymplectic transformations applied to $\mathcal{H}$?

Roadblock: we have proved that a stable invariant subspace exists if (A, B) controllable and $G, Q \succeq 0$, but there are Hamiltonian matrices that do not satisfy these assumptions; e.g., $\mathcal{H} = \left[\begin{smallmatrix} 1 & 2 \\ -1 & -1 \end{smallmatrix}\right]$ with eigenvalues $\pm i$.

$\implies$ algorithms to compute a HSF must become unstable when G, Q are ill-conditioned.

Extras: Chu–Liu–Mehrmann algorithm [Chu-Liu-Mehrmann '98]

A solution comes by going through another, different decomposition: $\mathcal{H} = URV^*$, with U, V orthosymplectic and

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$$

with $R_{11}R_{22}^*$ upper triangular.

(Reminds of the SVD.)

It can be computed in $O(n^3)$ via backward stable orthosymplectic transformations.

Note that R is *not* Hamiltonian and $\Lambda(\mathcal{H}) \neq \Lambda(R)$, in general.

URV decomposition — sketch

- Left-multiply by blkdiag(Q, Q) to get $\begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \end{bmatrix}$
- Left-multiply by a Givens on $(1, n+1)$ to get $\begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \end{bmatrix}$
- Right-multiply by blkdiag(Q, Q) to get $\begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & 0 & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \end{bmatrix}$
- Right-multiply by a Givens on $(2, n+2)$ to get $\begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & * & * & * & * & * \\ 0 & * & * & * & * & * \end{bmatrix}$
- Repeat on smaller blocks to zero out the $(2,1)$ block: $\begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & * & * & * & * \end{bmatrix}$, $\begin{bmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * & * \end{bmatrix}$.

URV — the final step

Finally, left multiply by blkdiag(Q, Q) to replace R_1, R_2 with QR_1, QR_2^* so that $QR_1R_2^*Q^*$ is upper triangular.

Note that $\mathcal{H} = URV$ together with symplecticity implies

$$\mathcal{H} = V \begin{bmatrix} -R_{22}^* & R_{12}^* \\ 0 & -R_{11}^* \end{bmatrix} U^*.$$

Then

$$\mathcal{H}^2 = V \begin{bmatrix} -R_{11}R_{22}^* & * \\ 0 & -R_{22}R_{11}^* \end{bmatrix} V^*.$$

This is a Schur-like decomposition that reveals the eigenvalues and eigenvectors of $\mathcal{H}^2$ (not of $\mathcal{H}$). However, if v_1 is an eigenvector of $\mathcal{H}^2$ then $\operatorname{span}(v_1, \mathcal{H}v_1)$ is an invariant subspace of $\mathcal{H}$, and it can be used to compute an eigenvector of $\mathcal{H}$ and deflate it (many details omitted).

Chapter 15

The sign function method for CAREs

Sign-like methods for CAREs

Matrix sign iteration

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \quad X_0 = \mathcal{H}.$$

It is not difficult to see that X_k is Hamiltonian at each step (i.e., $JX_k = -X_k^* J$).

Lemma

- Let M be Hamiltonian. Then M^{-1} is Hamiltonian, too.
- Let M_1, M_2 be Hamiltonian. Then $M_1 + M_2$ is Hamiltonian, too.

(Guiding idea: Hamiltonian matrices are 'like antisymmetric ones': properties that you expect for antisymmetric matrices will often hold for Hamiltonian, too.)

Structure-preserving sign iteration

In machine arithmetic, the X_k won't be exactly Hamiltonian — unless we modify our algorithm to ensure that they are.

Observation: $\mathcal{H}$ is Hamiltonian iff $J\mathcal{H}$ is symmetric.

Rewrite the iteration in terms of $Z_k := JX_k$:

$$Z_{k+1} = \frac{1}{2}(Z_k + JZ_k^{-1}J), \quad Z_0 = J\mathcal{H}.$$

This version preserves symmetry exactly (assuming the method we use for inversion does).

We can incorporate scaling.

Towards doubling

Recall: in the sign iteration, if we set $Y_k = (I - X_k)^{-1}(I + X_k)$, then $Y_{k+1} = -Y_k^2$.

In an ideal world without rounding errors, we could compute $Y_0, Y_1, Y_2, \ldots$, and then get the stable invariant subspace as $\ker Y_\infty$ (or, rather, the invariant subspace associated to the n smallest singular values of Y_∞, since in an ideal world without rounding errors it is nonsingular).

We can do something similar, if we work in a suitable format.

Standard Symplectic Form

Goal: write $Y_0 = (I - \mathcal{H})^{-1}(I + \mathcal{H})$ as

$$Y_0 = \begin{bmatrix} I & G_0 \\ 0 & F_0 \end{bmatrix}^{-1} \begin{bmatrix} E_0 & 0 \\ H_0 & I \end{bmatrix}.$$

Trick: this is equivalent to finding M such that

$$M \begin{bmatrix} (I - \mathcal{H}) & (I + \mathcal{H}) \end{bmatrix} = \begin{bmatrix} I & G_0 & E_0 & 0 \\ 0 & F_0 & H_0 & I \end{bmatrix}.$$

And this M must be the inverse of block columns $(1, 4)$.

Structural properties:

- if $\mathcal{H}$ is Hamiltonian, Y_0 is symplectic.
  Proof: via $(I - \mathcal{H})^* J (I - \mathcal{H}) = (I + \mathcal{H})^* J (I + \mathcal{H})$.
- If Y_0 is symplectic, $E_0 = F_0^*, G_0 = G_0^*, H_0 = H_0^*$.
- Moreover, if $G \succeq 0$, $H \succeq 0$, then $G_0 \succeq 0$, $H_0 \preceq 0$ (tedious).

Doubling algorithm

Plan Given $Y_k = \begin{bmatrix} I & G_k \\ 0 & E_k^* \end{bmatrix}^{-1} \begin{bmatrix} E_k & 0 \\ H_k & I \end{bmatrix}$, compute $Y_{k+1} = -Y_k^2 = \begin{bmatrix} I & G_{k+1} \\ 0 & E_{k+1}^* \end{bmatrix}^{-1} \begin{bmatrix} E_{k+1} & 0 \\ H_{k+1} & I \end{bmatrix}$.

Similar to the 'inverse-free sign method' described earlier.

The swap: If $Y_k = \mathcal{M}_k^{-1}\mathcal{N}_k$, then $-Y_k^2 = -\mathcal{M}_k^{-1}\mathcal{N}_k\mathcal{M}_k^{-1}\mathcal{N}_k = \mathcal{M}_k^{-1}\widehat{\mathcal{M}}_k^{-1}\widehat{\mathcal{N}}_k\mathcal{N}_k = (\widehat{\mathcal{M}}_k\mathcal{M}_k)^{-1}(\widehat{\mathcal{N}}_k\mathcal{N}_k)$, where $\widehat{\mathcal{M}}_k, \widehat{\mathcal{N}}_k$ satisfy $\widehat{\mathcal{M}}_k^{-1}\widehat{\mathcal{N}}_k = -\mathcal{N}_k\mathcal{M}_k^{-1}$, i.e.,

$$\begin{bmatrix} \widehat{\mathcal{M}}_k & \widehat{\mathcal{N}}_k \end{bmatrix} \begin{bmatrix} \mathcal{N}_k \\ \mathcal{M}_k \end{bmatrix} = 0.$$

Doubling: the swap

$$\begin{bmatrix} I & \widehat{G}_k & \widehat{E}_k & 0 \\ 0 & \widehat{F}_k & \widehat{H}_k & I \end{bmatrix} \begin{bmatrix} E_k & 0 \\ H_k & I \\ I & G_k \\ 0 & E_k^* \end{bmatrix} = 0$$

holds if

$$\begin{aligned} \begin{bmatrix} \widehat{G}_k & \widehat{E}_k \\ \widehat{F}_k & \widehat{H}_k \end{bmatrix} &= - \begin{bmatrix} E_k & 0 \\ 0 & E_k^* \end{bmatrix} \begin{bmatrix} H_k & I \\ I & G_k \end{bmatrix}^{-1} \\ &= \begin{bmatrix} E_k & 0 \\ 0 & E_k^* \end{bmatrix} \begin{bmatrix} G_k(I - H_k G_k)^{-1} & -(I - G_k H_k)^{-1} \\ -(I - H_k G_k)^{-1} & H_k(I - G_k H_k)^{-1} \end{bmatrix}. \end{aligned}$$

Doubling: the formulas

Putting everything together,

$$\begin{aligned} \begin{bmatrix} E_{k+1} & 0 \\ H_{k+1} & I \end{bmatrix} &= \begin{bmatrix} -E_k(I - G_k H_k)^{-1} & 0 \\ E_k^* H_k (I - G_k H_k)^{-1} & I \end{bmatrix} \begin{bmatrix} E_k & 0 \\ H_k & I \end{bmatrix} \\ &= \begin{bmatrix} -E_k(I - G_k H_k)^{-1} E_k & 0 \\ H_k + E_k^* H_k (I - G_k H_k)^{-1} E_k & I \end{bmatrix} \end{aligned}$$

and an analogous computation gives E_{k+1}^*, G_{k+1}:

Structured doubling algorithm

$$\begin{aligned} E_{k+1} &= -E_k(I - G_k H_k)^{-1} E_k, \\ G_{k+1} &= G_k + E_k G_k (I - H_k G_k)^{-1} E_k^*, \\ H_{k+1} &= H_k + E_k^* H_k (I - G_k H_k)^{-1} E_k. \end{aligned}$$

SDA: details

Note that (even when the series does not converge)

$$G_k(I - H_k G_k)^{-1} = G_k + G_k H_k G_k + G_k H_k G_k H_k G_k + \cdots = (I - G_k H_k)^{-1} G_k,$$

and this matrix is symmetric. If $G_k = B_k B_k^*$, then it can also be rewritten as $B_k(I - B_k^* H_k B_k)^{-1} B_k^*$ (inverting a symmetric matrix).

Monotonicity If $H_k \preceq 0$ then $G_k(I - H_k G_k)^{-1} \succeq 0$. Hence, $0 \preceq G_0 \preceq G_1 \preceq \ldots$, and $0 \succeq H_0 \succeq H_1 \succeq H_2 \succeq \ldots$

Cost As much as a $2n \times 2n$ inversion $M^{-1}N$, if you put everything together. Unlike the sign algorithm, we have a bound $\sigma_{\min}(I - H_k G_k) \geq 1$ (because $G_k \succeq 0$, $H_k \preceq 0$).

SDA: the dual equation

To analyze convergence, we need to introduce another matrix. Let Y be the matrix such that

$$\mathcal{H}\begin{bmatrix}-Y\\ I\end{bmatrix} = \begin{bmatrix}A & -G\\ -Q & -A^*\end{bmatrix}\begin{bmatrix}-Y\\ I\end{bmatrix} = \begin{bmatrix}-Y\\ I\end{bmatrix}\widehat{\mathcal{R}}$$

is the *anti-stable* invariant subspace of $\mathcal{H}$, i.e., $\Lambda(\widehat{\mathcal{R}}) \subset RHP$.

$\begin{bmatrix}I\\ Y\end{bmatrix}$ spans the stable subspace of $\mathcal{H}^* = -J\mathcal{H}J$; we can prove that the subspace has this form if (A^T, C^T) controllable (typically satisfied).

SDA: convergence (intuitively)

Theorem
In SDA, $E_k \to 0, G_k \to Y, H_k \to -X$. Convergence is quadratic, i.e., $\|H_k + X\| = \mathcal{O}(\rho^{2^k})$ for some $\rho \in [0,1)$, as $k \to \infty$.

Intuitive view $E_k \to 0$, approximately squared at each time. Hence

$$\mathcal{H}_k = \begin{bmatrix}I & G_k\\ 0 & E_k^*\end{bmatrix}^{-1}\begin{bmatrix}E_k & 0\\ H_k & I\end{bmatrix}$$

has n eigenvalues $\to 0$ and n that $\to \infty$. $\ker \mathcal{H}_k \approx \begin{bmatrix}I\\ -H_k\end{bmatrix}$, so $-H_k \to X$.

Dually, "$\ker \mathcal{H}_k^{-1}$" (a thing that shouldn't exist...) $\approx \begin{bmatrix}-G_k\\ I\end{bmatrix}$, so $G_k \to Y$.

SDA convergence (formally)

Proof some manipulations give

$$\mathcal{H}_0\begin{bmatrix}I\\ X\end{bmatrix} = (I-\mathcal{H})^{-1}(I+\mathcal{H})\begin{bmatrix}I\\ X\end{bmatrix} = \begin{bmatrix}I\\ X\end{bmatrix}(I-\mathcal{R})^{-1}(I+\mathcal{R}).$$

where $\mathcal{S} = (I-\mathcal{R})^{-1}(I+\mathcal{R})$ has eigenvalues in the unit circle. Thus

$$\begin{bmatrix}I & G_k\\ 0 & E_k^*\end{bmatrix}\begin{bmatrix}I\\ X\end{bmatrix} = \begin{bmatrix}I\\ X\end{bmatrix}\begin{bmatrix}E_k & 0\\ H_k & I\end{bmatrix}\begin{bmatrix}I\\ X\end{bmatrix}\mathcal{S}^{2^k}.$$

which implies

$$\begin{aligned} E_k &= (I+G_kX)\mathcal{S}^{2^k},\\ H_k + X &= E_k^*X\mathcal{S}^{2^k} = (\mathcal{S}^{2^k})^*(I+XG_k)\mathcal{S}^{2^k} \succeq 0. \end{aligned}$$

The same computation on the dual equation gives $G_k \preceq Y$, so G_k is bounded and $E_k \to 0, H_k + X \to 0$ (quadratically as $\mathcal{S}^{2^k}$).

Chapter 16

Methods for large-scale CAREs

Methods for large-scale control systems

We give a hint of the methods used for large-scale control systems.

What does a large-scale control system look like?

Example: the heat equation: finite-difference discretization of a 2D or 3D structure, possibly with a nontrivial shape.

- Large, sparse $A \in \mathbb{R}^{n\times n}$ produced by the discretization: the state evolves *locally*.
- $B \in \mathbb{R}^{n\times m}$ with $m \ll n$: the control acts only on a few points. Hence $G = BR^{-1}B^*$ has low rank.
- $Q = C^*C$ is also often taken to be low-rank: the energy is based on 'output' values measured in a few points.

Large-scale Lyapunov equations

We focus on a simple problem: solving Lyapunov equations $AX + XA^* + BB^* = 0$ with $\Lambda(A) \subset LHP$: then we can run Bass algorithm and Newton's method for CAREs.

Assumptions

- $\Lambda(A) \subset LHP$
- A sparse or at least with fast linear algebra operations $v \mapsto Av$, $v \mapsto (A - \alpha I)^{-1}v$ (e.g., sparse-plus-low-rank, Toeplitz).
- $B \in \mathbb{R}^{n\times m}$, with $m \ll n$. Actually we can take $m = 1$, $B = b \in \mathbb{R}^n$: a rank-m matrix is the sum of m rank-1 matrices, and the Lyapunov equation is linear.

Roadblock: the solution X is typically dense and full-rank ($X \succ 0$)!

Solution: often, $X \approx ZZ^*$ with a tall thin Z: it has *decaying singular values* and *low numerical rank* (we will see why).

ADI (alternating-direction implicit iteration)

Idea Let's convert our continuous-time problem (Lyapunov equation)

$$AX + XA^* + bb^* = 0 \tag{L}$$

to a discrete-time one (Stein equation)

$$\hat{A}X + X\hat{A}^* + \hat{b}\hat{b}^* = 0, \tag{S}$$

since those can be solved with a simpler fixed-point iteration.

Theorem

Let $\tau > 0$, so that $\Lambda(A - \tau I) \subset LHP$. Then, X solves (L) if and only if it solves (S) with $\hat{A} := (A - \tau I)^{-1}(A + \tau I)$, $\hat{b} := \sqrt{2\tau}(A - \tau I)^{-1}b$.

Proof Expand in two ways

$$(A - \tau I)X(A - \tau I)^* - (A + \tau I)X(A + \tau I)^* - 2\tau bb^* = 0.$$

Solving Stein equations

We can solve (S) with the fixed-point iteration

$$X_0 = 0, \quad X_k = \hat{A}X_{k-1}\hat{A}^* + \hat{b}\hat{b}^*. \tag{FP}$$

Lemma

If $\Lambda(A) \subset LHP$ and $\tau > 0$, then $\Lambda(\hat{A}) \subset \mathbb{D}$ (unit disk).

Proof If $\lambda \in LHP$, then $\operatorname{dist}(\lambda, -\tau) < \operatorname{dist}(\lambda, \tau)$, thus $\frac{|\lambda+\tau|}{|\lambda-\tau|} < 1$.

Convergence theorem

For the iteration (FP),

$$X_k - X = \hat{A}^k(X_0 - X)\hat{A}^{k*}.$$

Proof Induction.

From the two results we get *linear* convergence to the solution X.

Extras: Time discretization

Note that $\hat{A}$ and a (scaled) version of $\hat{b}$ be obtained by discretizing the control system with the *midpoint method*:

$$\dot{x} = Ax + Bu$$

is discretized to

$$\frac{x_{k+1}-x_k}{h} = \frac{1}{2}\left(Ax_k + Bu_k + Ax_{k+1} + Bu_{k+1}\right),$$

i.e.,

$$x_{k+1} = (I - \tfrac{h}{2}A)^{-1}(I + \tfrac{h}{2}A)x_k + (I - \tfrac{h}{2}A)^{-1}B(u_k + u_{k+1})\tfrac{h}{2}.$$

This specific method is particularly nice, because it preserves stability: the open-loop system $\dot{x} = Ax$ is stable iff $x_{k+1} = (I - \frac{h}{2}A)^{-1}(I + \frac{h}{2}A)x_k$ is so.

Low-rank formulation

We have

$$X_k = \hat{b}\hat{b}^* + \hat{A}\hat{b}\hat{b}^*\hat{A} * + \hat{A}^2\hat{b}\hat{b}^*\hat{A}^{2*} + \cdots + \hat{A}^k\hat{b}\hat{b}^*\hat{A}^{k*}.$$

Or, in terms of its *low-rank factor*

$$Z_k = \begin{bmatrix}\hat{b} & \hat{A}\hat{b} & \hat{A}^2\hat{b} & \dots & \hat{A}^k\hat{b}\end{bmatrix}, \quad X_k = Z_k Z_k^*.$$

Iterative computation

$$\begin{cases} v_1 = \hat{b}, \\ v_{k+1} = \hat{A}v_k = (A - \tau I)^{-1}(A + \tau I)v_k = v_k + 2\tau(A - \tau I)^{-1}v_k. \end{cases}$$

Cost: One shifted solve with A per iteration.

Multiple shifts

The key to get faster convergence is changing the value of τ at each step: given an arbitrary sequence of positive *shifts* (or *poles*) $\tau_1, \tau_2, \dots$, we set

$$\hat{A}_k := (A - \tau_k I)^{-1}(A + \tau_k I), \quad \hat{b}_k := \sqrt{2\tau_k}(A - \tau_k I)^{-1}b.$$

Then one can set up the iteration

$$X_0 = 0, \quad X_k = \hat{A}_k X_{k-1}\hat{A}_k^* + \hat{b}_k\hat{b}_k^*.$$

Proceeding in the same way one gets

$$Z_k = \begin{bmatrix}\hat{b}_k & \hat{A}_k\hat{b}_{k-1} & \hat{A}_k\hat{A}_{k-1}\hat{b}_{k-2} & \dots & \hat{A}_k\hat{A}_{k-1}\dots\hat{A}_2\hat{b}_1\end{bmatrix}.$$

Rearranging the computation

It's less clear from this formulation how to compute columns iteratively. However one can rearrange things using commutativity:

$$\begin{aligned}
&\frac{1}{\sqrt{2\tau_{k-2}}}\hat{A}_k\hat{A}_{k-1}\hat{b}_{k-2} \\
&= (A-\tau_k I)^{-1}(A+\tau_k I)(A-\tau_{k-1}I)^{-1}(A+\tau_{k-1}I)(A-\tau_{k-2}I)^{-1}b \\
&= \underbrace{(A-\tau_{k-2}I)^{-1}(A+\tau_{k-1}I)\,\underbrace{(A-\tau_{k-1}I)^{-1}(A+\tau_k I)\,\underbrace{(A-\tau_k I)^{-1}b}_{=:w_1}}_{=:w_2}}_{=:w_3}
\end{aligned}$$

Thus we get $w_1 = (A-\tau_k I)^{-1}\hat{b}$ and

$$\begin{aligned}
w_{j+1} &= (A-\tau_{k-j}I)^{-1}(A+\tau_{k-j+1}I)w_j \\
&= w_j + (\tau_{k-j}+\tau_{k-j+1})(A-\tau_{k-j}I)^{-1}w_j.
\end{aligned}$$

Low-rank ADI: the formulation

Reversing the order of the τ_j for simplicity, we get

Low-rank ADI

$$v_1 = \sqrt{2\tau_1}(A-\tau_1)^{-1}b, \quad v_j = \frac{\sqrt{2\tau_j}}{\sqrt{2\tau_{j-1}}}\left(v_j + (\tau_{j-1}+\tau_j)\left(A-\tau_j I\right)^{-1}v_j\right).$$

$$Z_k = \begin{bmatrix} v_1 & v_2 & \dots & v_k \end{bmatrix}.$$

One can also use complex shifts (details omitted, $\overline{\tau}_j$s appears).

ADI: convergence

Proceeding analogously to the one-shift case, one gets

$$X_k - X_* = \hat{A}_k\hat{A}_{k-1}\cdots\hat{A}_1(X_0-X_*)\hat{A}_1^*\cdots\hat{A}_{k-1}^*\hat{A}_k^* = g(A)(X_0-X_*)g(A)^*,$$

where $g(x) = \prod_{j=1}^k \frac{x-\tau_j}{x+\tau_j}$.

If $A = V\Lambda V^{-1}$, then

$$\|g(A)\| = \|Vg(\Lambda)V^{-1}\| \le \kappa(V)\max_{\lambda\in\Lambda(A)}\prod_{j=1}^k\frac{|\lambda-\tau_j|}{|\lambda+\tau_j|}.$$

How to choose τ_j's that make this norm small? If A has at most k distinct eigenvalues, we can choose $\tau_j = -\lambda_j$ to get $g(A) = 0$ and *exact convergence* in k steps.

If A has k *clusters*, we get a small $\|g(A)\|$ after k steps (cfr. Arnoldi convergence theory).

ADI convergence

The ADI shift choice problem

$$\eta_k = \min_{\tau_1,\dots,\tau_k} \max_{\lambda\in\Lambda(A)} \prod_{j=0}^{k-1} \frac{|\lambda-\tau_j|}{|\lambda+\tau_j|}.$$

To compute an optimal choice, we need $\Lambda(A)$, unfeasible. And the choice would change at each step k, requiring recomputation of previous iterates.

To solve the second issue, one usually computes a small number k' of shifts initially, before starting the iteration, and reuses them cyclically with $\tau_k = \tau_{\mathrm{mod}(k,k')}$. To solve the first:

Workaround 1 Often the max is taken on the largest/smallest eigenvalue. Hence we run a few steps of Arnoldi on A and A^{-1} to get $\{\mu_1,\dots,\mu_d\}$ that approximate the extremal eigenvalues of A.

ADI optimal shifts

Alternatively,

Workaround 2 replace $\Lambda(A)$ with a region $C \subset LHP$ enclosing the eigenvalues of A: for instance, if $A = A^*$, all eigenvalues are in an interval $C = [a, b]$. Then, look for

$$\hat{\eta}_k = \min_{\tau_0,\dots,\tau_k} \max_{\lambda\in C} \prod_{j=0}^{k-1} \frac{|\lambda-\tau_j|}{|\lambda+\tau_j|}.$$

This is a classical problem from approximation theory: look for polynomials that are *small* on C and *large* on $-C$. Explicit solutions can be constructed from elliptic functions for many choices of C. It is known that $\hat{\eta}_k \sim r^k$ for a certain $r < 1$ (that depends on $\frac{b}{a}$ in the symmetric case: worse bounds for ill-conditioned A).

Consequence Since $\|X_* - X_k\| \sim r^k$, and $\mathrm{rk}\, X_k = k$, it follows that $\sigma_{k+1}(X) \lesssim r^k$, so X has *low numerical rank*.

Extra: Residual computation

Detail As a stopping criterion for ADI, we may use the residual $AZ_kZ_k^* + Z_kZ_k^*A^* + BB^*$, but how to compute it without assembling large matrices?

For $X_k = Z_kZ_k^*$, with $Z_k \in \mathbb{R}^{n\times k}$, we have

$$AZ_kZ_k^* + Z_kZ_k^*A^* + BB^* = \begin{bmatrix} Z_k & AZ_k & B \end{bmatrix} \begin{bmatrix} 0 & I & 0 \\ I & 0 & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} Z_k & AZ_k & B \end{bmatrix}^*.$$

Using a QR of the tall thin $\begin{bmatrix} Z_k & AZ_k & B \end{bmatrix}$, we can compute this norm in $O(nk^2)$.

Rational Arnoldi

An alternative algorithm for large-scale Lyapunov equations comes from Krylov subspace ideas.

Note that the computed Z_k has columns of the form $r(A)b$, where $r(x) = p(x)/q(x)$, with fixed denominator $q(x) = (x - \tau_1)(x - \tau_2)\dots(x - \tau_k)$.

Hence, our approximation Z_k lives in the *rational Arnoldi subspace*

$$K_q(A, b) = \{q(A)^{-1}p(A)b\colon \deg p < k\} = q(A)^{-1}K_k(A, b).$$

Idea: first compute this subspace, then solve the projected equation.

Galerkin Projection

Given an orthonormal basis U_k of $K_q(A, b)$:

1. Set $X_k = U_k Y_k U_k^*$;
2. Assume 'orthogonal residual': $U_k^*(AX_k + X_k A^* + BB^*)U_k = 0$.

Produces a projected Lyapunov equation

$$(U_k^* A U_k)Y + Y(U_k^* A U_k)^* + U_k^* B B^* U_k = 0.$$

Difficulty 1 Even if $\Lambda(A) \subset LHP$, the same property does not always hold for $U_k^* A U_k$.

Recall: the eigenvalues of $A_k = U_k^* A U_k$ are in the *field of values* of A, which is hull $\Lambda(A)$ for normal A, but larger (possibly by much) for non-normal A.

Difficulty 2 (main one, shared with ADI): good *pole selection*.