TALLINN UNIVERSITY OF TECHNOLOGY
School of Science

Denis Akimov 186044YAFB

# Inverse problems for fractional oscillators

Bachelor's thesis

Supervisor: Jaan Janno
Professor

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Loodusteaduskond

Denis Akimov 186044YAFB

# Pöördülesanded murrulistele ostsillatoritele

Bakalaureusetöö

Juhendaja: Jaan Janno
Professor

Tallinn 2024

# Author's declaration of originality and supervisor's resolution

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Denis Akimov
Signature:

17.05.2024

This work corresponds to bachelor's thesis requirements in force.

Supervisor: Jaan Janno
Signature:

17.05.2024

# Abstract
## Inverse problems for fractional oscillators

The goal of this bachelor's thesis was to use the Laplace transform method to construct solutions for inverse problems of model estimation concerning fractional oscillators and to assess the correctness and stability of obtained solutions using numerical simulations in Python. This involves problems for linear differential equations with Caputo type operators of certain order range.

In the first chapter, important definitions and basics of fractional calculus were brought in. In the second chapter initial value problem and forcing term problem were evaluated for equations with one fractional derivative, and the general algorithm for numerical methods in Python was introduced. In the third chapter, forcing term problems were evaluated for equations with two fractional derivatives of non-equal order.

As a result of this work inverse problem solutions that estimate model parameters with good precision for theoretical data were constructed, but the stability of each solution got worse with each newly introduced unknown parameter. Under small random interference some of the solutions gave relative errors of more than 100%. From this the conclusion was drawn that without further research on improving stability of inverse problem solutions obtained using this method, only the simplest of them would be suitable for working with real measurements.

# Annotatsioon
## Pöördülesanded murrulistele ostsillatoritele

Käesoleva bakalaureusetöö eesmärgiks oli kasutada Laplace'i teisenduse meetodit, et konstrueerida lahendid murrulistele ostsillaatoritele püstitatud mudelhinnanguvate pöördülesannete jaoks ja kontrollida saadud lahendite täpsust ja stabiilsust arvutuslike simulatsioonide abil Pythonis. Tegemist on ülesannetega Caputo tüüpi operaatoreid sisaldavatele lineaarsetele diferentsiaalvõrranditele teatavas järguvahemikus.

Esimeses peatükkis toodi sisse olulised definitsioonid ja murrulise analüüsi alused. Teises peatükkis uuriti algväärtustega ülesannet ja vabaliikmega ülesannet võrranditele, mis sisaldavat ühte murrulist tuletist ja toodi sisse arvutusmeetodite põhialgoritm Pythonis. Kolmandas peatükis uuriti vabaliikmetega ülesandeid kahte erineva järguga murrulist tuletist sisaldavatele võrranditele.

Töö tulemusena saadi pöördülesannete lahendid, mis hindavad mudelite parameetreid hea täpsusega teoreetiliste andmete korral, kuid iga lisatud tundamatu parameetriga lahendi stabiilsus halvenes. Väikese juhusliku häirituse korral mõned lahendid andsid relatiivse vea, mis on suurem kui 100%. Sellest sai teha järelduse, et ilma stabiilsuse parandamiseks tehtavate lisauuringuteta sobivad ainult lihtsaimad kasutatud meetodiga koostatud pöördülesannete lahendid tööks reaalsete mõõtmistega.

# Table of contents

# 1. Introduction

Majority of questions solved in mathematical sciences are so called direct problems. We have cause $f$ and the model $L$, and our goal is acquiring the effect $x$. For a linear operator $L$, this can take a form of equation:

$$Lx = f \tag{1.1}$$

For direct problems we assume that model $L$ is well-defined and continuous, for any unique cause $f$ there is an unique effect $x$, and small errors within $f$ do not lead to big errors within $x$ (in other words, the problem is well-posed). Examples of direct problems are numerous: figuring out an object's trajectory from its starting position, velocity and mass; finding plane projections of a three dimensional geometric figure; writing a summary based on a read text.

Given the presented direct problem, two different inverse problems can be posed. Those are known as problems of causation (knowing the model $L$ and effect $x$, figuring out the cause $f$) and model identification (knowing both the cause $f$ and effect $x$, figuring out the model $L$). It is not guaranteed that an inverse problem will be well-posed. In fact, most of them are ill-posed and will show big errors with little perturbation within input data.[1] Inverse problems have several applications in various fields, including but not limited to: medical imaging, industrial processes monitoring, ozone layer tomography and financial market modelling.[2]

Another growing research interest is the fractional calculus. Apart from providing mathematically beautiful results, generalising operators from classical calculus to non-integer orders recently found many applications in different fields of research including physics[3], bioengineering[4] and economics[5].

In this thesis Laplace transform method will be used to construct solutions to inverse problems of model identification for linear fractional differential equations with Caputo derivatives of order between 1 and 2. Such equations are known as fractional oscillators. Statement of the inverse problem is as follows: knowing the solution $x(t)$ and the cause represented by the forcing term or the initial values, we must restore the linear differential operator $L$:

$$L = \left( a_1 \frac{d^{\alpha_1}}{dt^{\alpha_1}} + a_2 \frac{d^{\alpha_2}}{dt^{\alpha_2}} + ... + a_n \frac{d^{\alpha_n}}{dt^{\alpha_n}} + a_{n+1} \right) \tag{1.2}$$

by recovering the constants $a_1, a_2, ..., a_n, a_{n+1}$ and $\alpha_1, \alpha_2, ..., \alpha_n$. Then the stability of obtained inverse problem solution should be evaluated by perturbing the input data with small random interference. This will be done numerically using scripts written in Python language (code is presented in Appendix 2).

In Chapter 2 important definitions and properties used in this work will be introduced, as well as basics of fractional calculus and fractional oscillators. It should be noted that everything is presented under assumption that functions of time are causal, i.e vanishing for every $t \leq 0$. In Chapter 3 and Chapter 4 we will look at concrete examples of equations with $n = 1$ and $n = 2$ respectively.

# 2. Preliminaries

## 2.1 Laplace transform

We will start by introducing a very powerful mathematical tool that was used throughout the whole work. It is the Laplace transform, which is defined as value of the following improper integral:

$$\mathscr{L}\{f(t)\} = F(s) := \int_0^\infty f(t)e^{-st}dt, \tag{2.1}$$

for values of $s$ where the integral makes sense. Variable $s$ is the complex frequency, so time-domain function $f(t)$ is called the Laplace original and respective complex frequency domain function $F(s)$ is called the Laplace image. Functions $f$ and $F$ constitute a Laplace transform pair commonly written with the following notation:

$$f(t) \div F(s). \tag{2.2}$$

The Laplace transform existence theorem states that $\mathscr{L}\{f(t)\}(s)$ exists for all $s > \alpha$ if $f(t)$ is piecewise continuous on every finite interval in $[0, \infty)$ and satisfies $|f(t)| \leq Me^{\alpha t}$ for all $t \in [0, \infty)$. In other words, the function must be of exponential order $\alpha$. Directly from the integral definition it follows that Laplace transform is linear:

$$\mathscr{L}\{cf(t) + dg(t)\}(s) = c\mathscr{L}\{f(t)\}(s) + d\mathscr{L}\{g(t)\}(s). \tag{2.3}$$

The Laplace transform is unique, which means that for two functions $f(t)$ and $g(t)$ that produce the same Laplace transform $F(s)$, function $n(t) = f(t) - g(t)$ is a null function, i.e function for which the integral:

$$\int_0^a n(t)dt = 0, \tag{2.4}$$

for every $a > 0$. If $f(t)$ is continuously differentiable $n - 1$ times on $(0, \infty)$, then Laplace transform of an n-order derivative is given by:

$$\mathscr{L}\{f^{(n)}(t)\} = s^n F(s) - \sum_{k=1}^{n-1} s^{n-k} \lim_{t \to 0^+} \frac{d^{k-1}}{dt^{k-1}} f(t). \tag{2.5}$$

This property makes the Laplace transform very useful in solving differential equations, as they can be transformed into algebraic equations for $F(s)$.[6] On the other hand, n-order derivative in frequency domain can be obtained by applying it directly to the definition:

$$F^{(n)}(s) = \int_0^\infty f(t)(-t)^n e^{-st}dt. \tag{2.6}$$

If it is possible to analytically continue the Laplace image to the left of the imaginary axis and there exists an abscissa of convergence $\sigma_s$ for which function $F(s)$ is analytic in the half-plane

$\operatorname{Re}(s) \geq \sigma_s$, then the inverse Laplace transform can be introduced by the so-called Bromwich contour integral:

$$\mathscr{L}^{-1}\{F(s)\}(t) = f(t) = \frac{1}{2\pi i}\int_{\sigma-i\infty}^{\sigma+i\infty} e^{st}F(s)ds, \sigma \geq \sigma_s. \tag{2.7}$$

By choosing a specific contour the Bromwich integral can be expressed as the integral of a real-valued function of a real variable, which is then evaluated analytically or numerically.[7]

## 2.2 Convolution of two functions

Convolution of two functions (common notation $*$ is used) is an operation that produces a third function. It can be seen as a form of multiplication, and is used alongside addition to construct algebraic structures for function spaces. For $f(t)$ and $g(t)$ supported for $t \in [0,\infty)$, we define convolution $f * g$ as an integral:

$$(f*g)(t) := \int_0^t f(\tau)g(t-\tau)d\tau, \tag{2.8}$$

which represents overlap one function has when shifted over the other. Convolution operation is commutative, associative and distributive:

$$(f*g) = (g*f), \tag{2.9}$$

$$f*(g*h) = (f*g)*h, \tag{2.10}$$

$$f*(g+h) = f*g + f*h. \tag{2.11}$$

For functions $f(t)$ and $g(t)$ that have Laplace transforms $F(s)$ and $G(s)$ the following statement holds:

$$f*g \div F\cdot G. \tag{2.12}$$

So convolution in time domain is equal to multiplication in complex frequency domain. This is called the convolution theorem and it will prove itself useful when solving inhomogeneous differential equations further on.

## 2.3 Gamma function

Gamma function is defined for all $z \in \mathbb{C}\backslash\{-n : n \in \mathbb{N} \cup 0\}$ as a limit:

$$\Gamma(z) := \lim_{n\to\infty} \frac{n!n^z}{z(z+1)(z+2)...(z+n)}. \tag{2.13}$$

In the half-plane $\operatorname{Re}(z) > 0$ the function can be represented with the following indefinite integral:

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt. \tag{2.14}$$

With a property $\Gamma(1+z) = z\Gamma(z)$ the function is used to naturally extend the factorial operation on natural numbers to real or complex values of the argument. For $n \in \mathbb{N}$:

$$\Gamma(n+1) = n!. \tag{2.15}$$

Due to this Gamma function is sometimes called a continuous factorial.[8]

## 2.4 Mittag-Leffler functions

The class of special functions known as Mittag-Leffler functions plays a major role in fractional calculus. The throughout overview of those functions and their properties was given in [9], some of which are going to be brought in here. First of all, consider the known Taylor expansion of an exponential function:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}. \tag{2.16}$$

Factorial operation in the denominator can be replaced with the Gamma function:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{\Gamma(k+1)}. \tag{2.17}$$

Classical Mittag-Leffler function is defined as value of the power series:

$$E_\alpha(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k+1)}. \tag{2.18}$$

With $\alpha = 1$ we get back the exponential function, so Mittag-Leffler can be seen as direct generalisation of it. The series converges in the whole complex plane for $\mathrm{Re}\,(\alpha) > 0$, and diverges everywhere for $\mathrm{Re}\,(\alpha) < 0$ at $\mathbb{C} \setminus \{0\}$. For $\mathrm{Re}\,(z) = 0$, radius of convergence is equal to $e^{\frac{\pi}{2}|\mathrm{Im}\,(\alpha)|}$. For all $z \in \mathbb{C}$, following relations are valid:

$$E_1(\pm z) = e^{\pm z}, \tag{2.19}$$

$$E_2(-z^2) = \cos(z), \tag{2.20}$$

$$E_2(z^2) = \cosh(z). \tag{2.21}$$

For practical purposes Mittag-Leffler function of a real variable $t$ and a real parameter $\alpha$ is used. For the Mittag-Leffler function at $\pm t^\alpha$ the following Laplace transform pair is valid:

$$E_\alpha(\pm t^\alpha) \div \frac{s^{\alpha-1}}{s^\alpha \mp 1}. \tag{2.22}$$

A straightforward generalisation of a classical Mittag-Leffler function is obtained by introducing a second complex parameter $\beta$. Two-parametric Mittag-Leffler function is defined as a power series:

$$E_{\alpha,\beta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k+\beta)}, \mathrm{Re}\,(\alpha) > 0, \beta \in \mathbb{C}, \tag{2.23}$$

with a simple relationship with the classical ML function:

$$E_{\alpha,1}(z) = E_\alpha(z). \tag{2.24}$$

For a real variable $t$, the following Laplace transform pair is valid:

$$t^{\beta-1}E_{\alpha,\beta}(\lambda t^\alpha) \div \frac{s^{\alpha-\beta}}{s^\alpha - \lambda}. \tag{2.25}$$

Two-parametric Mittag-Leffler function can be differentiated using the following formula:

$$\frac{d^m}{dt^m}[t^{\beta-1}E_{\alpha,\beta}(t^\alpha)] = t^{\beta-m-1}E_{\alpha,\beta-m}(z^\alpha), m \geq 1. \tag{2.26}$$

A special case of the two-parametric ML function that is commonly used is so called $\alpha$-exponential, defined in the following way:

$$e^z_{\alpha,\lambda} := z^{\alpha-1}E_{\alpha,\alpha}(\lambda z^\alpha), z \in C \setminus \{0\}, \lambda \in \mathbb{C}. \tag{2.27}$$

From (2.25) Laplace transform pair for the $\alpha$-exponent can be deduced:

$$e^t_{\alpha,\lambda} \div \frac{1}{s^\alpha - \lambda}. \tag{2.28}$$

It should be noted that the main property of a regular exponential function generally does not hold for $\alpha$-exponential functions:

$$e^a_{\alpha,\lambda}e^b_{\alpha,\lambda} \neq e^{a+b}_{\alpha,\lambda}. \tag{2.29}$$

Further generalisation is the Prabhakar function (also known as third parametric Mittag-Leffler function), which is defined as value of the following power series:

$$E^\gamma_{\alpha,\beta} = \sum_{k=0}^{\infty} \frac{(\gamma)_k}{k!\Gamma(\alpha k + \beta)} z^k, \operatorname{Re}(\alpha) > 0, \operatorname{Re}(\beta) > 0, \gamma > 0, \tag{2.30}$$

where $(\gamma)_k$ is the Pochhammer's symbol, defined as:

$$(\gamma)_k = \gamma(\gamma+1)(\gamma+2)...(\gamma+k-1) = \frac{\Gamma(\gamma+k)}{\Gamma(\gamma)}. \tag{2.31}$$

For $\gamma = 1$ we get the two-parametric ML function:

$$E^1_{\alpha,\beta} = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + \beta)}. \tag{2.32}$$

For $\gamma = \beta = 1$ we get the classical ML function:

$$E^1_{\alpha,1} = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + 1)}. \tag{2.33}$$

Following Laplace transform pair is valid for the Prabhakar function:

$$t^{\beta-1}E^{\gamma}_{\alpha,\beta}(\lambda t^{\alpha}) \div \frac{s^{-\beta}}{(1-\lambda s^{-\alpha})^{\gamma}}. \tag{2.34}$$

## 2.5 Fractional calculus

Fractional calculus deals with generalisations of classical operations like differentiation and integration to fractional orders. Information in this section was primarily taken from [10] and [11]. More physical approach can be found in [12]. We will begin by defining classical integer-order operators. By ${}_xD$, we denote the derivative operator with respect to $x$:

$${}_xDf(x) := \frac{df(x)}{dx}. \tag{2.35}$$

By ${}_xI_\alpha$ we denote the integral operator with base-point $a$ on function that is Riemann-integrable on $[a,b]$:

$${}_xI_\alpha f(x) := \int_a^x f(t)dt, a \leq x \leq b. \tag{2.36}$$

Assuming $a = 0$ we drop the subprefix:

$${}_xIf(x) = \int_0^x f(t)dt. \tag{2.37}$$

In some suitable function space operator $D$ can be seen as the left inverse of $I_a$:

$$DI_af = f. \tag{2.38}$$

For $n \in \mathbb{N}$ we denote n-fold iterates of $D$ and $I_\alpha$ as $D^n$ and $I^n_\alpha$:

$${}_xD^nf(x) = \frac{d^n}{dx^n}f(x), \tag{2.39}$$

$${}_xI^n_af(x) = \int_a^x \int_a^{t_1} ... \int_a^{t_{n-1}} f(t_n)dt_n...dt_2dt_1. \tag{2.40}$$

Using Cauchy's formula for repeated integrations we can compute ${}_xI^n_a$ as:

$${}_xI^n_af(x) = \frac{1}{(n-1)!}\int_a^x (x-t)^{n-1}f(t)dt. \tag{2.41}$$

Inversion property still holds:

$$D^nI^n_af = f. \tag{2.42}$$

Semigroup properties apply to both operators:

$$D^mD^nf = D^nD^mfD^{n+m}f, \tag{2.43}$$

$$I^m_aI^n_af = I^n_aI^m_af = I^{n+m}_af. \tag{2.44}$$

Notably, with $m > n$ and n-times differentiable function on $[a, b]$ the following equation holds:

$$D^n f = D^m I_a^{m-n} f. \tag{2.45}$$

Now we can start looking at generalisations of operators 2.39 and 2.40 to fractional order $\alpha$. Riemann-Liouville fractional integral of order $\alpha$ at base point $a$ is defined as:

$$_tI_a^\alpha f(t) := \frac{1}{\Gamma(\alpha)} \int_a^t (t-\tau)^{\alpha-1} f(\tau) d\tau. \tag{2.46}$$

This integral exists if $f(t)$ is the locally integrable function and for $t \to 0$ behaves like $O(t^{-\nu})$ with $\nu < \alpha$. The definition can be directly obtained from (2.41) by replacing the factorial operation with the Gamma function. Semigroup properties of a classical operator are conserved:

$$I_a^\alpha I_a^\beta = I_a^\beta I_a^\alpha = I_a^{\alpha+\beta}. \tag{2.47}$$

To find Laplace transform of a Riemann-Liouville fractional integral with $a = 0$, we must consider that it is in fact a convolution integral:

$$_tI^\alpha f(t) = \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(\tau) d\tau = \frac{t^{\alpha-1}}{\Gamma(\alpha)} * f(t). \tag{2.48}$$

Using the convolution theorem we can obtain:

$$\mathscr{L}\{\frac{t^{\alpha-1}}{\Gamma(\alpha)} * f(t)\}(s) = \frac{1}{\Gamma(\alpha)} \mathscr{L}\{t^{\alpha-1}\}(s) \cdot F(s) = \frac{\Gamma(\alpha)}{\Gamma(\alpha)} \cdot \frac{F(s)}{s^\alpha} = \frac{F(s)}{s^\alpha}. \tag{2.49}$$

Returning to the property (2.45):

$$D^n f = D^m I_a^{m-n} f,$$

we replace the classical integral operator with Riemann-Liouville fractional integral, taking $n = \alpha$ and $m = \lceil \alpha \rceil$ (where $\lceil \cdot \rceil$ denotes greatest closest integer, also known as the ceiling function). Riemann-Liouville fractional derivative for $\alpha > 0$ is defined as:

$$_tD_a^\alpha f(t) := {_tD^m} \, {_tI_a^{m-\alpha}} f(t), \tag{2.50}$$

or by expanding the definition:

$$_tD_a^\alpha f(t) := \frac{1}{\Gamma(m-\alpha)} \frac{d^m}{dt^m} \int_a^t (t-\tau)^{m-\alpha-1} f(\tau) d\tau. \tag{2.51}$$

Riemann-Liouville derivative is a left-inverse of Riemann-Liouville integral:

$$D_a^\alpha L_a^\alpha f = f. \tag{2.52}$$

Unlike the Riemann-Liouville fractional integral, in general the semigroup property does not hold for the fractional derivatives. Equation:

$$D_a^\alpha D_a^\beta f = D_a^{\alpha+\beta} f \tag{2.53}$$

holds only under assumptions: $\alpha, \beta \geq 0$ and $f = I_a^{\alpha+\beta}\phi$, where $\phi$ is integrable on $[a, b]$.

Formula (2.51) shows an important difference between classical and fractional derivative operators. Classical operator is local, which means that calculating ${}_tD^n f(t)$ requires to know $f$ in a small neighbourhood of $t$. Indeed, we do not need to know the whole function to find the slope of a tangent. Fractional derivative on the other hand is not local. Calculating ${}_tD_a^\alpha f(t)$ requires integration, and therefore knowing $f$ on the entire interval $[a, t]$. This property makes fractional derivatives useful for modelling phenomena with memory effects, meaning the present is affected by each point in the past. But Riemann-Liouville's definition has clear disadvantages when working with physical models. First of all, consider the Riemann-Liouville fractional derivative of a power function:

$${}_tD^\alpha t^\nu = \frac{\Gamma(1+\nu)}{\Gamma(1+\nu-\alpha)} t^{\nu-\alpha}, \tag{2.54}$$

which is actually a direct generalisation of a n-th derivative formula (it should be noted that similar generalisations for other classical derivative formulas may not correspond to the Riemann-Liouville definition). From here, Riemann-Liouville derivative for a constant function $f(t) = 1$ can be calculated:

$${}_tD^\alpha 1 = {}_tD^\alpha t^0 = \frac{t^{-\alpha}}{\Gamma(1-\alpha)}. \tag{2.55}$$

This is the first unfortunate result. Riemann-Liouville derivative with fractional order $\alpha$ of a constant function is not equal to zero. It can be seen that at integer orders, the derivative vanishes due to reciprocal Gamma function having zeroes at 0 and negative integers. The Laplace transform of a RL derivative can be found by applying it directly to the definition:

$$\mathscr{L}\{{}_tD^n\, {}_tI^{m-\alpha} f(t)\}(s) = s^\alpha F(s) - \sum_{k=1}^{m} s^{m-k} \lim_{t\to 0^+} {}_tD^{k-1}\, {}_tI^{m-\alpha} f(t). \tag{2.56}$$

Here it can be seen that initial values for differential equations with Riemann-Lioville fractional operators must be provided as derivatives of a function ${}_tI^{m-\alpha} f(t)$, which are cumbersome to use. While there exist attempts to find physical meaning in those values, currently there is no way to obtain them by direct measurements. This and the result for constant functions are notable disadvantages that RL fractional derivatives have when modelling real-world phenomena with differential equations. A slightly different approach to fractional derivatives is obtained by inverting the order of operators in definition (2.50). Caputo fractional derivative for $\alpha > 0$ is defined as:

$${}_t^C D_a^\alpha f(t) := {}_tI_a^{m-\alpha}\, {}_tD^m f(t), \tag{2.57}$$

with a prefix $C$ added not to confuse the new operator with the Riemann-Liouville definition.

With the fractional integral acting on the derivative, we now require that $f$ is $m$-times differentiable, and its $m$-order derivative is integrable on $[a, b]$. This is much more strict, but it already comes with an advantage - Caputo derivative of a constant function is always equal to zero:

$$^{C}D_a^{\alpha}C = I_a^{m-\alpha}D^m C = I^{n-\alpha}0 = 0. \tag{2.58}$$

Expanding the definition of Caputo derivative leads to:

$$^{C}_{t}D_a^{\alpha}f(t) := \frac{1}{\Gamma(m-\alpha)}\int_a^t (t-\tau)^{m-\alpha-1}\frac{d^m}{d\tau^m}f(\tau)d\tau, \tag{2.59}$$

so the operator still has the non-local properties. Relationship between Caputo and Riemann-Lioville definitions is given by:

$$^{C}_{t}D_a^{\alpha}f(t) = {}_tD_a^{\alpha}\left(f(t) - \sum_{k=0}^{m-1} f^k(a)\frac{(t-a)^k}{k!}\right). \tag{2.60}$$

From here it can be seen that if the function $f(t)$ and first $m-1$ integer order derivatives are 0 at $t = a$, Caputo and Riemann-Lioville definitions become one of the same. Laplace transform of a Caputo derivative is given by:

$$\mathscr{L}\{{}_tI^{m-\alpha}\,{}^{C}_{t}D^m f(t)\}(s) = s^{\alpha}F(s) - \sum_{k=0}^{m} s^{\alpha-k}\lim_{t\to 0^+}\frac{d^{k-1}}{dt^{k-1}}f(t). \tag{2.61}$$

Initial values for equations with Caputo derivative are given by $n-1$ integer order derivatives of $f(t)$. For those needed data usually has clear meaning and can be provided with suitable measurements. For example, if $x(t)$ is displacement in time, then $\dot{x}$ and $\ddot{x}$ correspond to velocity and acceleration respectively. At the same time Caputo operator maintains the useful non-local properties. For this reason Caputo approach is preferred when describing physical phenomena with fractional differential equations. Now we can move on to defining fractional oscillators.

## 2.6 Fractional oscillators

Classical harmonic oscillator is a model described by the following differential equation:

$$m_t D^2 x + kx = 0. \tag{2.62}$$

Here $m$ represents the mass and $k$ represents the spring strength. We assume $m > 0$ and $k > 0$ so the equation makes physical sense. The solution to the equation is a sine wave:

$$x(t) = A\cos(\omega t + \phi), \tag{2.63}$$

where $\omega = \sqrt{\frac{k}{m}}$. If the friction force is present, a term with first order derivative proportional to a viscous damping coefficient $\gamma > 0$ is added, modifying the equation:

$$m_t D^2 x + \gamma_t D x + kx = 0. \tag{2.64}$$

If $4km > \gamma^2$, the solution is an exponentially damped sine wave (Fig 2.1):

$$x(t) = \frac{A}{2\sqrt{4km - \gamma^2}} e^{-t\frac{\gamma}{2m}} \cos(\omega t + \phi), \tag{2.65}$$

where $\omega = \frac{\sqrt{4km-\gamma^2}}{2m}$. This is one of the most fundamental results in classical physics.[12]

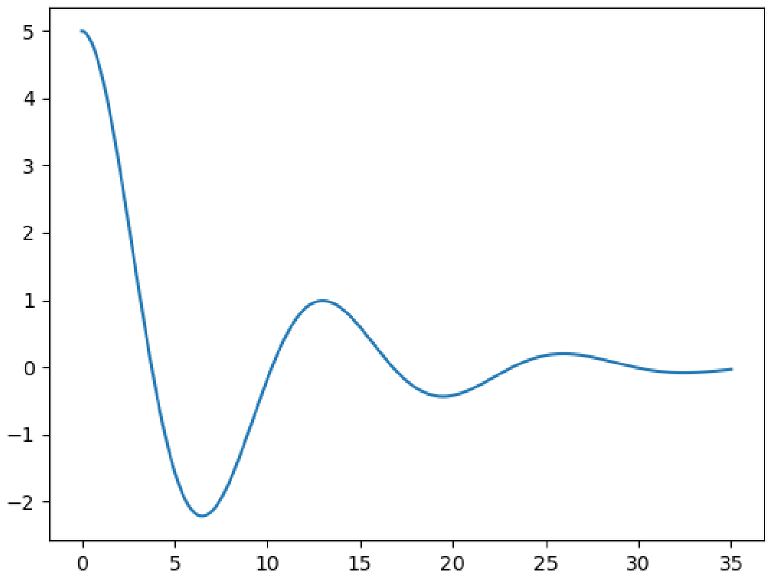


Figure 2.1. Exponentially decaying oscillation.

Now we generalise equation (2.62) by replacing the second order derivative with $2\alpha$ order Caputo type operator, where $\frac{1}{2} < \alpha \leq 1$. We get a fractional oscillator equation:

$$m\,{}^C_t D^{2\alpha} x + kx = 0. \tag{2.66}$$

With the set conditions operator ${}^C D^{2\alpha}$ becomes:

$${}^C_t D^{2\alpha} x(t) := \frac{1}{\Gamma(2-\alpha)} \int_0^t (t-\tau)^{1-\alpha} x''(\tau) d\tau. \tag{2.67}$$

This is not entirely correct. Physically we want every term in the equation to have the same units (in this case units of force). Operator ${}_t D^n$ has dimension $T^{-n}$, so it would be natural to assume that operator ${}^C_t D^{2\alpha}$ has dimension of $T^{-2\alpha}$. In order to correct that problem a new parameter $\sigma$ with dimension $T$ is introduced, defined as:

$$\dim\left[\frac{1}{\sigma^{1-\alpha}}\,{}_tD^{\alpha}\right] = T^{-1}. \tag{2.68}$$

Parameter $\sigma$ is called fractional time component and it can be related to fractional order $\alpha$ in a way that is specific for each model.[13] The dimensionally correct fractional oscillator equation would be:

$$\frac{m}{\sigma^{2(1-\alpha)}}\,{}^{C}_{t}D^{2\alpha}x + kx = 0. \tag{2.69}$$

We will write the first parameter with dimension $MT^{2\alpha-2}$ as $\mu$:

$$\mu = \frac{m}{\sigma^{2(1-\alpha)}}. \tag{2.70}$$

Finally the equation takes form:

$$\mu\,{}^{C}_{t}D^{2\alpha}x + kx = 0. \tag{2.71}$$

Solving the equation leads to two linearly independent solutions:

$$x_1(t) = E_{2\alpha}(-\omega t^{2\alpha}), \tag{2.72}$$

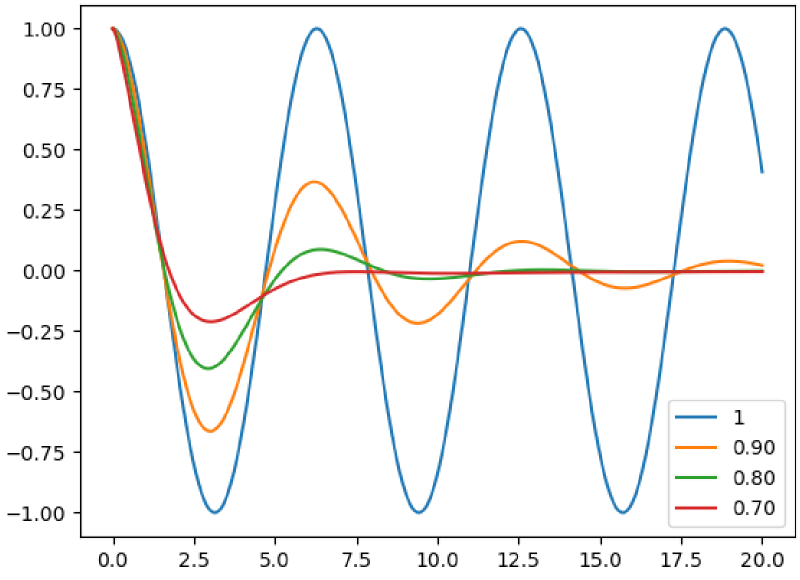
$$x_2(t) = t^{\alpha}E_{2\alpha,\alpha+1}(-\omega t^{2\alpha}), \tag{2.73}$$

where $\omega = \frac{k}{\mu}$. As it can be seen on Fig. 2.2, the oscillation experiences damping despite not being affected by any external friction. Furthermore, for $t \to \infty$ and $\alpha \in (0, 1)$:

$$E_{2\alpha}(-\omega t^{2\alpha}) = \frac{x^{-2\alpha}}{\omega\Gamma(1-2\alpha)}(1 + o(1)), \tag{2.74}$$

$$t^{\alpha}E_{2\alpha,\alpha+1}(-\omega t^{2\alpha}) = \frac{x^{1-2\alpha}}{\omega\Gamma(2-2\alpha)}(1 + o(1)). \tag{2.75}$$

The solution dampens algebraically, in other words much slower than the exponential decay seen in the solution for classical oscillator with friction.[10] Physical interpretation of a fractional oscillator is presented in [14] as the average of an ensemble of ordinary harmonic oscillators. Those oscillators have approximately the same frequency, so each response is compensated by an antiphase response of another oscillator. From this follows the intrinsic absorption property. Inverse problems for models of type (2.71) are evaluated in Chapter 3.



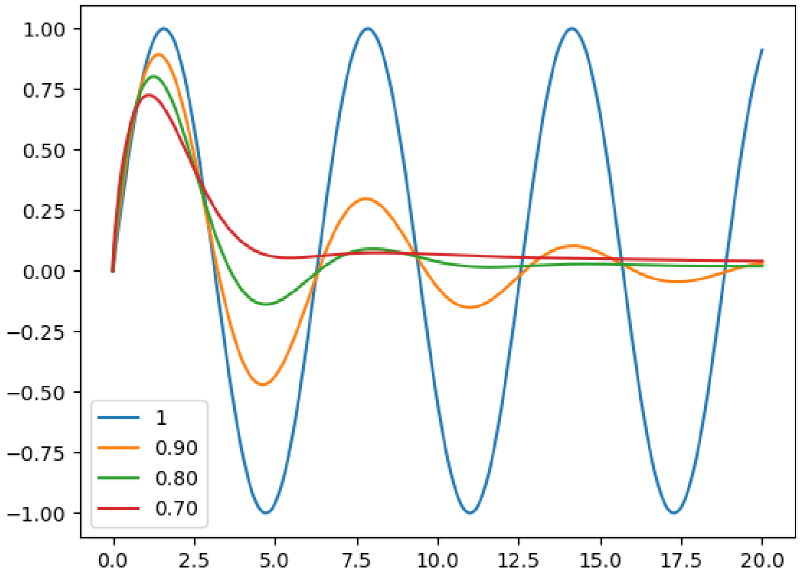


Figure 2.2. Graphs of $E_{2\alpha}(-t^{2\alpha})$ and $t^{\alpha}E_{2\alpha,\alpha+1}(-t^{2\alpha})$ at different $\alpha$ values. At $\alpha = 1$, $\cos(t)$ and $\sin(t)$ are recovered. Note the number of zeroes which tends to $\infty$ as $\alpha \to 1$.

A more general equation can be obtained by adding a new term with Caputo operator of order $2\beta$, where $0 < \beta \leq 1$ and $\beta < \alpha$:

$$\mu\, {}^C_tD^{2\alpha}x + \gamma\, {}^C_tD^{2\beta}x + kx = 0. \tag{2.76}$$

For the equation to make physical sense parameter $\gamma$ here would need to have dimension $MT^{2\beta-2}$. An example of an equation of that type is Bagley-Torvik equation which found applications in theory of viscoelasticity:

$$mD^2x + 2S\sqrt{\mu\rho}D^{\frac{3}{2}}x + kx = 0, \tag{2.77}$$

which describes movement of a plate of mass $m$ and surface area $S$ as it is attached to a spring with strength $k$ and submerged into a liquid with viscocity $\mu$ and density $\rho$.[15] Bagley-Torvik equation represents a specific case in which $\alpha = 1$ and $\beta = \frac{3}{4}$. Inverse problems for models of type (2.76) are evaluated in Chapter 4.

# 3. Inverse problems for fractional harmonic oscillators with one Caputo type operator

## 3.1 Initial value problem

In this section we will evaluate the following model:

$$ {}^{C}_{t}D^{2\alpha}x + \omega x = 0; x(0^{+}) = x_0; x'(0^{+}) = v_0, \tag{3.1}$$

where $\omega = \frac{k}{\mu}$.

### 3.1.1 Direct problem solution

To solve the direct problem for equation (3.1) we will apply the Laplace transform using property (2.61). In s-space the equation changes:

$$s^{2\alpha}X(s) - s^{2\alpha-1}x_0 - s^{2\alpha-2}v_0 + \omega X(s) = 0. \tag{3.2}$$

Solving for $X(s)$ we have:

$$X(s) = \frac{x_0 s^{2\alpha-1}}{s^{2\alpha}+\omega} + \frac{v_0 s^{2\alpha-2}}{s^{2\alpha}+\omega}. \tag{3.3}$$

And now using (2.25) we can take the inverse Laplace transform and get the desired solution:

$$x(t) = x_0 E_{2\alpha}(-\omega t^{2\alpha}) + v_0 t E_{2\alpha,2}(-\omega t^{2\alpha}). \tag{3.4}$$

Now let us move on to solving an inverse problem.

### 3.1.2 Inverse problem solution

For an inverse problem we measured the function $x(t)$ itself and we also know the initial values $x_0$ and $v_0$ as input. Our goal is to identify the model, in this case find values of parameters $\omega$ and $\alpha$. With $x(t)$ presented as a time series, measured at $t = [0, M]$, and assuming $x(t)e^{-st}$ approaches zero with $t \to M$ and $s > 0$ fast enough, we can evaluate the Laplace transform directly from the definition (2.1) using quadrature formulas:

$$X(s) \approx \int_0^M x(t)e^{-st}dt. \tag{3.5}$$

Composite Simpson's rule defined in SciPy module for Python[16] was used in this work. Additionally, s-space derivatives of $X$ can be evaluated using the relation:

$$X^n(s) \approx \int_0^M (-1)^n t^n x(t) e^{-st} dt. \tag{3.6}$$

With this, we can use (3.3) to construct a system of equations for $\omega$ and $\alpha$. The best option is to use $X(1)$ and $X'(1)$, as derivative takes $\alpha$ away from exponential and then $s = 1$ gets rid of it. First of all, we must find derivative of $X(s)$ analytically:

$$\frac{d}{ds}\left[\frac{x_0 s^{2\alpha-1}}{s^{2\alpha}+\omega} + \frac{v_0 s^{2\alpha-2}}{s^{2\alpha}+\omega}\right] = \frac{x_0 s^{2\alpha-2}(-2\alpha\omega + s^{2\alpha} + \omega)}{(s^{2\alpha}+\omega)^2} + \frac{v_0 s^{2\alpha-3}(-\alpha\omega + s^{2\alpha} + \omega)}{(s^{2\alpha}+\omega)^2}. \tag{3.7}$$

With $s = 1$:

$$X(1) = \frac{x_0 + v_0}{1+\omega}, \tag{3.8}$$

$$X'(1) = \frac{-\alpha(2x_0\omega + v_0\omega) + (x_0 + v0)(1+\omega)}{(1+\omega)^2}. \tag{3.9}$$

Finally, we get the required parameters:

$$\omega = \frac{x_0}{v_0} X(1), \tag{3.10}$$

$$\alpha = \frac{1}{2(x_0+v_0)(1-p)}\left(\frac{X'(1)}{p} + x_0 + 2v_0\right), \tag{3.11}$$

where $p = \frac{1}{1+\omega^2}$.

### 3.1.3 Numerical results

Now that we found the inverse problem solution, we must check if it is in fact valid and how stable it is. For this an algorithm was used: input parameters $\omega$ and $\alpha$ and solve the direct problem numerically at very high precision (500000 sample points), reduce the number of points to 5000 for simulating more realistic measurement rate and apply the inverse problem to the acquired time-series $x(t)$. Calculated parameters $\omega^*$ and $\alpha^*$ should be, in theory, approximately close to the input parameters for the direct problem. Now, realistically there are always measurement errors within our data. To simulate this we take small $\delta > 0$ and perturb $x(t)$ with random normally distributed error:

$$x_\delta(t) = x(t) + \delta E, E \sim N(0,1). \tag{3.12}$$

Then the inverse problem is applied to the perturbed data and parameters $\omega^*$ and $\alpha^*$ are obtained again. This is done 100 times, and then the average relative error to the direct problem input parameters is calculated:

$$\overline{\delta_\omega} = \overline{\frac{|\omega - \omega^*|}{\omega}}, \overline{\delta_\alpha} = \overline{\frac{|\alpha - \alpha^*|}{\alpha}}. \tag{3.13}$$

For solving the direct problem Python module numfracpy was used, which allows calculating Mittag-Leffler functions of one, two and three parameters [17]. Results for 3 different models are presented below ($\delta$ set to $10^{-3}$):

Model 1:

$$D^2x + 3x = 0; x(0^+) = 2; x'(0^+) = 0. \tag{3.14}$$

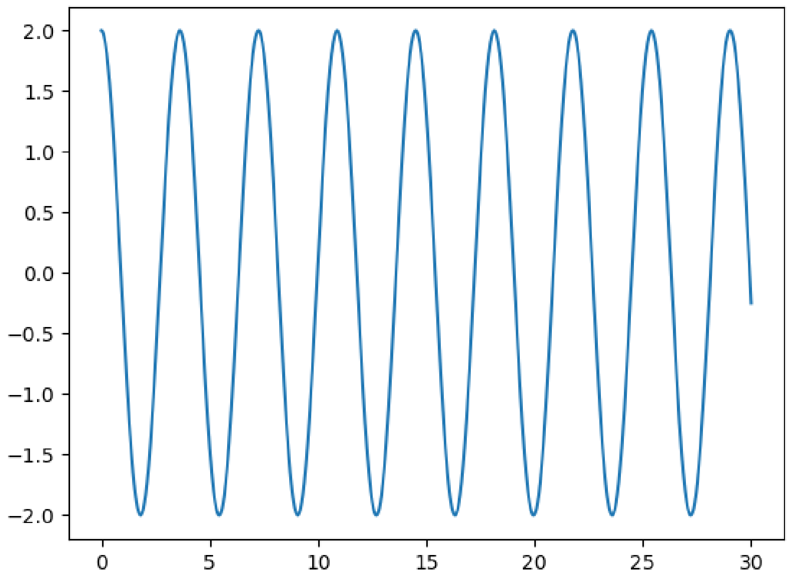


Figure 3.1. Direct problem solution to 3.14

$$\alpha^* = 0.9999999280473856$$

$$\omega^* = 3.0000005761273796$$

$$\overline{\frac{|\alpha - \alpha^*|}{\alpha}} = 0.009\%$$

$$\overline{\frac{|\omega - \omega^*|}{\omega}} = 0.08\%$$

Model 2:

$${}^{C}D^{2\cdot 0.75}x + 2.5x = 0; x(0^+) = 1; x'(0^+) = 1. \tag{3.15}$$

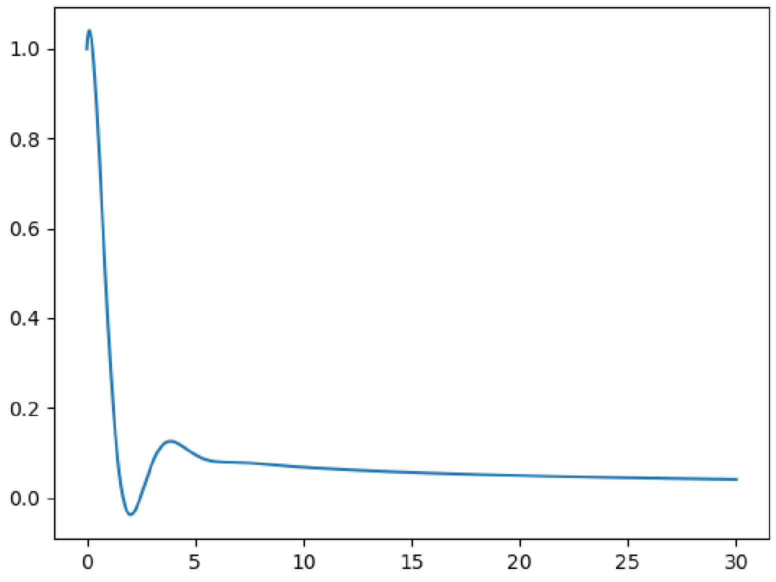


Figure 3.2. Direct problem solution to 3.15

$$\alpha^* = 0.749995279295238$$

$$\omega^* = 2.500026168062272$$

$$\frac{|\alpha - \alpha^*|}{\alpha} = 0.008\%$$

$$\frac{|\omega - \omega^*|}{\omega} = 0.08\%$$

Model 3:

$$^{C}D^{2\cdot 0.95}x + x = 0; x(0^+) = 3; x'(0^+) = 2. \tag{3.16}$$

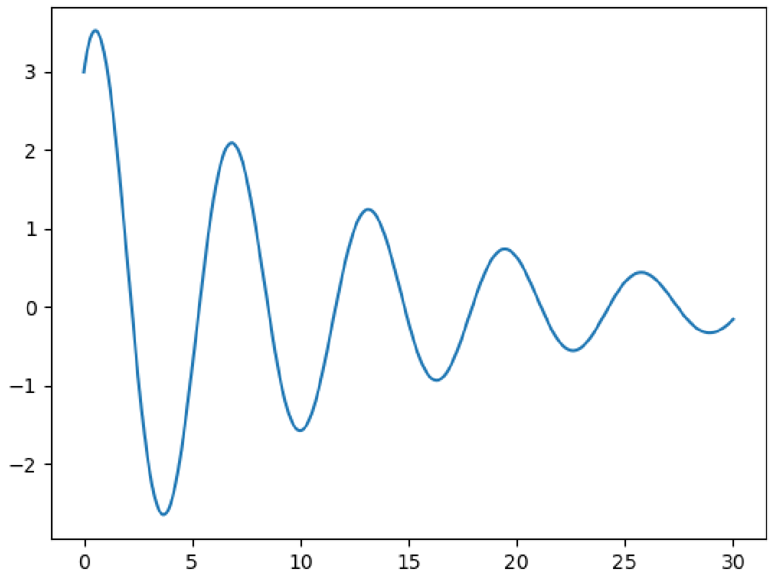


Figure 3.3. Direct problem solution to 3.16

$$\alpha^* = 0.949999798982163$$

$$\omega^* = 1.000000245078731$$

$$\overline{\frac{|\alpha - \alpha^*|}{\alpha}} = 0.005\%$$

$$\overline{\frac{|\omega - \omega^*|}{\omega}} = 0.009\%$$

The results seem quite promising, we got good precision and little mean relative errors for the parameters. It should be noted that for models 3.14 and 3.15 mean error for $\omega$ is one order bigger than that of $\alpha$. This is a behaviour that will be more noticeable further on. For now we will move to evaluating a forcing term problem for a similar equation.

## 3.2 Forcing term problem

In this section we will evaluate the model:

$$\mu D^{2\alpha} x + kx = f(t); x(0^+) = 0; x'(0^+) = 0. \tag{3.17}$$

Recalling (2.60), with initial values at zero Caputo and Riemann-Lioville definitions become equal, so we can drop the prefix. We assume that forcing term function $f(t)$ is piecewise continuous and is of exponential order.

### 3.2.1 Direct problem solution

As before, to solve the direct problem for (3.17) we apply the Laplace transform to the equation. In s-space the equation takes form:

$$\mu s^{2\alpha} X(s) + kX(s) = F(s), \tag{3.18}$$

and solving for $X(s)$ yields:

$$X(s) = \frac{F(s)}{\mu s^{2\alpha} + k}. \tag{3.19}$$

The solution of out equation in s-space can be seen a product of two functions:

$$X(s) = \left(\frac{1}{\mu s^{2\alpha} + k}\right) \cdot F(s). \tag{3.20}$$

We will now take the first multiple and try to find the inverse transform for it. Let us call that function $G(s)$ and do a little transformation:

$$G(s) = \frac{1}{\mu s^{2\alpha} + k} = \frac{1}{\mu}\frac{1}{s^{2\alpha} + \omega}. \tag{3.21}$$

Comparing the result with (2.28) leads to inverse Laplace transform of function G, given by an $\alpha$-exponential function:

$$g(t) = \frac{1}{\mu} e^{t}_{2\alpha,-\omega}. \tag{3.22}$$

Now using the convolution theorem (2.12) we obtain the full solution to the equation:

$$x(t) = \frac{1}{\mu}\int_0^t e^{t-\tau}_{2\alpha,-\omega} f(\tau) d\tau. \tag{3.23}$$

Directly evaluating the convolution integral numerically can be cumbersome. To obtain the results a convolution function defined in SciPy's signal processing submodule was used, which convolves two discrete sets of data using fast Fourier transforms.[18]

### 3.2.2 Inverse problem solution

For the inverse problem we have measured the function $x(t)$, and we have the function $f(t)$ as input. Our goal is to determine parameters $\mu$, $k$ and $\alpha$. To do this we come back to (3.19). Denoting the function in the denominator as $Q(s)$, we can write that as:

$$X(s) = \frac{F(s)}{Q(s)}, \tag{3.24}$$

and solving for $Q$:

$$Q(s) = \frac{F(s)}{X(s)}. \tag{3.25}$$

We can calculate Laplace transform $F(s)$ and $X(s)$ and s-space derivatives of those function using the same methods as described in the previous section. For derivatives of $Q(s)$ we require a rule for n-th derivative for quotient of two functions, similarly to how Leibniz's rule provides derivatives for product of two functions. An iterative formula for repeated differentiation of quotients was evaluated in [19], in our case taking the form:

$$Q^{(n)}(s) = \frac{1}{X(s)}\left(F^{(n)}(s) - n!\sum_{j=1}^{n}\frac{X^{(n+1-j)}(s)}{(n+1-j)!}\frac{Q^{(j-1)}(s)}{(j-1)!}\right). \tag{3.26}$$

On the other hand:

$$Q(s) = \mu s^{2\alpha} + k. \tag{3.27}$$

We take a derivative from both sides and do some more transformations:

$$Q'(s) = (2\alpha)\mu s^{2\alpha-1} | \cdot s^{1-2\alpha},$$

$$Q'(s)s^{1-2\alpha} = 2\alpha\mu | ()',$$

$$\left(Q'(s)s^{1-2\alpha}\right)' = 0.$$

Expanding this leads to:

$$Q''(s)s + Q'(s)(1-2\alpha) = 0.$$

Solving for $\alpha$ we get:

$$\alpha = \frac{1}{2} + \frac{sQ''(s)}{2Q'(s)}. \tag{3.28}$$

Having obtained $\alpha$, calculating parameters $\mu$ and $k$ is a matter of constructing a linear system of equations using (3.27). Infinite number of equation can be obtained. With:

$$Q^{(i)}(j) = A_{i,j}\mu + B_i k, \tag{3.29}$$

where:

$$B_i = \begin{cases} 1, & i = 0 \\ 0, & i > 0 \end{cases}, \tag{3.30}$$

$$A_{i,j} = j^{2\alpha-i}\frac{\Gamma(2\alpha+1)}{\Gamma(2\alpha-i+1)}. \tag{3.31}$$

The system we get is:

$$\begin{bmatrix} A_{i_1,j_1} & B_{i_1} \\ A_{i_2,j_2} & B_{i_2} \end{bmatrix}\begin{bmatrix} \mu \\ k \end{bmatrix} = \begin{bmatrix} Q^{(i_1)}(j_1) \\ Q^{(i_2)}(j_2) \end{bmatrix} \tag{3.32}$$

. With the equation matrix:

$$A = \begin{bmatrix} A_{i_1,j_1} & B_{i_1} \\ A_{i_2,j_2} & B_{i_2} \end{bmatrix}. \tag{3.33}$$

Ideally we want to pick a matrix $A$ with the smallest condition number $\kappa(A)$. Condition number can be calculated as:

$$\kappa(A) = ||A^{-1}|| \cdot ||A|| \geq 1, \tag{3.34}$$

where $||\cdot||$ denotes the matrix norm. The further away it is from 1, the more ill-conditioned the system is. We will use $L^2$ norm, for which the condition number is given by:

$$\kappa(A) = \frac{\sigma_{max}}{\sigma_{min}}. \tag{3.35}$$

Here $\sigma_{max}$ and $\sigma_{min}$ are largest and smallest singular values of $A$ respectively.[20] We can calculate the condition number with 2-norm using a function defined in NumPy's linalg module.[21] A necessary requirement is $i = 0$ for the first equation, since only with this condition parameter $k$ doesn't disappear. Calculated condition numbers for different equations with $\alpha = 0.75$ are provided in the table below:

| (i,j) | (0,1) | (0,2) | (0,3) | (1,1) | (1,2) | (1,3) | (2,1) | (2,2) | (2,3) |
|---|---|---|---|---|---|---|---|---|---|
| (0,1) | $\infty$ | 5.85 | 7.01 | 2.42 | 2.69 | 3.04 | 3.09 | 4.05 | 4.85 |
| (0,2) | 5.85 | $\infty$ | 15.56 | 7.36 | 6.20 | 5.89 | 12.67 | 17.44 | 21.17 |
| (0,3) | 7.01 | 15.56 | $\infty$ | 20.11 | 15.26 | 13.30 | 38.06 | 53.31 | 65.08 |

Table 1. Condition numbers for different pairs of equations at $\alpha = 0.75$.

The best option in this case is to pick $(0, 1)$ and $(1, 1)$. Plotting the condition number vs $\alpha$ value reveals it doesn't change significantly at $[0.5, 1]$ (see Fig 3.4)

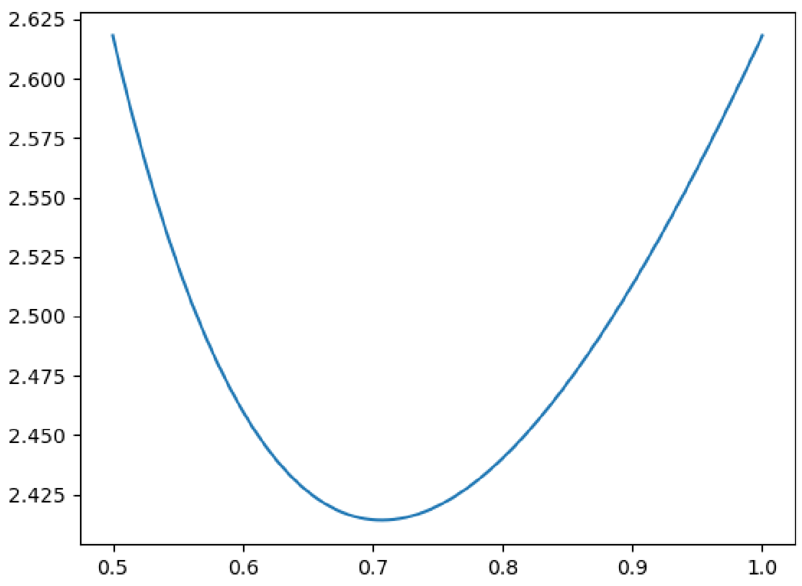


Figure 3.4. Condition number for system of equations with $(0, 1)$ and $(1, 1)$ with $\alpha \in [0.5, 1]$

With this we get a system:

$$\begin{cases} \mu + k = Q(1) \\ \mu(2\alpha) = Q'(1) \end{cases}. \tag{3.36}$$

with simple solution:

$$\mu = \frac{Q'(1)}{2\alpha - 1};\ k = Q(1) - \mu. \tag{3.37}$$

So the inverse problem solution takes the final form:

$$\begin{cases} \alpha = \frac{1}{2} + \frac{Q''(1)}{2Q'(1)} \\ \mu = \frac{Q'(1)}{2\alpha} \\ k = Q(1) - \mu \end{cases}. \tag{3.38}$$

### 3.2.3 Numerical results

Similarly to the model with initial value problems, we will now check the correctness and stability of our inverse problem solution on 3 concrete models. $\delta$ once again is $10^{-3}$.

Model 1:

$$3D^{2\cdot 0.90}x + 4x = e^{-2t}. \tag{3.39}$$

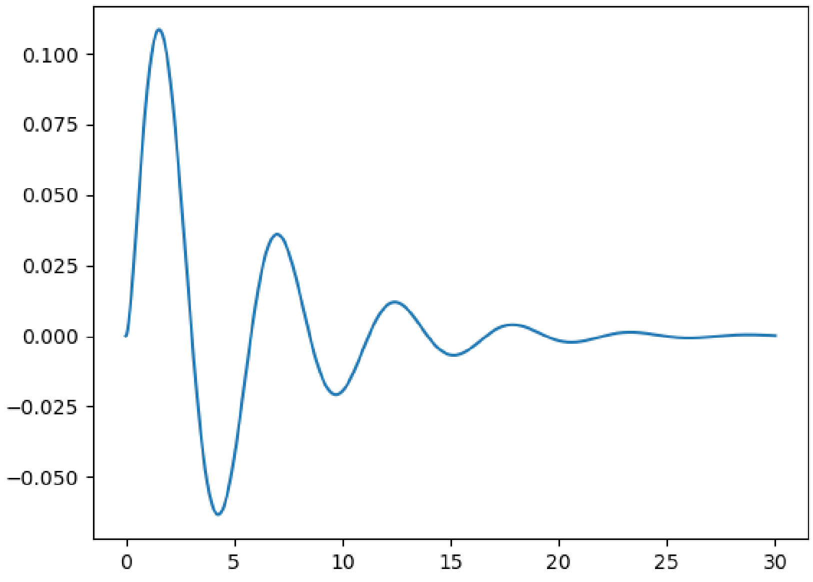


Figure 3.5. Direct problem solution to 3.39

$$\alpha^* = 0.899983162730104$$

$$\mu^* = 2.999662924024965$$

$$k^* = 3.9997095092241297$$

$$\overline{\frac{|\alpha-\alpha^*|}{\alpha}} = 0.2\%$$

$$\overline{\frac{|\mu-\mu^*|}{\mu}} = 0.7\%$$

$$\overline{\frac{|k-k^*|}{k}} = 0.9\%$$

Model 2:

$$D^{2 \cdot 0.84}x + x = t(t-30). \tag{3.40}$$

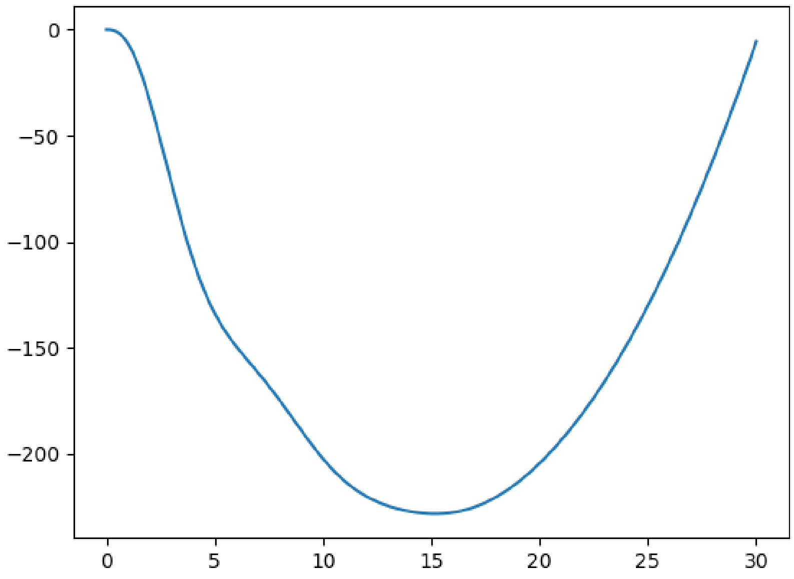


Figure 3.6. Direct problem solution to 3.40

$$\alpha^* = 0.8400000542036778$$

$$\mu^* = 1.000002050682682$$

$$k^* = 1.0000020418830877$$

$$\overline{\frac{|\alpha-\alpha^*|}{\alpha}} = 0.02\%$$

$$\overline{\frac{|\mu-\mu^*|}{\mu}} = 0.008\%$$

$$\overline{\frac{|k-k^*|}{k}} = 0.02\%$$

Model 3:

$$4D^{2\cdot 0.63}x + 5x = cos(t). \tag{3.41}$$

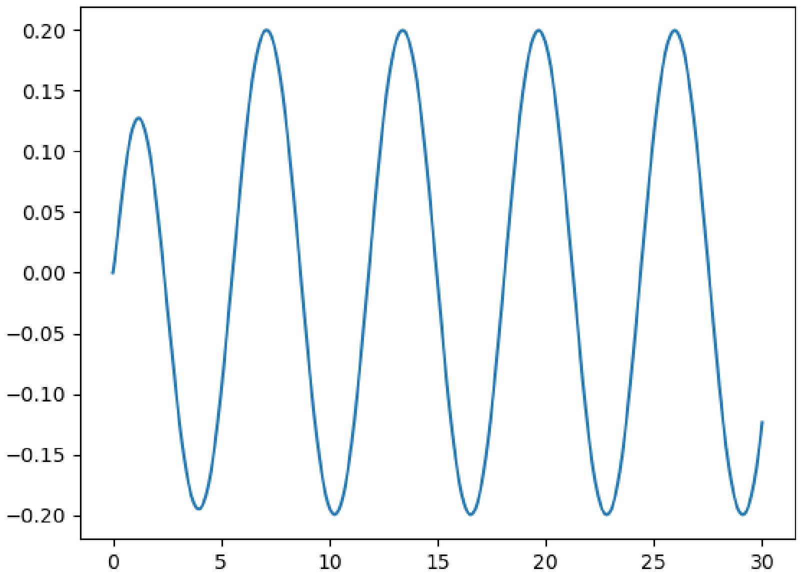


Figure 3.7. Direct problem solution to 3.41

$$\alpha^* = 0.6299212061401331$$

$$\mu^* = 4.000148503576592$$

$$k^* = 4.9991980561112195$$

$$\overline{\frac{|\alpha - \alpha^*|}{\alpha}} = 0.2\%$$

$$\overline{\frac{|\mu - \mu^*|}{\mu}} = 0.8\%$$

$$\overline{\frac{|k - k^*|}{k}} = 2\%$$

We still got more or less stable results, but the mean error is getting larger than in previous section. In the following chapter we will add one more derivative to the equation, increasing the number of parameters that must be estimated.

# 4. Inverse problems for fractional harmonic oscillators with two Caputo type operators

## 4.1 Derivatives of order $2\alpha$ and $\alpha$

In this section we will evaluate the following fractional equation:

$$\mu D^{2\alpha}x + \gamma D^{\alpha}x + kx = f(t), x(0^+) = 0, x'(0^+) = 0. \tag{4.1}$$

At $\alpha = 1$ it becomes the classical damped oscillation equation:

$$mD^2x + \gamma Dx + kx = f(t), \tag{4.2}$$

so it can be seen as a direct generalisation of that case.

### 4.1.1 Direct problem solution

As always, we begin with taking the Laplace transform from both sides of the equation. In s-space it takes form:

$$\mu s^{2\alpha}X(s) + \gamma s^{\alpha}X(s) + kX(s) = F(s). \tag{4.3}$$

And solving for $X(s)$ gives us:

$$X(s) = \frac{F(s)}{\mu s^{2\alpha} + \gamma s^{\alpha} + k}. \tag{4.4}$$

Solution for the equation in the time-domain is given by:

$$x(t) = \int_0^t f(\tau)g(t-\tau)d\tau, \tag{4.5}$$

where $g(t)$ is inverse Laplace transform of:

$$G(s) = \frac{1}{\mu^{2\alpha} + \gamma s^{\alpha} + k}.$$

The denominator is in fact a square polynomial in respect to $s^{\alpha}$, so it can be expanded as:

$$G(s) = \frac{1}{\mu(s^{\alpha} - b1)(s^{\alpha} - b2)},$$

$$b1 = \frac{-\gamma}{2\mu} + \frac{\sqrt{\gamma^2 - 4k\mu}}{2\mu},$$

$$b2 = \frac{-\gamma}{2\mu} - \frac{\sqrt{\gamma^2 - 4k\mu}}{2\mu}.$$

We now perform fractional decomposition:

$$\frac{G(s)}{m} = \frac{1}{(s^\alpha - b1)(s^\alpha - b2)} = \frac{A}{s^\alpha - b1} + \frac{B}{s^\alpha - b2} = \frac{A * (s^\alpha - b2) + B * (s^\alpha - b1)}{(s^\alpha - b1)(s^\alpha - b1)}.$$

We get the relation:

$$A * s^\alpha - A * b2 + B * s^\alpha - B * b1 = 1,$$

which gives us the following system of equations:

$$\begin{cases} A + B = 0 \\ A * b2 + B * b1 = -1 \end{cases}.$$

Solving for $A$ and $B$ gives:

$$A = \frac{-1}{b2 - b1}; B = \frac{1}{b2 - b1}.$$

Giving us the desired fractional decomposition:

$$\frac{G(s)}{m} = \frac{\frac{-1}{b2-b1}}{s^\alpha - b1} + \frac{\frac{1}{b2-b1}}{s^\alpha - b2} = \frac{1}{b2 - b1} * (\frac{1}{s^\alpha - b2} - \frac{1}{s^\alpha - b1}),$$

$$b2 - b1 = \frac{-\gamma}{2\mu} - \frac{\sqrt{\gamma^2 - 4k\mu}}{2\mu} - \frac{-\gamma}{2\mu} - \frac{\sqrt{\gamma^2 - 4k\mu}}{2\mu} = -\frac{\sqrt{\gamma^2 - 4k\mu}}{\mu},$$

$$G(s) = \frac{1}{\sqrt{\gamma^2 - 4k\mu}}(\frac{1}{s^\alpha - b1} - \frac{1}{s^\alpha - b2}).$$

Now we take the inverse Laplace transform, which once again involves the $\alpha$-exponentials:

$$g(t) = \frac{e^t_{b1,\alpha} - e^t_{b2,\alpha}}{\sqrt{\gamma^2 - 4k\mu}}. \tag{4.6}$$

And now by convolving $g(t)$ and $f(t)$ we obtain the full solution:

$$x(t) = \int_0^t \frac{e^\tau_{b1,\alpha} - e^\tau_{b2,\alpha}}{\sqrt{\gamma^2 - 4k\mu}} f(t - \tau) d\tau, \tag{4.7}$$

$$b1 = \frac{-\gamma}{2\mu} + \frac{\sqrt{\gamma^2 - 4k\mu}}{2\mu},$$

$$b2 = \frac{-\gamma}{2\mu} - \frac{\sqrt{\gamma^2 - 4k\mu}}{2\mu}.$$

### 4.1.2 Inverse problem solution

For an inverse problem for equation (4.1), we must find parameters $\mu$, $\gamma$, $k$ and $\alpha$. The function $Q(s)$ in this case takes form:

$$Q(s) = \mu s^{2\alpha} + \gamma s^\alpha + k. \tag{4.8}$$

Taking the derivative from both sides:

$$Q'(s) = 2\alpha\mu s^{2\alpha-1} + \alpha\gamma s^{\alpha-1}.$$

And multiplying by $s^{1-\alpha}$:

$$s^{1-\alpha}Q'(s) = 2\alpha\mu s^{\alpha} + \alpha\gamma.$$

We can continue this procedure:

$$(s^{1-\alpha}Q'(s))' = 2\alpha^2\mu s^{\alpha-1} \Rightarrow$$

$$s^{1-\alpha}(s^{1-\alpha}Q'(s))' = 2\alpha^2\mu \Rightarrow$$

$$\left[s^{1-\alpha}(s^{1-\alpha}Q'(s))\right]' = 0.$$

Expanding the last equation leads to:

$$(1-\alpha)(1-2\alpha)Q'(s) + 3(1-\alpha)sQ''(s) + s^2Q'''(s) = 0.$$

Now we choose different values $s_1$ and $s_2$, which gives us two equations:

$$(1-\alpha)(1-2\alpha)Q'(s_1) + 3(1-\alpha)sQ''(s_1) + s_1^2Q'''(s_1) = 0,$$

$$(1-\alpha)(1-2\alpha)Q'(s_2) + 3(1-\alpha)sQ''(s_2) + s_2^2Q'''(s_2) = 0.$$

We multiply the first equation by $Q'(s_2)$ and the second equation by $Q'(s_1)$, and then subtract the second equation from the first. This leads to:

$$3(1-\alpha)\left[s_1Q''(s_1)Q'(s_2) - s_2Q''(s_2)Q'(s_1)\right] = s_2^2Q'''(s_2)Q'(s_1) - s_1^2Q'''(s_1)Q'(s_2).$$

Before going further we must control that $s_1Q''(s_1)Q'(s_2) - s_2Q''(s_2)Q'(s_1)$ is not equal to zero. Considering that:

$$sQ''(s) = 2\alpha(2\alpha-1)\mu s^{2\alpha-1} + \alpha(\alpha-1)\gamma s^{\alpha-1},$$

$$Q'(s) = 2\alpha\mu s^{2\alpha-1} + \alpha\gamma s^{\alpha-1}.$$

We get:

$$s_1Q''(s_1)Q'(s_2) - s_2Q''(s_2)Q'(s_1) = .... = 2\alpha^3\mu\gamma s_1^{\alpha-1}s_2^{\alpha-1}(s_1^{\alpha} - s_2^{\alpha}) \neq 0.$$

With this, we can directly obtain $\alpha$:

$$\alpha = 1 - \frac{1}{3}\left[\frac{s_2^2Q'''(s_2)Q'(s_1) - s_1^2Q'''(s_1)Q'(s_2)}{s_1Q''(s_1)Q'(s_2) - s_2Q''(s_2)Q'(s_1)}\right]. \tag{4.9}$$

With $\alpha$ in our hands, once again figuring out rest of the parameters requires constructing a linear system of equations using:

$$Q^{(i)}(j) = A_{i,j}\mu + B_{i,j}\gamma + C_i k, \tag{4.10}$$

$$A_{i,j} = j^{2\alpha-i}\frac{\Gamma(2\alpha+1)}{\Gamma(2\alpha-i+1)}, \tag{4.11}$$

$$B_{i,j} = j^{\alpha-i}\frac{\Gamma(\alpha+1)}{\Gamma(\alpha-i+1)}, \tag{4.12}$$

$$C_i = \begin{cases} 1, & i=0 \\ 0, & i>0 \end{cases}. \tag{4.13}$$

With the equation matrix:

$$A = \begin{bmatrix} A_{i_1,j_1} & B_{i_1,j_1} & C_{i_1} \\ A_{i_2,j_2} & B_{i_2,j_2} & C_{i_2} \\ A_{i_3,j_3} & B_{i_3,j_3} & C_{i_3} \end{bmatrix}. \tag{4.14}$$

Condition number wise, the best option is to use equations $(0,1)$, $(1,1)$ and $(2,1)$. Plot for the $\kappa(A)$ versus $\alpha$ can be seen on Fig 4.1. The system gets more unstable as $\alpha$ is decreased and is at its best in range $\alpha = [0.8, 0.9]$.

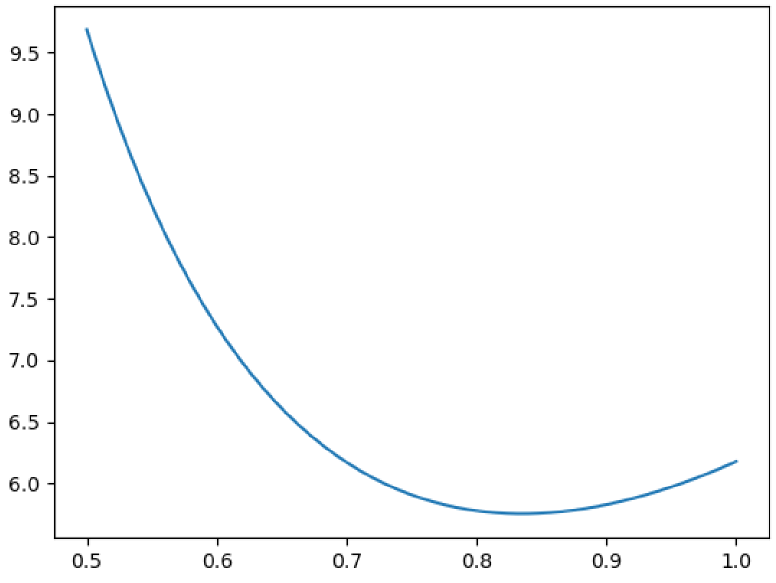


Figure 4.1. Condition number for system of equations with $(0,1)$, $(1,1)$ and $(2,1)$ with $\alpha \in [0.5, 1]$

### 4.1.3 Numerical results

As in previous sections, we will now numerically apply our inverse model to 3 concrete model examples. $\delta$ is set to $10^{-3}$.

Model 1:

$$2D^{2\cdot 0.90} + 3D^{0.90} + 4x = e^{-t}. \tag{4.15}$$

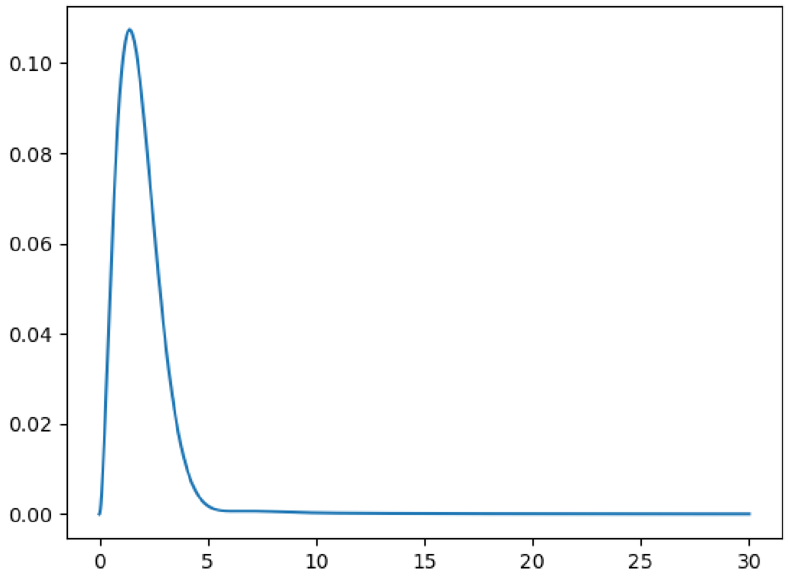


Figure 4.2. Direct problem solution to 4.15

$$\alpha^* = 0.8995681009135778$$

$$\mu^* = 2.00318531440018$$

$$\gamma^* = 2.9962319063620044$$

$$k^* = 4.000037143956829$$

$$\overline{\frac{|\alpha - \alpha^*|}{\alpha}} = 4\%$$

$$\overline{\frac{|\mu - \mu^*|}{\mu}} = 33\%$$

$$\overline{\frac{|\gamma - \gamma^*|}{\gamma}} = 35\%$$

$$\overline{\frac{|k - k^*|}{k}} = 3\%$$

Model 2:

$$5D^{2\cdot 0.69}x + D^{0.69}x + 1.5x = 0.1sin(2t). \tag{4.16}$$

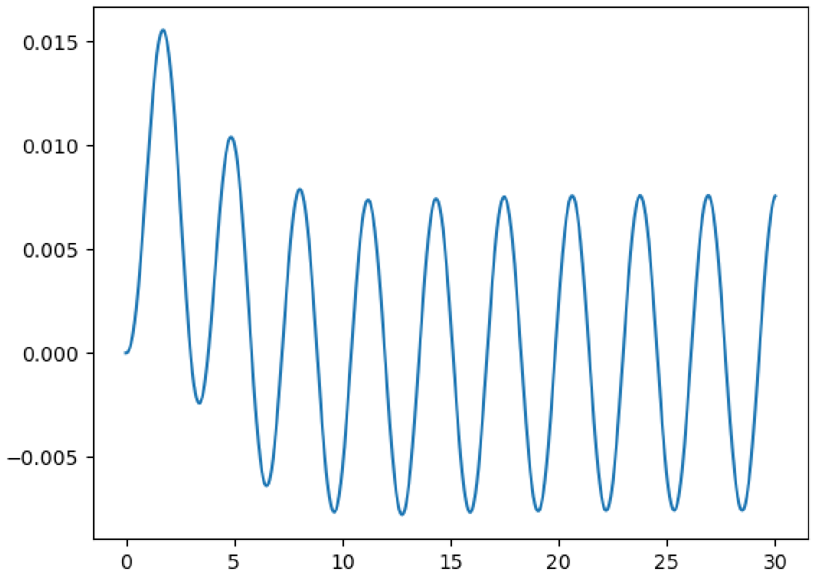


Figure 4.3. Direct problem solution to 4.16

$$\alpha^* = 0.6900266351019029$$

$$\mu^* = 4.999435488569127$$

$$\gamma^* = 1.000746460940464$$

$$k^* = 1.4998401003295614$$

$$\overline{\frac{|\alpha - \alpha^*|}{\alpha}} > 100\%$$

$$\overline{\frac{|\mu - \mu^*|}{\mu}} > 100\%$$

$$\overline{\frac{|\gamma - \gamma^*|}{\gamma}} > 100\%$$

$$\overline{\frac{|k - k^*|}{k}} > 100\%$$

Model 3:

$$3D^{2\cdot 0.84}x + 3D^{0.84}x + x = t(t - 30). \tag{4.17}$$

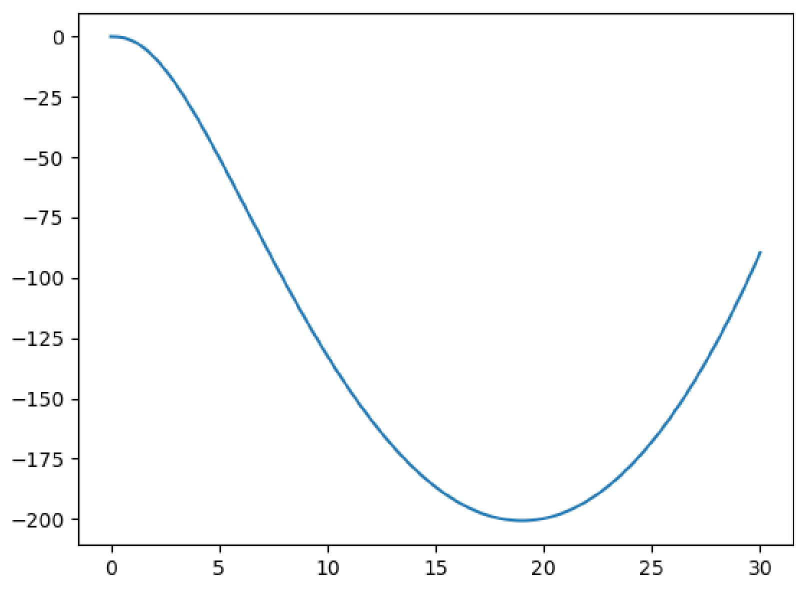


Figure 4.4. Direct problem solution to 4.17

$$\alpha^* = 0.8399932728146717$$

$$\mu^* = 3.0000917655332184$$

$$\gamma^* = 1.000746460940464$$

$$k^* = 1.0000148087619336$$

$$\frac{|\alpha - \alpha^*|}{\alpha} = 1\%$$

$$\frac{|\mu - \mu^*|}{\mu} = 17\%$$

$$\frac{|\gamma - \gamma^*|}{\gamma} = 20\%$$

$$\frac{|k - k^*|}{k} = 2\%$$

As we can see, the inverse problems for this type of equation are much more unstable, and the mean error grows as $\alpha$ gets smaller. Errors for parameters $\mu$ and $\gamma$ are one order bigger than those for parameters $\alpha$ and $k$.

## 4.2 Derivatives of order $2\alpha$ and $2\beta$

In this section we will evaluate a lot more general fractional equation:

$$\mu D^{2\alpha}x + \gamma D^{2\beta}x + kx = f(t), x(0^+) = 0, x'(0^+) = 0, \tag{4.18}$$

where it is assumed that $\beta \leq \alpha$. With $\beta = \frac{\alpha}{2}$ we get equation (4.1).

### 4.2.1 Direct problem solution

Solution to the equation (4.18) in the Laplace transform space takes form:

$$X(s) = \frac{F(s)}{\mu^{2\alpha} + \gamma^{2\beta} + k}, \tag{4.19}$$

and solution in the time domain:

$$x(t) = \int_0^t g(t-\tau)f(\tau)d\tau, \tag{4.20}$$

where $g(t)$ is inverse Laplace transform of $G(s)$:

$$G(s) = \frac{1}{\mu s^{2\alpha} + \gamma s^{2\beta} + k}. \tag{4.21}$$

Analytical solution is available only for some special cases of the equation. A solution to Bagley-Torvik equation with $\alpha = 1$, $\beta = \frac{3}{4}$, $\mu = 1$, $\gamma = 2$ and $k = 2$ is presented in [10], with the inverse Laplace transform of $G(s)$ taking form of a functional series with Prabhakar functions:

$$g(t) = \sum_{k=0}^{\infty}(-1)^k \frac{2^k}{k!} t^{2k+1} E^k_{0.5,2+3k/2}(-2t^{0.5}). \tag{4.22}$$

To provide a full numerical solution for general equation, inverse Laplace transform function provided within the mpmath module for Python was used. The algorithm works by replacing the Bromwich contour integral with a series obtained after application of the trapezoidal rule, convergence of which is then accelerated using a linear acceleration method [22]. As suggested, multiprocessing was used for parallel computation and even faster result.

### 4.2.2 Inverse problem solution

Now polynomial $Q(s)$ for us takes the following form:

$$Q(s) = \mu s^{2\alpha} + \gamma s^{2\beta} + k. \tag{4.23}$$

For an inverse problem, we now have to determine 5 parameters: $\alpha$, $\beta$, $\mu$, $\gamma$ and $k$. An additional condition that we introduce is $\beta < \alpha$. Now we will use the same procedure as in the previous section to get rid of parameters $\mu$, $\gamma$ and $k$:

$$Q(s) = \mu s^{2\alpha} + \gamma s^{2\beta} + k \Rightarrow$$

$$Q'(s) = 2\alpha\mu s^{2\alpha-1} + 2\beta\gamma s^{2\beta-1} \Rightarrow$$

$$s^{1-2\alpha}Q'(s) = 2\alpha\mu + 2\beta\gamma s^{2\beta-2\alpha} \Rightarrow$$

$$[s^{1-2\alpha}M'(s)]' = 2\beta(2\beta-2\alpha)\gamma s^{2\beta-2\alpha-1} \Rightarrow$$

$$s^{1+2\alpha-2\beta}[s^{1-2\alpha}M'(s)]' = 2\beta(2\beta-2\alpha)\gamma.$$

We arrive at equation:

$$[s^{1+2\alpha-2\beta}[s^{1-2\alpha}Q'(s)]']' = 0.$$

Expanding it leads to the following expression:

$$(1-2\beta-2\alpha+4\alpha\beta)Q'(s) + (3-2\alpha-2\beta)sQ''(s) + s^2Q'''(s) = 0,$$

which can be rewritten as:

$$(2\alpha+2\beta)[1+\frac{sQ''(s)}{Q'(s)}] - 4\alpha\beta = 1 + \frac{3sQ''(s)}{Q'(s)} + \frac{s^2Q'''(s)}{Q'(s)}.$$

Now for simplicity we will define two parameters $P_1$ and $P_2$:

$$P_1(s) = 1 + \frac{sQ''(s)}{Q'(s)}, \tag{4.24}$$

$$P_2(s) = 1 + \frac{3sQ''(s)}{Q'(s)} + \frac{s^2Q'''(s)}{Q'(s)}. \tag{4.25}$$

With this we have:

$$(2\alpha+2\beta)P_1(s) - 4\alpha\beta = P_2(s).$$

For two different points $s_1$ and $s_2$:

$$\begin{cases} (2\alpha+2\beta)P_1(s_1) - 4\alpha\beta = P_2(s_1) \\ (2\alpha+2\beta)P_1(s_2) - 4\alpha\beta = P_2(s_2) \end{cases}. \tag{4.26}$$

With:

$$X = 2\alpha + 2\beta,$$

$$Y = 4\alpha\beta,$$

we get a linear system of equations:

$$\begin{cases} X \cdot P_1(s_1) - Y = P_2(s_1) \\ X \cdot P_1(s_2) - Y = P_2(s_2) \end{cases}. \tag{4.27}$$

Before solving the system, we must ensure that it is not in fact singular. Determinant of the system would be:

$$P_1(s_1) - P_1(s_2) = 1 + \frac{s_1 Q''(s_1)}{Q'(s_1)} - 1 - \frac{s_2 Q''(s_2)}{Q'(s_2)} = \frac{s_1 Q''(s_1)}{Q'(s_1)} - \frac{s_2 Q''(s_2)}{Q'(s_2)} \Rightarrow$$

$$P_1(s_1) - P_1(s_2) = \frac{s_1 Q''(s_1) Q'(s_2) - s_2 Q''(s_2) Q'(s_1)}{Q'(s_1) Q'(s_2)}$$

This expression will not be equal to 0 if the numerator is not equal to 0:

$$s_1 Q''(s_1) Q'(s_2) - s_2 Q''(s_2) Q'(s_1) \neq 0.$$

Recalling that:

$$sQ''(s) = 2\alpha(2\alpha - 1)\mu s^{2\alpha-1} + 2\beta(2\beta - 1)\gamma s^{2\beta-1},$$

$$Q'(s) = 2\alpha\mu s^{2\alpha-1} + 2\beta\gamma s^{2\beta-1}.$$

We get:

$$s_1 Q''(s_1) Q'(s_2) - s_2 Q''(s_2) Q'(s_1) = ... = 8\alpha\beta\mu\gamma(\alpha - \beta)(s_1 s_2)^{2\beta-1}(s_1^{2\alpha-2\beta} - s_2^{2\alpha-2\beta}) \neq 0.$$

So the system (4.27) is not singular and therefore can be solved. We compute:

$$X = \frac{P_2(s_1) - P_2(s_2)}{P_1(s_1) - P_1(s_2)},$$

$$Y = X \cdot P_1(s_1) - P_2(s_1),$$

and get the following system for parameters $\alpha$ and $\beta$:

$$\begin{cases} 2\alpha + 2\beta = X \\ 4\alpha\beta = Y \end{cases}, \tag{4.28}$$

$$\begin{cases} \alpha + \beta = \frac{X}{2} \\ \alpha\beta = \frac{Y}{4} \end{cases}. \tag{4.29}$$

The system corresponds to a quadratic equation:

$$p^2 - \frac{X}{2} \cdot p + \frac{Y}{4} = 0,$$

with solutions:

$$p_{1,2} = \frac{-\frac{X}{2} \pm \sqrt{\frac{X^2}{4} - Y}}{2}.$$

Due to our condition that $\alpha > \beta$ we select:

$$\max(p_1, p_2) = \alpha$$

$$\min(p_1, p_2) = \beta$$

Now that we have $\alpha$ and $\beta$, once again we construct a linear system of equations:

$$Q^{(i)}(j) = A_{i,j}\mu + B_{i,j}\gamma + C_i k, \tag{4.30}$$

$$A_{i,j} = j^{2\alpha-i}\frac{\Gamma(2\alpha+1)}{\Gamma(2\alpha-i+1)}, \tag{4.31}$$

$$B_{i,j} = j^{2\beta-i}\frac{\Gamma(2\beta+1)}{\Gamma(2\beta-i+1)}, \tag{4.32}$$

$$C_i = \begin{cases} 1, & i = 0 \\ 0, & i > 0 \end{cases}. \tag{4.33}$$

Best option is still to use equations $(0,1)$, $(1,1)$ and $(2,1)$, although the condition number is larger than that acquired for the system in previous section.

### 4.2.3 Numerical results

Let us try again to apply the newly acquired inverse problem numerically. $\delta$ is set to $10^{-3}$.

Model 1:

$$D^2x + 3D^{\frac{3}{2}}x + 2x = \frac{1}{\sqrt{2\pi}}e^{-\frac{(x-5)^2}{2}}. \tag{4.34}$$

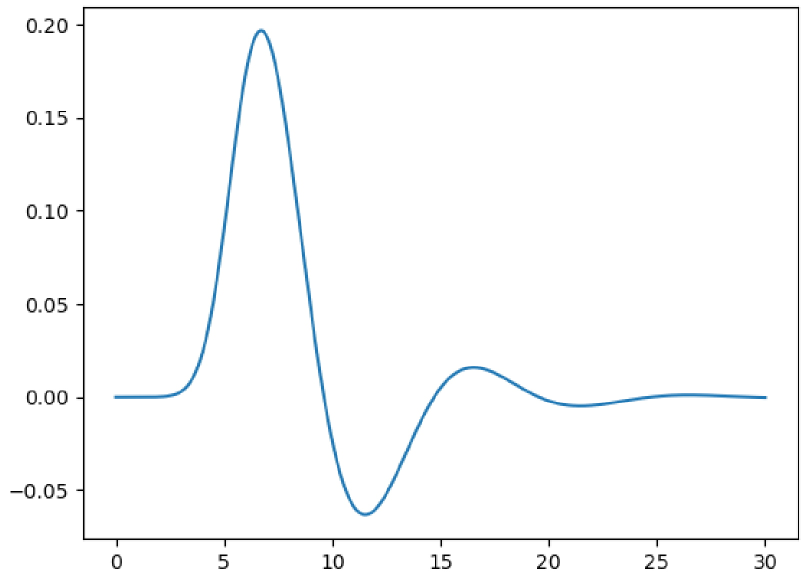


Figure 4.5. Direct problem solution to 4.34

$$\alpha^* = 0.9997590303280529$$

$$\beta^* = 0.7498932901349323$$

$$\mu^* = 1.0021682890722818$$

$$\gamma^* = 2.997866072355463$$

$$k^* = 1.9999776248307295$$

$$\overline{\frac{|\alpha - \alpha^*|}{\alpha}} > 100\%$$

$$\overline{\frac{|\beta - \beta^*|}{\beta}} > 100\%$$

$$\overline{\frac{|\mu - \mu^*|}{\mu}} > 100\%$$

$$\overline{\frac{|\gamma - \gamma^*|}{\gamma}} > 100\%$$

$$\overline{\frac{|k - k^*|}{k}} > 100\%$$

This problem is the most unstable so far. Let us reduce the $\delta$ to $10^{-4}$ and see what happens.

Model 2:

$$6D^{2\cdot 0.95}x + 2D^{2*0.75}x + x = e^{-0.4t}. \tag{4.35}$$

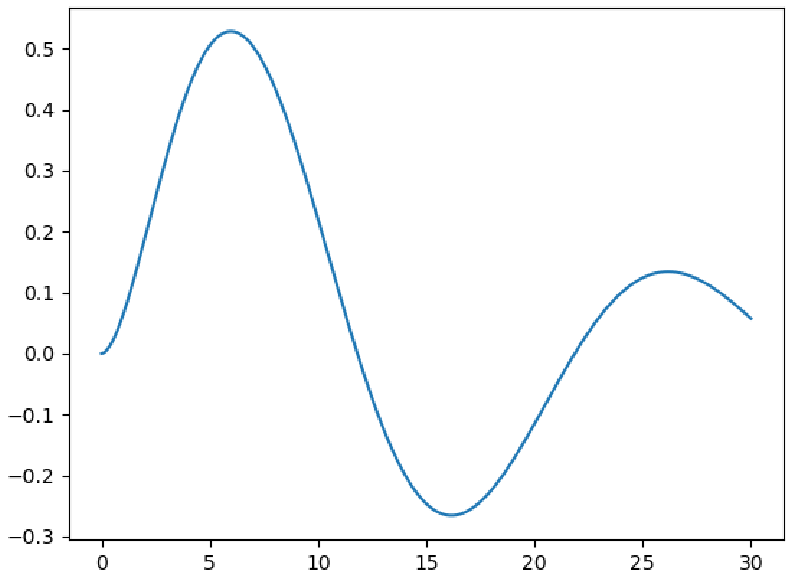


Figure 4.6. Direct problem solution to 4.35

$$\alpha^* = 0.9493693634264985$$

$$\beta^* = 0.7479471903249001$$

$$\mu^* = 6.037660024096634$$

$$\gamma^* = 1.9621752145976479$$

$$k^* = 0.9998027361804949$$

$$\frac{|\alpha - \alpha^*|}{\alpha} = 3\%$$

$$\frac{|\beta - \beta^*|}{\beta} = 16\%$$

$$\frac{|\mu - \mu^*|}{\mu} = 27\%$$

$$\frac{|\gamma - \gamma^*|}{\gamma} = 80\%$$

$$\frac{|k - k^*|}{k} = 2\%$$

Model 3:

$$2.3D^{2\cdot 0.75}x + 1.5D^{2*0.56}x + 1.4x = t^2(t-30). \tag{4.36}$$

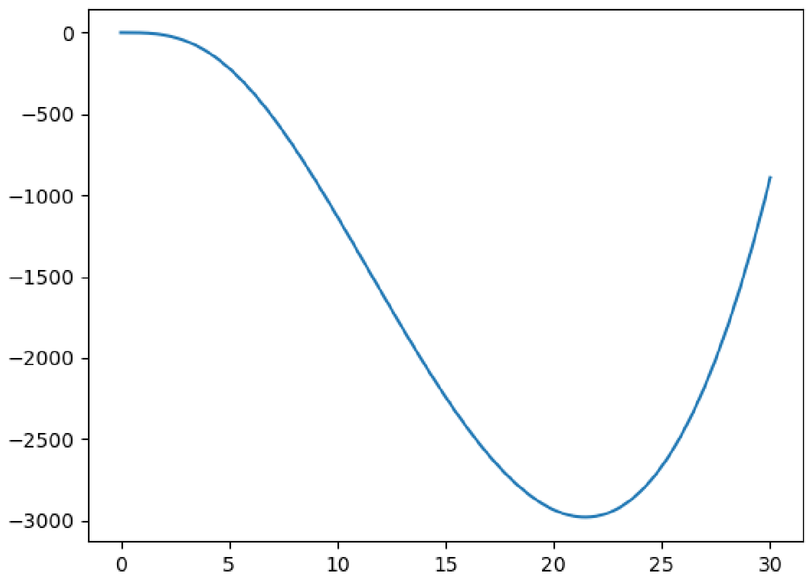


Figure 4.7. Direct problem solution to 4.36

$$\alpha^* = 0.7501305405909421$$

$$\beta^* = 0.560260460781982$$

$$\mu^* = 2.2964954417283288$$

$$\gamma^* = 1.503470764302789$$

$$k^* = 1.4000455509451897$$

$$\overline{\frac{|\alpha - \alpha^*|}{\alpha}} = 0.6\%$$

$$\overline{\frac{|\beta - \beta^*|}{\beta}} = 2\%$$

$$\overline{\frac{|\mu - \mu^*|}{\mu}} = 27\%$$

$$\overline{\frac{|\gamma - \gamma^*|}{\gamma}} = 8\%$$

$$\overline{\frac{|k - k^*|}{k}} = 0.1\%$$

Reducing size of the random perturbations slightly improved the results. Mean errors for some of the parameters are still order bigger than for others.

# 5. Summary

Throughout this work, Laplace transform method was used to construct inverse problem solutions for fractional oscillators with one and two Caputo derivative operators, and then the stability of each solution was controlled numerically. Despite all obtained solutions giving good results for unperturbed data, with each new introduced parameter the stability of an inverse problem got largely worse. One of the solutions for a model with 5 unknown parameters gave mean errors larger than 100% for every parameter after introducing small random interference into the input data. It can be concluded from this that only the simplest inverse problem solutions obtained with the used method are usable for real measurements, but opens up possibilities for further research in improving stability for solutions with two or potentially more fractional derivative operators.

## Acknowledgements

I would like to express my gratitude to supervisor Jaan Janno for introducing me into the beautiful world of fractional calculus, as well as providing help, support and minor language corrections.

## References

[1] Charles W. Groetsch, *Inverse Problems in The Mathematical Sciences* Springer (1993)

[2] Hanne Kekkonen, Yury Korolev, *Inverse Problems. Lecture notes.* University of Cambridge (2020) URL: `https://www.damtp.cam.ac.uk/research/cia/files/teaching/Inverse_Problems_2019/LectureNotes2019.pdf`

[3] R. Hilfer, *Applications of Fractional Calculus in Physics* World Scientific (2000)

[4] R.L. Magin, *Fractional Calculus in Bioengineering* Begell House Publisher (2006)

[5] Esra Karatas Akgül, Ali Akgül, Dumitru Baleanu, *Laplace Transform Method for Economic Models with Constant Proportional Caputo Derivative* Fractal and Fractional 2020, 4(3), 30 URL: `https://doi.org/10.3390/fractalfract4030030`

[6] Weisstein, Eric W., *Laplace Transform* From MathWorld–A Wolfram Web Resource. URL: `https://mathworld.wolfram.com/LaplaceTransform.html`

[7] J. Abate, W. Whitt, *Numerical Inversion of Laplace Transforms of Probability Distributions* ORSA Journal on Computing, 7(1):36-43

[8] Alar Leibak, *Kompleksmuutuja Funktsioonid* TTÜ Kirjastus (2015)

[9] R. Gorenflo, A.A. Kilbas, F. Mainardi, S.V. Rogosin, *Mittag-Leffler Functions, Related Topics and Applications* Springer Monographs in Mathematics (2014)

[10] Kai Diethelm, *The Analysis of Fractional Differential Equations* Springer Lecture Notes in Mathematics (2010)

[11] Alexander I. Zhmakin, *A Compact Introduction to Fractional Calculus* URL: `https://doi.org/10.48550/arXiv.2301.00037`

[12] Richard Herrmann, *Fractional Calculus An Introduction For Physicists* World Scientific (2011)

[13] José Francisco Gómez-Aguilar, J. Juan Rosales-García, José de Jesús Bernal-Alvarado, Teodoro Córdova-Fraga, Rafael Guzmán-Cabrera, *Fractional mechanical oscillators* Revista Mexicana De Fisica 58 (2012) 348–352 URL: `https://api.semanticscholar.org/CorpusID:54812905`

[14] A. A. Stanislavsky, *Fractional oscillator* URL: `https://doi.org/10.48550/arXiv.1111.3060`

[15] Hossein Fazli, Juan J. Nieto, *An investigation of fractional Bagley-Torvik equation* Open Mathematics (2019), 17:499–512 URL: `https://doi.org/10.1515/math-2019-0040`

[16] SciPy API Reference: integrate.simpson URL: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.simpson.html`

[17] Jorge Hernan Lopez, numfracpy module URL: https://pypi.org/project/numfracpy/

[18] SciPy API Reference: signal.convolve URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve.html

[19] Christos Xenophontos, *A formula for the nth derivative of the quotient of two functions* URL: https://doi.org/10.48550/arXiv.2110.09292

[20] Condition number. Encyclopedia of Mathematics. URL: http://encyclopediaofmath.org/index.php?title=Condition_number&oldid=50975

[21] NumPy API Reference: linalg.cond URL: https://numpy.org/doc/stable/reference/generated/numpy.linalg.cond.html

[22] Guillermo Navas-Palencia, *Faster numerical inverse Laplace transform in mpmath* URL: http://gnpalencia.org/blog/2022/invertlaplace/

# Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

## Appendix 2 - Python Scripts

Python scripts used to obtain numerical results in this work. Non-standard packages expected: mpmath, numfracpy, multiprocessing. CPU with 6 cores, 12 threads and 2200 MHz clock speed was used.

Script used for Section 3.1.3:

```
import matplotlib.pyplot as plt
import numpy as np
import numfracpy as nfp
import scipy as sc

def laplace_transform(s,x,t,der):
    x_laplace_int=x*((-t)**der)*np.exp(-s*t)
    X=sc.integrate.simpson(x_laplace_int, x=t)
    return X

def g_sol(t,p,x0):
    a=p[0]
    w=p[1]
    return [x0[0]*nfp.Mittag_Leffler_one(-w*i**(2*a), 2*a) +\
        x0[1]*i*nfp.Mittag_Leffler_two(-w*i**(2*a), 2*a, 2) for i in t]

def caputo_inverse(t,x,X0):

    X=laplace_transform(1,x,t,0)
    DX=laplace_transform(1,x,t,1)

    w2_c = ((X0[0]+X0[1])/X)-1

    p=1/(1+w2_c)
    p1=1/((X0[0]+X0[1])*(1-p))
    p2=X0[0]+2*X0[1]+(DX/p)

    a_c=0.5*p1*p2

    return [a_c,w2_c]

a=0.95
```

```
w=1
p=[a,w]

M=30.
N=500000

x0=[3,2]

t = np.linspace(0, M, N)
x=g_sol(t,p,x0)
plt.plot(t,x)
plt.savefig("31_m3.png")

t_new=t[0::500]
x_new=x[0::500]
inv=caputo_inverse(t_new,x_new,x0)
print(inv)

err=1e-3
a_err=np.zeros(100)
w_err=np.zeros(100)

for i in range(99):
    x_err=np.array([i + err*np.random.normal(loc=0.0, scale=1.0) for i in x_new])
    inv_err=caputo_inverse(t_new, x_err, x0)
    a_err[i]=np.abs(a-inv_err[0])
    w_err[i]=np.abs(w-inv_err[1])

print(np.mean(a_err))
print(np.mean(w_err))
```

Script used for Section 3.2.3::

```
import matplotlib.pyplot as plt
import numpy as np
import numfracpy as nfp
import math
import scipy as sc

def laplace_transform(s,x,t,der):
    x_laplace_int=x*((-t)**der)*np.exp(-s*t)
    X=sc.integrate.simpson(x_laplace_int, x=t)
    return X

def convolution(x,f,t,dt):
        n=len(t)
        g_t=sc.signal.fftconvolve(x,f, mode='full')
        g=np.array(g_t[:n])
        return np.real(g*dt)

def g_sol(t,p):
    a=p[0]
    m=p[1]
    k=p[2]
    g=[(1/m)*(i**(2*a-1))*nfp.Mittag_Leffler_two((-k/m)*i**(2*a), 2*a, 2*a) for i in t]
    return g

def Q(t,s,n,f,x):
    X=np.zeros(n+1)
    F=np.zeros(n+1)
    Q=np.zeros(n+1)
    for i in range(n+1):
        X[i]=laplace_transform(s,x,t,i)
        F[i]=laplace_transform(s,f,t,i)
    Q[0]=F[0]/X[0]
    for i in range(1,n+1):
        sm=0
        for j in range(1,i+1):
            sm=sm + (X[i+1-j]*Q[j-1])/(math.factorial((i+1-j))*math.factorial((j-1)))
        Q[i]=(1/X[0])*(F[i] - math.factorial(i)*sm)
    return Q

def caputo_inhom_inverse(x,f,t):
     Q1=Q(t,1,2,f,x)

     a_r = 0.5 + (Q1[2]/(2*Q1[1]))
     m_r = Q1[1]/(2*a_r)
     k_r = Q1[0] - m_r

     return [a_r,m_r,k_r]

a_c=0.7
m_c=3
```

```
k_c=4
p = [a_c,m_c,k_c]

def f(t):
    return np.exp(-2*t)

st=1e-30
sp=30
n=500000
t=np.linspace(st,sp,n)
dt=(sp-st)/n

g=g_sol(t,p)

x = convolution(g,f(t),t,dt)

plt.plot(t,x)
plt.savefig('image.png')

t_new=t[0::500]
x_new=x[0::500]

inv=caputo_inhom_inverse(x_new,f(t_new),t_new)

print(inv)

err=1e-3
a_err=np.zeros(100)
m_err=np.zeros(100)
k_err=np.zeros(100)

for i in range(99):
    x_err=np.array([i + err*np.random.normal(loc=0.0, scale=1.0) for i in x_new])
    inv_err=caputo_inhom_inverse(x_err,f(t_new),t_new)
    a_err[i]=np.abs(a_c-inv_err[0])
    m_err[i]=np.abs(m_c-inv_err[1])
    k_err[i]=np.abs(k_c-inv_err[2])

print(np.mean(a_err))
print(np.mean(m_err))
print(np.mean(k_err))
```

Script used for Section 4.1.3::

```
import matplotlib.pyplot as plt
import numpy as np
import numfracpy as nfp
import scipy as sc
import math
import cmath

def laplace_transform(s,x,t,der):
    x_laplace_int=x*((-t)**der)*np.exp(-s*t)
    X=sc.integrate.simpson(x_laplace_int, x=t)
    return X

def convolution(x,f,t,dt):
        n=len(t)
        g_t=sc.signal.fftconvolve(x,f, mode='full')
        g=np.array(g_t[:n])
        return np.real(g*dt)

def Q(t,s,n,f,x):
    X=np.zeros(n+1)
    F=np.zeros(n+1)
    Q=np.zeros(n+1)
    for i in range(n+1):
        X[i]=laplace_transform(s,x,t,i)
        F[i]=laplace_transform(s,f,t,i)
    Q[0]=F[0]/X[0]
    for i in range(1,n+1):
        sm=0
        for j in range(1,i+1):
            sm=sm + (X[i+1-j]*Q[j-1])/(math.factorial((i+1-j))*math.factorial((j-1)))
        Q[i]=(1/X[0])*(F[i] - math.factorial(i)*sm)
    return Q

def g_sol(t,p):
    a=p[0]
    m=p[1]
    u=p[2]
    k=p[3]
    z1=(-u+cmath.sqrt(u**2 - 4*k*m))/(2*m)
    z2=(-u-cmath.sqrt(u**2 - 4*k*m))/(2*m)
    A=1/(cmath.sqrt(u**2-4*k*m))
    v1=(t**(a-1))*nfp.Mittag_Leffler_two(z1*(t**a), a, a)
    v2=(t**(a-1))*nfp.Mittag_Leffler_two(z2*(t**a), a, a)
    v=A*(v1-v2)
    return np.real(v)

def caputo_damped_inverse(x,f,t):
    Q1=Q(t,1,3,f,x)
    Q2=Q(t,2,3,f,x)

    a_n=4*Q2[3]*Q1[1]-Q1[3]*Q2[1]
    a_d=3*(Q1[2]*Q2[1]-2*Q2[2]*Q1[1])
    a=1-(a_n/a_d)

    eq_l=[[1,1,1],[2*a,a,0], [2*a*(2*a-1),a*(a-1),0]]
    eq_r=[Q1[0],Q1[1],Q1[2]]
    s=sc.linalg.solve(eq_l,eq_r)
    m=s[0]
    u=s[1]
    k=s[2]

    return [a,m,u,k]

a_c=0.84
m_c=3
u_c=3
k_c=1
p = [a_c,m_c,u_c,k_c]

def f(t):
    return t*(t-30)

st=1e-30
sp=30
n=500000
t=np.linspace(st,sp,n)
dt=(sp-st)/n

g=[g_sol(i,p) for i in t]

x = convolution(g,f(t),t,dt)
```

```
plt.plot(t,x)
plt.savefig('41_m3.png')

t_new=t[0::500]
x_new=x[0::500]

inv=caputo_damped_inverse(x_new,f(t_new),t_new)

print(inv)

err=1e-3
a_err=np.zeros(100)
m_err=np.zeros(100)
u_err=np.zeros(100)
k_err=np.zeros(100)

for i in range(99):
    x_err=np.array([i + err*np.random.normal(loc=0.0, scale=1.0) for i in x_new])
    inv_err=caputo_damped_inverse(x_err,f(t_new),t_new)
    a_err[i]=np.abs(a_c-inv_err[0])
    m_err[i]=np.abs(m_c-inv_err[1])
    u_err[i]=np.abs(u_c-inv_err[2])
    k_err[i]=np.abs(k_c-inv_err[3])

print(np.mean(a_err))
print(np.mean(m_err))
print(np.mean(u_err))
print(np.mean(k_err))
```

Script used for Section 4.2.3::

```
import matplotlib.pyplot as plt
import numpy as np
import scipy as sc
import mpmath
import math
import cmath
from multiprocessing.pool import Pool
from functools import partial
import tqdm

def laplace_transform(s,x,t,der):
    x_laplace_int=x*((-t)**der)*np.exp(-s*t)
    X=sc.integrate.simpson(x_laplace_int, x=t)
    return X

def convolution(x,f,t,dt):
        n=len(t)
        g_t=sc.signal.fftconvolve(x,f, mode='full')
        g=np.array(g_t[:n])
        return np.real(g*dt)

def g_sol(t,p):
    a=p[0]
    b=p[1]
    m=p[2]
    u=p[3]
    k=p[4]
    lg= lambda s: 1/(m*(s**(2*a))+u*(s**(2*b)) + k)
    g=float(mpmath.invertlaplace(lg, t, method = 'cohen'))
    return g

def Q(t,s,n,f,x):
    X=np.zeros(n+1)
    F=np.zeros(n+1)
    Q=np.zeros(n+1)
    for i in range(n+1):
        X[i]=laplace_transform(s,x,t,i)
        F[i]=laplace_transform(s,f,t,i)
    Q[0]=F[0]/X[0]
    for i in range(1,n+1):
        sm=0
        for j in range(1,i+1):
            sm=sm + (X[i+1-j]*Q[j-1])/(math.factorial((i+1-j))*math.factorial((j-1)))
        Q[i]=(1/X[0])*(F[i] - math.factorial(i)*sm)
    return Q

def caputo_two_inverse(t,x,f):
    M1=Q(t,1,3,f,x)
    M2=Q(t,2,3,f,x)

    P11 = (M1[1]+M1[2])/M1[1]
    P12 = (M2[1] + 2*M2[2])/M2[1]
    P21 = (M1[1] + 3*M1[2] + M1[3])/M1[1]
    P22 = (M2[1] + 3*2*M2[2] + 4*M2[3])/M2[1]

    X=(P21-P22)/(P11-P12)
    Y=X*P11-P21

    B=-X/2
    C=Y/4

    D=(B**2)-4*C
    sD=np.abs(cmath.sqrt(D))

    a1 = (-B+sD)/2
    a2 = (-B-sD)/2
    if a1>=a2:
        a=a1
        b=a2
    else:
        a=a2
        b=a1
    eq_l=[[1,1,1],[2*a,2*b,0], [2*a*(2*a-1),2*b*(2*b-1),0]]
    eq_r=[M1[0],M1[1],M1[2]]
    s=sc.linalg.solve(eq_l,eq_r)
    m=s[0]
    u=s[1]
    k=s[2]
    return [a,b,m,u,k]

def f(t):
    return (t**2)*(t-30)

if __name__ == "__main__":
    p = Pool()
    st=1e-30
    sp=30
    n=500000
    t=np.linspace(st,sp,n)
    dt=(sp-st)/n
    a=0.75
    b=0.56
    m=2.3
    u=1.5
```

```
k=1.4
par=[a,b,m,u,k]

g_func = partial(g_sol, p=par)

mapped_values = list(tqdm.tqdm(p.imap(g_func, t), total=len(t)))

p.close()
p.join()

g=mapped_values

x = convolution(g,f(t),t,dt)

plt.plot(t,x)
plt.savefig('42_m3.png')

t_new=t[0::500]
x_new=x[0::500]

inv_sol=caputo_two_inverse(t_new,x_new,f(t_new))

print(inv_sol)

err=1e-4
samples=100
a_err=np.zeros(samples)
b_err=np.zeros(samples)
m_err=np.zeros(samples)
u_err=np.zeros(samples)
k_err=np.zeros(samples)

for i in range(samples-1):
    x_err=np.array([i + err*np.random.normal(loc=0.0, scale=1) for i in x_new])
    inv_err=caputo_two_inverse(t_new,x_err,f(t_new))
    a_err[i]=np.abs(a-inv_err[0])/a
    b_err[i]=np.abs(b-inv_err[1])/b
    m_err[i]=np.abs(m-inv_err[2])/m
    u_err[i]=np.abs(u-inv_err[3])/u
    k_err[i]=np.abs(k-inv_err[4])/k

print(np.mean(a_err))
```

```
    print(np.mean(b_err))
    print(np.mean(m_err))
    print(np.mean(u_err))
    print(np.mean(k_err))
```

Script used for calculating system condition numbers:

```
import numpy as np
import scipy as sc
import matplotlib.pyplot as plt

def C(i):
    if i==0:
        return 1
    else:
        return 0

def A(i,j,a):
    r = (j**(2*a-i))*(sc.special.gamma(2*a+1)/sc.special.gamma(2*a-i+1))
    return r

def B(i,j,b):
    r = (j**(2*b-i))*(sc.special.gamma(2*b+1)/sc.special.gamma(2*b-i+1))
    return r

def Amatrix(q1,q2,q3,a,b):
    return [[A(q1[0],q1[1],a), B(q1[0],q1[1],b) ,C(q1[0])],\
            [A(q2[0],q2[1],a), B(q2[0],q2[1],b) ,C(q2[0])],\
            [A(q3[0],q3[1],a), B(q3[0],q3[1],b) ,C(q3[0])]]

a = 0.75
b = 0.60
q1=[0,1]
q2=[0,2]
q3=[1,1]

Ak = np.linalg.cond(Amatrix(q1,q2,q3,a,b))
print(Ak)
```