Matrix functions

Nathan Rousselot

August 18, 2023

1 Introduction

In this document we introduce the notion of matrix functions. Say we have a function $f : \mathbb{C} \to \mathbb{C}$, then we can define the function f on a matrix $\mathbf{A}$ as follows: $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$. In the following, we will first provide some theoretical background on matrix functions. Then, we will describe the numerical issues coming with manipulating matrix functions. Finally, we will provide algorithm for efficient computation of matrix fuctions.

2 Theoretical background

2.1 Natural Definition

Polynomial functions

Let $p : \mathbb{C} \to \mathbb{C}$ be a polynomial function of degree d:

$$p(t) = \sum_{k=0}^{d} c_k t^k \tag{2.1}$$

Then, considering a matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$, and posing $\mathbf{A}^0 = I_n$, we can define the polynomial function $p : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ on a matrix $\mathbf{A}$ as follows:

$$p(\mathbf{A}) = \sum_{k=0}^{d} c_k \mathbf{A}^k \tag{2.2}$$

Rational functions

Let $f : \mathbb{C} \to \mathbb{C}$ be a rational function of the form:

$$f(t) := \frac{p(t)}{q(t)} \tag{2.3}$$

It is not immediate how one would approach this function with a matrix. We want to define

$$f(\mathbf{A}) := q(\mathbf{A})^{-1} p(\mathbf{A}) \tag{2.4}$$

However, this is not well defined if $q(\mathbf{A})$ is singular. In other words, we need to make sure that $q(\mathbf{A})$ is invertible. This is the case if and only if $q(\lambda) \neq 0$ for all eigenvalues λ of $\mathbf{A}$. This is a very strong condition, and it is not always possible to find a rational function f such that $q(\lambda) \neq 0$ for all eigenvalues λ of $\mathbf{A}$. We note that a choice in notation has been made in equation 2.4, the other notation is still valid.

Lemma 1. *Let* $\mathbf{A} \in \mathbb{C}^{n\times n}$*, and let* $p : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ *and* $q : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$*, two matrix polynomials. Then,*

$$q(\mathbf{A})p(\mathbf{A}) = p(\mathbf{A})q(\mathbf{A}) \tag{2.5}$$

Proof. Let $f(\mathbf{A}) := q(\mathbf{A})p(\mathbf{A})$, and $g(\mathbf{A}) = p(\mathbf{A})q(\mathbf{A})$. Then

$$\begin{aligned} f(\mathbf{A}) = \left(\sum_{k=0}^{d} c_k \mathbf{A}^k\right)\left(\sum_{j=0}^{m} b_j \mathbf{A}^j\right) \\ = \sum_{k=0}^{d}\sum_{j=0}^{m} c_k b_j \mathbf{A}^{j+k} \\ = \sum_{j=0}^{m}\sum_{k=0}^{d} b_j c_k \mathbf{A}^{k+j} \\ = \left(\sum_{j=0}^{m} b_j \mathbf{A}^j\right)\left(\sum_{k=0}^{d} c_k \mathbf{A}^k\right) = g(\mathbf{A}) \end{aligned}$$

□

Theorem 1. *Let* $\mathbf{A} \in \mathbb{C}^{n\times n}$*, and let* $p : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ *and* $q : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$*, two matrix polynomials such that* $q(\mathbf{A})$ *is non-singular. Then,*

$$q(\mathbf{A})^{-1}p(\mathbf{A}) = p(\mathbf{A})q(\mathbf{A})^{-1} \tag{2.6}$$

Proof. Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a matrix such that $q(\mathbf{A})$ is non-singular

$$q(\mathbf{A})^{-1}p(\mathbf{A}) = q(\mathbf{A})^{-1}p(\mathbf{A})q(\mathbf{A})q(\mathbf{A})^{-1}$$

Using Lemma 1

$$\begin{aligned} q(\mathbf{A})^{-1}p(\mathbf{A})q(\mathbf{A})q(\mathbf{A})^{-1} &= q(\mathbf{A})^{-1}q(\mathbf{A})p(\mathbf{A})q(\mathbf{A})^{-1} \\ &= p(\mathbf{A})q(\mathbf{A})^{-1} \end{aligned}$$

□

Power Series

Let $f : \mathbb{C} \to \mathbb{C}$ be a function that can be expressed as a power series:

$$f(t) = \sum_{k=0}^{\infty} c_k t^k \tag{2.7}$$

Then, considering a matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$, and posing $\mathbf{A}^0 = I_n$, we can define the power series function $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ on a matrix $\mathbf{A}$ as follows:

$$f(\mathbf{A}) = \sum_{k=0}^{\infty} c_k \mathbf{A}^k \tag{2.8}$$

In the scalar case, we know that the power series converges if $|t| < r$, where r is the radius of convergence. Obviously this translates in the matrix power series

Theorem 2. *Let $f : \mathbb{C}^{n\times n} \to \mathbb{C}^{n\times n}$ be a matrix power series. Then, the series converges if and only if $\rho(\mathbf{A}) < r$, where $\rho(\mathbf{A})$ is the spectral radius of $\mathbf{A}$, and r is the radius of convergence of the scalar power series.*

Proof is provided in Frommer and Simoncini 2008. In the case of a finite-order Laurent Series, *i.e*:

$$f(t) = \sum_{k=-d}^{d} c_k t^k \tag{2.9}$$

For the matrix case, we need to ensure convergence (similarly to power series), but also ensure existance of the inverse, as Laurent series do have negative powers. If both of those conditions are satisfied, we can write the Laurent series as a matrix function:

$$f(\mathbf{A}) = \sum_{k=-d}^{d} c_k \mathbf{A}^k \tag{2.10}$$

2.2 Spectrum-Based Definition

Diagonalizable Matrices

Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a diagonalizable matrix. That means that there exists a matrix $\mathbf{V} \in \mathbb{C}^{n\times n}$ such that $\mathbf{V}$ is invertible, and $\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$, where $\mathbf{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\mathbf{A}$:

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} \tag{2.11}$$

Theorem 3. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a diagonalizable matrix. Then we can define the function $f(\mathbf{A})$ as:*

$$f(\mathbf{A}) := \mathbf{V} f(\mathbf{\Lambda})\mathbf{V}^{-1} \tag{2.12}$$

with

$$f(\mathbf{\Lambda}) = \begin{bmatrix} f(\lambda_1) & 0 & \cdots & 0 \\ 0 & f(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\lambda_n) \end{bmatrix} \tag{2.13}$$

This property is very handy, as it allows us to compute matrix functions by simply applying the function to the eigenvalues of the matrix. Computationally, this avoids inverting matrices, and lots of matrix products. However, this property is only valid for diagonalizable matrices. This property puts constraints on $f(\mathbf{A})$, as its eigenvectors must form a basis $\mathbf{F}^n$. Another more practical constraint, that is sufficient but not necessary, is if $\mathbf{A}$ is a full rank matrix, then it is diagonalizable.

Defective Matrices

In some cases, the matrix $\mathbf{A}$ is not diagonalizable, that means the sum of the dimensions of the eigenspaces is less than n, we call that a *Defective Matrix.* In that case, we can generalize the principle of diagonalization using the Jordan canonical form of $\mathbf{A}$:

$$\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1} \tag{2.14}$$

where $\mathbf{J}$ is a Jordan matrix, and $\mathbf{V}$ is a matrix containing the generalized eigenvectors of $\mathbf{A}$. The Jordan matrix is a block diagonal matrix, where each block is a Jordan block. A Jordan block is a matrix of the form:

$$\mathbf{J}_k(\lambda) = \begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda & 1 \\ 0 & 0 & \cdots & 0 & \lambda \end{bmatrix} \tag{2.15}$$

Theorem 4. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a defective matrix. Then we can define the function $f(\mathbf{A})$ as:*

$$f(\mathbf{A}) := \mathbf{V} f(\mathbf{J}) \mathbf{V}^{-1} \tag{2.16}$$

with

$$f(\mathbf{J}) = \begin{bmatrix} f(\mathbf{J}_1(\lambda_1)) & 0 & \cdots & 0 \\ 0 & f(\mathbf{J}_2(\lambda_2)) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\mathbf{J}_k(\lambda_k)) \end{bmatrix} \tag{2.17}$$

and

$$f(\mathbf{J}_i(\lambda_i)) = \begin{bmatrix} f(\lambda_i) & f'(\lambda_i) & \frac{f''(\lambda_i)}{2!} & \cdots & \frac{f^{(k-1)}(\lambda_i)}{(k-1)!} \\ 0 & f(\lambda_i) & f'(\lambda_i) & \cdots & \frac{f^{(k-2)}(\lambda_i)}{(k-2)!} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\lambda_i) & f'(\lambda_i) \\ 0 & 0 & \cdots & 0 & f(\lambda_i) \end{bmatrix} \tag{2.18}$$

Obviously, both definitions of matrix functions, based on diagonalization and Jordan canonical form, presuppose that the spectral radius of $\mathbf{A}$, $\rho(\mathbf{A})$, is less than r, the radius of convergence.

2.3 Interpolation-based definition

Interestingly, in this section we will show that for any $\mathbf{A} \in \mathbb{C}^{n\times n}$ and any sufficiently differentiable f, we can find a polynomial p such that $f(\mathbf{A}) = p(\mathbf{A})$. First, let us observe from previous sections that only the eigenvalues of $\mathbf{A}$ are actually important for matrix polynomials. Also recall that every matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ with spectrum $\{\lambda_1, \ldots, \lambda_n\}$ has a minimal polynomial $\phi_{\mathbf{A}}$ given by

$$\phi_{\mathbf{A}}(t) := \prod_{i=1}^{k} (t - \lambda_k)^{n_i} \tag{2.19}$$

which is the unique monic minimal degree ($\text{degr}(\phi_{\mathbf{A}}) = n_1 + \cdots + n_k \leq n$) polynomial such that $\phi_{\mathbf{A}}(\mathbf{A}) = 0$.

Theorem 5. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a matrix with eigenvalues $\{\lambda_1, \ldots, \lambda_k\}$, and minimal polynomial given by equation 2.19. Then for any two polynomials p_1, p_2 we have that $p_1(\mathbf{A}) = p_2(\mathbf{A})$ if and only if*

$$\forall j \in \{1, \ldots, k\} : \forall i \in \{0, \ldots, n_k - 1\} : p_1^{(i)}(\lambda_j) = p_2^{(i)}(\lambda_j).$$

Proof. Let $\mathbf{A} \in \mathbb{C}^{n\times n}$, with spectrum $\{\lambda_1, \dots, \lambda_k\}$, and two polynomials p_1 and p_2 such that $p_1(\mathbf{A}) = p_2(\mathbf{A})$. Let us define q such that

$$q := p_1 - p_2$$

Then, $q(\mathbf{A}) = 0$, and is thus divisible by $\phi_{\mathbf{A}}$, meaning that

$$\forall j \in \{1, \dots, k\} : \forall i \in \{0, \dots, n_k - 1\}, q(\lambda_j) = 0 \Rightarrow p_1^{(i)}(\lambda_j) = p_1^{(i)}(\lambda_i)$$

Similarly, consider two polynomials p_1 and p_2 such that

$$\forall j \in \{1, \dots, k\} : \forall i \in \{0, \dots, n_k - 1\}, p_1^{(i)}(\lambda_j) = p_2^{(i)}(\lambda_i)$$

For $i = 0$, $q := p_1 - p_2 = 0$ on the spectrum of $\mathbf{A}$, and is then divisible by $\phi_{\mathbf{A}}$. Then

$$q = K\phi_{\mathbf{A}}$$

with K a polynomial. Then, $q(\mathbf{A}) = K(\mathbf{A})\phi_{\mathbf{A}}(\mathbf{A}) = 0$ since by definition, $\phi_{\mathbf{A}}(\mathbf{A}) = 0$. And thus, $p_1(\mathbf{A}) = p_2(\mathbf{A})$.
From this reasoning, we conclude that

$$\begin{aligned} & p_1(\mathbf{A}) = p_2(\mathbf{A}) \\ \Leftrightarrow \forall j \in \{1, \dots, k\} : \forall i \in \{0, \dots, n_k - 1\} : & \; p_1^{(i)}(\lambda_j) = p_2^{(i)}(\lambda_j) \end{aligned}$$

□

In theorem 5, the conditions involve the evaluation of the polynomials p_1 and p_2 and their derivatives up to order $n_k - 1$ at the eigenvalues of A.
However, when the spectrum of A is simple, each eigenvalue λ_j is of multiplicity $n_j = 1$. This means that there are no higher order terms corresponding to these eigenvalues in the minimal polynomial, or in other words, there are no repeated roots. Consequently, there is no need to consider the derivatives of the polynomials p_1 and p_2 because there are no repeated roots for the polynomials to "match up" with. Therefore, in this simpler case, we only need to check that the polynomials p_1 and p_2 agree at the eigenvalues of A. In formal terms, the condition becomes:

Corollary 5.1. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a matrix with eigenvalues $\{\lambda_1, \dots, \lambda_k\}$, and minimal polynomial given by equation 2.19. If $\mathbf{A}$ has simple spectrum, then for any two polynomials p_1, p_2 we have that $p_1(\mathbf{A}) = p_2(\mathbf{A})$ if and only if*

$$\forall j \in \{1, \dots, k\} : p_1(\lambda_j) = p_2(\lambda_j).$$

From this corollary, and from theorem 5, we can confirm our earlier statement : only the spectrum of $\mathbf{A}$ is important for matrix polynomials. More importantly, we observe that $p(\mathbf{A})$ is uniquely defined by its values on the spectrum of $\mathbf{A}$. It seems then natural to extend this definition to any function f.

Definition 1. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a matrix with minimal polynomial given as in equation 2.19, and let f be a function that is at least* $\max_k\{n_k - 1\}$ *times differentiable. Say p is its* $(n_1, \dots, n_k)$-Hermite interpolant *i.e. the polynomial satisfying*

$$\forall j \in \{1, \dots, k\} : \forall i \in \{0, \dots, n_k - 1\} : p^{(i)}(\lambda_j) = f^{(i)}(\lambda_j)$$

of minimal degree. Then we define $f(\mathbf{A}) = p(\mathbf{A})$.

3 The matrix-vector product $f(\mathbf{A})\mathbf{b}$

3.1 Introduction

To motivate the need for a matrix-vector product, we will consider a matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$. Let us consider the matrix exponential, such as :

$$e^{\mathbf{A}} = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k}{k!} \tag{3.20}$$

Not only does this computation is very heavy (see section 5.1 for computation strategies), but it also affects the structure of the matrix. Say for example, we have the following laplacian matrix:

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & -1 & 2 & -1 \\ 0 & 0 & \dots & 0 & 0 & -1 & 2 \end{bmatrix}$$

Obviously, $\mathbf{A}$ is a sparse matrix, and also a rank-structured one : it is a tridiagonal matrix. However, by computing $e^{\mathbf{A}}$, one will get a dense matrix. Storing a dense matrix often becomes challenging as the dimension of the problem grows. Besides storing, computing such a dense matrix is also an issue. However, the matrix-vector product $f(\mathbf{A})\mathbf{b}$ can be stored and computed efficiently.

3.2 The Method

As motivated by 3.1, when encountering specific structure in $\mathbf{A}$, such as sparsity, there is a lot to gain if we can store $f(\mathbf{A})\mathbf{b}$. In this section we will work towards a way to evaluate $f(\mathbf{A})\mathbf{b}$ in an intuitive way, thanks to Arnoldi method.

3.2.1 Formal Definitions

Firstly, we will define the notion of $\phi_{\mathbf{A},\mathbf{b}}$, i.e. the minimal polynomial of $\mathbf{A}$ with respect to the vector $\mathbf{b}$. This is simply the polynomial

$$\phi_{\mathbf{A},\mathbf{b}}(t) := \prod_{i=1}^{k} (t-\lambda_i)^{m_i} \tag{3.21}$$

of minimal degree such that $\phi_{\mathbf{A},\mathbf{b}}(\mathbf{A})\mathbf{b} = \mathbf{0}$. Here $\lambda_1, \dots, \lambda_k$ are again the eigenvalues of $\mathbf{A}$.

Lemma 2. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a matrix with eigenvalues $\{\lambda_1, \dots, \lambda_k\}$, and minimal polynomial given by equation 2.19. Then for any two polynomials p_1, p_2 we have that $p_1(\mathbf{A})\mathbf{b} = p_2(\mathbf{A})\mathbf{b}$ if and only if*

$$\forall j \in \{1, \dots, k\} : \forall i \in \{0, \dots, n_k - 1\} : p_1^{(i)}(\lambda_j) = p_2^{(i)}(\lambda_j).$$

Proof. From theorem 5, we know that

$$\begin{aligned} & p_1(\mathbf{A}) = p_2(\mathbf{A}) \\ \Leftrightarrow\ & \forall j \in \{1, \dots, k\} : \forall i \in \{0, \dots, n_k - 1\} : p_1^{(i)}(\lambda_j) = p_2^{(i)}(\lambda_j) \end{aligned}$$

We also know that

$$p_1(\mathbf{A}) = p_2(\mathbf{A}) \Leftrightarrow p_1(\mathbf{A})\mathbf{b} = p_2(\mathbf{A})\mathbf{b}$$

Thus, we conclude that

$$\begin{aligned} & p_1(\mathbf{A})\mathbf{b} = p_2(\mathbf{A})\mathbf{b} \\ \Leftrightarrow \forall j \in \{1, \ldots, k\} : \forall i \in \{0, \ldots, n_k - 1\} : & p_1^{(i)}(\lambda_j) = p_2^{(i)}(\lambda_j) \end{aligned}$$

□

Theorem 6. *Let f be a sufficiently differentiable function that has no singularities on the spectrum of a given matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$. Then, with p the unique Hermite interpolating polynomial of $\mathbf{A}$ w.r.t. $\mathbf{b}$ i.e.*

$$\forall j \in \{1, \ldots, k\} : \forall i \in \{0, \ldots, m_k - 1\} : p^{(i)}(\lambda_j) = f^{(i)}(\lambda_j)$$

we have that $f(\mathbf{A})\mathbf{b} = p(\mathbf{A})\mathbf{b}$.

Proof. Let f be a function that is at least $\max_k\{m_k - 1\}$ times differentiable, and let p_1 be its $(m_1, \ldots, m_k)$-Hermite interpolant. Then, by definition 1, we have that

$$f(\mathbf{A}) = p_1(\mathbf{A})$$

Then, by lemma 2, let us define p_2 such that

$$p_2(\mathbf{A})\mathbf{b} = p_1(\mathbf{A})\mathbf{b}$$

Then, we have that

$$p_1(\mathbf{A})\mathbf{b} = p_2(\mathbf{A})\mathbf{b} = f(\mathbf{A})\mathbf{b}$$

□

3.2.2 The Arnoldi Method

The Arnoldi method is a reduction method that allows low-rank approximation of a given matrix $\mathbf{A}$. It is based on the Hessenberg reduction of a matrix $\mathbf{A}$. More formally

Definition 2. *Let $\mathbf{A} \in \mathbb{C}^{n \times n}$, the Arnoldi method approximates its Hessenberg reduction given by*

$$\mathbf{V}^*\mathbf{A}\mathbf{V} = \mathbf{H} \tag{3.22}$$

where $\mathbf{V} \in \mathbb{C}^{n \times n}$ is unitary, and $\mathbf{H} \in \mathbb{C}^{n \times n}$ is upper Hessenberg. As it is a (low-rank) approximation, Arnoldi will compute the following decomposition

$$\mathbf{V}_k^*\mathbf{A}\mathbf{V}_k = \mathbf{H}_k \tag{3.23}$$

with $\mathbf{V}_k \in \mathbb{C}^{n \times k}$ unitary, and $\mathbf{H}_k \in \mathbb{C}^{k \times k}$ upper Hessenberg. That means that $\mathbf{H}_k$ is the orthogonal projection of $\mathbf{A}$ onto the kth Krylov subspace $\mathcal{K}_k(\mathbf{A}, \mathbf{v_1})$, with $\mathbf{v_1}$ the first column of $\mathbf{V}$.

The Arnoldi method is an iterative method, following this algorithm:
Here the orthogonalization process (lines 3-6) is a modified Gram-Schmidt process. Also, when implementing, we will consider an ℓ_2 norm. Numerical conditions will often imply loss of orthogonality. To avoid this, it will be necessary to reorthogonalize, *i.e.* to reapply the Gram-Schmidt process.

Algorithm 1: Arnoldi Iteration

Data: $A \in \mathbb{C}^{n\times n}$, $v_1 \in \mathbb{C}^n$ a unit vector in the chosen norm (line 7)

Result: $H_n \in \mathbb{C}^{n\times n}$

1 **for** $k = 1$**to**n **do**
2     $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}_k$;
3   **for** $i = 1$**to**k **do**
4         $h_{ik} \leftarrow \mathbf{v}_i^* \mathbf{w}$;
5         $\mathbf{w} \leftarrow \mathbf{w} - h_{ik}\mathbf{v}_i$;
6   **end**
7     $h_{k+1,k} \leftarrow \|\mathbf{w}\|$;
8   **if** $h_{k+1,k} = 0$ **then**
9   **break**;
10   **end**
11     $\mathbf{v_{k+1}} \leftarrow \mathbf{w}/h_{k+1,k}$
12 **end**

3.2.3 The matrix-vector product $f(\mathbf{A})\mathbf{b}$

Recall the problem is to approximate the matrix-vector product $f(\mathbf{A})\mathbf{b}$. Now we have all the tools to achieve that efficiently.

Lemma 3. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ and $\mathbf{b} \in \mathbb{C}^n$. Then, the matrix vector product $f(\mathbf{A})\mathbf{b}$ can be approximated by*

$$f(\mathbf{A})\mathbf{b} \approx \|\mathbf{b}\|_2 \mathbf{V}_k f(\mathbf{H}_k)\mathbf{e}_1 \tag{3.24}$$

Where $\mathbf{V}_k$ and $\mathbf{H}_k$ are the matrices computed by the Arnoldi method after k iterations. $\mathbf{H}_k$ is upper Hessenberg and $\operatorname{span}(\mathbf{V}_k) = \mathcal{K}_k(\mathbf{A}, \mathbf{b}) = \operatorname{span}\left(\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \ldots, \mathbf{A}^{k-1}\mathbf{b}\right)$*. $\mathbf{e}_1$ is the first column of the identity matrix.*

Proof. Consider we want to approximate de matrix-vactor product $f(\mathbf{A})\mathbf{b}$. We will use the Arnoldi method (algorithm 12). For initial unit vector we choose $\mathbf{v}_1 = \mathbf{b}/\|\mathbf{b}\|_2$ which is indeed an ℓ_2 unit vector. According to the Hessenberg reduction of $\mathbf{A}$, we have that

$$f(\mathbf{A})\mathbf{b} \approx f(\mathbf{V_k}\mathbf{H}_k\mathbf{V}_k^*)\mathbf{b}$$

Since $\mathbf{V}_k$ is unitary, we have

$$f(\mathbf{A})\mathbf{b} \approx \mathbf{V}_k f(\mathbf{H}_k)\mathbf{V}_k^*\mathbf{b}$$

Furthermore, as we chose $\mathbf{v}_1 = \mathbf{b}/\|\mathbf{b}\|_2$, we have that

$$\mathbf{b} = \|\mathbf{b}\|_2\mathbf{v}_1 = \|\mathbf{b}\|_2\mathbf{V}_k\mathbf{e}_1$$

Thus, we have that

$$f(\mathbf{A})\mathbf{b} \approx \|\mathbf{b}\|_2\mathbf{V}_k f(\mathbf{H}_k)\mathbf{V}_k^*\mathbf{V}_k\mathbf{e}_1 = \|\mathbf{b}\|_2\mathbf{V}_k f(\mathbf{H}_k)\mathbf{e}_1$$

□

Lemma 4. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ and let $\mathbf{V}_k$, $\mathbf{H}_k$ be the matrices computed by the Arnoldi method after k iterations. Then, for any polynomial p_j of degree $j \leq k-1$, we have that*

$$p_j(\mathbf{A})\mathbf{b} = \mathbf{V}_k p_j(\mathbf{H}_k)\mathbf{e}_1 \tag{3.25}$$

Lemma 5. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ and let $\mathbf{V}_k$, $\mathbf{H}_k$ be the matrices computed by the Arnoldi method after k iterations. Then, given p_{k-1} the unique Hermite interpolant of $\mathbf{A}$ w.r.t. $\mathbf{b}$ of degree $k-1$, we have that*

$$\|\mathbf{b}\|_2\mathbf{V}_k f(\mathbf{H}_k)\mathbf{e}_1 = p_{k-1}(\mathbf{A})\mathbf{b} \tag{3.26}$$

Proof. Since p_{k-1} is the unique Hermite interpolant of $\mathbf{A}$ w.r.t. $\mathbf{b}$ of degree $k-1$, we have that

$$\|\mathbf{b}\|_2\mathbf{V}_k f(\mathbf{H}_k)\mathbf{e}_1 = \|\mathbf{b}\|_2\mathbf{V}_k p_{k-1}(\mathbf{H}_k)\mathbf{e}_1$$

And since

$$\|\mathbf{b}\|_2\mathbf{V}_k p_{k-1}(\mathbf{H}_k)\mathbf{e}_1 = \|\mathbf{b}\|_2 p_{k-1}(\mathbf{A})\mathbf{q}_1 = p_{k-1}(\mathbf{A})\mathbf{b}$$

We indeed have

$$\|\mathbf{b}\|_2\mathbf{V}_k f(\mathbf{H}_k)\mathbf{e}_1 = p_{k-1}(\mathbf{A})\mathbf{b}$$

□

Theorem 7. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ and let $m = \deg(\phi_{\mathbf{A},\mathbf{b}})$ then*

$$f(\mathbf{A})\mathbf{b} = \|\mathbf{b}\|_2\mathbf{V}_m f(\mathbf{H}_m)\mathbf{e}_1 \tag{3.27}$$

Proof. Following theorem 6, we have that

$$f(\mathbf{A})\mathbf{b} = p_{m-1}(\mathbf{A})\mathbf{b}$$

With p_{m-1} the unique Hermite interpolant of $\mathbf{A}$ w.r.t. $\mathbf{b}$ of degree $m-1$ with $\deg(\phi_{\mathbf{A},\mathbf{b}}) = m$. Then, by lemma 3, 4 and 5, we have that

$$f(\mathbf{A})\mathbf{b} = \|\mathbf{b}\|_2\mathbf{V}_m f(\mathbf{H}_m)\mathbf{e}_1$$

□

4 Algorithms

In this section, we will implement basic algorithms. From the previous section, we will distinguish two cases :

- the dense case, where $f(\mathbf{A})$ is to be computed
- and the case where only the matrix-vector product $f(\mathbf{A})\mathbf{b}$ is to be computed

4.1 Dense case

First, let us consider the computation of $f(\mathbf{A})$. Meaning we put aside the matrix-vector product (for now). More specifically, our approach will be to use definition 1 to compute $f(\mathbf{A})$. All implementations are done in `Matlab 2023a`.

4.1.1 Dependencies

We are provided with a function `hess_and_phi()` that computed the Hessenberg reduction of a given matrix and its associated minimal polynomial. It takes as input a matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$. It then follows these simple steps:

1. Compute the Hessenberg reduction of $\mathbf{A}$, i.e. $\mathbf{V}^*\mathbf{A}\mathbf{V} = \mathbf{H}$
2. It computes its Jordan Canonical Form from $\mathbf{H}$
3. It computes the minimal polynomial of $\mathbf{A}$, $\phi_{\mathbf{A}}$ through its Jordan Canonical Form
4. It returns $\mathbf{V}$, $\mathbf{H}$, $\mathbf{J}$, λ_j (the eigenvalues of $\mathbf{A}$) and n_i (the multiplicity of λ_i in $\phi_{\mathbf{A}}$)

Note that step 2 makes perfect sense.

Theorem 8. *Let $\mathbf{A} \in \mathbb{C}^{n \times n}$ with a Hessenberg reduction defined by $\mathbf{V}^*\mathbf{A}\mathbf{V} = \mathbf{H}$. Then $\mathbf{A}$ and $\mathbf{H}$ share the same Jordan Canonical Form.*

Proof. $\mathbf{V}$ is unitary, hence $\mathbf{A}$ and $\mathbf{H}$ are similar matrices, meaning they have the same eigenvalues. Thus, they share the same Jordan Canonical Form. □

To recover the minimal polynomial of $\mathbf{A}$, one simply retrives the λ_j and the n_i from the previously described steps, and then compute the polynomial this way:

$$\phi_{\mathbf{A}}(t) = \prod_{i=1}^{k} (t - \lambda_k)^{n_i}$$

In the supplementary materials, you will find an implementation of this, called `construct_minimal_polynomial()`. One quick way to assess if the minimal polynomial is constructed correctly, is to evaluate it at $\mathbf{A}$. Taking the ℓ_2 norm of the result should be close to zero. Here, with `test1.mat` (a 5 by 5 matrix), we obtain $\phi_{\mathbf{A}}(\mathbf{A}) = 3.3523e - 12$. This is indeed numerically close to zero.
For the Hermite interpolation, several options are possible. First, a routine `hermite_interp()` is provided in the supplementary materials, it computed the Hermite interpolation by divided differences. However, I had more precise results with an alternative method proposed by Or Werner, BGU, Israel, based on the Hermite method. In the following, I used the latter approach, the routine is provided in the supplementary material aswell.

4.1.2 Implementation

From the previous elements, we can construct a routine called `matrix_function()` that takes as input a matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$, and a function f. It has extremely simple steps:

1. Retrieve λ_j and n_i from `hess_and_phi()`
2. Construct the matrix `FdF` which is f and its derivatives evaluated at the eigenvalues of $\mathbf{A}$
3. Construct the Hermite interpolation of $\mathbf{A}$ with `hermite_interp()`
4. Evaluate the Hermite interpolation at $\mathbf{A}$, and return the result

The routine is provided in the supplementary materials.

4.1.3 Results

In this part, we will compare the performance of our routine `matrix_function()` with the built-in `funm()` function from `Matlab`. The built-in Matlab function uses a Schur-Parlett algorithm from Davies and Higham 2003.

5 Applications

5.1 Matrix Exponential

Consider the simple system of ODEs

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x} \tag{5.28}$$

with initial condition $\mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{R}^n$. Then we know the solution to be given by $\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}_0$. However, for all but the stablest systems, this is not a good method, due to issues such as stability and stiffness. Here we consider for instance the 2D convection-diffusion equation for the flow $\mathbf{u}(x, y)$:

$$\frac{d\mathbf{u}}{dt} = \epsilon\Delta\mathbf{u} + \alpha \cdot \nabla\mathbf{u}$$

with Dirichlet boundary conditions and $\epsilon \in \mathbb{R}_0^+$ and $\alpha \in \mathbb{R}^2$. Simple time-stepping methods are known to be unstable at large time-steps, and our exponential scheme suffers from similar problems, i.e. t cannot be taken too large.

5.2 The sign function

In control theory we are often interested in the eigenvalues λ of system matrices with $Re(\lambda) > 0$, since they correspond to unstable poles. In the design of controllers it is therefore interesting to have an efficient way to count the number of eigenvalues of a matrix in the right half-plane $Re(z) > 0$. Here we will build such a method.

Theorem 9. *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ be a matrix with k_- eigenvalues in the left plane, k_+ eigenvalues in the right plane and none on the imaginary axis, counting multiplicity. Let $sgn : \mathbb{C} \mapsto \{1, -1\}$ be defined by*

$$sgn(z) = \begin{cases} 1 & Re(z) \geq 0 \\ -1 & Re(z) < 0. \end{cases}$$

Then $trace(sgn(\mathbf{A})) = k_+ - k_-$.

Proof. Let $\mathbf{A} \in \mathbb{C}^{n\times n}$. To stay in a very general scenario, and consider cases where $\mathbf{A}$, let us consider its Jordan Canonical Form :

$$\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$$

Recall that by theorem 4, consider f a function, then

$$f(\mathbf{A}) = \mathbf{V}f(\mathbf{J})\mathbf{V}^{-1}$$

Thus let us consider the decomposition where

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_{k_+} & 0 \\ 0 & \mathbf{J}_{k_-} \end{pmatrix}$$

such that $\mathbf{J}_{k_+}$ has k_+ eigenvalues in the right plane, and $\mathbf{J}_{k_-}$ has k_- eigenvalues in the left plane. Then, we have that

$$\operatorname{sgn}(\mathbf{J}) = \begin{pmatrix} \mathbf{I}_{k_+} & 0 \\ 0 & -\mathbf{I}_{k_-} \end{pmatrix}$$

where $\mathbf{I}_n$ is the identity matrix of size n. Then, we have that

$$\operatorname{trace}(\operatorname{sgn}(\mathbf{J})) = k_+ - k_-$$

And thus, we have that

$$\operatorname{trace}(\operatorname{sgn}(\mathbf{A})) = \operatorname{trace}(\operatorname{sgn}(\mathbf{V}\mathbf{J}\mathbf{V}^{-1})) = \mathbf{V}\operatorname{trace}(\operatorname{sgn}(\mathbf{J}))\mathbf{V}^{-1} = (k_+ - k_-)\mathbf{V}\mathbf{V}^{-1} = k_+ - k_-$$

□

References

Davies, Philip I and Nicholas J Higham (2003). "A Schur-Parlett algorithm for computing matrix functions". In: *SIAM Journal on Matrix Analysis and Applications* 25.2, pp. 464–485.

Frommer, Andreas and Valeria Simoncini (2008). "Matrix functions". In: *Model order reduction: theory, research aspects and applications*. Springer, pp. 275–303.