# On the numerical computation of $A^{\alpha}\mathbf{x} = \mathbf{b}$

by

Siming Tian

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs

in partial fulfillment of the requirements for the degree of

Master of Science

in

Applied Mathematics

Carleton University

Ottawa, Ontario

# Abstract

The paper, *ODE-based double-preconditioning for solving linear systems* $A^\alpha x = b$ *and* $f(A)x = b$, by Antoine and Lorin [5], introduces different types of preconditioners for efficiently computing large sparse systems $A^\alpha x = b$. In this thesis, we introduce the notion of matrix functions $f(A)$ and present several methods for computing $p$-th root matrices $A^{1/p}$. Finally we propose some algorithms for solving fractional linear systems.

# Acknowledgements

A very sincere thanks to my supervisor, Dr. E. Lorin, who has guided my research, taught me so much, and provided me with some research funds. I would also like to acknowledge all of the professors from the Department of Mathematics and Statistics, who over the past two years have greatly enlarged my knowledge and given me tools which are needed in my research. I also thank my family and friends who have given me endless support in my efforts.

# Contents

# List of Tables

# Chapter 1

# Introduction

This thesis is dedicated to the numerical computation of the rational power of *large* matrices, and the solutions of *large* fractional linear systems $A^\alpha x = b$ for $\alpha \in \mathbb{Q}^*$. This work is motivated by the numerical approximation of fractional differential equations, and along this thesis the matrix $A$ *often* represents a "finite dimensional" approximate Laplace operator.

## 1.1 About matrix functions

A matrix function $f$ with output $f(A)$ is a matrix of the same dimension as $A$ in our scenario. i.e. $f : A \mapsto f(A)$, where $A \in \mathbb{C}^{n\times n}$ and $f(A) \in \mathbb{C}^{n\times n}$, $n \in \mathbb{N}^*$.

Let us start with a simple example. Consider the following system of ordinary differential equations:

$$\begin{cases} \dot{x}_1 = 2x_1 - x_2 \\ \dot{x}_2 = x_1 + 2x_2 \end{cases} \tag{1.1}$$

with initial condition

$$\mathbf{x}(0) = \mathbf{x}_0 = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}.$$

The above system also reads $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x}$, where

$$A = \begin{bmatrix} 2 & -1 \\ 1 & 2 \end{bmatrix},$$

with initial condition $\mathbf{x}(0) = \mathbf{x}_0$. By *The Fundamental Theorem for Linear Systems* (see Chapter 1 of [1]), the unique solution to the above system is given by: $\mathbf{x}(t) = e^{At}\mathbf{x}_0$, where by definition (analogous to the Taylor series expansion of $e^x$ when $x$ is a scalar), for any $t \in \mathbb{R}$,

$$e^{At} = \sum_{k=0}^{\infty} \frac{A^k t^k}{k!}. \tag{1.2}$$

In this case, for any $t \in \mathbb{R}$,

$$e^{At} = e^{2t} \begin{bmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{bmatrix}.$$

Hence, the unique solution to the system is given by:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} c_1 e^{2t}\cos(t) - c_2 e^{2t}\sin(t) \\ c_1 e^{2t}\sin(t) + c_2 e^{2t}\cos(t) \end{bmatrix}.$$

Here, we have employed the matrix exponential $f(A) = e^A$ to define the solution to the linear system. In the next chapter, we will consider different methods to

define a general matrix function $f(A)$.

Analogous to the matrix exponential, other important kinds of matrix functions include matrix sign function $f(A) = sign(A)$, matrix square root $f(A) = A^{1/2}$, matrix sector function $f(A) = sect_p(A)$ and matrix $p$-th root $f(A) = A^{1/p}$. Those types of matrix functions will be discussed in details in the following chapters, including "fractional" linear systems.

## 1.2 Objectives and references

This thesis mainly deals with the computation of matrix $p$-th roots and their associated "fractional" linear systems.

The main reference used in this thesis, is the textbook called *Functions of Matrices: Theory and Computation*, written by *Higham*, see [2]. This book provides the basic definitions and tools on matrix functions. Furthermore, I have also used some relevant papers written by *Higham* and other experts in this field of research.

# Chapter 2

# Definitions and algorithms for $f(A)$

## 2.1 Different definitions of $f(A)$

As it was studied in Section 1.2 of [2], there are several ways to define $f(A)$. For example, when it is easy to compute the Jordan Canonical Form of $A$, $f(A)$ can be defined via a Jordan Canonical Form. In the following subsections, we list some possible definitions of $f(A)$.

### 2.1.1 Definition of $f(A)$ via Jordan Canonical Form

Let us recall the notion of *Jordan Canonical Form* of a matrix $A$, see Chapter 1 of [1].

**Theorem 2.1 (The Jordan Canonical Form)** *Let $A$ be a real matrix with $k$ real eigenvalues $\lambda_j$, $j = 1, \cdots, k$ and $n-k$ complex eigenvalues $\lambda_j = a_j + \mathtt{i}b_j$ and*

$\bar{\lambda}_j = a_j - \mathtt{i}b_j$, $j = k+1, \cdots, n$. *Then there exists a basis* $\{\mathbf{v}_1, \cdots, \mathbf{v}_k, \mathbf{v}_{k+1}, \mathbf{u}_{k+1}, \cdots, \mathbf{v}_n, \mathbf{u}_n\}$ *of* $\mathbb{R}^{2n-k}$, *where* $\mathbf{v}_j$, $j = 1, \cdots, k$ *and* $\mathbf{w}_j$, $j = k+1, \cdots, n$ *are generalized eigenvectors of* $A$, $\mathbf{u}_j = Re(\mathbf{w}_j)$ *and* $\mathbf{v}_j = Im(\mathbf{w}_j)$ *for* $j = k+1, \cdots, n$, *such that the matrix* $Z = [\mathbf{v}_1 \ \cdots \ \mathbf{v}_k \ \mathbf{v}_{k+1} \ \mathbf{u}_{k+1} \ \cdots \ \mathbf{v}_n \ \mathbf{u}_n]$ *is invertible, and* $Z^{-1}AZ = J = diag(J_1, J_2, \cdots, J_p)$, *i.e.*

$$Z^{-1}AZ = \begin{bmatrix} J_1 & & \\ & \ddots & \\ & & J_p \end{bmatrix}, \tag{2.1}$$

*where the elementary Jordan blocks* $J = J_k$, $k = 1, \cdots, p$ *are either of the form*

$$J_k = J_k(\lambda_k) = \begin{bmatrix} \lambda_k & 1 & 0 & \cdots & 0 \\ 0 & \lambda_k & 1 & \cdots & 0 \\ \cdots & & & & \\ 0 & \cdots & & \lambda_k & 1 \\ 0 & \cdots & & 0 & \lambda_k \end{bmatrix} \in \mathbb{C}^{m_k \times m_k}, \tag{2.2}$$

*(where* $Z$ *is non-singular and* $m_1 + m_2 + \cdots + m_p = n$*) for* $\lambda_k$ *one of the real eigenvalues of* $A$, *or of the form*

$$J_k = \begin{bmatrix} D & I_2 & 0 & \cdots & 0 \\ 0 & D & I_2 & \cdots & 0 \\ \cdots & & & & \\ 0 & \cdots & & D & I_2 \\ 0 & \cdots & & 0 & D \end{bmatrix}, \tag{2.3}$$

*and where* $D = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}$, $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ *and* $0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ *for* $\lambda = a + \mathtt{i}b$ *one of the complex eigenvalues of* $J_k$.

The following simple example illustrates this theorem:

**Example 2.1** *Let us determine the Jordan Canonical Form of the matrix*

$$A = \begin{bmatrix} 3 & -2 \\ 1 & 1 \end{bmatrix}.$$

*The eigenvalues of matrix* $A$ *are* $\lambda_1 = 2 + \mathtt{i}$ *and* $\lambda_2 = 2 - \mathtt{i}$, *where* $\mathtt{i}$ *represents the complex number* $\sqrt{-1}$. *The corresponding eigenvectors are:*

$$\mathbf{v}_1 = \begin{bmatrix} 1 + \mathtt{i} \\ 1 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 1 - \mathtt{i} \\ 1 \end{bmatrix}.$$

*Thus, from Theorem 2.1, we have*

$$P = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, P^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix},$$

*and*

$$M = P^{-1}AP = \begin{bmatrix} 2 & -1 \\ 1 & 2 \end{bmatrix},$$

*which represents the Jordan Canonical Form of* $A$.

Now we have enough tools to define the matrix function via Jordan Canonical Form:

**Definition 2.1 (matrix function via Jordan Canonical Form)** *Let $A$ have the* Jordan Canonical Form *(2.2) and (2.3), then*

$$f(A) = Zf(J)Z^{-1} = Zdiag(f(J_k))Z^{-1}, \tag{2.4}$$

*where*

$$f(J_k) = \begin{bmatrix} f(\lambda_k) & f'(\lambda_k) & \cdots & \dfrac{f^{m_k-1}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \cdots \\ & & \ddots & f'(\lambda_k) \\ 0 & & & f(\lambda_k) \end{bmatrix}. \tag{2.5}$$

*See Section 1.2.1 of [2].*

In the above definition, $f^{m_k-1}(\lambda_k)$ denotes the $(m_k-1)$-th derivative of $f$.

The following simple example illustrates the above definition. Consider the following Jordan block

$$J = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix},$$

and $f(x) = x^2$, (2.5) gives

$$f(J) = \begin{bmatrix} f(2) & f'(2) \\ 0 & f(2) \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 0 & 4 \end{bmatrix},$$

which can be verified to be equal to $J^2$.

### 2.1.2 Definition of $f(A)$ via polynomials

Some background on polynomials with matrix arguments are needed in order to define $f(A)$ via polynomial interpolation, see Section 1.2.2 of [2].

**Definition 2.2** *The* minimal polynomial $\psi$ *of* $A \in \mathbb{C}^{n\times n}$ *is defined as the unique monic polynomial* $\psi$ *of lowest degree such that* $\psi(A) = 0$.

**Definition 2.3** *The function* $f$ *is said to be defined on the* spectrum *of* $A$ *if the values* $f^{(j)}(\lambda_i)$ *(denoting the* $j$*-th derivative of* $f$ *evaluated at* $\lambda_i$*), for* $j \in \{0, \cdots, n_i - 1\}$ *and* $i \in \{1, \cdots, s\}$ *(where* $\lambda_1, \cdots, \lambda_s$ *are the distinct eigenvalues of* $A$ *and* $n_i$ *is the order of the largest Jordan block in which* $\lambda_i$ *appears) exist. These are the* values *of the function* $f$ *on the spectrum of* $A$.

Now we have enough to define the matrix function.

**Definition 2.4 (matrix function via Hermite interpolation)** *Let* $f$ *be defined on the* spectrum *of* $A \in \mathbb{C}^{n\times n}$ *and let* $\psi$ *be the* minimal polynomial *of* $A$. *Set* $f(A) = p(A)$*, where* $p$ *is the polynomial of degree less than* $\sum_{i=0}^{s} n_i = deg(\psi)$ *that satisfies the* interpolation conditions

$$p^{(j)}(\lambda_i) = f^{(j)}(\lambda_i), j \in \{0, \cdots, n_i - 1\}, \qquad i \in \{1, \cdots, s\} \tag{2.6}$$

*Then there is a unique such* $p$ *with a minimal degree and it is known as the* Hermite interpolating polynomial.

An illustrative example is the following: consider $f(t) = \sqrt[3]{t}, \quad t \in \mathbb{R}$, and

$$A = \begin{bmatrix} 4 & 2 \\ 6 & 5 \end{bmatrix}.$$

The eigenvalues are $\lambda_1 = 1$ and $\lambda_2 = 8$, so that $s = 2$ and $n_1 = n_2 = 1$. Then the required interpolant satisfying $p(1) = f(1) = 1$, and $p(8) = f(8) = 2$ is:

$$p(t) = f(1)\frac{t-8}{1-8} + f(8)\frac{t-1}{8-1} = \frac{1}{7}(t+6).$$

Hence

$$f(A) = p(A) = \frac{1}{7}(A + 6I) = \frac{1}{7}\begin{bmatrix} 10 & 2 \\ 6 & 11 \end{bmatrix}.$$

The reader can check that $f(A)^3 = A$. Note the formula

$$A^{1/3} = \frac{A + 6I}{7},$$

holds for any diagonalizable $n \times n$ matrix $A$, having eigenvalues 1 and/or 8 and having a minimal polynomial that divides $\psi(t) = (t-1)(t-8)$, including the identity matrix of course.

### 2.1.3 Definition of matrix function via Cauchy integral

By the *Cauchy integral theorem* (see Section 1.2.3 of [2]), we are able to define a function of a matrix:

**Definition 2.5 (matrix function via Cauchy integral)** *For $A \in \mathbb{C}^{n\times n}$,*

$$f(A) = \frac{1}{2\pi \mathtt{i}} \int_{\Gamma} f(z)(zI - A)^{-1} dz, \tag{2.7}$$

*where $f$ is analytic on and inside a closed contour $\Gamma$, that encloses the spectrum of $A$, see Section 1.2.3 of [2].*

Note that Definition 2.1 (Jordan canonical form) and Definition 2.4 (Hermite interpolation) are equivalent (see Section 1.2.4 of [2]). If $f$ is analytic then Definition 2.5 (Cauchy integral) is equivalent to Definitions 2.1 and 2.4. Further details can be found in Section 1.2.4 of [2].

For a large sparse matrix $A \in \mathbb{C}^{n\times n}$, directly computing $f(A)$ is computationally costly in general, even with simple functions $f$, such as $f(A) = A^3$. Therefore it is important to have some efficient techniques for constructing $f(A)$, and introducing some numerical computing methods for $f(A)$ become crucial at this point.

## 2.2 Algorithms for computing $f(A)$

### 2.2.1 Taylor Series representation

One of the most fundamental tools for evaluating or approximating a matrix function $f(A)$, is *Taylor's series expansion*. However, the drawbacks of this technique are obvious: it is potentially quite restrictive because its radius of convergence is potentially small, and the error between the exact function and the approximated function introduced by truncated Taylor's series can be quite large. The following

theorem gives us the validity of a matrix Taylor series, which basically illustrates the first drawback, see Section 4.3 of [2].

**Theorem 2.2 (convergence of matrix Taylor series)** *Suppose $f$ has a Taylor series expansion*

$$f(z) = \sum_{k=0}^{\infty} a_k (z - \alpha)^k, \tag{2.8}$$

*where $a_k = \dfrac{f^{(k)}(\alpha)}{k!}$ with radius of convergence, $r$. If $A \in \mathbb{C}^{n \times n}$, then $f(A)$ is defined and is given by*

$$f(A) = \sum_{k=0}^{\infty} a_k (A - \alpha I)^k \tag{2.9}$$

*iff each of the distinct eigenvalues $\lambda_1, ..., \lambda_s$ of $A$ satisfies one of the following conditions*

1. $|\lambda_i - \alpha| < r$,
2. $|\lambda_i - \alpha| = r$ *and the series for $f^{(n_i - 1)}(\lambda)$ (where $n_i$ is the index of $\lambda_i$) is convergent at the point $\lambda = \lambda_i, i \in \{1, ..., s\}$.*

One of the most illustrative examples of a Taylor series expansion had already been mentioned in Section 1.1 Equation (1.2), which is the matrix Taylor series of exponential matrix function. Two other important examples based on Taylor series expansions (which is also analogous to the scalar function case) are:

$$\cos(A) = I - \frac{A^2}{2!} + \frac{A^4}{4!} - \frac{A^6}{6!} + ...$$

$$\sin(A) = A - \frac{A^3}{3!} + \frac{A^5}{5!} - \frac{A^7}{7!} + ...$$

The common feature in these three examples is that, the radii of convergence are infinite, so that these series can be used to better approximate the original functions if we include more and more terms in the series. Further details about the truncation errors and the proof of the theorem can be found in Section 4.3 of [2].

### 2.2.2 Rational approximation of $f(A)$

One of the most powerful tools for approximating $f(A)$ is *rational function approximation*. A rational function is of the form $r_{ab}$ which has numerator $f_{ab}$ and denominator $g_{ab}$, where $f_{ab}$ and $g_{ab}$ are polynomials, with degree at most $a$ and $b$ respectively (where $a, b \in \mathbb{N}$).

Among all kinds of rational approximations, Padé's approximants play an important role. Before introducing the formal definition of Padé's approximants, we need to introduce one more notation, see Section 4.4 of [2].

**Definition 2.6** *Let us denote* $\mathbf{R}_{a,b}$ *by the space (not vector space) of rational functions with numerator and denominator of degrees at most* $a$ *and* $b$*, respectively.*

**Definition 2.7** *For a given scalar function* $f$*, the rational function* $r_{ab} = \dfrac{f_{ab}}{g_{ab}}$ *is* $a$ $[a/b]$-Padé approximant *of* $f$*, if* $r_{a,b} \in \mathbf{R}_{a,b}, g_{ab}(0) = 1$*, and*

$$f(x) - r_{ab}(x) = O(x^{a+b+1}), \qquad x \in \mathbb{R}. \tag{2.10}$$

(Note that Equation 2.10 means the order of the difference between the original

function and its approximated rational function is $x^{a+b+1}$.)

Certain properties of Padé's approximations are crucial. One of them is the uniqueness of Padé's approximants, see Section 4.4.2 of [2].

**Theorem 2.3** *If a $[a/b]$-Padé approximant exists, then it is unique.*

Further details of rational approximations and the proof of the previous theorem could be found in Section 4.4.2 of [2].

### 2.2.3 Representations of Padé approximation

There are various methods to represent Padé's approximants. The most common one is the continued fraction representations: (note that the regularity of $f$ is required)

$$f(x) = b_0 + \cfrac{a_1 x}{b_1 + \cfrac{a_2 x}{b_2 + \cfrac{a_3 x}{b_3 + \dots}}}, \qquad x \in \mathbb{R}.$$

In particular, when $b_1 = b_2 = ... = 1$ and the $a_i's$ are nonzero, then $f(x) = r_m(x)$ becomes:

$$r_m(x) \equiv r_{mm}(x) = b_0 + \cfrac{a_1 x}{b_1 + \cfrac{a_2 x}{b_2 + \cfrac{a_3 x}{b_3 + \dots + \cfrac{a_{2m-1}(x)}{b_{2m-1} + \cfrac{a_{2m}(x)}{b_{2m}}}}}} \quad . \tag{2.11}$$

Note that the convergent $r_m(x)$ are the $[0/0], [1/0], [1/1], [2/1], [2/2], ...$(this means the degree $[a/b]$) Padé approximants of $f$, see Section 4.4.2 of [2].

Partial fraction form of $r_m$ in (2.11) is another kind of representation, and

reads

$$r_m(x) = \sum_{j=1}^{m} \frac{\alpha_j^{(m)} x}{1 + \beta_j^{(m)} x}, \qquad x \in \mathbb{R}, \tag{2.12}$$

where the coefficients $\beta_j^{(m)}$ are the negative value of the reciprocals (see Section 4.4.3 of [2] for more details) of the roots of the denominator polynomial $q_{mm}(x)$. When $\beta_j^{(m)}$ is complex, quadratic denominators can be used, see Section 4.4.3 of [2].

### 2.2.4 Other techniques to approximate $f(A)$

Interpolating polynomials is another way for approximating $f(A)$. The definition $f(A) = p(A)$, where $p$ is the Hermite interpolating polynomial, gives us one kind of numerical computing method. However, this method is not so useful in practice since finding all eigenvalues of a large sparse matrix A is generally impossible.

Another kind of technique which is commonly used is called *matrix iterations* which consists in iterating

$$X_{k+1} = g(X_k), \qquad k \geq 0. \tag{2.13}$$

In practice, we usually take $X_0 = I$ or $X_0 = A$, and the function $g$ might depend on $A$ in different contexts. We will use matrix iterations to find Padé approximations of $f(A)$. This will be discussed in the next chapter.

# Chapter 3

# Matrix $p$-th root computation

As mentioned in the previous chapter, Padé's approximants would be particularly accurate when approximating a generic nonlinear matrix function. In the following sections, we will introduce different types of commonly used matrix functions and their corresponding Padé's approximants.

## 3.1 Matrix sign function and the matrix square root

Computing matrix sign function is important since there are some connections between the computation of matrix sign function and the computation of matrix square root. Similarly, there are also some connections between the computation of matrix $p$-th root and the computation of matrix sector function as well. Let us start with computing matrix sign function.

### 3.1.1 Matrix sign function

Analogous to the scalar sign function which is defined as

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

for $z \in \mathbb{C}$ lying off the imaginary axis. Note that $\text{sign}(z)$ can also be expressed as $\text{sign}(z) = e^{\mathbf{i}arg(z)}$ or $\text{sign}(z) = z/|z|$ for $z \neq 0$, or $\text{sign}(z) = z/(z^2)^{1/2}$.

The *matrix sign function* can be defined using three different ways, see Chapter 2. Note that $I_a$ and $I_b$ are $a \times a$ and $b \times b$ identity matrices, respectively.

1. If $A = PDP^{-1}$ is a Jordan canonical form and $D = \text{diag}(D_1, D_2)$ where the real part of the eigenvalues of $D_1 \in \mathbb{C}^{a\times a}$ are negative and those of $D_2 \in \mathbb{C}^{b\times b}$ are positive, where $a$ and $b$ are two positive integers, then:

$$\text{sign}(A) = P \begin{bmatrix} -I_a & 0 \\ 0 & I_b \end{bmatrix} P^{-1}. \tag{3.1}$$

2.

$$\text{sign}(A) = A(A^2)^{-1/2}, \tag{3.2}$$

3. The integral representation of $\text{sign}(A)$ is given as the following:

$$\text{sign}(A) = \frac{2}{\pi} A \int_0^\infty (t^2 I + A^2)^{-1} dt. \tag{3.3}$$

Note that Equation (3.3) can be derived from Equation (3.2) by the Cauchy integral formula (2.7). Further details on this can be found at the beginning of Chapter 5 of [2].

From Equation (3.2), we may conclude that the matrix sign function is closely related to the square root of the matrix. The following theorem is useful to find the relationship between the matrix sign function and the matrix square root, see the beginning of Chapter 5 of [2]:

**Theorem 3.1** *Let $A, B \in \mathbb{C}^{n\times n}$ and suppose $AB$ and $BA$ have no eigenvalue on $\mathbb{R}^-$, then:*

$$sign\left(\begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & C \\ C^{-1} & 0 \end{bmatrix}, \tag{3.4}$$

*where $C = A(BA)^{-1/2}$.*

**Proof** (outline) Let

$$P = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix}.$$

Then

$$P^2 = \begin{bmatrix} AB & 0 \\ 0 & BA \end{bmatrix},$$

by direct computation. Note that the matrix $P$ cannot have any eigenvalues on the imaginary axis since otherwise $P^2$ would have an eigenvalue on $\mathbb{R}^-$.

Now by definition of the sign function,

$$sign(P) = P(P^2)^{-\frac{1}{2}} = \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \begin{bmatrix} AB & 0 \\ 0 & BA \end{bmatrix}^{-\frac{1}{2}}$$

$$= \begin{bmatrix} 0 & A \\ B & 0 \end{bmatrix} \begin{bmatrix} (AB)^{-\frac{1}{2}} & 0 \\ 0 & (BA)^{-\frac{1}{2}} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & A(BA)^{-\frac{1}{2}} \\ B(AB)^{-\frac{1}{2}} & 0 \end{bmatrix} = \begin{bmatrix} 0 & C \\ D & 0 \end{bmatrix}.$$

Now since

$$I = (sign(P))^2 = \begin{bmatrix} 0 & C \\ D & 0 \end{bmatrix}^2 = \begin{bmatrix} CD & 0 \\ 0 & DC \end{bmatrix},$$

then $D = C^{-1}$. □

In order to compute the matrix sign function numerically, we employ the idea of matrix iterations which was mentioned earlier. One of them which is commonly used is called Newton's iteration method, and is given by:

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}), \qquad X_0 = A, \tag{3.5}$$

assuming that $A \in \mathbb{C}^{n\times n}$ have no pure imaginary eigenvalues. Then, it can be shown that the Newton iterates $X_k$ converges quadratically to $S = \text{sign}(A)$.

Note that Newton's iteration method is one kind of rational matrix iteration method to compute the matrix sign function and will be detailed in Subsection 3.1.4.

### 3.1.2 The Padé family of iterations for matrix sign function

Now, we have enough tools to derive the Padé family of iterations for the matrix sign function.

Let $y_k = x_k^{-1}$, in the Newton formula $x_{k+1} = (x_k + x_k^{-1})/2$, (see Equation (3.5)) we have: $1/y_{k+1} = (y_k^{-1} + y_k)/2 = (y_k^2 + 1)/2y_k$. Rearrange it, we get:

$$y_{k+1} = 2y_k/(y_k^2 + 1), \qquad y_0 = a, \tag{3.6}$$

for all $k \geq 0$ and $a \in \mathbb{R}$, which is called the "inverse Newton" variant and which has quadratic convergence to $\text{sign}(a)$, see Section 5.4 of [2].

Taking this iterative step again, we have $y_{k+2} = 2y_{k+1}/(y_{k+1}^2 + 1)$. Combine this equation with (3.6), we have: $y_{k+2} = (y_k^4 + 6y_k^2 + 1)/4y_k(y_k^2 + 1)$. Now we could define the convergent iteration as:

$$y_{k+1} = \frac{y_k^4 + 6y_k^2 + 1}{4y_k(y_k^2 + 1)}, \qquad y_0 = a, \tag{3.7}$$

which corresponds to the inverse of the [1,2]- Padé approximation as shown in table (3.1).

Next we focus on the derivation of the Padé iteration formula. Note that we could write

$$\text{sign}(z) = \frac{z}{(z^2)^{1/2}} = \frac{z}{(1-(1-z^2))^{1/2}} = \frac{z}{(1-\xi)^{1/2}}, \tag{3.8}$$

where $\xi = 1-z^2$, for non-pure imaginary $z \in \mathbb{C}$. Hence, we need an approximation of $h(\xi) = (1-\xi)^{-1/2}$, in order to approximate sign$(z)$. We may assume that $|\xi| < 1$ (otherwise $h(\xi)$ is undefined). Now, we will concentrate on the Padé approximations to $(1-\xi)^{-1/2}$:

First of all, we need the notion of hypergeometric series, see Chapter 5 of [3]:

$$ {}_2F_1(\alpha,\beta;\gamma;x) = 1+\frac{\alpha\beta}{1\times\gamma}x+\frac{\alpha(\alpha+1)\beta(\beta+1)}{1\times 2\times\gamma(\gamma+1)}x^2+\frac{\alpha(\alpha+1)(\alpha+2)\beta(\beta+1)(\beta+2)}{1\times 2\times 3\times\gamma(\gamma+1)(\gamma+2)}x^3+... \tag{3.9} $$

(${}_2F_1$ means that 2 parameters $\alpha$ and $\beta$ are used on the numerator and 1 parameter $\gamma$ is used on the denominator in Equation 3.9), $\alpha$ and $\beta$ could be any complex constants, but $\gamma$ cannot be $0, -1, -2, ....$ Note that

$$ (1-\xi)^{-1/2} = {}_2F_1(1/2,1;1;\xi) = \cfrac{1}{1+\cfrac{(-1/2)\xi}{1-\cfrac{(1/2)(-1/2+1)\xi}{1+\cfrac{(1/6)(-1/2-1)\xi}{1-\cfrac{(1/6)(-1/2+2)\xi}{1+...}}}}} \tag{3.10} $$

which is called *Gauss' Continued Fraction*, see Chapter 5 of [3].

By Definition 2.7 (refer to equation (2.10) for more details), we need an explicit expression of $f_{ab}(\xi)$ and $g_{ab}(\xi)$ for the Padé approximant $r_{ab}(\xi)$ of $h(\xi)$ since $r_{ab}(\xi) = f_{ab}(\xi)/g_{ab}(\xi)$.

Define $(\alpha)_n = (\alpha)(\alpha+1)...(\alpha+n+1)$ with $(\alpha)_0 = 1$. Then $f_{ab}(\xi)$ and $g_{ab}(\xi)$

are given by (see [4]):

$$f_{ab}(\xi) = \sum_{n=0}^{a} \frac{(1/2)_n (1/2-b)_b (n-a-b)_b}{n!(-a-b)_b (n+1/2-b)_b} \xi^n \equiv \sum_{n=0}^{a} f_n^{ab} \xi^n, \tag{3.11}$$

$$g_{ab}(\xi) = {}_2F_1(-b, -1/2-a; -a-b; \xi) = \sum_{n=0}^{b} \frac{(-b)_n (-1/2-a)_n \xi^n}{n!(-a-b)_n} \equiv \sum_{n=0}^{b} g_n^{ab} \xi^n. \tag{3.12}$$

The following theorem illustrates the local error of the above Padé recursions:

**Theorem 3.2** *[4] For $a \geq b-1$ and $|\xi| < 1$,*

$$g_{ab}^2(\xi) - (1-\xi) f_{ab}^2(\xi) = \xi^{a+b+1} (\sum_{i=1}^{\mu} c_i \xi^i), \tag{3.13}$$

*where $c_i = c_i(a,b) > 0$ for $0 \leq i \leq \mu \equiv max(2a+1, 2b) - (a+b+1)$, and*

$$g_{ab}^2(1) = \sum_{i=1}^{\mu} c_i. \tag{3.14}$$

Further details and the proof of this theorem can be found in [4].

Now we have enough tools to find the Padé approximations of the matrix sign function (3.8). The scalar Padé iteration of (3.8) is given by:

$$x_{n+1} = x_n \frac{f_{ab}(1-x_n^2)}{g_{ab}(1-x_n^2)}, \qquad x_0 = a. \tag{3.15}$$

**Padé recursions for the matrix sign function**

| | $a=0$ | $a=1$ | $a=2$ | $a=3$ |
|---|---|---|---|---|
| $b=0$ | $x$ | $\frac{x}{2}(3-x^2)$ | $\frac{x}{8}(15-10x^2+3x^4)$ | $\frac{x}{16}(35-35x^2+21x^4-5x^6)$ |
| $b=1$ | $\frac{2x}{1+x^2}$ | $\frac{x(3+x^2)}{1+3x^2}$ | $\frac{x}{4}\frac{15+10x^2-x^4}{1+5x^2}$ | $\frac{x}{8}\frac{35+35x^2-7x^4+x^6}{(1+7x^2)}$ |
| $b=2$ | $\frac{8x}{3+6x^2-x^4}$ | $\frac{4x(1+x^2)}{1+6x^2+x^4}$ | $\frac{x(5+10x^2+x^4)}{1+10x^2+5x^4}$ | $\frac{x}{2}\frac{(35+105x^2+21x^4-x^6)}{(3+42x^2+35x^4)}$ |
| $b=3$ | $\frac{16x}{5+15x^2-5x^4+x^6}$ | $\frac{8x(3+5x^2)}{(5+45x^2+15x^4-x^6)}$ | $\frac{2x(3+10x^2+3x^4)}{1+15x^2+15x^4+x^6}$ | $\frac{x(7+35x^2+21x^4+x^6)}{1+21x^2+35x^4+7x^6}$ |

Table 3.1: Iteration functions $r_{ab}$ from the Padé family (3.17).

where $f_{ab}/g_{ab}$ is the $[a/b]$ Padé approximant of $h(\xi)$, and $a$ is the initial value. Table (3.1) gives the expressions for the RHS of (3.15), see [4].

Now consider the following Padé iteration for the matrix sign function $S = sign(A)$, for $n \geq 0$,

$$g_{ab}(I - X_n^2)X_{n+1} = X_n f_{ab}(I - X_n^2), \qquad X_0 = A. \tag{3.16}$$

The following theorem yields the convergence of (3.16):

**Theorem 3.3 (Convergence of Padé iterations)** *Let $A \in \mathbb{C}^{n\times n}$ have no pure imaginary eigenvalues. For the iteration (3.16) with $a+b > 0$ and any subordinate matrix norm, the following holds:*

1. *For $a \geq b-1$, if $||I - A^2|| < 1$ then $X_n \to sign(A)$ as $n \to \infty$, and $||I - X_n^2|| < ||I - A^2||^{(a+b+1)^n}$.*

*2. For $a = b - 1$ or $a = b$,*

$$(S - X_n)(S + X_n)^{-1} = [(S - A)(S + A)^{-1}]^{(a+b+1)^n},$$

*and so $X_n \to sign(A)$ as $n \to \infty$.*

The details of this theorem can be found in Section 5.4 of [2]. The proof of the statement 1 of this theorem can be found in the proof of theorem 3.1 of [4], and the proof of the statement 2 of this theorem can be found in the proof of theorem 3.2 of [4].

*Principal Padé iterations* is another important class of Padé iterations. It corresponds to the cases $a = b - 1$ or $a = b$, where $a, b \in \mathbb{N}$. For those $a$ and $b$ define

$$g_r(x) \equiv g_{a+b+1}(x) = r_{ab}(x), \qquad x \in \mathbb{R}. \tag{3.17}$$

Note that the $g_r$'s are the iteration functions from Table (3.1) taken a zig-zag fashion from the main diagonal and the first subdiagonal:

$$\begin{aligned} &g_1(x) = x, \qquad g_2(x) = \frac{2x}{1+x^2}, \qquad g_3(x) = \frac{x(3+x^2)}{1+3x^2}, \\ &g_4(x) = \frac{4x(1+x^2)}{1+6x^2+x^4}, \qquad g_5(x) = \frac{x(5+10x^2+x^4)}{1+10x^2+5x^4}, \\ &g_6(x) = \frac{x(6+20x^2+6x^4)}{1+15x^2+15x^4+x^6}. \end{aligned} \tag{3.18}$$

From Theorem (3.3) it can be shown that the iteration $X_{n+1} = g_r(X_n)$ converges to $\text{sign}(X_0)$, at order $r$ (where $r$ represents the notation $g_r(x)$) when $\text{sign}(X_0)$ is defined. The next theorem shows some properties of *principal Padé's iterations*:

**Theorem 3.4** *The* principal Padé iteration *(see Section 5.4 of [2]) function $g_r$ (where $r$ is the order) in (3.18) has the following properties (for $x \in \mathbb{R}$):*

1. $g_r(x) = \dfrac{(1+x)^r - (1-x)^r}{(1+x)^r + (1-x)^r}$. *i.e.* $g_r = \dfrac{p_r(x)}{q_r(x)}$, *where $p_r(x)$ and $q_r(x)$ are the odd and even parts of $(1+x)^r$, respectively.*

2. $g_r(x) = tanh(r \times arctanh(x))$.

3. $g_r(g_s(x)) = g_{rs}(x)$.

4. $g_r$ *has the partial fraction expression*

$$g_r(x) = \frac{2}{r} {\sum_{i=0}^{\lceil \frac{r-2}{2} \rceil}}' \frac{x}{sin^2(\frac{(2i+1)\pi}{2r}) + cos^2(\frac{(2i+1)\pi}{2r})x^2}, \tag{3.19}$$

*where the prime on the summation symbol means that the last term in the sum takes half of its original value when $r$ is odd.*

Further details about the matrix sign function and the proof of this theorem can be found in Section 5.4 of [2].

### 3.1.3 Matrix square root

Analogous to the square root function, we can also define the matrix square root, as it is one of the most commonly used matrix functions. In this section, we will concentrate more on the principle square root of $A$, which is uniquely well-defined by the following theorem (see Section 1.7 of [2]):

**Theorem 3.5 (Principal square root)** *Assume $A \in \mathbb{C}^{n\times n}$ has no eigenvalue on $\mathbb{R}^-$ (in order to avoid the line of singularity). There exists a unique square root*

*X of A, for which all eigenvalues lie in the open right half-plane, and is defined as the* principal square root *of A, and reads* $X = A^{1/2}$. *Note that if A is a real-valued matrix then so is* $A^{1/2}$.

Recall from Definition 2.7 that we can express any matrix functions via a Cauchy integral. The matrix principal square root can then be expressed as follows:

$$A^{1/2} = \frac{2}{\pi} A \int_0^\infty (t^2 I + A)^{-1} dt. \tag{3.20}$$

Note that if we replace the matrix $B$ in Equation (3.4) by $I$, the identity matrix, then $C = A(BA)^{-1/2} = A^{1/2}$. We obtain the following equation from (3.4):

$$\operatorname{sign}\left(\begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{bmatrix}. \tag{3.21}$$

It can be shown that if we replace $A$ by $\begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix}$ in Equation (3.4) and only evaluate the (1,2)-block of (3.3), we would get the Equation (3.21).

In order to compute the matrix principal square root numerically, we employ the idea of iterations discussed before. In order to evaluate the matrix principle square root, a specific form of $g(X_k)$ in Equation (2.13) will be introduced in the following theorem:

**Theorem 3.6** *Let* $A \in \mathbb{C}^{n\times n}$ *have no eigenvalues on* $\mathbb{R}^-$. *Consider any iteration*

*of the form* $X_{k+1} = g(X_k) \equiv X_k h(X_k^2)$ *that converges to* $sign(X_0)$ *for*

$$X_0 = \begin{bmatrix} 0 & A \\ I & 0 \end{bmatrix},$$

*at order* $m$ *(that is* $||X_{k+1} - X_{exact}|| \leq c||X_k - X_{exact}||^m$*, for some* $c > 0$*). Then, in the coupled iteration*

$$\begin{cases} Y_{k+1} = Y_k h(Z_k Y_k), Y_0 = A, \\ Z_{k+1} = h(Z_k Y_k) Z_k, Z_0 = I, \end{cases} \tag{3.22}$$

$Y_k \to A^{\frac{1}{2}}$ *and* $Z_k \to A^{-\frac{1}{2}}$ *as* $k \to \infty$*, both at order* $m$*. Moreover,* $Y_k$ *commutes with* $Z_k$*, and* $Y_k = AZ_k$ *for all* $k$*.*

Further details and proof of this theorem can be found, see Section 6.7 of [2].

### 3.1.4 Padé approximation for the principle square root

First we introduce the idea of Newton's method for solving $X^2 = A$:

Let $\widetilde{X}$ be an approximate solution and let $\widetilde{X} + E = X$, where the error $E$ is to be determined and is assumed to be sufficiently small. Then $A = (\widetilde{X} + E)^2 = \widetilde{X}^2 + \widetilde{X}E + E\widetilde{X} + E^2$. When $E$ is assumed to be sufficiently small, we drop the $E^2$-term. Then Newton's method is the following:

Given $X_0$, we need to solve

$$\begin{aligned} &X_k E_k + E_k X_k = A - X_k^2 \\ &X_{k+1} = X_k + E_k, \\ &for \quad k \geq 0. \end{aligned} \tag{3.23}$$

It had been justified in Section 6.3 of [2], that if $X_0$ commutes with $A$ and all the iterates in (3.22) are well-defined, then $X_k$ commutes with $A$ and $X_{k+1} = (X_k + X_k^{-1}A)/2$ for all $k$. We could choose $X_0 = A$ so that $X_0$ commutes with $A$. This leads to the Newton iteration of matrix square root:

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1}A), \qquad X_0 = A. \tag{3.24}$$

The following theorem states the convergence of Newton square root iteration (3.24):

**Theorem 3.7** *Assume that $A \in \mathbb{C}^{n\times n}$ has no eigenvalue on $\mathbb{R}^-$. Newton square root iterates $X_k$ from (3.24) with arbitrary $X_0$ which commutes with $A$ are related to Newton sign iterates*

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

*by $X_k \equiv A^{1/2}S_k$. Hence, provided that $A^{-1/2}X_0$ has no pure imaginary eigenvalues, the $X_k$ are defined and $X_k$ converges quadratically to $A^{1/2}sign(A^{-1/2}X_0)$. If the spectrum of $A^{-1/2}X_0$ lies in the right half-plane, then $X_k$ converges quadrati-*

*cally to* $A^{1/2}$ *and*

$$||X_{k+1} - A^{1/2}|| \leq \frac{1}{2}\|X_k^{-1}\|\|X_k - A^{1/2}\|^2. \tag{3.25}$$

Note that $S$ is chosen such that $S = \text{sign}(A)$ in the above theorem and the details of the proof can be found in Section 6.3 of [2].

If we define $Y_k = A^{-1}X_k$ in (3.22), then $X_{k+1} = (X_k + Y_k^{-1})/2$ and $Y_{k+1} = A^{-1}X_{k+1} = (Y_k + X_k^{-1})/2$, based on the fact that $X_0$ commutes with $A$. Here is another explicit iterative method for solving the principle matrix square root :

$$\begin{aligned} X_{k+1} &= \frac{1}{2}(X_k + Y_k^{-1}), X_0 = A, \\ Y_{k+1} &= \frac{1}{2}(Y_k + X_k^{-1}), Y_0 = I. \end{aligned} \tag{3.26}$$

Under conditions of Theorem 3.7, we have:

$$\lim_{k\to\infty} X_k = A^{1/2}, \qquad \lim_{k\to\infty} Y_k = A^{-1/2}. \tag{3.27}$$

Compare (3.26) with (3.24), we can see that there is no matrix product in (3.26). However, we have to iterate a system instead of a single equation in (3.26).

Note that Theorem 3.6 provides an alternative derivation of (3.26) by taking $g(X) = (X + X^{-1})/2 = X(I + X^{-2})/2 \equiv Xh(X^2)$ with $h(X) = (I + X^{-1})/2$. Then, by (3.22), $Y_{k+1} = Y_k h(Z_k Y_k) = Y_k(I + (Z_k Y_k)^{-1})/2 = Y_k(I + Y_k^{-1}Z_k^{-1})/2 = (Y_k + Z_k^{-1})/2$. By similar calculation, we have $Z_{k+1} = (Z_k + Y_k^{-1})/2$.

Padé iteration for principal matrix square root is obtained by Theorem 3.6 to

the Padé family (3.18):

$$\begin{aligned} Y_{k+1} &= Y_k f_{ab}(1 - Z_k Y_k) g_{ab}(1 - Z_k Y_k)^{-1}, Y_0 = A, \\ Z_{k+1} &= f_{ab}(1 - Z_k Y_k) g_{ab}(1 - Z_k Y_k)^{-1} Z_k, Z_0 = I. \end{aligned} \tag{3.28}$$

In terms of the convergence of Padé iteration, $Y_k \to A^{1/2}$ and $Z_k \to A^{-1/2}$ with order $a+b+1$ if $a = b-1$ or $a = b$. Otherwise if $a \geq b+1$ then we need an additional condition $\|\text{diag}(I - A, I - A)\| < 1$ for this iteration to be convergent, and with order $a + b + 1$. This result can be deduced from Theorem 3.3 and Theorem 3.6.

## 3.2 Matrix $p$-th root (including matrix sector function)

In this chapter, we generalize the idea of connecting matrix square root and matrix sign function to the idea of connecting matrix $p$-th root and matrix sector function. Let us start with matrix sector function.

### 3.2.1 Matrix sector function

The matrix sector function for an arbitrary matrix $A$ (given that $A \in \mathbb{C}^{n \times n}$ having no eigenvalues with argument $(2k+1)\pi/p, k = 0, \cdots, p-1$) is defined as:

$$\text{sect}_p(A) = A(A^p)^{-1/p}.$$

This value may or may not be unique.

Note that when $n = 2$, the scalar 2-sector function of $z$ is the sign function of $z$, i.e.

$$\text{sector}_2(z) \equiv S_2(z) = \text{Sign}(z) = e^{\mathtt{i}\pi k}, \qquad for \qquad k \in [0, 1],$$

Further details of this definition can be found in [6].

We can also define the matrix sector function via Jordan canonical form and via a Cauchy Integral:

1. Jordan canonical form of $\text{sect}_p(A)$:

$$\text{sect}_p(A) = P\text{diag}(s_p(\lambda_1), ..., s_p(\lambda_n))P^{-1},$$

   where $P^{-1}AP = D$ is a Jordan canonical form of $A \in \mathbb{C}^{n\times n}$ and the diagonal of $D$ is the vector $(\lambda_1, ..., \lambda_n)$ where $\lambda_1, ..., \lambda_n$ are the eigenvalues of $A$.

2. Integral representation of $\text{sect}_p(A)$:

$$\text{sect}_p(A) = \frac{p\sin(\pi/p)}{\pi} A \int_0^\infty (t^p I + A^p)^{-1} dt.$$

### 3.2.2 Padé family of iterations for sector function

Now we focus on the Padé family of iterations for sector function. Note that we can rewrite

$$s_n(z) = \frac{z}{\sqrt[p]{z^p}} = \frac{z}{\sqrt[p]{1-(1-z^p)}} = \frac{z}{(1-\xi)^{1/p}} \tag{3.29}$$

where we define $\xi = 1 - z^p$.

Hence we need an approximation of

$$\phi(\xi) = (1-\xi)^{-1/p}, \tag{3.30}$$

in order to approximate $s_p(z)$. Note that $\phi(\xi)$ can be expressed in terms of hypergeometric functions of the form of Equation (3.10), where:

$$\phi(\xi) = {}_2F_1(1/p, 1; 1; \xi) = \sum_{j=0}^{\infty} \frac{(1/p)_j}{j!} z^j, \tag{3.31}$$

where $(1/p)_j = (1/p) \times (1/p+1)...(1/p+j-1)$ with $(1/p)_0 = 1$, and $\xi \in (0,1)$. Note that Equation (3.31) is valid, when $|z| < 1$, see Section 2 of [7].

Let $f_{ab}(\xi)/g_{ab}(\xi)$ be the $[a/b]$ Padé approximant of $\phi(\xi)$. Then the polynomial $g_{ab}$ has the following form:

$$g_{ab}(\xi) = {}_2F_1(-b, -1/p-a; -a-b; \xi) = \sum_{j=0}^{b} g_j^{(ab)} \xi^j = \sum_{j=0}^{b} \frac{(-b)_j(-1/p-a)_j}{j!(-a-b)_j} \xi^j, \tag{3.32}$$

where $|\xi| \leq 1$. From (3.35) and (2.3), we are able to derive the following formula for $f_{ab}$:

$$f_{ab}(\xi) = \sum_{j=0}^{a} f_j^{(ab)} \xi^j = \sum_{j=0}^{a} \frac{(1/p)_j(1/p-b)_b(j-a-b)_b}{j!(-a-b)_b(j+1/p-b)_b} \xi^j. \tag{3.33}$$

Analogous to the scalar sign function case, the Padé iteration for the function $s_n(z)$ has the following form corresponds to the Padé approximant $[a/b]$: For

$l \geq 0$,

$$x_{l+1} = r_{ab}(x_l) = x_l \frac{f_{ab}(1 - x_l^p)}{g_{ab}(1 - x_l^p)}, \qquad x_0 = z, \tag{3.34}$$

where $z \in \mathbb{R}$ and $p \in \mathbb{N}$.

The Padé family of iterations for the matrix sector function $sect_p(A)$, where $A \in \mathbb{C}^{n \times n}$, is given by the following:

$$X_{l+1} = r_{ab}(X_l), X_0 = A \qquad where \qquad g_{ab}(I - X_l^p) r_{ab} = X_l f_{ab}(I - X_l^p). \tag{3.35}$$

Taking values of $a = 0, 1, 2; \quad b = 0, 1, 2$ for $r_{ab}$ in (3.35) generates the following iteration functions, see Table 1 of [7]. The following functions apply for all $x \in \mathbb{R}$:

$$r_{00}(x) = x, \qquad r_{10}(x) = \frac{x}{p}[(1 + p) - x^p],$$

$$r_{20}(x) = \frac{x}{2p^2}[(2p^2 + 3p + 1) - (4p + 2)x^p + (p + 1)x^{2p}],$$

$$r_{01}(x) = \frac{px}{(p - 1) + x^p}, \qquad r_{11}(x) = x \frac{(p + 1) + (p - 1)x^p)}{(p - 1) + (p + 1)x^p},$$

$$r_{21}(x) = \frac{x}{2p} \frac{(2p^2 + 3p + 1) + (4p^2 - 2p - 2)x^p + (1 - p)x^{2p}}{(p - 1) + (2p + 1)x^p},$$

$$r_{02}(x) = \frac{2p^2 x}{(2p^2 - 3p + 1) + (4p - 2)x^p + (1 - p)x^{2p}},$$

$$r_{12}(x) = \frac{2px[(p+1)+(2p-1)x^p]}{(2p^2-3p+1)+(4p^2+2p-2)x^p+(p+1)x^{2p}},$$

$$r_{22}(x) = \frac{x[(2p^2+3p+1)+(8p^2-2)x^p+(2p^2-3p+1)x^{2p}]}{(2p^2-3p+1)+(8p^2-2)x^p+(2p^2+3p+1)x^{2p}}.$$

Next, we focus on the properties of the *principal* Padé iterations for the matrix sector function:

Recall that the principal Padé iterations only correspond to the Padé approximants $[r/r]$. Therefore the principal Padé iteration for the scalar sector function $s_r(z)$, generated by the Padé approximant, is given by the following, see Section 3 of [7]: For $l \geq 0$,

$$x_{l+1} = x_l \frac{\sum_{j=0}^{r} b_j^{(r)} x_l^{pj}}{\sum_{j=0}^{r} c_j^{(r)} x_l^{pj}} = r_{rr}(x_l) = x_l \frac{f_{rr}(1-x_l^p)}{g_{rr}(1-x_l^p)}, \tag{3.36}$$

where $x_0 = z$ and

$$b_j^{(r)} = (-1)^j \sum_{l=j}^{r} \binom{l}{j} f_l^{(rr)} = \binom{r}{j} \frac{r!}{(2r)!p^r} \prod_{l=r-j+1}^{r} (lp-1) \prod_{l=j+1}^{r} (lp+1), \tag{3.37}$$

$$c_j^{(r)} = (-1)^j \sum_{l=j}^{r} \binom{l}{j} g_l^{(rr)} = \binom{r}{j} \frac{r!}{(2r)!p^r} \prod_{l=j+1}^{r} (lp-1) \prod_{l=r-j+1}^{r} (lp+1). \tag{3.38}$$

### 3.2.3 Matrix $p$-th root ($p \in \mathbb{N}^*$)

It is assumed that the reader is familiar with the scalar $p$-th root function (i.e. $f(x) = \sqrt[p]{x}$). Note that the case when $p = 2$ had already been discussed in the last section, in this section we will consider the case for $p > 2$. Analogous to the scalar $p$-th root function, we can also define the matrix $p$-th root, which is used for defining the matrix sector function. The existence of $p$-th root of a matrix $A$ is given in the following theorem, see section 7.1, Theorem 7.3 of [2] for details:

**Theorem 3.8 (existence of $p$-th root)** *$A \in \mathbb{C}^{n\times n}$ has a $p$-th root iff the "ascent sequence" of integers $d_1, d_2, ...$ defined by*

$$d_i = dim(null(A^i)) - dim(null(A^{i-1})),$$

*has the property that for each integer $\nu \geq 0$ no more than one element of the sequence lies strictly between $p\nu$ and $p(\nu + 1)$.*

The uniqueness of the principal $p$-th root of a matrix $A$ is explained in the next theorem, see section 7.1, Theorem 7.2 of [2] for details:

**Theorem 3.9 (principal $p$-th root)** *See Section 7.1 of [2]. Let $A \in \mathbb{C}^{n\times n}$ have no eigenvalues on $\mathbb{R}^-$. There exists a unique $p$-th root $X$ of $A$ for which all of whose eigenvalues lie in the segment $\{z : -\pi/p < arg(z) < \pi/p\}$, and it is called the primary matrix function of $A$. Here $X$ is called the principal $p$-th root of $A$ and denote $X = A^{1/p}$. Note that if $A$ is real then $A^{1/p}$ is also real.*

For justifications of the above two theorems, see Section 7.1 of [2].

As appeared in Chapter 2, we can also define matrix $p$-th root function via Jordan canonical form or via Cauchy integral:

1. (Classification of $p$-th roots via Jordan canonical form) Let the nonsingular matrix $A \in \mathbb{C}^{n\times n}$ have a Jordan canonical form $P^{-1}AP = D = diag(D_1, D_2, ..., D_m)$, with $D_k = D_k(\lambda_k)$, and let $s \leq m$ be the number of distinct eigenvalues of $A$. Let $L_k^{(j_k)} = L_k^{(j_k)}(\lambda_k)$ for $k = \{1, ..., m\}$, denote the $p$ $p$-th roots given by (2.5), where $j_k \in \{1, 2, ..., p\}$ denotes the branch of the $p$-th root function. Then $A$ has $p^s$ $p$-th roots that are primary functions of $A$, which are given by

$$X_j = Pdiag(L_1^{(j_1)}, L_2^{(j_2)}, ..., L_m^{(j_m)})P^{-1}, \qquad j \in \{1, ..., p^s\},$$

corresponding to all possible choices of $j_1, ..., j_m$, subject to the constraint that $j_i = j_k$ when $\lambda_i = \lambda_k$. They form parametrized families

$$X_j(U) = PUdiag(L_1^{(j_1)}, L_2^{(j_2)}, ..., L_m^{(j_m)})U^{-1}P^{-1}, \qquad j \in \{p^s + 1, ..., p^m\},$$

where $j_k \in \{1, 2, ..., p\}$, $U$ is any arbitrary nonsingular matrix that commutes with $J$, and for each $j$ there exists $i$ and $k$, depends on $j$, such that $\lambda_i = \lambda_k$ with $j_i \neq j_k$.

2. Integral representation of $A^{1/p}$

$$A^{1/p} = \frac{p\sin(\pi/p)}{\pi} A \int_0^\infty (t^p I + A)^{-1} dt \tag{3.39}$$

This equation holds for all real $p > 1$, see Section 7.1 of [2].

### 3.2.4 Padé iteration for the matrix $p$-th root

It is possible to deduce Padé iteration for the principal $p$-th root from any iteration for the scalar $p$-sector function. Note that $x_{l+1} = x_l h(x_l)$ ($h$ is a function of $x_l$) for $x_0 \in \Phi_j$ where $\Phi_j = \{z \in \mathbb{C}\backslash\{0\} : |arg(z) - 2j\pi/p| < \pi/p\}$, for $j \in \{0, ..., p-1\}$ converges to $s_p(x_0)$ iff

$$z_{l+1} = z_l h(x_0 z_l), \qquad z_0 = 1, \tag{3.40}$$

converges to $\epsilon_j x_0^{-1} = (x_0^p)^{-1/p}$. If $h(z)$ is of the form

$$h(z) = \frac{\omega_1(z^p)}{\omega_2(z^p)}, \tag{3.41}$$

where $\omega_1$ and $\omega_2$ are polynomials, then the iteration (3.40) with $x_0 = a^{-1/p}$ is

$$z_{l+1} = z_l \frac{\omega_1(a^{-1} z_l^p)}{\omega_2(a^{-1} z_l^p)}, \qquad z_0 = 1. \tag{3.42}$$

Note that we can also multiply both $\omega_1$ and $\omega_2$ by a suitable power of $a$ to eliminate the inverse of $a$. In addition note that the Padé family is of the form (3.41).

From the Padé family for the matrix sector function, we are able to derive a family of iterations for the matrix $p$-th root

$$X_{l+1} = X_l F_{ab}(I - A^{-1} X_l^p) G_{ab}(I - A^{-1} X_l^p)^{-1}, \qquad X_0 = I,$$

where $R_{ab} = F_{ab}/G_{ab}$ and $R_{ab}$ denotes the rational approximation of the matrix $p$-th root $A^{1/p}$. It can be shown that the above iterations are numerically unstable, see Section 5 of [7].

Another kind of iteration for the matrix $p$-th root is as follows: let $Y_l = A^{-1}X_l^p$, then (see Section 5 of [7]):

$$Y_{l+1} = A^{-1}X_{l+1}^p = A^{-1}X_l^p[F_{ab}(I - A^{-1}X_l^p)]^p[G_{ab}(I - A^{-1}X_l^p)]^{-p}$$

$$= Y_l[F_{ab}(I - Y_l)(G_{ab}(I - Y_l))^{-1}]^p.$$

Let $h(t) = F_{ab}(1-t)/G_{ab}(1-t)$, we obtain the following coupled iteration for the matrix principal $p$-th root

$$\begin{aligned} X_{l+1} &= X_l h(Y_l), X_0 = I, \\ Y_{l+1} &= Y_l h(Y_l)^p, Y_0 = A^{-1}. \end{aligned} \tag{3.43}$$

Note that if the eigenvalues of $A$ are within the convergence region, then, by solving the coupled iteration (3.43), we get:

$$\lim_{l\to\infty} X_l = A^{1/p}, \qquad \lim_{l\to\infty} Y_l = I. \tag{3.44}$$

Note that the iteration (3.43) is stable, see Section 5 of [7] for more details.

Several algorithms exist to approximate the matrix $p$-th root. For more details on other algorithms for computing the $p$-th root, refer to [8] and Chapter 4.

# Chapter 4

# Fractional linear systems

There exists several direct and iterative algorithms for solving fractional linear systems. In the following submitted paper, we study numerically some representative methods (both direct and iterative) for computing fractional linear systems (FLS) $A^{\alpha}\mathbf{x} = \mathbf{b}$, where $A \in \mathbb{C}^{n\times n}$ or $A \in \mathbb{R}^{n\times n}$, and $\alpha \in \mathbb{Q}^{*}$, $\mathbf{x}$ and $\mathbf{b}$ are $1 \times n$ vectors.

# A numerical study of fractional linear algebraic system solvers

Emmanuel LORIN[a,b], Simon TIAN[a]

[a] *School of Mathematics and Statistics, Carleton University, Ottawa, Canada, K1S 5B6*
[b] *Centre de Recherches Mathématiques, Université de Montréal, Montréal, Canada, H3T 1J4*

**Abstract**

This paper is devoted to the study of numerical methods for solving large, sparse or full *fractional linear algebraic systems* (FLAS). The intent is to provide relevant and fair accuracy and efficiency comparisons of several solvers for this type of linear systems, typically involved in the approximation of fractional partial differential equations.

*Keywords:* Cauchy integral; Padé's approximants; Ordinary differential equation solver; Preconditioner; $p$th roots; GMRES; Fractional differential equations.

**Contents**



## 1. Introduction

This paper is dedicated to the numerical computation of the rational power of *large* matrices, and primarily to solutions to *large* fractional linear algebraic systems (FLAS) $A^\alpha x = b$, for $\alpha \in \mathbb{Q}\backslash\{0\}$. This work is largely motivated by the numerical approximation of fractional differential equations, even more specifically partial differential equations involving fractional Laplace operators. As a consequence, along the paper the matrix $A$ will often represent finite dimensional approximate

*Email addresses:* elorin@math.carleton.ca (Emmanuel LORIN), SimonTian3@cmail.carleton.ca (Simon TIAN)

Laplace operators, including external potentials, with or without nonlocal random perturbations. Although, there exists an extensive literature on this question from the theoretical point of view or for "small" matrices, the efficiency of those methods for large systems is rarely discussed which motivates the study presented in this paper. We are interested in the accuracy and complexity of several "standard" or "straightforward" methods applied to matrices with different structures. We propose in particular to focus on the $p$th root of a matrix, for which there exists a very solid literature, see for instance [13, 14, 6, 26, 24, 4, 16, 11], and which constitutes a key problem for our primary objective, that is solving fractional linear systems. The purpose of this paper is hence not to present yet another method, but rather to propose as much as possible, a fair and instructive comparison between several existing methods when dealing with "large" matrices. In particular, we compare the performance of the methods on sparse, full, real, complex matrices and for various values of $\alpha \in \mathbb{Q}\backslash\{0\}$. In the following, we will recall the *theoretical* computational complexity for computing matrix $p$th roots and fractional linear systems, assuming that the matrices and linear systems are *full*, that is usually requiring $O(N^3)$ operations. However from a practical point of view, for sparse matrices the genuine complexity for solving linear algebraic systems is rather $O(N^\beta)$ for some $1 < \beta \leqslant 3$ using standard iterative methods [21]. This question will be addressed in Section 3. In this paper, we will

- recall some standard FLAS solvers and their basic mathematical properties. These solvers will be classified as *direct* (when they are based on the approximation of analytical -but complex- solutions to the FLAS) or "iterative" (when they are based on approximate sequences converging to the solution to the FLAS).
- recall some ODE-solver and Cauchy integral preconditioning techniques, which were recently proposed in [4], [3].
- provide rigorous comparisons of the performances of all the studied solvers for different types of matrices (sparse, full, real, complex).
- provide some comparisons regarding the performance depending on the value of the parameter $\alpha$ involved in the FLAS.
- propose some conclusions regarding the performance of the solvers based on our study.

Overall, we will show that the ODE-based method possesses the best performance (with a lot a flexibility in the choice of the ODE-solver), in particular combined with a ODE-solver preconditioning technique (Subsection 2.3). The Cauchy integral method also shows good performances, but seems sensitive to the choice of the contour (shape, how close to the spectrum,...) which is problematic when the spectrum is not localized in $\mathbb{C}$. More generally, on the proposed tests, the *direct* methods (ODE, Cauchy, Padé) will show better performances than the chosen *iterative* ones; the latter all showing relatively similar performances.

### *1.1. Motivation and framework*

This work is motivated by the growing interest in the scientific community in fractional differential equations [20, 8, 15, 1, 19]. The latter is due in particular, to the development of fractional PDE models (when dealing with nonlocal phenomena) in several fields of application (quantum

physics, mechanics, epidemiology, applied probability,...) which hence requires efficient computational methods. A key example is the fractional Poisson equation on $\mathbb{R}^d$ [20], such that on bounded domain $\Omega \subset \mathbb{R}^d$ $(d = 1, 2, 3)$, we consider

$$\begin{array}{rcl} -(-\triangle)^{\alpha} u & = & f, \text{ in } \Omega, \\ u & = & 0, \text{ on } \mathbb{R}^d \backslash \Omega, \end{array} \tag{1}$$

where $f \in L^p(\Omega)$, $1 < p < \infty$, and where the fractional Laplacian can be defined [8] for $\alpha \in (0, 1)$ and any $u \in \mathcal{S}(\mathbb{R}^d)$, as

$$(-\triangle)^{\alpha} u(\boldsymbol{x}) = C(\alpha) \text{p.v.} \int_{\mathbb{R}^d} \frac{u(\boldsymbol{x}) - u(\boldsymbol{y})}{|\boldsymbol{x} - \boldsymbol{y}|^{2+2\alpha}} d\boldsymbol{y} = C(\alpha) \lim_{\varepsilon \to 0^+} \int_{\mathbb{R}^d \backslash B_{\varepsilon}(\boldsymbol{x})} \frac{u(\boldsymbol{x}) - u(\boldsymbol{y})}{|\boldsymbol{x} - \boldsymbol{y}|^{2+2\alpha}} d\boldsymbol{y},$$

where $B_{\varepsilon}(\boldsymbol{x})$ is the ball of radius $\varepsilon$ and center $\boldsymbol{x}$, $C(\alpha)$ is the constant defined by

$$C(\alpha) \ := \ \Big( \int_{\mathbb{R}^d} \frac{1 - \cos(\xi_1)}{|\boldsymbol{\xi}|^{2+2\alpha}} d\boldsymbol{\xi} \Big)^{-1}, \tag{2}$$

and p.v. denotes the principal value. Despite the complexity of this nonlocal operator, an approximate solution $u_h$ can be performed by solving

$$A_h^{\alpha} u_h \ = \ f_h \, , \tag{3}$$

where $A_h$ (resp. $f_h$) is finite dimensional approximation of $-\triangle$ (resp. $f$) on $\Omega$, which greatly simplifies the approximation of (1). For a smooth function $\varphi$, one gets

$$A_h \varphi = \triangle \varphi + \mathcal{O}\big(h^q R_1(\phi)\big) \, .$$

Using spectral decomposition arguments (see [20], Subsection 2.5), we indeed get

$$A_h^{\alpha} \varphi = \triangle^{\alpha} \varphi + \mathcal{O}\big(h^{q\alpha} R_{\alpha}(\varphi)\big),$$

with $R_1$ and $R_{\alpha}$ some smooth differential operators. A finite difference/volume discretization of the latter, hence requires the solution to large fractional algebraic linear systems. This matrix power based idea was also used in [5] within a computational methods for solving fractional Schrödinger and diffusion equations. Global random perturbations on the matrix $A_h$ will also be considered, thus leading to a full matrix. The latter hence corresponds to an approximate randomly perturbed fractional Laplace operators. Notice that $(-\triangle)^{\alpha}$ can no more be directly approximated by $A_h^{\alpha}$ when other boundary conditions are used; see again [20] for details.
In this paper, we will focus on the efficiency and rate of convergence of different algorithms for i) computing the rational power of a matrix, and ii) the solution to fractional algebraic linear systems.

We will consider different types of matrices (sparse, full, complex, *etc*) and will provide comparisons as rigorous as possible from the computational complexity point of view, in order to compute for $\alpha \in \mathbb{Q} \backslash \{0\}$, the solution to

$$A^{\alpha} x \ = \ b,$$

with $A \in \mathbb{R}^{N \times N}$ or $\in \mathbb{C}^{N \times N}$ having a spectrum in $\mathbb{C} \backslash \mathbb{R}_-$. From a practical point of view, we rewrite $\alpha = \lfloor \alpha \rfloor + q/p$ with $p, q \in \mathbb{N} \backslash \{0\}$ and $q < p$, so that we formally have for instance

$A^\alpha = A^{\lfloor\alpha\rfloor}A^{1/p}\cdots A^{1/p}$. As a consequence, we propose to focus on the $p$th root of $A$, and the solution to

$$A^{1/p}x = b\,,$$

and compare different strategies/algorithms for solving the corresponding fractional linear systems. We recall that for $k \geqslant 0$, and any $\alpha \in \mathbb{R}^*$ the *Cauchy integral* reads

$$A^\alpha \;=\; (2\pi\mathtt{i})^{-1}A^k\int_{\Gamma_A} z^{\alpha-k}(zI-A)^{-1}dz\,, \tag{4}$$

where $\Gamma_A$ is a closed contour in the complex plane *enclosing the spectrum of the matrix* $A$, $I$ is the identity matrix in $\mathbb{R}^{N\times N}$ and $\mathtt{i} = \sqrt{-1}$. From a computational point of view, the choice of the contour is of important matter, as we will see in Subsection 2.1 (see **Cauchy integral method**) and is also discussed in details in [3]. In another context (eigenvalue computation), this is a question which is also addressed in [25].

In this paper, two main types of methods will be considered: *direct* and *iterative methods*. Direct methods, such as the Cauchy integral based method (Algorithm 8), will refer as non-iterative algorithms approximating the exact definition of the $p$th root, or of the exact solution to fractional linear systems. Iterative methods, such as the Newton method (Algorithm 2), will instead provide a convergent sequence of approximate matrix $p$th roots, or approximate solutions to fractional linear systems. We recall the following fundamental results on the existence and uniqueness of principal $p$th roots [13].

**Theorem 1.1.** *$A \in \mathbb{C}^{N\times N}$ has a pth root if and only if the sequence of integers $d_1, d_2, \cdots$ defined by*

$$d_i = dim(null(A^i)) - dim(null(A^{i-1}))$$

*is such that for each integer $\nu \geqslant 0$, no more than one element of the sequence lies strictly between $p\nu$ and $p(\nu+1)$.*

The uniqueness of the principal pth root of a matrix $A$ is addressed in the next theorem:

**Theorem 1.2 (principal $p$th root).** *Let $A \in \mathbb{C}^{N\times N}$ with no eigenvalues in $\mathbb{R}^-$. There exists a unique pth root $X$ of $A$ for which all of those eigenvalues lie in the segment $\{z : -\pi/p < arg(z) < \pi/p\}$, and it is a primary matrix function of $A$. Moreover if $A$ is real then $A^{1/p}$ is also real.*

At this stage, we assume that $A \in \mathbb{K}^{N\times N}$, where the field $\mathbb{K}$ is either $\mathbb{C}$ or $\mathbb{R}$, and such that the spectrum of $A$ belongs to $\mathbb{C}\backslash\mathbb{R}_-$ in order to avoid the line of singularity. We also assume that *a priori* $N$ is "large", that is such that computing the full spectrum of $A$ is numerically inaccessible. The complexity on large systems was not necessarily addressed in the literature, we then also propose a discussion on this aspect for the different algorithms. Finally, let us discuss a fundamental question regarding the relevance of using iterative methods computing matrix $p$th roots for solving fractional linear algebraic systems. Indeed, the requirement of solving (generalized) Sylvester's equations make their use for solving fractional linear systems questionable. However, as we will see below, the other standard methods which are recalled below, are also very computational complex, as they all require to solve a very large number of intermediate linear systems. This hence

motivate a rigorous comparison. A detailed discussion on the outcomes of the experiments, will be proposed in Section 3.
We next recall the basics of different algorithms for computing the $p$th root of a matrix $A$, as well as their main properties from the complexity and convergence rate (accuracy) points of view.

*1.2. Direct algorithms for computing $A^{1/p}$, $p \in \mathbb{N}^*$*

The matrix $A$ is a $N \times N$ matrix which is assumed to be non-singular. The chosen angle is to focus on the relative computational complexity depending on the type of matrices. Let us also recall that an explicit knowledge of the spectrum $\mathrm{Sp}(A) := \{\lambda_k\}_{1 \leqslant k \leqslant N}$ of the matrix $A$ would of course leads to an efficient computation of $A^{1/p}$. Assuming that the transition matrix $P_A$ and diagonal matrix $\Lambda_A$ are explicitly known ($A = P_A \Lambda_A P_A^{-1}$), then using for instance the residue theorem of the Cauchy integral (4), one gets

$$A^\alpha = \sum_{k=1}^{n} \mathrm{Res}\big(z^\alpha (zI - A)^{-1}, \lambda_k\big) = P_A^{-1} \sum_{k=1}^{n} \mathrm{Res}\big(z^\alpha (zI - \Lambda_A)^{-1}, \lambda_k\big) P_A = P_A^{-1} \sum_{k=1}^{n} D_A^{(k)} P_A,$$

where $D_A^{(k)} = \{d_{A;ij}^{(k)}\}_{1 \leqslant i,j \leqslant N}$, and

$$d_{A;jj}^{(k)} \;=\; \begin{cases} \lambda_j^\alpha & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases}, \qquad d_{A;ij} = 0, \;\; \text{if } i \neq j\,.$$

Obviously, we have $\Lambda_A = \sum_{k=1}^{n} D_A^{(k)}$. In this paper, we will exclude this situation, which would make formally "trivial" the computation of the solution to fractional linear systems, but which is not realistic for large matrices.

**Schur's method** [24]. Schur's method for computing the $p$th root of a matrix is a direct method which consists in i) a quasi-triangularization, that is Schur's decomposition in a block-triangular matrix [9], ii) an iterative $p$th root computations on $1 \times 1$ or $2 \times 2$ matrices and iii) the construction of the $p$th root of $A$ thanks to unitary transition matrices. We present in Algorithm 1 the full algorithm as proposed in [24, 13]. This approach is of the same type as a diagonalization, although naturally much less computationally complex. Whenever the matrix is small or sparse enough, we can expect that this will be an efficient method. In total, Schur's algorithm requires $O(pN^3)$ operations.
**Other direct algorithms.** Exact expressions of the $p$th root of matrix can be formally obtained using the Cauchy integral approximation [3] (Subsection 2.1), or Padé's approximants at least when $p = 2$. Although, the latter would be appropriate for solving fractional linear systems, they are however not suitable for computing a matrix $p$th root, as the corresponding algorithms make appear inverse matrices which can naturally not be computed for large matrices. We hence do exclude those methods for now, be we will use them in Subsection 2.1, where they actually become relevant. Beyond Schur's decomposition or diagonalization, other decompositions could theoretically be naturally be used.

*1.3. Iterative algorithms for computing $A^{1/p}$, $p \in \mathbb{N}^*$*

In this section, we present several iterative methods for computing the $p$th root of a matrix. There exists many variants of the presented methods, we then select standard ones.

**Algorithm 1** Schur's method for $A^{1/p}$

1: Schur's decomposition: $A = QTQ^T$, where $T$ a quasi-triangular matrix [9] where the block $T_{ii}$ are either $m \times m$ matrix with $m = 1$ or 2.

2: Computation of $U_{ij}$ for $1 \leqslant i, j \leqslant m$. For each $j = 1, m$

- $U_{jj} = T_{jj}^{1/p}$
- $V_{jj}^{(k)} = T_{jj}^{k+1}$, for $-1 \leqslant k \leqslant p-2$
- Computation of $U_{ij}$. For $1 \leqslant i, j-1$
  - $B_k = \sum_{l=i+1}^{j-1} U_{il} V_{lj}^{(k)}$ for $0 \leqslant k \leqslant p-2$
  - $U_{ij}$ solution to

$$\sum_{k=0}^{p-1} V_{ii}^{(p-2-k)} U_{ij} V_{jj}^{(k-1)} \quad = \quad T_{ij} - \sum_{k=0}^{p-2} V_{ii}^{(p-3-k)} B_k \,,$$

  - where $V_{ij}^{(k)} = \sum_{l=0}^{k} V_{ii}^{(k-l-1)} U_{ij} V_{jj}^{l-1} + \sum_{l=0}^{k-1} V_{ii}^{(k-2-l)} B_l$ for $0 \leqslant k \leqslant p-2$

3: $A^{1/p} = QUQ^T$.

**Newton's method** [13, 16]. The most simple and standard iterative method for computing a matrix $p$th root is Newton's method. We denote by $X_0$ the initial guess; typically $X_0$ is chosen as $X_0 = A$ or $I$. The principal $p$th root $A^{1/p}$ of $A$ whose eigenvalues lie in the sector $S_p = \{z \in \mathbb{C}\backslash\{0\}, \pi/p < \arg z < \pi/p\}$ reads as follows, see Algorithm 2. Notice that the choice

**Algorithm 2** Newton's method I for $A^{1/p}$

1: $X_0$ such that $X_0 A = A X_0$

2: Until convergence:

$$X_{k+1} = \frac{(p-1)X_k + X_k^{1-p} A}{p}.$$

of $X_0$ such that $X_0 A = A X_0$ is crucial to solve the generalized Sylvester equations with a cubic complexity (using Schur's method). Indeed, for say $p = 2$, the algorithm also reads: for $X_k$ and $A$ given, find $X_{k+1}$ such that

$$X_k X_{k+1} + X_{k+1} X_k \quad = \quad X_k^2 + A \,,$$

which is a standard Sylvester equation (find $X$ such that $CX + XC = D$ for given matrices $C$, $D$). An improved version is also proposed in [16]. In particular, it is proven that for $X_0 = I$, the method is convergent for any matrix with eigenvalues in $\{z \in \mathbb{C},\, \mathrm{Re} z > 0,\, |z| \leqslant 1\}$.

Each iteration requires $O(\log(p)N^3)$ operations (using binary power technique for computing $A^k$, $k \geqslant 2$), and has *quadratic* convergence [16] for eigenvalues of modulus less than 1, when $X_0 = I$.

**Coupled Newton method.** In [13, 10] was proposed a coupled Newton methods, also referred as a Newton-Schur method, which allows for stability improvement. The algorithm is recalled in Algorithm 3 and is convergent to $A^{-1/p}$ (take $p \leftarrow -p$ to get $A^{1/p}$).

**Algorithm 3** Coupled Newton method for $A^{1/p}$

1: $X_0 = I$ and $M_0 = A$
2: Until convergence:

$$\begin{aligned} X_{k+1} &= X_k\Big(\frac{(p-1)I + M_k}{p}\Big), \\ M_{k+1} &= \Big(\frac{(p-1)I + M_k}{p}\Big)^{-p} M_k\,. \end{aligned}$$

Hence, the method is quadratically convergent to $A^{-1/p}$ for initial guess $X_0 = I/c$ and $c \in \mathbb{R}_+$ if all the eigenvalues of $A$ belong to the set

$$E(c,p) \;=\; \mathrm{conv}\big\{\{z \,:\, |z - c^p| \leqslant c^p\}, (p+1)c^p\big\}\backslash\{0, (p+1)c^p\}\,.$$

We again refer to [10].

**Padé-based method for $A^{1/p}$.**

The method referred below as Padé-based method, consists in using standard Padé's approximants computing the power of complex numbers [4]; the latter can also be used to efficiently compute a matrix $p$th root. For $p \in \mathbb{N}^*$, an *iterative sequence* of approximations converging to $A^{1/p}$ can easily be derived. However, it is well known that for $p = 2$, an exact solution to the sequence of approximations can be obtained [2]. In this latter case, we can then derive a *direct* method. $\theta$-rotating Padé's approximants for $p = 2$ are recalled hereafter, see also [4],

$$\sqrt{z} \approx p_m^{(1/2)}(z) = \sum_{k=0}^{m} a_k^{(m)} - \sum_{k=1}^{m} \frac{a_k^{(m)} d_k^{(m)}}{z + d_k^{(m)}},$$

where the coefficients are given, for $\theta \in [0, \pi/2)$, by

$$a_0^{(m)} = 0, \qquad a_k^{(m)} = \frac{e^{\mathtt{i}\theta}}{m\cos^2\Big(\dfrac{(2k+1)\pi}{4m}\Big)}, \qquad d_k^{(m)} = e^{\mathtt{i}\theta}\tan^2\Big(\frac{(2k+1)\pi}{4m}\Big)\,. \tag{5}$$

Hence, we formally have

$$A^{-1/2} \;=\; \lim_{m\to+\infty} A^{-1}\Big(\sum_{k=0}^{m} a_k^{(m)} I - \sum_{k=1}^{m} a_k^{(m)} d_k^{(m)} (A + d_k^{(m)} I)^{-1}\Big)\,.$$

Although relatively simple this Padé-based method is in fact useless from a practical point of view in order to compute $A^{1/2}$. Indeed, the latter requires the computation of inverse matrices which is

not practically tractable for large matrices. It is then more suitable to use an iterative algorithm. However, this approach will naturally become interesting when solving fractional linear systems $A^{1/2}x = b$. When $p$ is not equal to 2, as far as we know, there does not exist a simple explicit expression of Padé's approximants. However, there exists some iterative algorithms allowing for constructing sequences of approximations of matrix $p$th roots. It is numerically observed that convergence occurs in the set $\{z \in \mathbb{C} : |1 - z^p| < 1\}$. Reference on convergence of Padé iterations can be found in [13, 17, 18]. The algorithm that we propose is Algorithm 4.

**Algorithm 4** Padé-based method for $A^{1/p}$

1: Set $X_0 = I$ and $Y_0 = A^{-1}$

2: For $k \in \{0, \cdots, m\}$, solve

$$\begin{aligned} X_{k+1} &= X_k h(Y_k) \\ Y_{k+1} &= Y_k h(Y_k)^p \end{aligned}$$

with

$$h(z) = \frac{P_{lm}(1-z)}{Q_{lm}(1-z)}, \tag{6}$$

and for $(\alpha)_k = \alpha(\alpha+1)...(\alpha+k-1), (\alpha)_0 = 1$

$$\begin{aligned} P_{lm}(z) &= \sum_{j=0}^{l} \frac{(p^{-1})_j (p^{-1}-m)_m (j-l-m)_m}{j!(-l-m)_m (j+p^{-1}-m)_m} z^j \\ Q_{lm}(z) &= \sum_{j=0}^{l} \frac{(-m)_j (-p^{-1}-l)_j}{j!(-l-m)_j} z^j \,, \end{aligned} \tag{7}$$

where is used the notation

$$(\alpha)_j = \alpha(\alpha+1)\cdots(\alpha+k-1), \qquad (\alpha)_0 = 1\,.$$

3: At convergence $X_k \to A^{1/p}$.

**Tsai's method.** The last iterative algorithm that we present is Tsai's Algorithm 5, see [26] which is one of the oldest algorithms for computing matrix $p$th roots. The method is numerically stable and the convergence is again shown to be quadratic [26].

As this paper is primarily devoted to solving linear systems, we do not propose an exhaustive comparison of the above algorithms for the $p$th root computation, see [13]. We rather propose to compare the performance of the 3 iterative methods for computing $A^{1/4}$ and $A^{1/2}$, where $A \in \mathbb{R}^{N\times N}$ is a 5-point finite difference approximation of $-\triangle + V$, that is $A$ is penta-diagonal matrix with real and positive eigenvalues.

**Experiment 1.** We illustrate the convergence of the proposed methods Fig. 1 (Left), and their CPU-time as a function of $N$ with a threshold error in matrix norm $\|A^{1/p} - A_h^{1/p}\|_2$ less or equal to $10^{-3}$ for $p = 2$ and $p = 4$ in Fig. 1 (Right). For the 3 methods, the convergence is reached very

**Algorithm 5** Tsai's method for $A^{1/p}$

1: $G_0 = A$ and $R_0 = I$
2: Until convergence

$$\begin{aligned} G_{k+1} &= G_k\Big((2I+(p-2)G_k)(I+(p-1)G_k)^{-1}\Big)^p \\ R_{k+1} &= R_k\Big((2I+(p-2)G_k)^{-1}(I+(p-1)G_k)\Big). \end{aligned}$$

The approximate solution is given by $R_k$.

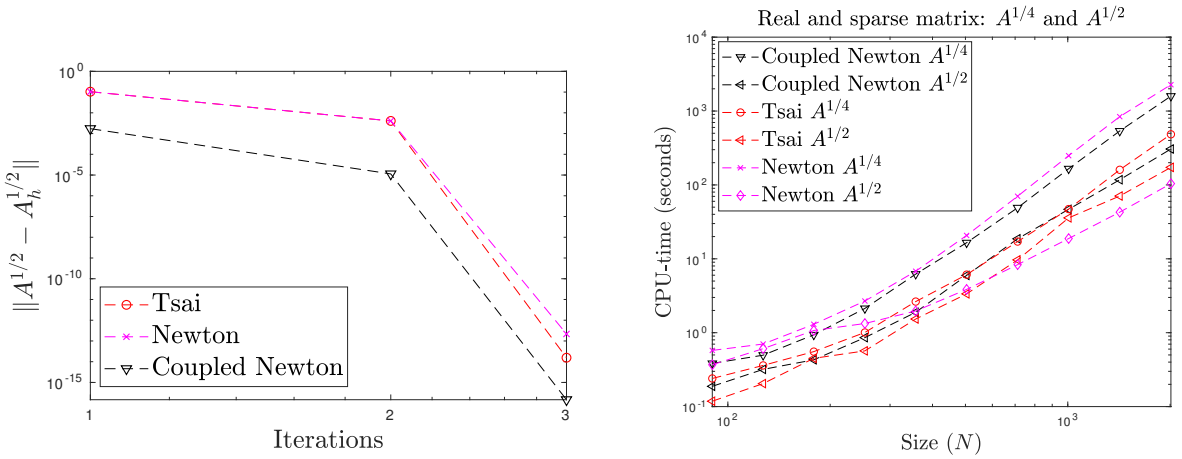


Figure 1: **Experiment 1.** (Left) Convergence as function of iterations. (Right) The corresponding CPU-time as function of $N$

rapidly in term of number of iterations, although each iteration is obviously very computationally complex (Sylvester's equation). It is interesting to notice that the efficiency is relatively different for the 3 methods, but as expected the larger $p$ the slower the convergence of these iterative methods, as they then require to solve "more" linear systems.

### *1.4. Organization*

This rest of the paper is organized as follows. Section 2 is devoted to the numerical computation of solutions to fractional linear algebraic systems, using direct (Subsection 2.1) and iterative (Subsection 2.2) algorithms. Some ad-hoc preconditioning methods are presented in Subsections 2.3 and 2.4. In Subsection 2.5, we then propose a series of numerical experiments dedicated to the comparison of the different strategies proposed in the paper. Finally, in Section 3 we propose a discussion and arguments on the relative performances of the different methods used in the paper, and provide some concluding remarks and perspectives.

## 2. Computational methods for FLAS

As explained above, the solution to $A^\alpha x = b$, for $\alpha = \lfloor \alpha \rfloor + q/p$ with $p, q \in \mathbb{Q}\backslash\{0\}$ and $q < p$, can be solved iteratively for $k = 1, \cdots, q-1$

$$A^{\lfloor \alpha \rfloor} x_0 = b, \qquad A^{1/p} x_{k+1} = x_k\,.$$

As a consequence, we mainly focus on the solution to $A^{1/p}x = b$. We propose two types of algorithms: direct and iterative ones. Although the linear systems involved in all the methods will be systematically solved using an iterative solver (GMRES [23]), the direct terminology refers to the way the type of solvers to achieve the solutions to the fractional linear systems.

*2.1. Direct Algorithms*

We recall in this subsection several direct algorithms for solving $A^{1/p}x = b$, with $A$ having its spectrum in $\mathbb{C}\backslash\mathbb{R}_-$.

**Differential equation solver for $A^\alpha x = b$.** [7]. The differential approach is particularly convenient as it allows for a direct computation of the solution to any fractional linear system with $\alpha \in \mathbb{R}\backslash\{0\}$. In this case, we directly consider the solution to $A^\alpha x = b$, for $\alpha \in \mathbb{R}\backslash\{0\}$. Considering the $N$-dimensional differential system

$$x'(\tau) = -\alpha(A-I)\big(I+\tau(A-I)\big)^{-1}x(\tau), \qquad x(0) = b \in \mathbb{R}^N\,, \tag{8}$$

admits a solution $x(\tau) = \big(I+\tau(A-I)\big)^\alpha b$ such that $x(1) = A^{-\alpha}b$, where $I$ is the identity matrix in $\mathbb{R}^{N\times N}$ (see [6, 7, 12, 14, 13]). By replacing $\alpha$ by $-\alpha$ in (8), we can compute $x = A^{-\alpha}b$ through (8). Let us introduce $\{\tau_k\}_{0\leqslant k\leqslant K_A}$ as a finite sequence of uniformly sampled discrete time steps such that $\tau_k = k\Delta\tau_A$, with $0 \leqslant k \leqslant K_A$ and setting $\tau_{K_A} = 1$. For $\alpha \in \mathbb{R}_+\backslash\{0\}$, we propose to approximate $A^{-\alpha}b$ iteratively by discretizing (8) first with a second-order *Crank-Nicolson* (CN) scheme

$$\begin{aligned} x_{k+1} \;=\; & x_k - \frac{\Delta\tau_A}{2}\alpha(A-I)\big(I+\tau_k(A-I)\big)^{-1}x_k \\ & -\frac{\Delta\tau_A}{2}\alpha(A-I)\big(I+\tau_{k+1}(A-I)\big)^{-1}x_{k+1}\,, \end{aligned}$$

where $x_k$ is an approximation of $x(\tau_k)$ solution to (8) and $\Delta\tau_A$ is the time step. This leads to the 3-steps CN algorithm. At each iteration $k$, $0 \leqslant k \leqslant K_A - 1$, we need to solve two linear systems using traditional *iterative solvers combined with preconditioners*. At the final time $\tau_{K_A} = 1$, there exists a positive constant $C > 0$ such that we have the following error bound

$$\|x_{K_A} - A^{-\alpha}b\|_2 \leqslant C\Delta\tau_A^2\,,$$

where $\|v\|_2 = \big(\sum_{j=1}^N |v_j|^2\big)^{1/2}$, for any vector $v \in \mathbb{C}^N$. Regarding the computational complexity, it is easy to see that each iteration requires the solution to 2 linear systems with this order 2 ODE solver. The corresponding algorithm is detailed in Algorithm 6. We also present an order 4 Runge-Kutta solver

$$\|x_{K_A} - A^{-\alpha}b\|_2 \leqslant C\Delta\tau_A^4\,,$$

in Algorithm 7. As the RK4 method is explicit, but naturally still requires solution to linear systems, it allows for better performances than CN for a fixed precision.

Hence, the overall algorithm, using an order $s$ ODE solver ($s = 2$ for CN, $s = 4$ for RK4), has a complexity given by $O(sK_AN^3)$.

In order to improve the convergence and efficiency, it was proposed in [4] to introduce a so-called *ODE-solver preconditioning*, which allows for increasing the size of the time step in the ODE solver, then reducing $K_A$ with the same precision. It was analytically and numerically shown in [4], that

**Algorithm 6** Crank-Nicolson method for $A^\alpha x = b$

1: Compute $z_k$ solution to $\big(I + \tau_k(A-I)\big)z_k = x_k$.

2: Set $w_k := x_k - \dfrac{\Delta\tau_A}{2}\alpha(A-I)z_k$, and calculate $\omega_{k+1}$ such that

$$\big(I + \tau_{k+1}(A-I)\big)\omega_{k+1} \;=\; w_k\,.$$

3: Compute $y_{k+1}$ solution to

$$(I + \tau_{k+1+\frac{\alpha}{2}}(A-I))y_{k+1} \;=\; \omega_{k+1}\,.$$

4: Deduce $x_{k+1}$ solution to

$$x_{k+1} \;=\; (I + \tau_{k+1}(A-I))y_{k+1}\,.$$

5: Deduce $x_{K_A}$ approximates $A^{-\alpha}b$.

**Algorithm 7** Runge-Kutta method for $A^\alpha x = b$

1: Compute $w_{1,k}$ solution to

$$\begin{aligned}\big(I + \tau_k(A-I)\big)z_{1,k} &= x_k\,,\\ w_{1,k} &= -\alpha\Delta\tau_A(A-I)z_{1,k}\,.\end{aligned}$$

2: Compute $w_{2,k}$ solution to

$$\begin{aligned}\big(I + \tau_{k+1/2}(A-I)\big)z_{2,k} &= x_k + z_{1,k}/2\,,\\ w_{2,k} &= -\alpha\Delta\tau_A(A-I)z_{2,k}\,.\end{aligned}$$

3: Compute $w_{3,k}$ solution to

$$\begin{aligned}\big(I + \tau_{k+1/2}(A-I)\big)z_{3,k} = x_k + z_{2,k}/2\,,&\\ w_{3,k} &= -\alpha\Delta\tau_A(A-I)z_{3,k}\,.\end{aligned}$$

4: Compute $w_{4,k}$ solution to

$$\begin{aligned}\big(I + \tau_{k+1/2}(A-I)\big)z_{4,k} = x_k + z_{3,k}\,,&\\ w_{4,k} &= -\alpha\Delta\tau_A(A-I)z_{4,k}\,.\end{aligned}$$

5: Compute $x_{k+1}$

$$x_{k+1} \;=\; \frac{1}{6}\big(w_{1,k} + 2w_{2,k} + 2w_{3,k} + w_{4,k}\big)\,.$$

6: Deduce $x_{K_A}$ approximates $A^{-\alpha}b$.

it was hence possible to drastically improve the efficiency of this approach, see also Subsection 2.3.

**Cauchy integral method [3].** The most natural and direct method for solving $A^\alpha x = b$, is based on the Cauchy integral definition of the matrix power (4) taking $\alpha = 1/p$. Let us also mention that the method proposed in this paper, has also some close connections with the matrix-transfer method which was proposed in [27, 28]. From a practical point of view, the contour integral is numerically computed by using a quadrature rule leading to the formal approximate matrix computation (for $k = 1$ in (4))

$$A_h^\alpha := (2\pi \mathtt{i})^{-1} A \sum_{1 \leqslant j \leqslant J_A} h_j w_j z_j^{\alpha-1} (z_j I - A)^{-1}\,, \tag{9}$$

where $\{w_j\}_{1 \leqslant j \leqslant J_A}$ are the quadrature weights and $\{z_j\}_{1 \leqslant j \leqslant J_A}$ the integration nodes along the integral contour. Notice that preliminary computations are required to initially determine a contour enclosing the matrix spectrum. For real spectra, the most simple contour is a "thin" (in imaginary direction) rectangle constructed from the smallest and largest matrix eigenvalues. For complex spectra, by defaults, the simples approach consists in considering a circular contour centered at the origin and of radius given by the largest eigenvalues. The largest and smallest eigenvalues are typically obtained by a power-like and inverse power-like methods [22]. We refer to [3] for details. It was numerically noticed that the precision is unfortunately quite sensitive to the choice of contour; see also [25] in an eigenvalue problem framework. The local discretization steps of the path are denoted by $h_j$, and $h = \max_{1 \leqslant j \leqslant J_A} h_j$. In matrix norm, the order of convergence $\sigma$ is such that

$$\left\| A_h^\alpha - (2\pi \mathtt{i})^{-1} A \int_{\Gamma_A} z^{\alpha-1} (zI - A)^{-1} dz \right\| \leqslant C h^\sigma\,.$$

From this result, we can deduce that formally, the solution to $A^\alpha x = b$ can be expressed, taking $\alpha$ by $-\alpha$ in (9), by

$$x_h = (2\pi \mathtt{i})^{-1} A \sum_{1 \leqslant j \leqslant J_A} h_j w_j z_j^{\alpha-1} (z_j I - A)^{-1} b\,.$$

The algorithm is presented in Algorithm 8.

**Algorithm 8** Cauchy integral-based method for $A^\alpha x = b$

1: A rectangular contour, $z_j \in \Gamma_A^{(h)}$ and $z_{j+1} = z_j + h_{j+1}$, with $h_j = \delta x_j + \mathtt{i}\delta y_j$. Denoting $(z_j I - A)^{-1} f = u_j$.

2: We solve for any $j$

$$(z_j I - A) u_j \;\; = \;\; f, \text{ for all } 1 \leqslant j \leqslant J_A\,, \tag{10}$$

3: Then reconstruct the approximate solution

$$x_h \;\; := \;\; S_h^{(-\alpha)} f = (2\pi \mathtt{i})^{-1} \sum_{1 \leqslant j \leqslant J_A} h_j w_j z_j^{-\alpha} u_j\,. \tag{11}$$

Taking an order $\sigma$ quadrature, we hence get

$$\|x_h - x\| \leqslant Ch^\sigma \,.$$

Overall the computational complexity of the method is hence $O(\sigma J_A N^3)$ where $J_A$ is the number of nodes of an order $\sigma$ quadrature along the contour enclosing the matrix spectrum. In order to improve the convergence and efficiency, it was proposed in [3] different types of preconditioners $M$. Basically, the idea consists in using a Cauchy integral preconditioning allowing for a reduction of the length of the Cauchy integral contour enclosing the spectrum of the preconditioned matrix $MA$. It was analytically and numerically shown in [3], that it is possible to drastically improve the efficiency of this approach, see also Subsection 2.4 for details.

**Padé-based method for $A^{1/2}x = b$.** As recall in the previous section, Padé's approximants can efficiently be estimated when $\alpha = 1/2$ [4]. That is for any vector $b \in \mathbb{K}^N$, we formally have

$$A^{-1/2}b \;=\; \lim_{m\to+\infty} \Big( \sum_{k=0}^{m} a_k^{(m)} I - \sum_{k=1}^{m} a_k^{(m)} d_k^{(m)} (A + d_k^{(m)} I)^{-1} \Big) A^{-1} b \,.$$

where the sequence $\{a_k^{(m)}\}_k$ and $\{d_k^{(m)}\}_k$ are defined in (5).
Then, we deduce the Algorithm 9.

---

**Algorithm 9** Padé-based method for $A^{1/2}x = b$

---

1: For $k \in \{0, \cdots, m\}$, solve

$$Av_b = b, \qquad (A + d_k^{(m)} I) w_k = v_b \,.$$

2: Approximate solution $x_m$ is given by

$$x_m \;=\; \sum_{k=0}^{m} a_k^{(m)} v_b - \sum_{k=1}^{m} a_k^{(m)} d_k^{(m)} w_k \,.$$

---

$$\|x_m - x\| \;=\; \Big\| \Big( \sum_{k=m+1}^{\infty} a_k^{(m)} I - \sum_{k=1}^{m} a_k^{(m)} d_k^{(m)} (A + d_k^{(m)} I)^{-1} \Big) A^{-1} b \Big\| \,.$$

This error estimate is naturally strongly dependent on the spectrum and condition number of $A$. The computational cost of the Padé-based approach is hence if $O(mN^3)$ due again to the requirement of solutions to linear systems.

Whenever, $p \neq 2$, as noticed above the construction of Padé's approximants is much more complex. A possible approach consists in solving constructing $A^{-1/p}b$, where $p$ is replaced by $-p$ in the Padé algorithm. Basically,

$$\left\{ \begin{array}{rcl} x_{k+1} & = & X_k h(Y_k) b \\ y_{k+1} & = & Y_k h(Y_k)^{-p} b \end{array} \right.$$

where $h$ is defined in (6) and (7) and hence requires the solution to several linear systems. This iterative method was not numerically tested below.

*2.2. Iterative algorithms*

We now present the iterative algorithms for solving fractional linear systems which are directly deduced from the algorithms from Subsection 1.3.

**Newton's method.** Newton's algorithm for solving fractional linear systems corresponding to Algorithm 2 reads as follows, see Algorithm 10.

---

**Algorithm 10** Newton's method I for $A^{1/p}x = b$

---

1: $X_0$ such that $X_0 A = A X_0$

2: Iterate

$$X_{k+1} = \frac{(p-1)X_k + X_k^{1-p}A}{p}.$$

2: Determine $x_{k+1}$ approximate solution to $A^{1/p}x = b$, by solving

$$X_{k+1}x_{k+1} \quad = \quad b\,.$$

---

**Tsai's method for** $A^{1/p}x = b$ Tsai's algorithm for solving fractional linear systems corresponding to Algorithm 5 reads as follows, see Algorithm 11.

---

**Algorithm 11** Tsai's method for $A^{1/p}$

---

1: $G_0 = A$ and $R_0 = I$

2: Iterate

$$\begin{aligned} G_{k+1} &= G_k\Big((2I + (p-2)G_k)(I + (p-1)G_k)^{-1}\Big)^p \\ R_{k+1} &= R_k\Big((2I + (p-2)G_k)^{-1}(I + (p-1)G_k)\Big)\,. \end{aligned}$$

2: Determine $x_{k+1}$ approximate solutions to $A^{1/p}x = b$, by solving

$$R_{k+1}x_{k+1} \quad = \quad b\,.$$

---

*2.3. ODE-based solver preconditioning*

In the comparison tests which are performed in Subsection 2.5, it will be shown that the ODE-based solvers are the most efficient and flexible, even if they naturally do not allow for the computation of matrix powers. In this subsection, we propose a simple way to greatly improve the efficiency of the ODE-based fractional linear system solver using a special type of preconditioning, see [4]. In particular, *Preconditioned Crank-Nicolson* (PCN) and *Preconditioned Runge-Kutta 4* (PRK4) methods based on the scaling matrix $M = I/\|A\|_2$, where $\|A\|_2$ is the 2-norm of a matrix $A$, are introduced. The latter consists in the most simple preconditioning for diagonally dominant matrices, but more complex ones were developed in [4]. For Matrix $A$, and with $M$ a corresponding scaling matrix, we trivially have the identity $A^{-\alpha} = M^{\alpha}(MA)^{-\alpha}$. We can then compute the finite sequence $\{x_k^{(MA)}\}_{0\leqslant k\leqslant K_{MA}}$ of vector solutions, setting $\tau_{K_{MA}} = \tau_{K_A} = K_{MA}\Delta\tau_{MA} = 1$, following the PCN algorithm.

**Step 1.** Compute $z_k^{(MA)}$ such that: $\big(I+\tau_k(MA-I)\big)z_k^{(MA)} = x_k^{(MA)}$.

**Step 2.** Define $w_k^{(MA)} := x_k^{(MA)} - \dfrac{\Delta\tau_{MA}}{2}\alpha(MA-I)z_k^{(MA)}$, and obtain $\omega_{k+1}^{(MA)}$ as the solution to the preconditioned linear system

$$\big(I+\tau_{k+1}(MA-I)\big)\omega_{k+1}^{(MA)} \;=\; w_k^{(MA)}\,.$$

**Step 3.** Compute $z_{k+1}^{(MA)}$ solution to

$$(I+\tau_{k+1+\frac{\alpha}{2}}(MA-I))z_{k+1}^{(MA)} \;=\; \omega_{k+1}^{(MA)}\,.$$

**Step 4.** Deduce $x_{k+1}^{(MA)}$ solution

$$x_{k+1}^{(MA)} \;=\; (I+\tau_{k+1}(MA-I))z_{k+1}^{(MA)}\,.$$

Each of these linear systems is solved by using an *iterative method*. At the final time $\tau_{K_{MA}}$, we get an estimate of $(MA)^{-\alpha}b$, and therefore we deduce $x = M^\alpha(MA)^{-\alpha}b$ through

$$M^\alpha x_{K_{MA}}^{(MA)} \;\approx\; M^\alpha(MA)^{-\alpha}b = A^{-\alpha}b\,.$$

More precisely there exists a constant $C>0$ such that:

$$\|M^\alpha x_{K_{MA}}^{(MA)} - A^{-\alpha}b\|_2 \leqslant C\Delta\tau_{MA}^2\,.$$

In particular, from [4], we have:

**Proposition 2.1.** *Let us consider the DCN and PCN algorithms, where we assume that the linear systems are solved exactly. Then, there exists a constant $c>0$ such that*

$$\|x_{K_A}^{(A)} - A^{-\alpha}b\|_2 \;\leqslant\; c\Delta\tau_A^2\frac{e^{\alpha(\rho(A)-1)}-1}{\alpha}.$$

*Moreover, denoting by $\rho(\cdot)$ the spectral radius function, we have the following estimates:*

1. *If we assume that $\rho(MA-I)>1$, then we have*

$$\|M^\alpha x_{K_{MA}}^{(MA)} - A^{-\alpha}b\|_2 \;\leqslant\; c\Delta\tau_{MA}^2\frac{e^{\alpha(\rho(MA)-1)}-1}{\alpha}.$$

.

2. *If $1>\rho(MA-I)>\varepsilon$, for some $\varepsilon\in(0,1)$, one gets*

$$\|M^\alpha x_{K_{MA}}^{(MA)} - A^{-\alpha}b\|_2 \;\leqslant\; c\Delta\tau_{MA}^2\frac{e^{\alpha(1-\varepsilon)^{-1}}-1}{\alpha}.$$

This proposition actually justifies the use of an ODE solver preconditioning for solving fractional linear systems. Indeed, although the preconditioning does not change the order of convergence of the solver, it allows for a reduction of the prefactor in the error estimates, which now depends on the spectral radius of the preconditioned matrix $MA$.

**Experiment 2.** Let us solve the system $A^\alpha x = b$, where $A = B + B^T + D \in \mathbb{R}^{N\times N}$, with $N = 512$

and $\alpha = 3/4$. The matrix $B = \{b_{ij}\}_{1 \leqslant ij \leqslant N}$ is such that $b_{ij} = \texttt{rand}(0,1)$, and $D = \{d_{ij}\}_{1 \leqslant ij \leqslant N}$ is the diagonal matrix defined by $d_{ii} = 20 + \texttt{rand}(0,1)$, $1 \leqslant i \leqslant N$. The right-hand side $b$ is given in $\mathbb{R}^N$. We report in Fig. 2 (Right) the convergence graph of $\|x - x_h\|_\infty = \|A^{-\alpha}b - A_h^{-\alpha}b\|_\infty$, where $A^{-\alpha}b$ is the exact solution to the linear system, and $x_h = A_h^{-\alpha}b$ (the index $h$ refers to as an approximate value of $x = A^{-\alpha}$) is computed by solving (8) with a Direct CN (DCN) scheme, Direct RK4 (DRK4) scheme, and scaled PCN and PRK4 (Preconditioned RK4) schemes, as described in Algorithms 6 and 7. In Fig. 2 (Left), we report the spectrum of $A$ and $MA$. These results show the efficiency of the scaling technique for accelerating the computation of $A^{-\alpha}b$. In particular, we notice the fourth- (resp. second-)order convergence of the RK4 (resp. CN) schemes. The scaling approach is more specifically interesting for efficiently evaluating $A^{-\alpha}b$.

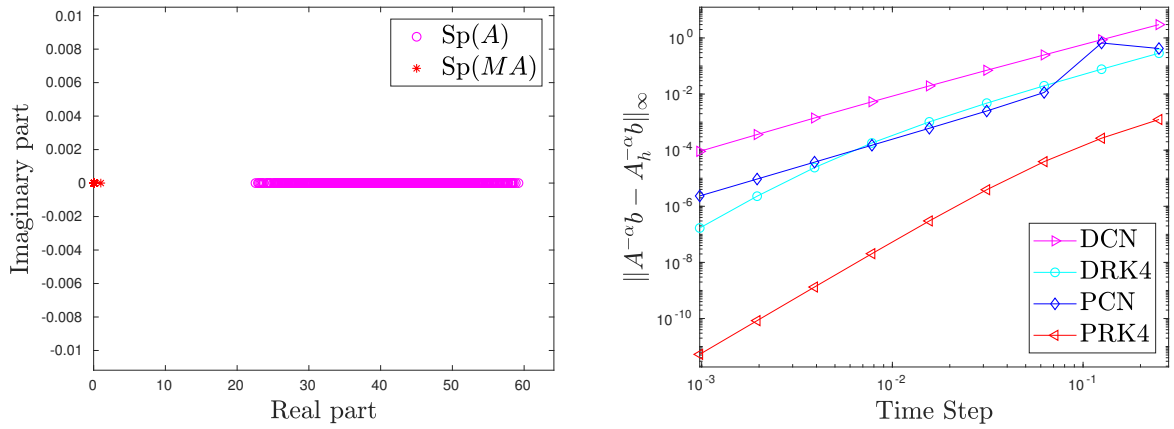


Figure 2: **Experiment 2.** (Left) Spectrum of $A$ and $MA$. (Right) Convergence rate of the DCN/DRK4 and scaled PCN/PRK4 schemes.

**Experiment 2bis.** This test is similar to the previous one except that $A = B + D \in \mathbb{R}^{N \times N}$ ($N = 400$), which has a complex spectrum with eigenvalues having positive real parts, see Fig. 3 (Left). The results are reported in Fig. 3 (Right), where we again observe an improvement of the convergence of the preconditioned ODE-solvers vs the non-preconditioned ones.

### *2.4. Cauchy integral preconditioning*

In the tests which are performed below, it will also be shown that the Cauchy integral solvers can be efficient. In this subsection, we propose to present a simple way to greatly improve the efficiency of the Cauchy integral solver. The *Cauchy integral preconditioning* strategy allows for a drastic reduction of the length of the integral contour (4), then leading to a much faster quadrature than with a direct computation of $A^\alpha x = b$. A Cauchy integral preconditioner is a matrix $M$ such that, assuming again that the spectrum of $MA$ lies in $\mathbb{C}\backslash\mathbb{R}_-$ with

$$(MA)^\alpha \;=\; (2\pi\mathtt{i})^{-1} MA \int_{\Gamma_{MA}} z^{\alpha-1}(zI - MA)^{-1} dz\,, \tag{12}$$

where we expect that $\ell(\Gamma_{MA}) \ll \ell(\Gamma_A)$, $\ell$ denoting the length of a curve in the complex plan. Typically, $M$ is chosen as a preconditioner for solving the linear system $Ax = b$, i.e. $M \approx A^{-1}$.

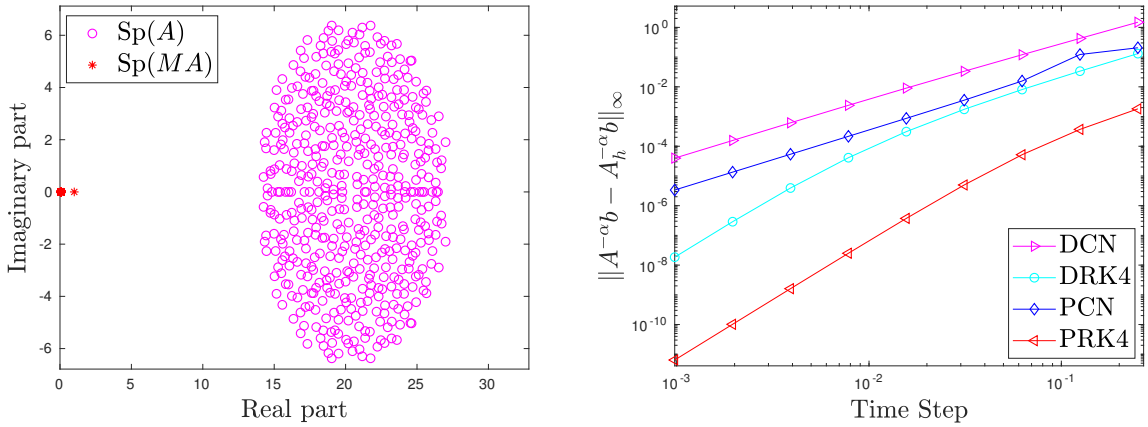


Figure 3: **Experiment 2bis.** (Left) Spectrum of $A$ and $MA$. (Right) Convergence rate of the DCN/DRK4 and scaled PCN/PRK4 schemes.

However, additional constraints need to be added. The integral preconditioner of interest is two-fold [3]

1. *clustering of the spectrum of the preconditioned matrix* $MA$,
2. *accurate estimate of the center of the spectrum of* $MA$, *more specifically* 1.

Getting a shorter integration path for the Cauchy integral, hence reduces the cost of the numerical quadrature used to approximate the Cauchy integral. Computing (4) from (12) is expected to be more efficient than with a direct computation. The simplest Cauchy integral preconditioner is a scaling matrix. The latter may be of limited interest, but in some cases can still be highly efficient. It simply consists in defining $M = I/\|A\|_2$. Hence, we have

$$A^\alpha \quad = \quad M^{-\alpha}(MA)^\alpha\,. \tag{13}$$

**Experiment 3.** To illustrate the above ideas, let us consider the matrix $A = B + 0.25B^T + C$, where $B_{ij} N\texttt{rand}(0,1)$ and $C_{ij} = 20N + \texttt{rand}(0,1)\delta_{ij}$, $1 \leqslant i,j \leqslant N$, for $N = 100$. The matrix $A$ has a complex-valued spectrum. For $\alpha = 3/4$, the spectrum of $A$, $\mathrm{Sp}(A)$ is reported as well as the enclosing rectangular $\Gamma_A$ and circular $\mathcal{C}_A$ contours in Fig. 4 (Left) and a zoom on the spectrum of $MA$, $\mathrm{Sp}(MA)$, and some enclosing rectangular $\Gamma_{MA}$ and circular $\mathcal{C}_{MA}$ contours are reported in Fig. 4 (Right). In Fig. 5), we then observe a great improvement in convergence using the Cauchy integral preconditioning, see also [3].

### *2.5. Numerical comparisons*

In the following, we finally propose to numerically compare the convergence rate, and the computational complexity (via the CPU-time estimate) for different types of matrices (sparse, full, real, complex). All the numerical methods which are proposed require solutions to linear systems. The latter are solved using GMRES [23] with tolerance of $10^{-8}$ and restart at 20 iterations.

**Experiment 4. : Real sparse matrix.** In this case, we propose to solve $A^\alpha x = b$, where $A \in \mathbb{R}^{N\times N}$ is a 5-point finite difference approximation of $-\triangle + V$ with non-constant $V$ (with

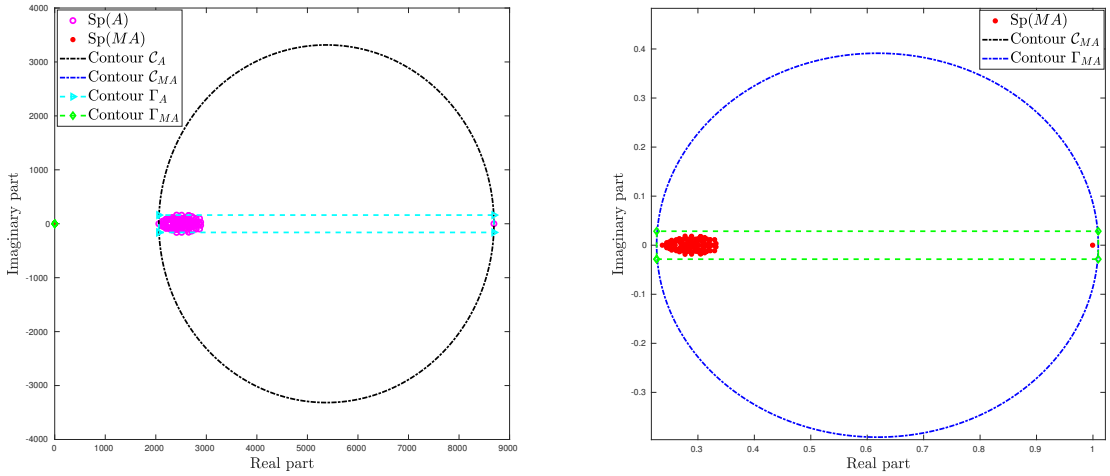


Figure 4: **Experiment 3.** (Left) Spectra Sp($A$), Sp($MA$) with $A \in \mathbb{R}^{N\times N}$, $\mathcal{C}_A$, $\mathcal{C}_{MA}$ and $\Gamma_A$, $\Gamma_{MA}$. (Right) Zoom on Sp($MA$).

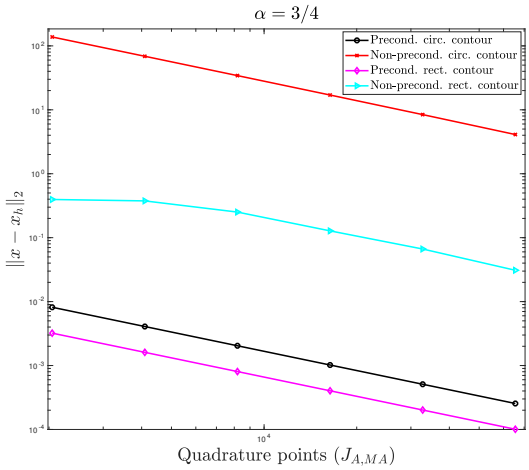


Figure 5: **Experiment 3.** Relative error ($\alpha = 3/4$) vs the number of integration points.

$V_j = \exp(-(jh)^2)$ with $j = -N/2, \cdots, N/2-1$ and $h = 2/(N-1)$), that is $A$ is (sparse) penta-diagonal matrix with real and positive eigenvalues. Finally, $b$ is taken as the identity vector. We illustrate the convergence of the proposed methods, and their appropriate relative computational complexity (by computing CPU-times in seconds). In Fig. (6) (Left) taking $N = 1024$, we report the $\ell^2$-norm error as function of the time steps for $p = 2$, illustrating an order 2 convergence for CN and the order 4 convergence for RK4. The corresponding CPU-time is reported in Fig. 6 (Right). We report the CPU-time as a function of $N$ for the CN and RK4 methods and a threshold error $\|x - x_h\|_2 \leqslant 10^{-3}$.

We next report in Fig. 8 (Left) the spectrum and the rectangular contour used in the Cauchy

integral approach, again for $N = 1024$. In Fig. 8 (Right), we present the graph of convergence for the Cauchy integral approach. For Padé-based method with $p = 2$ and $N = 1024$, we report in Fig.

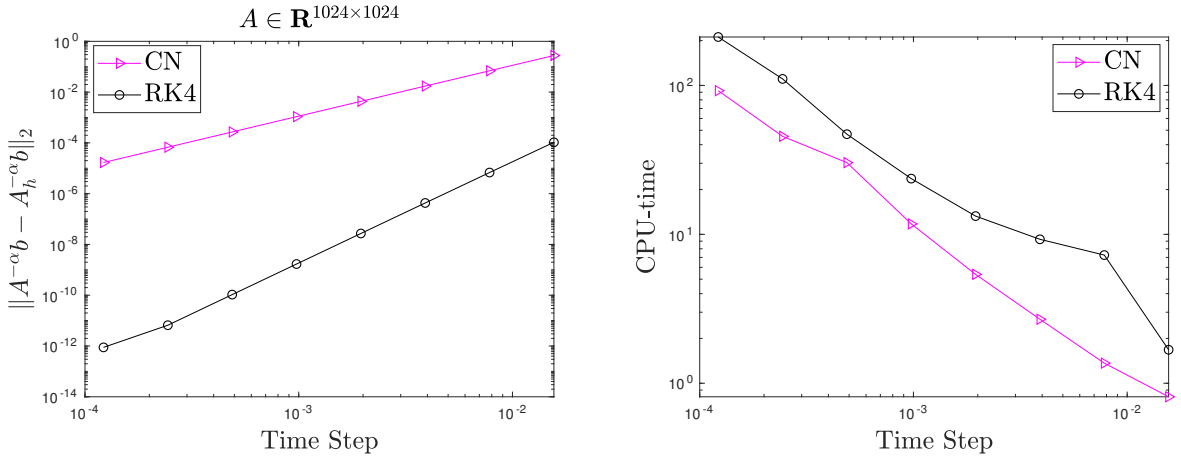


Figure 6: **Experiment 4.** (Left) Convergence rate for CN and RK4: $\ell^2$-norm error as function time steps. (Right) The corresponding CPU-time as function of $N$.

7 the $\ell^2$-norm error as function of the number of Padé's approximants.

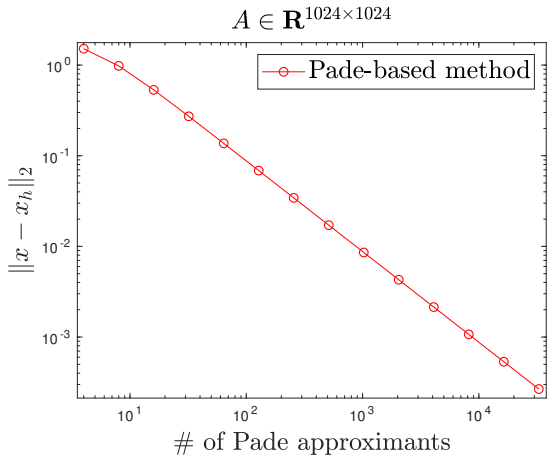


Figure 7: **Experiment 4.** Convergence rate for Padé.

**Experiment 5.** We next compare the relative efficiency of the different approaches for $p = 2$, and a $\ell^2$-norm error $\|x - x_h\|_2$ less or equal to $10^{-3}$. More specifically, we report the CPU-time (in seconds) for 6 different methods, namely Cauchy, Padé, Newton, coupled Newton, Tsai and RK4, in Fig. (9). The best results are obtained with the RK4 method, while the iterative methods are less efficient, which is not surprising as the latter require the computation of matrix powers. In Fig. 10, we similarly compare the CPU-time in seconds using the RK4, Cauchy integral and Newton methods for $A^{1/4}x = b$, $A^{1/2}x = b$, $A^{3/4}x = b$. The conclusion is the same regarding the efficiency of the RK4 solver for which the fastest convergence is observed for $\alpha = 1/4$, while Newton's method

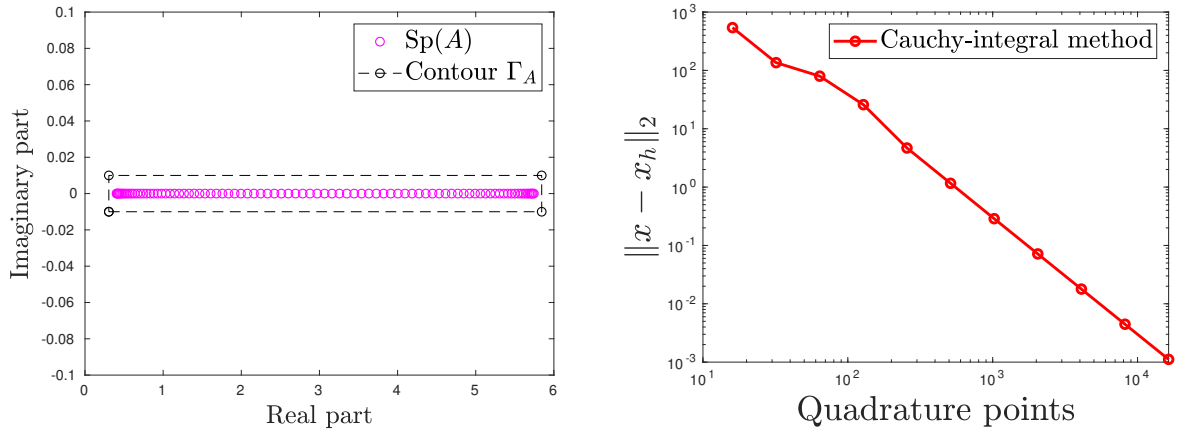


Figure 8: **Experiment 4.** (Left) Spectrum and Cauchy integral contour. (Right) Convergence for Cauchy integral method.

is shown to converge more rapidly at $\alpha = 1/2$.

**Experiment 6. : Full real and complex matrix.** In this section, we introduce a real symmetric (resp. complex) random perturbation $10^{-2}\texttt{rand}(N, N)$ of the above sparse matrix $A$ approximating the Laplace operator with potential. The matrix is hence no more sparse, and then expect a higher computational complexity. The corresponding respective CPU-times are reported in Fig. 11 (Top) (resp. (Bottom)). Again, the ODE-based method is the most efficient. The Newton-based (including Tsai's) iterative methods are yet the less efficient.

## 3. Discussion and concluding remarks

The series of tests which was performed above, allows us to reach some conclusions regarding the relative performances of the different fractional linear system solvers. In those tests, all the intermediate linear systems were solved using the same Krylov-type solver (GMRES), with the same parameters. No preconditioners were used in order to reduce as much as possible, bias in the comparisons which would occur by involving additional parameters. Although the theoretical computational complexity of the linear system solver is cubic, practically, especially for sparse matrices, the effective complexity can be as "small as linear". For this reason, in the following we denote by $\beta$ the order of complexity for solving a linear system depending on its sparsity; thus $1 < \beta \leqslant 3$. Regarding the performance of fractional linear system computations, Newton's, coupled Newton's and Tsai's methods have a relatively poor efficiency relatively to the others, expect for "small" matrices. Overall the RK4 methods was the most efficient in all the tested cases. For full real or complex matrices, the Cauchy-integral approach was shown to be very efficient as well, in particular for large matrices. Cauchy integral preconditioning allows for even more efficient performances, see [3]. Notice however, that Cauchy integral approach is relatively sensitive to the contour choice. It is observed that the number of quadrature points usually need to be taken very large, thus requiring the solution to numerous linear systems. For sparse matrices, Padé's approximant-based method showed some good performances when $\alpha = 1/2$. However it was observed that for full matrices,

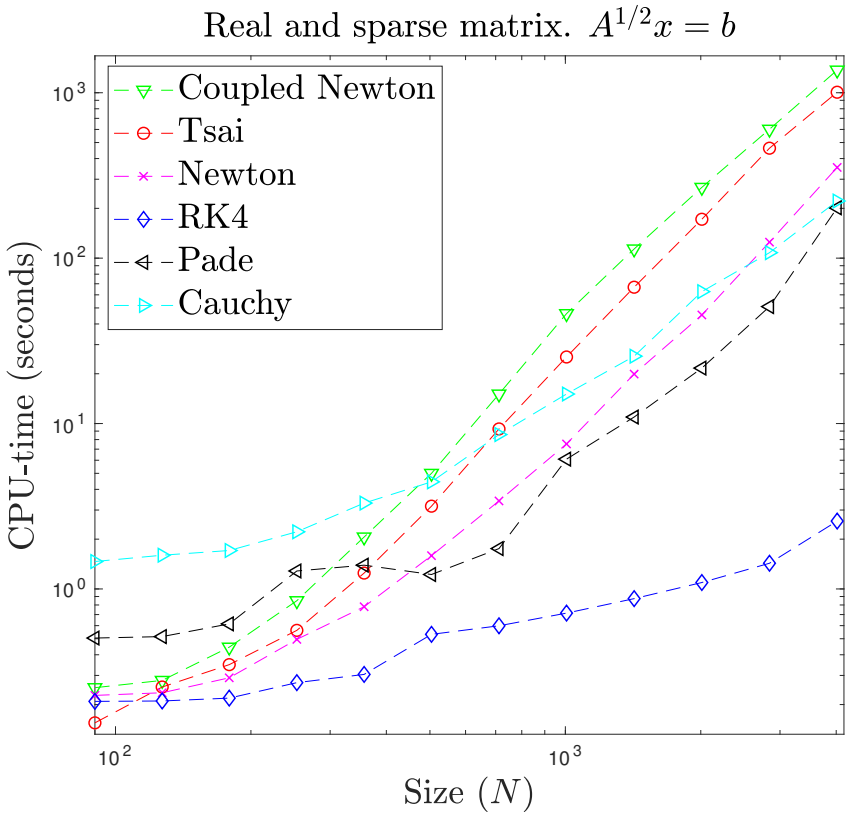


Figure 9: **Experiment 5.** CPU-time in seconds using RK4, Cauchy integral, Tsai, Newton, coupled Newton, Padé methods for $p = 2$, for $A^{1/2}x = b$ with $A$ sparse and real matrix.

this approach was not that efficient anymore. These results can actually be explained using the following arguments. Unlike the iterative methods, intermediate parameters need to be fixed in the methods referred *as direct*. More specifically, the Cauchy integral method requires i) to construct a contour enclosing the spectrum of matrix $A$, then, ii) to implement a quadrature to approximate the Cauchy integral. At each quadrature node, standard linear systems must hence be solved. Efficiency and accuracy of the approximation is hence dependent on the choice of the contour [3] and of the quadrature parameters (order, number of nodes). Similarly, the ODE-based method is dependent on the choice of i) the order of the solver and ii) of the time-steps. Interestingly, it is thanks to this flexibility that it was possible to develop *ad-hoc* preconditioning techniques [4, 3], beyond standard linear system preconditionings. This latter remark is also linked to the fundamental relevance of using $p$th root matrix solvers for computing the solution to a fractional linear algebraic system. Basically, direct Padé-based method has a complexity $O(mN^\beta)$, an order $s$ ODE-based solver has a complexity $O(sK_AN^\beta)$, and Cauchy integral method using an order $\sigma$ quadrature has a complexity $O(\sigma J_AN^\beta)$, while generalized Sylvester's equation solvers used in the iterative methods have a complexity $O(N^{1+\beta})$. Hence, the key point is then the relative values of $sK_A/N$, $\sigma J_A/N$, and $m/N$ which is linked to the matrix spectrum or its condition number. If a large matrix $A$ has a spectrum which is concentrated in a small region (small condition number), we can precisely

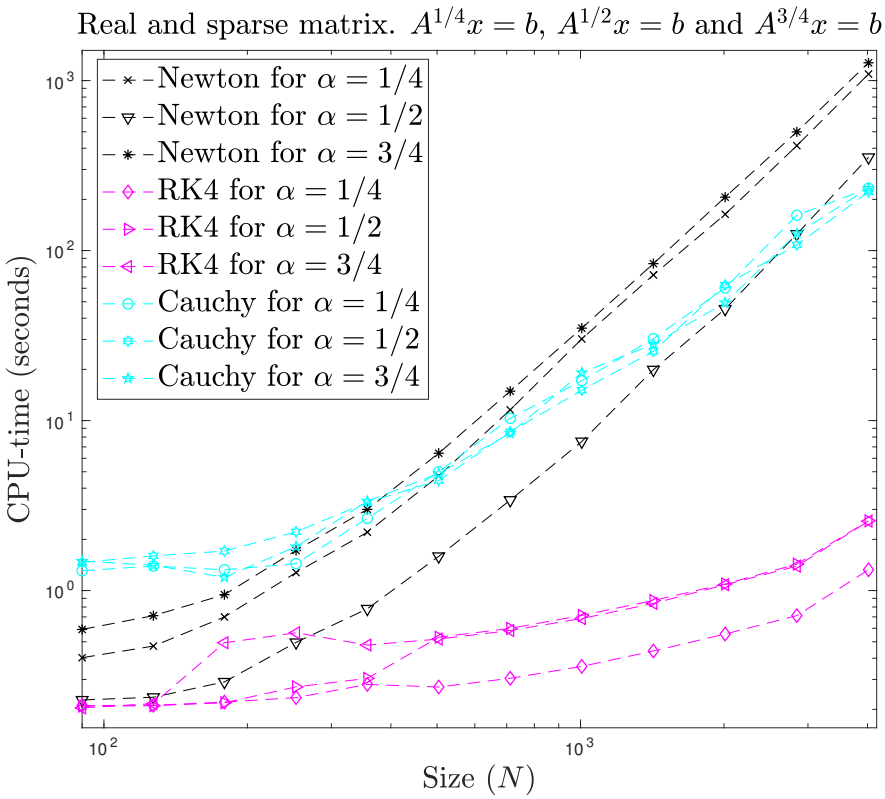


Figure 10: CPU-time in seconds using RK4, Cauchy integral, Newton for $q/p = 1/4,\ 1/2,\ 3/4$ with $A$ sparse and real matrix.

compute the Cauchy integral (resp. ODE-based) method with $\sigma J_A \ll N$ (resp. $sK_A \ll N$). Similar conclusions hold for the direct Padé-based method, although this is more technical to justify. Whenever, the spectrum has a very large condition number, $\sigma J_A$ can easily be larger than $N$, hence making the Cauchy (resp. ODE-based) approach relatively less efficient. Preconditioning, see [3] and Subsection 2.4 (resp. [4] and Subsection 2.3) allows for forcing a concentration of the spectrum of the preconditioned matrix $MA$, so that $J_{MA} \ll N$ (resp. $K_{MA} \ll N$). These arguments also justify that for $N$ not too large, iterative Newton-like methods can be more efficient than direct ones (Cauchy, Padé, ODE).

In conclusion, although a fully exhaustive comparison is difficult, we have proposed a numerical study which allowed us to conclude that the ODE-based solvers are the most efficient and flexible for solving FLAS. In fact, the coupling of the ODE-solver with a preconditioning (ODE-solver preconditioner) is even more efficient. In a forthcoming paper, we will be interested in the generalization of the performance study to sparse and full systems of the form $f(A)x = b$.

## References

[1] L. Aceto and P. Novati. Efficient implementation of rational approximations to fractional differential operators. *J. Sci. Comput.*, 76(1):651–671, 2018.

[2] X. Antoine, C. Besse, and P. Klein. Absorbing boundary conditions for the two-dimensional Schrödinger equation with an exterior potential. Part I: Construction and *a priori* estimates. *Math. Models Methods Appl. Sci.*, 22(10):1250026, 38, 2012.

[3] X. Antoine and E. Lorin. Double-preconditioning for fractional linear systems. application to fractional Poisson equations. *Submitted*, 2020.

[4] X. Antoine and E. Lorin. ODE-based double-preconditioning for solving linear systems $A^\alpha x = b$ and $f(A)x = b$. *Revision Numer. Lin. Alg. with App.*, 2020.

[5] X. Antoine, E. Lorin, and Y. Zhang. Derivation and analysis of computational methods for fractional laplacian equations with absorbing layers. *Numer. Algo.*, To appear. 2020.

[6] E. Carson and N. J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM J. Sci. Comput.*, 40(2):A817–A847, 2018.

[7] P. I. Davies and N. J. Higham. Computing $f(A)b$ for matrix functions $f$. In *QCD and Numerical Analysis III*, volume 47 of *Lect. Notes Comput. Sci. Eng.*, pages 15–24. Springer, Berlin, 2005.

[8] E. Di Nezza, G. Palatucci, and E. Valdinoci. Hitchhiker's guide to the fractional Sobolev spaces. *Bulletin des Sciences Mathématiques*, 136(5):521–573, 2012.

[9] G. H. Golub and G. Meurant. *Matrices, moments and quadrature with applications*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2010.

[10] C.-H. Guo and N.J. Higham. A Schur-Newton method for the matrix $p$th root and its inverse. *SIAM Journal on Matrix Analysis and Applications*, 28(3):788–804, 2006.

[11] S. Gttel and L. Knizhnerman. A black-box rational arnoldi variant for Cauchy-Stieltjes matrix functions. *BIT Numerical Mathematics*, 53(3):595–616, 2013.

[12] N. Hale, N. J. Higham, and L. N. Trefethen. Computing $A^\alpha$, $\log(A)$, and related matrix functions by contour integrals. *SIAM J. Numer. Anal.*, 46(5):2505–2523, 2008.

[13] N. J. Higham. *Functions of matrices*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2008. Theory and computation.

[14] N.J. Higham. Evaluating Padé approximants of the matrix logarithm. *SIAM Journal on Matrix Analysis and Applications*, 22(4):1126–1135, 2001.

[15] Y. Huang and A. Oberman. Numerical methods for the fractional Laplacian: a finite difference–quadrature approach. *SIAM J. Numer. Anal.*, 52(6):3056–3084, 2014.

[16] B. Iannazzo. On the Newton method for the matrix $p$th root. *SIAM J. Matrix Anal. Appl.*, 28(2):503–523, 2006.

[17] C. Kenney and A. J. Laub. Rational iterative methods for the matrix sign function. *SIAM J. Matrix Anal. Appl.*, 12(2):273–291, 1991.

[18] B. Laszkiewicz and K. Zitak. A Padé family of iterations for the matrix sector function and the matrix $p$th root. *Numerical Linear Algebra with Applications*, 16(11-12):951–970, 2009.

[19] X. Li and C. Xu. Existence and uniqueness of the weak solution of the space-time fractional diffusion equation and a spectral method approximation. *Commun. Comput. Phys.*, 8(5):1016–1051, 2010.

[20] A. Lischke, G. Pang, M. Gulian, F. Song, C. Glusa, X. Zheng, Z. Mao, W. Cai, M. Meerschaert, M. Ainsworth, and G. E. Karniadakis. What is the fractional laplacian? *arXiv:1801.09767v2*, 2018.

[21] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[22] Y. Saad. *Numerical methods for large eigenvalue problems*, volume 66 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2011. Revised edition of the 1992 original [ 1177405].

[23] Y. Saad and M.H. Schultz. GMRES - A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[24] M. I. Smith. A Schur algorithm for computing matrix $p$th roots. *SIAM J. Matrix Anal. Appl.*, 24(4):971–989, 2003.

[25] P.T.P. Tang and E. Polizzi. FEAST as a subspace iteration eigensolver accelerated by approximate spectral projection. *SIAM J. Matrix Anal. Appl.*, 35(2):354–390, 2014.

[26] J. S. H. Tsai, L. S. Shieh, and R. E. Yates. Fast and stable algorithms for computing the principal $n$th root of a complex matrix and the matrix sector function. *Comput. Math. Appl.*, 15(11):903–913, 1988.

[27] Q. Yang, I. Turner, F. Liu, and M. Ilić. Novel numerical methods for solving the time-space fractional diffusion equation in two dimensions. *SIAM J. Sci. Comput.*, 33(3):1159–1180, 2011.

[28] P. Zhuang, F. Liu, V. Anh, and I. Turner. Numerical methods for the variable-order fractional advection-diffusion equation with a nonlinear source term. *SIAM J. on Numer. Anal.*, 47(3):1760–1781, 2009.

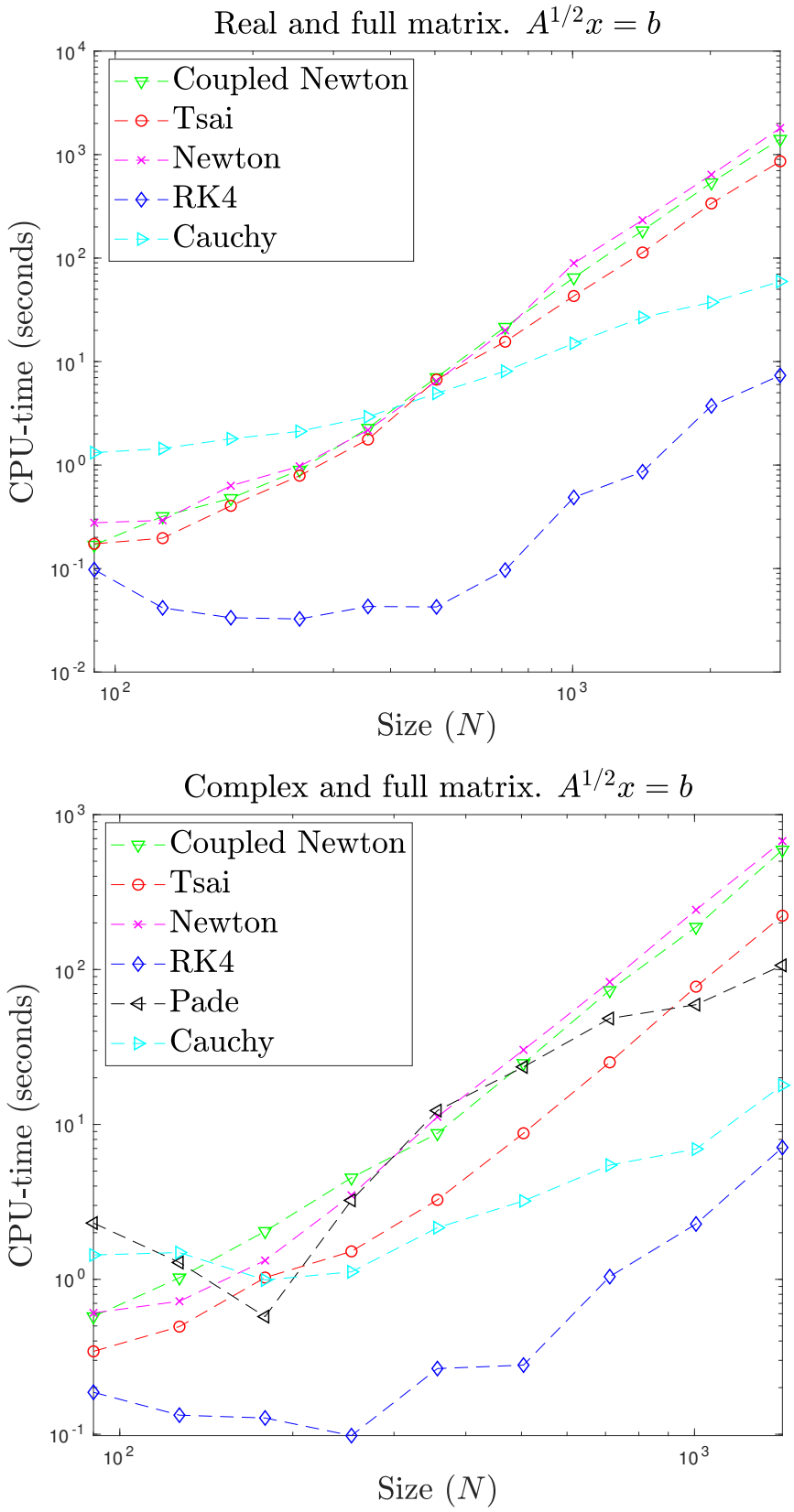




Figure 11: **Experiment 6.** CPU-time in seconds using RK4, Cauchy integral, Tsai, Newton, coupled Newton, Padé methods for $p = 2$. (Top) Full real matrix. (Bottom) Full complex matrix.

# Chapter 5

# Discussion and concluding remarks

Along this thesis, we first introduce the notion of matrix functions including some nontrivial examples in Chapter 1. Then, we introduce the formal definition of the matrix function $f(A)$ via different forms including Jordan canonical forms, polynomials and Cauchy integral. In addition, different types of algorithms for computing $f(A)$ are introduced including Taylor series, rational approximations and Padé's approximations. Then, we move to the matrix $p$-th root computation: in Chapter 3, we introduce matrix sign function, matrix square root and matrix sector function before introducing matrix $p$-th root. Within this chapter, we do some detailed explanation on the Padé approximations of each types of matrix functions mentioned before. Finally, we have proposed direct and iterative algorithms for the computation of FLAS.

In conclusion, we have proven that ODE-based solvers are the most efficient

and flexible for solving FLAS. The coupling of the ODE-solver with a preconditioning is even more efficient. In addition, among all kinds of numerical methods for the ODE-based solver, RK4 is the most efficient one.

Since it is possible to derive the explicit Padé approximation formula for $A^{1/p}$ where $p > 2$ and $p \in \mathbb{N}^*$, but they are not appeared in this thesis, those explicit formulae could be derived in the future.

# Bibliography


[1] P. Lawrence (2001) *Differential Equations and Dynamical Systems*, *Texts in Applied Mathematics, third edition*

[2] N.J. Higham (2008) *Functions of Matrices: Theory and Computation*, *Society for Industrial and Applied Mathematics*

[3] A. George Baker, JR. (1975) *Essentials of Padé Approximants*, *Academic Press*

[4] C. Kenney, J. Alan Laub (1991) *Rational Iterative Methods for The Matrix Sign Function*, *1991 Society for Industrial and Applied Mathematics.*

[5] X. Antoine, E. Lorin (2019) *ODE-based double-preconditioning for solving linear systems $A^{\alpha}X = B$ and $F(A)X = B$, HAL Archive ouverte*

[6] L. Shieh, Y. Tsay, C. Wang. (1984) *Matrix sector functions and their applications to systems theory*, *IET Digital Library, Volume 131, Issue 5*

[7] B. Laszkiewicz, K. Zietak (2009) *A Padé family of iterations for the matrix sector function and the matrix p-th root*, *Numerical Linear Algebra with Applications*

[8] A. Bini, N.J. Higham, B. Meini (2005) *Algorithms for the matrix p-th root*, *Numerical Algorithms, Springer*

[9] E. Lorin, S. Tian (2020) *A numerical study of fractional linear algebraic system solvers.* (Submitted to Mathematics and Computers in Simulation on August 11, 2020)

[10] S. Guttel, L. Knizhnerman (2013) *A black-box rational Arnoldi variant for Cauchy-Stieltjes matrix functions*, *Central Geophysical Expedition.*