

Low-Rank Approximation

Lecture 1 – Low-rank Approximability

Leonardo Robol, University of Pisa, Italy
Cagliari, 23 Sep 2019

Here is an outline on the content of these lectures.

1. Introduction, theoretical tools, structures of interest.
2. How to approximate a low-rank matrix?
3. Functional low-rank approximation and Chebfun2.
4. Matrix equations.
5. Matrix equations in practice – more general rank structures.

As we will see throughout these lectures – low-rank approximation is a general concept that fits different applications:

- Data **compression**.
- Acceleration of **transforms** (Fast multipole schemes / polynomial transforms / singular kernels).
- Data identification (PCA).
- Convenient way of treating **bivariate** (and possibly multivariate) **functions**.

Notation

Throughout this course:

- A is an $m \times n$ matrix. That is:

$$A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}, \quad a_{ij} \in \mathbb{C} \text{ or } a_{ij} \in \mathbb{R}.$$

- A^T is the **transposed matrix** (row exchanged with columns).
- $U \in \mathbb{C}^{m \times k}$ and $V \in \mathbb{C}^{n \times k}$ will usually be **tall and skinny** matrices. We use them to approximate $A \approx UV^T$.
- u, v, w are **vectors**.
- $\|\cdot\|$ denote matrix and vector **norms**.
- $\sigma_j(A)$ will denote **singular values** of A .

What is rank?

Back to our first day of linear algebra, we can think of a matrix $A \in \mathbb{C}^{m \times n}$ as a **linear operator** from \mathbb{C}^n to \mathbb{C}^m that sends v into Av .

$$w = Av = \begin{bmatrix} a_1 & \dots & a_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = a_1 v_1 + \dots + a_n v_n.$$

- The **column-rank** is the dimension of the range of A .
- The **row-rank** is the column rank of A^T

What is rank?

Back to our first day of linear algebra, we can think of a matrix $A \in \mathbb{C}^{m \times n}$ as a **linear operator** from \mathbb{C}^n to \mathbb{C}^m that sends v into Av .

$$w = Av = \begin{bmatrix} a_1 & \dots & a_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = a_1 v_1 + \dots + a_n v_n.$$

- The **column-rank** is the dimension of the range of A .
- The **row-rank** is the column rank of A^T

For matrices, we have row-rank = col-rank. That's why we call it just **rank**.

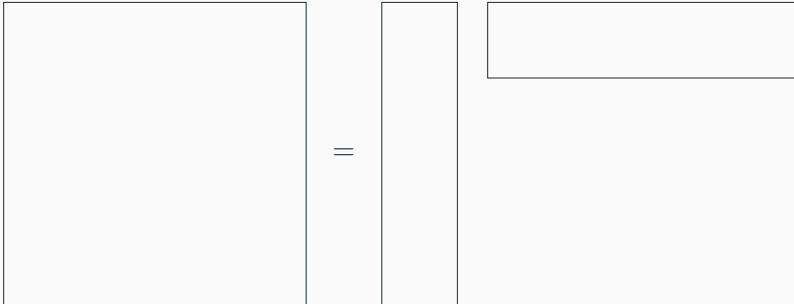
Why do we like low-rank matrices?

Several reasons, which we will explore throughout these lectures, but mostly the advantages are (assume $m = n$)

- **Reduced storage** (typically from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$)
- **Reduced complexities**: often linear algebra operations can go down from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$.
- **Parameter identification**: reduction to the “real” number of degrees of freedom. This will be the kind of results related to PCA, which we will discuss in the next days.

Storage reduction

Assume we have a $m \times n$ matrix A of rank k , then we can store it as

$$A = UV^T$$


Storage on the left is $\mathcal{O}(mn)$, on the right is $\mathcal{O}(k(m+n))$. This has tons of applications; for instance, **image compression**.

Image compression

Consider the following image with 900×1600 pixels:



8 bit per pixel, RGB channels

Storage is $3 \cdot 8 \cdot 900 \cdot 1600$ bits ≈ 4.2 MB.

The exact rank is 900.

Image compression

Consider the following image with 900×1600 pixels:



8 bit per pixel, RGB channels

Storage is $3 \cdot 8 \cdot 900 \cdot 1600$ bits \approx 4.2 MB.

The exact rank is 900.

We can **approximate it** as an image with a lower rank
(and thus cheaper to store!).

Image compression

Consider the following image with 900×1600 pixels:



8 bit per pixel, RGB channels

Approximated with rank 100.

Storage is $3 \cdot 8 \cdot 100 \cdot (900 + 1600)$ bits \approx 730 KB.

Reminder: the exact rank was 900.

Image compression

Consider the following image with 900×1600 pixels:



8 bit per pixel, RGB channels

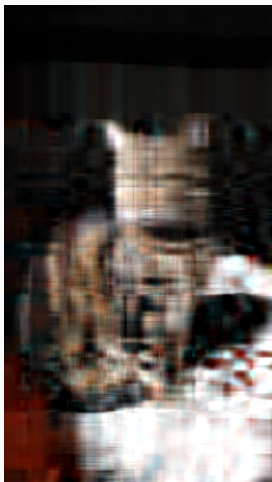
Approximated with rank 50.

Storage is $3 \cdot 8 \cdot 50 \cdot (900 + 1600)$ bits ≈ 360 KB.

Reminder: the exact rank was 900.

Image compression

Consider the following image with 900×1600 pixels:



8 bit per pixel, RGB channels

Approximated with rank 10.

Storage is $3 \cdot 8 \cdot 10 \cdot (900 + 1600)$ bits ≈ 73 KB.

Reminder: the exact rank was 900.

Complexity reduction: an integral transform

Consider the following **integral transform** for functions over $[0, 1]$:

$$\hat{f}(x) = \int_0^1 \frac{f(y)}{1+x+y} dy.$$

If we discretize $[0, 1]$ with N points by $x_j = jh$, $h = \frac{1}{N-1}$ and $j = 0, \dots, N-1$, then:

$$\hat{f}(x_j) \approx -h \frac{3f(0) + f(1)}{6} + \sum_{j=0}^{N-1} h \frac{g(y_j)}{1+x_i+y_j}. \quad (\text{trapezoidal rule})$$

In linear algebra terms, we have

$$\hat{f} = Cf + f_{01}, \quad C_{ij} = \frac{1}{1+x_i+y_j}$$

Complexity reduction: an integral transform

Consider the following **integral transform** for functions over $[0, 1]$:

$$\hat{f}(x) = \int_0^1 \frac{f(y)}{1+x+y} dy.$$

If we discretize $[0, 1]$ with N points by $x_j = jh$, $h = \frac{1}{N-1}$ and $j = 0, \dots, N-1$, then:

$$\hat{f}(x_j) \approx -h \frac{3f(0) + f(1)}{6} + \sum_{j=0}^{N-1} h \frac{g(y_j)}{1+x_i+y_j}. \quad (\text{trapezoidal rule})$$

In linear algebra terms, we have

$$\hat{f} = Cf + f_{01}, \quad C_{ij} = \frac{1}{1+x_i+y_j}$$

Without further assumptions, the cost of a matrix-vector multiplication is $\mathcal{O}(N^2)$ flops.

The matrix C is a **Cauchy matrix**, and can be approximated by a rank $k \ll N$ one $C \approx UV^T$. Then,

$$\hat{f} = Cf = U(V^T f), \quad \text{that requires } \mathcal{O}(Nk) \text{ flops.}$$

Low-rank approximation in action

The matrix C is a **Cauchy matrix**, and can be approximated by a rank $k \ll N$ one $C \approx UV^T$. Then,

$$\hat{f} = Cf = U(V^T f), \quad \text{that requires } \mathcal{O}(Nk) \text{ flops.}$$

Example: $N = 1000$, $k = 5$, we have UV^T of rank k such that

$$\|C - UV^T\|_2 \leq \|C\|_2 \cdot \epsilon, \quad \epsilon \approx 10^{-9}$$

Low-rank approximation in action

The matrix C is a **Cauchy matrix**, and can be approximated by a rank $k \ll N$ one $C \approx UV^T$. Then,

$$\hat{f} = Cf = U(V^T f), \quad \text{that requires } \mathcal{O}(Nk) \text{ flops.}$$

Example: $N = 1000$, $k = 5$, we have UV^T of rank k such that

$$\|C - UV^T\|_2 \leq \|C\|_2 \cdot \epsilon, \quad \epsilon \approx 10^{-9}$$

- Are all matrices low-rank?
- Can we say a priori when the matrix we care about is?
- May we consider also more general, and maybe “weaker”, structures?

More general rank structures

We have considered the definition of **rank** and **numerical rank**. Later on, we will consider also more general (but related) structures.

Informal definition

We say that the matrix A is *hierarchically low-rank* if it can be partitioned as:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

such that:

- A_{12} and A_{21} are low-rank.
- A_{11} and A_{22} are again hierarchically low-rank, or they are “small”.

In practice, often we select a rank k and we check if the off-diagonal blocks are of rank at most k , until the diagonal blocks do dimension at most k .

The analogous definition might be given for numerical rank.

Numerical hierarchical rank

Let A be a matrix such that for every partitioning:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

such that:

- A_{12} , A_{21} are numerical of rank k : there exist perturbations δA_{21} and δA_{12} such that $A_{21} + \delta A_{21}$ and $A_{12} + \delta A_{12}$ are both rank k , and $\|\delta A_{21}\|_2, \|\delta A_{12}\|_2 \leq \epsilon$.
- This property continues to hold recursively on the diagonal blocks.

Does this implies the existence of a hierchically rank k matrix $A + \delta A$, such that $\|\delta A\|_2 \lesssim \epsilon$? Yes, but we need to assume $\|\delta A\|_2 \leq \log(n)\epsilon$.

A lot of matrices are in this class:

- Banded matrices.
- Green matrices.
- Discretization of integral operators.
- The class is closed under addition, multiplication, and inversion: any of these operations still gives a matrix inside the class¹.

¹Up to a moderate increase in the rank.

Efficient representation for hierarchically low-rank matrices

Efficiently storing (and operating on) such matrices is inherently more complicated than in the low-rank case. For a rank k matrix, we can always build the factorization

$$A = UV^T, \quad U \in \mathbb{C}^{m \times k}, \quad V \in \mathbb{C}^{n \times k}.$$

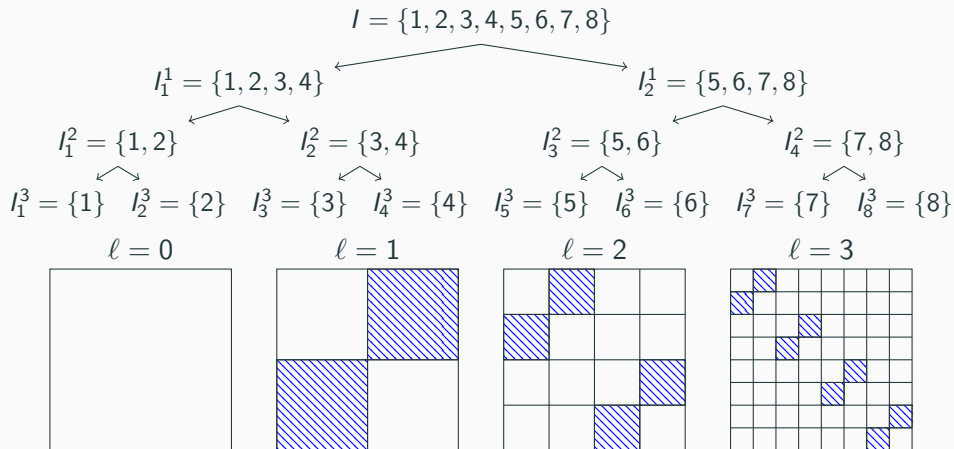
The analogous in the hierarchical case requires a recursive representation. Given the index sets:

$$I = \{1, \dots, m\}, \quad J = \{1, \dots, n\},$$

we build two trees recursive representing the splitting of the indices. For square matrices, we can consider just one index set I .

Row and column clusters

The tree representing the recursive splitting is called **row** or **column cluster**.



HODLR representation

The simpler representation is obtained by storing each off-diagonal block at each level as a low-rank matrix outer product UV^T :

- At each level ℓ , we need 2^ℓ blocks;
- The blocks at level ℓ are of size $\approx \frac{n}{2^\ell} \times \frac{n}{2^\ell}$, so they require about $\frac{nk}{2^\ell}$ each: $\mathcal{O}(nk)$ storage for every level.
- Since we have about $\mathcal{O}(\log n)$ levels, this yield a storage cost $\mathcal{O}(nk \log n)$.

Most arithmetic operation are easily implementable within this format, and they require similar complexities: for instance, for the sum $A + B$ we have $\mathcal{O}(n(k_A + k_B)^2 \log n)$.

Typical complexity: $\mathcal{O}(nk^2 \log^\alpha n)$, $1 \leq \alpha \leq 2$.

Fast matrix-vector product

As in the low-rank case, this has advantages not only for the storage, but also for efficiently applying structured operators:

$$Av = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} A_{11}v_1 + A_{12}v_2 \\ A_{21}v_1 + A_{22}v_2 \end{bmatrix}$$

We need to implement:

- A fast matrix-vector for the products $A_{21}v_1$ and $A_{12}v_2$: these are **low-rank**, so we already know how to do it!
- Fast matrix-vector products for $A_{11}v_1$ and $A_{22}v_2$; these matrices are again **hierarchically low-rank**, so either we handle this directly (if they are small), or we do it recursively.

Example: sum in the HODLR format

The sum can be implemented recursively:

$$A + B = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11} + B_{11} & A_{12} + B_{12} \\ A_{21} + B_{21} & A_{22} + B_{22} \end{bmatrix}$$

- The diagonal blocks are handled directly if they are sufficiently small, otherwise the procedure is called recursively.
- The off-diagonal blocks are handled by generating a low-rank representation for the sum:

$$A_{12} = U_A V_A^T, \quad B_{12} = U_B V_B^T \implies A_{12} + B_{12} = \begin{bmatrix} U_A & U_B \end{bmatrix} \begin{bmatrix} V_A & V_B \end{bmatrix}^T,$$

and analogously for A_{21} , B_{21} .

- This new representation might be redundant in general; the hierarchical rank of $A + B$ is at most the sum of the hierarchical ranks of A and B .

Another example: matrix multiplication

$$AB = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$$

Here we have:

- Products of low-rank matrices:

$$(U_A V_A^T) \cdot (U_V B_V^T) = U_A (V_A^T U_V) B_V^T.$$

- Products of hierarchical matrices with a low-rank matrix:

$$A(U_B V_B^T) = (AU_B) V_B^T \quad (\text{need fast matvec})$$

- Products of two hierarchical matrices, which is handled recursively.
- Hierarchical rank of AB is at most the sum of the hierarchical ranks of A and B^2 .

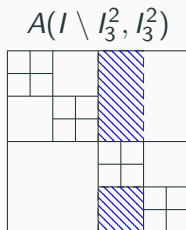
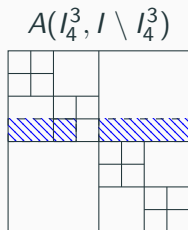
²Under some additional assumptions, otherwise it might increase of a $\log n$ factor

HSS representation

If we make stronger hypotheses on the rank structure, we can get rid of the logarithmic factor. This requires a more involved representation; matrices in this form are called **hierarchical semi separable** (HSS).

Let I_i^ℓ be the index sets at level i , then, we call:

- **HSS block rows**, the submatrices $A(I_i^\ell, I \setminus I_i^\ell)$.
- **HSS block columns**, the submatrices $A(I \setminus I_i^\ell, I_i^\ell)$.



Construction of the HSS representation

Under the assumption of the previous slide, we can construct the basis for each HSS block row and columns at the lowest level. We call these matrices:

- U_i^ℓ , for the basis for the rows at i -th level.
- V_i^ℓ , for the basis of the columns at the i -th level.

Then, assuming U_i^d, V_i^d are known, where d is the number of levels:

$$U_i^{(\ell)} = \begin{bmatrix} U_{i_1}^{(\ell+1)} & 0 \\ 0 & U_{i_2}^{(\ell+1)} \end{bmatrix} R_{U,i}^{(\ell)}, \quad V_j^{(\ell)} = \begin{bmatrix} V_{j_1}^{(\ell+1)} & 0 \\ 0 & V_{j_2}^{(\ell+1)} \end{bmatrix} R_{V,j}^{(\ell)},$$

- The operators $R_{U,i}^{(\ell)}$ are called **translation operators**.
- Similar operators $R_{V,i}$ are the defined for the columns.
- Existence is guaranteed by the conditions on the rank of the HSS block row and columns.
- Each off-diagonal block is represented as $U_i^\ell B_{ij}^\ell (V_j^\ell)^*$. We call the B_{ij} **core blocks**.

Storage cost for HSS matrices

To store an HSS matrix, we need:

- The basis U_i^ℓ and V_i^ℓ at the lowest level: $\mathcal{O}(nk)$ storage.
- The translation operators at level ℓ : there are 2^ℓ of them, and they require $\mathcal{O}(k^2)$ storage.
- The core blocks are $\mathcal{O}(2^\ell)$, and require $\mathcal{O}(k^2)$ each.

We now sum all the contributions, accounting for the fact that the number of levels is smaller than $\log \frac{n}{k}$, yields $\mathcal{O}(nk)$ storage.

Note that, generally, an HODLR matrix with ranks k might be an HSS matrix with rank $k \log n$ — but in all practical cases these two ranks match, so the HSS format requires less storage.

Let A be a matrix such that:

- All the HSS block rows are of rank k up to a perturbation smaller than ϵ .
- The same holds for the HSS block columns.

Does this implies the existence of a HSS matrix $A + \delta A$ with HSS rank k , such that $\|\delta A\|_2 \lesssim \epsilon$? Yes, but we need to assume $\|\delta A\|_2 \leq 2\sqrt{n}\epsilon$.

Proof is slightly more demanding.

We will meet HODLR and HSS matrices again at the end of these lectures, when we will have enough theory to construct and work with them.

In practice, all the complexity in the implementation can be overcome by using some pre-made tools: we will use `hm-toolbox` (Hierarchical Matrices Toolbox).

HODLR and HSS matrices in practice

We will meet HODLR and HSS matrices again at the end of these lectures, when we will have enough theory to construct and work with them.

In practice, all the complexity in the implementation can be overcome by using some pre-made tools: we will use `hm-toolbox` (Hierarchical Matrices Toolbox).

Brief recap:

- We have seen that low-rank matrices can accelerate arithmetic operations.
- Even for matrices that are not low-rank, we may still be able to find low-rank parts in them, and make operations fast, reduce storage costs.

The point is: how do we know beforehand when these good low-rank approximations **exist**? And how do we find them **in practice**?

Brief linear algebra recap: a **norm** on a vector space (for us, \mathbb{R}^m), is a function $\|\cdot\| : \mathbb{R}^m \rightarrow \mathbb{R}$ such that

1. $\|v\| = 0 \iff v = 0$.
2. $\|\alpha v\| = |\alpha| \cdot \|v\|$.
3. $\|v + w\| \leq \|v\| + \|w\|$ (triangular inequality).

Brief linear algebra recap: a **norm** on a vector space (for us, \mathbb{R}^m), is a function $\|\cdot\| : \mathbb{R}^m \rightarrow \mathbb{R}$ such that

1. $\|v\| = 0 \iff v = 0$.
2. $\|\alpha v\| = |\alpha| \cdot \|v\|$.
3. $\|v + w\| \leq \|v\| + \|w\|$ (triangular inequality).

If we lose property 1, then this is a **semi-norm**.

Important special case: if $\langle v, w \rangle$ is a **scalar product**, then

$$\|v\| := \sqrt{\langle v, v \rangle}$$

is a norm. Not all norms are like this!

Mostly, we will consider the following norms for matrices:

- **Induced norms**, or **operator norms**, which are defined by a norm $\|\cdot\|$ on \mathbb{R}^m :

$$\|A\| := \max_{\|v\|=1} \|Av\|.$$

In practice, we consider only $\|v\|_p := \left(\sum_{j=1}^n |v_j|^p\right)^{\frac{1}{p}}$.

- The **Frobenius norm** $\|A\|_F$, defined by

$$\|A\|_F := \left(\sum_{i,j} |A_{ij}|^2\right)^{\frac{1}{2}} = \sqrt{\text{tr}(A^T A)}.$$

Important norms and properties

Reminder: Orthogonal/Unitary matrices are such that $U^T U = U U^T = I$, or $U^H U = U U^H = I$ (on the real and complex field, respectively).

- The norms $\|\cdot\|_2$ and $\|\cdot\|_F$ are **unitarily invariant**, i.e., for every unitary norm U we have $\|AU\|_2 = \|A\|_2$ and $\|AU\|_F = \|A\|_F$.
- $\|A\|_\infty = \|A^T\|_1$, and

$$\|A\|_\infty := \max_{i=1,\dots,n} \sum_{j=1}^n |A_{ij}|.$$

- The Euclidean norm $\|\cdot\|_2$ is induced by a **scalar product**:

$$\langle v, w \rangle = v^T w = w^T v.$$

- All our norms are **submultiplicative**, i.e., $\|AB\| \leq \|A\| \cdot \|B\|$.
- The Frobenius norm is induced by the **scalar product on matrices**:

$$\langle A, B \rangle := \text{tr}(B^T A) = \text{tr}(A^T B).$$

Approximation using the SVD

Truncating the SVD gives optimal approximant with respect to the norms $\|\cdot\|_2$ and $\|\cdot\|_F$. Let us consider $A = U\Sigma V^T$ and

$$A_k := U \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & 0_{m-k, n-k} & & \\ & & & & & \end{bmatrix} V^T.$$

- A_k has rank $\leq k$, and exactly k if $\sigma_k \neq 0$.
- A_k is the **best rank k approximant** in the Euclidean and Frobenius norms.

Optimal approximants

For A_k , we have that for every B of rank at most k ,

$$\|A - B\|_2 \geq \|A - A_k\|_2 = \sigma_{k+1}.$$

and analogously for the Frobenius norm:

$$\|A - B\|_F \geq \|A - A_k\|_F = \left(\sum_{j=k+1}^{\min\{m,n\}} \sigma_j^2 \right)^{\frac{1}{2}}.$$

Brief recap:

- The SVD is the “**optimal** tool” for low-rank approximation;
- However, it is **very expensive** (it requires $\mathcal{O}(mn^2)$ if $m \geq n$ flops).

How much does it cost in practice?

We can try to time the computation of the SVD on a laptop, for $n \times n$ matrices:

$m = n$	Time (s)
128	0.005
256	0.011
512	0.060
1024	0.688
2048	5.234
4096	34.39
\vdots	\vdots

- This becomes quickly unpractical!
- One has to exploit the structure at our disposal to obtain a faster procedure.

The theory of low-rank approximation is closely related to the ones of **matrix functions**. We need to take a small detour in this theory. Throughout this section, all the matrices are square ($n \times n$).

- Matrix functions are interesting objects that often arise in applications.
- Well-known examples are:
 - **matrix powers** (A^k , or even A^α where $\alpha \in \mathbb{R}_+$)
 - the **matrix exponential** e^A — the computation of e^{tA} is related to solving ODEs and PDEs.
- Some less known functions are often used as well, such as $\log(A)$.

We do not need the full theory of matrix functions, but to answer the following:

- If $f(z)$ is a polynomial/rational function/analytic function, what does it mean to compute $f(A)$?
- How large is $\|f(A)\|_{2/F}$? Can we answer looking at the **spectrum** of A ?

The second result in particular will be key to understand how accurate are the low-rank approximation that we consider. Often the error will be expressed as $\|f(A)\|_{2/F}$.

Polynomials

Let $p(z)$ be a degree d polynomial, then $p(A)$ is defined by

$$p(A) = p_0 I + p_1 A + \dots + p_d A^d, \quad p(z) = \sum_{j=0}^d p_j z^j.$$

- This is **well-defined for any square matrix** A — no hypotheses needed.
- Note that $p(A)$ and A **commute** — and indeed if A is diagonalizable they have the same eigenvectors.

Assume $Av = \lambda v$. Then, $p(A)v = p(\lambda)v$. Then,

- The eigenvector stay the same (and so do invariant subspaces, and Jordan chains).
- if λ is an eigenvalues of A , then $p(\lambda)$ is an eigenvalue of $p(A)$.

These results are sometimes called **spectral mapping theorem**.

Analytic/Holomorphic functions

The definition for polynomials might suggest the following more general one:

Definition

Let $f(z)$ be an holomorphic function over the open ball $B(z_0, R)$, and A be a matrix with spectral radius $\rho(A - z_0 I) < R$. Then,

$$f(A) = \sum_{j=0}^{\infty} \frac{f^{(j)}(z_0)}{j!} (A - z_0 I)^j.$$

- If $f(z)$ is a polynomial, we have the same definition of before.
- If $z_0 = 0$ and $f(z) = e^z$ we get the **matrix exponential**.
- We have exactly the same property concerning eigenvalues and eigenvectors of A and $f(A)$ (same eigenvectors, mapped eigenvalues): again the **spectral mapping theorem**.
- Note that for every invertible V , $f(V^{-1}AV) = V^{-1}f(A)V$.

Looking at the spectrum

The comments on the spectrum inspire another definition:

Definition

Let A be diagonalizable, and V a basis of eigenvectors so that $V^{-1}AV = D$ is diagonal. Then,

$$f(A) := V^{-1}f(D)V, \quad f(D) := \begin{bmatrix} f(D_{11}) & & \\ & \ddots & \\ & & f(D_{nn}) \end{bmatrix}.$$

- Equivalent to the previous definition for analytic $f(z)$, but valid for **more general functions**.
- On the other hand, is valid only for **less general matrices**!
- But in reality, **“almost” all matrices are diagonalizable** (these are dense in the class of all matrices) – so we do not have problems for any **continuous function**.

A satisfying definition

We can give a definition that overcome all the drawbacks that we had, using the Jordan Canonical form.

Definition

Let $f(z)$ be a function defined at the eigenvalues of λ_i of A , and such that has derivatives of order at least equal to the multiplicities of the eigenvalues at those points. Then, let $J = V^{-1}AV$ be a JCF of A , then $f(A) := Vf(J)V^{-1}$ where

$$J = \begin{bmatrix} J_1 & & \\ & \ddots & \\ & & J_k \end{bmatrix}, \quad f(J_i) := \begin{bmatrix} f(\lambda_i) & f'(\lambda_i) & \dots & f^{(n_i-1)}(\lambda_i) \\ & \ddots & \ddots & \vdots \\ & & \ddots & f'(\lambda_i) \\ & & & f(\lambda_i) \end{bmatrix}.$$

- Covers all the previous cases we have considered.
- Quite difficult to use in practice.

The last definition is the more “general” — and allows to draw some conclusions on the algebraic structure.

- Matrix functions change the **eigenvalues** (resp. Jordan blocks) through the relation $\lambda \mapsto f(\lambda)$ (resp. $J_\lambda \mapsto f(J_\lambda)$).
- The **eigenvector** stay the same, and so do the Jordan chains.
- Note that the algebraic multiplicities might change, because f might not be injective (and therefore have $f(\lambda_1) = f(\lambda_2)$ with $\lambda_1 \neq \lambda_2$) — but the length and structure of the chains is preserved.

Another definition for holomorphic functions

There is a last useful definition that is valid for holomorphic function, and is at the basis of the **holomorphic functional calculus**.

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

where Γ is a path enclosing once the spectrum of A , with positive orientation (i.e., counter-clockwise). The equivalence is easy to prove, for instance, for diagonalizable matrices, using the **residue theorem**.

Approximating the action of matrix functions

Assume we are given a matrix A and a vector b : how do we approximate $x = f(A)b$?

- If $f(z) = e^z$ this finds application in **exponential integrators** and direct solution of ODEs. Indeed, if you have

$$\begin{cases} \dot{x} = Ax \\ x(0) = b \end{cases} \implies x(t) = e^{tA}b.$$

- A special case is given by $f(z) = z^{-1}$: we have the solution of a **linear system**:
 $x = A^{-1}b \iff Ax = b.$

How to efficiently compute such objects?

Small-scale case

If A is “small”, say at most 100×100 , we can compute $f(A)$ explicitly, and then compute directly $f(A)b$ by matrix-vector multiplication.

- We have good **specialized methods** for some functions (MATLAB functions `expm` for e^z , `logm` for $\log(z)$, `sqrtm` for \sqrt{z} , ...).
- Generic method may work well for other cases (MATLAB `funm`, based on **Schur-Parlett** algorithm – requires to know derivatives of $f(z)$).

What about larger matrices?

- In these cases, we usually know how to perform $v \mapsto Av$ efficiently.
- Sometimes, $v \mapsto (A - \sigma I)^{-1}v$ is available at a reasonable cost.

Projection methods

To handle a large scale A we can use **projection methods**. General idea:

- Construct a “good” **projection space** \mathcal{U} , spanned by the orthogonal matrix $Q \in \mathbb{R}^{n \times \ell}$.
- Construct $A_\ell := Q_\ell^T A Q_\ell \in \mathbb{R}^{k \times k}$.
- Compute $f(A_\ell)$ (a small matrix function!).
- Approximate $f(A)b \approx Q_\ell f(A_\ell) Q_\ell^T b$.

Exercise

Assume $A = UU^T$, with $U \in \mathbb{R}^{n \times \ell}$ has rank at most ℓ ; then, for every $f(z)$ with $f(0) = 0$, we have

$$f(A) = Q_\ell f(A_\ell) Q_\ell^T, \quad Q_\ell \text{ spans } U.$$

What happens if $f(0) \neq 0$?

A closer look at this approach

Note that if the previous approach works, then

$$f(A)b \approx Q_\ell f(A_\ell) Q_\ell^T b$$

means that the action of $f(A)$ on b is **well-approximated by a low-rank matrix** $Q_\ell f(A_\ell) Q_\ell^T$. How to choose this subspace?

- If we understand this, we might be able to say something about the low-rank approximation problem more in general!
- Several choices are possible for the subspace spanned by Q_ℓ . We will concentrate on **Krylov subspaces**.

Definition

Given a square matrix A and a vector b , we define the *polynomial Krylov subspace spanned by A and b* as follows:

$$\mathcal{K}_\ell(A, b) := \text{span}\{b, Ab, \dots, A^{\ell-1}b\}$$

- Generically, we expect it to have dimension ℓ .
- Clearly, $\mathcal{K}_\ell(A, b) \subseteq \mathcal{K}_{\ell+1}(A, b)$.
- For every polynomial $p(z)$ of degree at most $\ell - 1$, we have $p(A)b \in \mathcal{K}_\ell(A, b)$.
Indeed, we can easily verify:

$$\mathcal{K}_\ell(A, b) := \{p(A)b \mid p(z) \text{ with } \deg \leq \ell - 1\}.$$

A link with polynomial approximation

Since Krylov subspaces contain polynomials in A – we may use them to approximate $f(A)b$ if $f(z) \approx p(z)$, for a (low degree) polynomial.

Theorem (Weierstrass)

Let Ω be a compact subset of \mathbb{R} . Then, every continuous function can be approximated uniformly well with polynomials. That is, given $f(z)$ there exists a sequence of polynomials $p_k(z)$ such that

$$\lim_{k \rightarrow \infty} \|f(z) - p_k(z)\|_{\infty} = 0$$

A related result holds in the complex plane as well:

Theorem (Mergelyan)

Let K a compact set such that $\mathbb{C} \setminus K$ is connected. Then, every continuous function defined on K such that its restriction to the interior part of K is holomorphic can be uniformly approximated by polynomials.

A link with polynomial approximation

- Since $f(z) \approx p(z)$, can we claim that $f(A) \approx p(A)$?
- If that's true, then $\mathcal{K}_\ell(A, b)$ contains a good approximation of $f(A)b$, since it contains $p(A)b$; how to compute it in practice?
- Numerically, Weierstrass' theorem is not as strong as it might seem ... why?

A link with polynomial approximation

- Since $f(z) \approx p(z)$, can we claim that $f(A) \approx p(A)$?
- If that's true, then $\mathcal{K}_\ell(A, b)$ contains a good approximation of $f(A)b$, since it contains $p(A)b$; how to compute it in practice?
- Numerically, Weierstrass' theorem is not as strong as it might seem ... why?

The main limitation is that we don't know anything about the **speed of convergence**.

Building a basis for the polynomial Krylov subspace

Recall that, for our projection method, we need an orthogonal basis of $\mathcal{K}_\ell(A, b)$. How to retrieve it? We use the Arnoldi iteration:

- We can choose Q_ℓ its first $\ell - 1$ cols are equal to $Q_{\ell-1}$.
- Note that if Q_ℓ spans $\mathcal{K}_\ell(A, b)$, then $AQ_\ell \subseteq \mathcal{K}_{\ell+1}(A, b)$.

Building a basis for the polynomial Krylov subspace

Recall that, for our projection method, we need an orthogonal basis of $\mathcal{K}_\ell(A, b)$. How to retrieve it? We use the Arnoldi iteration:

- We can choose Q_ℓ its first $\ell - 1$ cols are equal to $Q_{\ell-1}$.
- Note that if Q_ℓ spans $\mathcal{K}_\ell(A, b)$, then $AQ_\ell \subseteq \mathcal{K}_{\ell+1}(A, b)$. Indeed,

$$v \in \mathcal{K}_\ell(A, b) \iff v = p(A)b, \quad \deg(p) \leq \ell - 1.$$

Then, $Av = Ap(A)b = q(A)b$ where $\deg q \leq \ell$, and therefore $Av \in \mathcal{K}_{\ell+1}(A, b)$.

- This implies that the ℓ columns of AQ_ℓ can be written as linear combination of the $\ell + 1$ columns of $Q_{\ell+1}$; in matrix terms:

$$AQ_\ell = Q_{\ell+1}H_\ell, \quad H_\ell := \begin{bmatrix} \times & \dots & \times \\ \times & & \vdots \\ & \ddots & \times \\ & & \times \end{bmatrix}$$

Arnoldi iteration

The matrices Q_ℓ, H_ℓ can be built incrementally as ℓ increases.

- If $\ell = 0$, we have

$$H_0 = 0_{1 \times 0}, \quad Q_0 = 0_{n \times 0}, \quad Q_1 := \frac{b}{\|b\|_2}$$

- If we know the Arnoldi relation for ℓ , we can construct $Q_{\ell+2}$ and $H_{\ell+1}$ imposing

$$AQ_{\ell+1} = Q_{\ell+2}H_{\ell+1} = \left[\begin{array}{c|c} Q_{\ell+1} & v_{\ell+2} \end{array} \right] \left[\begin{array}{c|c} H_\ell & w_{\ell+1} \\ \hline & \beta_{\ell+1} \end{array} \right].$$

- $v_{\ell+2}$ and $w_{\ell+1}$ are computed by **Gram-Schmidt reorthogonalization** – as if $\beta_{\ell+1}$.

Arnoldi iteration

The matrices Q_ℓ, H_ℓ can be built incrementally as ℓ increases.

- If $\ell = 0$, we have

$$H_0 = 0_{1 \times 0}, \quad Q_0 = 0_{n \times 0}, \quad Q_1 := \frac{b}{\|b\|_2}$$

- If we know the Arnoldi relation for ℓ , we can construct $Q_{\ell+2}$ and $H_{\ell+1}$ imposing

$$AQ_{\ell+1} = Q_{\ell+2}H_{\ell+1} = \left[\begin{array}{c|c} Q_{\ell+1} & v_{\ell+2} \end{array} \right] \left[\begin{array}{c|c} H_\ell & w_{\ell+1} \\ \hline & \beta_{\ell+1} \end{array} \right].$$

- $v_{\ell+2}$ and $w_{\ell+1}$ are computed by **Gram-Schmidt reorthogonalization** – as if $\beta_{\ell+1}$.

We can checkout a few examples in MATLAB.

Obtaining the projection

Right now we have Q_ℓ , the basis of the space. But H_ℓ allows to extract $A_\ell = Q_\ell^T A Q_\ell$ easily. Indeed,

$$A Q_\ell = Q_{\ell+1} H_\ell \implies Q_\ell^T A Q_\ell = (Q_\ell^T Q_{\ell+1}) H_\ell.$$

- The columns of Q_ℓ and $Q_{\ell+1}$ are orthogonal, so we have

$$Q_\ell^T Q_{\ell+1} := [\gamma_{ij}]_{\substack{i=1,\dots,\ell \\ j=1,\dots,\ell+1}} = \begin{bmatrix} I_\ell & 0_{\ell \times 1} \end{bmatrix}.$$

- Putting the pieces together yields:

$$H_{\ell+1} = \begin{bmatrix} A_\ell \\ \gamma_{\ell+1,\ell} \mathbf{e}_\ell^T \end{bmatrix}$$

A crucial property of Krylov subspace is the following.

Theorem (Exactness)

If A be any square matrix, b a vector Q_ℓ spanning $\mathcal{K}(A, b)$ and

$$v_\ell := Q_\ell f(Q_\ell^T A Q_\ell) Q_\ell^T b.$$

If $f(z)$ is a polynomial of degree at most $\ell - 1$, then $v_\ell = f(A)b$.

Accuracy of the projection method

Our setting:

- We consider a matrix A and a vector b , and build a basis Q_ℓ of $\mathcal{K}_\ell(A, b)$, and $A_\ell := Q_\ell^T A Q_\ell$.
- We approximate $f(A)b \approx Q_\ell f(A_\ell) Q_\ell^T$.

Accuracy of the projection method

Our setting:

- We consider a matrix A and a vector b , and build a basis Q_ℓ of $\mathcal{K}_\ell(A, b)$, and $A_\ell := Q_\ell^T A Q_\ell$.
- We approximate $f(A)b \approx Q_\ell f(A_\ell) Q_\ell^T b$.

Theorem

Let A be a square matrix, Q_ℓ orthogonal spanning $\mathcal{K}_\ell(A, b)$. Then,

$$\|f(A)b - Q_\ell f(A_\ell) Q_\ell^T b\|_2 \leq \min_{\deg p \leq \ell-1} \left(\|r(A)\|_2 + \|r(A_\ell)\|_2 \right) \cdot \|b\|_2,$$

where $r(z) = f(z) - p(z)$.

Accuracy of the projection method

Our setting:

- We consider a matrix A and a vector b , and build a basis Q_ℓ of $\mathcal{K}_\ell(A, b)$, and $A_\ell := Q_\ell^T A Q_\ell$.
- We approximate $f(A)b \approx Q_\ell f(A_\ell) Q_\ell^T b$.

Theorem

Let A be a square matrix, Q_ℓ orthogonal spanning $\mathcal{K}_\ell(A, b)$. Then,

$$\|f(A)b - Q_\ell f(A_\ell) Q_\ell^T b\|_2 \leq \min_{\deg p \leq \ell-1} (\|r(A)\|_2 + \|r(A_\ell)\|_2) \cdot \|b\|_2,$$

where $r(z) = f(z) - p(z)$.

If A is symmetric and there exists $p(z)$ with $|p(z) - f(z)| \leq \epsilon$, then

$$\|f(A)b - Q_\ell f(A_\ell) Q_\ell^T b\|_2 \leq 2\|b\|_2 \epsilon.$$

The exactness property is key to proving the previous result. We have, by setting $A_\ell = Q_\ell^T A Q_\ell$,

$$\begin{aligned}
 & \|f(A)b - Q_\ell f(A_\ell) Q_\ell^T b\|_2 \\
 & \leq \|f(A)b - \underbrace{p(A)b + Q_\ell p(A_\ell) Q_\ell^T b}_{=0, \text{ thanks to the exactness}} - Q_\ell f(A_\ell) Q_\ell^T b\|_2 \\
 & \leq \|f(A)b - p(A)b\| + \|Q_\ell(f(A_\ell) - p(A_\ell)) Q_\ell^T b\|_2 \\
 & = \|r(A)b\|_2 + \|r(A_\ell) Q_\ell^T b\|_2,
 \end{aligned}$$

where $r(z) = f(z) - p(z)$. As a last step, we can take the vector b and $Q_\ell^T b$ out of the norms — and both are smaller than $\|b\|_2$ in norm:

$$\|f(A)b - Q_\ell f(A_\ell) Q_\ell^T b\|_2 \leq \left(\|r(A)\|_2 + \|r(A_\ell)\|_2 \right) \cdot \|b\|_2.$$

Important take-home messages

Several nice features:

- The projection method is **quasi-optimal** – as the recovered approximation is almost as good as the best polynomial approximation.
- However, there is **no need to know the approximation a priori** — the quasi-optimal result is automatically extracted!

... and some limitations as well:

- For many interesting functions, polynomial approximation **converge slowly**.
- For instance, $f(z) = 1/z$ or $f(z) = z^\alpha$ with $-1 < \alpha < 1$.
- For this case, one can resort to **rational functions**, instead of polynomials (more on this later!).

Making the bound explicit

The measured accuracy depends on $\|r(A)\|_2$ and $\|r(A_m)\|_2$ – can we make these numbers more “explicit”?

- Intuitively, if $|f(z) - p(z)| \leq \epsilon$ on the set of eigenvalues is small, then $r(A)$ is small as well.

Making the bound explicit

The measured accuracy depends on $\|r(A)\|_2$ and $\|r(A_m)\|_2$ – can we make these numbers more “explicit”?

- Intuitively, if $|f(z) - p(z)| \leq \epsilon$ on the set of eigenvalues is small, then $r(A)$ is small as well.
- Sometime intuition can be misleading:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad r(z) = \gamma(z - 1), \quad \|r(A)\|_2 = ?$$

Making the bound explicit

The measured accuracy depends on $\|r(A)\|_2$ and $\|r(A_m)\|_2$ – can we make these numbers more “explicit”?

- Intuitively, if $|f(z) - p(z)| \leq \epsilon$ on the set of eigenvalues is small, then $r(A)$ is small as well.
- Sometime intuition can be misleading:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad r(z) = \gamma(z - 1), \quad \|r(A)\|_2 = ?$$

- A direct computation yields $\|r(A)\|_2 = |\gamma|$. $r(z) \equiv 0$ on the spectrum, yet $\|r(A)\|_2$ can be arbitrarily large!

As we have seen, the spectrum might not describe (in general) the norm of $\|r(A)\|_2$.

- There are classes of matrices for which this is true.
- We define **normal matrices** as matrices which commute with their transpose, i.e., for which we have:

$$AA^T = A^T A$$

- Examples: **symmetric** (resp. Hermitian) or **orthogonal** (resp. unitary) matrices.

Theorem

An $n \times n$ matrix is normal if and only if it can be diagonalized by orthogonal / unitary matrices. That is, A normal iff there exists Q with $Q^T Q = I$ such that

$$Q^T A Q = D, \quad D \text{ diagonal.}$$

- It is immediate to check that a matrix $A = Q^T D Q$ is normal.
- On the other hand, if A normal then $Q^T A Q$ is normal; then, we can compute the Schur form $T = Q^T A Q$ — and upper triangular normal matrices are diagonal (easy, induction).

If A is normal, then we have

$$f(A) = Q^T f(D) Q \implies \|f(A)\|_2 = \|f(D)\|_2 = \max_{j=1, \dots, n} |f(\lambda_k)|,$$

where λ_k are the eigenvalues of A .

- For normal matrices, the spectrum tell us everything on functions of matrices.
- Evaluating the function on the set $\Lambda(A)$ gives us a bound for the norm of $f(A)$.
- Can this property be extended to more general matrices?

Definition

A set Ω is a K -spectral set for A if, for any matrix function $f(z)$, we have

$$\|f(A)\|_2 \leq K \max_{z \in \Omega} |f(z)|.$$

- Example: For normal matrices, $\Omega := \Lambda(A)$ is a 1-spectral set.
- Idea: we can try to find suitable sets, larger than $\Lambda(A)$, such that this property continue to hold.

Theorem

If A is diagonalizable, let V such that $V^{-1}AV = D$. Then,

$$\|f(A)\|_2 \leq \kappa_2(V) \cdot \max_{j=1,\dots,n} |f(\lambda_j)|.$$

Then, $\Lambda(A)$ is a $\kappa_2(V)$ -spectral set for A .

- Unfortunately, if $\kappa_2(V)$ is large this result can become almost meaningless.
- A similar result can be stated based on the Jordan Canonical Form (even though it's not very useful in practice) – how would it look like?

Looking at contour integrals

Remember the integral definition of matrix functions:

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

This can be used to construct K -spectral sets by leveraging the **pseudospectrum**.

Looking at contour integrals

Remember the integral definition of matrix functions:

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

This can be used to construct K -spectral sets by leveraging the **pseudospectrum**.

- Let us denote by $R_A(z) := (zI - A)^{-1}$ the **resolvent** of A .
- Clearly, by a slight abuse of notation, $\|R_A(z)\|_2 = \infty$ is and only if z is an eigenvalue of A .

Looking at contour integrals

Remember the integral definition of matrix functions:

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

This can be used to construct K -spectral sets by leveraging the **pseudospectrum**.

- Let us denote by $R_A(z) := (zI - A)^{-1}$ the **resolvent** of A .
- Clearly, by a slight abuse of notation, $\|R_A(z)\|_2 = \infty$ if and only if z is an eigenvalue of A .

This connection has been used to give a more “robust” definition of spectrum – the ϵ -pseudospectrum.

Definition

The set $\Lambda_\epsilon(A)$, defined as follows

$$\Lambda_\epsilon(A) := \{z \in \mathbb{C} \mid \|zI - A\| \geq \epsilon^{-1}\},$$

is called the ϵ -pseudospectrum of A .

Definition

The set $\Lambda_\epsilon(A)$, defined as follows

$$\Lambda_\epsilon(A) := \{z \in \mathbb{C} \mid \|zI - A\| \geq \epsilon^{-1}\},$$

is called the ϵ -pseudospectrum of A .

- This is a “continuous” extension of the spectrum. Clearly, it always include the spectrum: $\Lambda(A) \subseteq \Lambda_\epsilon(A)$.
- Encodes the idea of “robust eigenvalues”.
- Natural application to stability analysis.
- Might work with any norm (for us: mainly $\|\cdot\|_2$).

Building a K -spectral set using the pseudospectrum

Let Γ_ϵ be a positive orientation contour of $\Lambda_\epsilon(A)$. Then,

$$f(A) = \frac{1}{2\pi\mathbf{i}} \int_{\Gamma_\epsilon} f(z)(zI - A)^{-1} dz \implies \|f(A)\|_2 \leq \frac{L(\Gamma_\epsilon)}{2\pi\epsilon} \cdot \max_{z \in \Lambda_\epsilon(A)} |f(z)|,$$

where $L(\Gamma_\epsilon)$ is the length of the boundary of $\Lambda_\epsilon(A)$. In particular, $\Lambda_\epsilon(A)$ is a K -spectral set for $K := \frac{L(\Gamma_\epsilon)}{2\pi\epsilon}$.

- This construction is valid for any ϵ : choosing the right value can give you good bounds.
- Choosing the optimal ϵ can be tricky, as describing (and computing the length of the boundary) $\Lambda_\epsilon(A)$.

See: `example_pseudospectrum.m`

There are quite obvious difficulties with replacing the spectrum with the pseudospectrum:

1. The first is often available from theoretical considerations; the second, instead, is usually much harder to determine.
2. In particular, given a matrix A , how do we actually compute $\Lambda_\epsilon A$?
3. The choice of ϵ can be tricky.

We briefly discuss point 2., and we concentrate on the case where $\|\cdot\| = \|\cdot\|_2$.

Strategy 1: Poor's man pseudospectrum

We may go back at the original definition, and consider:

$$\Lambda_\epsilon(A) := \bigcup_{\|\delta A\|_2 \leq \epsilon} \Lambda(A + \delta A)$$

Prototype algorithm:

- Choose a number k sufficiently large.
- Compute k random matrices δA_j for $j = 1, \dots, k$.
- Scale these matrices to have norm ϵ .
- For each of them, compute the eigenvalues of the perturbed matrix $A + \delta A_j$.
- Plot everything together.

Problems: quite costly, not necessarily very descriptive.

Bounding $\Lambda_\epsilon(A)$

We can determine a region including $\Lambda_\epsilon(A)$ in view of the following observation: If $|z| > \|A\|_2$, then

$$(zI - A)^{-1} = z^{-1}(I - z^{-1}A)^{-1} \implies \|(zI - A)^{-1}\| \leq \frac{1}{|z|} \cdot \frac{1}{1 - |z|^{-1}\|A\|_2} = \frac{1}{|z| - \|A\|_2}.$$

This allows to prove the following:

Lemma

The pseudospectrum $\Lambda_\epsilon(A)$ satisfies

$$\Lambda_\epsilon(A) \subseteq \{|z| \leq \|A\|_2 + \epsilon\}$$

Proof.

If $|z| > \|A\|_2 + \epsilon$ then

$$\|(zI - A)^{-1}\|_2 \leq \frac{1}{|z| - \|A\|_2} < \epsilon^{-1},$$

so such z cannot belong to $\Lambda_\epsilon(A)$.

Strategy 2: grid discretization

Since we have an inclusion result for $\Lambda_\epsilon(A)$, we can:

- Determine a box in the complex plane where the pseudospectrum is included for different values of ϵ .
- Evaluate the function $\|(zI - A)^{-1}\|_2$ on that grid.
- Construct the contour plot starting from the above data.

Main issue: the evaluation of $\|(zI - A)^{-1}\|_2$ is cubic if using the straightforward algorithm. That amounts to a total cost $\mathcal{O}(kn^3)$ where:

- k is the number of sampling points.
- n is the size of the matrix A .

Strategy 2: grid discretization (+ Schur form)

How can we make the computation of $\|(zI - A)^{-1}\|_2$ more efficient?

- One may estimate the norm by a power iteration:

$$x_{k+1} = (zI - A)^{-1}(zI - A)^{-1}x_k,$$

for some starting vector x_0 . Then, $\|x_{k+1}\|_2/\|x_k\|_2 \rightarrow \|(zI - A)^{-1}\|_2$.

- This would still cost $\mathcal{O}(n^3)$ per grid point in general.
- We can precompute the **Schur factorization** of A :

$$A = QTQ^*, \quad Q \text{ unitary.}$$

In particular, $zI - A = zI - QTQ^*$ and this implies that

$$\|(zI - A)^{-1}\|_2 = \|(zI - T)^{-1}\|_2$$

Luckily, triangular linear systems are cheaper to solve (only $\mathcal{O}(n^2)$). Therefore, we have total complexity:

$$\mathcal{O}(n^3 + kn^2),$$

where k is the number of grid points.

Summary (pseudospectrum)

- The previous algorithm is the one included in `eigtool`.
- It works fairly well for small to medium size matrices.
- The large scale case is more complicated: one can couple similar ideas with running inverse Lanczos iteration: difficult to recycle at different values of z .
- This process clearly shows the difficulty in computing the constant L_ϵ , especially a priori.

There is another alternative to construct K -spectral sets using the **field of values**.

Definition

Let A be any square matrix; the *field of values of A* , denoted by $\mathcal{W}(A)$, is defined as follows:

$$\mathcal{W}(A) := \{x^H A x \mid \|x\|_2 = 1\}$$

- Clearly, it is a compact set, included in $\overline{B}(0, \|A\|_2)$.
- If A is normal, then the field of values is the convex hull of its eigenvalues.

Theorem (Crouzeix)

The spectral set is a $(1 + \sqrt{2})$ -spectral set³ with respect to the 2-norm. That is,

$$\|f(A)\|_2 \leq (1 + \sqrt{2}) \cdot \max_{z \in \mathcal{W}(A)} |f(z)|.$$

³It is conjectured that the optimal constant is 2, instead of $1 + \sqrt{2}$.

Some properties

- $\mathcal{W}(\alpha A) = \alpha \mathcal{W}(A)$.
- $\mathcal{W}(A + B) \subseteq \mathcal{W}(A) + \mathcal{W}(B)$.
- $\mathcal{W}(A) \subseteq B(0, \|A\|)$ for every induced norm $\|\cdot\|$.
- The field of values is convex (Hausdorff-Toeplitz theorem).

See: `example_fov.m`