Generalized fractional algebraic linear system solvers

Xavier Antoine, Emmanuel Lorin

Convergent multi-matrix fractional linear system solvers

Xavier ANTOINE* Emmanuel LORIN†‡

Abstract

This paper is devoted to the numerical computation of fractional linear systems involving several matrix power functions, that is finding x solution to $\sum_{\alpha\in\mathbb{R}} A^\alpha x = b$. These systems will be referred to as *Multi-matrix Fractional Linear Systems* (MFLS). We propose several gradient methods for solving these very computationally complex problems, which themselves require the solution to *standard Fractional Linear Systems* (FLS) $A^\alpha x = b$, with $\alpha \in \mathbb{R}$. The latter usually requires the solution to many *classical linear systems* $Ax = b$. We also show that in some cases, the solution to a MFLS problem can be obtained as the solution to a first-order hyperbolic system of conservation laws, and we discuss some connections between this approach and gradient-type methods. The convergence study is developed and numerical experiments are proposed to illustrate and compare the methods.

Keywords. Fractional linear systems, differential equation solver, iterative solver, gradient method, GMRES, fractional PDE

Contents

*Université de Lorraine, CNRS, Inria, IECL, F-54000 Nancy, France. **E-mail:** xavier.antoine@univ-lorraine.fr
†Centre de Recherches Mathématiques, Université de Montréal, Montréal, Canada, H3T 1J4.
‡School of Mathematics and Statistics, Carleton University, Ottawa, Canada, K1S 5B6. **E-mail:** elorin@math.carleton.ca

1 Introduction

In this paper, we are interested in the numerical solution to general Multi-matrix Fractional Linear Systems (MFLS), i.e. linear systems of the form

$$\text{Determine } x \in \mathbb{R}^n, \text{ such that } \sum_{i=1}^{N} A^{\alpha_i} x \;=\; b, \tag{1}$$

where i) A is a $\mathbb{R}^{n\times n}$ matrix with eigenvalues in $\mathbb{C}\backslash\mathbb{R}^-$ and $b \in \mathbb{R}^n$, ii) $\{\alpha_i\}_{1\leqslant i\leqslant N}$ is a finite sequence of real-valued exponents such that $1 > \alpha_1 > \alpha_2 > \cdots > \alpha_N > 0$, with $N \geqslant 2$. We also assume that (1) has a unique solution. This is the case for instance for positive definite matrices A since $\det(\sum_{i=1}^N A^{\alpha_i}) \geqslant \sum_{i=1}^N \det(A)^{\alpha_i} > 0$. The extension of the proposed methods to exponents larger than 1 requires some additional technical details, but the algorithms as well as the principle of the proofs remain valid. Indeed, we can formally rewrite A^α for any $\alpha > 1$, in the form $A^{\lfloor\alpha\rfloor}A^{\alpha-\lfloor\alpha\rfloor}$, where $\lfloor\alpha\rfloor$ is the integer part of α, which justifies that we restrict the study to $\alpha_i < 1$, for any index $1 \leqslant i \leqslant N$. Regarding the case of negative exponents, the multiplication of the equation (1) by A^p, for some $p \in \mathbb{N}^*$, would simply lead to problems with positive exponents. The objective of this paper is then to derive and analyze some original and efficient methods to solve the MFLS (1). This work is largely motivated by the recent development of the mathematical analysis and numerical approximation of fractional partial differential equations, and more generally fractional models, in particular fractional Laplacian-based models, including time-dependent fractional heat or fractional Schrödinger equations [20]. Typically, approximations of models of the form

$$-\sum_{i=1}^{N}(-\triangle)^{\alpha_i}u(\boldsymbol{x}) \;=\; f(\boldsymbol{x})\,,$$

for $\boldsymbol{x}(:= x_1$ in 1D and $:= (x_1, x_2)$ in 2D) defined in a bounded domain Ω with null Dirichlet boundary condition $u = 0$ at $\partial\Omega$, can sometimes be reduced to fractional linear systems of the form (1), when we define $-(-\triangle)^\alpha$ as a *spectral fractional Laplacian*, i.e. $(-\triangle)^\alpha u = \sum_{k=1}^\infty \lambda_k^\alpha (u, e_k)_{L^2(\Omega)} e_k$. Here, $\{e_k\}_k$ (resp. $\{\lambda_k\}_k$) are the eigenfunctions (resp. eigenvalues) of $-\triangle$ with null Dirichlet boundary conditions, and $(\cdot,\cdot)_{L^2(\Omega)}$ denotes the $L^2(\Omega)$-inner product (see e.g. [20]). On the other hand, we recall that, for $k \geqslant 0$, the *Cauchy integral representation* reads

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

where Γ_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 an important question as discussed in [2, 21].

There is an extensive literature on the computation of *standard* Fractional Linear Systems (FLS)

$$A^{\alpha}x = b\,, \tag{3}$$

and more generally on the computation of matrix functions or their spectrum [2, 3, 12, 13, 14, 16, 17, 18, 25, 28, 4]. Based on several examples, it was numerically observed [21] that the ODE-based solvers (4) for sparse matrices are often the most efficient in comparison with many other direct or iterative computational FLS solvers (Padé, Newton, Cauchy,...). In this paper, when needed, the FLS (3) are solved by the ODE-based approach which is recalled now. We refer to [7, 10, 13, 14, 15] for details about this approach.

1.1 ODE-solver for FLS

Since the FLS is one fundamental building block of our algorithm for solving the MFLS, let us first explain how the ODE-based solver works for one single n-dimensional system. For $\tau \in [0,1]$, we consider

$$x'(\tau) = -\alpha(A-I)\big(I+\tau(A-I)\big)^{-1}x(\tau), \qquad x(0) = b \in \mathbb{R}^n\,, \tag{4}$$

whose solution $x(\tau) = \big(I+\tau(A-I)\big)^{-\alpha}b$ is such that $x(1) = A^{-\alpha}b$. The advantage of this approach is that there is no need to compute any matrix power, and all the existing methodologies for solving ODE systems can be applied. Practically, let us introduce the uniformly sampled discrete times $\tau_j = j\Delta\tau$, with $0 \leqslant j \leqslant J$, $\tau_J = 1$, and constant time step $\Delta\tau$. For $\alpha \in \mathbb{R}_+\backslash\{0\}$, we propose to approximate $A^{-\alpha}b$ iteratively by discretizing (4). For instance, using a second-order *Crank-Nicolson* (CN) scheme yields

$$x^{(j+1)} = x^{(j)} - \frac{\Delta\tau}{2}\alpha(A-I)\big(I+\tau_j(A-I)\big)^{-1}x^{(j)} - \frac{\Delta\tau}{2}\alpha(A-I)\big(I+\tau_{j+1}(A-I)\big)^{-1}x^{(j+1)}\,,$$

where $x^{(j)}$ is an approximation of $x(\tau_j)$ solution to (4) at $t = \tau_j$. The full iterative scheme is presented in Algorithm 1. At each iteration $j \in \{1, \cdots, J\}$, we need to solve two linear systems using traditional *iterative solvers combined with preconditioners*.

Algorithm 1 Crank-Nicolson (CN) method for computing $x = A^{-\alpha}b$

1: For $j \geqslant 0$ and $x^{(j)}$ known, compute $z^{(j)}$ solution to $\big(I+\tau_j(A-I)\big)z^{(j)} = x^{(j)}$.

2: Set $w^{(j)} := x^{(j)} - \dfrac{\Delta\tau}{2}\alpha(A-I)z^{(j)}$, and calculate $\omega^{(j+1)}$ such that

$$\big(I+\tau_{j+1}(A-I)\big)\omega^{(j+1)} \;=\; w^{(j)}\,.$$

3: Compute $y^{(j+1)}$ solution to

$$(I+\tau_{j+1+\frac{\alpha}{2}}(A-I))y^{(j+1)} \;=\; \omega^{(j+1)}\,.$$

4: Obtain $x^{(j+1)}$ as

$$x^{(j+1)} \;=\; (I+\tau_{j+1}(A-I))y^{(j+1)}\,.$$

5: Deduce $x^{(J)}$ to approximate $A^{-\alpha}b$.

If needed, higher-order ODE-solvers could be used. For example, the corresponding fourth-order Runge-Kutta (RK4) scheme is presented in Algorithm 2. Hence, at the final time $\tau_J = 1$,

there exists a positive constant $C > 0$ such that

$$\|x^{(J)} - A^{-\alpha}b\|_2 \leqslant C\Delta\tau^p\,,$$

setting $\|v\|_2 = \big(\sum_{j=0}^n |v_j|^2\big)^{1/2}$ for any vector $v \in \mathbb{R}^n$, and where $p = 2$ for the Crank-Nicolson scheme and $p = 4$ for the RK4 scheme. In Appendix, we propose an efficient preconditioning technique to improve the convergence of the ODE-solver.

Algorithm 2 Runge-Kutta (RK4) method for computing $x = A^{-\alpha}b$

1: For $j \geqslant 0$ and $x^{(j)}$ known, compute $w_1^{(j)}$ solution to

$$\begin{aligned}\big(I + \tau_j(A-I)\big)z_1^{(j)} &= x^{(j)}\,,\\ w_1^{(j)} &= -\alpha\Delta\tau(A-I)z_1^{(j)}\,.\end{aligned}$$

2: Compute $w_2^{(j)}$ solution to

$$\begin{aligned}\big(I + \tau_{j+1/2}(A-I)\big)z_2^{(j)} &= x^{(j)} + z_1^{(j)}/2\,,\\ w_2^{(j)} &= -\alpha\Delta\tau(A-I)z_2^{(j)}\,.\end{aligned}$$

3: Compute $w_3^{(j)}$ solution to

$$\begin{aligned}\big(I + \tau_{j+1/2}(A-I)\big)z_3^{(j)} &= x^{(j)} + z_2^{(j)}/2\,,\\ w_3^{(j)} &= -\alpha\Delta\tau(A-I)z_3^{(j)}\,.\end{aligned}$$

4: Compute $w_4^{(j)}$ solution to

$$\begin{aligned}\big(I + \tau_{j+1}(A-I)\big)z_4^{(j)} &= x^{(j)} + z_3^{(j)}\,,\\ w_4^{(j)} &= -\alpha\Delta\tau(A-I)z_4^{(j)}\,.\end{aligned}$$

5: Obtain $x^{(j+1)}$ such that

$$x^{(j+1)} = \frac{1}{6}\big(w_1^{(j)} + 2w_2^{(j)} + 2w_3^{(j)} + w_4^{(j)}\big)\,.$$

6: Deduce $x^{(J)}$ to approximate $A^{-\alpha}b$.

1.2 Organization of the paper

In Section 2, we derive and analyze several simple iterative methods for solving the MFLS (1), including gradient methods, GMRES, and Taylor's expansion techniques. In addition, all the methods are validated and compared through numerical simulations. In Section 3, we introduce an original Partial Differential Equation (PDE)-based solver for MFLS, which is a natural extension of the standard FLS ODE-based solver (4) to MFLS. Some preliminary experiments to validate the method are proposed in Subsection 3.3, and the connection of the new approximate PDE solvers with gradient methods is explained in Subsection 3.4. We discuss some possible extensions in Section 4. Finally, we conclude in Section 5.

2 Derivation and analysis of the MFLS iterative solvers

In this section, we derive and prove the convergence of gradient-type solvers for the MFLS, explain how to apply the GMRES procedure, as well as Taylor's series expansion. We first study the 2-matrix case ($N = 2$), and then generalize to the N-matrix MFLS. The key point is that we naturally cannot explicitly compute the matrix powers, which would be too much computationally complex.

2.1 Gradient solvers

We first introduce some gradient solvers for MFLS. In the following, we use the ℓ^2-norm defined by $\|A\| = \|A\|_2 = \sigma_{\max}(A) = \sqrt{\lambda_{\max}(A^*A)} = \sqrt{\rho(A^*A)}$, where A^* is the complex conjugate matrix of A. We also denote by $\kappa(A) = \|A\|\|A^{-1}\|$ the condition number of A.

For solving the MFLS (3), the classical constant step gradient method consists in solving

$$x_{k+1} = x_k + \rho\big(b - \sum_{i=1}^{N} A^{\alpha_i} x_k\big),$$

for $\rho > 0$. For positive definite matrices A, the convergence occurs whenever $0 < \rho < 2/\sum_{i=1}^{N} \lambda_n^{\alpha_i}$, where λ_n is the largest eigenvalue of A. Hence, we have

$$\|x_k - x\| \leqslant \beta^k \|x - x_0\|,$$

where $\beta < 1$. The optimal value of ρ is known to be $2/\sum_{i=1}^{N}(\lambda_n^{\alpha_i} + \lambda_1^{\alpha_i})$ which is however a very restrictive condition [11]. To improve the convergence condition, it is possible to make vary the optimal value of ρ at each step, i.e. considering $\rho_k > 0$. Practically, evaluating the optimal value of ρ is however very computationally complex, and each iteration of the gradient method requires N computations of standard FLS following Algorithm 3 below.

Algorithm 3 Standard gradient method with optimal parameter

1: Select the initial guess x_0.

2: For $k \geqslant 0$, compute $r_k = \sum_{i=1}^{N} A^{\alpha_i} x_k - b$.

3: The optimal parameter reads

$$\rho_k = \frac{\|r_k\|}{\big\langle \sum_{i=1}^{N} A^{\alpha_i} r_k, r_k \big\rangle}.$$

4: $x_{k+1} = x_k - \rho_k r_k$.

In this paper, we consider the general derivation of gradient-type algorithms with an adaptive step ρ_k, but for the numerical simulations, only the constant step $\rho = \rho_k$ is tested. In this case, for positive definite matrices, since the convergence strongly depends on the largest eigenvalue $\sum_{i=1}^{N} \lambda_n^{\alpha_i}$, we will rather consider a preconditioned version of the MFLS thanks to $A^{-\alpha_1}$, leading to a less restrictive convergence constraint: $0 < \rho < 2/(1 + \sum_{i=2}^{N} \lambda_n^{\alpha_i - \alpha_1})$. More precisely, the gradient-based algorithms can be seen as the approximation to a gradient flow equation of the form

$$\partial_t x(t) = -\nabla_x J(x), \tag{5}$$

with initial guess $x(0) = x_0$ for the functional (with symmetrical matrices) preconditioned by $A^{-\alpha_1}$

$$J(x) = \frac{1}{2}((I + \sum_{i=2}^{N} A^{\alpha_i - \alpha_1})x, x) - (A^{-\alpha_1} b, x)\,,$$

where $1 > \alpha_1 > \cdots > \alpha_N > 0$.

2.1.1 The case of the 2-matrix MFLS

In this section, we consider a MFLS with $N = 2$, i.e. for $0 < \beta < \alpha < 1$. We then consider the discretization of the gradient flow with adaptive step $\rho_k > 0$, i.e.

$$x_{k+1} = x_k - \rho_k \nabla_x J(x^k) = \rho_k A^{-\alpha} b - (\rho_k - 1)x_k - \rho_k A^{\beta-\alpha} x_k.$$

To accelerate the convergence of the solution to gradient flows, the iPiano method was introduced in [8, 22] in the framework of mathematical imaging. It was proved to be very efficient by improving the convergence rate. The idea behind the method is to add an inertial term $\beta_{\text{iPiano}}(x_k - x_{k-1})$ to ensure the acceleration of the algorithm, thus leading to

$$x_{k+1} = x_k - \rho_k((I + A^{\beta-\alpha})x_k - A^{-\alpha} b) + \beta_{\text{iPiano}}(x_k - x_{k-1}).$$

In terms of gradient flow, and for a simplified configuration, the iPiano scheme corresponds to the so-called *Heavy-ball* method. Indeed, this consists in using an explicit scheme for solving the *Heavy-ball* friction dynamical system

$$\partial_t^2 x(t) + \gamma \partial_t x(t) = -\nabla_x J(x). \tag{6}$$

In applications [1, 22], a standard value for the parameter β_{iPiano} is usually $0.8 - 0.9$. We come back later to this choice in the numerical examples.

Let us recall that there are two levels of convergence in the proposed methodology. One is related to intermediate standard (1-matrix) FLS, and the second one is in connection to the construction of the iterated sequence $\{x_k\}_k$. Let us note that for $\rho_k = 1$ and $\beta_{\text{iPiano}} = 0$, we formally have, for $k \geqslant 0$

$$x_k = \big(I - (-1)^k A^{k(\beta-\alpha)}\big)\big(A^\alpha + A^\beta\big)^{-1} b + (-1)^k A^{k(\beta-\alpha)} x_0\,. \tag{7}$$

However, this is practically useless. Let us prove the following convergence results for the associated Algorithm 4.

Proposition 1 *We consider the system*

$$\big(A^\alpha + A^\beta\big)x \;\; = \;\; b\,, \tag{8}$$

with $1 > \alpha > \beta > 0$ and A a hermitian matrix. Let us also assume that the intermediate fractional linear systems are all solved by using a p-th order ODE-solver for (4), leading to the evaluation $x_k^{(J)}$ of x_k at time $\tau_J = 1$. Then, we have

1. *For $\rho_k = 1$ and $\beta_{\text{iPiano}} = 0$, the following result holds for Algorithm 4:*

   - *For $\|A\| > \kappa(A)$, the iterated sequence $\{x_k^{(J)}\}_{k\in\mathbb{N}}$ defined in (10) is convergent and satisfies*

$$\|x - x_k^{(J)}\| \;\; \leqslant \;\; \Big(\frac{\kappa(A)}{\|A\|}\Big)^{k(\alpha-\beta)} \|x_0 - x\| + c(A)\Delta\tau^p\,.$$

- *For $\|A\| < \kappa(A)$, the sequence $\{x_k^{(J)}\}_{k\in\mathbb{N}}$ defined by*

$$x_{k+1} = x_k - \rho_k((I + A^{\alpha-\beta})x_k - A^{-\beta}b) + \beta_{\text{iPiano}}(x_k - x_{k-1}), \tag{9}$$

is convergent and is such that

$$\|x - x_k^{(J)}\| \leqslant \Big(\frac{\|A\|}{\kappa(A)}\Big)^{k(\alpha-\beta)}\|x_0 - x\| + c(A)\Delta\tau^p\,.$$

2. *We consider (10) for $\rho_k = \rho$ constant and $\beta_{\text{iPiano}} \geqslant 0$, and assume in addition that A is diagonalizable in $\mathbb{R}$. Then, the convergence of the iterated occurs in Algorithm 4 if the following condition holds*

$$\max_{1\leqslant i\leqslant n}\Big|(1-\rho-\beta_{\text{iPiano}}) + \rho\lambda_i^{\beta-\alpha}\big)\big(1 \pm \sqrt{1 - \frac{4\beta_{\text{iPiano}}}{\Big((1-\rho-\beta_{\text{iPiano}}) + \rho\lambda_i^{\beta-\alpha}\big)^2}}\Big)\Big| < 2\,,$$

where $\{\lambda_i\}_{1\leqslant i\leqslant n}$ denote the eigenvalues of A.

Algorithm 4 Preconditioned gradient method with adaptive step ρ_k and iPiano parameter β_{iPiano} for the 2-matrix MFLS

1: Select the initial guess x_0 and set $x_{-1} = x_0$.
2: Iterate for $k \geqslant 0$ until convergence

$$x_{k+1} = x_k - \rho_k((I + A^{\beta-\alpha})x_k - A^{-\alpha}b) + \beta_{\text{iPiano}}(x_k - x_{k-1}) \tag{10}$$

for some adaptive step ρ_k and parameter β_{iPiano}.

Proof. By continuity argument, if $\{x_k\}_k$ is convergent, then it converges to the solution of the system. Now, let us state the proofs of the 2 items.

1. We study the convergence for the case $\|A\| > \kappa(A)$, $\|A\| < \kappa(A)$ can be treated similarly. Let us denote by $x_k^{(J)}$ the approximate solution at iteration k using an order-p ODE solver with $J = 1/\Delta\tau$. From (7), we have

$$x := (A^\alpha + A^\beta)^{-1}b\,,$$

and for $k \geqslant 1$,

$$\begin{aligned}\|x_k\| &\leqslant \|A^{k(\beta-\alpha)}\|\|x_0\| + \big(1 + \|A^{k(\beta-\alpha)}\|\big)\|x\| \\ &\leqslant \|A^{\beta-\alpha}\|^k\|x_0\| + \big(1 + \|A^{\beta-\alpha}\|^k\big)\|x\|\,.\end{aligned}$$

Since $\alpha > 0$ and $\beta - \alpha < 0$, and notice that for any $\gamma > 0$, one gets

$$\|A^{-\gamma}\| = \sqrt{\lambda_{\max}\big((A^{-\gamma})^* A^{-\gamma}\big)} = \sqrt{\lambda_{\max}^{\gamma}\big((A^{-1})^* A^{-1}\big)} = \|A^{-1}\|^\gamma.$$

Then, from $\|A^{-1}\| = \kappa(A)\|A\|^{-1}$, we deduce that

$$\begin{aligned}\|x_k\| &\leqslant \|A^{-1}\|^{k(\alpha-\beta)}\|x_0\| + \big(1 + \|A^{-1}\|^{k\alpha}\big)\|x\| \\ &\leqslant \big(\kappa(A)/\|A\|\big)^{k(\alpha-\beta)}\|x_0\| + \big(1 + \big(\kappa(A)/\|A\|\big)^{k\alpha}\big)\|x\|\,.\end{aligned}$$

In addition, we have: $\|x_{\ell+1}-x_\ell\| \leqslant \|A^{\beta-\alpha}\|\|x_\ell-x_{\ell-1}\|$. We then conclude the convergence of the algorithm as $\|A\| > 1$ and $\beta - \alpha < 0$.

Let us next consider, for $k \geqslant 1$,

$$\begin{aligned}
\|x - x_k^{(J)}\| &= \|x - x_k\| + \|x_k - x_k^{(J)}\| \\
\|x - x_k^{(J)}\| &= \|x - A^{-\alpha}b + A^{\beta-\alpha}x_{k-1}\| + c(A)\Delta\tau^p \\
&= \|A^{-\alpha}b - A^{\beta-\alpha}x - A^{-\alpha}b + A^{\beta-\alpha}x_{k-1}\| + c(A)\Delta\tau^p \\
&= \|A^{\beta-\alpha}(x_{k-1} - x)\| + c(A)\Delta\tau^p \\
&\leqslant \|A^{\beta-\alpha}\|^k\|x_0 - x\| + c(A)\Delta\tau^p\,.
\end{aligned}$$

Again, $\|A^{-\gamma}\| = \|A^{-1}\|^\gamma$ since A is hermitian, with $0 < \gamma < 1$, and therefore we obtain

$$\begin{aligned}
\|x - x_k^{(J)}\| &\leqslant \|A^{-1}\|^{k(\alpha-\beta)}\|x_0 - x\| + c(A)\Delta\tau^p \\
&\leqslant \big(\kappa(A)/\|A\|\big)^{k(\alpha-\beta)}\|x_0 - x\| + c(A)\Delta\tau^p\,,
\end{aligned}$$

which provides a rate of convergence of the overall algorithm for $\alpha - \beta > 0$.

2. Let us assume that the step size is constant, i.e. $\rho_k = \rho$ for all k, and consider first the case where $\beta_{\text{iPiano}} = 0$. Hence the algorithm simply reads

$$x_{k+1} = \rho A^{-\alpha}b - \big((1-\rho)I - \rho A^{\beta-\alpha}\big)x_k\,.$$

This leads to

$$x_k = \big(I - (-1)^k\big((1-\rho)I + \rho A^{\beta-\alpha}\big)^k\big)(A^\alpha + A^\beta)^{-1}b + (-1)^k\big((1-\rho)I + \rho A^{\beta-\alpha}\big)^k x_0\,.$$

We easily extend the idea developed above

$$\begin{aligned}
x_k - x &= \rho A^{-\alpha}b - \big((\rho-1)I + \rho A^{\beta-\alpha}\big)x - \rho A^{-\alpha}b + \big((1-\rho)I + \rho A^{\beta-\alpha}\big)x_{k-1} \\
&= \big((1-\rho)I + \rho A^{\beta-\alpha}\big)(x_{k-1} - x)\,.
\end{aligned} \tag{11}$$

Hence, we have

$$\|x_k - x\| = \big\|\big((1-\rho)I + \rho A^{\beta-\alpha}\big)^k\big\|\|x - x_0\|\,.$$

As a consequence, the parameter ρ gives some flexibility regarding the rate of convergence. Setting $D := (1-\rho)I + \rho A^{\beta-\alpha}$ and the residual as $e_k = x_k - x$, we get: $e_k = De_{k-1}$. Assuming that $A \in \mathbb{R}^{n\times n}$ is diagonalizable in $\mathbb{R}$, we then deduce that D is diagonalizable in the same basis P as A. We denote by $\{\lambda_i\}_{1\leqslant i\leqslant n}$ the eigenvalues of A. Hence by setting $f_k := Pe_k = (f_{k;1}, \cdots, f_{k;n})$, we obtain

$$f_{k;i} = \big((1-\rho) + \rho\lambda_i^{\beta-\alpha}\big)^k f_{0;i}. \tag{12}$$

The convergence occurs whenever

$$\max_i \big|(1-\rho) + \rho\lambda_i^{\beta-\alpha}\big| < 1\,,$$

and ideally when it is as small as possible.

We now consider the iPiano version of the algorithm ($\beta_{\text{iPiano}} \neq 0$) and evaluate

$$\begin{aligned}
x_k - x &= \rho A^{-\alpha}b - \big((\rho-1)I + \rho A^{\beta-\alpha}\big)x \\
&\quad -\rho A^{-\alpha}b + \big((1-\rho)I + \rho A^{\beta-\alpha}\big)x_{k-1} - \beta_{\text{iPiano}}(x_{k-1} - x_{k-2}) \\
&= \big((1-\rho-\beta_{\text{iPiano}})I + \rho A^{\beta-\alpha}\big)(x_{k-1} - x) + \beta_{\text{iPiano}}(x_{k-2} - x)\,.
\end{aligned}$$

Setting $F := (1-\rho-\beta_{\text{iPiano}})I + \rho A^{\beta-\alpha}$ and $e_k = x_k - x$, we then have: $e_k = Fe_{k-1} + \beta_{\text{iPiano}} e_{k-2}$. For $A \in \mathbb{R}^{n\times n}$ diagonalizable in $\mathbb{R}$, we obtain that F is a diagonalizable matrix in the basis P, with eigenvalues $\{\zeta_i\}$ such that: $\zeta_i = (1-\rho-\beta_{\text{iPiano}}) + \rho\lambda_i^{\beta-\alpha}$, where $\{\lambda_i\}_{1\leqslant i\leqslant n}$ still denotes the eigenvalues of A. Hence for $f_k := Pe_k = (f_{k;1}, \cdots, f_{k;n})$, we prove that

$$f_{k;i} \;=\; \zeta_i f_{k-1;i} + \beta_{\text{iPiano}} f_{k-2;i}\,. \tag{13}$$

We set $r_i^{\pm}$ as the roots to the characteristic equation related to (13) and given by

$$\begin{aligned} r_i^{\pm} &= \zeta_i \frac{1 \pm \sqrt{1 - \dfrac{4\beta_{\text{iPiano}}}{\zeta_i^2}}}{2} \\ &= \big((1-\rho-\beta_{\text{iPiano}}) + \rho\lambda_i^{\beta-\alpha}\big) \frac{1 \pm \sqrt{1 - \dfrac{4\beta_{\text{iPiano}}}{\big((1-\rho-\beta_{\text{iPiano}}) + \rho\lambda_i^{\beta-\alpha}\big)^2}}}{2}. \end{aligned}$$

The convergence of the algorithm requires that $\max_{1\leqslant i\leqslant n} |r_i^{\pm}| < 1$ or $|r_i^{\pm}| = 1$ of multiplicity 1 [26]. The parameter β_{iPiano} and ρ hence give more flexibility to accelerate the convergence rate of the gradient algorithm. □

Regarding the complexity, at each iteration, it is necessary to compute standard fractional linear systems for computing $A^{-\alpha}b_k$ and $b_k = b - A^{\beta}x_k$ with an ODE-based solver.

2.1.2 Extension to N-matrix MFLS

We now consider the general case assuming that (1) has a unique solution, with the exponents $1 > \alpha_1 > \alpha_2 > \cdots > \alpha_N > 0$. We can therefore derive the preconditioned gradient-type methods with adaptive step ρ_k for MFLS as given by Algorithm 5.

Algorithm 5 Preconditioned gradient method with adaptive step ρ_k and iPiano parameter β_{iPiano} for N-matrix MFLS

1: Select the initial guess x_0 and set $x_{-1} = x_0$.
2: Iterate until convergence for $k \geqslant 0$

$$x_{k+1} \;=\; \rho_k A^{-\alpha_1} b - (\rho_k - 1)x_k - \rho_k \sum_{i=2}^{N} A^{\alpha_i - \alpha_1} x_k + \beta_{\text{iPiano}}(x_k - x_{k-1})\,, \tag{14}$$

for some adaptive step ρ_k and parameter β_{iPiano}.

As before, it is possible to get an explicit formal expression of the solution to the algorithm

$$x_k \;=\; \Big(I - (-1)^k \big(\sum_{i=2}^{N} A^{(\alpha_i-\alpha)}\big)^k\Big)\big(\sum_{i=1}^{N} A^{\alpha_i}\big)^{-1} b + (-1)^k \big(\sum_{i=2}^{N} A^{(\alpha_i-\alpha_1)}\big)^k x_0,$$

for $\rho_k = 1$. Again this formulation is not useful practically, but could help to study the mathematical properties of the method. We prove the following convergence result, corresponding to the preconditioned gradient without iPiano additional term (i.e. $\beta_{\text{iPiano}} = 0$). An analogous result to the $N = 2$ case could also be obtained for the iPiano-version up to the additional technical details.

Proposition 2 *We consider the system* (1) *with* A *a hermitian matrix, and we assume that the intermediate fractional linear systems are solved by using an order-p ODE-solver for* (4)*. Then for* $\rho_k = 1$*, we have the following results*

- *For* $\|A\| > \kappa(A)(N-1)^{\frac{1}{\alpha_1-\alpha_2}}$*, the sequence of iterated defined by* (14) *in Algorithm 5 is convergent and the following estimate holds*

$$\|x - x_k^{(J)}\| \leqslant (N-1)^k \Big(\frac{\kappa(A)}{\|A\|}\Big)^{k(\alpha_1-\alpha_2)} \|x_0 - x\| + c(A)\Delta\tau^p\,.$$

- *If* $\|A\| < \kappa(A)(N-1)^{\frac{1}{\alpha_N-\alpha_{N-1}}}$*, then the iterated sequence given by*

$$x_{k+1} = \rho_k A^{-\alpha_N} b - (\rho_k - 1)x_k - \rho_k \sum_{i=1}^{N-1} A^{\alpha_i-\alpha_N} x_k + \beta_{\text{iPiano}}(x_k - x_{k-1})\,, \quad (15)$$

converges and we have

$$\|x - x_k^{(J)}\| \leqslant (N-1)^k \Big(\frac{\|A\|}{\kappa(A)}\Big)^{k(\alpha_N-\alpha_{N-1})} \|x_0 - x\| + c(A)\Delta\tau^p\,.$$

Proof. By uniqueness assumption and continuity argument, if x_k is convergent, then it converges to the solution of the system. We proceed as above by setting now

$$b_k = b - \sum_{i=2}^{N} A^{\alpha_i} x_k.$$

In the following, we prove the convergence in the case $\|A\| > \kappa(A)(N-1)^{1/(\alpha_1-\alpha_2)}$. Since we formally have

$$x = (A^{\alpha_1} - \sum_{i=2}^{N} A^{\alpha_i})^{-1} b\,,$$

we can prove the following sequence of inequalities

$$\begin{aligned}
\|x_k\| &\leqslant \big\|\big(\sum_{i=2}^{N} A^{\alpha_i-\alpha_1}\big)^k\big\|\|x_0\| + \big(1 + \big\|\big(\sum_{i=2}^{N} A^{\alpha_i-\alpha_1}\big)^k\big\|\big)\|x\| \\
&\leqslant \big\|\sum_{i=2}^{N} A^{\alpha_i-\alpha_1}\big\|^k \|x_0\| + \big(1 + \big\|\sum_{i=2}^{N} A^{\alpha_i-\alpha_1}\big\|^k\big)\|x\| \\
&\leqslant (N-1)^k \|A^{\alpha_2-\alpha_1}\|^k \|x_0\| + \big(1 + (N-1)^k \|A^{\alpha_2-\alpha_1}\|^k\big)\|x\| \\
&\leqslant (N-1)^k \|A^{-1}\|^{k(\alpha_1-\alpha_2)} \|x_0\| + \big(1 + (N-1)^k \|A^{-1}\|^{k(\alpha_1-\alpha_2)}\big)\|x\| \\
&\leqslant (N-1)^k \big(\kappa(A)/\|A\|\big)^{k(\alpha_1-\alpha_2)} \|x_0\| + \big(1 + (N-1)^k \big(\kappa(A)/\|A\|\big)^{k(\alpha_1-\alpha_2)}\big)\|x\|\,.
\end{aligned}$$

Regarding the rate of convergence, we similarly show that

$$\begin{aligned}
\|x - x_k^{(J)}\| &= \|x - x_k\| + \|x_k - x_k^{(J)}\| \\
&= \|x - A^{-\alpha_1} b + \sum_{\ell=2}^{N} A^{\alpha_\ell - \alpha_1} x_{k-1}\| + c(A)\Delta\tau^p \\
&= \|A^{-\alpha_1} b - \sum_{\ell=2}^{N} A^{\alpha_\ell - \alpha_1} x - A^{-\alpha_1} b + \sum_{l=2}^{N} A^{\alpha_l - \alpha_1} x_{k-1}\| + c(A)\Delta\tau^p \\
&= \|\sum_{\ell=2}^{N} A^{\alpha_\ell - \alpha_1}(x_{k-1} - x)\| + c(A)\Delta\tau^p \\
&\leqslant \big\|\sum_{\ell=2}^{N} A^{\alpha_\ell - \alpha_1}\big\|^k \|x_0 - x\| + c(A)\Delta\tau^p \\
&\leqslant \big(\sum_{\ell=2}^{N} \|A^{\alpha_\ell - \alpha_1}\|\big)^k \|x_0 - x\| + c(A)\Delta\tau^p \,,
\end{aligned}$$

where we have used that for A hermitian, $\|A^{-\gamma}\| = \|A^{-1}\|^\gamma$ for $0 < \gamma < 1$. Hence, we conclude that

$$\begin{aligned}
\|x - x_k^{(J)}\| &\leqslant \big(\sum_{\ell=2}^{N} \|A^{-1}\|^{\alpha_1 - \alpha_\ell}\big)^k \|x_0 - x\| + c(A)\Delta\tau^p \\
&\leqslant (N-1)^k \|A^{\alpha_2 - \alpha_1}\|^k \|x_0 - x\| + c(A)\Delta\tau^p \,.
\end{aligned}$$

The rest of the proof is straightforward. □

2.2 GMRES solver

As far as we know, the use of GMRES [24] for solving MFLS or even FLS has never been discussed in the literature. Basically, the principle consists in directly solving the preconditioned equation

$$(I + \sum_{i=2}^{N} A^{\alpha_i - \alpha_1})x = A^{-\alpha_1} b, \tag{16}$$

with the expectation that $A^{-\alpha_1}$ is a good preconditioner of the original equation, leading then to a clustering of the eigenvalues of system (16) around 1 in the complex plane. In the case where A is a finite dimensional approximate Laplacian, this is relatively reasonable since the operators $(-\Delta)^{\alpha_i - \alpha_1}$ are negative order pseudodifferential operators with eigenvalues converging towards zero for large spatial frequencies. At each GMRES iteration, the method relies on the computation of N-matrix-vector products $y^k := A^{\alpha_i - \alpha_1} x_k$ performed by using the ODE-based solver. In the present paper, we consider the version of GMRES with restart to limit the memory storage. Although, it is expected that GMRES usually performs better than a gradient-type method, there are still some configurations where an opposite conclusion can be expected. Let us consider for instance systems such that $(A^\alpha - A^{\alpha-\varepsilon})x = b$, where ε is close to zero. Rewriting the latter in the form $(I - A^{-\varepsilon})x = A^{-\alpha} b$, the matrix $I - A^{-\varepsilon}$ is close to the null matrix naturally making *a priori* GMRES inefficient compared to a gradient method as the ones presented before. In addition, building a preconditioner for $I - A^{-\varepsilon}$ is far from being trivial, while the FLS-preconditioning developed in [3] is still suitable for the gradient methods. This will be illustrated in Example 3.

2.3 Truncated Taylor's series expansion approximation

Finally, for completeness, a standard Taylor series expansion can be used to solve

$$\big(A^\alpha + A^\beta\big)x \;=\; b\,.$$

We formally rewrite, for $\alpha > \beta > 0$,

$$x \;=\; (I + A^{\beta-\alpha})^{-1}A^{-\alpha}b \;=\; \sum_{\ell\geqslant 0}(-1)^\ell A^{\ell(\beta-\alpha)-\alpha}b\,.$$

Hence, approximating x by a truncated series x_M leads to

$$x_M \;=\; \sum_{0\leqslant\ell\leqslant M}(-1)^\ell A^{\ell(\beta-\alpha)-\alpha}b\,, \tag{17}$$

with naturally

$$\|x_M - x\| \;\leqslant\; \sum_{\ell>M}\|A^{\beta-\alpha}\|^\ell\|A^{-\alpha}\|\|b\|\,.$$

Practically, this requires to compute $A^{\ell(\beta-\alpha)-\alpha}b$, for $\ell = 0, \cdots, M$, which is performed by the ODE-based solvers. Hence, trivially the following error estimate occurs for (17)

$$\|x_M^{(J)} - x\| \;\leqslant\; \sum_{\ell>M}\|A^{\beta-\alpha}\|^\ell\|A^{-\alpha}\|\|b\| + c(A)M\Delta\tau^p\,.$$

Unlike, the above approach the algorithm is embarrassingly parallel. For a truncation number M in (17), the total number of standard FLS to solve is $M+1$. By comparison, Algorithm 4 after k iterations, requires the solution to $2k$ standard FLS.

2.4 Numerical examples

We propose now several numerical experiments to illustrate the behavior of the iterative solvers which were derived and analyzed above. When relevant, we will compare the convergence of these methods.

Example 1. In this first set of numerical simulations, we show the convergence of the gradient methods for different deterministic and stochastic fractional Poisson equations, in one or two dimensions.

Example 1a. We consider the approximation of the following spectrally defined one-dimensional fractional Poisson equation

$$-(-\triangle)^\alpha u - (-\triangle)^\beta u = f,$$

on $[-5, 5]$, with $\alpha = 0.75$, $\beta = 0.5$ and $f(x_1) = \exp(-x_1^2)$, with null Dirichlet boundary condition, by the following 2-matrix MFLS $\big(A^\alpha + A^\beta\big)x = b$. The matrix $A \in \mathbb{R}^{n\times n}$ is a five-point stencil finite-difference approximation to $-\triangle$ with $n = 64$. We refer to [3, 20] for details about the justification of this approximation. In this example, we consider 15 iterations of the preconditioned gradient method (Algorithm 4; with $\rho = 1$ and $\beta_{\text{iPiano}} = 0$) and make vary the time step $\Delta\tau$ in the ODE solver (4) from 2^{-6} to 2^{-11}. The solution is represented in Fig. 1. We report the logarithm of the ℓ^2-error surface between the exact solution (computed through the spectrum of A) and the preconditioned gradient solution (Algorithm 4; with $\rho = 1$) as a function of $\Delta\tau$ and the number of iterations k in Fig. 2, using Crank-Nicolson (Left) and Runge-Kutta

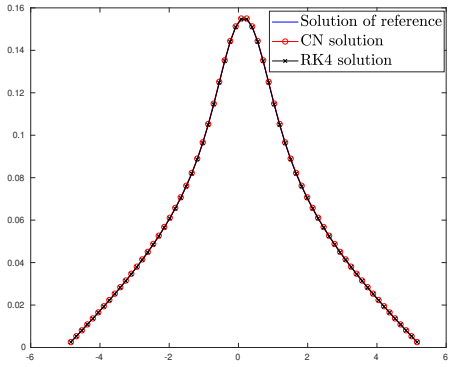

Figure 1: **Example 1a.** Solution to the fractional Poisson equation.

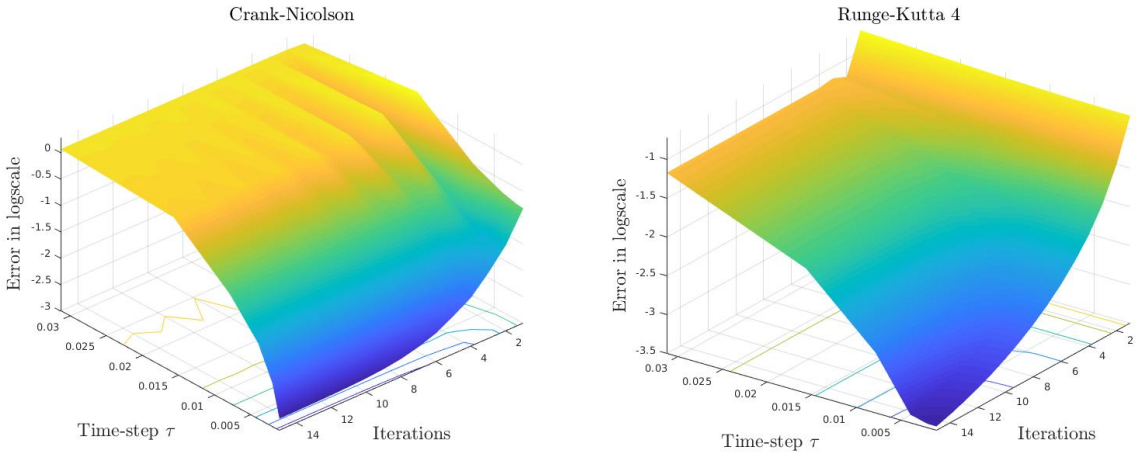

Figure 2: **Example 1a.** ℓ^2-norm error in logscale as a function of the time-step $\Delta\tau$ and iteration k of the preconditioned gradient (Algorithm 4; with $\rho = 1$ and $\beta_{\text{iPiano}} = 0$) with CN (Left) and RK4 (Right).

4 (Right). We also plot in Fig. 3 the error for the truncated Taylor's series expansion given by (17), with Crank-Nicolson and Runge-Kutta 4, as a function of the truncation index M and $\Delta\tau$. We observe that both methods are convergent, and naturally the RK4 solver is more accurate than CN.

Example 1b. In the following experiment, we compare the error behavior as a function of the number of standard FLS calls using RK4 with a given time step, between the preconditioned gradient (Algorithm 4; with $\rho = 0.5$ and $\beta_{\text{iPiano}} = 0$) and the standard gradient method (Algorithm 3; with $\rho = 0.5$), for approximating the fractional equation $-(-\triangle+V)^{\alpha}u-(-\triangle+V)^{\beta}u = f$ on $(-5, 5)$, where the potential is $V(x_1) = V_0 \exp(-x_1^2/2)$ and for different values of the potential intensity: $V_0 = 0, 10, 100, 500$ (with a corresponding norm $\|A\|_2 \approx 5.2, 5.3, 5.4, 5.7$ and condition number $\kappa(A) \approx 2271, 582, 276, 213$). We again fix $\alpha = 0.75$ and $\beta = 0.5$. We report on Fig. 4 the ℓ^2-norm error as a function of the number of standard FLS to solve (8) with a given time step $\Delta\tau = 2^{-12}$ and $n = 100$. We observe that the preconditioned gradient leads to a faster convergence, although the difference between both methods tends to be less visible when the

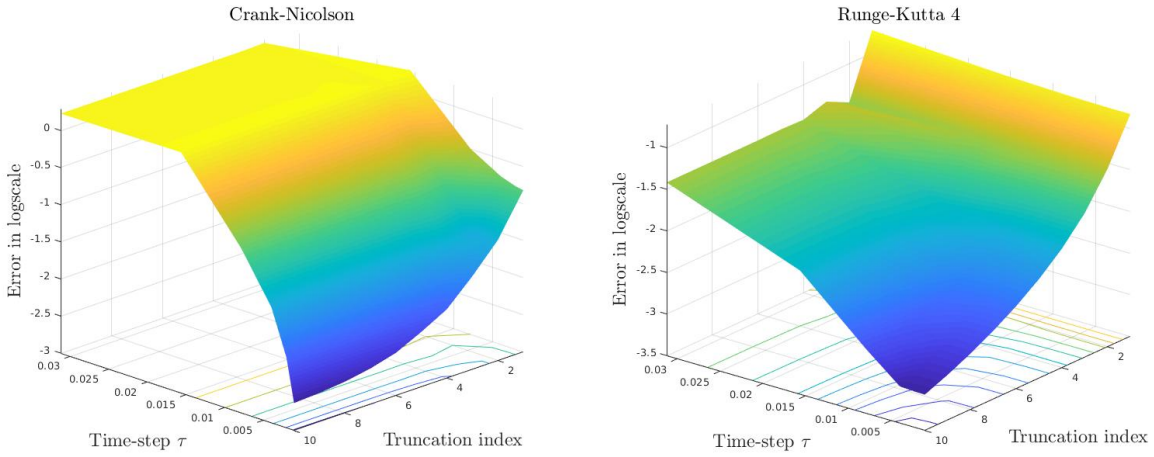

Figure 3: **Example 1a.** ℓ^2-norm error in logscale as a function of the time-step $\Delta\tau$ and truncation index M in the Taylor's series expansion with CN (Left) and RK4 (Right).

potential intensity V_0 gets larger. The preconditioning of the system allows for an improvement of the efficiency of the solver when refining the mesh, i.e. when increasing n. For example, we report on Fig. 5 the case of $V_0 = 100$ with $\alpha = 0.75$ and $\beta = 0.25$ and we plot the number of standard FLS vs. n (= 100, 200, 400) when fixing the error level to 10^{-4}. We observe that the convergence is indeed much faster using the Algorithm 4 than Algorithm 3. Moreover Algorithm 4 convergence is accelerated with increasing n. For all these reasons, we now only retain the preconditioned gradient with RK4, and analyze the effect of β_{iPiano}, the comparison with the GMRES and restarted GMRES is considered next. However, we recall that nevertheless, our conclusions here remain relatively related to the fact that fractional Laplacian-type equations are considered, and that the conclusion may be different when considering general matrices A not connected to fractional PDEs.

Example 1c. We propose now an example with a 4-matrix MFLS corresponding to the finite-difference approximation of the fractional PDE

$$-\sum_{i=1}^{4}(-\triangle u + Vu + \mathcal{W} * u)^{\alpha_i} = f\,, \tag{18}$$

on $(-2, 2)$, with $\alpha_i = 0.1(2i-1)$, for $i = 1, 2, 3, 4$, where $*$ denotes the convolution product, and where we either have

1. $V(x_1) = V_0 \exp(-x_1^2/2)$ with $V_0 = 0$, $V_0 = 10^2$ and $\mathcal{W}(x_1) = 0$, leading to the MFLS $\sum_{i=1}^{4} A^{\alpha_i} x = b$, where A is a finite dimensional approximation of $-\triangle + V$, or
2. $V(x_1) = 0$ and $\mathcal{W}(x_1)$ is a nonlocal random perturbation following a uniform law $\mathcal{N}(0, 10^{-2})$. This corresponds to add to the matrix A (approximating $-\triangle$) a random matrix $N = \text{rand}(n, n)$, and to solve the following MLFS $\sum_{i=1}^{4}(A+N)^{\alpha_i} x = b$.

The function f is given by $f(x_1) = \exp(-2x_1^2)$. Again, we observe on Fig. 6 the convergence in the three situations, with a slightly slower convergence rate for the random potential case.

Example 1d. We consider now A as a two-dimensional finite-difference approximate fractional Laplacian including a gaussian potential $V(x_1, x_2) = \exp(-(x_1^2 + x_2^2)/2)$ on the square domain

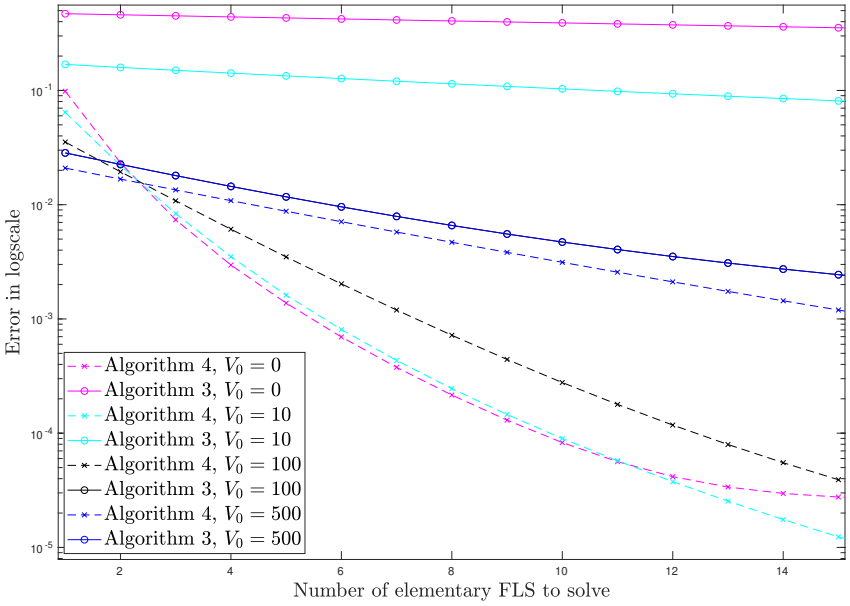

Figure 4: **Experiment 1b.** ℓ^2-norm error in logscale as a function of the number of standard FLS to solve (8) with the preconditioned gradient method (Algorithm 4; with $\rho = 0.5$ and $\beta_{\text{iPiano}} = 0$) and the standard gradient method (Algorithm 3; with $\rho = 0.5$).

$[-1, 1]^2$, with homogeneous Dirichlet boundary conditions. We use a five-point stencil discretization for the Laplacian with 32 points in each direction, leading then to $A \in \mathbb{R}^{32^2 \times 32^2}$. The right hand side b is a projection on the finite-difference grid of the source function $f(x_1, x_2) = \exp(-5(x_1^2 + x_2^2))$. We fix the exponents to be $\alpha = 0.75$ and $\beta = 0.25$. We report the logarithm of the ℓ^2-error between the reference solution and the solution computed by the preconditioned gradient (Algorithm 4; with $\rho = 1$ and $\beta_{\text{iPiano}} = 0$) as a function of $\Delta\tau = 2^{-10}$, and the number of iterations k in Fig. 7, illustrating again the good convergence of the preconditioned gradient method.

Example 2. We compare now the convergence rate of gradient methods with possibly iPiano term β_{iPiano} and the GMRES approach [24] on the preconditioned system (see Subsection 2.2).
Example 2a. The matrix $A \in \mathbb{R}^{n \times n}$ with $n = 100$, is an approximate one-dimensional five-point stencil Laplacian including a potential $V(x_1) = 10 \exp(-2x_1^2)$, on $[-5, 5]$. We solve $(A^\alpha + A^\beta)x = b$, with $\alpha = 0.75$ and $\beta = 0.25$. We take $\Delta\tau = 2^{-10}$ and report the graph of convergence in Fig. 8 (Left). More specifically, we are interested in the comparison between GMRES and the iPiano-version of the gradient method introduced in this paper. As expected the convergence rate is strongly dependent on the value of the parameters β_{iPiano} and ρ with the iPiano solver. However, the fastest convergence rate is obtained with GMRES. The best value of β_{iPiano} is 0.1 for these one-dimensional problems
Example 2b. A similar comparison is now proposed when the matrix $A \in \mathbb{R}^{n \times n}$ is a two-dimensional approximate five-point stencil perturbed Laplacian $-\triangle(1 + \mathcal{W}(x_1))$, with $n = 32^2$ points on the computational domain $[-5, 5]^2$, and where $\mathcal{W}(x_1)$ is a local random perturbation (uniform law) $\mathcal{N}(0, 5 \times 10^{-2})$. We then solve $(A^\alpha + A^\beta)x = b$ with $\alpha = 0.9$, $\beta = 0.1$. We take $\Delta\tau = 2^{-6}$ and report the graph of convergence in Fig. 8 (Right). Overall, and as it was

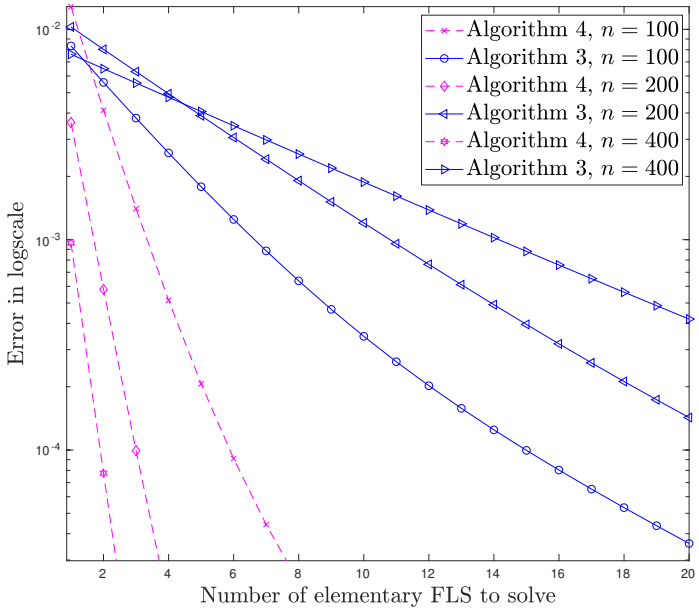

Figure 5: **Experiment 1b.** ℓ^2-norm error in logscale as a function of the number of standard FLS to solve (8) with the preconditioned gradient method (Algorithm 4; with $\rho = 0.5$ and $\beta_{\text{iPiano}} = 0$) and the standard gradient method (Algorithm 3; with $\rho = 0.5$) where $\alpha = 0.75$ and $\beta = 0.25$, and for $n = 100, 200, 400$.

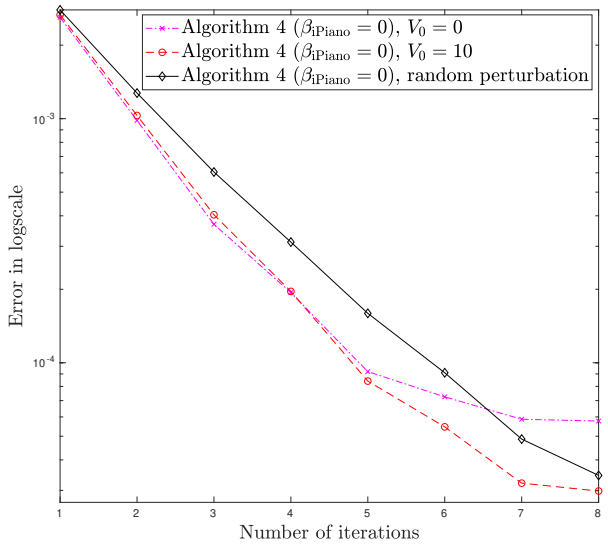

Figure 6: **Example 1c.** ℓ^2-norm error in logscale as a function of the iteration k to solve (18) with the preconditioned gradient (Algorithm 4; with $\beta_{\text{iPiano}} = 0$), for i) $V(x_1) = V_0 \exp(-x_1^2/2)$ with $V_0 = 0$, $V_0 = 10^2$ and ii) $V = 10^{-2}\texttt{rand}(n, n)$.

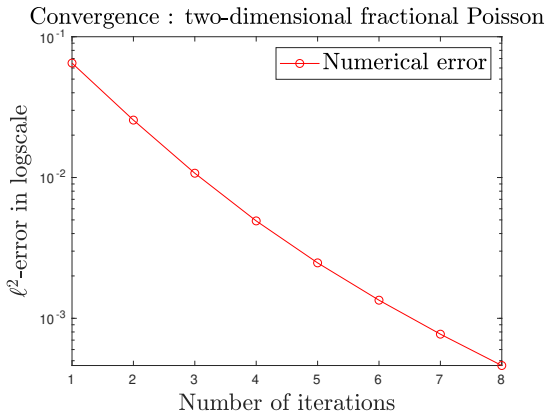

Figure 7: **Example 1d.** ℓ^2-norm error in logscale as a function of the iteration k of the preconditioned gradient (Algorithm 4; with $\beta_{\text{iPiano}} = 0$) for the 2D approximate fractional Poisson equation.

expected for $\alpha - \beta$ not too small, GMRES performs much better than gradient-type methods, including the iPiano versions. With the latter method, the fastest convergence is obtained with $\beta_{\text{iPiano}} = 0.6, 0.7$ and $\rho = 0.4, 0.5$. However, as we will observe in the next experiment, whenever $\alpha - \beta$ is small the conclusion can be different.

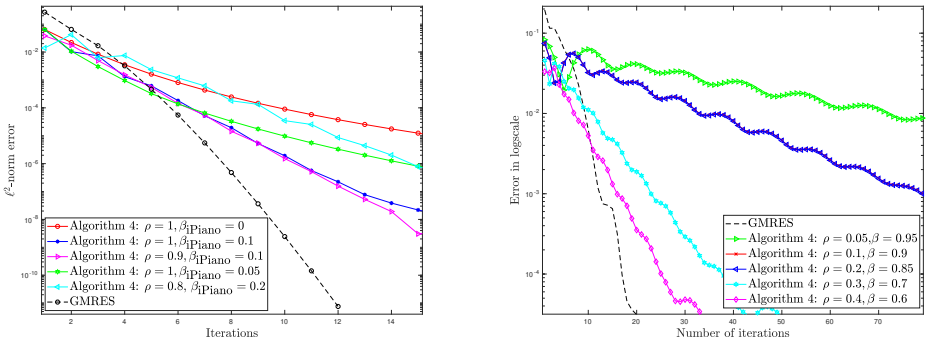

Figure 8: Graph of convergence in ℓ^2-norm. (Left) **Experiment 2a.** One-dimensional fractional Laplacians. (Right) **Experiment 2b.** Two-dimensional perturbed fractional Laplacians.

Example 3. In this example, we illustrate a case where GMRES fails to provide a faster convergence than a gradient method (Algorithm 4) with $\rho = 1$. We consider the system $(A^\alpha - A^{\alpha-\varepsilon})x = b$, where $\varepsilon = 0.1$, 5×10^{-2}, 10^{-2} and $\alpha = 0.9$. The matrix A is defined as a two-dimensional finite-difference approximation of $-(-\triangle + V(x_1, x_2) + \mathcal{W}(x_1, x_2))^\alpha + (-\triangle + V + \mathcal{W})^{\alpha-\varepsilon}$, with homogeneous Dirichlet boundary conditions, where $V(x_1, x_2) = 100 \exp\big(-(x_1^2 + x_2^2)/50\big)$ and $\mathcal{W}(x_1, x_2)$ follows an uniform law $\mathcal{N}(0, 5 \times 10^{-2})$. The tolerance is fixed to 10^{-9} for the convergence of the GMRES with a restart value equal to 40, as well as a restart parameter at 10 iterations when $\beta = 0.89$ for comparison. In this case, the standard FLS are assumed to be solved exactly at each iteration. The numerical results reported on Fig. 9 show that the smaller ε, the slower the GMRES convergence, and in this case a standard gradient method

indeed performs better, which is consistent with the discussion from Subsection 2.2.

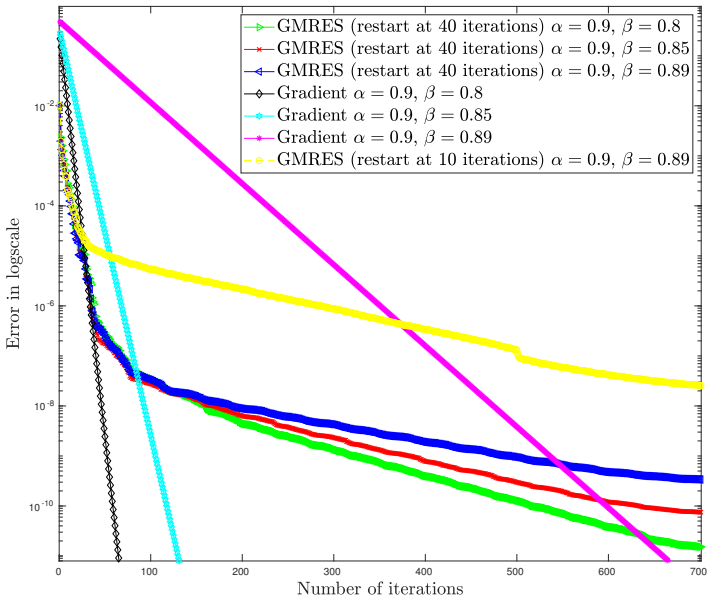

Figure 9: **Experiment 3.** Graph of convergence in ℓ^2-norm : GMRES and preconditioned gradient method (Algorithm 4).

3 A Partial Differential Equation-based solver

In this Section, we propose an original PDE-based method for solving (1). We still assume that $1 > \alpha_1 > \alpha_2 > \cdots > \alpha_N > 0$, with $N \geqslant 2$. The proposed method can be seen as an extension of the ODE-based solver (4) to N-variables. We will also show in Subsection 3.4 that, at the discrete level, some approximate PDE solvers can actually degenerate into a gradient method.

3.1 Strategy for the N-matrix MFLS

The objective is to derive a PDE whose solution at a given point provides the solution to (1). To this end, we introduce N real-valued variables t_i, for $i = 1, \cdots, N$, and we assume that the solution x to the PDE on $[0, 1/N]^N$ is of the following form

$$x(t_1, \cdots, t_N) \quad = \quad \big((1 - \sum_{i=1}^{N} t_i)I + t_i A^{\alpha_i}\big)^{-1} b\,.$$

Hence, we notice that $x(0, \cdots, 0) = b$ and that $N^{-1}x\big(1/N, \cdots, 1/N\big) = \big(\sum_{i=1}^{N} A^{\alpha_i}\big)^{-1} b$ which is then the solution to (1). In order to derive the PDE, we remark that for any $i \in \{1, \cdots, N\}$

$$\begin{aligned} \partial_{t_i} x(t_1, \cdots, t_N) \quad &= \quad \big((1 - \sum_{i=1}^{N} t_i)I + \sum_{i=1}^{N} t_i A^{\alpha_i}\big)^{-2} b - A^{\alpha_i}\big((1 - \sum_{i=1}^{N} t_i)I + t_i A^{\alpha_i}\big)^{-2} b \\ &= \quad (I - A^{\alpha_i})\big((1 - \sum_{i=1}^{N} t_i)I + t_i A^{\alpha_i}\big)^{-1} x(t_1, \cdots, t_N)\,. \end{aligned}$$

We then deduce that for any $(i,j) \in \{1,\cdots,N\}^2$ with $i \neq j$:

$$\big(I - A^{\alpha_j}\big)\partial_{t_i} x(t_1,\cdots,t_N) \;=\; \big(I - A^{\alpha_i}\big)\partial_{t_j} x(t_1,\cdots,t_N)\,.$$

In a forthcoming paper, we will propose a complete analysis of the corresponding PDE and its efficient approximation for the general case. In this paper, we focus on the two-matrix version ($N = 2$) of the algorithm.

3.2 The case of the 2-matrix MFLS

We propose to explicitly derive the algorithm for $N = 2$ with $0 < \beta < \alpha$, i.e.

$$(A^\alpha + A^\beta)x \;=\; b\,. \tag{19}$$

For computational efficiency, we slightly transform the system. Let us set $\delta = (\alpha+\beta)/2$ and $\gamma = (\alpha-\beta)/2$ in such a way that we now consider

$$A^\delta(A^\gamma + A^{-\gamma})x \;=\; b\,.$$

This system can be solved in two steps: i) a standard FLS $A^\delta y = b$, and then ii) $(A^\gamma + A^{-\gamma})x = y$. Hence, for $y \in \mathbb{R}^n$, we focus on the computation of the MFLS

$$(A^\gamma + A^{-\gamma})x \;=\; y\,.$$

For $(t,\tau) \in [0,1/2]^2$, we consider

$$x(t,\tau) \;=\; \Big(\big(1-(t+\tau)\big)I + tA^{-\gamma} + \tau A^\gamma\Big)^{-1} y\,,$$

solution to

$$\big(I - A^\gamma\big)\partial_t x(t,\tau) \;=\; \big(I - A^{-\gamma}\big)\partial_\tau x(t,\tau),\; x(0,0) = y\,, \tag{20}$$

and such that $x(1/2,1/2) = (A^\gamma + A^{-\gamma})^{-1}y$ which is the expected solution. Then, we get

$$\begin{aligned}
\partial_t x(t,\tau) &= \big(I - A^\gamma\big)^{-1}\big(I - A^{-\gamma}\big)\partial_\tau x(t,\tau)\\
&= A^{-\gamma}\big(A^{-\gamma/2} - A^{\gamma/2}\big)^{-1}\big(A^{\gamma/2} - A^{-\gamma/2}\big)\partial_\tau x(t,\tau)\\
&= -A^{-\gamma}\partial_\tau x(t,\tau)\,.
\end{aligned}$$

Interestingly, if $A^{-\gamma}$ is diagonalizable in $\mathbb{R}$, we obtain a first-order one-dimensional linear hyperbolic system of conservation laws (HSCL) for $(t,\tau) \in [0,1/2]^2$,

$$\partial_t x(t,\tau) + A^{-\gamma}\partial_\tau x(t,\tau) \;=\; 0\,. \tag{21}$$

For wellposedness reasons, it is then necessary to impose some boundary conditions at $\tau = 0$ and/or $\tau = 1/2$ *a priori* depending on the sign of the eigenvalues of $A^{-\gamma}$. Let us recall for the readers who are unfamiliar with first-order HSCL that systems of conservation laws $\partial_t x + \partial_\tau f(x) = 0$ are called hyperbolic if the Jacobian matrix $df(x)$ is diagonalizable in $\mathbb{R}$ for any $x \in \mathbb{R}^n$. We refer to [6, 9, 19] for an exhaustive presentation of this type of PDEs. The system (21) is hence linear with constant jacobian matrix $A^{-\gamma}$. Such systems can be solved analytically in $\mathbb{R}\times\mathbb{R}_+$, provided we have access to the eigenvalues $\{\mu_k^{-\gamma}\}_k$, and to left/right eigenvectors $\{\ell_k\}_k$, $\{r_k\}_k$, to $A^{-\gamma}$. Practically, this is usually not possible for large and sparse matrices A or $A^{-\gamma}$. Moreover, system (21) has to be solved for $\tau \in [0,1/2]$, with $x(0,\tau) = \big((1-\tau)I + \tau A^\gamma\big)^{-1} y$

and $x(t,0) = \big((1-t)I + tA^{-\gamma}\big)^{-1} y$. As a consequence, we cannot use a standard scheme based on an approximate Riemann solver [27] *if* we do not have access to the sign of the eigenvalues of A or $A^{-\gamma}$. *It is however important to recall that the proposed method does not require* $A^{-\gamma}$ *to be diagonalizable in* $\mathbb{R}$. For any $\Delta t > 0$, we consider in the following the approximate solution to (20) from t to $t + \Delta t$.

Let us now approximate this equation on $[0,1/2]^2$, with $x(0,\tau) = \big((1-\tau)I + \tau A^{\gamma}\big)^{-1} y$ and $x(t,0) = \big((1-t)I + tA^{-\gamma}\big)^{-1} y$. The latter are discretized as described below. We denote by x_j^i an approximation to $x(t_i,\tau_j)$, for $i,j = 0$ to $(i,j) = (M_t, M_\tau)$, where $0 = t_0 < \cdots < t_{M_t} = 1/2$ and $0 = \tau_0 < \cdots < \tau_{M_\tau} = 1/2$. For convenience, we assume that $\Delta\tau = \tau_{j+1} - \tau_j = 1/2M_\tau$ (resp. $\Delta t = t_{i+1} - t_i = 1/2M_t$) is constant. The solution to the PDE can be approximately performed by using standard numerical solvers [27]. Hence, denoting by $x_{M_\tau}^{M_t}$ an approximation to the solution at $(1,1)$ of (19) by using an order p solver in t (with time step Δt) and q in τ (with time step $\Delta\tau$), then there exist two constants $c_t(A)$ and $c_\tau(A)$ such that

$$\|x_{M_\tau}^{M^t} - x\| \quad \leqslant \quad c_t(A)\Delta t^p + c_\tau(A)\Delta\tau^q\,.$$

The approximate derivatives can be computed for instance, as follows

$$D_- x_i^j = \frac{x_i^j - x_{i-1}^j}{\Delta t},\ D_0 x_i^j = \frac{x_{i+1}^j - x_{i-1}^j}{2\Delta t},\ D_+ x_i^j = \frac{x_{i+1}^j - x_i^j}{\Delta t},$$
$$D^- x_i^j = \frac{x_i^j - x_i^{j-1}}{\Delta\tau},\ D^0 x_i^j = \frac{x_i^{j+1} - x_i^{j-1}}{2\Delta\tau},\ D^+ x_i^j = \frac{x_i^{j+1} - x_i^j}{\Delta\tau}.$$

A convergent approximation to (19) is naturally dependent on A, and a general solver reads

$$\sum_{k=-r}^{r} c_k x_{i+k}^{j+1} \quad = A^{-\gamma}\frac{\Delta t}{\Delta\tau}\sum_{k=-p}^{s} e_k \sum_{\ell=-q}^{q} d_\ell x_{i+\ell}^{j+k}\,, \tag{22}$$

for some coefficients $\{c_k\}_k$, $\{d_k\}_k$ and $\{e_k\}_k$ involved in the approximate derivatives and for some integers p, q, and $s \leqslant 1$.

Practically, this approach could be very inefficient if it is necessary to compute two standard FLS for each t_i, τ_j, leading to a $O(M_t M_\tau)$ FLS calls for computing the approximate solution $x_{M_\tau}^{M^t}$. Moreover, stability issues should be addressed which can limit the size of the time step. To improve the efficiency of the method by taking larger time steps Δt and $\Delta\tau$ (and then smaller values of M_τ and M_t), we could consider higher-order approximate PDE solvers or/and use implicit solvers (corresponding to $s = 1$ in (22)). However, the latter would also lead to the need for computing higher dimensional systems. Since standard explicit finite-difference schemes for first-order hyperbolic systems usually have a small stencil, it is in fact *not* necessary to compute x_i^j for any $0 \leqslant i \leqslant M_t$ and $0 \leqslant j \leqslant M_\tau$. For instance, let us assume that an upwind scheme involving only x_{j-1}^i and x_j^i is employed to compute x_j^{i+1}, and eventually estimate $x_{M_\tau}^{M^t}$. Then, we only need to compute $x_{M_\tau-1}^{M_t-1}$, $x_{M_\tau}^{M_t-1}$, and then $x_{M_\tau-2}^{M_t-2}$, $x_{M_\tau-1}^{M_t-2}$,...,$x_{M_\tau-k}^{M_t-k}$, $x_{M_\tau-k+1}^{M_t-k}$, for $k \leqslant M_t$. This corresponds to computing $4M_\tau + 1$ FLS rather than $2M_t M_\tau + 1$, with a Lax-Friedrichs scheme or $5M_t M_\tau + 1$ with a Lax-Wendroff scheme [26]. For A diagonalizable in $\mathbb{R}$, a stable Lax-Friedrichs scheme (or upwind scheme if we have information on the sign of the eigenvalues of A) can easily be implemented (see Step 4 of Algorithm 6). Regarding the initial condition, we can for instance iteratively proceed as follows. From the representation

$$\int_\tau^{\tau+\Delta\tau} \big((1-s)I + sA^{\pm\gamma}\big)x(0,s)ds = y,$$

Algorithm 6 PDE-based solvers

1: Solve $A^\delta y = b$ and set $x_0^0 = y$.

2: For $j \geqslant 0$, compute x_j^0, for any $0 \leqslant j \leqslant M_\tau$, with

$$x_{j+1}^0 = y + \tau_j (I - A^\gamma) x_j^0 \,.$$

3: Compute x_0^i, for any $0 \leqslant i \leqslant M_t$, with

$$x_0^{i+1} = y + t_i (I - A^{-\gamma}) x_0^i \,.$$

4: For any $0 \leqslant i \leqslant M_t - 1$ and any $1 \leqslant j \leqslant M_\tau$

$$x_j^{i+1} = \frac{1}{2}(x_{j+1}^i + x_{j-1}^i) - \frac{\Delta t}{2\Delta\tau} A^{-\gamma}(x_{j+1}^i - x_{j-1}^i) \text{ (Lax-Friedrichs scheme)}$$

or

$$x_j^{i+1} = x_j^i - \frac{\Delta t}{\Delta\tau} A^{-\gamma}(x_j^i - x_{j-1}^i) \text{ (Upwind scheme)}$$

or

$$x_j^{i+1} = x_j^i - \frac{\Delta t}{2\Delta\tau} A^{-\gamma}(x_{j+1}^i - x_{j-1}^i) + \frac{\Delta t^2}{2\Delta\tau} A^{-2\gamma}(x_{j+1}^i - 2x_j^i + x_{j-1}^i) \text{ (Lax-Wendroff scheme)}$$

5: Deduce the solution to (19) : $x = 2x_{M_\tau}^{M_t}$.

we can use an approximation by using any standard quadrature rule over $[\tau, \tau + \Delta\tau]$ and FLS calls (see Steps 1 and 2 in Algorithm 6). We finally obtain Algorithm 6 for the PDE-based solver.

Using the standard theory of numerical analysis for hyperbolic systems of conservation laws [26, 27], we have the following proposition.

Proposition 3 *The Algorithm 6 with the upwind scheme for computing* (19)*, where A is diagonalizable in $\mathbb{R}$ (with eigenvalues λ_k, $1 \leqslant k \leqslant n$) is convergent under the Courant-Friedrichs-Lewy (CFL) condition*

$$\max_{1 \leqslant k \leqslant n} \left|\lambda_k^{(\beta-\alpha)/2}\right| \frac{\Delta t}{\Delta \tau} \leqslant 1\,. \tag{23}$$

Moreover, there exist two positive constants $c_t(A) > 0$ and $c_\tau(A) > 0$ such that

$$\|x^{M_t} - 2x\| \quad \leqslant \quad c_t \Delta t + c_\tau \Delta \tau\,,$$

where $x^{M_t} = (x_0^{M_t}, \cdots, x_{M_\tau}^{M_t})^T$ and the approximate solution to (21) *is $x_{M_\tau}^{M_t}/2$.*

The stability condition can easily be fulfilled by computing the eigenvalue with largest modulus of A. This PDE-based solver will be studied in more details in a forthcoming paper since we think that there is a potential for improvement of the methodology proposed here.

3.3 Numerical examples

In this subsection, we propose some simple examples to illustrate this new approach. Future investigations and experiments will be proposed in a forthcoming paper.

Example 4a. In this experiment, we numerically illustrate the convergence of the PDE-based solver derived above. We consider the system $(A^\alpha + A^\beta)x = b$ with $\alpha = 0.9$, $\beta = 0.1$, $A \in \mathbb{R}^{256\times 256}$ and we implement Algorithm 6 approximating the following Poisson equation on $[-1/4, 1/4]$, $-K\big((-\triangle)^\alpha u - (-\triangle)^\beta\big)u = f$ with null Dirichlet boundary conditions, $K = 10^{-3}$, and where $f(x_1) = \exp(-50x_1^2)$. We again use the spectral definition of the fractional Laplacian which allows us to approximate the fractional Poisson equation by a MLFS $(A^\alpha + A^\beta)x = b$. The standard FLS are solved using RK4 with a very small time-step. The PDE is solved by using i) upwind scheme, and ii) Lax-Friedrichs scheme which respectively require to solve $4M_\tau + 1$, $2M_tM_\tau + 1$ standard FLS. Let us recall that unlike the upwind schemes, the Lax-Friedrichs scheme does not require the knowledge of the sign of the eigenvalues. We take $M_t = 8$, and let M_τ vary from 4 to 256 (corresponding to space-steps from 2^{-4} to 2^{-10}). In Fig. 10 (Top-Left), we report the corresponding reference and numerical solutions at $(t, \tau) = (1/2, 1/2)$. We plot on Fig. 10 (Top-Right) the norm error between the PDE-based solution and a solution of reference, $|x_{M_\tau}^{M_t} - x(1/2, 1/2)|$ as a function of $\Delta\tau$. Notice that the loss of precision for $\Delta\tau$ small enough is in fact due to the scheme instability which appears when the CFL condition is violated (23). This is confirmed numerically in Fig. 10 (Bottom), where the CFL number $\left|\lambda_{\max}\right|^{(\alpha-\beta)/2}\Delta t/\Delta\tau$ is reported vs. $\Delta\tau$. Future investigations should include higher order convergent schemes and more efficiency comparisons.

Example 4b. We consider now the same test as before, but for the exponents $(\alpha, \beta) = (1.9.0.1)$ (the rest of the data is similar). We report the results on Fig. 11. We observe that all the schemes are now stable for any values of $\Delta\tau$ and that overall the three schemes behave similarly.

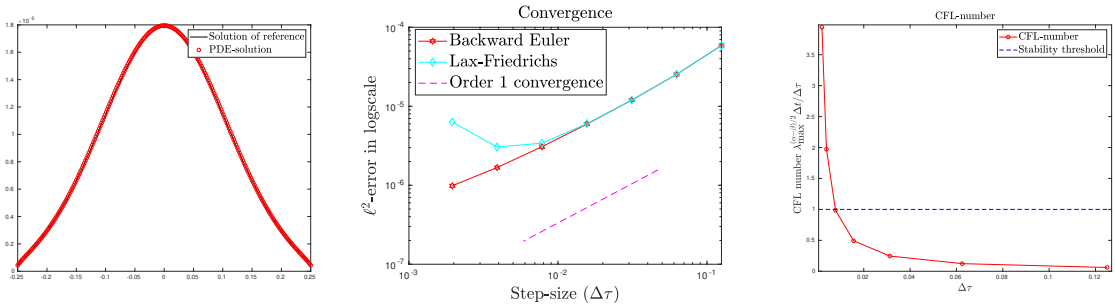

Figure 10: **Example 4a.** $(\alpha, \beta) = (0.9.0.1)$. (Top-Left) Solution of reference and PDE-based numerical solution. (Top-Right) Graph of convergence. (Bottom) CFL-number as a function of $\Delta\tau$.

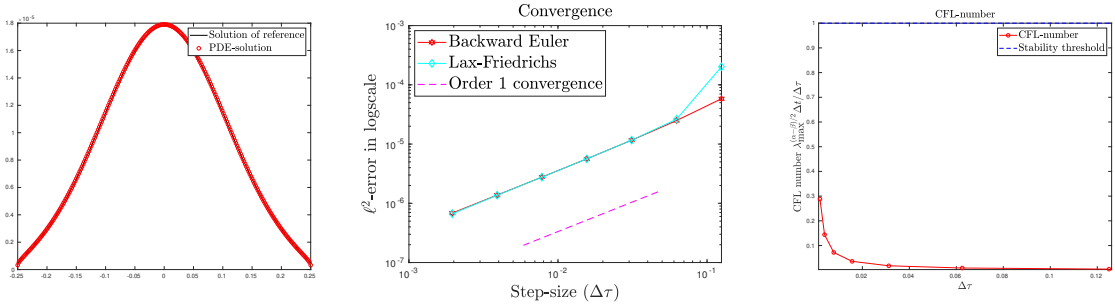

Figure 11: **Experiment 4b.** $(\alpha, \beta) = (1.9.0.1)$, (Left) Solution of reference and PDE-based numerical solution. (Middle) Graph of convergence. (Right) CFL-number as a function of $\Delta\tau$.

Experiment 4c. We proceed with the same configuration as **Example 4b**, but where we solve

$$-K\big((-\triangle + \mathcal{W})^{\alpha} u - (-\triangle + \mathcal{W})^{\beta}\big) u = f,$$

where $\mathcal{W}$ is a random potential (uniform law) $\mathcal{N}(0, 10^3)$. In this case, the eigenvalues of $A^{(\alpha-\beta)/2}$ are complex. We globally observe a good behavior of the schemes on Fig. 12, with again stability issues for enough $\Delta\tau$.

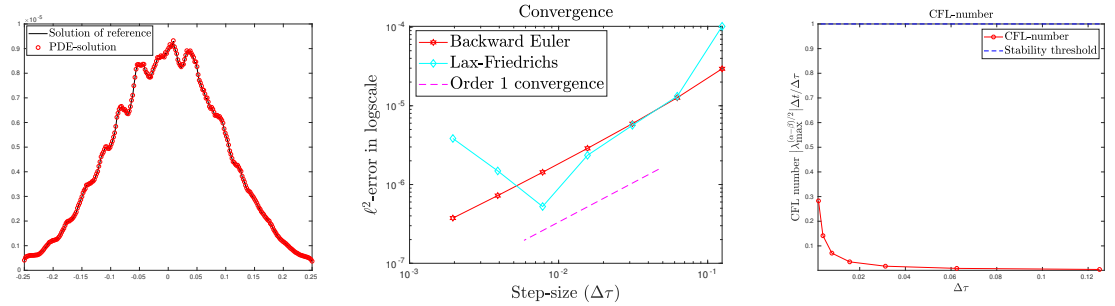

Figure 12: **Experiment 4c.** (Left) Solution of reference and PDE-based numerical solution. (Middle) Graph of convergence. (Right) CFL-number as a function of $\Delta\tau$.

3.4 Connection between the approximate PDE-solver and the gradient method

In this subsection, we are interested in the link between the PDE-based approach and the gradient methods developed in Subsection 2.1, for solving

$$(A^\alpha + A^\beta)x \quad = \quad b\,. \tag{24}$$

We set $\delta = (\alpha+\beta)/2$ and $\gamma = (\alpha-\beta)/2$, and as explained above, the PDE-based method is applied to

$$(A^\gamma + A^{-\gamma})x \quad = \quad A^{-\delta}b\,. \tag{25}$$

The preconditioned gradient method with $\rho = 1$ leads to the sequence of iterates

$$x_{k+1} \quad = \quad A^{-\alpha}b - A^{\alpha-\beta}x_k\,. \tag{26}$$

Let us first recall some basic facts about the solutions to hyperbolic systems. For $(t,\tau) \in [0,1/2]^2$ and $A \in \mathbb{R}^n$ diagonalizable in $\mathbb{R}_+$, we consider the following system

$$\partial_t x + A^{-\gamma}\partial_\tau x \quad = \quad 0,\;\; x(0,\tau) = x_0(\tau),\;\; x(t,0) = x^0(t)\,. \tag{27}$$

For $\tau > t$, the solution to this system reads

$$x(t,\tau) \quad = \quad \sum_{s=1}^{n} \ell_s^T x_0(\tau - \mu_s^{-\gamma} t) r_s\,,$$

where $\{\ell_s\}_s$ and $\{r_s\}_s$ are the left and right eigenvectors of A and $\{\mu_s\}_s$ the corresponding eigenvalues.

We consider now an explicit approximate PDE-solver $\{y_i^k\}_{1\leqslant i\leqslant M_\tau}^{1\leqslant k\leqslant M_t}$ in the form:

$$y_i^{k+1} \quad = \quad y_i^k - \frac{\Delta t}{\Delta \tau} A^{-\gamma} \sum_{\ell=0}^{q} d_\ell y_{i-\ell}^k\,, \tag{28}$$

where $\Delta t = 1/2M_t$ and $\Delta\tau = 1/2M_\tau$. Assuming that the scheme is convergent at order p in space (τ) and 1 in time (t), we then have $y_{M_\tau}^{M_t} \propto (A^{-\gamma} + A^{-\gamma})^{-1} b + O(\Delta\tau^p) + O(\Delta t)$. For the sake of simplicity, we assumed i) that the scheme is upwinded, which allows to avoid the boundary condition at $\tau = 1/2$, and ii) that $M_\tau > M_t$ in order to avoid the boundary condition issues at $\tau = 0$. The computational solver can be rewritten using matrix notation. We denote for $k \leqslant M_t$, $Y_k := \{y_i^k\}_{1\leqslant i\leqslant M_\tau}$, such that

$$Y_{k+1} \quad = \quad \mathcal{A} Y_k\,,$$

where the block matrix $\mathcal{A}$ with $M_\tau \times M_\tau$ blocks of size $n \times n$, is such that for $1 \leqslant i \leqslant M_\tau$ and $1 \leqslant \ell \leqslant q$

$$\mathcal{A}_{i;i} = I - d_0 \frac{\Delta t}{\Delta\tau} A^{-\gamma},\;\; \mathcal{A}_{i;i-\ell} = -d_\ell \frac{\Delta t}{\Delta\tau} A^{-\gamma}\,.$$

We prove the following result.

Proposition 4 *The PDE solver* (28) *can be reformulated as a preconditioned gradient method* (26) *for solving* (24).

Proof. We denote $z_k := y_k^k$ in (28), and we reformulate the latter for $k < M_t$, as the following iterative method

$$z_{k+1} \;=\; -d_1 \frac{\Delta t}{\Delta \tau} A^{-\gamma} z_k + F_k, \tag{29}$$

where F_k is a correction term, defined as

$$F_k := \Big(I - \frac{\Delta t}{\Delta \tau} d_0 A^{-\gamma}\Big) y_{k+1}^k - \frac{\Delta t}{\Delta \tau} A^{-\gamma} \sum_{\ell=2}^{q} d_\ell y_{k+1-\ell}^k \,. \tag{30}$$

Now in (30), $y_{k+1-\ell}^k$ is such that by construction of the weights $\{d_\ell\}_\ell$ and by projection on the grid

$$\begin{aligned} y_{k+1-\ell}^k &= \textstyle\sum_{s=1}^n \ell_s^T x_0(\tau_{k+1-\ell} - \mu_s^{-\gamma} t_{k+1}) r_s + O(k\Delta t) + O(\Delta \tau^q) \\ &= \textstyle\sum_{s=1}^n \ell_s^T x^0_{\lfloor k+1-\ell-\mu_s^{-\gamma} t_{k+1} \rfloor} r_s + O(k\Delta t) + O(\Delta \tau) \,. \end{aligned}$$

Thus, we have

$$\begin{aligned} F_k &= y_{k+1}^k - \frac{\Delta t}{\Delta \tau} A^{-\gamma} \textstyle\sum_{\ell=0;\ell\neq 1}^q d_\ell y_{k+1-\ell}^k \\ &= \textstyle\sum_{s=1}^n \ell_s^T \Big(x^0_{\lfloor k+1-\mu_s^{-\gamma} t_{k+1} \rfloor} - \frac{\Delta t}{\Delta \tau} A^{-\gamma} \sum_{\ell=0;\ell\neq 1}^q d_\ell x^0_{\lfloor k+1-\ell-\mu_s^{-\gamma} t_{k+1} \rfloor} \Big) r_s + O(k\Delta t) + O(\Delta \tau) \,. \end{aligned}$$

Let us recall that initially $x(0,\tau) = \big((1-\tau)I + \tau A^\gamma\big)^{-1} A^{-\delta} b$, with in particular $x(0,0) = A^{-\delta} b$. In order to evaluate $x(0,\tau)$, we proposed to iteratively compute

$$\int_\tau^{\tau+\Delta\tau} \big((1-s)I + sA^\gamma\big) x(0,s) ds \;=\; A^{-\delta} b.$$

In other words F_k is only related to the computation of $A^{-\delta} b$, through $\{x_j^0\}_{1 \leqslant j \leqslant M_\tau}$.

Finally, by taking i) $-d_0 \Delta t / \Delta \tau = -1$, and ii) setting $x_k := A^{-\gamma} z_k$ (such that $x_0 = A^{-\gamma-\delta} b = A^{-\alpha} b$), the iterative method (29) reads (since $2\gamma = \alpha - \beta$)

$$x_{k+1} \;=\; A^{-\gamma} F_k - A^{\beta-\alpha} x_k \,,$$

and where $A^{-\gamma} F_k$, by consistency, is an approximation to $A^{-\alpha} b/2$. We hence recover the preconditioned gradient method (26). □

4 Extensions and remarks

4.1 MFLS involving several matrices

We shortly discuss natural extensions of the methods developed in this paper to systems of the form

$$(A^\alpha + B^\beta) x \;=\; b \,, \tag{31}$$

where $0 < \beta < \alpha < 1$ and A and B are two distinct matrices. This system can still be seen as an approximation to a fractional PDE, where A and B are now two distinct finite-dimensional approximate differential operators. For the computational efficiency, we slightly modify the system. Let us rewrite (31) as follows

$$A^{\alpha/2} (A^{\alpha/2} B^{-\beta/2} + A^{\alpha/2} B^{-\beta/2}) B^{\beta/2} x \;=\; b \,. \tag{32}$$

Such systems can then be solved in 3 steps

1. Solve $A^{\alpha/2}z = b$ by using (4).
2. Solve $(C + C^{-1})y = z$, where $C = A^{\alpha/2}B^{-\beta/2}$. This system can be solved by using the methods developed in this paper for 2-matrix MFLS.
3. Solve $B^{\beta/2}x = y$.

4.2 Fractional PDEs with variable exponents

In a future work, we will also be interested in equations of the form

$$-\sum_{i=1}^{N}(-\triangle)^{\alpha_i(\boldsymbol{x})}u \;\; = \;\; f\,,$$

on a bounded domain Ω, with null Dirichlet boundary condition on $\partial\Omega$ and where $\{\alpha_i\}_{i=1,\cdots,N}$ are N space dependent functions from Ω to $(0,1)$. Such models typically correspond to medium non-homogeneities in the framework of material science. Introducing a K-point real space grid $\{\xi_k\}_{1\leqslant k\leqslant K}$ on Ω, using again the spectral definition of the fractional Laplacian, we aim to numerically solve, for all $1 \leqslant k \leqslant K$,

$$\sum_{i=1}^{N} A^{\alpha_i(\xi_k)}x_k \;\; = \;\; b\,,$$

which provides $\mathbf{x} = (x_1, \cdots, x_K) \in \mathbb{R}^{K\times K}$. This problem is hence an extremely complex problem, as it requires the computation of K MFLS, which can however be embarrassingly implemented in parallel. Notice in particular that the intermediate FLS can now be solved using an ODE-solver for

$$\partial_\tau x(\tau,\xi) = -\alpha(\xi)(A-I)\big(I+\tau(A-I)\big)^{-1}x(\tau,\xi), \qquad x(0,\xi) = b \in \mathbb{R}^n\,, \tag{33}$$

whose solution $x(\tau,\xi) = \big(I+\tau(A-I)\big)^{\alpha(\xi)}b$ is such that $x(1,\xi) = A^{-\alpha(\xi)}b$.

4.3 Time-dependent fractional PDE

Compared to the standard fractional Poisson equation $-(-\triangle)^\alpha u = f$, it was shown in this paper that fractional equations involving several fractional Laplace operators $-\sum_\alpha(-\triangle)^\alpha u = f$, is in general much more computationally complex. However, the conclusion can interestingly be different in the time-dependent extension to fractional heat or Schrödinger equations [20]. Let us consider FPDEs on $(t,\boldsymbol{x}) \in [0,T]\times\Omega$ with null Dirichlet boundary conditions, that we assume to be well-posed, of the form

$$\partial_t u(t,\boldsymbol{x}) = \sum_{i=1}^{N} c_i(-\triangle)^{\alpha_i}u(t,\boldsymbol{x}),\;\; u(0,\boldsymbol{x}) = u_0(\boldsymbol{x})\,, \tag{34}$$

where $\{c_i\}_{i=1,\cdots,N}$ are some given complex numbers. We denote by $\boldsymbol{u}(t) \in \mathbb{R}^n$ an approximate solution to (34) at time t. A semi-discrete in space version of the equation would lead to at least one MFLS of the form

$$\partial_t \boldsymbol{u} \;\; = \;\; \sum_{i=1}^{N} c_i A^{\alpha_i}\boldsymbol{u}, \;\; \text{for any } \;(t,\boldsymbol{x}) \in [T, T+\Delta T]\times\Omega\,. \tag{35}$$

In other words, from time T to $T+\Delta T$, if an explicit numerical scheme is used, the computation of (35) can be performed by just evaluating $A^{\alpha_i}\boldsymbol{u}(t)$ for all $i \in \{1, \cdots, N\}$. For stability reasons, it may be more appropriate to use an implicit scheme which however would lead to the computation to MFLS. In order to drastically improve the efficiency of the computation, we can simply use an operator splitting method with an explicit numerical scheme for each split equation. Thus, denoting $T_i = T + \Delta T$, for $i = 1, \cdots, N$, we solve

$$\left\{\begin{array}{lcl} \partial_t \boldsymbol{v}_1 &=& c_1 A^{\alpha_1}\boldsymbol{v}_1 \ \text{ on } \ \in [T, T_1]\times\Omega,\ \boldsymbol{v}_1(0,\cdot) = \boldsymbol{u}(T)\,, \\ \partial_t \boldsymbol{v}_2 &=& c_2 A^{\alpha_2}\boldsymbol{v}_2 \ \text{ on } \ \in [T, T_2]\times\Omega,\ \boldsymbol{v}_2(0,\cdot) = \boldsymbol{v}_1(T_1)\,, \\ \cdots &=& \cdots \\ \partial_t \boldsymbol{v}_N &=& c_N A^{\alpha_N}\boldsymbol{v}_N \ \text{ on } \ \in [T, T_N]\times\Omega,\ \boldsymbol{v}_N(0,\cdot) = \boldsymbol{v}_{N-1}(T_{N-1})\,. \end{array}\right. \tag{36}$$

Hence, each equation of (36) "only" requires the computation to N standard FLS, while the direct computation of (35) would require the computation of MFLS at each iteration. Naturally, for $\boldsymbol{u}(T) = \boldsymbol{v}(T)$ a first-order splitting would lead to $\|\boldsymbol{u}(T+\Delta T) - \boldsymbol{v}(T+\Delta T)\| \leqslant C\Delta T^2$. Higher order splitting schemes would increase the computational complexity, but would also allow to increase ΔT.

5 Conclusion

In this paper, we have derived and analyzed some standard and original computational methods for solving multi-matrix fractional linear systems (MFLS). The latter can appear in the numerical computation of fractional Poisson equations involving several spectrally defined fractional Laplace operators. The computation of MFLS is shown to be very computationally complex, as they require the solution to several intermediate standard fractional linear systems. We have proposed a few relatively efficient gradient-type methods which provided a reasonable convergence rate. However, it was shown that, by far, GMRES is in general the most efficient algorithm, except in some specific cases. A new direct method has also been developed requiring the solution to a first-order partial differential equation. Although the latter allows to use all the existing technology for solving PDE with efficient and higher order convergence, a basic approximation can be very computationally complex if not properly designed. A discussion between the gradient methods and approximate PDE solvers was also presented. In a forthcoming paper, we will analyze in more details this latter approach.

A Accelerated FLS solvers

To accelerate the ODE-based solver for FLS, we propose an original two-step preconditioning technique. The principle is relatively simple and consists in the following steps.

- Construction of a preconditioner P such that $P^{-1}A \approx I$, and such that P and A commute. We refer for instance to [5] where a large number of commuting preconditioners are derived for linear systems solved using a conjugate gradient method. Some of these preconditioners can naturally be used to other iterative Krylov-type solvers.

- Construction on a fine mesh (step $\delta\tau$, and J_τ iterations such that $J_\tau\delta\tau = 1$) of an approximation to $P^{-\alpha}b$, by efficiently numerically solving on $[0,1]$ by an ODE-solver

$$y'(\tau) = -\alpha(P-I)\big(I+\tau(P-I)\big)^{-1}y(\tau), \qquad y(0) = b \in \mathbb{R}^n\,. \tag{37}$$

  This provides a precise approximation $\{y^{(j)}\}_j$ to $y(\tau) = (I+\tau(P-I))^{-\alpha}b$, such that $y(1) = P^{-\alpha}b$. Practically, the solution to linear systems involving P must be computed

much faster than those involving A. Typically, P can be taken as incomplete-LU or Jacobi preconditioners as long as they commute with A.

- Computation on a coarser mesh (step $\Delta T \gg \Delta t$ and $J_T \ll J_\tau$ iterations) with an ODE-solver of

$$z'(\tau) = -\alpha(A-P)\big(P+\tau(A-P)\big)^{-1}z(\tau), \qquad z(0) = P^{-\alpha}b \in \mathbb{R}^n\,. \tag{38}$$

This provides an accurate approximation $\{z^{(j)}\}_j$, even on a coarse grid, to $z(\tau) = (P+\tau(A-P))^{-\alpha}b$, such that $z(1) = A^{-\alpha}b$. *Notice that the commutativity of A and P is required in order to justify that z is indeed solution to* (38). Practically, the initial data in (38) will be numerically given by the approximate solution $y^{(J_T)}$ to $y(1)$ in (37).

To analytically justify the approach and for the sake of simplicity, let us assume that solving a linear system involving P (resp. A) has a quadratic (cubic) complexity and that the equations (37) and (38) are solved by using a standard implicit Euler method (although in practice we will use higher order methods), i.e. there exists $c(P) > 0$ such that

$$\|y^{(J_T)} - P^{-\alpha}b\| \;\leqslant\; c(P)\delta\tau\,.$$

Hence a direct approach (4) has a complexity $O(J_\tau n^3)$ on a fine mesh $\delta\tau$, while the preconditioned approach developed here has a complexity $O(J_\tau n^2 + J_T n^3)$, with $J_T \ll J_\tau$. For instance, if $J_\tau \propto n$, we gain an order of complexity. Regarding the accuracy of the preconditioned technique, let us denote by f the following matrix valued function involved in the original direct ODE-solver (4) and g the one appearing in (38)

$$f(\tau) = -\alpha(I-A)(I+\tau(A-I))^{-1},\;\; g(\tau) = -\alpha(P-A)(P+\tau(A-P))^{-1}\,.$$

Hence, we have

$$df(\tau) = -\alpha(I-A)^2(I+\tau(A-I))^{-2},\;\; dg(\tau) = -\alpha(I-P^{-1}A)^2(I+\tau(P^{-1}A-I))^{-2}\,.$$

Then denoting $\Lambda_f := \sup_{\tau\in[0,1]}\|df(\tau)\|$ and $\Lambda_g := \sup_{\tau\in[0,1]}\|dg(\tau)\|$, for $P^{-1}A \approx I$, we get

$$\Lambda_g \;\lessapprox\; \alpha\|I-P^{-1}A\|^2\,.$$

For instance, if we assume that A is positive definite, a precise estimate of Λ_f is easily obtained as

$$\Lambda_f \;=\; \alpha \sup_{\tau\in[0,1]}\max_{1\leqslant i\leqslant n}\Big[\frac{1-\lambda_i}{1+\tau(\lambda_i-1)}\Big]^2\,.$$

Beyond the order of convergence, the error of the method is typically dependent of $\exp(\Lambda_{f,g})$ (through the constant in the error estimate) [23]. As a consequence the smaller $\Lambda_{f,g}$, the smaller the error. More precisely, for $\{x^{(j)}\}_j$ (resp. $\{z^{(j)}\}_j$) approximate solution to (4) (resp. (38)), we obtain

$$\|x^{(J_\tau)} - A^{-\alpha}b\| \leqslant \exp(\Lambda_f)\delta\tau\,,\;\; \|z^{(J_T)} - A^{-\alpha}b\| \leqslant \exp(\Lambda_g)(\|y^{(J_T)} - P^{-\alpha}b\| + \Delta\tau)\,.$$

Although $\Delta\tau \gg \delta\tau$, this effect is counter-balanced by the fact that $\exp(\Lambda_g) \ll \exp(\Lambda_f)$. Whenever we consider non-commutative preconditioners, the construction of efficient ODE-solvers for FLS is a bit more tricky, and in this case we again refer to [3].

References

[1] X. Antoine, C. Besse, R. Duboscq, and V. Rispoli. Acceleration of the imaginary time method for spectrally computing the stationary states of Gross-Pitaevskii equations. *Computer Physics Communications*, 219:70–78, 2017.

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

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

[4] X. Antoine, E. Lorin, and Y. Zhang. Derivation and analysis of computational methods for fractional laplacian equations with absorbing layers. *Numerical Algorithms*, 2020.

[5] S. F. Ashby, T. A. Manteuffel, and P. E. Saylor. A taxonomy for conjugate gradient methods. *SIAM J. Numer. Anal.*, 27(6):1542–1568, 1990.

[6] S. Benzoni-Gavage and D. Serre. *Multidimensional Hyperbolic Partial Differential Equations*. Oxford Mathematical Monographs. The Clarendon Press, Oxford University Press, Oxford, 2007. First-order systems and applications.

[7] 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.

[8] A. Chambolle and T. Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

[9] C. M. Dafermos. *Hyperbolic Conservation Laws in Continuum Physics*, volume 325 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. Springer-Verlag, Berlin, fourth edition, 2016.

[10] 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.

[11] G. H. Golub and G. Meurant. *Matrices, Moments and Quadrature with Applications*. Princeton Series in Applied Mathematics. Princeton University Press, Princeton, NJ, 2010.

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

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

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

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

[16] B. Iannazzo. On the Newton method for the matrix pth 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. Zietak. A Padé family of iterations for the matrix sector function and the matrix pth root. *Numerical Linear Algebra with Applications*, 16(11-12):951–970, 2009.

[19] P. G. LeFloch. *Hyperbolic systems of conservation laws.* Lectures in Mathematics ETH Zürich. Birkhäuser Verlag, Basel, 2002. The theory of classical and nonclassical shock waves.

[20] A. Lischke, G. Pang, M. Gulian, F. Song, C. Glusa, X. Zheng, Z. Mao, W. Cai, M.M. Meerschaert, M. Ainsworth, and G. Em Karniadakis. What is the fractional Laplacian? a comparative review with new results. *Journal of Computational Physics*, 404:109009, 2020.

[21] E. Lorin and S. Tian. A numerical study of fractional linear algebraic system solvers. *Submitted.*, 2020.

[22] P. Ochs, Y. Chen, T. Brox, and T. Pock. iPiano: Inertial Proximal Algorithm for Nonconvex Optimization. *SIAM Journal on Imaging Sciences*, 7(2):1388–1419, 2014.

[23] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical mathematics*, volume 37 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 2000.

[24] 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.

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

[26] J. C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations.* The Wadsworth & Brooks/Cole Mathematics Series. Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA, 1989.

[27] E. F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics.* Springer-Verlag, Berlin, second edition, 1999. A practical introduction.

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