Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Instituto de Matemática e Estatística

Jhoab Pessoa de Negreiros

Métodos de Diferenças Finitas para a Equação de Difusão Fracionária

Rio de Janeiro
2023

Jhoab Pessoa de Negreiros

Métodos de Diferenças Finitas para a Equação de Difusão Fracionária

Tese apresentada, como requisito parcial para obtenção do título de Doutor, ao Programa de Pós-Graduação em Ciências Computacionais e Modelagem Matemática, da Universidade do Estado do Rio de Janeiro.

Orientadores: Profª Dra. Cristiane Oliveira de Faria
Prof. Dr. Carlos Antonio de Moura

Rio de Janeiro
2023

CATALOGAÇÃO NA FONTE

UERJ/REDE SIRIUS/BIBLIOTECA CTC/A

N385 Negreiros, Jhoab Pessoa de.
Métodos de diferenças finitas para a equação de difusão fracionária/ Jhoab Pessoa de Negreiros. – 2023.
173 f.: il.

Orientadores: Cristiane Oliveira de Faria, Carlos Antonio de Moura
Tese (Doutorado em Ciências Computacionais e Modelagem Matemática) - Universidade do Estado do Rio de Janeiro, Instituto de Matemática e Estatística.

1. Diferenças finitas - Teses. 2. Derivadas (Matemática) - Teses. 3. Modelagem matemática - Teses. I. Faria, Cristiane Oliveira de. II. Moura, Carlos Antonio de. III. Universidade do Estado do Rio de Janeiro. Instituto de Matemática e Estatística. IV. Título.

CDU 517.962

Patricia Bello Meijinhos CRB7/5217 - Bibliotecária responsável pela elaboração da ficha catalográfica

Jhoab Pessoa de Negreiros

Métodos de Diferenças Finitas para a Equação de Difusão Fracionária

Tese apresentada, como requisito parcial para obtenção do título de Doutor, ao Programa de Pós-Graduação em Ciências Computacionais e Modelagem Matemática, da Universidade do Estado do Rio de Janeiro.

Aprovada em 18 de Dezembro de 2023.

Banca Examinadora:

Prof$^{\underline{a}}$ Dra. Cristiane Oliveira de Faria (Orientadora)
Instituto de Matemática e Estatística – UERJ

Prof. Dr. Carlos Antonio de Moura (Orientador)
Instituto de Matemática e Estatística – UERJ

Prof. Dr. Henrique Sarmento Malvar
Microsoft Research

Prof. Dr. Rubens de Figueiredo Camargo
Universidade Estadual Paulista

Prof. Dr. Vinícius Machado Martinez
Instituto Federal do Paraná

Prof. Dr. Hilbeth Parente Azikri de Deus
Universidade Tecnológica Federal do Paraná

Prof$^{\underline{a}}$ Dra. Patrícia Nunes da Silva
Instituto de Matemática e Estatística – UERJ

Rio de Janeiro
2023

DEDICATÓRIA

A Deus. À minha esposa, filho, família, professores e amigos, pela paciência, apoio e companheirismo.

AGRADECIMENTOS

A Deus pela proteção constante e por sempre guiar os meus passos!

À minha esposa Aline, que sempre me incentivou em meus projetos, colocando-se à disposição para me ajudar e me encorajar em todos os momentos..

Ao meu filho Davi Lucas, que, com seus nove anos de idade, demonstrava compreensão nos momentos em que eu precisava fechar a porta do escritório.

Aos meus pais, Antônio e Gizelda, por me criarem e educarem de forma muito sábia.

Agradeço de forma especial aos meus orientadores, a professora Cristiane Oliveira de Faria e o professor Carlos Antonio de Moura, que nunca mediram esforços para me ajudar e me encorajar, sempre muito dedicados e competentes, exemplos de professores.

Aos professores Rubens Camargo, Vinícius Martinez e Carlos Frederico, que compuseram a banca de qualificação e apresentaram valiosas observações acerca do trabalho.

A Gisele e a Giselma, das quais tenho a sorte de ser irmão.

À minha tia Cícera e à minha sogra Oliva, que nos momentos em que estiveram presentes, sempre nos ajudaram, principalmente no cuidado com o meu filho.

Ao meu amigo de orientação, Alisson Pinto, companheiro em tantas disciplinas e em momentos não triviais.

Ao meu ex-aluno e amigo Alessandro de Jesus, que me deu dicas valiosas na construção dos primeiros códigos do meu trabalho.

Aos meus colegas de trabalho Angelo Siqueira, Abel Lozano, Sérgio Ricardo, Tereza Canalli e Valessa Leal, que me incentivaram e motivaram de alguma forma a retornar aos estudos.

A todos os professores com quem tive contato desde a minha graduação até o último curso que concluí nesta pós-graduação.

À UERJ e ao programa de Pós-Graduação em Ciências Computacionais e Modelagem Matemática (PPG-CompMat) por proporcionarem que esse sonho fosse realizado.

Que darei eu ao Senhor, por todos os benefícios que me tem feito? (Sl 116, 12)

RESUMO

NEGREIROS, Jhoab Pessoa de. **Métodos de diferenças finitas para a equação de difusão fracionária**. 2023. 174 f. Tese (Doutorado em Ciências Computacionais e Modelagem Matemática) – Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2023.

Existe uma extensa gama de formulações que envolvem o termo derivada fracionária, e esse número continua aumentando. Isso se deve ao fato de que a derivada fracionária é um conceito matemático complexo e abrangente, que pode ser usado para descrever uma ampla gama de fenômenos. Considerando o crescente número de definições, este trabalho apresenta um estudo por meio dos métodos numéricos com seis derivadas fracionárias: Riemann-Liouville, Caputo, Chen, Katugampola, Caputo-Fabrizio e Atangana-Baleanu. Essas derivadas são aplicadas à equação da difusão fracionária em uma dimensão com coeficiente constante, onde a variação temporal fracionária é simulada no termo de difusão. A seleção desses operadores foi fundamentada na classificação realizada por Teodoro em sua tese de doutorado, a qual, a partir do critério proposto por Ortigueira e Machado em 2015 − composto por cinco propriedades que nos auxiliam a concluir quando um operador específico é uma derivada fracionária −, verificou se esses ou muitos outros operadores podem ser considerados derivadas fracionárias, de acordo com o referido critério. A abordagem numérica adotada é o método das diferenças finitas, empregando esquemas baseados nos métodos clássicos de Euler progressivo e regressivo. A análise numérica de estabilidade e convergência é realizada no método tipo Euler progressivo e regressivo com a derivada fracionária segundo Riemann-Liouville. O desbalanceamento gerado pela derivada fracionária na equação da difusão fracionária é corrigido pela inserção do parâmetro de correção dimensional τ no modelo. A contribuição central deste trabalho foi o desenvolvimento de métodos numéricos e a construção dos respectivos códigos para a difusão de ordem fracionária. Experimentos numéricos são apresentados, exibindo resultados com o objetivo de confirmar as conclusões teóricas, verificar a taxa de convergência, realizar comparações entre diferentes abordagens e compreender a importância da validade das propriedades do critério de Ortigueira e Machado para a modelagem. Algoritmos, juntamente com seus códigos em linguagem Python correspondentes, estão disponíveis.

Palavras-chave: Derivadas fracionárias. Equação da difusão fracionária. Equação da difusão fracionária. Critério de Ortigueira e Machado.

ABSTRACT

NEGREIROS, Jhoab Pessoa de. **Finite Difference Methods for Fractional Diffusion Equation**. 2023. 174 f. Tese (Doutorado em Ciências Computacionais e Modelagem Matemática) – Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2023.

There is an extensive range of formulations involving the term fractional derivative, and this number continues to increase. This is due to the fact that fractional derivative is a complex and comprehensive mathematical concept that can be used to describe a wide range of phenomena. Considering the growing number of definitions, this work presents a study using numerical methods with six fractional derivatives: Riemann-Liouville, Caputo, Chen, Katugampola, Caputo-Fabrizio, and Atangana-Baleanu. These derivatives are applied to the fractional diffusion equation in one dimension with a constant coefficient, where the fractional temporal variation is simulated in the diffusion term. The selection of these operators was based on the classification carried out by Teodoro in his doctoral thesis, which, based on the criterion proposed by Ortigueira and Machado in 2015 − composed of five properties that help us conclude when a specific operator is a fractional derivative − verified whether these or many other operators can be considered fractional derivatives, according to the aforementioned criterion. The numerical approach adopted is the finite difference method, employing schemes based on the classical progressive and regressive Euler methods. The numerical analysis of stability and convergence is carried out using the forward and backward Euler-type method with the fractional derivative according to Riemann-Liouville. The imbalance generated by the fractional derivative in the fractional diffusion equation is corrected by inserting the dimensional correction parameter τ into the model. The central contribution of this work was the development of numerical methods and the construction of respective codes for fractional-order diffusion. Numerical experiments are presented, displaying results with the aim of confirming theoretical conclusions, verifying the convergence rate, making comparisons between different approaches, and understanding the importance of the validity of the properties of the Ortigueira and Machado criterion for modeling. Algorithms, along with their corresponding Python language codes, are available.

Keywords: Fractional derivatives. Fractional diffusion equation. Finite difference approximations. Ortigueira and Machado criterion.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

No século XVII, de forma independente, Leibniz e Newton desenvolveram o cálculo diferencial de ordem inteira. Esse ramo da matemática, inspirado pelo estudo de fenômenos que envolvem o movimento de corpos e sua variação, desenvolveu-se e alcançou um rigor suficiente já no século XIX. Durante este mesmo período, o chamado cálculo de ordem não inteira, ou simplesmente fracionária, teve origem na correspondência trocada em 1695 entre l'Hôpital e Leibniz, na qual o primeiro pergunta: "Qual é o significado da derivada de ordem 1/2?". Leibniz respondeu: "Um aparente paradoxo, do qual algum dia, consequências úteis poderão ser feitas".

O desenvolvimento das bases do cálculo fracionário se deu de forma gradual e contou com a colaboração de grandes nomes da matemática: Euler e Lagrange no século XVIII, Laplace, Fourier, Abel, no século XIX. Outros matemáticos e físicos contribuíram para o crescimento do cálculo fracionário. Contudo, foram três conferências internacionais que marcaram a popularização e divulgação do cálculo fracionário e suas aplicações. Ross organizou em 1974 a primeira nos Estados Unidos (Universidade de New Haven) e a segunda em 1984, na Escócia (Universidade de Stratchclyde Glasgow). Após cinco anos, em 1989, no Japão (Universidade de Nihon), ocorreu a terceira (TEODORO; OLIVEIRA; OLIVEIRA, 2017).

Nas últimas décadas, a continuidade e expansão dos estudos do cálculo fracionário foram alavancadas pela contribuição de muitos grupos de pesquisa liderados por renomados cientistas, dentre os quais mencionamos: Baleanu, Ortigueira, Mainardi, Tenreiro Machado, Caputo, dentre outros. No Brasil, podemos mencionar o grupo de físicos liderado por Lenzi, no Paraná e também se destacam os estudos realizados pelo professor Edmundo Capelas e seus coautores (CAMARGO; OLIVEIRA, 2015).

Os avanços no cálculo fracionário geraram um crescimento nas aplicações em diversas áreas da ciência, com destaque na abordagem de problemas que envolvem os conceitos de não localidade e efeito de memória, dos quais uma parte não pode ser modelada com simplicidade pelo cálculo diferencial de ordem inteira (MAINARDI, 2022). O sucesso nas aplicações tornou o cálculo fracionário mais relevante, o que acelerou seu desenvolvimento. O progresso no campo teórico, acompanhado de um número crescente de formulações para a derivada fracionária, nos levou a perguntar: quais desses operadores podem ser classificados como derivada fracionária?

Em 1974, Ross propôs um critério, (ROSS, 1975), composto por cinco propriedades que um operador deve satisfazer para ser chamado de derivada fracionária, a saber:

1 A derivada fracionária de uma função analítica[1] é analítica;

2 A derivação fracionária, quando a ordem é um inteiro positivo n, $n \in \mathbb{N}$, deve produzir o mesmo resultado da n-ésima derivação ordinária, ou seja,

$$D^n f(x) = \frac{d^n f(x)}{dx^n}$$

e quando é um inteiro negativo $-n$, $n \in \mathbb{N}$, deve produzir o mesmo resultado da repetição n-ésima da integração ordinária, ou seja,

$$D^{-n} f(x) = \int_0^x \int_0^{\xi_{n-1}} \dots \int_0^{\xi_1} f(\xi) d\xi d\xi_1 \dots d\xi_{n-1};$$

3 A derivada de ordem zero de uma função é a própria função $D^0 f(x) = f(x)$;

4 A derivada fracionária é um operador linear;

5 A lei dos expoentes[2], $D^\alpha D^\beta f(x) = D^{\alpha+\beta} f(x)$, é satisfeita para $\alpha < 0$ e $\beta < 0$;

Em 2015, Ortigueira e Machado reformularam esse critério (ORTIGUEIRA; MACHADO, 2015), tendo em vista a necessidade de uma derivada fracionária satisfazer a generalização da regra de Leibniz. As cinco propriedades propostas são:

1 A derivada fracionária é um operador linear;

2 A derivada de ordem zero de uma função é a própria função $D^0 f(x) = f(x)$;

3 A derivação fracionária, quando a ordem é um inteiro deve produzir o mesmo resultado da derivação ordinária.

4 A lei dos expoentes, $D^\alpha D^\beta f(x) = D^{\alpha+\beta} f(x)$, é satisfeita para $\alpha < 0$ e $\beta < 0$;

5 Vale a generalização da regra de Leibniz, a saber,

$$D^\alpha \left(f(x)g(x)\right) = \sum_{k=0}^{\infty} \binom{\alpha}{k} D^k f(x) D^{\alpha-k} g(x),$$

sendo

$$\binom{\alpha}{k} = \frac{\Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-k+1)}.$$

[1] Uma função analítica é uma função que pode ser localmente expandida em série de Taylor

[2] A lei dos expoentes também é conhecida na literatura como propriedade de semigrupo.

Teodoro (TEODORO, 2019), em sua tese de doutorado, classificou os operadores mais estabelecidos e encontrados na literatura em três classes distintas, a saber, derivadas clássicas, derivadas locais e derivadas com núcleo não singular. Em seguida foi verificado se cada um desses operadores pode ser classificado como uma derivada fracionária, segundo o critério de Ortigueira e Machado[3].

Os resultados obtidos foram que os operadores da primeira classe cumprem as propriedades do critério de Ortigueira e Machado. Portanto, esses operadores podem ser chamados de derivadas fracionárias. Na classe das derivadas locais, nenhuma derivada analisada satisfaz as propriedades referentes à lei dos expoentes e à generalização da regra de Leibniz. Em vista disso, essa classe não pode ser considerada derivada fracionária segundo tal critério. As derivadas da terceira classe, as de núcleo não singular, também não podem ser chamadas de fracionárias segundo o critério de Ortigueira e Machado, pois nenhuma delas satisfaz à generalização da regra de Leibniz. Contudo, neste trabalho, manteremos a nomenclatura derivada fracionária para todos os operadores de ordem arbitrária que mencionaremos.

Considerando os diversos problemas que são estudados com as ferramentas do cálculo fracionário, destaca-se o da difusão anômala, que é, em algum sentido, uma generalização da difusão usual aproximada pela equação clássica da difusão, muitas vezes chamada de equação do calor. A generalização da equação da difusão fracionária pelo uso das derivadas de ordem fracionária tem sido eficaz para modelar a difusão em sistemas anômalos, tais como: a difusão em transporte de fluidos em meios porosos (SPOHN, 1993), difusão em plasmas (BERRYMAN, 1977), difusão em fractais (STEPHENSON, 1995), entre outros.

Variações da equação da difusão com derivadas fracionárias aplicadas relativamente à variável temporal e ou espacial têm sido empregadas em diferentes problemas (FARIA; MOURA; NEGREIROS, 2023). Uma versão da equação da difusão fracionária com a derivada temporal não inteira foi utilizada com relativo sucesso no problema do transporte não fickiano em meios porosos (ZHOKH; STRIZHAK, 2018), contaminante de fratura em matriz de rocha porosa (FOMIN; CHUGUNOV; HASHIDA, 2010), dentre outros. Todavia, nesta pesquisa, vamos considerar a equação da difusão com derivada fracionária temporal aplicada no termo de difusão, conforme proposto em dois artigos de revisão por Metzler e Klafter (METZLER; KLAFTER, 2000; METZLER; KLAFTER, 2004).

Aqui, a partir da classificação dos operadores fracionários realizada por (TEODORO, 2019), escolhemos seis. Na classe das derivadas clássicas, os operadores escolhidos foram o de Riemann-Liouville e Caputo. Nas derivadas locais, escolhemos Chen e

[3] Segundo (TEODORO, 2019) a escolha desse critério se justifica por ser mais restritivo que o proposto por Ross.

Katugampola, e na classe dos operadores com núcleo não singular, escolhemos a derivada fracionária proposta por Caputo-Fabrizio e Atangana-Baleanu. Esses seis operadores foram aplicados à equação da difusão fracionária em um problema de valor inicial e de fronteira. A derivada fracionária considerada é a relativa à variável temporal, com ordem $\alpha \in \mathbb{R}$, $0 < \alpha < 1$, como exigido pela versão da equação da difusão fracionária estudada.

Com o desenvolvimento e os avanços nas aplicações do cálculo fracionário, surge um problema específico na área: a carência de métodos numéricos. Seguindo nessa direção, a fim de contribuir para esse déficit, esta pesquisa realizará o estudo da equação da difusão fracionária com diferentes operadores, adotando uma abordagem numérica. Então, os estudos realizados neste trabalho são de alguma forma uma extensão natural de trabalhos que utilizaram a clássica equação da difusão de ordem inteira e estudaram a partir delas o problema da difusão usual, com os métodos numéricos.

Começamos escolhendo a abordagem numérica, que consiste no método das diferenças finitas. A partir dessa escolha e da formulação da equação da difusão fracionária, torna-se necessário obter as aproximações para as derivadas, tanto para as inteiras quanto para as fracionárias. No caso das derivadas inteiras, são aplicadas as aproximações clássicas: diferenças avançadas e recuadas relativas à variável temporal com ordem um, e diferenças centradas relativa à variável espacial com ordem dois. Por outro lado, cada uma das seis derivadas fracionárias é aproximada por diferentes esquemas numéricos. As derivadas clássicas de Riemann-Liouville e Caputo são aproximadas pelo operador de Grünwald-Letnikov (BALEANU et al., 2012) e pelo método denominado L_1 (OLDHAM; SPANIER, 1974), respectivamente. Os operadores de núcleo não singular foram aproximados por esquemas relativamente recentes. (QURESHI; RANGAIG; BALEANU, 2019) propuseram um esquema para a derivada de Caputo-Fabrizio e (YADAV; PANDEY; SHUKLA, 2019) propuseram um esquema para a derivada de Atangana-Baleanu. As derivadas fracionárias de Chen e de Katugampola, devido à sua natureza local, permitiram que suas aproximações fossem propostas a partir de adaptações das diferenças regressivas. A partir dessas aproximações numéricas, propusemos esquemas baseados nos métodos de Euler progressivo e regressivo.

Tendo em vista o objetivo principal deste trabalho, que consiste em uma abordagem numérica da equação da difusão fracionária com diferentes operadores não inteiros, esse trabalho está disposto da seguinte maneira: no Capítulo 1, realizamos uma revisão sobre os fenômenos difusivos e a equação de difusão fracionária, abordando tanto a sua versão clássica de ordem inteira quanto a fracionária. O Capítulo 2 é dedicado aos conceitos preliminares que surgem ao longo do trabalho. Iniciamos com a definição das funções Gama e de Mittag-Leffler; em seguida, apresentamos a derivada fracionária de Grünwald-Letnikov. Prosseguimos desenvolvendo os principais conceitos das aproximações por diferenças finitas e finalizamos explorando os conceitos fundamentais de consistência, estabilidade e convergência. No Capítulo 3, as derivadas clássicas de Riemann-Liouville e de Caputo são o objeto de estudo. Essas derivadas são aplicadas à equação da difusão fracionária. Para a derivada de Riemann-Liouville, propomos métodos do tipo Euler progressivos e regressivos, e estudamos sua estabilidade e convergência. Para a derivada de Caputo, aplicamos o método do tipo Euler regressivo. Os esquemas numéricos são apresentados, assim como seus respectivos algoritmos, e conduzimos experimentos numéricos. Nos Capítulos 4 e 5, aplicamos as derivadas locais de Chen e de Katugampola, bem como as derivadas de núcleo não singular de Caputo-Fabrizio e Atangana-Baleanu no problema da difusão fracionária. Os estudos nestes capítulos seguiram a construção dos esquemas numéricos do tipo Euler regressivo. Em seguida, apresentamos seus algoritmos e finalizamos realizando experimentos com os métodos.

E por fim, no Capítulo 6 apresentamos resultados numéricos que são obtidos dos esquemas propostos nos três capítulos anteriores. Foram realizadas comparações entre os resultados numéricos da equação da difusão fracionária para os diferentes operadores, analisando os efeitos ocasionados pelo emprego do parâmetro de correção dimensional τ no balanceamento da equação e a influência da ordem da derivada fracionária α, bem como sua correspondência com as propriedades do critério de Ortigueira e Machado. Também realizamos um experimento para medir o tempo de execução dos códigos propostos no trabalho. As ordens de convergência obtidas pelos métodos numéricos desenvolvidos estiveram em concordância com as aproximações empregadas. No Apêndice B disponibilizaremos os códigos em linguagem Python dos métodos numéricos propostos no trabalho.

1 EQUAÇÃO DE DIFUSÃO CLÁSSICA E FRACIONÁRIA

Este capítulo será dedicado ao estudo de dois aspectos fundamentais na compreensão dos processos de difusão: a difusão clássica, que é um fenômeno próprio da natureza, aproximado por modelos matemáticos consolidados; e a difusão anômala, um fenômeno subdifusivo ou superdifusivo que tem sido estudado com relativo sucesso por meio de modelos fracionários.

1.1 Difusão clássica e seus modelos matemáticos

A palavra difusão origina-se do latim *diffundare*, composta pelo prefixo *dif* (separar, em todas as direções) mais *fundere* (derramar, espalhar). Portanto, é um termo que significa espalhar-se, esse é um termo muito abrangente e por isso, alguns significados diferentes são esperados a depender do contexto. Contudo, todos possuem um sentido em comum, o movimento de elementos de um lugar de alta concentração para um de menor, isto é, um fenômeno comum na natureza e ocorre, em geral, quando um sistema encaminha-se para o estado de equilíbrio.

A compreensão da difusão tem despertado interesse em função de um grande número de fenômenos relacionados e ligados à biologia, como a difusão de gases nos pulmões e a difusão de substâncias nas células, à engenharia, como a difusão de calor em materiais e a difusão de gases em materiais porosos, à física, exemplificada pela difusão de partículas em um fluido e pela difusão de luz, e até às ciências econômicas, onde observamos a difusão de inovações e a difusão de informações no mercado financeiro. O matemático francês Joseph Fourier conhecido pelas inúmeras contribuições à física matemática, das quais podemos destacar as séries de Fourier, a análise de Fourier, os coeficientes de Fourier e a transformada de Fourier, foi também um dos precursores nos estudos sobre a difusividade e calor em sólidos. Em 1807, um trabalho intitulado *Théorie de la Propagation de la Chaleur dans les Solides*[4] foi submetido à Academia Francesa e teve sua primeira publicação em 1822 (FOURIER, 2009). Neste trabalho Fourier descreve o processo transiente da condução de calor em termos de uma equação diferencial parcial parabólica, que de acordo com a notação por ele utilizada, é dada por

$$\frac{\partial \nu}{\partial t} = \frac{K}{CD}\left(\frac{\partial^2 \nu}{\partial x^2} + \frac{\partial^2 \nu}{\partial y^2} + \frac{\partial^2 \nu}{\partial z^2}\right) \tag{1}$$

[4] Teoria de Propagação de Calor em Sólidos

em que ν é a temperatura do objeto sólido em análise, t o tempo, K a condutividade térmica intrínseca do material, C o calor específico do material, D a densidade do sólido e x, y e z são as coordenadas espaciais

As obras publicadas por Fourier tiveram influência no desenvolvimento de diferentes trabalhos (NARASIMHAN, 1999). O químico Thomas Graham, produz uma série de trabalhos em difusão de gases e em sais em líquidos que posteriormente são as bases de estudos de Fick, que compõem as leis fenomenológicas da difusão.

Em 1855, o fisiologista alemão Adolf Eugen Fick publicou seu primeiro artigo sobre ciências físicas, *Über Diffusion*[5] (PICK, 1855). Neste artigo é apresentada a formulação matemática utilizada para descrever a difusão, que hoje é conhecida como a primeira lei de Fick, a saber,

> 1ª Lei de Fick - A taxa de difusão para espécies químicas em uma solução aumenta com a diferença na concentração entre duas regiões adjacentes. Essa diferença atua como uma força motriz para o movimento espontâneo das partículas do soluto na direção da região de menor concentração.

No processo de difusão, o fluxo de concentração expressa a quantidade de substância que passa por uma seção transversal de área unitária perpendicular ao movimento em um certo intervalo de tempo, chamada de densidade de corrente, e esta é igual à taxa de transferência de concentração multiplicada por um constante de proporcionalidade. A formulação matemática desta lei em uma dimensão pode ser escrita como

$$J = -\mathcal{D}\frac{\partial \rho}{\partial x}, \tag{2}$$

em que J é o fluxo de concentração, $\mathcal{D}$ é o coeficiente de difusão, que depende das propriedades do meio, ρ é a concentração da solução em questão e o sinal negativo indica que o fluxo se dá no sentido contrário da taxa de crescimento da concentração. A equação (2) é escrita no espaço tridimensional como

$$\boldsymbol{J} = -\mathcal{D}\nabla\rho, \tag{3}$$

com $\rho = \rho(\boldsymbol{r}, t)$ e $\boldsymbol{r} = (x_1, x_2, x_3)$. Os estudos realizados por Fick são sob condições ideais, isto é, despreza forças externas que poderiam atuar no sistema, por exemplo, a da gravidade.

A concentração de partículas em um determinado sistema, em uma abordagem simplificada, considera que a substância difundida não é nem absorvida nem emitida pelo

[5] Sobre a difusão

meio, dessa forma, surge uma lei de conservação que implica na equação de continuidade

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \boldsymbol{J} = 0, \tag{4}$$

em que t é o tempo e $\boldsymbol{J}$ é o fluxo da quantidade física ρ. Combinando a equação de continuidade com a equação (3), obtemos a conhecida equação da difusão

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (\mathcal{D} \nabla \rho). \tag{5}$$

Para o caso em que o número de partículas não é conservado no sistema, ou seja, em que partículas são inseridas ou retiradas do sistema a equação de continuidade possui um termo não homogêneo, a saber,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \boldsymbol{J} = \delta, \tag{6}$$

em que δ é a densidade da fonte. Se $\delta > 0$, são criadas ou emitidas partículas para o sistema e este termo fonte representa uma fonte. Se $\delta < 0$, são destruídas ou absorvidas partículas para o sistema e este termo fonte representa um sumidouro. Neste caso, a equação de difusão assume a seguinte forma

$$\frac{\partial \rho}{\partial t} = \nabla \cdot (\mathcal{D} \nabla \rho) + \delta. \tag{7}$$

Se se considerar o coeficiente de difusão D como constante e aplicar o divergente às equações (5) e (7), tem-se

$$\frac{\partial \rho}{\partial t} = \mathcal{D} \nabla^2 \rho \tag{8a}$$

$$\frac{\partial \rho}{\partial t} = \mathcal{D} \nabla^2 \rho + \delta, \tag{8b}$$

sendo (8a) a equação da difusão sem fonte e (8b) com um termo fonte sumidouro.

1.2 Difusão anômala e seu modelo matemático

A medida da capacidade de difusão das partículas em um sistema pode ser quantificada pela média dos quadrados dos deslocamentos em relação a uma posição anterior de referência de cada partícula. Esse processo usado para estudar a difusão das partículas é chamado de deslocamento quadrático médio MSD[6] ou segundo momento da distribuição, ou ainda variância, que é uma função do tempo, a saber,

$$\langle x(t)^2 \rangle := \frac{1}{N} \sum_{i=1}^{N} |\boldsymbol{x}_i(t) - \boldsymbol{x}_{\boldsymbol{i}}(0)|^2 \tag{9}$$

onde N é o número de partículas, $\boldsymbol{x}_{\boldsymbol{i}}(0)$ é o vetor posição de referência da i-ésima partícula e $\boldsymbol{x}_i(t)$ é o vetor posição da i-ésima partícula no tempo[7] t.

A difusão usual vista na Seção 1.1 é qualquer processo difusivo em que o deslocamento quadrático médio evolui de forma linear no tempo, isto é, quando é proporcional a t,

$$\langle x(t)^2 \rangle \propto t^{\gamma} \tag{10}$$

com $\gamma = 1$. Quando $\gamma \neq 1$, tem-se a difusão anômala, que pode ser classificada em relação a sua natureza, como subdifusiva ou superdifusiva. Para $0 < \gamma < 1$ em (10), o processo é subdifusivo e superdifusivo quando $\gamma > 1$. A Figura 1 exibe o gráfico de três regimes de difusão em que o desvio médio quadrático $\langle x(t)^2 \rangle$ está em função do tempo reduzido[8] t.

A difusão anômala está associada diretamente ao trabalho realizado pelo matemático e físico britânico Lewis Fry Richardson. Em 1926, ele escreveu um artigo intitulado *Atmospheric Diffusion shown on a Distance Neighbour Graph* (RICHARDSON, 1926). Esse trabalho sobre difusão turbulenta é considerado um marco dos fenômenos turbulentos e na difusão anômala. Os experimentos conduzidos por Richardson visavam estudar o comportamento do coeficiente de difusão K de diferentes materiais (RICHARDSON, 1926). Ele mostra que a equação de difusão de Fick,

$$\frac{\partial \nu}{\partial t} = K \frac{\partial^2 \nu}{\partial x^2} \tag{11}$$

onde ν é a concentração do número de partículas por comprimento, não é adequada para aproximar a difusão das correntes turbulentas da atmosfera.

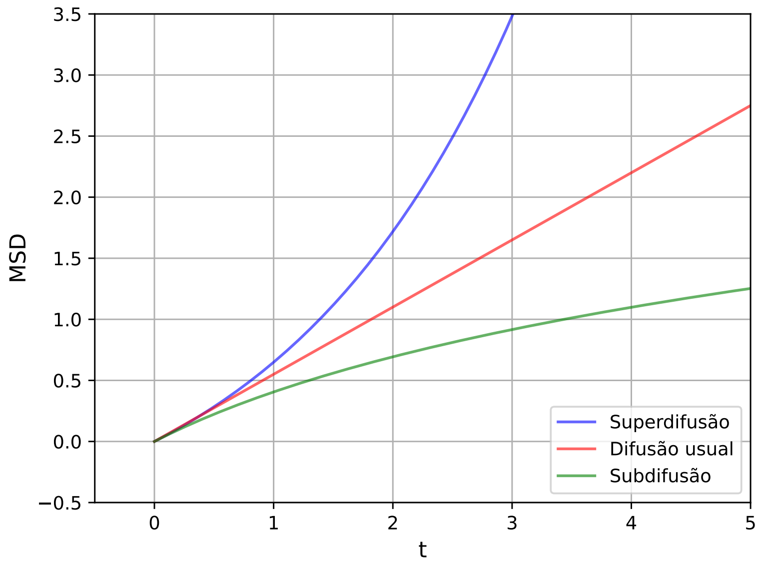
Tomando como base (11), Richardson propõe ao invés de considerar a concentração ν da substância que está sendo difundida como função da posição. Ele utiliza o número de vizinhos por comprimento, q, como função da distância l entre eles, o que resultou na

[6] Do inglês *Mean square displacement.*

[7] A notação " := " indica que (o termo que compõe) o primeiro membro é definido pela expressão no segundo – conforme adotado em algumas linguagens de programação.

[8] O termo "tempo reduzido" é comumente usado em contextos como dinâmica de fluidos, análise de transporte de partículas, entre outros, onde a escala temporal é normalizada.

Figura 1 - Gráfico da difusão usual e anômala.



Legenda: Deslocamento quadrático médio MSD em função do tempo t para difusão usual e anômala.

Fonte: O autor, 2023.

equação chamada de *Non-Fickian Diffusion*

$$\frac{\partial q}{\partial t} = \frac{\partial}{\partial l}\left(F(l)\frac{\partial q}{\partial l}\right) \tag{12}$$

sendo $F(l)$ uma função crescente de l.

Em seu trabalho, Richardson aproxima o valor de $F(l)$ com a construção de gráficos dos respectivos logaritmos da difusividade K e da distância de separação l a partir dos dados experimentais disponíveis na literatura e também próprios. Ajustando os dados com o valor da difusividade $K = 0,2l^{4/3}$, chegou à equação

$$\frac{\partial q}{\partial t} = \varepsilon\frac{\partial}{\partial l}\left(l^{4/3}\frac{\partial q}{\partial l}\right) \tag{13}$$

sendo a constante ε da ordem de $0,4\text{cm}^{2/3}\text{s}^{-1}$. A solução encontrada por Richardson para essa equação foi

$$q = A(4t\varepsilon/9)^{-3/2}\,\mathrm{e}^{-\frac{\alpha^2}{4t\varepsilon/9}}, \tag{14}$$

em que A é uma constante independente de t e de α. Em uma parte do trabalho, Richardson se dedicou a calcular os momentos da distribuição. Como ela é par, os momentos de ordem ímpar são nulos e o resultado do segundo momento obtido é $\langle x(t)^2\rangle = \frac{105}{16}(4t\varepsilon/9)^3$, que é a variância. O resultado que Richardson alcançou para a difusão em correntes turbulentas de ar é proporcional a t^3, o que representa uma difusão que ocorre de forma muito mais rápida quando comparada àquela obtida por Einstein em 1905, que se refere ao movimento irregular de partículas suspensas em um líquido proporcional ao tempo, que é o caso normal. Por isso, dizemos que a difusão nesse sistema é anômala do tipo superdifusivo.

Após os estudos de Richardson sobre difusão turbulenta, houve um período de três décadas com poucas publicações sobre o assunto. No entanto, desde então, a difusão anômala tem sido estudada na teoria de transporte. O físico Harvey Scher e o matemático Elliot W. Montroll ficaram conhecidos pelo estudo dos fenômenos subdifusivos, no qual buscavam desenvolver um modelo adequado para explicar o comportamento anômalo de transporte em materiais sólidos amorfos[9] que apresentavam difusão não usual.

Scher e Montroll trabalhavam para a corporação *Xerox* e, por isso, fizeram trabalhos experimentais com materiais amorfos que visavam o desenvolvimento de novas máquinas fotocopiadoras. Os pesquisadores deram uma grande contribuição para resolver o problema da carência de modelos teóricos, quando publicaram o artigo *Anomalous transit-time dispersion in amorphous solids* (SCHER; MONTROLL, 1975).

A equação da difusão fracionária pode ser obtida por meio da caminhada aleatória contínua no tempo. Esse tipo de caminhada aleatória está associada a uma grande quantidade de sistemas em diversas áreas, como o estudo do fluxo através de meios porosos (SCHEIDEGGER, 1958), sistemas relacionados a gases unidimensionais (HARRIS; SELLS; GUTH, 1953) e a dinâmica da difusão em cristais (RICE; FRISCH, 1960), entre outros.

O modelo de caminhada aleatória contínua no tempo (CTRW[10]) tem como base a ideia de que o comprimento de um passo dado pela partícula, assim como o intervalo de tempo entre dois passos, não são constantes e estão associados a algum tipo de função de probabilidade (PDF[11]). A caminhada aleatória é considerada um modelo aparentemente simples, porém possui inúmeras aplicações em modelos matemáticos, físicos e biológicos (CARVER; O'TOOLE; RAIFORD, 1930). A partir dessa ideia central, é possível chegar à equação da difusão fracionária, como é feito por Denner em sua dissertação de mestrado (VIEIRA, 2015). Nela, ele parte da análise da caminhada aleatória contínua no tempo e mostra como é possível derivar equações diferenciais de difusão com existência de operadores fracionários.

---

[9] Sólidos amorfos, também chamados de materiais vítreos ou não cristalinos, são materiais que não possuem uma estrutura cristalina bem definida. Tais materiais são compostos por átomos ou moléculas que estão dispostos de maneira desordenada, sem uma repetição regular de padrões.

[10] Do inglês *Continuous Time Random Walk.*

[11] Do inglês *Probability Distribution Function.*

O problema de valores iniciais e de contorno com a condição de Dirichlet[12] da equação da difusão fracionária consiste em determinar a distribuição de temperatura, a função $u(x,t)$, tal que

$$\frac{\partial u(x,t)}{\partial t} = D_{0,t}^{\alpha}\left(\mathcal{K}\tau^{\alpha}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T, \tag{15a}$$

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L, \tag{15b}$$

$$u(0,t) = l(t), \quad 0 \leq t \leq T, \tag{15c}$$

$$u(L,t) = r(t), \quad 0 \leq t \leq T. \tag{15d}$$

A equação (15a) foi proposta por Metzler e Klafter 2000, na qual $D_{0,t}^{\alpha}$ denota uma derivada fracionária temporal não local de ordem α, com $0 < \alpha < 1$ e $\phi(x)$, $l(t)$, $r(t)$ e $f(x,t)$ são funções suficientemente regulares dadas. A constante $\mathcal{K}$ é conhecida como difusividade térmica, e esse parâmetro $\mathcal{K}$ depende da condutividade térmica κ, da densidade ρ e do calor específico do material s, isto é, $K = \kappa/\rho s$ que tem dimensão $[\mathcal{K}] = x^2/t$ (CANNON, 1984).

A adequação da equação da difusão de ordem inteira para a de ordem arbitrária gera um desbalanceamento dimensional em (15a). Assim, a partir de (GÓMEZ-AGUILAR et al., 2012) e (GÓMEZ-AGUILAR; RAZO-HERNÁNDEZ; GRANADOS-LIEBERMAN, 2014) aplica-se um novo parâmetro τ da seguinte maneira

$$\frac{1}{\tau^{-\alpha}}\left[\frac{d^{\alpha}}{dt^{\alpha}}\right] = \frac{1}{\tau^{-\alpha}}\frac{1}{t^{\alpha}}, \tag{16}$$

sendo τ um parâmetro cuja unidade é tempo, de modo que a equação fracionária (15a) preserve a consistência da dimensão. Dessa forma, denominamos $\tau^{\alpha}\mathcal{K}$ como o coeficiente da difusão fracionária.

[12] A condição de contorno de Dirichlet especifica o valor que a solução u deve tomar em todos os pontos da fronteira do domínio.

2 CONCEITOS PRELIMINARES

Neste capítulo, apresentamos definições fundamentais para a compreensão do trabalho. Iniciamos com a função gama e a função de Mittag-Leffler, que surgem em algumas definições no cálculo fracionário. A seguir, é apresentada a formulação da derivada fracionária de Grünwald-Letnikov, seguida dos conceitos e alguns esquemas dos métodos das diferenças finitas. Finalizamos abordando as noções de consistência, estabilidade e convergência.

2.1 Função Gama

A função gama, também conhecida como função fatorial, desempenha um importante papel em algumas definições no estudo do cálculo fracionário, por estender a operação fatorial para um número não inteiro.

Definição 1. *A função gama é definida pela integral imprópria*

$$\Gamma(z) := \int_0^\infty e^{-t} t^{z-1} dt. \tag{17}$$

com $Re(z) > 0$*.*

A exponencial garante a convergência da integral na vizinhança do infinito; já a convergência em zero ocorre para qualquer que seja o número complexo z, com $\mathrm{Re}(z) > 0$, ou seja, no semiplano direito do plano complexo. Na outra porção do plano complexo a generalização pode ser vista pela definição da função gama com limite

$$\Gamma(z) := \lim_{n\to\infty} \frac{n! n^z}{z(z+1)...(z+n)}. \tag{18}$$

Um caminho para chegar à definição da função gama apresentada acima a partir de (17) passa pela substituição de e^{-t} pelo limite

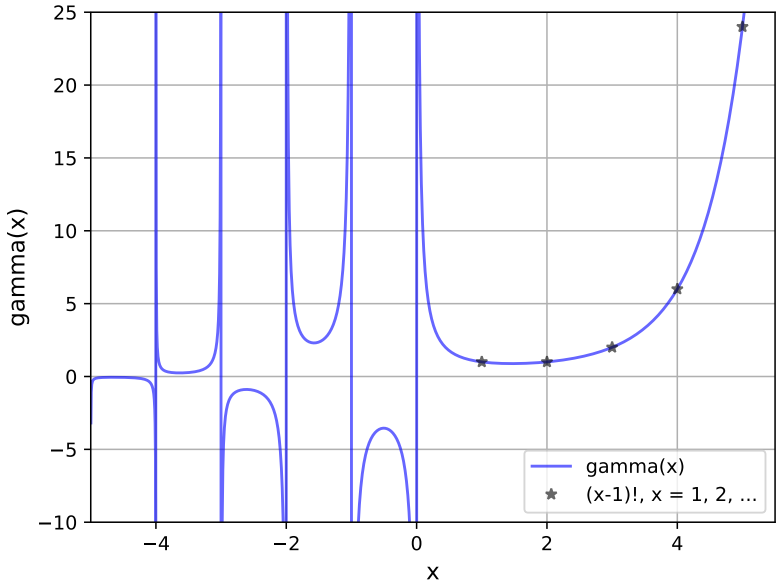
$$\lim_{n\to\infty} \left(1 - \frac{t}{n}\right)^n = \mathrm{e}^{-t} \tag{19}$$

e a realização de n integrações por partes (PODLUBNY, 2002).

Como exibido na Figura 2 os cálculos para valores com parte real negativa são possíveis, exceto para os polos da função gama que são os números inteiros negativos, isto é, $z \in \mathbb{C} - \mathbb{Z}_-$.

Uma propriedade importante para a função gama é $\Gamma(z+1) = z\Gamma(z)$, a qual é

Figura 2 - Função gama.



Legenda: Gráfico da função gama $\Gamma(x)$ no domínio $-5 < x < 5$ e fatorial de $(x-1)$.
Fonte: O autor, 2023.

muito usada na simplificação de alguns resultados. Uma vez que $\Gamma(1) = 1$, para $z = n \in \mathbb{N}$ temos, $\Gamma(n) = (n-1)!$.

Uma outra função cujo conhecimento é indispensável no estudo de cálculo fracionário é a função beta, muito utilizada para calcular derivadas fracionárias de potências de funções.

Definição 2. *Sejam $p > 0$ e $q > 0$, a função beta de parâmetros p e q é dado por*

$$B(p,q) = \int_0^1 t^{p-1}(1-t)^{q-1}dt. \tag{20}$$

A relação entre a função beta e a função gama, a saber,

$$B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}, \tag{21}$$

é muito útil, pois em diversas situações nos permite realizar cancelamentos e simplificações algébricas.

2.2 Função de Mittag-Leffler

Em 1903, Mittag-Leffler propôs a generalização para a função exponencial (MOSCHEN, 2006), e hoje essa função é conhecida em sua homenagem por função de Mittag-Lefller. Vejamos agora a definição devida a Mittag-Leffler.

Definição 3 (Função de Mittag-Leffler de um parâmetro)**.** *A função de Mittag-Leffler é dada pela série*

$$E_a(z) = \sum_{n=0}^{\infty} \frac{z^n}{\Gamma(an+1)} \tag{22}$$

onde a é o parâmetro da função e z é a variável de interesse.

A função $E_a(z)$ é uma generalização da função exponencial, pois quando consideramos $a = 1$ temos,

$$E_1(z) = \sum_{n=0}^{\infty} \frac{z^n}{\Gamma(n+1)} = \sum_{n=0}^{\infty} \frac{z^n}{n!} = e^z$$

Em 1905, Wiman (ANDERS, 1905) propôs e estudou uma generalização da função de Mittag-Leffler, que hoje é conhecida com função de Mittag-Leffler com dois parâmetros.

Definição 4 (Função de Mittag-Leffler de dois parâmetros)**.** *A função de Mittag-Leffler de dois parâmetros é dada pela série*

$$E_{a,b}(z) = \sum_{n=0}^{\infty} \frac{z^n}{\Gamma(an+b)} \tag{23}$$

onde, a e b são os parâmetros da função e z é a variável de interesse.

Note que quando $b = 1$ na Equação (24) recuperamos a função de Mittag-Leffler de um parâmetro. De fato,

$$E_{a,1}(z) = \sum_{n=0}^{\infty} \frac{z^n}{\Gamma(an+1)} = E_a(z).$$

Em 1971, Prabhakar (PRABHAKAR et al., 1971) introduziu a função de Mittag-Leffler com três parâmetros. Antes de apresentar sua definição vamos definir o símbolo de Pochhammer.

Definição 5 (Símbolo de Pochhammer)**.** *O símbolo de Pochhammer é definido por:*

$$(\gamma)_n := \begin{cases} 1, & para \quad n = 0; \\ \gamma(\gamma+1)\ldots(\gamma+n-1), & para \quad n \in \mathbb{N}. \end{cases}$$

O símbolo de Pochhammer pode ser escrito em termo da função gama, isto é,

$$(\gamma)_n = \alpha(\gamma + 1) \dots (\gamma + n - 1) = \frac{\Gamma(\gamma + n)}{\Gamma(\gamma)},$$

note que neste caso $\gamma \in \mathbb{C} - \mathbb{Z}_-$.

Definição 6 (Função de Mittag-Leffler de três parâmetros)**.** *A função de Mittag-Leffler de três parâmetros é dada pela série*

$$E^{\rho}_{a,b}(z) = \sum_{n=0}^{\infty} \frac{(\rho)_n z^n}{\Gamma(an + b)n!} \tag{24}$$

onde $(\rho)_n$ *é o símbolo de Pochhammer.*

A função de Mittag-Leffler de três parâmetros é uma generalização das funções de Mittag-Leffler de um e de dois parâmetros, visto que, ao fixarmos o valor de um dos parâmetros como $\rho = 1$ na função de Mittag-Leffler de três parâmetros, obtemos

$$E^{1}_{a,b} = \sum_{n=0}^{\infty} \frac{(1)_n z^n}{\Gamma(an + b)n!} = \sum_{n=0}^{\infty} \frac{n! z^n}{\Gamma(an + b)n!} = \sum_{n=0}^{\infty} \frac{z^n}{\Gamma(an + b)} = E_{a,b}(z).$$

As funções de Mittag-Leffler desempenham um papel muito importante no cálculo fracionário, pois permitem a identificação de transformadas inversas, uma tarefa que pode ser bastante trabalhosa na solução de equações diferenciais.

2.3 Derivada fracionária segundo Grünwald-Letnikov

A derivada fracionária segundo Grünwald-Letnikov é uma das primeiras definições normalmente estudadas quando se inicia no cálculo fracionário. Esta formulação foi proposta por Anton Karl Grünwald, em 1867 (GRÜNWALD, 1867) e Aleksey Vasilievich Letnikov, em 1868 (LETNIKOV, 1868). O operador de Grünwald-Letnikov é obtido a partir da generalização da derivada de ordem n. Considera-se uma função f suficientemente regular de ordem $n \in \mathbb{N}$:

$$
\begin{aligned}
D^{(1)}f(t) &= \lim_{\Delta t\to 0}\frac{f(t)-f(t-\Delta t)}{\Delta t}\\
D^{(2)}f(t) &= \lim_{\Delta t\to 0}\frac{f(t)-2f(t-\Delta t)+f(t-2\Delta t)}{\Delta t^2}\\
D^{(3)}f(t) &= \lim_{\Delta t\to 0}\frac{f(t)-3f(t-\Delta t)+3f(t-2\Delta t)-f(t-3\Delta t)}{\Delta t^3}\\
\vdots\ &\ \vdots\ \ \vdots\\
D^{(n)}f(t) &= \lim_{\Delta t\to 0}\frac{1}{\Delta t^n}\sum_{k=0}^{n}(-1)^k\binom{n}{k}f(t-k\Delta t), \qquad (25)
\end{aligned}
$$

onde $D^{(2)}f(t) = D^{(1)}\left[D^{(1)}f(t)\right]$, $D^{(3)}f(t) = D^{(1)}\left[D^{(2)}f(t)\right]$ e assim por diante. Em (25), surgem os coeficientes binomiais, isto é,

$$
\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\dots(n-k-1)}{k!}.
$$

Definição 7. *A partir da expressão (25) define-se a derivada segundo Grünwald-Letnikov, $D^\alpha f(t) \equiv {}_{GL}D^\alpha f(t)$, substituindo diretamente $n \in \mathbb{N}$ por $\alpha > 0$ e o somatório por uma série infinita:*

$$
{}_{GL}D^\alpha f(t) = \lim_{\Delta t\to 0}\frac{1}{\Delta t^\alpha}\sum_{k=0}^{\infty}(-1)^k\binom{\alpha}{k}f(t-k\Delta t), \qquad (26)
$$

em que

$$
\binom{\alpha}{k} = \frac{\Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-k+1)}. \qquad (27)
$$

Como $\alpha \in \mathbb{R}-\mathbb{Z}_-$ e $k \in \mathbb{N}$, o coeficiente binomial generalizado (27) não se anula para valores de $k > \alpha$, diferentemente do que ocorre no caso em que $\alpha \in \mathbb{N}$.

A representação da formulação de Grunwald-Letnikov como um limite não é interessante, pois impossibilita grande liberdade de manipulação, por exemplo, na resolução de uma equação diferencial. Contudo, dentre outras razões esse operador tem significativa relevância devido ao fato de ter uma ampla aplicabilidade na abordagem numérica e sua equivalência como outros operadores fracionários, como será visto no Capítulo 3.

2.4 Aproximações por Diferenças Finitas

Nesta seção, apresentamos os conceitos fundamentais relativos ao método das diferenças finitas. Esse método consiste em substituir cada um dos operadores de diferenciação, explicitados na equação sob estudo, por uma aproximação de diferenças finitas.

As escolhas são feitas com base em aproximações obtidas a partir da série de Taylor, que produzem fórmulas que aproximam valores em um ponto com dependência de um ou mais pontos no domínio da equação considerada.

Em 1731, Brook Taylor (1685 − 1731), na publicação *Methodus Incrementorum Directa et Inversa*, introduziu a série

$$f(a+x) = f(a) + f'(a)x + f''(a)\frac{x^2}{2!} + f'''(a)\frac{x^3}{3!} + \ldots + f^{(n)}(a)\frac{x^n}{n!} + \ldots, \tag{28}$$

que passou a ser conhecida como série de Taylor.

O teorema da expansão de Taylor é a base teórica dos métodos de diferenças finitas. Sua demonstração pode ser encontrada em (COURANT, 1963).

Teorema 1. (Teorema de Taylor) *Seja f uma função real de uma variável, derivável até a ordem $n+1$ em um intervalo I, que contém x, com $n \geq 0$ inteiro. Então, para cada ponto $x+h$ em I, existe um número real ξ, entre x e $x+h$, tal que*

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2!} + \ldots + f^{(n)}(x)\frac{h^n}{n!} + E_n(x+h), \tag{29}$$

onde

$$E_n(x+h) := f^{n+1}(\xi)\frac{h^{n+1}}{(n+1)!}\,.$$

O valor absoluto

$$|E_n(x+h)| = |f(x+h) - P_n(x+h)|$$

é chamado de erro associado à aproximação

$$P_n(x+h) := f(x) + f'(x)h + f''(x)\frac{h^2}{2!} + \ldots + f^{(n)}(x)\frac{h^n}{n!}\,, \tag{30}$$

denominada polinômio de Taylor de ordem n, associado a f, x e h.

A essência principal do método das diferenças finitas é discretizar o domínio onde a equação diferencial está definida e substituir as derivadas por aproximações envolvendo apenas valores numéricos da função (CUMINATO; MENEGUETTE, 2013). Contudo, antes de discretização é necessário apresentar a definição de malha. Para isso, considere o domínio a seguir

$$D := \{(x,t)|\ 0 \leq x \leq L,\ 0 \leq t \leq T\}$$

sendo L e T constantes positivas. Optamos por malhas retangulares homogêneas, com a orientação dos eixos coordenados. São mais simples e uma delas é exibida na Figura 3.

A malha é construída tomando primeiramente o segmento $[0,L]$, que deve ser dividido em M subintervalos de mesmo[13] comprimento Δx; esse é o passo associado

[13] Daí a expressão *homogênea*.

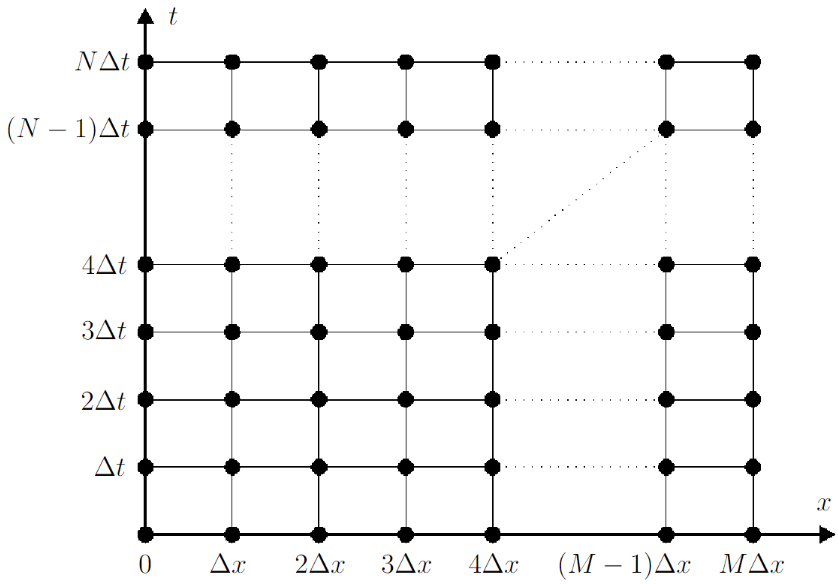
à variável espacial. Em seguida, o intervalo $[0,T]$ é dividido em N subintervalos de comprimento Δt, o passo associado à variável temporal. Dessa forma, define-se uma malha com os nós (x_i, t_n) dados por

$$x_i := i\Delta x, \quad i = 0,1,2,\ldots,M, \quad L = M\Delta x \ ,$$

$$t_n := n\Delta t, \quad n = 0,1,2,\ldots,N, \quad T = N\Delta t \ .$$

sendo M e N inteiros positivos.

Figura 3 - Malha Computacional.



Legenda: Representação de uma malha computacional homogênea bidimensional.
Fonte: O autor, 2023.

O processo, denominado discretização do domínio, dá início ao esquema que permite aproximar um problema contínuo por um problema discreto, passando-se da busca da solução em espaços de dimensão infinita para espaços de dimensão finita. Por exemplo, a equação da difusão do calor de ordem inteira no domínio já discretizado passa a ser escrita nos nós gerados, segundo a formulação

$$\frac{\partial u}{\partial t}(x_i,t_n) = \mathcal{K}\frac{\partial^2 u}{\partial x^2}(x_i,t_n) + f(x_i,t_n), \quad 0 < x < L, \quad 0 < t \leq T, . \tag{31}$$

O próximo passo envolve a discretização das derivadas, ou seja, a aproximação dos operadores de derivadas parciais na equação diferencial por meio de operadores de diferenças finitas.

Considere a derivada primeira em relação à variável t em (31). Ao efetuar o desenvolvimento da função $u(x,t)$ por meio da série de Taylor em torno dos pontos $t+\Delta t$

e $t - \Delta t$, obtém-se, respectivamente,

$$u(x, t + \Delta t) = u(x, t) + \Delta t u_t(x, t) + \frac{\Delta t^2}{2!} u_{tt}(x, t) + \frac{\Delta t^3}{3!} u_{ttt}(x, t) + \dots . \tag{32}$$

$$u(x, t - \Delta t) = u(x, t) - \Delta t u_t(x, t) + \frac{\Delta t^2}{2!} u_{tt}(x, t) - \frac{\Delta t^3}{3!} u_{ttt}(x, t) + \dots . \tag{33}$$

Como estamos interessados na primeira derivada, truncamos a série no terceiro termo. Ao reorganizar as equações (32) e (33) e dividi-las por Δt, obtemos, respectivamente,

$$\delta_t^+ U := \frac{u(x, t + \Delta t) - u(x, t)}{\Delta t} = u_t(x, t) + O\left(\frac{\Delta t}{2!}, u_{tt}\right) \tag{34}$$

$$\delta_t^- U := \frac{u(x, t) - u(x, t - \Delta t)}{\Delta t} = u_t(x, t) - O\left(\frac{\Delta t}{2!}, u_{tt}\right) \tag{35}$$

Agora no segundo membro da equação (31), é necessário aproximar a derivada segunda com relação à variável x. Para isso, devemos usar desenvolvimentos em série de Taylor da função $u(x, t)$ para os pontos $x + \Delta x$ e $x - \Delta x$. Daí, obtém-se

$$u(x + \Delta x, t) = u(x, t) + \Delta x u_x(x, t) + \frac{\Delta x^2}{2!} u_{xx}(x, t) + \frac{\Delta x^3}{3!} u_{xxx}(x, t) + \frac{\Delta x^4}{4!} u_{xxxx}(x, t) + \dots \tag{36}$$

$$u(x - \Delta x, t) = u(x, t) - \Delta x u_x(x, t) + \frac{\Delta x^2}{2!} u_{xx}(x, t) - \frac{\Delta x^3}{3!} u_{xxx}(x, t) + \frac{\Delta x^4}{4!} u_{xxxx}(x, t) - \dots \tag{37}$$

Somando as duas séries, reorganizando os termos e dividindo por Δx^2, temos

$$\delta_x^2 U := \frac{u(x - \Delta x, t) - 2u(x, t) + u(x + \Delta x, t)}{\Delta x^2} = u_{xx}(x, t) + O\left(\frac{\Delta x^2}{12}, u_{xxxx}\right) \tag{38}$$

que é uma aproximação para a derivada segunda em relação à variável x, denominada de diferença centrada na variável x, com um erro local da ordem de Δx^2.

2.5 Esquemas de Diferenças Finitas

A partir das aproximações encontradas na seção anterior, podemos obter alguns dos métodos de diferenças finitas mais conhecidos, como veremos a seguir.

Nos pontos (x_i, t_n) da malha denotamos $u(x_i, t_n)$ por U_i^n e substituindo (34) e (38) na equação diferencial (31), que conduz à equação de diferenças finitas

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \mathcal{K}\left(\frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2}\right) + f_i^n\ . \tag{39}$$

Esse esquema numérico é conhecido como Método de Diferenças Progressivas. A principal vantagem desse método é a facilidade de sua implementação, devido à sua natureza explícita, isto é,

$$U_i^{n+1} = \upsilon U_{i-1}^n + (1 - 2\upsilon)U_i^n + \upsilon U_{i+1}^n + \Delta t f_i^n, \tag{40}$$

onde υ denota o quociente $\mathcal{K}\Delta t/\Delta x^2$.

No entanto, o método das diferenças progressivas é condicionalmente estável, o que pode ser considerada uma deficiência, pois exige uma forte condição entre as taxas de convergência a zero dos parâmetros Δx e Δt, a saber,

$$\frac{\Delta t}{(\Delta x)^2} \leq \text{constante fixa.} \tag{41}$$

Haverá convergência das aproximações construídas somente se a condição CFL[14] (Courant-Fridrichs-Lewy) for satisfeita (MOURA; KUBRUSLY, 2013).

Para $1 \leq i \leq M - 1$, podemos escrever a equação (40) na forma matricial, isto é,

$$\boldsymbol{U^{n+1}} = \mathcal{M}\boldsymbol{U^n} + \boldsymbol{C^n} + \Delta t \boldsymbol{f^n}, \tag{42}$$

onde $\boldsymbol{U^n} = \left(U_1^n, U_2^n, \ldots, U_{N-1}^n\right)^T$, $\boldsymbol{C^n} = (\upsilon U_0^n, 0, \ldots, 0, \upsilon U_N^n)^T$, $\boldsymbol{f^n} = \left(f_1^n, f_2^n, \ldots, f_{N-1}^n\right)^T$ e

$$\mathcal{M} := \begin{pmatrix} 1-2\upsilon & \upsilon & 0 & \ldots & 0 \\ \upsilon & 1-2\upsilon & \upsilon & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \upsilon & 1-2\upsilon & \upsilon \\ 0 & \ldots & 0 & \upsilon & 1-2\upsilon \end{pmatrix}.$$

A seguir, apresentamos o algoritmo do Método de Euler Progressivo referente à equação (42).

Método de Euler Progressivo

---

[14] É uma condição de estabilidade numérica que se aplica a esquemas de diferenças finitas usados para resolver equações diferenciais parciais (EDPs) de forma aproximada. Ela estabelece uma restrição na escolha do tamanho do passo de tempo e do tamanho do espaço de discretização na malha utilizada. Essa restrição é necessária para evitar a amplificação de erros numéricos e garantir que o comportamento físico correto seja reproduzido de forma estável.

Propósito:

Solução numérica da equação da difusão clássica de ordem inteira.

$$\frac{\partial u(x,t)}{\partial t} = \mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema explícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$\upsilon = \mathcal{K}\Delta t/\Delta x^2$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $\mathcal{M}_i^j = 1 - 2\upsilon$;

se $i == j-1$ ou $i == j+1$, então $\mathcal{M}_i^j = \upsilon$;
senão $\mathcal{M}_i^j = 0$, (matriz tridiagonal);

Passo 6

Faça $n = 0$;
Enquanto $n < N$ faça
crie, V, vetor de ordem M;
para $k = 0, 1, \ldots, n$ faça
$V = \mathcal{M} \times U^n$;
$V_0 = V_0 + \upsilon \times U_0^n$;
$V_{M-1} = V_{M-1} + \upsilon \times U_M^n$;
$U^{n+1} = V + \Delta t \times F^n$;
$n = n + 1$;

Passo 7

Para $n = 0, 1, \ldots, N$ faça
imprima U^n;

Passo 8

PARE (O procedimento está completo).

O método das diferenças regressivas é um esquema que possui a mesma ordem de precisão do método das diferenças progressivas (BURDEN, 2011). No entanto, ele é incondicionalmente estável, ou seja, a convergência global não depende da relação mantida assintoticamente entre Δx e Δt (QUARTERONI; SACCO; SALERI, 2010). Substituindo (35) e (38) em (31), temos

$$\frac{U_i^n - U_i^{n-1}}{\Delta t} = \mathcal{K}\left(\frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2}\right) + f_i^n \, . \tag{43}$$

Como $\upsilon := \mathcal{K}\Delta t/\Delta x^2$ a partir da expressão (43), chegamos a

$$-\upsilon U_{i-1}^n + (1+2\upsilon)U_i^n - \upsilon U_{i+1}^n = U_i^{n-1} + \Delta t f_i^n, \tag{44}$$

que mostra termos um método implícito. Para $1 \leq i \leq M-1$, a forma matricial de (44) é

$$\mathcal{N}\boldsymbol{U^n} = \boldsymbol{U^{n-1}} - \boldsymbol{C^n} + \Delta t \boldsymbol{f^n}, \tag{45}$$

onde a matriz $\mathcal{N}$ é escrita da seguinte forma

$$\mathcal{N} := \begin{pmatrix} 1+2\upsilon & -\upsilon & 0 & \dots & 0 \\ -\upsilon & 1+2\upsilon & -\upsilon & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -\upsilon & 1+2\upsilon & -\upsilon \\ 0 & \dots & 0 & -\upsilon & 1+2\upsilon \end{pmatrix}.$$

O algoritmo do método de Euler regressivo, referente à equação (45), é descrito a seguir.

Método de Euler Regressivo

Propósito:

Solução numérica da equação da difusão clássica de ordem inteira.

$$\frac{\partial u(x,t)}{\partial t} = \mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema implícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \dots, M$ e para cada $n = 0, 1, \dots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$\upsilon = \mathcal{K}\Delta t/\Delta x^2$;

Passo 2

Para $i = 0, 1, \dots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M - 1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $i = 0, 1, \ldots, M - 1$ faça

para $j = 0, 1, \ldots, M - 1$ faça

se $i == j$, então $\mathcal{N}_i^j = 1 + 2\upsilon$;

se $i == j - 1$ ou $i == j + 1$, então $\mathcal{N}_i^j = -\upsilon$;

senão $\mathcal{N}_i^j = 0$, (matriz tridiagonal);

Passo 6

Faça $n = 0$;

Enquanto $n < N$ faça

$U_0^{n-1} = U_0^{n-1} - \upsilon \times U_0^n$;

$U_M^{n-1} = U_M^{n-1} - \upsilon \times U_M^n$;

resolva, $\mathcal{N}U^n = U^{n-1} + \Delta t \times F^n$;

$n = n + 1$;

Passo 7

Para $n = 0, 1, \ldots, N$ faça

imprima U^n;

Passo 8

PARE (O procedimento está completo).

Como não houve modificação na forma de aproximar as derivadas, o método das diferenças regressivas, possui a mesma ordem de precisão do das diferenças progressivas, isto é, $O(\Delta t + \Delta x^2)$. Nesse sentido, o método de Crank-Nicolson é mais indicado por ser de segunda ordem em termos de precisão, tanto no tempo quanto no espaço, isto é, $O(\Delta t^2 + \Delta x^2)$. Ele é um método implícito amplamente utilizado por ser confiável e eficiente para resolver equações diferenciais parciais, especialmente para problemas de difusão parabólica (LEVEQUE, 2007).

O algoritmo de Crank-Nicolson apresenta um aspecto particular: ele simula a equação no ponto médio entre dois nós consecutivos dos níveis temporais. Por isso, ele pode ser pensado como a média aritmética entre os métodos de diferenças progressivas e regressivas, como visto a seguir

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} \; = \; \frac{\mathcal{K}}{2}\left(\frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2} + \frac{U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}}{\Delta x^2}\right) + f_i^{n+1/2}\,, \qquad (46)$$

onde $f_i^{n+1/2} := (f_i^n + f_i^{n+1})/2$. Como $\upsilon := \mathcal{K}\Delta t/\Delta x^2$, escrevemos

$$-\frac{\upsilon}{2}U_{i-1}^{n+1} + (1+\upsilon)U_i^{n+1} - \frac{\upsilon}{2}U_{i+1}^{n+1} = \frac{\upsilon}{2}U_{i-1}^n + (1-\upsilon)U_i^n + \frac{\upsilon}{2}U_{i+1}^n + \Delta t f_i^{n+1/2}. \qquad (47)$$

Para $1 \leq i \leq M-1$, a forma matricial para (47) é

$$\mathbf{N}\boldsymbol{U^{n+1}} \; = \; \mathbf{M}\boldsymbol{U^n} + \boldsymbol{C^n} - \boldsymbol{C^{n+1}} + \Delta t\boldsymbol{f^{n+1/2}}, \qquad (48)$$

onde:

$$\mathbf{M} := \begin{pmatrix} 1-\upsilon & 0,5\upsilon & 0 & \dots & 0 \\ 0,5\upsilon & 1-\upsilon & 0,5\upsilon & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0,5\upsilon & 1-\upsilon & 0,5\upsilon \\ 0 & \dots & 0 & 0,5\upsilon & 1-\upsilon \end{pmatrix}$$

e

$$\mathbf{N} := \begin{pmatrix} 1+\upsilon & -0,5\upsilon & 0 & \dots & 0 \\ -0,5\upsilon & 1+\upsilon & -0,5\upsilon & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -0,5\upsilon & 1+\upsilon & -0,5\upsilon \\ 0 & \dots & 0 & -0,5\upsilon & 1+\upsilon \end{pmatrix}.$$

Abaixo, fornecemos o algoritmo do Método de Crank-Nicolson para a equação (47).

Método de Crank-Nicolson

Propósito:

Solução numérica da equação da difusão clássica de ordem inteira.

$$\frac{\partial u(x,t)}{\partial t} = \mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema implícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$\upsilon = \mathcal{K}\Delta t/\Delta x^2$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $\mathbf{M}_i^j = 1 - \upsilon$;

se $i == j-1$ ou $i == j+1$, então $\mathbf{M}_i^j = 0,5\upsilon$;

senão $\mathbf{M}_i^j = 0$, (matriz tridiagonal);

Passo 6

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $\mathbf{N}_i^j = 1 + \upsilon$;

se $i == j-1$ ou $i == j+1$, então $\mathbf{N}_i^j = -0,5\upsilon$;

senão $\mathbf{N}_i^j = 0$, (matriz tridiagonal);

Passo 7

Faça $n = 0$;

Enquanto $n < N$ faça

crie, V, vetor de ordem M;
para $k = 0, 1, \ldots, n$ faça
$V = \mathbf{M} \times U^n$;
$V_0 = V_0 + \upsilon \times U_0^n - \upsilon \times U_0^{n+1}$;
$V_{M-1} = V_{M-1} + \upsilon \times U_M^n - \upsilon \times U_M^{n+1}$;
resolva, $\mathbf{N}U^{n+1} = V + \Delta t(F^n + F^{n+1})/2$;
$n = n + 1$;

Passo 8

Para $n = 0, 1, \ldots, N$ faça
imprima U^n;

Passo 9

PARE (O procedimento está completo).

A escolha de um dos métodos: diferenças progressivas, regressivas ou de Crank-Nicolson, depende da natureza do problema, das exigências de precisão e estabilidade, e do equilíbrio entre eficiência computacional e acurácia numérica. Os códigos referentes aos métodos estão disponíveis em linguagem Python no Apêndice A.

2.6 Consistência, estabilidade e convergência

Nesta seção, vamos explorar os conceitos fundamentais para avaliar a qualidade e confiabilidade dos algoritmos utilizados na resolução de problemas matemáticos por meio de métodos computacionais. Ao aplicarmos um método numérico a uma equação diferencial, é pertinente questionar o quanto a solução obtida se aproxima da solução analítica. Por isso, torna-se necessário realizar o estudo da consistência, estabilidade e convergência do método.

2.6.1 Consistência

Uma condição importante para uma aproximação por diferenças finitas é que ela seja consistente com a equação diferencial que está sendo discretizada. Quando aplica-se às aproximações as derivadas de uma equação diferencial, uma equação de diferenças finitas EDF é formada e consequentemente o erro advindo exclusivamente pela substituição da solução exata na equação de diferenças é chamado de erro de truncamento local. Todavia, quando o espaçamento da malha, tanto no espaço Δx quanto no tempo Δt, tende a zero, é desejável que a equação de diferenças finitas se aproxime da equação diferencial parcial. Essa é a condição para garantir a consistência do método.

2.6.2 Estabilidade

Em 1956, Peter Lax e Robert Richtmyer publicaram o trabalho no qual introduziram o conceito mais geral de estabilidade (LAX; RICHTMYER, 1956). Esse conceito afirma que se um método numérico não é estável, então quaisquer erros ou perturbações são amplificados sem limites. Em outras palavras, o conceito de estabilidade está relacionado ao crescimento ou decrescimento dos erros introduzidos nos cálculos.

Ao aplicar um método numérico a uma equação diferencial, é de suma importância realizar o estudo da sua estabilidade, uma vez que essa está diretamente ligada à sua confiabilidade. Os métodos numéricos são classificados em três categorias em relação à sua estabilidade. São elas:

- Condicionalmente estáveis: São métodos numéricos nos quais a estabilidade depende de certas condições ou restrições nos parâmetros do problema ou no passo de discretização. Esses métodos podem ser estáveis apenas dentro de determinados limites ou faixas de valores. Caso essas condições não sejam atendidas, a estabilidade pode ser comprometida e erros amplificados podem ocorrer.

- Incondicionalmente estáveis: São métodos numéricos nos quais os erros e perturbações introduzidos nos cálculos são controlados e limitados. Esses métodos garantem que pequenas perturbações nos dados de entrada resultem em erros correspondentes também limitados na solução numérica. A estabilidade é preservada independentemente do tamanho da malha ou do número de iterações.

- Incondicionalmente instáveis: São métodos numéricos nos quais pequenas perturbações nos dados de entrada resultam em erros amplificados e incontroláveis na solução numérica. Esses métodos não são confiáveis para a resolução numérica de problemas, pois erros insignificantes podem se propagar e levar a resultados completamente errôneos. Métodos instáveis geralmente devem ser evitados na prática.

Há alguns critérios para o estudo da estabilidade de um método numérico, dos quais destacamos o critério de Von Neumann e o critério da matriz.

- O critério de Von Neumann, descrito por Charney e Fjortoft em 1950 (CHARNEY; FJÖRTOFT; NEUMANN, 1950), é um método amplamente utilizado para analisar a estabilidade de métodos numéricos que envolvem a solução de equações diferenciais parciais. Neste método, a solução da equação de diferenças finitas é expandida em série de Fourier. A redução ou crescimento do fator de amplificação indica se o método é ou não estável.

- O critério da matriz, descrito por Eddy em 1958 (EDDY, 1958), é um método utilizado para analisar a estabilidade de métodos numéricos. Diferentemente do critério de Von Neumann, ele considera o efeito de condições de contorno não periódicas na análise de estabilidade. No entanto, em muitas aplicações, calcular a condição de estabilidade na prática não é uma tarefa fácil.

2.6.3 Convergência

A convergência se baseia nos dois conceitos vistos anteriormente, consistência e estabilidade. A convergência de um método numérico é extremamente importante do ponto de vista prático, pois deseja-se que a solução numérica calculada reproduza da maneira mais próxima possível a solução exata do problema.

Destaca-se que, para a solução numérica se aproximar gradualmente da solução exata, é necessário que seja possível recuperar a equação diferencial original por meio da redução do espaçamento dos passos da malha e, consequentemente, reduzir o erro de truncamento local. Então, a consistência é uma condição necessária para a convergência (RUDIN et al., 1976). No entanto, a consistência não é a única condição para a convergência, uma vez que é possível ter um esquema consistente, mas não convergente. Para determinar a convergência, é necessário analisar a estabilidade do método.

O Teorema de Equivalência é um resultado fundamental na análise do método das diferenças finitas para a solução numérica de equações diferenciais parciais. Ele estabelece que, para um problema de valor inicial bem posto[15] e um método de discretização consistente, a estabilidade é uma condição necessária e suficiente para a convergência (RICHTMYER; MORTON, 1967).

[15] O matemático francês *Jacques Hadamard* definiu um problema bem-posto, como sendo aquele que cumpre as três condições, a saber: Existência - o problema deverá possuir solução; Unicidade - a solução deverá ser única; Estabilidade - a solução tem uma dependência contínua (suave) com os dados de entrada (NEGREIROS, 2010).

3 DIFUSÃO FRACIONÁRIA COM DERIVADAS CLÁSSICAS

Este capítulo será dedicado ao estudo de duas formulações para as derivadas fracionárias: a formulação de Riemann-Liouville e a de Caputo. Essas duas formulações serão denominadas como derivadas fracionárias clássicas, conforme a classificação estabelecida por Teodoro (TEODORO, 2019). Serão apresentados os métodos numéricos propostos para a equação da difusão fracionária (15) utilizando o operador fracionário de Riemann-Liouville e de Caputo. A abordagem será feita através do método das diferenças finitas, com a aplicação dos esquemas baseados no método de Euler progressivo e regressivo para o operador de Riemann-Liouville, e regressivo para o operador de Caputo. O estudo da análise de estabilidade e convergência dos esquemas numéricos será realizado para a derivada de Riemann-Liouville. Além disso, serão apresentados exemplos de aplicação dos métodos e seus respectivos algoritmos para ambos os operadores.

3.1 Derivada fracionária segundo Riemann-Liouville

Em 1869, Nikoly Sonin publicou o primeiro trabalho que emprega o operador conhecido como derivada de Riemann-Liouville (SONIN, 1869). Esse operador é definido em termos da integral fracionária:

Definição 8. *Para $\alpha > 0$ definem-se as integrais fracionárias de Riemann-Liouville à esquerda* (49) *e à direita* (50) *para uma função f integrável a Riemann em (a, b), respectivamente por*

$$ {}_{RL}I^{\alpha}_{a,t}f(t) := \frac{1}{\Gamma(\alpha)}\int_a^t (t-s)^{\alpha-1}f(s)ds \tag{49}$$

$$ {}_{RL}I^{\alpha}_{t,b}f(t) := \frac{1}{\Gamma(\alpha)}\int_t^b (s-t)^{\alpha-1}f(s)ds \ , \tag{50}$$

sendo $\Gamma(\cdot)$ a função gama e α a ordem da integral fracionária.

A formulação dessa derivada de ordem arbitrária é definida em termos da integral fracionária com base no fato de a derivação ser operação inversa da integração e na lei dos expoentes (CAMARGO; OLIVEIRA, 2015).

Definição 9. *As derivadas fracionárias de Riemann-Liouville de ordem $\alpha > 0$ à esquerda* (51) *e à direita* (52) *de uma função $f(t)$, $t \in (a, b)$ são definidas respectivamente como*

$$ {}_{RL}D^{\alpha}_{a,t}f(t) := \frac{1}{\Gamma(m-\alpha)}\frac{d^m}{dt^m}\int_a^t (t-s)^{m-\alpha-1}f(s)ds \ , \tag{51}$$

$$ {}_{RL}D^{\alpha}_{t,b}f(t) := \frac{(-1)^m}{\Gamma(m-\alpha)}\frac{d^m}{dt^m}\int_t^b (s-t)^{m-\alpha-1}f(s)ds \ , \tag{52} $$

sendo m o inteiro positivo que satisfaz $m-1<\alpha\leq m$, chamado teto inteiro de α.

Essa formulação cumpre as cinco propriedades do critério proposto por Ortigueira e Machado, isto é: a derivada fracionária de Riemann-Liouville é um operador linear; a derivada de ordem zero de uma função é a própria função; a derivada fracionária para α inteiro gera o mesmo resultado da derivada ordinária equivalente; a lei dos expoentes é satisfeita e vale a generalização da regra de Leibniz (TEODORO, 2019).

O problema de valores iniciais e de contorno da equação de difusão fracionária que descreve a subdifusão/superdifusão (SO; LIU, 2004) consiste em determinar uma função $u(x,t)$, definida para $0<x<L$ e $0<t\leq T$, tal que

$$ \frac{\partial u(x,t)}{\partial t} = {}_{RL}D^{\alpha}_{0,t}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0<x<L, \quad 0<t\leq T, \tag{53a} $$

$$ u(x,0) = \phi(x), \quad 0\leq x\leq L, \tag{53b} $$

$$ u(0,t) = l(t), \quad 0\leq t\leq T, \tag{53c} $$

$$ u(L,t) = r(t), \quad 0\leq t\leq T, \tag{53d} $$

onde $0<\alpha<1$ é a ordem da derivada fracionária e $\phi(x)$, $l(t)$, $r(t)$ e $f(x,t)$ são funções suficientemente regulares dadas.

A derivada fracionária de Grünwald-Letnikov para uma função suficientemente regular $u(t)$ é equivalente à sua derivada fracionária no sentido de Riemann-Liouville (BALEANU et al., 2012). Este fato torna possível obter aproximações para a derivada fracionária de Riemann-Liouville se empregarmos a de Grünwald-Letnikov, isto é,

$$ {}_{RL}D^{\alpha}_{0,t}u(t)\Big|_{t=t_n} = \frac{1}{\Delta t^{\alpha}}\sum_{k=0}^{n}(-1)^k\binom{\alpha}{k}u(t_{n-k}) + O(\Delta t). \tag{54} $$

A expressão (54) é uma aproximação linear (de ordem 1) para qualquer $\alpha>0$. Contudo, para $1<\alpha<2$, ela pode gerar esquemas numéricos instáveis (MEERSCHAERT; TADJERAN, 2004).

3.2 Método tipo Euler progressivo com Riemann-Liouville

Nesta seção, desenvolveremos o método tipo Euler progressivo ($MtEP$) aplicado ao problema (53). Este esquema explícito é particularmente interessante devido à sua simplicidade e fácil implementação[16]. Aplicamos em (53a) as aproximações (34), (38) e (54). Dessa forma, o problema (53) é escrito da seguinte maneira, com $i = 1, 2, \ldots, M-1$ e $n = 1, 2, \ldots, N$:

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} = \mathcal{K}\left(\frac{\tau}{\Delta t}\right)^{\alpha} \sum_{k=0}^{n} \omega(k) \left(\frac{U_{i-1}^{n-k} - 2U_i^{n-k} + U_{i+1}^{n-k}}{\Delta x^2}\right) + f_i^n, \tag{55a}$$

$$U_i^0 = \phi(x_i) \quad (i = 0, 1, \ldots, M), \tag{55b}$$

$$U_0^n = l(t_n) \quad (n = 0, 1, \ldots, N), \tag{55c}$$

$$U_M^n = r(t_n) \quad (n = 0, 1, \ldots, N), \tag{55d}$$

onde $\omega(k) = (-1)^k \binom{\alpha}{k}$ e $f_i^n = f(x_i, t_n)$ em (55a). A partir de (55a), escrevemos

$$U_i^{n+1} = U_i^n + r \sum_{k=0}^{n} \omega(k)(U_{i-1}^{n-k} - 2U_i^{n-k} + U_{i+1}^{n-k}) + \Delta t f_i^n, \tag{56}$$

com $r := \mathcal{K}\frac{\Delta t}{\Delta x^2}\left(\frac{\tau}{\Delta t}\right)^{\alpha}$. Para $1 \leq i \leq M-1$, a forma matricial para (56) é

$$\boldsymbol{U^{n+1}} = \boldsymbol{U^n} + \sum_{k=0}^{n} \omega(k)\left(A\boldsymbol{U^{n-k}} + \boldsymbol{C^{n-k}}\right) + \Delta t \boldsymbol{f^n}, \tag{57}$$

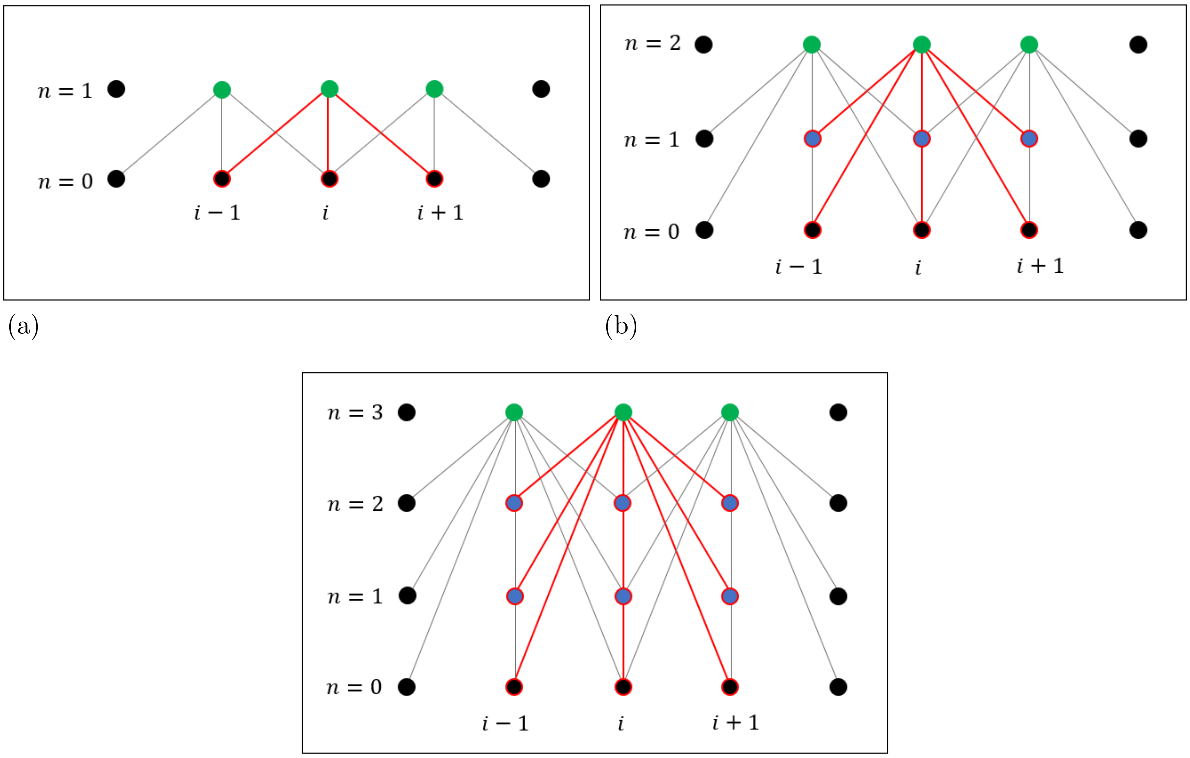
onde $\boldsymbol{f^n} = \left(f_1^n, f_2^n, \ldots, f_{N-1}^n\right)^T$, $\boldsymbol{U^n} = \left(U_1^n, U_2^n, \ldots, U_{N-1}^n\right)^T$, $\boldsymbol{C^n} = (rU_0^n, 0, \ldots, 0, rU_N^n)^T$ e

$$A := \begin{pmatrix} -2r & r & 0 & \ldots & 0 \\ r & -2r & r & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & r & -2r & r \\ 0 & \ldots & 0 & r & -2r \end{pmatrix}.$$

As Figuras 4a, 4b e 4c exibem a dependência dos valores calculados com relação aos nós nos três primeiros níveis para o $MtEP$, respectivamente. O estêncil formado pelas arestas vermelhas mostra os valores que precisam ser conhecidos para o cálculo do nó da posição i no passo n. Esta visualização comprova o efeito de memória quando utilizamos a derivada fracionária (FARIA; MOURA; NEGREIROS, 2021).

[16] Código na linguagem Python disponível no Apêndice B.1

Figura 4 - Estêncil do método tipo Euler progressivo com a derivada de Riemann-Liouville.



(a) (b) (c)

Legenda: As figuras exibem os estênceis em vermelho na posição i e nos níveis: (a) $n = 1$, (b) $n = 2$ e (c) $n = 3$, para o problema (15), com a derivada segundo Riemann-Liouville, quando aplicado o método tipo Euler progressivo.

Fonte: O autor, 2023.

3.2.1 Estabilidade do *MtEP* com Riemann-Liouville

A estabilidade de um esquema numérico está relacionada ao crescimento ou decaimento dos erros decorrentes das aproximações nos operadores e das múltiplas operações aritméticas associadas com a solução das equações algébricas. Esse estudo pode ser realizado por diferentes abordagens entre as quais podemos citar o método da energia (GAO; SUN, 2011), o método das matrizes (LIU et al., 2004) e o critério de von Neumann que é também conhecido como método de Fourier (STRIKWERDA, 2004).

O critério de von Neumann é fundamentado nas séries de Fourier e este possibilita dar condições necessárias e suficientes para a estabilidade de esquemas de diferenças finitas. Esse critério é baseado no princípio de superposição, ou seja, o erro global é o somatório de erros mais simples, também conhecidos por harmônicos.

A análise de estabilidade do problema (55) será realizada pelo critério de von Neumann. Seja $U_i^n = d_n \, \mathrm{e}^{j\sigma i\Delta x}$, onde $j^2 = -1$ e $\sigma = 2\pi m/L$. Inserindo essa expressão em (56), temos

$$\begin{aligned}
d_{n+1}\,\mathrm{e}^{j\sigma i\Delta x} &= d_n\,\mathrm{e}^{j\sigma i\Delta x} + r\sum_{k=0}^{n}\omega(k)\left(d_{n-k}\,\mathrm{e}^{j\sigma(i-1)\Delta x} - 2d_{n-k}\,\mathrm{e}^{j\sigma i\Delta x} + d_{n-k}\,\mathrm{e}^{j\sigma(i+1)\Delta x}\right)\\
d_{n+1} &= d_n + r\sum_{k=0}^{n}\omega(k)d_{n-k}\left(\mathrm{e}^{-j\sigma\Delta x} - 2 + \mathrm{e}^{j\sigma\Delta x}\right)\\
d_{n+1} &= d_n + r\sum_{k=0}^{n}\omega(k)d_{n-k}\left(2\cos\left(\sigma\Delta x\right) - 2\right)\\
d_{n+1} &= d_n - 4r\,\mathrm{sen}^2\left(\frac{\sigma\Delta x}{2}\right)\sum_{k=0}^{n}\omega(k)d_{n-k}\;. \qquad (58)
\end{aligned}$$

Teorema 2. *O método explícito tipo Euler progressivo* (55) *é estável se*

$$\mathcal{K}\frac{\Delta t}{\Delta x^2}\left(\frac{\tau}{\Delta t}\right)^{\alpha} \leq \frac{1}{2^{1+\alpha}}.$$

Demonstração. A estabilidade em (58) é determinada pelo comportamento de d_n. Por Von Neumann podemos escrever

$$d_{n+1} = \gamma(\sigma)d_n, \qquad (59)$$

e assumindo que $\gamma := \gamma(\sigma)$ não depende do tempo (STRIKWERDA, 2004). Então (58) implica uma expressão fechada para o fator de amplificação γ. Aplicando $d_{n+1} = \gamma d_n$ e $d_{n-k} = \gamma^{-k} d_n$ em (58), temos

$$\gamma d_n = d_n - 4r\sin^2\left(\frac{\sigma\Delta x}{2}\right)\sum_{k=0}^{n}\omega(k)\gamma^{-k}d_n \qquad (60)$$

ou

$$\gamma = 1 - 4r\sin^2\left(\frac{\sigma\Delta x}{2}\right)\sum_{k=0}^{n}\omega(k)\gamma^{-k}. \tag{61}$$

Se $|\gamma| > 1$ para algum σ, o fator da solução d_n cresce para infinito de acordo com (59) e o método é instável. Considerando o valor extremo $\gamma = -1$, obtemos de (61) o seguinte limite de estabilidade em r, isto é,

$$-1 = 1 - 4r\sin^2\left(\frac{\sigma\Delta x}{2}\right)\sum_{k=0}^{n}(-1)^k\binom{\alpha}{k}(-1)^{-k} \tag{62}$$

$$1 = 2r\sin^2\left(\frac{\sigma\Delta x}{2}\right)\sum_{k=0}^{n}\binom{\alpha}{k}. \tag{63}$$

$$r = \frac{1}{2\sin^2\left(\frac{\sigma\Delta x}{2}\right)\sum_{k=0}^{n}\binom{\alpha}{k}} \tag{64}$$

O valor de r em (64) depende do número de iterações n. No entanto, esta dependência é fraca; dessa forma aproxima-se $\sum_{k=0}^{n}\binom{\alpha}{k}$ por $\sum_{k=0}^{\infty}\binom{\alpha}{k} = 2^\alpha$ (YUSTE; ACEDO, 2005). O valor máximo do quadrado da função *seno* é limitado por um. Portanto, podemos fornecer um limite conservador e fácil de ser aplicado para a estabilidade do método tipo Euler progressivo

$$\mathcal{K}\frac{\Delta t}{\Delta x^2}\left(\frac{\tau}{\Delta t}\right)^\alpha \leq \frac{1}{2^{1+\alpha}}\ . \tag{65}$$

$\square$

Vale notar que, se $\alpha \to 0$, podemos pensar que, em algum sentido, a equação da difusão fracionária (53a) se reduz à equação clássica da difusão (31). Por outro lado, ao fazer α tender a 0 em (65), essa condição tende para

$$\mathcal{K}\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}, \tag{66}$$

que é a condição de estabilidade do método de Euler progressivo quando aplicado à equação clássica da difusão (LEVEQUE, 2007).

3.2.2 Convergência do *MtEP* com Riemann-Liouville

A propriedade mais fundamental que um esquema numérico deve possuir para ser útil é que suas soluções se aproximem da solução exata da equação diferencial parcial correspondente e que essa aproximação melhore à medida que os espaçamentos da malha, Δx e Δt, tendam a zero. Quando essa condição é satisfeita, chamamos esse esquema de convergente (STRIKWERDA, 2004).

O estudo da convergência será realizado por meio de uma relação precisa entre as condições de consistência, estabilidade e convergência para um problema bem-posto. Esse resultado é decorrente do teorema da equivalência de Lax-Richtmyer, que será enunciado a seguir.

Teorema 3 (Teorema da Equivalência de Lax-Richtmyer)**.** *Um esquema de diferenças finitas consistente para uma equação diferencial parcial, cujo problema de valor inicial é bem-posto, é convergente se e somente se for estável.*

Demonstração. Ver (STRIKWERDA, 2004) □

O teorema de equivalência de Lax-Richtmyer relaciona um conceito importante que é difícil de estabelecer (convergência) diretamente com outros conceitos que são relativamente mais fáceis de verificar (consistência e estabilidade) e estabelece essa relação de forma muito precisa. A aplicação do teorema concentra-se na parte que afirma que consistência e estabilidade implicam convergência. Todavia, o resultado do teorema também é útil no sentido inverso. Ele afirma que não devemos considerar esquemas instáveis, pois nenhum deles será convergente. Assim, a classe de esquemas razoáveis é precisamente delimitada como aqueles que são consistentes e estáveis; nenhum outro esquema é merecedor de considerações.

Definição 10. *Dada uma equação diferencial parcial, $Pu = f$ e um esquema de diferenças finitas $P_{\Delta x,\Delta t}u = f$, dizemos que o esquema de diferenças finitas é consistente com a equação diferencial parcial se, para qualquer função suficientemente regular $\psi(x,t)$,*

$$P\psi - P_{\Delta x,\Delta t}\psi \to 0 \quad \textit{quando} \quad \Delta x,\, \Delta x \to 0,$$

a convergência ocorre de forma pontual para cada par (x,t).

Teorema 4. *O método tipo Euler progressivo* (55a) *aplicado à equação da difusão fracionária com a derivada de Riemann-Liouville* (53a) *é consistente.*

Demonstração. Usando a notação apresentada na Definição 10, temos: o operador P da equação da difusão fracionária é $\frac{\partial}{\partial t} - K_\alpha\, {}_{RL}D^\alpha_{0,t}\left(\frac{\partial^2}{\partial x^2}\right)$, de modo que

$$P\psi = \psi_t - K_\alpha\, {}_{RL}D^\alpha_{0,t}\psi_{xx} - f.$$

Para o esquema do método tipo Euler progressivo, o operador de diferença fica

$$P_{\Delta x,\Delta t}\psi = \frac{\psi_i^{n+1} - \psi_i^n}{\Delta t} - K_\alpha \frac{1}{\Delta t^\alpha}\sum_{k=0}^{n}\omega(k)\left(\frac{\psi_{i-1}^{n-k} - 2\psi_i^{n-k} + \psi_{i+1}^{n-k}}{\Delta x^2}\right) - f_i^n,$$

onde $\psi_i^n = \psi(i\Delta x, n\Delta t)$. Agora, desenvolvendo a série de Taylor da função ψ em relação a t e x em torno de $(i\Delta x, n\Delta t)$, temos

$$\psi_i^{n+1} = \psi_i^n + \Delta t\psi_t + \frac{\Delta t^2}{2!}\psi_{tt} + O(\Delta t^3) \tag{67}$$

$$\psi_{i-1}^n = \psi_i^n - \Delta x\psi_x + \frac{\Delta x^2}{2!}\psi_{xx} - \frac{\Delta x^3}{3!}\psi_{xxx} + \frac{\Delta x^4}{4!}\psi_{xxxx} - O(\Delta x^5) \tag{68}$$

$$\psi_{i+1}^n = \psi_i^n + \Delta x\psi_x + \frac{\Delta x^2}{2!}\psi_{xx} + \frac{\Delta x^3}{3!}\psi_{xxx} + \frac{\Delta x^4}{4!}\psi_{xxxx} + O(\Delta x^5), \tag{69}$$

onde as derivadas no segundo membro de (67) a (69) são avaliadas em (x_i, t_n). Consideraremos também a aproximação da derivada fracionária de Riemann-Liouville da função ψ em $t = t_n$:

$${}_{RL}D_{0,t}^{\alpha}\psi(t)\Big|_{t=t_n} = \frac{1}{\Delta t^{\alpha}}\sum_{k=0}^{n}(-1)^k\binom{\alpha}{k}\psi(t_{n-k}) + O(\Delta t). \tag{70}$$

Aplicando ajustes e manipulações convenientes em (67), (68) e (69), obtemos

$$\begin{aligned} P_{\Delta x,\Delta t}\psi &= \left(\psi_t + \frac{\Delta t}{2}\psi_{tt} + O(\Delta t^2)\right) - \frac{K_\alpha}{\Delta t^\alpha}\sum_{k=0}^{n}\omega(k)\left(\psi_{xx} + \frac{\Delta x^2}{12}\psi_{xxxx} + O(\Delta t^4)\right) \\ &+ O(\Delta t) - f_i^n. \end{aligned}$$

Fazendo

$$P\psi - P_{\Delta x,\Delta t}\psi = -\frac{1}{2}\Delta t\psi_{tt} + \frac{K_\alpha}{12}\Delta x^2\; {}_{RL}D_{0,t}^{\alpha}\psi_{xxxx} + O(\Delta t)$$

obtemos

$$P\psi - P_{\Delta x,\Delta t} \to 0 \quad \text{quando} \quad (\Delta x, \Delta t) \to 0$$

e, portanto, o esquema é consistente. □

Definição 11. *Um esquema de diferenças finitas de um passo que aproxima uma equação diferencial parcial é um esquema convergente se, para qualquer solução da equação diferencial parcial, $u(t,x)$, e soluções do esquema de diferenças finitas, U_i^n, tal que U_i^0 converge para $\phi(x)$ à medida que $i\Delta x$ converge para x, então U_i^n converge para $u(t,x)$ à medida que $(i\Delta x, n\Delta t)$ converge para (t,x) conforme Δx, Δt convergem para 0.*

Teorema 5. *O método tipo Euler progressivo* (55a) *aplicado à equação da difusão com derivada segundo Riemann-Liouville* (53a) *é convergente se*

$$\mathcal{K}\frac{\Delta t}{\Delta x^2}\left(\frac{\tau}{\Delta t}\right)^{\alpha} \leq \frac{1}{2^{1+\alpha}}\,. \tag{71}$$

Demonstração. De acordo com o Teorema 4, o método $MtEP$ é consistente e, pelo Teorema 2, vimos que se a condição (71) for satisfeita, o método é estável. Portanto, sob as condições do teorema da equivalência de Lax-Richtmyer e aplicando os resultados, concluímos que o método $MtEP$ aplicado à equação de difusão fracionária com derivada de Riemann-Liouville é convergente. $\square$

3.2.3 Experimentos numéricos do $MtEP$ com Riemann-Liouville

Aqui, vamos apresentar alguns testes com o método do tipo Euler progressivo aplicado à equação da difusão fracionária com a derivada de Riemann-Liouville. Serão comparadas as soluções aproximadas com a solução analítica correspondente.

Experimento 3.1. *A função* $u(x,t) = t^{1,5}\exp(\pi(x-\alpha))$ *é a solução encontrada para o problema* (53)*, relativa ao domínio* $(x,t) \in [0,1]\times[0,1]$*, estando os dados usados descritos a seguir:*

- $f(x,t) = \left(1,5t^{0,5} - \pi^2 \frac{\Gamma(2,5)}{\Gamma(2,5-\alpha)} t^{1,5-\alpha}\right)\exp(\pi(x-\alpha))$;
- $\phi(x) = 0$; $\quad l(t) = t^{1,5}\exp(-\pi\alpha)$; $\quad r(t) = t^{1,5}\exp(\pi(1-\alpha))$.

A Tabela 1 apresenta os resultados encontrados para o erro na norma L^2*, com* $\mathcal{K} = 1$*,* $\Delta t = 1/40000$ *e* $\tau = 1$ *fixados e variando o número de passos na variável espacial* Δx*, como também a ordem da derivada fracionária* α*.*

Tabela 1 - Método tipo Euler progressivo com derivada fracionária de Riemann-Liouville.

M	$\alpha = 0,1$	$\alpha = 0,2$	$\alpha = 0,3$	$\alpha = 0,4$	$\alpha = 0,5$
10	$2,8564 \times 10^{-2}$	$2,0960 \times 10^{-2}$	$1,5387 \times 10^{-2}$	$1,1299 \times 10^{-2}$	NaN
20	$7,1664 \times 10^{-3}$	$5,2543 \times 10^{-3}$	$3,8541 \times 10^{-3}$	NaN	NaN
30	$3,1813 \times 10^{-3}$	$2,3294 \times 10^{-3}$	NaN	NaN	NaN
40	$1,7848 \times 10^{-3}$	$1,3044 \times 10^{-3}$	NaN	NaN	NaN
50	$1,1382 \times 10^{-3}$	NaN	NaN	NaN	NaN

Legenda: Erro na norma L^2 com $\mathcal{K} = 1$, $\Delta t = 1/40000$, $\tau = 1$, M é a quantidade de passos na variável x e α é a ordem da derivada fracionária.

Fonte: O autor, 2023.

Os resultados numéricos obtidos confirmam o resultado da análise numérica (65) *que exibe uma dependência entre* Δt*,* Δx*,* τ *e* α*, como apresentado na Tabela 2.*

Os resultados exibidos nas Tabelas 1 e 2 motivam a busca por outro esquema na abordagem do problema (53), visto que, mesmo tomando valores na ordem de 10^{-5} para

Tabela 2 - Condição de estabilidade do método tipo Euler progressivo com derivada fracionária de Riemann-Liouville.

M	$\frac{\Delta t^{0,9}}{\Delta x^2} \leq \frac{1}{2^{1,1}}$		$\frac{\Delta t^{0,8}}{\Delta x^2} \leq \frac{1}{2^{1,2}}$		$\frac{\Delta t^{0,7}}{\Delta x^2} \leq \frac{1}{2^{1,3}}$		$\frac{\Delta t^{0,6}}{\Delta x^2} \leq \frac{1}{2^{1,4}}$		$\frac{\Delta t^{0,5}}{\Delta x^2} \leq \frac{1}{2^{1,5}}$	
10	$0,0072$		$0,0208$		0,0601		0,1733		~~0,5000~~	
20	$0,0289$		$0,0833$		0,2402		~~0,6931~~		~~2,0000~~	
30	$0,0649$	$0,4665$	$0,1873$	$0,4353$	~~0,5405~~	0,4061	~~0,5596~~	0,3789	~~4,5000~~	0,3536
40	$0,1154$		$0,3330$		~~0,9609~~		~~2,7726~~		~~8,0000~~	
50	$0,1803$		~~0,5203~~		~~1,5014~~		~~4,3322~~		~~12,500~~	

Legenda: Verificação da condição de estabilidade (65) do método tipo Euler progressivo dos correspondentes testes exibidos na Tabela 1

Fonte: O autor, 2023.

Δt, não foi obtida convergência (NaN[17]) para valores de α maiores ou iguais a $0,5$. Além disso, a necessidade de escolher valores muito pequenos para Δt demanda um acréscimo significativo no tempo de processamento do código e também no uso da memória requerida, o que pode inviabilizar seu uso. A abordagem explícita não será mais utilizada para os próximos operadores fracionários, pois a condição de estabilidade torna impraticável a obtenção da solução numérica para valores de α próximos de um.

3.2.4 Algoritmo do $MtEP$ com Riemann-Liouville

Nesta seção, apresentamos o algoritmo do método de Euler progressivo aplicado à equação da difusão fracionária com derivada fracionária de Riemann-Liouville. Esse algoritmo foi estruturado com base na equação (57), e o código em linguagem Python está disponível no Apêndice B.1.

Método tipo Euler Progressivo – Riemann-Liouville

Propósito:

Solução numérica da equação da difusão fracionária com Riemann-Liouville.

$$\frac{\partial u(x,t)}{\partial t} = {}_{RL}D^{\alpha}_{0,t}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

[17] NaN (acrônimo em inglês para Not a Number) é símbolo usado nas linguagens para representar um valor numérico irrepresentável, com os vínculos do sistema operacional e do equipamento.

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema explícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

α: ordem da derivada fracionária;

τ: parâmetro de correção dimensional;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$r = \tau^\alpha \mathcal{K}(\Delta t)^{1-\alpha}/(\Delta x)^2$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $k = 0, 1, \ldots, N$ faça

$\omega_k = (-1)^k \binom{\alpha}{k}$;

Passo 6

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $A_i^j = -2r$;

se $i == j-1$ ou $i == j+1$, então $A_i^j = r$;
senão $A_i^j = 0$, (matriz tridiagonal);

Passo 7

Faça $n = 0$;
Enquanto $n < N$ faça
crie, V e W, vetores de ordem M;
para $k = 0, 1, \ldots, n$ faça
$V = A \times U^{n-k}$;
$V_0 = V_0 + r \times U_0^{n-k}$;
$V_{M-1} = V_{M-1} + r \times U_M^{n-k}$;
$W = W + \omega_k \times V$;
$U^{n+1} = U^n + W + \Delta t \times F^n$;
$n = n + 1$;

Passo 8

Para $n = 0, 1, \ldots, N$ faça
imprima U^n;

Passo 9

PARE (O procedimento está completo).

3.3 Método tipo Euler regressivo com Riemann-Liouville

Aqui, faremos a abordagem na equação da difusão fracionária pelo método tipo Euler regressivo ($MtER$). Neste esquema implícito, faremos a análise de estabilidade, convergência e unicidade, baseando-nos no artigo de (CHEN et al., 2007). Aplicamos em (53a) as aproximações (35), (38) e (54). Dessa forma, o problema (53) é escrito da seguinte maneira, com $i = 1, 2, \ldots, M-1$ e $n = 1, 2, \ldots, N$,

$$\frac{U_i^n - U_i^{n-1}}{\Delta t} = \frac{\tau^\alpha \mathcal{K}}{\Delta t^\alpha} \sum_{k=0}^{n} \omega(k) \left(\frac{U_{i-1}^{n-k} - 2U_i^{n-k} + U_{i+1}^{n-k}}{\Delta x^2} \right) + f_i^n, \tag{72a}$$

$$U_i^0 = \phi(x_i) \quad (i = 0, 1, \ldots, M), \tag{72b}$$

$$U_0^n = l(t_n) \quad (n = 0, 1, \ldots, N), \tag{72c}$$

$$U_M^n = r(t_n) \quad (n = 0, 1, \ldots, N), \tag{72d}$$

onde $\omega(k) = (-1)^k \binom{\alpha}{k}$ e $f_i^n = f(x_i, t_n)$ em (72a). A partir de (72a), escrevemos

$$U_i^n = U_i^{n-1} + r \sum_{k=0}^{n} \omega(k)(U_{i-1}^{n-k} - 2U_i^{n-k} + U_{i+1}^{n-k}) + \Delta t f_i^n, \tag{73}$$

com $r := \mathcal{K}\frac{\Delta t}{\Delta x^2}\left(\frac{\tau}{\Delta t}\right)^{\alpha}$. Para $1 \leq i \leq M-1$, a forma matricial para (73) é

$$\boldsymbol{U^n} = \boldsymbol{U^{n-1}} + \sum_{k=0}^{n} \omega(k)\left(A\boldsymbol{U^{n-k}} + \boldsymbol{C^{n-k}}\right) + \Delta t \boldsymbol{f^n}, \tag{74}$$

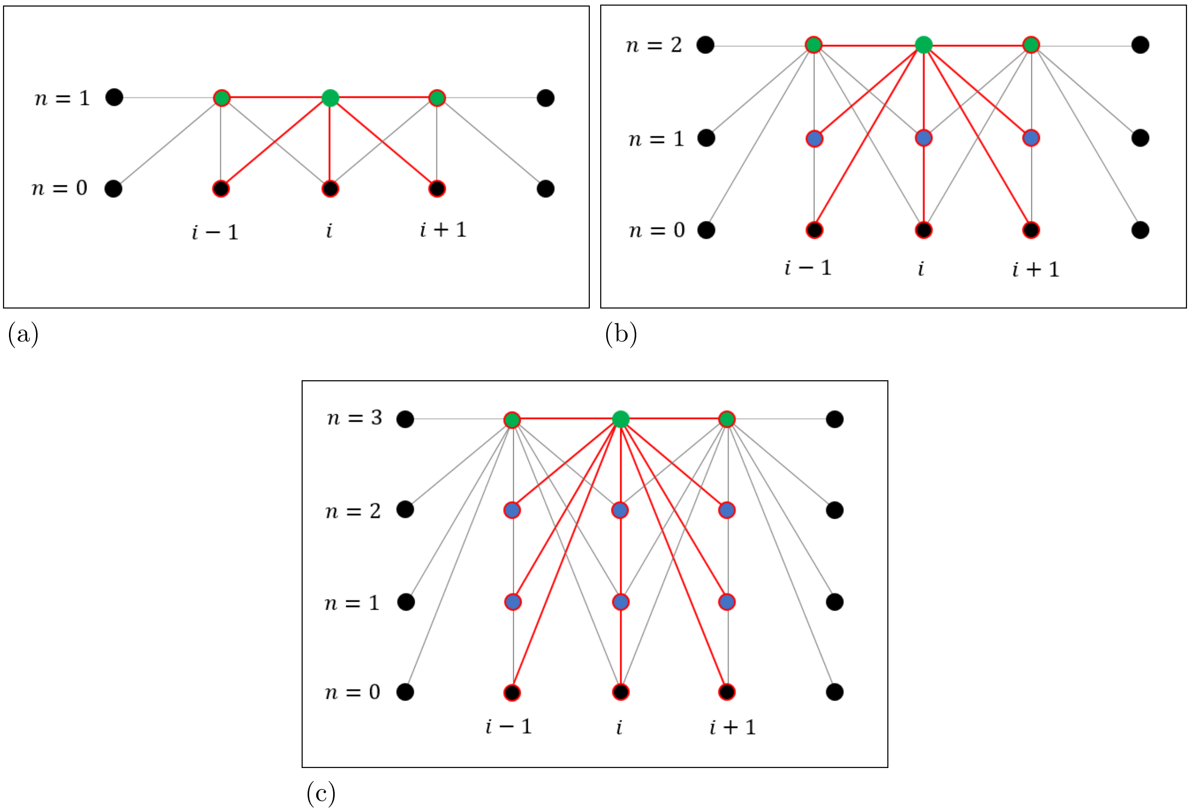
ou

$$\left(\boldsymbol{I} - A\right)\boldsymbol{U^n} = \boldsymbol{U^{n-1}} + \sum_{k=1}^{n} \omega(k)\left(A\boldsymbol{U^{n-k}} + \boldsymbol{C^{n-k}}\right) + \boldsymbol{C^n} + \Delta t \boldsymbol{f^n}, \tag{75}$$

onde os vetores $\boldsymbol{U^n}$, $\boldsymbol{C^n}$, $\boldsymbol{f^n}$ e a matriz A são iguais aos definidos como na Seção 3.2.

As Figuras 5a, 5b e 5c, exibem a dependência dos valores calculados com relação aos nós nos três primeiros níveis para o método tipo Euler regressivo, respectivamente. O estêncil formado pelas arestas vermelhas mostra os valores que precisam ser conhecidos para o cálculo do nó i no passo n. Esta visualização comprova o efeito de memória quando utilizamos a derivada fracionária.

Figura 5 - Estêncil do método tipo Euler regressivo com a derivada de Riemann-Liouvile.



Legenda: As figuras exibem os estênceis em vermelho na posição i e nos níveis: (a) $n = 1$, (b) $n = 2$ e (c) $n = 3$, para o problema (15), com a derivada segundo Riemann-Liouville.

Fonte: O autor, 2023.

3.3.1 Estabilidade do $MtER$ com Riemann-Liouville

Agora vamos realizar a análise de estabilidade do método tipo Euler regressivo (NEGREIROS; FARIA; MOURA, 2022). Primeiramente escrevemos (72a) da seguinte forma

$$u_i^n = u_i^{n-1} + r\sum_{k=0}^{n}\omega(k)(u_{i-1}^{n-k} - 2u_i^{n-k} + u_{i+1}^{n-k}) + \Delta t f_i^n, \tag{76}$$

onde $r := \mathcal{K}\frac{\Delta t}{\Delta x^2}\left(\frac{\tau}{\Delta t}\right)^{\alpha}$.

Seja U_i^n a solução aproximada do problema (72), definimos o erro de aproximação

$$\rho_i^n = u_i^n - U_i^n \quad (i = 1, 2, \ldots, M-1;\ n = 0, 1, \ldots, N)$$

e

$$\boldsymbol{\rho^n} := \left[\rho_1^n, \rho_2^n, \ldots, \rho_{M-1}^n\right]^t.$$

Obtemos a seguinte equação para o erro de arredondamento

$$\rho_i^n = \rho_i^{n-1} + r\sum_{k=0}^{n}\omega(k)(\rho_{i-1}^{n-k} - 2\rho_i^{n-k} + \rho_{i+1}^{n-k}). \tag{77}$$

Assumindo que a solução da equação (77) tem a forma $\rho_i^n = d_n\,\mathrm{e}^{j\sigma i\Delta x}$, onde $j^2 = -1$ e $\sigma = 2\pi m/L$, e substituindo em (77), temos

$$\begin{aligned}
d_n e^{j\sigma i\Delta x} &= d_{n-1}\,\mathrm{e}^{j\sigma i\Delta x} + r\sum_{k=0}^{n}\omega(k)\partial_x^2 d_{n-k}e^{j\sigma i\Delta x}\\
d_n e^{j\sigma i\Delta x} &= d_{n-1}\,\mathrm{e}^{j\sigma i\Delta x} + r\left(\mathrm{e}^{-j\sigma\Delta x} - 2 + \mathrm{e}^{j\sigma\Delta x}\right)\sum_{k=0}^{n}\omega(k)d_{n-k}\,\mathrm{e}^{j\sigma i\Delta x}\\
d_n &= d_{n-1} + r\left(2\cos(\sigma\Delta x) - 2\right)\sum_{k=0}^{n}\omega(k)d_{n-k}\\
d_n &= d_{n-1} - 4r\,\mathrm{sen}^2\left(\frac{\sigma\Delta x}{2}\right)\sum_{k=0}^{n}\omega(k)d_{n-k}.
\end{aligned} \tag{78}$$

Lema 1. *Os coeficientes* $\omega(k)$ *satisfazem:* $\omega(0) = 1$, $\omega(1) = -\alpha$, $\sum_{k=0}^{\infty}\omega(k) = 0$, $\omega(k) < 0$ *para* $(k = 1, 2, \ldots)$ *e* $-\sum_{k=1}^{n}\omega(k) < 1$.

Demonstração. Desenvolvendo os binômios e usando a relação $z! = \Gamma(z+1)$, temos

$\omega(0) = (-1)^0\binom{\alpha}{0} = \frac{\Gamma(\alpha+1)}{\Gamma(1)\Gamma(\alpha+1)} = \frac{\Gamma(\alpha+1)}{\Gamma(\alpha+1)} = 1;$

$\omega(1) = (-1)^1\binom{\alpha}{1} = -\frac{\Gamma(\alpha+1)}{\Gamma(2)\Gamma(\alpha-1+1)} = -\frac{\alpha\Gamma(\alpha)}{\Gamma(\alpha)} = -\alpha.$

Para obter o valor da série usaremos a regra de Stifel, a saber, $\binom{\alpha-1}{k-1} + \binom{\alpha-1}{k} = \binom{\alpha}{k}$

$$\begin{aligned}
\sum_{k=0}^{\infty} \omega(k) &= \binom{\alpha}{0} - \binom{\alpha}{1} + \binom{\alpha}{2} - \binom{\alpha}{3} + \dots \\
&= 1 - \left[\binom{\alpha-1}{0} + \binom{\alpha-1}{1}\right] + \left[\binom{\alpha-1}{1} + \binom{\alpha-1}{2}\right] - \left[\binom{\alpha-1}{2} + \binom{\alpha-1}{3}\right] + \dots \\
&= 1 - \binom{\alpha-1}{0} \\
&= 0.
\end{aligned}$$

Escrevendo recursivamente $\omega(k)$ temos: $\omega(k) = \omega(k-1)(k-\alpha-1)/k$, para $(k = 2, 3, \dots)$. Agora, como $\omega(1) < 0$ e $k > \alpha + 1$, para $(k = 2, 3, \dots)$, logo $\omega(k) < 0$ $(k = 1, 2, \dots)$.

Sabe-se que $\sum_{k=1}^{\infty} \omega(k) = -1$, pois $\sum_{k=0}^{\infty} \omega(k) = 1 + \sum_{k=1}^{\infty} \omega(k) = 0$, mas como $\omega(k) < 0$ $(k = 1, 2, \dots)$, temos $\sum_{k=1}^{n} \omega(k) > -1$ ou $-\sum_{k=1}^{n} \omega(k) < 1$. □

Aplicando o Lema 1 em (78), temos

$$\begin{aligned}
d_n &= d_{n-1} - \widetilde{r}\left(d_n - \alpha d_{n-1} + \sum_{k=2}^{n} \omega(k) d_{n-k}\right) \\
d_n &= \frac{1 + \alpha\widetilde{r}}{1 + \widetilde{r}} d_{n-1} - \frac{\widetilde{r}}{1 + \widetilde{r}} \sum_{k=2}^{n} \omega(k) d_{n-k} \qquad (79)
\end{aligned}$$

onde $\widetilde{r} := 4r\,\mathrm{sen}^2\left(\frac{\sigma\Delta x}{2}\right)$.

Proposição 1. *A partir da equação* (79)*, se tem* $|d_n| \leq |d_0|$*,* $(n = 1, 2, \dots, N)$*.*

Demonstração. A prova dar-se-á por indução matemática. Quando $n = 1$ em (79), temos

$$d_1 = \frac{1 + \alpha\widetilde{r}}{1 + \widetilde{r}} d_0.$$

Note que $0 < \alpha < 1$ e $\widetilde{r} > 0$, o que nos leva a

$$|d_1| = \frac{1 + \alpha\widetilde{r}}{1 + \widetilde{r}} |d_0| \leq |d_0|.$$

Supomos que $|d_l| \leq |d_0|$, $\quad l = 1, 2, \dots, n-1$ e aplicando o Lema 1, temos

$$\begin{aligned}
|d_n| &= \left| \frac{1+\alpha\widetilde{r}}{1+\widetilde{r}} d_{n-1} - \frac{\widetilde{r}}{1+\widetilde{r}} \sum_{k=2}^{n} \omega(k) d_{n-k} \right| \\
&\leq \frac{1+\alpha\widetilde{r}}{1+\widetilde{r}} |d_{n-1}| + \frac{\widetilde{r}}{1+\widetilde{r}} \sum_{k=2}^{n} |\omega(k)||d_{n-k}| \\
&\leq \left[\frac{1+\alpha\widetilde{r}}{1+\widetilde{r}} + \frac{\widetilde{r}}{1+\widetilde{r}} \sum_{k=2}^{n} |\omega(k)| \right] |d_0| \\
&= \left[\frac{1+\alpha\widetilde{r}}{1+\widetilde{r}} + \frac{\widetilde{r}}{1+\widetilde{r}} \left(-\sum_{k=1}^{n} \omega(k) - \alpha \right) \right] |d_0| \\
&\leq \left[\frac{1+\alpha\widetilde{r}}{1+\widetilde{r}} + \frac{\widetilde{r}}{1+\widetilde{r}} (1-\alpha) \right] |d_0| \\
&= |d_0|.
\end{aligned}$$

□

Definimos a função ρ^n $(n = 1, 2, \ldots, N)$ na malha computacional

$$\rho^n(x) = \begin{cases} \rho_i^n, & \text{se} \quad x_i - \frac{\Delta x}{2} < x \leq x_i + \frac{\Delta x}{2}, \; i = 1, 2, \ldots, M-1 \\ 0, & \text{se} \quad 0 \leq x \leq \frac{\Delta x}{2} \text{ ou } L - \frac{\Delta x}{2} < x \leq L, \end{cases}$$

que pode ser expandida em série de Fourier da seguinte forma

$$\rho^n(x) = \sum_{m=-\infty}^{\infty} d_n(m)\, \mathrm{e}^{j2\pi m x/L} \quad (n = 1, 2, \ldots, N),$$

onde

$$d_n(m) = \frac{1}{L} \int_0^L \rho^n(x)\, \mathrm{e}^{j2\pi m x/L}\, dx.$$

Introduzindo a seguinte norma:

$$\begin{aligned}
\|\rho^n\|_2 &= \left(\Delta x \sum_{i=1}^{M-1} |\rho_i^n|^2 \right)^{1/2} \\
&= \left(\int_0^{\frac{\Delta x}{2}} |\rho^n(x)|^2\, dx + \sum_{i=1}^{M-1} \int_{x_i - \frac{\Delta x}{2}}^{x_i + \frac{\Delta x}{2}} |\rho^n(x)|^2\, dx + \int_{L-\frac{\Delta x}{2}}^{L} |\rho^n(x)|^2\, dx \right)^{1/2} \\
&= \left(\int_0^L |\rho^n(x)|^2\, dx \right)^{1/2} \qquad (80)
\end{aligned}$$

e aplicando a identidade de Parseval

$$\int_0^L |\rho^n(x)|^2\, dx = \sum_{m=-\infty}^{\infty} |d_n(m)|^2,$$

obtemos

$$\|\rho^n\|_2^2 = \sum_{m=-\infty}^{\infty} |d_n(m)|^2. \tag{81}$$

Teorema 6. *O método implícito tipo Euler regressivo* (72) *é incondicionalmente estável.*

Demonstração. Aplicando a Proposição 1 e usando (81), temos

$$\|\rho^n\|_2 \leq \left\|\rho^0\right\|_2, \quad n = 1, 2, \ldots, N,$$

o que prova que o método tipo Euler regressivo (72) é incondicionalmente estável. □

Assim como ocorre com o método de Euler regressivo quando aplicado à equação da difusão clássica, o $MtER$ é incondicionalmente estável. Isso significa que sua estabilidade não depende da configuração da malha, da ordem α nem do parâmetro de correção dimensional τ.

3.3.2 Convergência do $MtER$ com Riemann-Liouville

Aqui, primeiramente introduzimos o seguinte lema.

Lema 2. $\Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) = \frac{1}{\Gamma(1-\alpha)} + O(\Delta t)$.

Demonstração. Pela definição,

$$_{RL}D^{\alpha}_{0,t} u(t)\Big|_{t=t_n} = \Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) u(t_{n-k}) + O(\Delta t). \tag{82}$$

Tomando $u(t) = 1$ e $t_n = 1$ em (82), temos

$$_{RL}D^{\alpha}_{0,t}(1)\Big|_{t=1} = \Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) + O(\Delta t),$$

ou

$$\Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) = \ _{RL}D^{\alpha}_{0,t}(1)\Big|_{t=1} + O(\Delta t).$$

Pela definição (51), chegamos a

$$\begin{aligned} _{RL}D^{\alpha}_{0,t}(1) &= \frac{1}{\Gamma(1-\alpha)} \frac{d}{dt} \int_0^t (t-s)^{-\alpha} ds \\ &= \frac{1}{\Gamma(1-\alpha)} \frac{d}{dt} \frac{t^{1-\alpha}}{1-\alpha} \\ &= \frac{1}{\Gamma(1-\alpha)} t^{-\alpha}. \end{aligned} \tag{83}$$

Para $t = 1$, temos $1/\Gamma(1-\alpha)$, o que conclui a prova. $\square$

Agora, definimos

$$R_i^n := \delta_t^- U_i^n - \Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) \delta_x^2 U_i^{n-k} - f_i^n \tag{84}$$

A partir das aproximações numéricas das derivadas e do Lema 2, desenvolvemos

$$\begin{aligned}
\Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) \delta_x^2 U_i^{n-k} &= \Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) \left(\frac{\partial^2 u(x_i, t_n)}{\partial x^2} + O(\Delta x^2) \right) \\
&= \Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) \frac{\partial^2 u(x_i, t_n)}{\partial x^2} + O(\Delta x^2) \Delta t^{-\alpha} \sum_{k=0}^{n} \omega(k) \\
&= {}_{RL}D_{0,t}^{\alpha} \frac{\partial^2 u(x_i, t_n)}{\partial x^2} + O(\Delta t) + O(\Delta x^2) \left(\frac{1}{\Gamma(1-\alpha)} + O(\Delta t) \right) \\
&= {}_{RL}D_{0,t}^{\alpha} \frac{\partial^2 u(x_i, t_n)}{\partial x^2} + O(\Delta t + \Delta x^2),
\end{aligned}$$

e como

$$\delta_t^- U_i^n = \frac{\partial u(x_i, t_n)}{\partial t} + O(\Delta t),$$

temos consequentemente

$$R_i^n = O(\Delta t + \Delta x^2) \quad (n = 1, 2, \ldots, N;\ i = 1, 2, \ldots, ..., M-1).$$

Portanto, existe uma constante c_1 tal que

$$|R_i^n| \leq c_1(\Delta t + \Delta x^2), \tag{85}$$

onde $c_1 := \max \left\{ c_1^{(n,i)} \right\}$, para todo $(n = 1, 2, \ldots, N;\ i = 1, 2, \ldots, ..., M-1)$. Seja

$$\mathrm{e}_i^n = u(x_i, t_n) - u_i^n,\ n = 1, 2, \ldots, N;\ i = 1, 2, \ldots, M-1$$

e

$$\mathrm{e}^n := \left[\mathrm{e}_1^n, \mathrm{e}_2^n, \ldots, \mathrm{e}_{M-1}^n\right]^t,\ R^n := \left[R_1^n, R_2^n, \ldots, R_{M-1}^n\right]^t.$$

De (84), para $n = 1, 2, \ldots, N$ e $i = 1, 2, \ldots, M-1$, temos

$$u(x_i, t_n) = u(x_i, t_{n-1}) + r \sum_{k=0}^{n} \omega(k) \delta_x^2 U_i^{n-k} + \Delta t f_i^n + \Delta t R_i^n.$$

Subtraindo a equação acima de (76), obtemos

$$e_i^n = e_i^{n-1} + r\sum_{k=0}^{n}\omega(k)(e_{i-1}^{n-k} - 2e_i^{n-k} + e_{i+1}^{n-k}) + \Delta t R_i^n. \tag{86}$$

Analisamos a convergência do problema (72) usando o método de Fourier e seguindo passos semelhantes aos da análise de estabilidade. Definimos as funções na malha:

$$\mathrm{e}^n(x) = \begin{cases} \mathrm{e}_i^n, & \text{se} \quad x_i - \frac{\Delta x}{2} < x \leq x_i + \frac{\Delta x}{2},\ i = 1, 2, \ldots, M-1 \\ 0, & \text{se} \quad 0 \leq x \leq \frac{\Delta x}{2} \text{ ou } L - \frac{\Delta x}{2} < x \leq L, \end{cases} \quad (n = 0, 1, \ldots, N)$$

e

$$R^n(x) = \begin{cases} R_i^n, & \text{se} \quad x_i - \frac{\Delta x}{2} < x \leq x_i + \frac{\Delta x}{2},\ i = 1, 2, \ldots, M-1 \\ 0, & \text{se} \quad 0 \leq x \leq \frac{\Delta x}{2} \text{ ou } L - \frac{\Delta x}{2} < x \leq L, \end{cases} \quad (n = 0, 1, \ldots, N)$$

que podem ser expandidas em série de Fourier da seguinte forma:

$$\mathrm{e}^n(x) = \sum_{m=-\infty}^{\infty} \xi_n(m)\, \mathrm{e}^{j2\pi m x/L} \quad (n = 1, 2, \ldots, N)$$

e

$$R^n(x) = \sum_{m=-\infty}^{\infty} \eta_n(m)\, \mathrm{e}^{j2\pi m x/L} \quad (n = 1, 2, \ldots, N),$$

onde

$$\xi_n(m) = \frac{1}{L}\int_0^L \mathrm{e}^n(x)\, \mathrm{e}^{-j2\pi m x/L}\, dx$$

e

$$\eta_n(m) = \frac{1}{L}\int_0^L R^n(x)\, \mathrm{e}^{-j2\pi m x/L}\, dx.$$

Da mesma forma, também temos

$$\|\mathrm{e}^n\|_2^2 = \left(\Delta x \sum_{i=1}^{M-1} |\mathrm{e}_i^n|^2\right)^{1/2} = \sum_{m=-\infty}^{\infty} |\xi_n(m)|^2 \quad (n = 0, 1, \ldots, N) \tag{87}$$

e

$$\|R^n\|_2^2 = \left(\Delta x \sum_{i=1}^{M-1} |R_i^n|^2\right)^{1/2} = \sum_{m=-\infty}^{\infty} |\eta_n(m)|^2 \quad (n = 0, 1, \ldots, N). \tag{88}$$

Com base na análise acima, podemos supor que

$$\mathrm{e}_i^n = \xi_n\, \mathrm{e}^{j\sigma i \Delta x}$$

e

$$R_i^n = \eta_n\, \mathrm{e}^{j\sigma i \Delta x}.$$

Substituindo as expressões acima em (86), obtemos

$$\begin{aligned}
\xi_n e^{j\sigma i\Delta x} &= \xi_{n-1}\,\mathrm{e}^{j\sigma i\Delta x} + r\sum_{k=0}^{n}\omega(k)\delta_x^2\xi_{n-k}e^{j\sigma i\Delta x} + \Delta t\eta_n e^{j\sigma i\Delta x}\\
\xi_n &= \xi_{n-1} + r\left(2\cos\left(\sigma\Delta x\right)-2\right)\sum_{k=0}^{n}\omega(k)\xi_{n-k} + \Delta t\eta_n\\
\xi_n &= \xi_{n-1} - 4r\,\mathrm{sen}^2\left(\frac{\sigma\Delta x}{2}\right)\sum_{k=0}^{n}\omega(k)\xi_{n-k} + \Delta t\eta_n
\end{aligned}$$

ou

$$(1+\widetilde{r})\xi_n = (1-\omega(1)\widetilde{r})\xi_{n-1} - \widetilde{r}\sum_{k=2}^{n}\omega(k)\xi_{n-k} + \Delta t\eta_n. \tag{89}$$

Aplicando o Lema 1 em (89) podemos escrever

$$\xi_n = \frac{1+\alpha\widetilde{r}}{1+\widetilde{r}}\xi_{n-1} - \frac{\widetilde{r}}{1+\widetilde{r}}\sum_{k=2}^{n}\omega(k)\xi_{n-k} + \frac{\Delta t\eta_n}{1+\widetilde{r}} \quad (n=1,2,\ldots,N). \tag{90}$$

Proposição 2. *Suponha que* ξ_n $(n=1,2,\ldots,N)$ *é a solução de* (90)*, então existe uma constante positiva* c_2 *tal que*

$$|\xi_n| \le n\Delta t c_2|\eta_1|, \quad n=1,2,\ldots,N.$$

Demonstração. Primeiro, notando que $\mathrm{e}^0 = 0$, temos

$$\xi_0 \equiv \xi_0(m) = 0.$$

De $|R_i^n| \le c_1(\Delta t + \Delta x^2)$ e de (88), temos

$$\|R^n\|_2 \le c_1\sqrt{M\Delta x}(\Delta t+\Delta x^2) = c_1\sqrt{L}(\Delta t+\Delta x^2), \quad n=1,2,\ldots,N. \tag{91}$$

Com base na convergência da série $\sum_{m=-\infty}^{\infty}|\eta_n(m)|^2$, existe uma constante positiva $c_2^{(n)}$, tal que

$$|\eta_n| \equiv |\eta_n(m)| \le c_2^{(n)}|\eta_1| \equiv c_2^{(n)}|\eta_1(m)|. \tag{92}$$

Portanto, temos

$$|\eta_n| \le c_2|\eta_1|, \quad n=1,2,\ldots,N \tag{93}$$

onde

$$c_2 = \max_{1\le n\le N}\left(c_2^{(n)}\right). \tag{94}$$

Vamos completar a prova usando indução matemática. Para k = 1, de (90), temos

$$\xi_1 = \frac{1+\alpha\widetilde{r}}{1+\widetilde{r}}\xi_0 + \frac{\Delta t\eta_n}{1+\widetilde{r}} = \frac{\Delta t\eta_n}{1+\widetilde{r}}.$$

Por (92), chegamos a

$$|\xi_1| \leq \Delta t|\eta_1| \leq c_2\Delta t|\eta_1|.$$

Suponha que

$$|\xi_p| \leq c_2 p\Delta t|\eta_1|, \quad p = 1, 2, \ldots, n-1$$

e, notando que $0 < \alpha < 1$, $\widetilde{r} > 0$ e (94), de (90), aplicando o Lema 1, temos

$$\begin{aligned}
|\xi_n| &= \left|\frac{1+\alpha\widetilde{r}}{1+\widetilde{r}}\xi_{n-1} - \frac{\widetilde{r}}{1+\widetilde{r}}\sum_{k=2}^{n}\omega(k)\xi_{n-k} + \frac{\Delta t}{1+\widetilde{r}}\eta_n\right| \\
&\leq \frac{1+\alpha\widetilde{r}}{1+\widetilde{r}}|\xi_{n-1}| + \frac{\widetilde{r}}{1+\widetilde{r}}\sum_{k=2}^{n}|\omega(k)||\xi_{n-k}| + \frac{\Delta t}{1+\widetilde{r}}|\eta_n| \\
&\leq \left[\frac{1+\alpha\widetilde{r}}{1+\widetilde{r}}(n-1) + \frac{\widetilde{r}}{1+\widetilde{r}}\sum_{k=2}^{n}|\omega(k)|(n-k) + \frac{1}{1+\widetilde{r}}\right] c_2\Delta t|\eta_1| \\
&\leq \left[\frac{1+\alpha\widetilde{r}}{1+\widetilde{r}}(n-1) + \frac{\widetilde{r}}{1+\widetilde{r}}(n-1)\sum_{k=2}^{n}|\omega(k)| + \frac{1}{1+\widetilde{r}}\right] c_2\Delta t|\eta_1| \\
&= \left[\frac{1+\alpha\widetilde{r}}{1+\widetilde{r}}(n-1) + \frac{\widetilde{r}}{1+\widetilde{r}}(n-1)\left(\sum_{k=1}^{n}|\omega(k)| - \alpha\right) + \frac{1}{1+\widetilde{r}}\right] c_2\Delta t|\eta_1| \\
&\leq \left[\frac{1+\alpha\widetilde{r}}{1+\widetilde{r}}(n-1) + \frac{\widetilde{r}}{1+\widetilde{r}}(n-1)(1-\alpha) + \frac{1}{1+\widetilde{r}}\right] c_2\Delta t|\eta_1| \\
&= \left[n - 1 + \frac{1}{1+\widetilde{r}}\right] c_2\Delta t|\eta_1| \\
&\leq (n-1+1)\, c_2\Delta t|\eta_1| \\
&= c_2 n\Delta t|\eta_1|.
\end{aligned}$$

□

Teorema 7. *O método implícito tipo Euler regressivo* (72) *é convergente na norma* L_2 *e a ordem de convergência é* $O(\Delta t + \Delta x^2)$.

Demonstração. Aplicando a Proposição 2 e (91), e observando (87) e (88), obtemos

$$\|\mathrm{e}^n\|_2 \leq c_2 n\Delta t \left\|R^1\right\|_2 \leq c_1 c_2 n\Delta t\sqrt{L}(\Delta t + \Delta x^2).$$

Como $n\Delta t \leq T$, temos

$$\|\mathrm{e}^n\|_2 < c(\Delta t + \Delta x^2),$$

onde $c = c_1 c_2 T\sqrt{L}$. Isto completa a demonstração. □

3.3.3 Unicidade do método tipo Euler regressivo

A unicidade da solução de um método numérico é uma propriedade fundamental, pois é desejável garantir que, para um determinado conjunto de entradas, a solução do problema seja única. Nesse contexto, a unicidade da solução está relacionada à consistência matemática do método e à sua capacidade de convergir para uma solução única. Para $1 \leq i \leq M-1$, a forma matricial para (73) é

$$\boldsymbol{U^n} = \boldsymbol{U^{n-1}} + \sum_{k=0}^{n} \omega(k) \left(A\boldsymbol{U^{n-k}} + \boldsymbol{C^{n-k}} \right) + \Delta t \boldsymbol{f^n}, \tag{95}$$

ou

$$(I - A)\boldsymbol{U^n} = \boldsymbol{U^{n-1}} + \sum_{k=1}^{n} \omega(k) \left(A\boldsymbol{U^{n-k}} + \boldsymbol{C^{n-k}} \right) + \boldsymbol{C^n} + \Delta t \boldsymbol{f^n}, \tag{96}$$

ou ainda

$$A'\boldsymbol{U^n} = b, \tag{97}$$

onde $b := \boldsymbol{U^{n-1}} + \sum_{k=1}^{n} \omega(k) \left(A\boldsymbol{U^{n-k}} + \boldsymbol{C^{n-k}} \right) + \boldsymbol{C^n} + \Delta t \boldsymbol{f^n}$ e

$$A' := \begin{pmatrix} 1+2r & -r & 0 & \dots & 0 \\ -r & 1+2r & -r & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -r & 1+2r & -r \\ 0 & \dots & 0 & -r & 1+2r \end{pmatrix}.$$

Teorema 8. *A solução da equação* (97) *é única.*

Demonstração. Como $r > 0$, então a matriz do sistema linear (97) é uma matriz estritamente diagonalmente dominante. Portanto, A' é uma matriz não singular, o que prova o teorema. □

3.3.4 Experimentos numéricos do *MtER* com Riemann-Liouville

Nesta seção, vamos mostrar alguns testes do problema da difusão fracionária com a derivada de Riemann-Liouville quando aplicado o método tipo Euler regressivo.

Experimento 3.2. *Os dados utilizados neste problema são idênticos aos do Experimento 3.1. Este teste visa verificar se a ordem relativa à variável temporal é preservada pela norma L_2.*

Experimento 3.3. *Agora, vamos trabalhar uma segunda aplicação, onde as Figuras 6a, 6b e 6c exibem soluções numéricas do problema* (53) *no domínio* $(x,t) \in [0,1] \times [0,2]$. *Os*

Tabela 3 - Método tipo Euler regressivo com derivada fracionária de Riemann-Liouville.

N	$\alpha = 0,2$	ordem	$\alpha = 0,5$	ordem	$\alpha = 0,8$	ordem
8	$3,2701 \times 10^{-2}$		$4,0344 \times 10^{-2}$		$2,5335 \times 10^{-2}$	
16	$1,6365 \times 10^{-2}$	0,9987	$2,0271 \times 10^{-2}$	0,9929	$1,2959 \times 10^{-2}$	0,9672
32	$8,1665 \times 10^{-3}$	1,0008	$1,0148 \times 10^{-2}$	0,9956	$6,5703 \times 10^{-3}$	0,9736
64	$4,0723 \times 10^{-3}$	1,0018	$5,0717 \times 10^{-3}$	0,9973	$3,3127 \times 10^{-3}$	0,9784
128	$2,0305 \times 10^{-3}$	1,0024	$2,5333 \times 10^{-3}$	0,9983	$1,6645 \times 10^{-3}$	0,9820

Legenda: Erro baseado na norma L^2 com $\Delta x = 1/1000$, $\mathcal{K} = 1$, $\tau = 1$; N é a quantidade de passos na variável t e α a ordem da derivada fracionária.

Fonte: O autor, 2023.

parâmetros a seguir foram fixados em $\mathcal{K} = 1$, $\Delta x = 1/100$ *e* $\Delta t = 2/400$, *considerando os seguintes dados de entrada:*

- $f(x,t) = (1 + \pi^2 t)(1 + t) \exp(t) \operatorname{sen}(\pi x)$;
- $\phi(x) = 0; \quad l(t) = 0; \quad r(t) = t \exp(t)$.

O efeito de memória das derivadas fracionárias justifica escolhas reduzidas do número de passos na variável temporal, — o que não é possível na abordagem explícita —, visto que, no problema abordado (53), a derivada fracionária atua na variável temporal e sua aproximação numérica é dada por (54), que possui um somatório dependente do número de passos na variável temporal.

3.3.5 Algoritmo do $MtER$ com Riemann-Liouville

Aqui, apresentamos o algoritmo do método de Euler regressivo aplicado à equação da difusão fracionária com derivada fracionária segundo Riemann-Liouville. Esse algoritmo foi construído a partir da equação (75), o código escrito na linguagem Python está disponível no Apêndice B.2

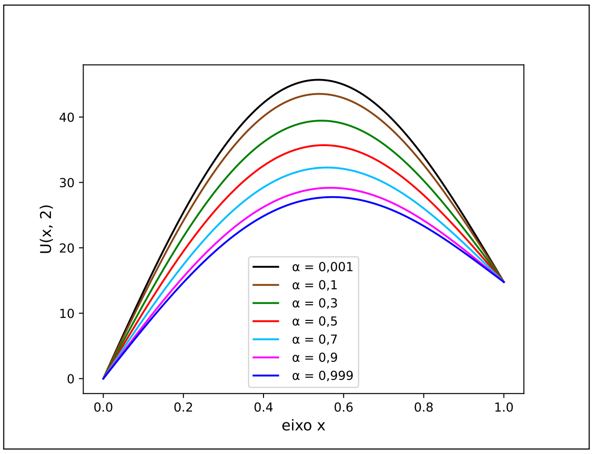
Método tipo Euler Regressivo – Riemann-Liouville

Propósito:

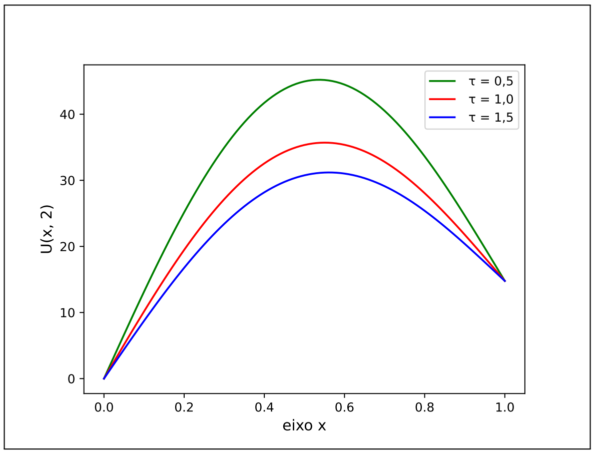
Solução numérica da equação da difusão fracionária com Riemann-Liouville

$$\frac{\partial u(x,t)}{\partial t} = {}_{RL}D_{0,t}^{\alpha} \left(\tau^{\alpha} \mathcal{K} \frac{\partial^2 u(x,t)}{\partial x^2} \right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

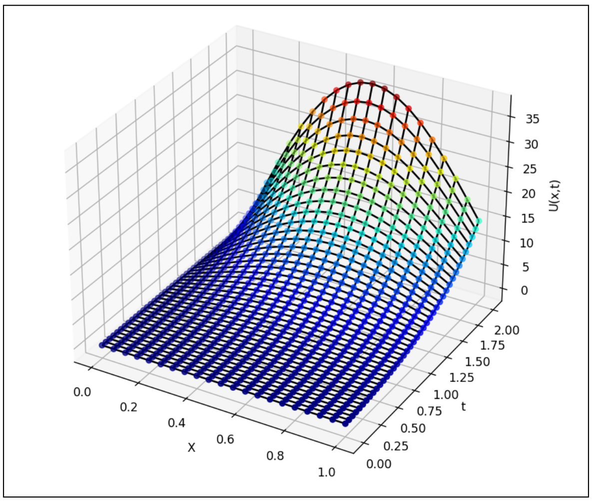
Figura 6 - Solução numérica da equação de difusão fracionária com a derivada de Riemann-Liouville.



(a)



(b)



(c)

Legenda: As Figuras (a) e (b) exibem cortes das soluções em $t = 2$. Na primeira figura, variamos o parâmetro α enquanto mantemos $\tau = 1$. Na segunda figura, fixamos $\alpha = 0,5$ e variamos o valor de τ. Já na Figura (c), é apresentado o gráfico de superfície para $\alpha = 0,5$ e $\tau = 1$.

Fonte: O autor, 2023.

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema implícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

α: ordem da derivada fracionária;

τ: parâmetro de correção dimensional;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$r = \tau^\alpha \mathcal{K}(\Delta t)^{1-\alpha}/(\Delta x)^2$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $k = 0, 1, \ldots, N$ faça

$\omega_k = (-1)^k \binom{\alpha}{k}$;

Passo 6

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $A_i^j = -2r$;

se $i == j-1$ ou $i == j+1$, então $A_i^j = r$;
se não $A_i^j = 0$, (matriz tridiagonal);

Passo 7

Crie I, matriz identidade de ordem M;
Faça $n = 1$;
Enquanto $n \leq N$ faça
crie, V e W, vetores de ordem M;
para $k = 1, 2, \ldots, n$ faça
$V = A \times U^{n-k}$;
$V_0 = V_0 + r \times U_0^{n-k}$;
$V_{M-1} = V_{M-1} + r \times U_M^{n-k}$;
$W = W + \omega_k \times V$;
resolva, $(I - A)U^n = U^{n-1} + W + \Delta t \times F^n$;
$n = n + 1$;

Passo 8

Para $n = 0, 1, \ldots, N$ faça
imprima U^n;

Passo 9

PARE (O procedimento está completo).

3.4 Difusão fracionária com Caputo

A derivada fracionária proposta em 1969 pelo matemático italiano Michele Caputo, em seu livro (CAPUTO, 1969) é também classificada como uma derivada fracionária clássica. Essa formulação foi introduzida objetivando fugir do inconveniente em que a derivada de uma função constante é diferente de zero, como ocorre com a derivada fracionária segundo Riemann-Liouville.

A definição da derivada fracionária de Caputo é baseada e semelhante de certa forma àquela proposta por Riemann-Liouville, contudo, é mais restritiva, uma vez que exige que a função $f(t)$ seja pelo menos n vezes derivável.

Definição 12. *Seja $[a, b]$ um intervalo finito do eixo real $\mathbb{R}$. Para $\alpha \in \mathbb{C}$ $(Re(\alpha) \geq 0)$ as derivadas fracionárias de Caputo, à esquerda ${}_CD^{\alpha}_{a,t}f(t)$ e à direita ${}_CD^{\alpha}_{t,b}f(t)$, de ordem α, são definidas por*

$$ {}_CD^{\alpha}_{a,t}f(t) := \frac{1}{\Gamma(n-\alpha)} \int_a^t \frac{f^{(n)}(s)}{(t-s)^{\alpha-n+1}} ds \tag{98}$$

e

$$ {}_CD^{\alpha}_{t,b}f(t) := \frac{(-1)^n}{\Gamma(n-\alpha)} \int_t^b \frac{f^{(n)}(s)}{(s-t)^{\alpha-n+1}} ds, \tag{99}$$

respectivamente, onde m é um inteiro positivo satisfazendo $m - 1 < \alpha \leq m$.

Uma consequência imediata do fato de que aplicar as derivadas de Caputo a uma função constante resulta em obter zero como resultado, para todo $\alpha > 0$, é que as equações diferenciais contendo a derivada de Caputo demandam condições iniciais usuais. Já para derivadas de Riemann-Liouville, tal fato não ocorre, a solução costuma ser singular no tempo inicial, exigindo condições iniciais não usuais e de interpretação não clara.

A formulação fracionária de Caputo, assim como a de Riemann-Liouville, cumpre as cinco propriedades do critério proposto por Ortigueira e Machado (TEODORO, 2019).

A derivada de Riemann-Liouville e a derivada de Caputo de uma função f obedecem as seguintes relações (PODLUBNY, 1998):

$$ {}_{RL}D^{\alpha}_{a,t}f(t) \; = \; {}_{C}D^{\alpha}_{a,t}f(t) + \sum_{k=0}^{m-1} \frac{f^{(k)}(a)(t-a)^{k-\alpha}}{\Gamma(k+1-\alpha)}, \tag{100} $$

onde $m - 1 < \alpha < m$, m é um inteiro positivo, $f \in C^{m-1}[a, t]$ e $f^{(m)}$ é integrável em $[a, t]$. Além disso, se $f \in C^{m}[a, t]$, então aplicando a expansão da série de Taylor em (100), chegamos a

$$ {}_{RL}D^{\alpha}_{a,t}\left[f(t) - \phi(t)\right] \; = \; {}_{C}D^{\alpha}_{a,t}f(t) \tag{101} $$

onde $\phi(t) = \sum_{k=0}^{m-1} \frac{f^{(k)}(a)}{\Gamma(k+1)}(t-a)^k$. Essas relações (100) e (101) entre os operadores de derivada de Riemann-Liouville e Caputo permitem focar nas propriedades e resultados de um deles.

O problema de valores iniciais e de contorno da equação de difusão fracionária, que descreve a difusão anômala com a derivada de Caputo, é

$$ \frac{\partial u(x,t)}{\partial t} = {}_{C}D^{\alpha}_{0,t}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T, \tag{102a} $$

$$ u(x,0) = \phi(x), \quad 0 \leq x \leq L, \tag{102b} $$

$$ u(0,t) = l(t), \quad 0 \leq t \leq T, \tag{102c} $$

$$ u(L,t) = r(t), \quad 0 \leq t \leq T, \tag{102d} $$

onde $0 < \alpha < 1$ é a ordem da derivada fracionária; $\phi(x)$, $l(t)$, $r(t)$ e $f(x,t)$ são funções suficientemente regulares dadas.

O método denominado L1 foi originalmente introduzido para aproximar a derivada de Riemann-Liouville com $0 < \alpha < 1$ (OLDHAM; SPANIER, 1974). Todavia, pela relação (101), pode-se escrever

$$ {}_{RL}D^{\alpha}_{a,t}f(t) \; = \; {}_{C}D^{\alpha}_{a,t}f(t) + \frac{f(0)}{\Gamma(1-\alpha)}t^{-\alpha}. $$

Sendo $t = t_n$, obtém-se

$$
\begin{aligned}
\left[{}_C D^{\alpha}_{a,t} f(t)\right]_{t=t_n} &= \frac{1}{\Gamma(1-\alpha)} \int_{t_0}^{t_n} (t_n - s)^{-\alpha} f'(s) ds \\
&= \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} (t_n - s)^{-\alpha} f'(s) ds \\
&\approx \frac{1}{\Gamma(1-\alpha)} \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} (t_n - s)^{-\alpha} \frac{f(t_{k+1}) - f(t_k)}{\Delta t} ds \\
&= \frac{1}{\Delta t^{\alpha}} \sum_{k=0}^{n-1} b_{n-k-1} \left(f(t_{k+1} - f(t_k)) \right), \qquad (103)
\end{aligned}
$$

onde $t_0 = 0$ e $b_k = \frac{1}{\Gamma(2-\alpha)} \left[(k+1)^{1-\alpha} - k^{1-\alpha} \right]$. Portanto, deduz-se que

$$
\left[{}_{RL} D^{\alpha}_{a,t} f(t)\right]_{t=t_n} \approx \frac{f(0)}{\Gamma(1-\alpha)} t^{-\alpha} + \frac{1}{\Delta t^{\alpha}} \sum_{k=0}^{n-1} b_{n-k-1} \left(f(t_{k+1} - f(t_k)) \right) \qquad (104)
$$

O método L1 (104) possui a seguinte estimativa de erro

$$
\left| \frac{f(0)}{\Gamma(1-\alpha)} t^{-\alpha} + \frac{1}{\Delta t^{\alpha}} \sum_{k=0}^{n-1} b_{n-k-1} \left(f(t_{k+1} - f(t_k)) \right) - \left[{}_{RL} D^{\alpha}_{a,t} f(t)\right]_{t=t_n} \right| \leq C \Delta t^{2-\alpha},
$$

onde C é uma constante positiva dependente somente de α e f (LANGLANDS; HENRY, 2005).

3.5 Método tipo Euler regressivo com Caputo

Nesta seção, desenvolveremos o método tipo Euler regressivo aplicado ao problema (102). Para construir esse método implícito aplicamos as aproximações (35) e (38) para as derivadas inteiras e o esquema L1 (103) na derivada fracionária em (102). Obtemos

$$
\delta_t^- U_i^n = \mathcal{K} \left(\frac{\tau}{\Delta t} \right)^{\alpha} \sum_{k=0}^{n-1} b_{n-k-1} \left(\delta_x^2 U_i^{k+1} - \delta_x^2 U_i^k \right) + f_i^n, \qquad (105a)
$$

$$
U_i^0 = \phi(x_i) \quad (i = 0, 1, \ldots, M), \qquad (105b)
$$

$$
U_0^n = l(t_n) \quad (n = 0, 1, \ldots, N), \qquad (105c)
$$

$$
U_M^n = r(t_n) \quad (n = 0, 1, \ldots, N), \qquad (105d)
$$

onde $b_k = \frac{1}{\Gamma(2-\alpha)}\left[(k+1)^{1-\alpha} - k^{1-\alpha}\right]$. A partir de (105a), escrevemos

$$U_i^n = U_i^{n-1} + r\sum_{k=0}^{n-1} b_{n-k-1}\left[U_{i-1}^{k+1} - U_{i-1}^{k} - 2(U_i^{k+1} - U_i^k) + U_{i+1}^{k+1} - U_{i+1}^{k}\right] + \Delta t f_i^n \quad (106)$$

com $r := \mathcal{K}\frac{\Delta t}{\Delta x^2}\left(\frac{\tau}{\Delta t}\right)^{\alpha}$. Para $1 \leq i \leq M-1$, a forma matricial para (106) é

$$\boldsymbol{U^n} = \boldsymbol{U^{n-1}} + \sum_{k=0}^{n-1} b_{n-k-1}\left(A(\boldsymbol{U^{k+1}} - \boldsymbol{U^k}) + \boldsymbol{C^{k+1}} - \boldsymbol{C^k}\right) + \Delta t \boldsymbol{f^n}, \quad (107)$$

ou

$$\begin{aligned}(\boldsymbol{I} - b_0 A)\,\boldsymbol{U^n} \;&=\; \sum_{k=0}^{n-2} b_{n-k-1}\left(A(\boldsymbol{U^{k+1}} - \boldsymbol{U^k}) + \boldsymbol{C^{k+1}} - \boldsymbol{C^k}\right)\\ &+\; (\boldsymbol{I} - b_0 A)\,\boldsymbol{U^{n-1}} + b_0\left(\boldsymbol{C^{k+1}} - \boldsymbol{C^k}\right) + \Delta t \boldsymbol{f^n}, \quad (108)\end{aligned}$$

onde os vetores $\boldsymbol{U^n}$, $\boldsymbol{C^n}$, $\boldsymbol{f^n}$ e a matriz A são iguais aos definidos na Seção 3.2.

A dependência para um ponto calculado pelo método tipo Euler regressivo com a derivada de Caputo é igual àquela obtida com a derivada de Riemann-Liouville, conforme ilustrado nas Figuras 5a, 5b e 5c.

3.5.1 Experimentos numéricos do *MtER* com Caputo

Nesta seção, realizamos experimentos que apresentam a solução numérica ao aplicarmos o método do tipo Euler regressivo no problema de difusão fracionária com a derivada de ordem arbitrária proposta por Caputo. No Experimento 3.4, verificamos a ordem de convergência do método utilizado em relação à derivada temporal. No Experimento 3.5, são exibidos nos gráficos das Figuras 7a e 7b os efeitos para diferentes valores dos parâmetros α e τ.

Experimento 3.4. *Aqui, vamos investigar a ordem de convergência do método tipo Euler regressivo relacionada à variável temporal, quando aplicado à equação da difusão fracionária com a derivada fracionária proposta por Caputo. A função* $u(x,t) = t^{1,5}\exp(\pi(x-\alpha))$ *é a solução numérica encontrada para o problema* (102)*, relativa ao domínio* $(x,t) \in [0,1]\times[0,1]$*, os dados usados estão descritos a seguir:*

- $f(x,t) = \left(1,5t^{0,5} - \pi^2\frac{\Gamma(2,5)}{\Gamma(2,5-\alpha)}t^{1,5-\alpha}\right)\exp(\pi(x-\alpha))$;
- $\phi(x) = 0$; $\quad l(t) = t^{1,5}\exp(-\pi\alpha)$; $\quad r(t) = t^{1,5}\exp(\pi(1-\alpha))$.

A Tabela 3 exibe os resultados encontrados para $\Delta x = 1/1000$ *e* $\tau = 1$*, variando tanto o número de passos N na variável temporal, como a ordem da derivada fracionária*

α.

Tabela 4 - Ordem de convergência do *MtER* aplicado ao Problema 102 – Caputo.

N	$\alpha = 0,2$	ordem	$\alpha = 0,5$	ordem	$\alpha = 0,8$	ordem
8	$1,7288 \times 10^{-2}$		$4,4971 \times 10^{-3}$		$1,3126 \times 10^{-2}$	
16	$1,0948 \times 10^{-2}$	0,6591	$4,8385 \times 10^{-4}$	3,2164	$5,7759 \times 10^{-2}$	1,1843
32	$6,4030 \times 10^{-3}$	0,7165	$4,4200 \times 10^{-4}$	1,6734	$2,4807 \times 10^{-3}$	1,2018
64	$3,5467 \times 10^{-3}$	0,7617	$4,7770 \times 10^{-4}$	1,0783	$1,0439 \times 10^{-3}$	1,2175
128	$1,9000 \times 10^{-3}$	0,7964	$3,3349 \times 10^{-4}$	0,9383	$4,3072 \times 10^{-4}$	1,2324

Legenda: Os erros na norma L^2 e os valores correspondentes da ordem de convergência são exibidos para os valores $0,2$, $0,5$ e $0,8$ da ordem α da derivada fracionária segundo Caputo. Os números de passos N em relação à variável t estão na primeira coluna, enquanto os parâmetros $\mathcal{K} = 1$, $\Delta x = 1/1000$ e $\tau = 1$ permanecem fixos.

Fonte: O autor, 2023.

Os resultados exibidos na Tabela 6 nos motivam a explorar o método para outros exemplos e com outras variações dos parâmetros de entrada. Visto que, para $\alpha = 0,2$ a ordem ficou abaixo do valor, enquanto para $\alpha = 0,5$ as aproximações são excelentes, contudo sem que as ordens tenham uma coerência com o refinamento da malha.

Experimento 3.5. *Nesta segunda aplicação, o método tipo Euler progressivo é empregado no problema* (102) *ao domínio* $(x,t) \in [0,1] \times [0,2]$. *Os parâmetros a seguir foram fixados em* $\mathcal{K} = 1$, $\Delta x = 1/100$ *e* $\Delta t = 2/400$, *considerando os seguintes dados de entrada:*

- $f(x,t) = (1+\pi^2 t)(1+t)\exp(t)\,\mathrm{sen}(\pi x)$;
- $\phi(x) = 0$; $\quad l(t) = 0$; $\quad r(t) = t\exp(t)$.

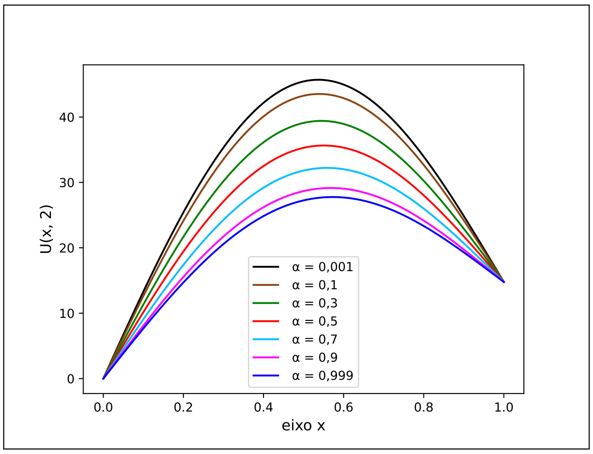
3.5.2 Algoritmo do *MtER* com Caputo

A seguir, apresentamos o algoritmo do método de Euler regressivo aplicado à equação da difusão fracionária com a derivada não inteira segundo Caputo. Este algoritmo foi desenvolvido com base na equação (108), e o código correspondente em Python está disponível no Apêndice B.3.

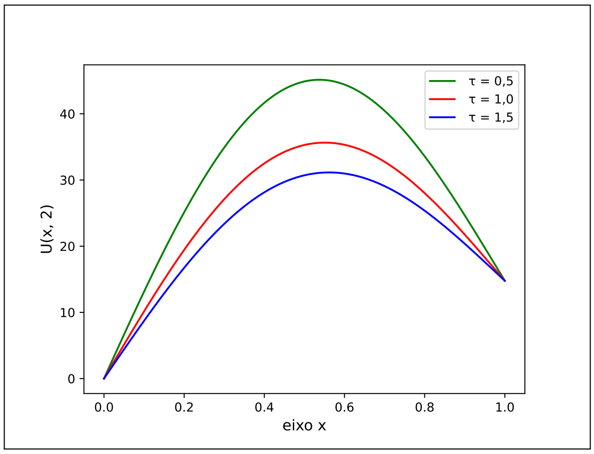
Método tipo Euler Regressivo – Caputo

Propósito:

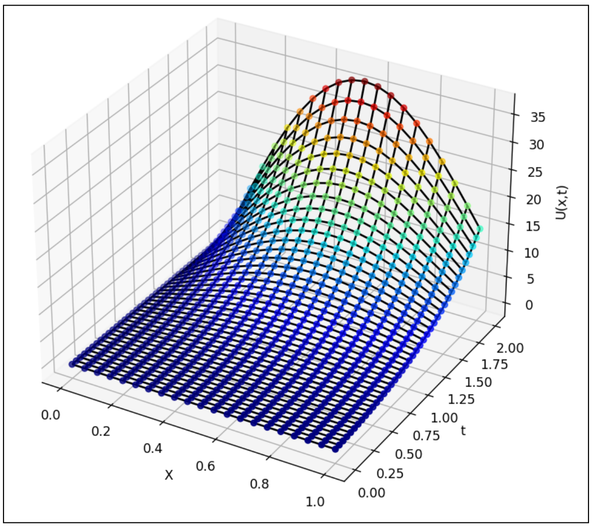
Figura 7 - Solução numérica da equação de difusão fracionária com a derivada de Caputo.



(a)



(b)



(c)

Legenda: As Figuras (a) e (b) exibem cortes das soluções em $t = 2$. Na primeira figura, variamos o parâmetro α enquanto mantemos $\tau = 1$. Na segunda figura, fixamos $\alpha = 0,5$ e variamos o valor de τ. Já na Figura (c), é apresentado o gráfico de superfície para $\alpha = 0,5$ e $\tau = 1$.

Fonte: O autor, 2023.

Solução numérica da equação da difusão fracionária com Caputo,

$$\frac{\partial u(x,t)}{\partial t} = {}_C D_{0,t}^{\alpha}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema implícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

α: ordem da derivada fracionária;

τ: parâmetro de correção dimensional;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$r = \tau^{\alpha}\mathcal{K}(\Delta t)^{1-\alpha}/(\Delta x)^2$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $k = 0, 1, \ldots, N$ faça

$b_k = ((k+1)^{1-\alpha} - k^{1-\alpha})/\Gamma(2-\alpha)$;

Passo 6

Para $i = 0, 1, \ldots, M - 1$ faça

para $j = 0, 1, \ldots, M - 1$ faça

se $i == j$, então $A_i^j = -2r$;

se $i == j - 1$ ou $i == j + 1$, então $A_i^j = r$;

senão $A_i^j = 0$, (matriz tridiagonal);

Passo 7

Crie, I, matriz identidade de ordem M;

Faça $n = 1$;

Enquanto $n \leq N$ faça

crie, V, W e X, vetores de ordem M;

para $k = 0, 1, \ldots, n - 1$ faça

$V = A \times (U^{k+1} - U^k)$;

$V_0 = V_0 + r \times (U_0^{k+1} - U_0^k)$;

$V_{M-1} = V_{M-1} + r \times (U_M^{k+1} - U_M^k)$;

$W = W + b_k \times V$;

$X = (I - b_0 A) \times U^{n-1}$;

$X_0 = X_0 + b_0 r \times (U_0^n - U_0^{n-1})$;

$X_{M-1} = X_{M-1} + b_0 r \times (U_M^n - U_M^{n-1})$;

resolva, $(I - b_0 A)U^n = W + X + \Delta t \times F^n$;

$n = n + 1$;

Passo 8

Para $n = 0, 1, \ldots, N$ faça

imprima U^n;

Passo 9

PARE (O procedimento está completo).

---

4 DIFUSÃO FRACIONÁRIA COM DERIVADAS LOCAIS

Neste capítulo apresentamos as derivadas fracionárias locais, também conhecidas como derivadas fractais (CHEN et al., 2010). As derivadas fracionárias locais podem ser escritas em termos da derivada ordinária de ordem um, portanto, podem ser interpretadas como derivadas fracionárias especiais (HE, 2014) As derivadas fracionárias locais constituem formulações relativamente recentes e, em geral, foram introduzidas em associação a problemas específicos.

4.1 Derivada Fracionária de Chen

Nesta seção, apresentamos a derivada fracionária local introduzida pelo pesquisador Wen Chen (CHEN, 2006). Essa formulação tem sido utilizada para modelar diversos fenômenos, dentre os quais podemos citar a turbulência (CHEN, 2005) e a difusão anômala (CHEN, 2006). A derivada não inteira, conforme definida por Chen, é expressa por meio de um limite semelhante à derivada ordinária.

Definição 13. *Sejam $t \in \mathbb{R}$ e $f(t)$ uma função. A derivada de Chen de ordem $\alpha > 0$ de f é definida a partir do limite na equação* (109)

$$ {}_{CH}D_t^{\alpha} = \lim_{s \to t} \frac{f(t) - f(s)}{t^{\alpha} - s^{\alpha}} \tag{109}$$

Aplicando a regra de l'Hôpital em (109), podemos escrever a derivada de Chen de forma alternativa como

$$ {}_{CH}D_t^{\alpha} = \frac{f'(t)}{\alpha t^{\alpha-1}} \tag{110}$$

isto é,

$$ {}_{CH}D_t^{\alpha} = \lim_{s \to t} \frac{f(t) - f(s)}{t^{\alpha} - s^{\alpha}} = \lim_{s \to t} \frac{-f'(s)}{-\alpha s^{\alpha-1}} = \frac{f'(t)}{\alpha t^{\alpha-1}}$$

onde a f' denota derivada de ordem inteira em relação à variável t.

A derivada fracionária de Chen, conforme referenciado por (109), atende apenas ao critério de linearidade proposto por Ortigueira e Machado (ORTIGUEIRA; MACHADO, 2015). Os outros quatro critérios - derivada de ordem zero, lei dos expoentes, derivada de ordem inteira e regra de Leibniz - não são satisfeitos (TEODORO, 2019).

A equação de difusão fracionária em conjunto com as condições iniciais e de contorno, que aproxima a difusão anômala com a derivada fracionária local de Chen, é

$$\frac{\partial u(x,t)}{\partial t} = {}_{CH}D_t^{\alpha}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \;\; 0 < t \leq T, \tag{111a}$$

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L, \tag{111b}$$

$$u(0,t) = l(t), \quad 0 \leq t \leq T, \tag{111c}$$

$$u(L,t) = r(t), \quad 0 \leq t \leq T, \tag{111d}$$

onde $\phi(x)$, $l(t)$, $r(t)$ e $f(x,t)$ são funções suficientemente regulares dadas e $0 < \alpha < 1$ é a ordem da derivada fracionária.

Para aproximar a derivada de Chen a partir da equação (109), aplicamos os esquemas de diferenças finitas, a saber:

$${}_{CH}D_t^{\alpha} f(t)|_{t=t_n} = \frac{1}{\alpha(n\Delta t)^{\alpha-1}}\frac{f(t_{n+1}) - f(t_n)}{\Delta t} + O(\Delta t) \tag{112}$$

$${}_{CH}D_t^{\alpha} f(t)|_{t=t_n} = \frac{1}{\alpha(n\Delta t)^{\alpha-1}}\frac{f(t_n) - f(t_{n-1})}{\Delta t} + O(\Delta t) \tag{113}$$

onde (112) é um esquema avançado e (113) um esquema recuado.

4.1.1 Método tipo Euler regressivo com a derivada de Chen

Nesta seção, vamos construir o método do tipo Euler regressivo para a equação da difusão fracionária, utilizando a formulação proposta por Chen. Aplicando as aproximações (35) e (38) para as derivadas inteiras, e a aproximação (113) para a derivada fracionária, no problema (111), para $i = 1, 2, \ldots, M-1$ e $n = 1, 2, \ldots, N$, chegamos a

$$\frac{U_i^n - U_i^{n-1}}{\Delta t} = \frac{\mathcal{K}\tau^{\alpha}}{\alpha(n\Delta t)^{\alpha-1}}\left(\frac{\delta_x^2 U_i^n - \delta_x^2 U_i^{n-1}}{\Delta t}\right) + f_i^n, \tag{114a}$$

$$U_i^0 = \phi(x_i) \;\; (i = 0, 1, \ldots, M), \tag{114b}$$

$$U_0^n = l(t_n) \;\; (n = 0, 1, \ldots, N), \tag{114c}$$

$$U_M^n = r(t_n) \;\; (n = 0, 1, \ldots, N), \tag{114d}$$

onde $\delta_x^2 U_i^n$ representa as diferenças centradas no ponto (x_i, t_n).

Denotando $\partial_x^2 U_i^n := \delta_x^2 U_i^n \Delta x^2$, em (114a) e após realizar cancelamentos e manipulações convenientes, temos

$$U_i^n = U_i^{n-1} + \mathcal{P}(n) r \left(\partial_x^2 U_i^n - \partial_x^2 U_i^{n-1}\right) + \Delta t f_i^n, \tag{115}$$

onde $\mathcal{P}(n) := n^{1-\alpha}/\alpha$ e $r := \mathcal{K}\frac{\Delta t}{\Delta x^2}\left(\frac{\tau}{\Delta t}\right)^{\alpha}$. Para $1 \leq i \leq M-1$, a forma matricial para a equação (115) é

$$\boldsymbol{U}^{\boldsymbol{n}} = \boldsymbol{U}^{\boldsymbol{n-1}} + \mathcal{P}(n)\left[A(\boldsymbol{U}^{\boldsymbol{n}} - \boldsymbol{U}^{\boldsymbol{n-1}}) + \boldsymbol{C}^{\boldsymbol{n}} - \boldsymbol{C}^{\boldsymbol{n-1}}\right] + \Delta t \boldsymbol{f}^{\boldsymbol{n}} \tag{116}$$

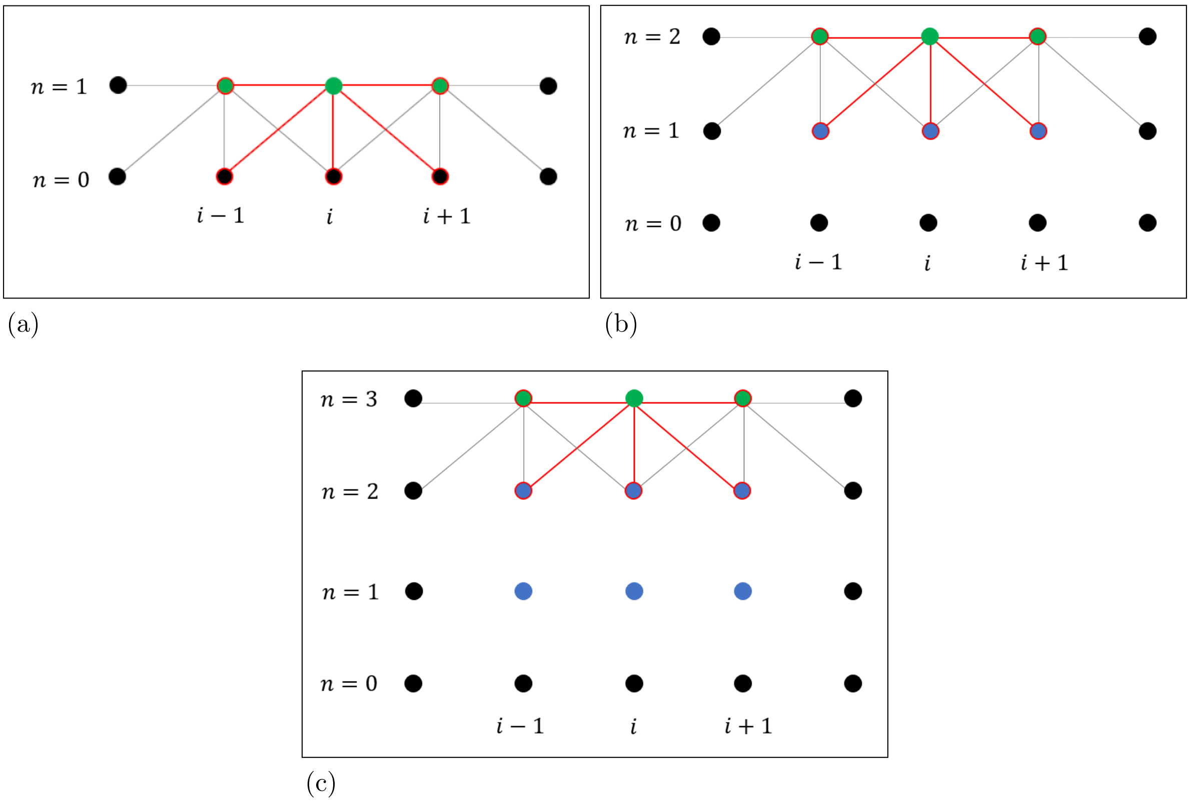
ou

$$\left(\boldsymbol{I} - \mathcal{P}(n)A\right)\boldsymbol{U}^{\boldsymbol{n}} = \left(\boldsymbol{I} - \mathcal{P}(n)A\right)\boldsymbol{U}^{\boldsymbol{n-1}} + \mathcal{P}(n)\left(\boldsymbol{C}^{\boldsymbol{n}} - \boldsymbol{C}^{\boldsymbol{n-1}}\right) + \Delta t \boldsymbol{f}^{\boldsymbol{n}}. \tag{117}$$

A matriz A e os vetores $\boldsymbol{U}^{\boldsymbol{n}}$, $\boldsymbol{C}^{\boldsymbol{n}}$ e $\boldsymbol{f}^{\boldsymbol{n}}$ são definidos como na Seção 3.2.

As Figuras 8a, 8b e 8c exibem a dependência dos valores calculados com relação aos nós nos três primeiros níveis para o método tipo Euler regressivo, com a derivada fracionária de Chen. O estêncil formado pelas arestas vermelhas mostra os valores que precisam ser conhecidos para o cálculo do nó da posição i e no nível n. Esta visualização comprova que o operador fracionário de Chen é local, como a derivada de ordem inteira (LEVEQUE, 2007).

Figura 8 - Estêncil do método tipo Euler regressivo com a derivada de Chen.



Legenda: As figuras exibem os estênceis em vermelho na posição i e nos níveis: (a) $n = 1$, (b) $n = 2$ e (c) $n = 3$, para o problema (15), com a derivada segundo Chen.

Fonte: O autor, 2023.

4.1.2 Experimentos numéricos do $MtER$ com Chen

Nesta seção, realizamos experimentos que apresentam a solução numérica ao aplicarmos o método de Euler regressivo no problema de difusão fracionária com a derivada fracionária local de Chen. A ordem de convergência em relação à variável temporal é investigada no primeiro exemplo. No segundo exemplo, são exibidos nos gráficos os efeitos na mudança dos parâmetros α e τ.

Experimento 4.1. *Neste exemplo, será calculada a ordem de convergência do método do tipo Euler regressivo para a equação de difusão fracionária com a formulação proposta por Chen. A função* $u(x,t) = t^{1,5+\alpha}\,\mathrm{sen}\,(\pi x)$ *é a solução explícita encontrada para o problema* (111)*, referente ao domínio* $(x,t) \in [0,1] \times [0,1]$*, e os dados utilizados estão descritos a seguir:*

- $f(x,t) = (1+\alpha)t^{0,5+\alpha}\,(1+\pi^2 t^{\alpha-1}/\alpha)\,\mathrm{sen}\,(\pi x)$;
- $\phi(x) = 0; \quad l(t) = 0; \quad r(t) = 0.$

A Tabela 5 exibe os resultados encontrados, para $\mathcal{K} = 1$, $\Delta x = 1/1000$ *e* $\tau = 1$, *variando o número de passos* N *na variável temporal, como também a ordem da derivada fracionária* α.

Tabela 5 - Ordem de convergência do $MtER$ aplicado ao Problema 111 – Chen.

N	$\alpha = 0,2$	ordem	$\alpha = 0,5$	ordem	$\alpha = 0,8$	ordem
8	$7,1101 \times 10^{-2}$		$8,8389 \times 10^{-2}$		$1,0381 \times 10^{-1}$	
16	$3,6262 \times 10^{-2}$	0,9714	$4,4195 \times 10^{-2}$	1,0000	$5,1392 \times 10^{-2}$	1,0143
32	$1,8366 \times 10^{-2}$	0,9764	$2,2098 \times 10^{-2}$	1,0000	$2,5560 \times 10^{-2}$	1,0110
64	$9,2595 \times 10^{-3}$	0,9803	$1,1049 \times 10^{-2}$	1,0000	$1,2744 \times 10^{-2}$	1,0087
128	$4,6545 \times 10^{-3}$	0,9833	$5,5248 \times 10^{-3}$	1,0000	$6,3632 \times 10^{-3}$	1,0070

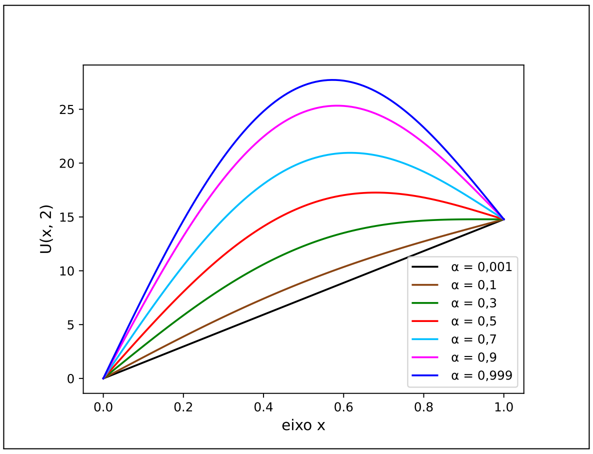
Legenda: Os erros na norma L^2 e os valores correspondentes da ordem de convergência são exibidos para os valores $0,2$, $0,5$ e $0,8$ da ordem α da derivada fracionária segundo Chen. Os números de passos N em relação à variável t estão na primeira coluna, enquanto os parâmetros $\mathcal{K} = 1$, $\Delta x = 1/1000$ e $\tau = 1$ permanecem fixos.

Fonte: O autor, 2023.

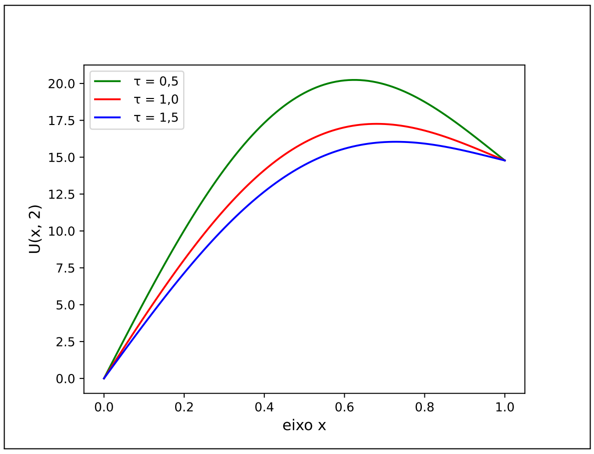
Experimento 4.2. *Neste segundo teste aplicamos o método tipo Euler progressivo ao problema* (111) *no domínio* $(x,t) \in [0,1] \times [0,2]$*, considerando os seguintes dados de entrada:*

- $f(x,t) = (1+\pi^2 t)(1+t)\exp{(t)}\,\mathrm{sen}\,(\pi x)$;
- $\phi(x) = 0; \quad l(t) = 0; \quad r(t) = t\exp{(t)}.$

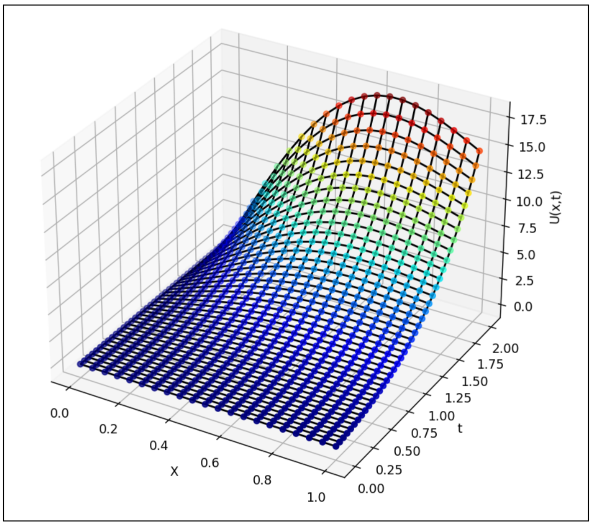
Figura 9 - Solução numérica da equação de difusão fracionária com a derivada de Chen.



(a)



(b)



(c)

Legenda: As Figuras (a) e (b) exibem cortes das soluções em $t = 2$. Na primeira figura, variamos o parâmetro α enquanto mantemos $\tau = 1$. Na segunda figura, fixamos $\alpha = 0,5$ e variamos o valor de τ. Já na Figura (c), é apresentado o gráfico de superfície para $\alpha = 0,5$ e $\tau = 1$.

Fonte: O autor, 2023.

4.1.3 Algoritmo do $MtER$ com Chen

Nesta seção, fornecemos o algoritmo do método de Euler regressivo aplicado à equação de difusão fracionária com derivada fracionária local de Chen. Esse algoritmo foi construído a partir da equação (117), o código escrito na linguagem Python está disponível no Apêndice B.4

Método tipo Euler Regressivo – Chen

Propósito:

Solução numérica da equação da difusão fracionária com Chen

$$\frac{\partial u(x,t)}{\partial t} = {}_{CH}D^{\alpha}_{0,t}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema implícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

α: ordem da derivada fracionária;

τ: parâmetro de correção dimensional;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$r = \tau^{\alpha}\mathcal{K}(\Delta t)^{1-\alpha}/(\Delta x)^2$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $n = 0, 1, \ldots, N$ faça

$P_n = n^{1-\alpha}/\alpha$;

Passo 6

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $A_i^j = -2r$;

se $i == j-1$ ou $i == j+1$, então $A_i^j = r$;

senão $A_i^j = 0$, (matriz tridiagonal);

Passo 7

Crie, I, matriz identidade de ordem M;

Faça $n = 1$;

Enquanto $n \leq N$ faça

crie V, vetor de ordem M;

$V = (I - P_n \times A)U^{n-1}$;

$V_0 = V_0 + r \times P_n(U_0^n - U_0^{n-1})$;

$V_{M-1} = V_{M-1} + r \times P_n(U_M^n - U_M^{n-1})$;

resolva, $(I - P_n \times A)U^n = V + \Delta t \times F^n$;

$n = n + 1$;

Passo 8

Para $n = 0, 1, \ldots, N$ faça

imprima U^n;

Passo 9

PARE (O procedimento está completo).

4.2 Derivada fracionária de Katugampola

Nesta seção, apresentamos a derivada fracionária definida por Katugampola (KATUGAMPOLA, 2014). Essa formulação cumpre as propriedades clássicas, incluindo: linearidade, regra do produto, regra do quociente, regra da potência, regra da cadeia, nula a derivada da função constante, validade dos teorema de Rolle e do valor médio.

Definição 14. *Sejam $t \in \mathbb{R}$ e $f(t)$ uma função real. A derivada fracionária de Katugampola de ordem α, sendo $0 < \alpha < 1$, da função f é dada por*

$$ {}_K D^{\alpha} f(t) := \lim_{\epsilon \to 0} \frac{f(te^{\epsilon t^{-\alpha}}) - f(t)}{\epsilon}, \tag{118} $$

para todo $t > 0$.

A Definição 14 é uma generalização da derivada clássica que usa a abordagem de limite. Para $\alpha = 1$, a derivada de Katugampola é equivalente à definição clássica de derivada de primeira ordem da função f.

Teorema 9. *Considere $\alpha \in (0, 1]$ e f uma função diferenciável em um ponto $t > 0$, então*

$$ {}_K D^{\alpha} f(t) = t^{1-\alpha} f'(t). \tag{119} $$

Demonstração. Por hipótese, $f(\cdot)$ é diferenciável. Assim, por Taylor

$$ te^{\epsilon t^{-\alpha}} = t + \epsilon t^{1-\alpha} + \mathcal{O}(\epsilon^2). $$

Aplicando essa relação em (118), obtém-se

$$ {}_K D^{\alpha} f(t) = \lim_{\epsilon \to 0} \frac{f(t + \epsilon t^{1-\alpha} + \mathcal{O}(\epsilon^2)) - f(t)}{\epsilon}. $$

Chamando $h = \epsilon t^{1-\alpha} (1 + \mathcal{O}(\epsilon))$, quando $\epsilon \to 0 \;\Rightarrow\; h \to 0$, temos

$$ {}_K D^{\alpha} f(t) = \lim_{h \to 0} t^{1-\alpha} (1 + \mathcal{O}(\epsilon)) \frac{f(t+h) - f(t)}{h}, $$

ou

$$ {}_K D^{\alpha} f(t) = t^{1-\alpha} f'(t). $$

□

Katugampola (KATUGAMPOLA, 2014) também definiu a derivada fracionária para um valor arbitrário de α, quando $\alpha \in (n, n+1]$, sendo $n \in \mathbb{N}$.

Definição 15. *Sejam $t \in \mathbb{R}$, $n \in \mathbb{N}$ e $f(t)$ uma função real n vezes diferenciável. A derivada fracionária de Katugampola de ordem α, sendo $n < \alpha \leq n + 1$, da função f é dada por*

$$ {}_K D^{\alpha} f(t) := \lim_{\epsilon \to 0} \frac{f^{(n)}(te^{\epsilon t^{n-\alpha}}) - f^{(n)}(t)}{\epsilon}, \tag{120} $$

para todo $t > 0$.

Seguindo os mesmos passos do Teorema 9 em (120) temos

$$ {}_K D^{\alpha} f(t) = t^{1-\alpha+n} f^{(n+1)}(t), \tag{121} $$

sendo $n \in \mathbb{N}$ e f $n+1$ vezes diferenciável.

A formulação proposta por Katugampola é classificada por Teodoro (2019) em sua tese de doutorado como um operador local e nesse trabalho demostra-se que tal operador é linear, a derivada fracionária para α inteiro produz o mesmo resultado da derivada ordinária correspondente, e a regra de Leibniz é satisfeita. Todavia, a derivada de ordem zero de uma função não é a própria função e a lei dos expoentes não é satisfeita. Com isso, segundo o critério estabelecido por Ortigueira e Machado a derivada segundo Katugampola não pode ser classificada como fracionária.

4.2.1 Método tipo Euler regressivo com a derivada de Katugampola

Aqui propomos o método tipo Euler regressivo para a equação da difusão fracionária com o operador fracionário de Katugampola. As derivadas de ordem inteira no problema (15) são aproximadas pelas diferenças regressivas (35) e centradas (38). Aplicando a derivada fracionária de Katugampola em (15a), temos que o problema (15) é escrito da seguinte maneira, com $i = 1, 2, \ldots, M-1$ e $n = 1, 2, \ldots, N$:

$$ \frac{U_i^{n+1} - U_i^n}{\Delta t} = \mathcal{K}\tau^{\alpha}\, {}_K D^{\alpha} \left(\frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2} \right) + f_i^n, \tag{122a} $$

$$ U_i^0 = \phi(x_i) \quad (i = 0, 1, \ldots, M), \tag{122b} $$

$$ U_0^n = l(t_n) \quad (n = 0, 1, \ldots, N), \tag{122c} $$

$$ U_M^n = r(t_n) \quad (n = 0, 1, \ldots, N). \tag{122d} $$

As características do operador local de Katugampola permitem elaborar um esquema numérico para a derivada fracionária utilizando somente as diferenças regressivas, visto que na equação (119), obtivemos a relação formal entre a derivada fracionária de Katugampola com a derivada ordinária de ordem um, a saber,

$$ {}_K D_t^{\alpha} f(t) = t^{1-\alpha} f'(t) $$

para $0 < \alpha < 1$.

Aplicando diferenças regressivas nas diferenças centradas em (122a) e multiplicando o resultado por $t_n^{1-\alpha}$, se obtém o esquema numérico a seguir,

$$ \frac{U_i^n - U_i^{n-1}}{\Delta t} = \mathcal{K}\tau^{\alpha} t_n^{1-\alpha} \left(\frac{\delta_x^2 U_i^n - \delta_x^2 U_i^{n-1}}{\Delta t} \right) + f_i^n, \tag{123} $$

onde $\delta_x^2 U_i^n$ representa as diferenças centradas.

Denotando $\partial_x^2 U_i^n/\Delta x^2 := \delta_x^2 U_i^n$ e como $t_n = n\Delta t$, após realizar cancelamentos e manipulações convenientes, temos

$$U_i^n = U_i^{n-1} + r\mathcal{T}(n)\left(\partial_x^2 U_i^n - \partial_x^2 U_i^{n-1}\right) + \Delta t f_i^n, \tag{124}$$

onde $\mathcal{T}(n) := n^{1-\alpha}$. Para $1 \leq i \leq M-1$, a forma matricial para (124) é

$$\boldsymbol{U^n} = \boldsymbol{U^{n-1}} + \mathcal{T}(n)\left[A(\boldsymbol{U^n} - \boldsymbol{U^{n-1}}) + \boldsymbol{C^n} - \boldsymbol{C^{n-1}}\right] + \Delta t\boldsymbol{f^n} \tag{125}$$

ou

$$(\boldsymbol{I} - \mathcal{T}(n)A)\,\boldsymbol{U^n} = (\boldsymbol{I} - \mathcal{T}(n)A)\,\boldsymbol{U^{n-1}} + \mathcal{T}(n)\left(\boldsymbol{C^n} - \boldsymbol{C^{n-1}}\right) + \Delta t\boldsymbol{f^n}. \tag{126}$$

A matriz A e os vetores $\boldsymbol{U^n}$, $\boldsymbol{C^n}$ e $\boldsymbol{f^n}$ são definidos como na Seção 3.2.

4.2.2 Experimento numérico do $MtER$ com Katugampola

Nesta seção, iremos apresentar resultados numéricos do problema da difusão fracionária utilizando a derivada fracionária de Katugampola por meio do método do tipo Euler regressivo. Apresentaremos dois exemplos: o primeiro tem o objetivo de verificar a precisão e a ordem de convergência do esquema numérico. No segundo exemplo, exploraremos o efeito da ordem da derivada fracionária e do parâmetro de correção dimensional.

Experimento 4.3. *A função* $u(x,t) = t^{1,5}\exp(\pi(x-\alpha))$ *é a solução encontrada para o problema* (53)*, com os seguintes dados:*

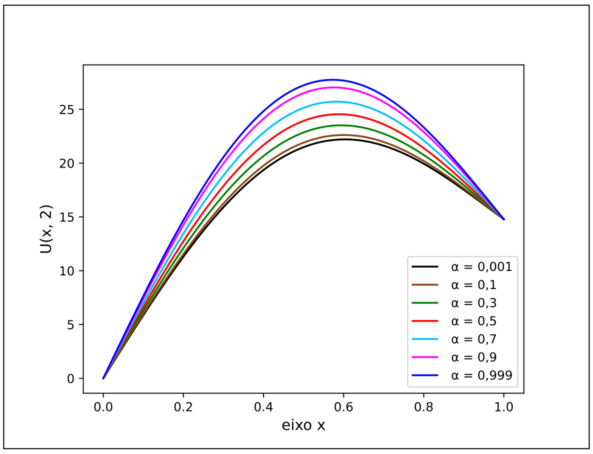
- $f(x,t) = 1,5t^{0,5}(1 - t^{1-\alpha}\pi^2)\exp\left(\pi(x-\alpha)\right)$;
- $\phi(x) = 0$; $l(t) = t^{1,5}\exp(-\pi\alpha)$; $r(t) = t^{1,5}\exp(\pi(1-\alpha))$.

A Tabela 6 exibe os resultados numéricos encontrados, relativos ao domínio $(x,t) \in [0,1] \times [0,1]$*, quando fixados* $\mathcal{K} = 1$*,* $\Delta x = 1/1000$ *e* $\tau = 1$*. As soluções numéricas calculadas foram precisas e as ordens ficaram em conformidade com as aproximações do esquema empregado.*

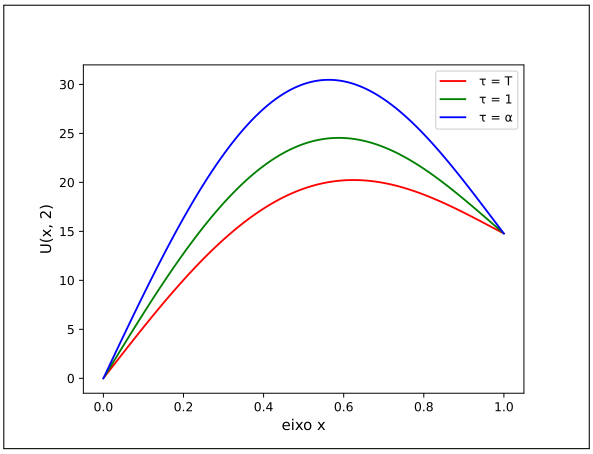
Experimento 4.4. *Em um segundo exemplo será exibida a solução numérica do problema* (122) *no domínio* $(x,t) \in [0,1] \times [0,2]$*. Os parâmetros a seguir foram fixados em* $\mathcal{K} = 1$*,* $\Delta x = 1/100$ *e* $\Delta t = 2/400$*, considerando os seguintes dados de entrada:*

- $f(x,t) = (1+\pi^2 t)(1+t)\exp(t)\,\mathrm{sen}(\pi x)$;
- $\phi(x) = 0$; $l(t) = 0$; $r(t) = t\exp(t)$.

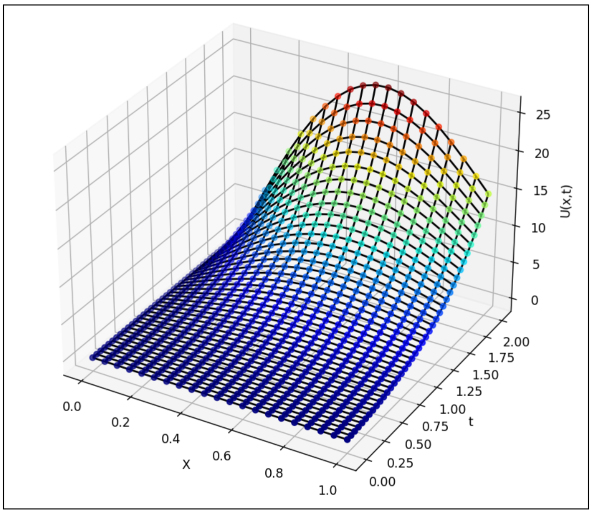
Figura 10 - Solução numérica da equação de difusão fracionária com a derivada de Katugampola.



(a)



(b)



(c)

Legenda: As Figuras (a) e (b) exibem cortes das soluções em $t = 2$. Na primeira figura, variamos o parâmetro α enquanto mantemos $\tau = 1$. Na segunda figura, fixamos $\alpha = 0,5$ e variamos o valor de τ. Já na Figura (c), é apresentado o gráfico de superfície para $\alpha = 0,5$ e $\tau = 1$.

Fonte: O autor, 2023.

Tabela 6 - Método tipo Euler regressivo com derivada fracionária de Katugampola.

N	$\alpha = 0,2$	ordem	$\alpha = 0,5$	ordem	$\alpha = 0,8$	ordem
8	$1,4044 \times 10^{-1}$		$6,4947 \times 10^{-2}$		2.8266×10^{-2}	
16	$6,4226 \times 10^{-2}$	1,1287	$3,2169 \times 10^{-2}$	1,0136	$1,4569 \times 10^{-2}$	0,9562
32	$2,9219 \times 10^{-2}$	1,1325	$1,5796 \times 10^{-2}$	1,0199	$7,4306 \times 10^{-3}$	0,9638
64	$1,3411 \times 10^{-2}$	1,1295	$7,7282 \times 10^{-3}$	1,0237	$3,7626 \times 10^{-3}$	0,9698
128	$6,2585 \times 10^{-3}$	1,1220	$3,7810 \times 10^{-3}$	1,0256	$1,8960 \times 10^{-3}$	0,9745

Legenda: Erro na norma L^2 com $\mathcal{K} = 1$, $\Delta x = 1/1000$, $\tau = 1$, N é a quantidade de passos na variável t e α a ordem da derivada fracionária segundo Katugampola.

Fonte: O autor, 2023.

4.2.3 Algoritmo do $MtER$ com Katugampola

Nesta seção, fornecemos o algoritmo do método de Euler regressivo aplicado à equação de difusão fracionária com derivada fracionária local de Katugampola. Esse algoritmo foi construído a partir da equação (126), o código escrito na linguagem Python está disponível no Apêndice B.5

Método tipo Euler Regressivo – Katugampola

Propósito:

Solução numérica da equação da difusão fracionária com Katugampola

$$\frac{\partial u(x,t)}{\partial t} = {}_K D^{\alpha}_{0,t}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema implícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

α: ordem da derivada fracionária;

τ: parâmetro de correção dimensional;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$r = \tau^\alpha \mathcal{K}(\Delta t)^{1-\alpha}/(\Delta x)^2$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $n = 0, 1, \ldots, N$ faça

$T_n = n^{1-\alpha}$;

Passo 6

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $A_i^j = -2r$;

se $i == j-1$ ou $i == j+1$, então $A_i^j = r$;

senão $A_i^j = 0$, (matriz tridiagonal);

Passo 7

Crie, I, matriz identidade de ordem M;

Faça $n = 1$;

Enquanto $n \leq N$ faça

crie V, vetor de ordem M;

$V = (I - T_n \times A)U^{n-k}$;

$V_0 = V_0 + r \times T_n(U_0^n - U_0^{n-1})$;

$V_{M-1} = V_{M-1} + r \times T_n(U_M^n - U_M^{n-1})$;

resolva, $(I - T_n \times A)U^n = V + \Delta t \times F^n$;

$n = n + 1$;

Passo 8

Para $n = 0, 1, \ldots, N$ faça

imprima U^n;

Passo 9

PARE (O procedimento está completo).

---

5 DIFUSÃO FRACIONÁRIA COM DERIVADAS COM NÚCLEO NÃO SINGULAR

Neste capítulo, apresentamos duas derivadas fracionárias com núcleo não singular que emergiram na literatura a partir da derivada proposta por Caputo e Fabrizio (CAPUTO; FABRIZIO, 2015). Após o trabalho de Caputo-Fabrizio, surgiram outras formulações com núcleo não singular. Atangana e Baleanu 2016 introduziram duas novas formulações para a derivada, baseadas na derivada de Caputo-Fabrizio, com o núcleo definido por uma função de Mittag-Leffler. Aqui, de maneira semelhante ao que fizemos nos capítulos anteriores, estudaremos a equação da difusão utilizando essas duas formulações de derivadas fracionárias.

5.1 Difusão fracionária com Caputo-Fabrizio

No ano de 2015, Michele Caputo e Mauro Fabrizio apresentaram uma nova definição para derivada fracionária (CAPUTO; FABRIZIO, 2015). Esse operador considera o núcleo da integral fracionária uma função não singular[18] e é mencionado no trabalho que a formulação proposta possui propriedades motivadoras adicionais àquelas de versões anteriores. Essa derivada de ordem arbitrária é denominada derivada fracionária segundo Caputo-Fabrizio.

A formulação proposta por Caputo-Fabrizio foi de alguma forma a inspiração para várias outras formulações que emergiram na literatura com núcleo não singular. Por exemplo, Yang, Srivastava e Tenreiro Machado (YANG; SRIVASTAVA; MACHADO, 2015) apresentaram uma nova formulação cujo núcleo é uma função exponencial; Atangana e Baleanu (ATANGANA; BALEANU, 2016) propuseram duas novas formulações, cujo núcleo é dado por um função de Mittag-Leffler; há registro de outras.

Definição 16. *Para $0 \leq \alpha \leq 1$, $-\infty \leq a < t$ e $f \in H^1(a,b)$ com $b > a$, a derivada fracionária no tempo, de ordem α, segundo Caputo-Fabrizio,*

$$ {}_{CF}D^{\alpha}_{a,t} f(t) := \frac{M(\alpha)}{1-\alpha} \int_a^t \exp\left(-\alpha \frac{t-s}{1-\alpha}\right) f'(s) ds \tag{127}$$

onde $t \geq 0$ e $M(\alpha)$ é uma função de normalização em que $M(0) = M(1) = 1$.

Assim como a derivada fracionária segundo Caputo, a derivada de Caputo-Fabrizio

[18] Uma função é dita singular quando ela não possui inversa e não-singular caso contrário.

de uma constante é zero e o núcleo é não singular em $t = s$. Esse operador tem sido utilizado com êxito em problemas de eletromagnetismo (CAPUTO; FABRIZIO, 2016), difusão (GÓMEZ-AGUILAR et al., 2016), circuito resistor-indutor (ABRO; MEMON; UQAILI, 2018), entre outros.

Veja que, quando consideramos a ordem da derivada α igual a zero, não recuperamos a função original, isto é,

$$ {}_{CF}D^{0}_{a,t}f(t) = M(0)\int_a^t f'(s)\exp(0)ds = f(t) - f(a); $$

dessa forma, a derivada de Caputo-Fabrizio de ordem zero de uma dada função é a própria função desde que $f(a) = 0$.

Assim como as derivadas fracionárias de Chen e de Katugampola, a derivada de Caputo-Fabrizio não cumpre os cinco critérios propostos por Ortigueira e Machado, a saber, a derivada de Caputo-Fabrizio é um operador linear, a derivada de ordem zero de uma dada função é a própria função desde que $f(a) = 0$, a derivada de Caputo-Fabrizio recupera o caso inteiro (ordem zero e um), a lei dos expoentes não é satisfeita e não vale a generalização de regra de Leibniz.

5.1.1 Método tipo Euler regressivo com Caputo-Fabrizio

Nesta seção abordaremos o problema da difusão fracionária com derivada fracionária de Caputo-Fabrizio. As derivadas de ordem inteira no problema (15) são aproximadas pelas diferenças regressivas (35) e centradas (38). Aplicando a derivada fracionária de Caputo-Fabrizio em (15a), temos que o problema (15) é escrito da seguinte maneira, com $i = 1, 2, \ldots, M-1$ e $n = 1, 2, \ldots, N$:

$$ \frac{U_i^n - U_i^{n-1}}{\Delta t} = \mathcal{K}\tau^{\alpha}\, {}_{CF}D^{\alpha}_{0,t}\left(\frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2}\right) + f(x_i, t_n), \tag{128a} $$

$$ U_i^0 = \phi(x_i) \quad (i = 0, 1, \ldots, M), \tag{128b} $$

$$ U_0^n = l(t_n) \quad (n = 0, 1, \ldots, N), \tag{128c} $$

$$ U_M^n = r(t_n) \quad (n = 0, 1, \ldots, N). \tag{128d} $$

A aproximação empregada na derivada de ordem arbitrária é a proposta em (QURESHI; RANGAIG; BALEANU, 2019). Este esquema é montado a partir do uso da clássica fórmula de diferenças finitas de dois pontos para a derivada primeira. Supondo que $f(t) \in H^1(a,b)$ para $t = t_n$ e $0 < \alpha < 1$, temos

$$\begin{aligned}
\left[{}_{CF}D_{0,t}^{\alpha}f(t)\right]_{t=t_n} &= \frac{M(\alpha)}{1-\alpha}\int_0^{t_n} f'(s)\exp\left[-\alpha\frac{(t_n-s)}{1-\alpha}\right]ds \\
&\approx \frac{M(\alpha)}{1-\alpha}\sum_{k=0}^{n-1}\left[\frac{f_{k+1}-f_k}{\Delta t}+\mathcal{O}(h)\right]\int_{t_k}^{t_{k+1}}\exp\left[-\alpha\frac{(t_n-s)}{1-\alpha}\right]ds \\
&= \frac{M(\alpha)}{\Delta t(1-\alpha)}\frac{1-\alpha}{\alpha}\sum_{k=0}^{n-1}\left[f_{k+1}-f_k+\mathcal{O}(h^2)\right]\exp\left[-\alpha\frac{(t_n-s)}{1-\alpha}\right]\Big|_{t_k}^{t_{k+1}}.
\end{aligned}$$

Substituindo os limites de integração e depois de algumas manipulações algébricas obtém-se

$$\left[{}_{CF}D_{0,t}^{\alpha}f(t)\right]_{t=t_n} = \frac{M(\alpha)}{\alpha\Delta t}\left(e^{\frac{\alpha}{1-\alpha}\Delta t}-1\right)\sum_{k=0}^{n-1}\left(f_{n-k}-f_{n-k-1}\right)e^{-\frac{\alpha}{1-\alpha}(k+1)\Delta t}.$$

Pode-se substituir k por $k-1$ para obter a seguinte expressão:

$$\left[{}_{CF}D_{0,t}^{\alpha}f(t)\right]_{t=t_n} = \frac{M(\alpha)}{\alpha\Delta t}\left(e^{\frac{\alpha}{1-\alpha}\Delta t}-1\right)\sum_{k=1}^{n}\left(f_{n-k+1}-f_{n-k}\right)e^{-\frac{\alpha}{1-\alpha}k\Delta t}. \tag{129}$$

A equação (129) é a discretização de primeira ordem do operador de derivada fracionária de Caputo-Fabrizio. Aplicando esse esquema na derivada fracionária em (128a), para $0<\alpha<1$, obtém-se

$$\delta_t^- U_i^n = \mathcal{K}\tau^\alpha\frac{M(\alpha)}{\alpha\Delta t}\left(e^{\frac{\alpha}{1-\alpha}\Delta t}-1\right)\sum_{k=1}^{n}\left(\delta_x^2U_i^{n-k+1}-\delta_x^2U_i^{n-k}\right)e^{-\frac{\alpha}{1-\alpha}k\Delta t}+f_i^n \tag{130}$$

tomando a função de amortização como $M(\alpha)=1$. Para $1\le i\le M-1$, a forma matricial para (130) é

$$\boldsymbol{U^n} = \boldsymbol{U^{n-1}}+\sum_{k=1}^{n}\mathcal{E}(k)\left[B(\boldsymbol{U^{n-k+1}}-\boldsymbol{U^{n-k}})+\boldsymbol{C^{n-k+1}}-\boldsymbol{C^{n-k}}\right]+\Delta t\boldsymbol{f^n} \tag{131}$$

ou

$$\begin{aligned}
(\boldsymbol{I}-\mathcal{E}(1)B)\,\boldsymbol{U^n} &= \sum_{k=2}^{n}\mathcal{E}(k)\left[B(\boldsymbol{U^{n-k+1}}-\boldsymbol{U^{n-k}})+\boldsymbol{C^{n-k+1}}-\boldsymbol{C^{n-k}}\right] \\
&+ (\boldsymbol{I}-\mathcal{E}(1)B)\,\boldsymbol{U^{n-1}}+\mathcal{E}(1)(\boldsymbol{C^n}-\boldsymbol{C^{n-1}})+\Delta t\boldsymbol{f^n},
\end{aligned}\tag{132}$$

onde $\mathcal{E}(k):=\exp\left(-\frac{\alpha}{1-\alpha}k\Delta t\right)$. Os vetores $\boldsymbol{U^n}$, $\boldsymbol{f^n}$ e $\boldsymbol{C^n}$ são definidos exatamente como na Seção 3.2 enquanto a matriz

$$B := \begin{pmatrix} -2s & s & 0 & \dots & 0 \\ s & -2s & s & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & s & -2s & s \\ 0 & \dots & 0 & s & -2s \end{pmatrix},$$

sendo $s := \frac{\mathcal{K}\tau^\alpha}{\alpha\Delta x^2}\left[\exp\left(\frac{\alpha}{1-\alpha}\Delta t\right) - 1\right]$.

A dependência para um ponto calculado pelo método tipo Euler regressivo com a derivada de Caputo-Fabrizio é igual àquela obtida com a derivada de Riemann-Liouville, conforme ilustrado nas Figuras 5a, 5b e 5c.

5.1.2 Experimento numérico do $MtER$ com Caputo-Fabrizio

Nesta seção, vamos mostrar resultados numéricos do problema da difusão fracionária com a derivada fracionária de Caputo-Fabrizio pelo o método tipo Euler regressivo.

Experimento 5.1. *A função* $u(x,t) = t\exp(t)\exp(\pi(x-\alpha))$ *é a solução encontrada para o problema* (53)*, com os seguintes dados:*

- $f(x,t) = \left(\alpha\pi^2\exp\left(\frac{\alpha t}{\alpha-1}\right) + \exp(t)(-\pi^2(\alpha+t)+t+1)\right)\exp(\pi(x-\alpha))$;
- $\phi(x) = 0$; $l(t) = t\exp(t)\exp(-\pi\alpha)$; $r(t) = t\exp(t)\exp(\pi(1-\alpha))$.

A Tabela 7 exibe os resultados numéricos encontrados, relativos ao domínio $(x,t) \in [0,1]\times[0,1]$*, quando fixado* $\mathcal{K} = 1$*,* $\Delta x = 1/1000$ *e* $\tau = 1$*. As soluções numéricas calculadas foram precisas e as ordens ficaram em conformidade com as aproximações do esquema empregado.*

Tabela 7 - Método tipo Euler regressivo com derivada fracionária de Caputo-Fabrizio.

N	$\alpha = 0.2$	ordem	$\alpha = 0,5$	ordem	$\alpha = 0,8$	ordem
10	8.8598×10^{-3}		1.6828×10^{-2}		2.2789×10^{-2}	
20	4.5167×10^{-3}	0.9720	8.4740×10^{-3}	0.9897	1.0023×10^{-2}	1.1850
40	2.2805×10^{-3}	0.9790	4.2545×10^{-3}	0.9919	4.6806×10^{-3}	1.1418
80	1.1459×10^{-3}	0.9836	2.1326×10^{-3}	0.9934	2.2640×10^{-3}	1.1105
160	5.7435×10^{-4}	0.9868	1.0683×10^{-3}	0.9944	1.1187×10^{-3}	1.0871

Legenda: Erro na norma L^2 com $\mathcal{K} = 1$, $\Delta x = 1/1000$, $\tau = 1$, N é a quantidade de passos na variável t e α a ordem da derivada fracionária segundo Caputo-Fabrizio.

Fonte: O autor, 2023.

Experimento 5.2. *Agora, vamos abordar em um segundo exemplo, onde será exibido a solução numérica do problema* (128) *no domínio* $(x,t) \in [0,1] \times [0,2]$, *considerando os seguintes dados de entrada e os parâmetros* $\mathcal{K} = 1$, $\Delta x = 1/100$ *e* $\Delta t = 2/400$*:*

- $f(x,t) = (1+\pi^2 t)(1+t)\exp(t)\,\mathrm{sen}\,(\pi x)$;
- $\phi(x) = 0; \quad l(t) = 0; \quad r(t) = t\exp(t)$.

As Figuras 11a e 11b exibem cortes das soluções numéricas para diferentes valores de α e τ, respectivamente, onde utilizamos $\mathcal{K} = 1$, $\Delta x = 1/100$ e $\Delta t = 2/400$. Por outro lado, a Figura 11c apresenta o gráfico de superfície correspondente à utilização de $\Delta x = 1/20$ e $\Delta t = 2/40$.

5.1.3 Algoritmo do $MtER$ com a derivada de Caputo-Fabrizio

Nesta seção, apresentamos o algoritmo do método de Euler regressivo aplicado à equação da difusão fracionária com derivada fracionária segundo Caputo-Fabrizio. Esse algoritmo foi construído a partir da equação (132), o código escrito na linguagem Python está disponível no Apêndice B.6

Método tipo Euler Regressivo – Caputo-Fabrizio

Propósito:

Solução numérica da equação da difusão fracionária com Caputo-Fabrizio:

$$\frac{\partial u(x,t)}{\partial t} = {}_{CF}D^{\alpha}_{0,t}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema implícito)

Entrada:

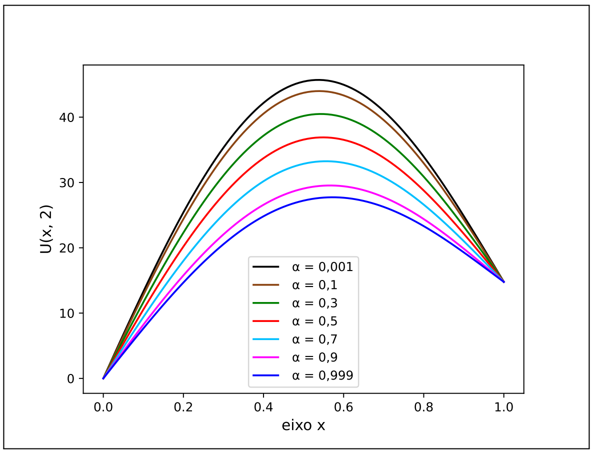
Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

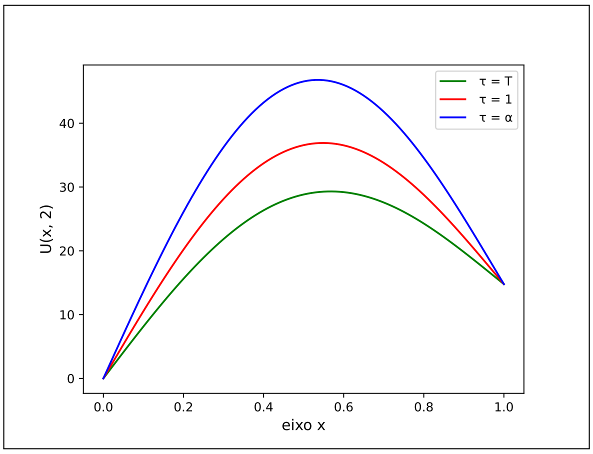
$\mathcal{K}$: coeficiente de difusão;

α: ordem da derivada fracionária;

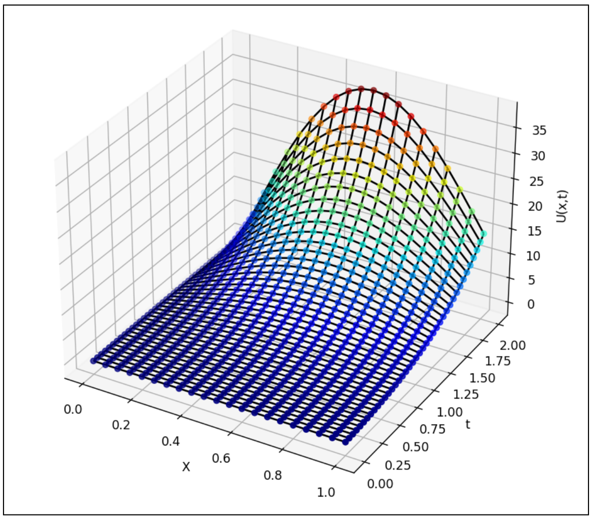
Figura 11 - Solução numérica da equação de difusão fracionária com a derivada de Caputo-Fabrizio.



(a)



(b)



(c)

Legenda: As Figuras (a) e (b) exibem cortes das soluções em $t = 2$. Na primeira figura, variamos o parâmetro α enquanto mantemos $\tau = 1$. Na segunda figura, fixamos $\alpha = 0,5$ e variamos o valor de τ. Já na Figura (c), é apresentado o gráfico de superfície para $\alpha = 0,5$ e $\tau = 1$.

Fonte: O autor, 2023.

τ: parâmetro de correção dimensional;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$s := \frac{\mathcal{K}}{\alpha \Delta x^2}\left[\exp\left(\frac{\alpha}{1-\alpha}\Delta t\right) - 1\right]$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $k = 0, 1, \ldots, N$ faça

$E_k = \exp\left(-\frac{\alpha}{1-\alpha}k\Delta t\right)$;

Passo 6

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $B_i^j = -2s$;

se $i == j-1$ ou $i == j+1$, então $B_i^j = s$;

se não $B_i^j = 0$, (matriz tridiagonal);

Passo 7

Crie, I, matriz identidade de ordem M;

Faça $n = 1$;

Enquanto $n \leq N$ faça

crie, V e W, vetores de ordem M;

para $k = 2, 3, \ldots, n$ faça

$V = B \times (U^{n-k+1} - U^{n-k})$;

$V_0 = V_0 + s \times (U_0^{n-k+1} - U_0^{n-k})$;

$V_{M-1} = V_{M-1} + s \times (U_M^{n-k+1} - U_M^{n-k})$;

$W = W + E_k \times V$;

$W = W + (I - E_1 \times B)U^{n-1}$;

$W_0 = W_0 + s \times E_1(U_0^n - U_0^{n-1})$;
$W_{M-1} = W_{M-1} + s \times E_1(U_M^n - U_M^{n-1})$;
resolva, $(I - E_1 \times B)U^n = W + \Delta t \times F^n$;
$n = n + 1$;

Passo 8

Para $n = 0, 1, \ldots, N$ faça
imprima U^n;

Passo 9

PARE (O procedimento está completo).

5.2 Derivada Fracionária de Atangana-Baleanu

No ano de 2016, Abdon Atangana e Dumitru Baleanu utilizaram o fato de que a função de Mittag-Leffler (MOSCHEN, 2006) generaliza a função exponencial e introduziram duas derivadas fracionárias: uma no sentido de Caputo e outra no sentido de Riemann-Liouville. Esses operadores de núcleo não local e não singular são denominados como a derivada fracionária de Atangana-Baleanu do tipo Caputo e do tipo Riemann-Liouville (ATANGANA; BALEANU, 2016). Essas formulação têm sido usadas em diversas aplicações, tais como: modelos de transferência de calor (ATANGANA; BALEANU, 2016), em modelos de fluidos (SHEIKH et al., 2017), dinâmicas populacionais (PETER et al., 2021), dentre outros.

Definição 17. *Sejam $f \in H^1(a,b)$, $b > a$ e $\alpha \in [0,1]$. A derivada fracionária segundo Atangana-Baleanu do tipo Caputo é dada por*

$$ {}_{ABC}D^{\alpha}_{a,t} f(t) := \frac{M(\alpha)}{1-\alpha} \int_a^t E_\alpha\left(-\alpha \frac{(t-s)^\alpha}{1-\alpha}\right) f'(s)ds, \tag{133}$$

onde $E_\alpha(\cdot)$ é a função de Mittag-Leffler de um parâmetro e $M(\alpha)$ é o análogo ao fator de normalização da derivada de Caputo-Fabrizio.

Note que, quando consideramos a ordem da derivada α igual a zero, não recuperamos a função original, exceto quando a função calculada no extremo inferior da integral é zero, isto é,

$$ {}_{ABC}D^{0}_{a,t} f(t) = M(0) \int_a^t E_0(0) f'(s)ds = \int_a^t f'(s)ds = f(t) - f(a).$$

Definição 18. *Sejam $f \in H^1(a,b)$, $b > a$ e $\alpha \in [0,1]$. A derivada fracionária segundo*

Atangana-Baleanu do tipo Riemann-Liouville é dada por

$$
{}_{ABR}D^{\alpha}_{a,t}f(t) := \frac{M(\alpha)}{1-\alpha}\frac{d}{dt}\int_a^t E_\alpha\left(-\alpha\frac{(t-s)^\alpha}{1-\alpha}\right)f(s)ds, \tag{134}
$$

onde $E_\alpha(\cdot)$ e $M(\alpha)$ são dados como na Definição 17.

Na formulação de Atagana-Baleanu do tipo Riemann-Liouville, conseguimos recuperar a função original quando a ordem da derivada α for igual a zero. De fato,

$$
{}_{ABR}D^{0}_{a,t}f(t) = M(0)\frac{d}{dt}\int_a^t E_0(0)f(s)ds = \frac{d}{dt}\int_a^t f(s)ds = f(t).
$$

Uma outra diferença entre os dois tipos da derivada de Atagana-Baleanu é em relação à derivada de uma constante C, uma vez que ${}_{ABC}D^{\alpha}_{a,t}C = 0$, o que não ocorre para ${}_{ABR}D^{\alpha}_{a,t}C$. As duas derivadas de Atangana-Baleanu são relacionadas através da relação (TEODORO, 2019)

$$
{}_{ABR}D^{\alpha}_{a,t}f(t) = {}_{ABC}\, D^{\alpha}_{a,t}f(t) + \frac{M(\alpha)}{1-\alpha}E_\alpha\left(-\alpha\frac{(t-a)^\alpha}{1-\alpha}\right)f(a). \tag{135}
$$

As derivadas de Atangana-Baleanu, assim como as de Caputo-Fabrizio, não podem ser consideradas derivadas fracionárias, segundo o critério estabelecido por Ortigueira e Machado (ORTIGUEIRA; MACHADO, 2015). As derivadas de Atangana-Baleanu são operadores lineares; vale a generalização da regra de Leibniz. As derivadas de ordem um coincidem com a derivada ordinária. A derivada ${}_{ABR}D^{\alpha}_{0,t}$ de ordem zero recupera a função original, e a derivada ${}_{ABC}D^{\alpha}_{0,t}$ de ordem zero só recupera a função original se $f(a) = 0$, conforme já mencionado, e não é válida a lei dos expoentes (TEODORO, 2019).

5.2.1 Método tipo Euler regressivo com Atagana-Baleanu

Nesta seção é abordado o problema da difusão fracionária com derivada fracionária de Atagana-Baleanu do tipo Caputo. As derivadas de ordem inteira no problema (15) são aproximadas pelas diferenças regressivas (35) e centradas (38). Aplicando o operador de Atagana-Baleanu do tipo Caputo em (15a), temos que o problema (15) é escrito da seguinte maneira, com $i = 1, 2, \ldots, M-1$ e $n = 1, 2, \ldots, N$:

$$\frac{U_i^n - U_i^{n-1}}{\Delta t} = \mathcal{K}\tau^\alpha \, {}_{ABC}D_{0,t}^\alpha \left(\frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2} \right) + f(x_i, t_n), \tag{136a}$$

$$U_i^0 = \phi(x_i) \quad (i = 0, 1, \ldots, M), \tag{136b}$$

$$U_0^n = l(t_n) \quad (n = 0, 1, \ldots, N), \tag{136c}$$

$$U_M^n = r(t_n) \quad (n = 0, 1, \ldots, N). \tag{136d}$$

A definição da derivada de Atangan-Baleanu do tipo Caputo é definida para $\alpha \in$ [0, 1], mas o esquema de aproximação proposto por Yadav, Pandey e Shurla (YADAV; PANDEY; SHUKLA, 2019) resume o caso quando $\alpha/(1-\alpha) \leq 1$, isto é, $\alpha \leq 1/2$. O esquema é construído por meio do uso da fórmula de diferenças finitas de dois pontos para a derivada primeira. Supondo que $f(t) \in H^1(a, b)$ para $t = t_n$, temos

$$\begin{aligned}
{}_{ABC}D_{0,t}^\alpha f(t)|_{t=t_n} &= \frac{M(\alpha)}{1-\alpha} \int_0^{t_n} f'(s) E_\alpha \left[\frac{-\alpha}{1-\alpha} (t_n - s)^\alpha \right] ds \\
&= \frac{M(\alpha)}{1-\alpha} \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} f'(s) E_\alpha \left[\frac{-\alpha}{1-\alpha} (t_n - s)^\alpha \right] ds \\
&= \frac{M(\alpha)}{1-\alpha} \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} \frac{f_{k+1} - f_k}{\Delta t} E_\alpha \left[\frac{-\alpha}{1-\alpha} (t_n - s)^\alpha \right] ds + R_n \\
&= \frac{M(\alpha)}{1-\alpha} \sum_{k=0}^{n-1} \frac{f_{k+1} - f_k}{\Delta t} \left\{ (t_n - t_k) E_{\alpha,2} \left[\frac{-\alpha}{1-\alpha} (t_n - t_k)^\alpha \right] \right. \\
&\quad \left. - (t_n - t_{k+1}) E_{\alpha,2} \left[\frac{-\alpha}{1-\alpha} (t_n - t_{k+1})^\alpha \right] \right\} + R_n \\
&= \frac{M(\alpha)}{1-\alpha} \sum_{k=0}^{n} \mathcal{C}_k^n f(t_k) + R_n
\end{aligned} \tag{137}$$

onde, se representarmos $E_{\alpha,2} \left[\frac{-\alpha}{1-\alpha} (t_n - t_k)^\alpha \right]$ com E_k^n, então

$$\mathcal{C}_k^n := \begin{cases} (n-1)E_1^n - nE_0^n, & \text{se} \quad k = 0, \\ (n-k+1)E_{k-1}^n - 2(n-k)E_k^n + (n-k-1)E_{k+1}^n, & \text{se} \quad 0 < k < n, \\ E_{n-1}^n, & \text{se} \quad k = n, \end{cases} \tag{138}$$

e R_n é o erro de truncamento (YADAV; PANDEY; SHUKLA, 2019).

Aplicando o esquema numérico (137) à equação (136a), para $0 < \alpha < 1/2$, temos

$$\frac{U_i^n - U_i^{n-1}}{\Delta t} = \mathcal{K}\tau^\alpha \frac{M(\alpha)}{1-\alpha} \sum_{k=0}^{n} \mathcal{C}_k^n \left(\frac{U_{i-1}^k - 2U_i^k + U_{i+1}^k}{\Delta x^2} \right) + f_i^n; \tag{139}$$

tomando a função de amortização como $M(\alpha) = 1$, chegamos a

$$U_i^n = U_i^{n-1} + \frac{\mathcal{K}\tau^\alpha \Delta t}{(1-\alpha)\Delta x^2} \sum_{k=0}^{n} \mathcal{C}_k^n \left(U_{i-1}^k - 2U_i^k + U_{i+1}^k \right) + \Delta t f_i^n. \tag{140}$$

Para $1 \leq i \leq M-1$, a forma matricial para (140) fica:

$$\boldsymbol{U^n} = \boldsymbol{U^{n-1}} + \sum_{k=0}^{n} \mathcal{C}_n^k \left(D\boldsymbol{U^k} + \boldsymbol{C^k} \right) + \Delta t \boldsymbol{f^n} \tag{141}$$

ou

$$\left(\boldsymbol{I} - \mathcal{C}_n^n D\right) \boldsymbol{U^n} = \boldsymbol{U^{n-1}} + \sum_{k=0}^{n-1} \mathcal{C}_k^n \left(D\boldsymbol{U^k} + \boldsymbol{C^k} \right) + \mathcal{C}_n^n \boldsymbol{C^n} + \Delta t \boldsymbol{f^n} \tag{142}$$

onde os vetores $\boldsymbol{U^n}$, $\boldsymbol{f^n}$ e $\boldsymbol{C^n}$ são definidos exatamente como na Seção 3.2 enquanto a matriz

$$D := \begin{pmatrix} -2v & v & 0 & \dots & 0 \\ v & -2v & v & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & v & -2v & v \\ 0 & \dots & 0 & v & -2v \end{pmatrix},$$

sendo $v := \mathcal{K}\tau^\alpha \Delta t / \left[(1-\alpha)\Delta x^2\right]$.

A dependência para um ponto calculado na equação da difusão fracionária pelo método tipo Euler regressivo com a formulação de Atagana-Baleanu do tipo Caputo é igual àquela obtida com a derivada de Riemann-Liouville, conforme ilustrado nas Figuras 5a, 5b e 5c.

5.2.2 Experimento numérico do $MtER$ com Atagana-Baleanu

Nesta seção, vamos mostrar resultados numéricos do problema da difusão fracionária com a derivada fracionária de Atagana-Baleanu do tipo Caputo pelo o método tipo Euler regressivo.

Experimento 5.3. *A função* $u(x,t) = t^4 \sin(\pi x)$ *é a solução encontrada para o problema* (136)*, com os seguintes dados.*

- $f(x,t) = \left(4t^3 + \frac{24\pi^2 t^4}{1-\alpha} E_{\alpha,5}\left[\frac{-\alpha}{1-\alpha} t^\alpha\right]\right) \sin(\pi x)$;
- $\phi(x) = 0; \quad l(t) = 0; \quad r(t) = 0.$

A Tabela 7 exibe os resultados numéricos encontrados, relativos ao domínio $(x,t) \in [0,1] \times [0,1]$*, quando fixados* $\mathcal{K} = 1$*,* $\Delta x = 1/1000$ *e* $\tau = 1$*.*

Tabela 8 - Método tipo Euler regressivo com derivada fracionária de Atagana-Baleanu do tipo Caputo.

N	$\alpha = 0,1$	ordem	$\alpha = 0,3$	ordem	$\alpha = 0,5$	ordem
10	$3,2976 \times 10^{-2}$		$3,1879 \times 10^{-2}$		$3,7544 \times 10^{-2}$	
20	$1,6957 \times 10^{-2}$	0,9595	$1,6155 \times 10^{-2}$	0,9806	$1,8416 \times 10^{-2}$	1,0276
40	$8,5972 \times 10^{-3}$	0,9697	$8,1265 \times 10^{-3}$	0,9860	$9,1016 \times 10^{-3}$	1,0222
80	$4,3285 \times 10^{-3}$	0,9765	$4,0744 \times 10^{-3}$	0,9893	$4,5208 \times 10^{-3}$	1,0180
160	$2,1718 \times 10^{-3}$	0,9811	$2,0398 \times 10^{-3}$	0,9915	$2,2524 \times 10^{-3}$	1,0148

Legenda: Erro na norma L^2 com $\mathcal{K} = 1$, $\Delta x = 1/1000$, $\tau = 1$, N é a quantidade de passos na variável t e α a ordem da derivada fracionária segundo Atagana-Baleanu do tipo Caputo.

Fonte: O autor, 2023.

Os resultados exibidos na Tabela 8 podem ser considerados bons, visto que obtivemos boas aproximações e a ordem de convergência está em concordância com as aproximações empregadas.

Experimento 5.4. *Agora, vamos abordar em um segundo exemplo, onde será exibido a solução numérica do problema* (136) *no domínio* $(x, t) \in [0, 1] \times [0, 2]$, *considerando os seguintes dados de entrada:*

- $f(x, t) = (1 + \pi^2 t)(1 + t)\exp(t)\,\text{sen}\,(\pi x)$;
- $\phi(x) = 0$; $\quad l(t) = 0$; $\quad r(t) = t\exp(t)$.

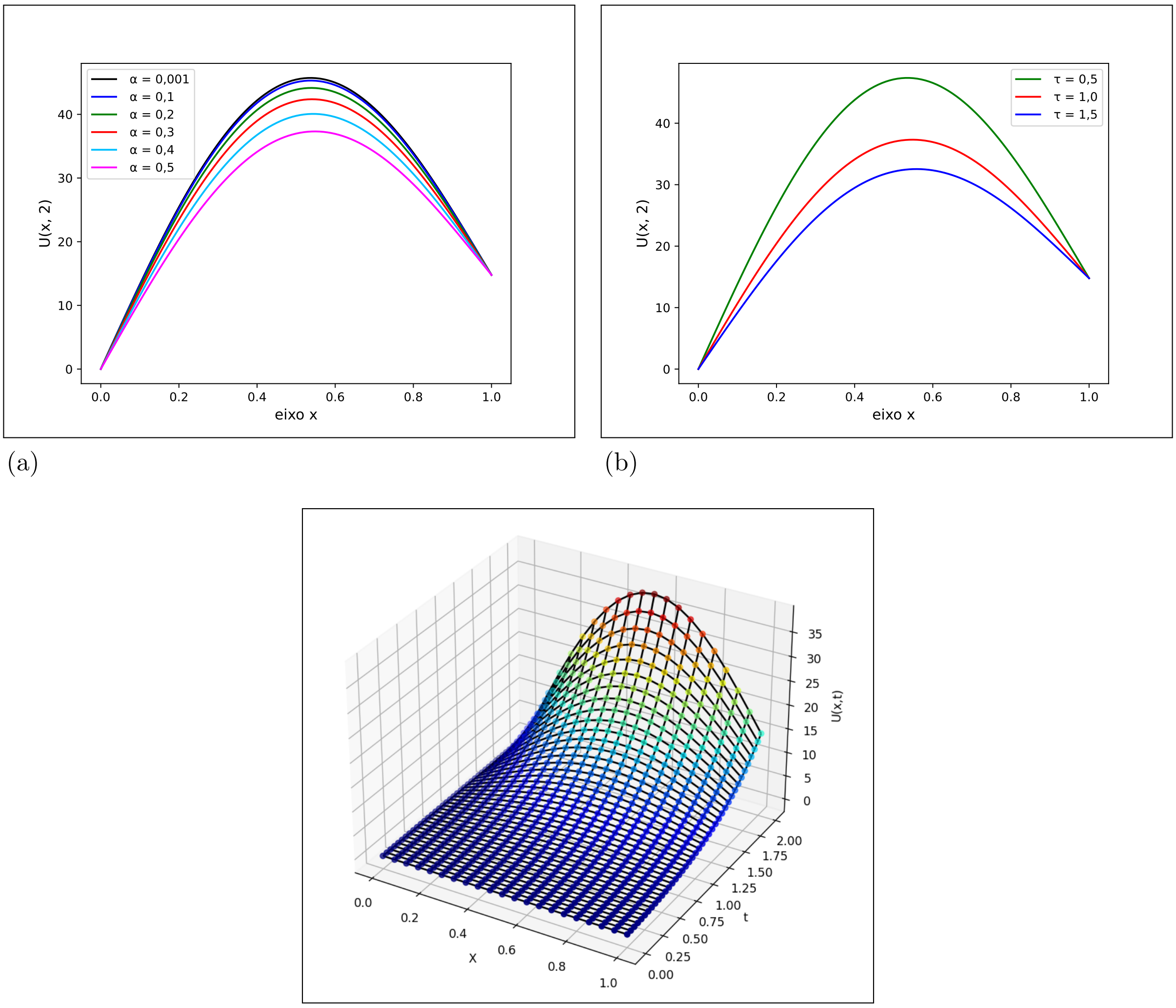
As Figuras 12a e 12b exibem cortes das soluções numéricas para diferentes valores de α e τ, respectivamente, onde utilizamos $\mathcal{K} = 1$, $\Delta x = 1/100$ e $\Delta t = 2/400$. Por outro lado, a Figura 12b apresenta o gráfico de superfície correspondente à utilização de $\Delta x = 1/20$ e $\Delta t = 2/40$.

Os resultados exibidos na Figura 12a ficaram em concordância com os diferentes valores de α considerados. Contudo, para $\alpha = 0,999$, não houve convergência do método, o que nos motiva a investigar o esquema que aproxima a derivada de Atangana-Baleanu do tipo Caputo, bem como o código.

5.2.3 Algoritmo do $MtER$ com a derivada de Atagana-Baleanu

Nesta seção, apresentamos o algoritmo do método de Euler regressivo aplicado à equação da difusão fracionária com derivada fracionária de Atagana-Baleanu do tipo Caputo. Esse algoritmo foi construído a partir da equação (142), o código escrito na linguagem Python está disponível no Apêndice B.7

Figura 12 - Solução numérica da equação de difusão fracionária com a derivada de Atagana-Baleanu do tipo Caputo.



(a) (b) (c)

Legenda: As Figuras (a) e (b) exibem cortes das soluções em $t = 2$. Na primeira figura, variamos o parâmetro α enquanto mantemos $\tau = 1$. Na segunda figura, fixamos $\alpha = 0,5$ e variamos o valor de τ. Já na Figura (c), é apresentado o gráfico de superfície para $\alpha = 0,5$ e $\tau = 1$.

Fonte: O autor, 2023.

Método tipo Euler Regressivo – Atagana-Baleanu

Propósito:

Solução numérica da equação da difusão fracionária com a derivada fracionária de Atagana-Baleanu do tipo Caputo,

$$\frac{\partial u(x,t)}{\partial t} = {}_{ABC}D^{\alpha}_{0,t}\left(\tau^{\alpha}\mathcal{K}\frac{\partial^2 u(x,t)}{\partial x^2}\right) + f(x,t), \quad 0 < x < L, \quad 0 < t \leq T,$$

sujeita às condições de contorno

$$u(0,t) = l(t) \quad u(L,t) = r(t), \quad 0 \leq t \leq T,$$

e às condições iniciais

$$u(x,0) = \phi(x), \quad 0 \leq x \leq L.$$

Método:

Diferenças finitas (esquema implícito)

Entrada:

Extremidade L, instante final T;

M: número de nós em x, N: número de nós em t;

$\mathcal{K}$: coeficiente de difusão;

α: ordem da derivada fracionária;

τ: parâmetro de correção dimensional;

Saída:

Aproximações U_i^n para cada $i = 0, 1, \ldots, M$ e para cada $n = 0, 1, \ldots, N$;

Passo 1

Faça

$\Delta x = L/M$, (calcula o comprimento do passo Δx);

$\Delta t = T/M$, (calcula o comprimento do passo Δt);

$v = \tau^{\alpha}\Delta t/[(1-\alpha)\Delta x^2]$;

Passo 2

Para $i = 0, 1, \ldots, M$ faça

$U_i^0 = \phi(i\Delta x)$, (valores iniciais);

Passo 3

Para $n = 0, 1, \ldots, N$ faça

$U_0^n = l(n\Delta t)$, (condição na fronteira esquerda);

$U_M^n = r(n\Delta t)$, (condição na fronteira direita);

Passo 4

Para $n = 0, 1, \ldots, N$ faça

para $i = 1, 2, \ldots, M-1$ faça

$F_i^n = f(i\Delta x, n\Delta t)$, (termo fonte);

Passo 5

Para $n = 0, 1, \ldots, N$ faça

para $k = 1, 2, \ldots, M-1$ faça

se $k == 0$, então $\mathcal{C}_k^n = (n-1)E_1^n - nE_0^n$;

se $k == n$, então $\mathcal{C}_k^n = E_{n-1}^n$;

senão $\mathcal{C}_k^n = (n-k+1)E_{k-1}^n - 2(n-k)E_k^n + (n-k-1)E_{k+1}^n$;

Passo 6

Para $i = 0, 1, \ldots, M-1$ faça

para $j = 0, 1, \ldots, M-1$ faça

se $i == j$, então $D_i^j = -2v$;

se $i == j-1$ ou $i == j+1$, então $D_i^j = v$;

senão $D_i^j = 0$, (matriz tridiagonal);

Passo 7

Crie, I, matriz identidade de ordem M;

Faça $n = 1$;

Enquanto $n \leq N$ faça

crie, V e W, vetores de ordem M;

para $k = 0, 1, \ldots, n-1$ faça

$V = D \times U^k$;

$V_0 = V_0 + v \times U_0^k$;

$V_{M-1} = V_{M-1} + v \times U_M^k$;

$W = W + \mathcal{C}_k^n \times V$;

$W_0 = W_0 + v \times \mathcal{C}_n^n U_0^n$;

$W_{M-1} = W_{M-1} + v\mathcal{C}_n^n U_M^n$;

resolva, $(I - \mathcal{C}_n^n \times D)U^n = U^{n-1} + W + \Delta t \times F^n$;

$n = n + 1$;

Passo 8

Para $n = 0, 1, \ldots, N$ faça

imprima U^n;

Passo 9

PARE (O procedimento está completo).

---

6 RESULTADOS NUMÉRICOS

Neste capítulo, realizamos experimentos para verificar o efeito de α e τ na modelagem. Analisamos o tempo de execução dos códigos e a modelagem com cada operador. Os códigos dos exemplos foram escritos em linguagem Python 3 e executados no Notebook Colaboratory[19].

6.1 Efeito da ordem α

Nesta seção, avalia-se a eficiência dos códigos à medida que a ordem α se aproxima de zero e de um. Compararemos esses resultados extremos com as ordens inteiras $\alpha = 0$ e $\alpha = 1$. Note que a derivada fracionária de ordem zero de uma função é a própria função[20] e que, quando a ordem é um número inteiro igual a um, deve produzir o mesmo resultado que a primeira derivada ordinária[21]. A escolha de $\alpha = 0$ faz com que a equação da difusão fracionária se reduza ao caso clássico, a saber,

$$u_t = \mathcal{K} u_{xx} + f, \tag{143}$$

este problema foi aproximado pelo método de Crank-Nicolson. Tomando agora $\alpha = 1$ se obtém a seguinte equação

$$u_t = \mathcal{K} u_{xxt} + f, \tag{144}$$

sendo utilizada a solução analítica no experimento a seguir.

Experimento 6.1. *Este experimento visa, sobretudo, comparar as soluções numéricas para diferentes valores de α com as derivadas de Riemann-Liouville, Caputo, Chen, Katugampola, Caputo-Fabrizio e Atangana-Baleanu do tipo Caputo. Considere o problema da difusão fracionária, no domínio $(x,t) \in [0,1] \times [0,2]$, estando os dados de entrada descritos a seguir:*

- $f(x,t) = (1 + 25\pi^2/36)(1+t)\exp(t)\,\mathrm{sen}\,(5\pi x/6)$;
- $\phi(x) = 0$; $\quad l(t) = 0$; $\quad r(t) = 0,5t\exp(t)$;

com os parâmetros $\mathcal{K} = 1$, $\tau = 1$, $\Delta x = 1/50$ e $\Delta t = 2/200$.

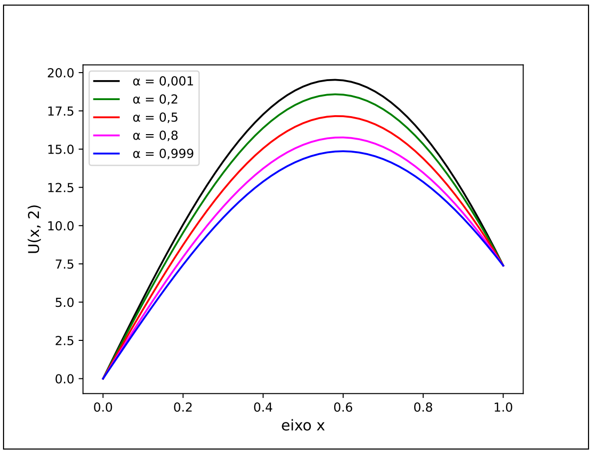
---

[19] O Colaboratory ou "Colab" é um produto do Google Research, que permite que qualquer pessoa escreva e execute código Python pelo navegador.

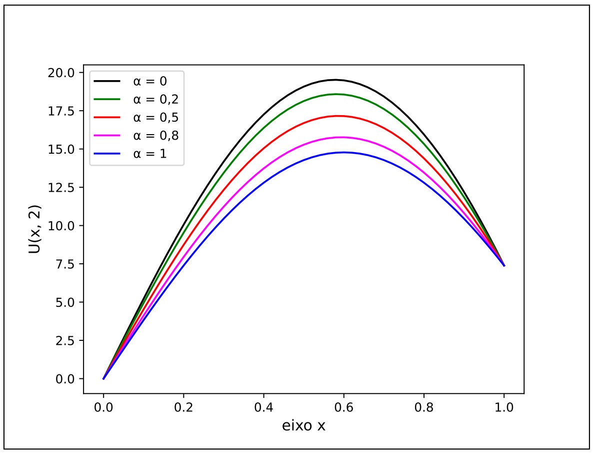
[20] Essa afirmação é equivalente a o operador satisfazer a propriedade do critério de Ortigueira e Machado.

[21] Um caso particular da terceira propriedade do critério de Ortigueira e Machado.

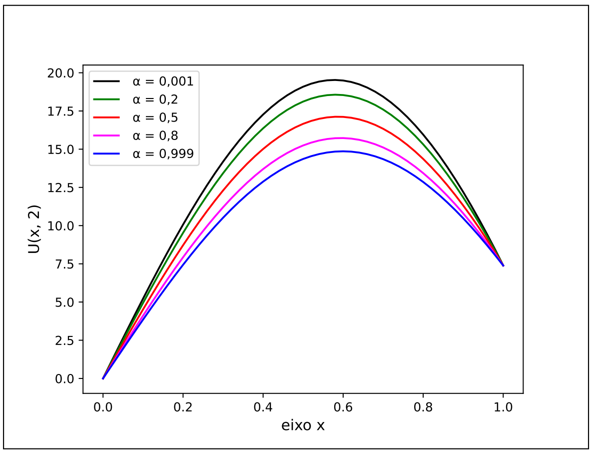
Figura 13 - Solução numérica da equação de difusão fracionária com as derivadas clássicas



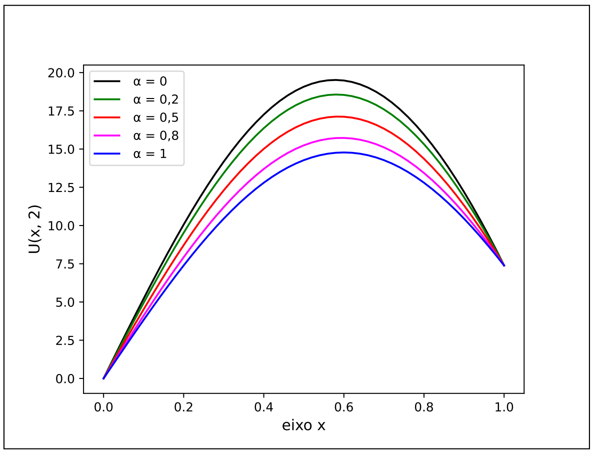
(a)



(b)



(c)

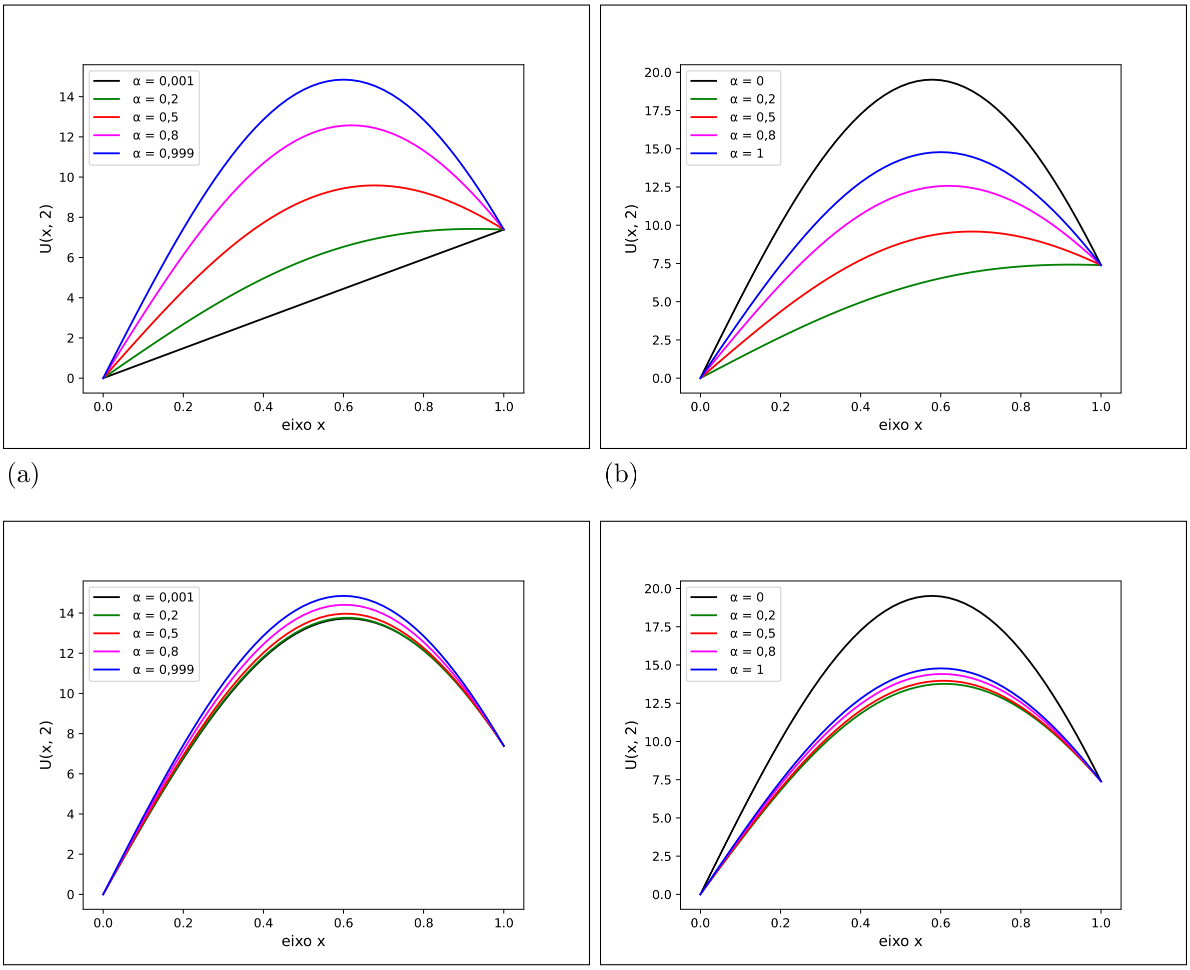


(d)

Legenda: As figuras (13a), (13b), (13c) e (13d) exibem cortes das soluções da equação da difusão fracionária com a derivada de Riemann-Liouville (a) e (b) e Caputo (c) e (d) em $t = 2$. Os parâmetros utilizados foram $\mathcal{K} = 1$, $\tau = 1$, $\Delta x = 1/50$ e $\Delta t = 2/200$. Para $\alpha = 0$ e $\alpha = 1$, a solução corresponde às equações (143) e (144), respectivamente.

Fonte: O autor, 2023.

Figura 14 - Solução numérica da equação de difusão fracionária com as derivadas locais

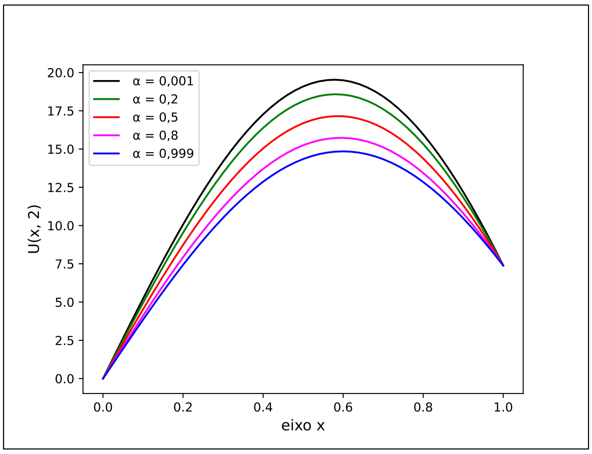


(a) (b) (c) (d)

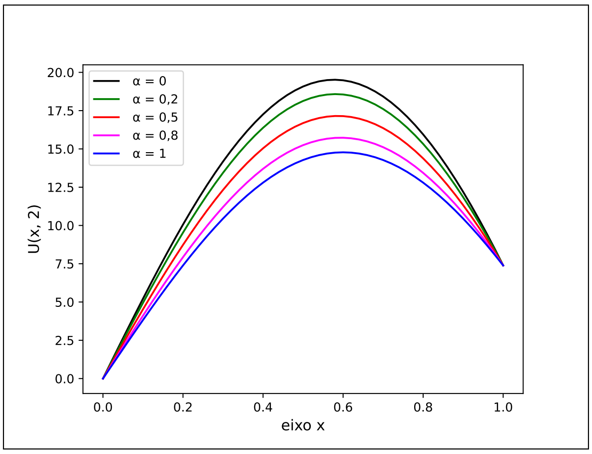
Legenda: As figuras (14a), (14b), (14c) e (14d) exibem cortes das soluções da equação da difusão fracionária com a derivada de Chen (a) e (b) e Katugampola (c) e (d) em $t = 2$. Os parâmetros utilizados foram $\mathcal{K} = 1$, $\tau = 1$, $\Delta x = 1/50$ e $\Delta t = 2/200$. Para $\alpha = 0$ e $\alpha = 1$, a solução corresponde às equações (143) e (144), respectivamente.

Fonte: O autor, 2023.

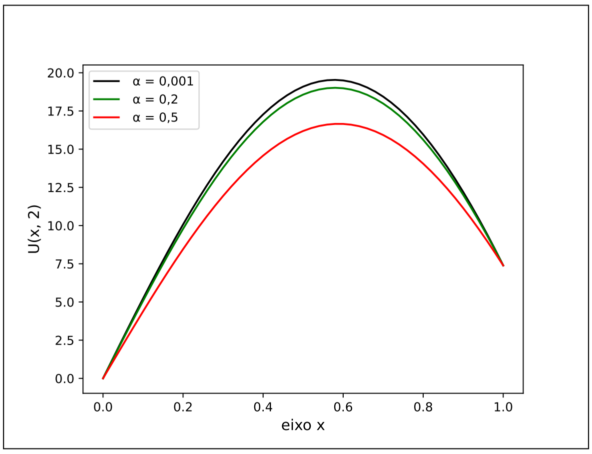
Figura 15 - Solução numérica da equação de difusão fracionária com as derivadas com núcleo não singular



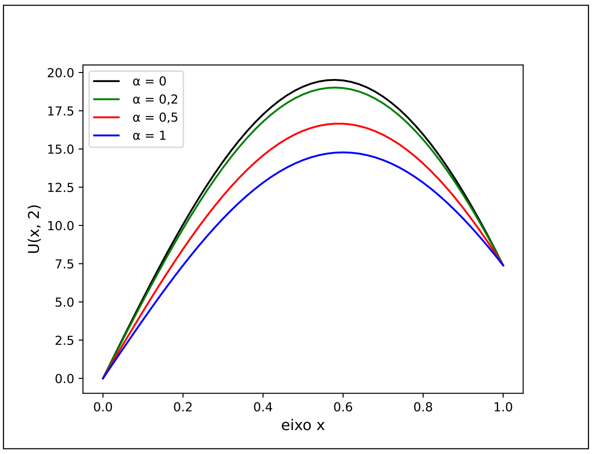
(a)



(b)



(c)



(d)

Legenda: As figuras (15a), (15b), (15c) e (15d) exibem cortes das soluções da equação da difusão fracionária com a derivada de Caputo-Fabrizio (a) e (b) e Atangana-Baleanu (c) e (d) em $t = 2$. Os parâmetros utilizados foram $\mathcal{K} = 1$, $\tau = 1$, $\Delta x = 1/50$ e $\Delta t = 2/200$. Para $\alpha = 0$ e $\alpha = 1$, a solução corresponde às equações (143) e (144), respectivamente.

Fonte: O autor, 2023.

As derivadas clássicas de Riemann-Liouville e de Caputo, exibidas na Figura 13, satisfazem a segunda condição ($D^0_{a,t}f(t) = f(t)$) e a terceira condição ($D^1_{a,t}f(t) = f'(t)$) do critério de Ortigueira e Machado. As soluções numéricas da equação da difusão, tanto com a derivada de Riemann-Liouville quanto com a derivada de Caputo em $\alpha = 0,001$ e $\alpha = 0,999$, aproximam-se das respectivas soluções inteiras em $\alpha = 0$ e $\alpha = 1$, confirmando assim os resultados teóricos.

A Figura (14a), referente à derivada de Chen, e (14c), referente à de Katugampola, mostram que as soluções numéricas estão em conformidade com os diferentes valores de α. Contudo, esses operadores locais não satisfazem a segunda condição do critério de Ortigueira e Machado, ou seja, para a derivada de Chen, temos (${}_{CH}D^0 t f(t) = \infty$) e para Katugampola (${}_{K}D^0 t f(t) = t f'(t)$). Isso explica a falta de conformidade das soluções numéricas em relação aos valores fracionários de α com os valores inteiros em ambos os operadores, como exibido nas Figuras (14b) e (14d).

Para este experimento, as derivadas de Caputo-Fabrizio e Atangana-Baleanu do tipo Caputo satisfazem a segunda e a terceira condição do critério de Ortigueira e Machado. Portanto, os resultados esperados são semelhantes aos obtidos com as derivadas de Riemann-Liouville e de Caputo, como é observado nas Figuras (15a) e (15b), referentes à de Caputo-Fabrizio. Contudo, devido à limitação do esquema empregado para a derivada de Atangana-Baleanu, não é possível realizar toda a análise; no entanto, os resultados exibidos nas Figuras (15c) e (15d) estão dentro do que se esperava.

6.2 Efeito do parâmetro τ

Nesta seção será estudado o efeito causado quando considerados diferentes valores para o parâmetro de correção dimensional τ que foi inserido para corrigir o desbalanceamento da equação da difusão fracionária.

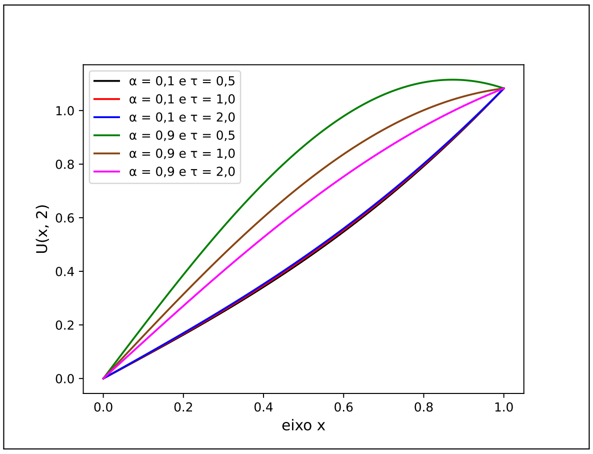
Experimento 6.2. *Este experimento visa verificar o efeito da variação do parâmetro de correção dimensional* τ *para os valores* $(0,5)$*,* $(1,0)$ *e* $(2,0)$ *na solução numérica, mantendo fixa a ordem* α *da derivada fracionária. O problema considerado é o da difusão fracionária, no domínio* $(x,t) \in [0,1] \times [0,2]$*, com os dados de entrada:*

- $f(x,t) = 4(1 + \frac{\pi^2}{4})(1-t)\exp(-t)\,\mathrm{sen}\,(\frac{\pi x}{2})$;
- $\phi(x) = 0; \quad l(t) = 0; \quad r(t) = 4t\,\mathrm{sen}\,(\frac{\pi x}{2})$;

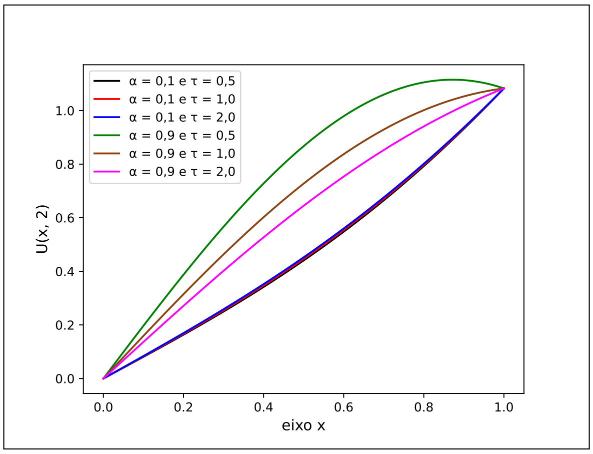
com os parâmetros $\mathcal{K} = 1$*,* $\Delta x = 1/100$ *e* $\Delta t = 2/500$*.*

Os resultados do Experimento 6.2 exibidos nas Figuras (16), (17) e (18) indicam que diferentes valores do parâmetro de correção dimensional τ podem ter em algumas situações forte influência na modelagem. Os resultados obtidos com os operadores de

Figura 16 - Solução numérica da equação de difusão fracionária com a derivada de Riemann-Liouville (a) e Caputo (b).



(a)

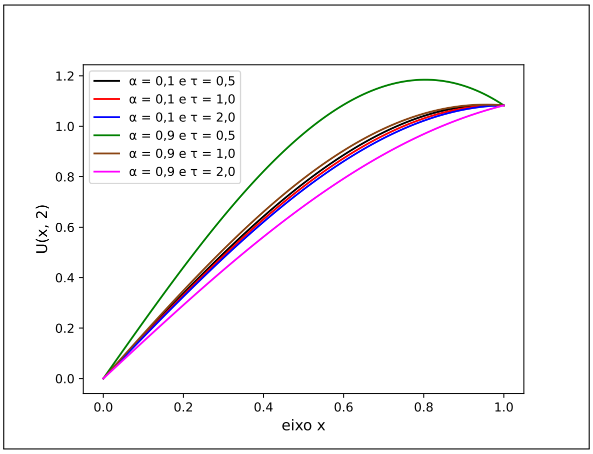


(b)

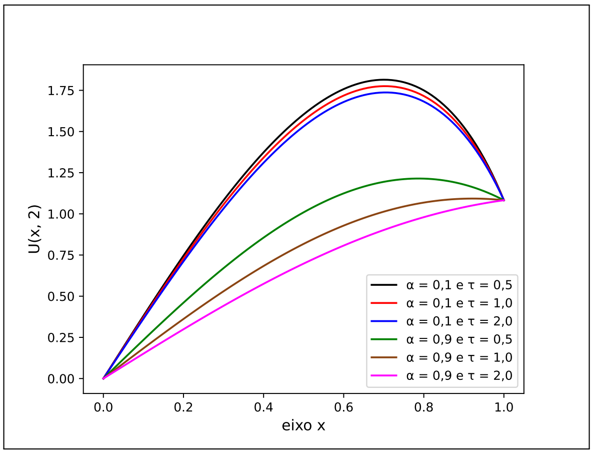
Legenda: As figuras exibem cortes das soluções em $t = 2$, onde foram utilizados $\Delta x = 1/100$ e $\Delta t = 2/500$. As Figuras (a) e (b) mostram a solução quando variamos o parâmetro de correção dimensional τ, mantendo α fixo.

Fonte: O autor, 2023.

Figura 17 - Solução numérica da equação de difusão fracionária com a derivada de Chen (a) e Katugampola (b).



(a)

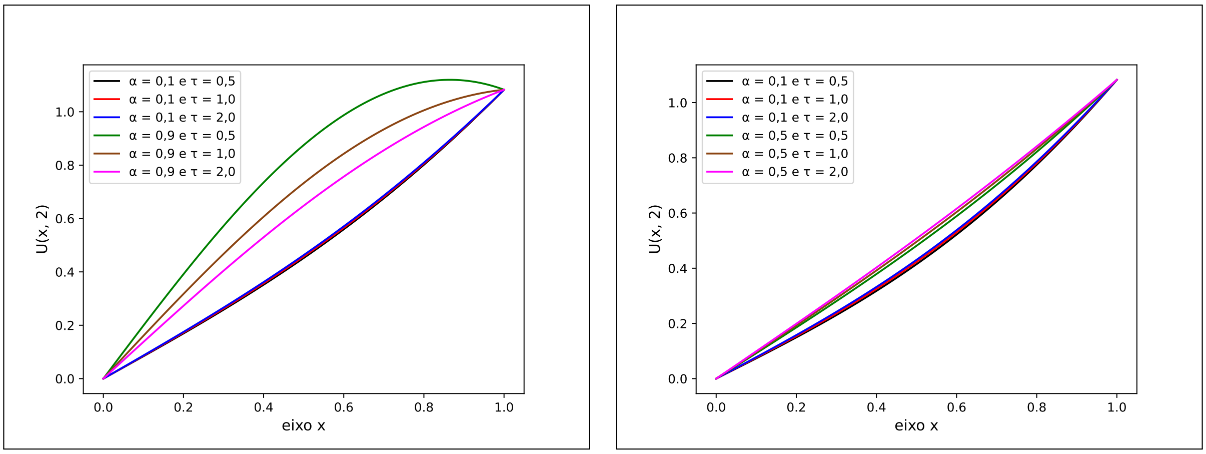


(b)

Legenda: As figuras exibem cortes das soluções em $t = 2$, onde foram utilizados $\Delta x = 1/100$ e $\Delta t = 2/500$. As Figuras (a) e (b) mostram a solução quando variamos o parâmetro de correção dimensional τ, mantendo α fixo.

Fonte: O autor, 2023.

Figura 18 - Solução numérica da equação de difusão fracionária com a derivada de Caputo-Fabrizio (a) e Atangana-Baleanu (b).



(a) (b)

Legenda: As figuras exibem cortes das soluções em $t = 2$, onde foram utilizados $\Delta x = 1/100$ e $\Delta t = 2/500$. As Figuras (a) e (b) mostram a solução quando variamos o parâmetro de correção dimensional τ, mantendo α fixo.

Fonte: O autor, 2023.

Riemann-Liouville, Caputo e Caputo-Fabrizio foram quantitativamente bem próximos com uma forte influência do τ quando a ordem $\alpha = 0,9$. Os resultados obtidos pelos operadores locais mostram também que para $\alpha = 0,9$ o parâmetro τ teve grande influência nas modelagem. Já para a derivada de Atangana-Baleanu os diferentes valores de τ não tiveram significativa influência no nível da ordem de aproximação mostrada.

6.3 Análise do desempenho de códigos

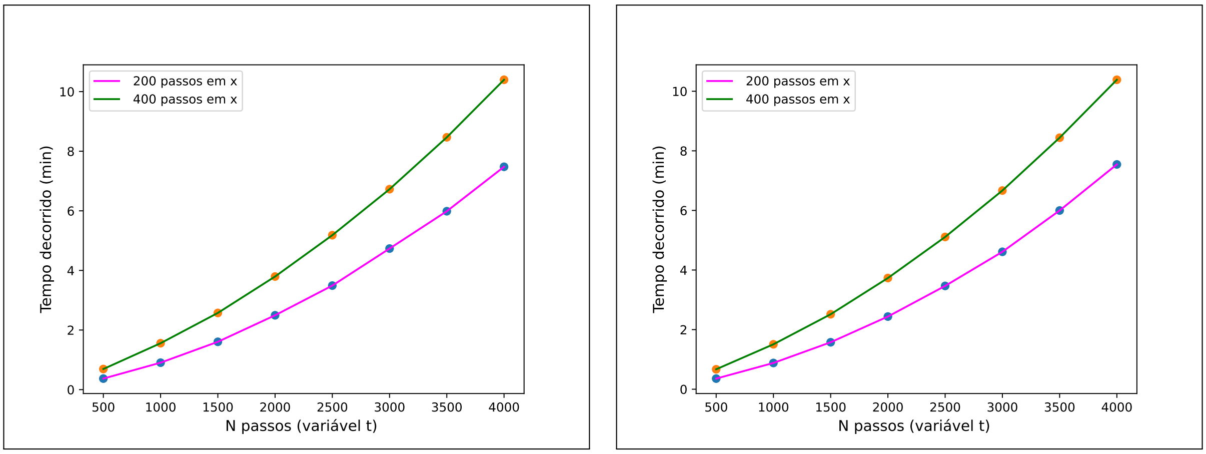
Nesta seção, será apresentado um experimento para medir o tempo de execução dos diferentes códigos propostos neste trabalho. Este teste visa não apenas quantificar o tempo necessário para a conclusão de tarefas específicas, mas também fornecer *insights* valiosos sobre a eficácia relativa dos algoritmos utilizados.

Experimento 6.3. *Este experimento visa, verificar o tempo de execução dos códigos propostos neste trabalho, no domínio* $(x, t) \in [0, 1] \times [0, 1]$, *com os dados:*

- $f(x, t) = (1,5t^{0,5} - \pi^2) \exp{(\pi x)} \operatorname{sen}{(\pi x)}$;
- $\phi(x) = 0; \quad l(t) = 0; \quad r(t) = 0;$

com os parâmetros $\mathcal{K} = 1$, $\alpha = 0,5$ *e* $\tau = 1$.

Figura 19 - Análise do desempenho dos Algoritmos (B.2) e (B.3).

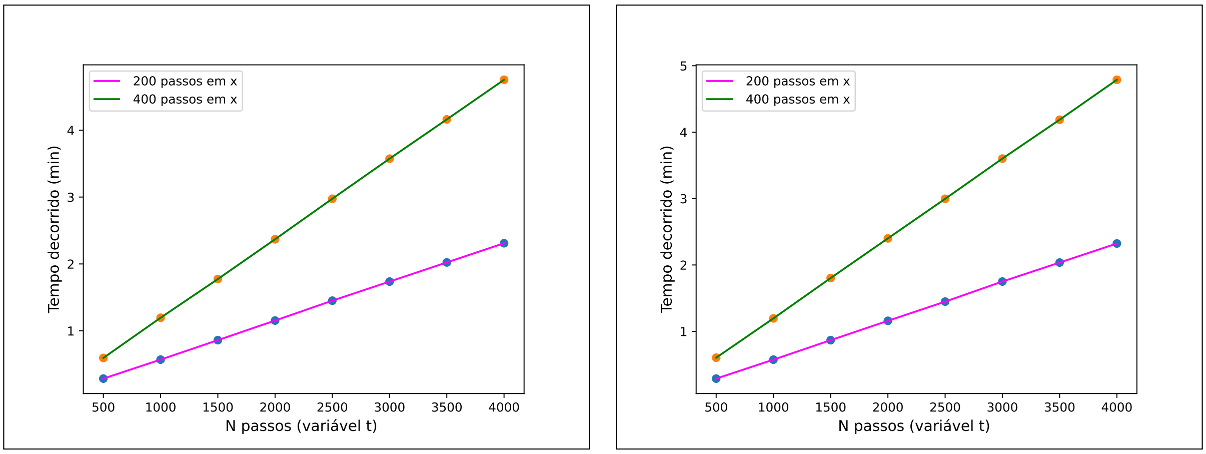


(a) (b)

Legenda: As figuras (a) e (b) apresentam o tempo de execução dos códigos aplicados ao problema da difusão fracionária, utilizando as derivadas clássicas de Riemann-Liouville e Caputo, respectivamente.

Fonte: O autor, 2023.

Figura 20 - Análise do desempenho dos Algoritmos (B.4) e (B.5).

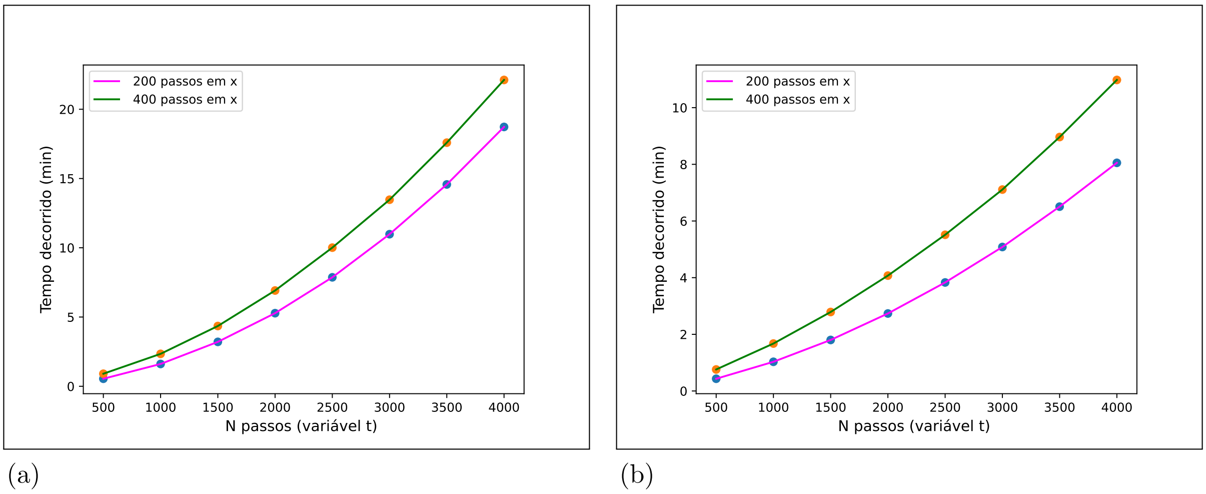


(a) (b)

Legenda: As figuras (a) e (b) apresentam o tempo de execução dos códigos aplicados ao problema da difusão fracionária, utilizando as derivadas locais de Chen e Katugampola, respectivamente.

Fonte: O autor, 2023.

Figura 21 - Análise do desempenho dos Algoritmos (B.6) e (B.7).



Legenda: As figuras (a) e (b) apresentam o tempo de execução dos códigos aplicados ao problema da difusão fracionária, utilizando as derivadas de núcleo não singular de Caputo-Fabrizio e de Atangana-Baleanu, respectivamente.
Fonte: O autor, 2023.

As Figuras 19, 20 e 21 exibem o tempo de execução dos códigos referidos. Os algoritmos na linguagem Python 3 foram executados nos seguintes contextos:

- Sistema operacional: Windows 11 Home Single Language;
- Processador (CPU): 11th Gen Intel(R) Core(TM) i5;
- Memória (RAM): 8,00 GB;
- Ambiente de desenvolvimento: PyCharm Community Edition 2023.

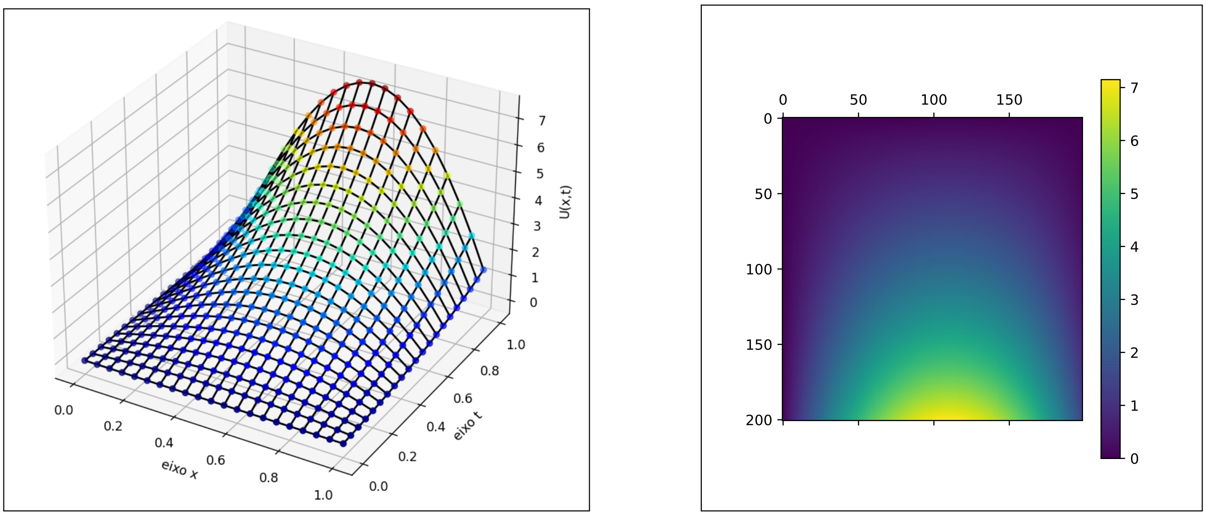
Os resultados exibidos nas Figuras 19, 20 e 21 ficaram dentro do esperado. A complexidade de tempo dos códigos com derivadas fracionárias não locais aproxima um crescimento quadrático, enquanto a dos operadores locais é linear.

6.4 Influência dos operadores na modelagem da difusão fracionária

Nesta seção, realizamos testes que visam comparar a modelagem obtida pela solução numérica do problema da difusão fracionária com os diferentes operadores.

Experimento 6.4. *Neste experimento, utilizamos os mesmos dados do Experimento 6.1, com o objetivo de comparar os resultados numéricos nos gráficos de superfície.*

Figura 22 - Gráfico de superfície da EDF com a derivada de Riemann-Liouville.

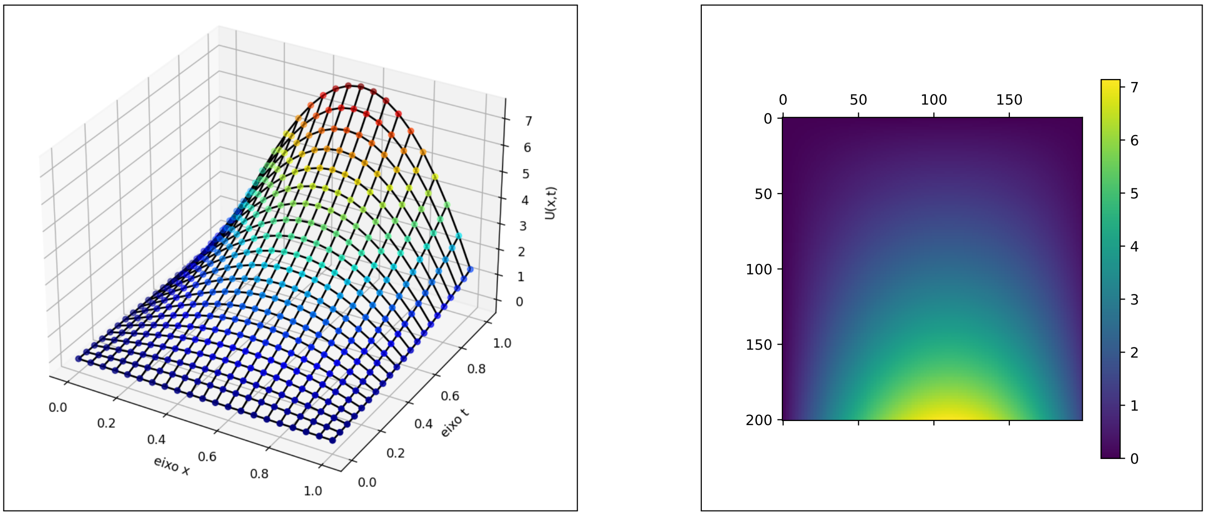


(a) (b)

Legenda: As Figuras (22a) e (22b) exibem a solução da EDF com a derivada de Riemann-Liouville. Os parâmetros utilizados foram: $\Delta x = 1/200$, $\Delta t = 1/200$, $\mathcal{K} = 0,5$, $\alpha = 0,5$ e $\tau = 0,5$.

Fonte: O autor, 2023.

Figura 23 - Gráfico de superfície da EDF com a derivada de Caputo.

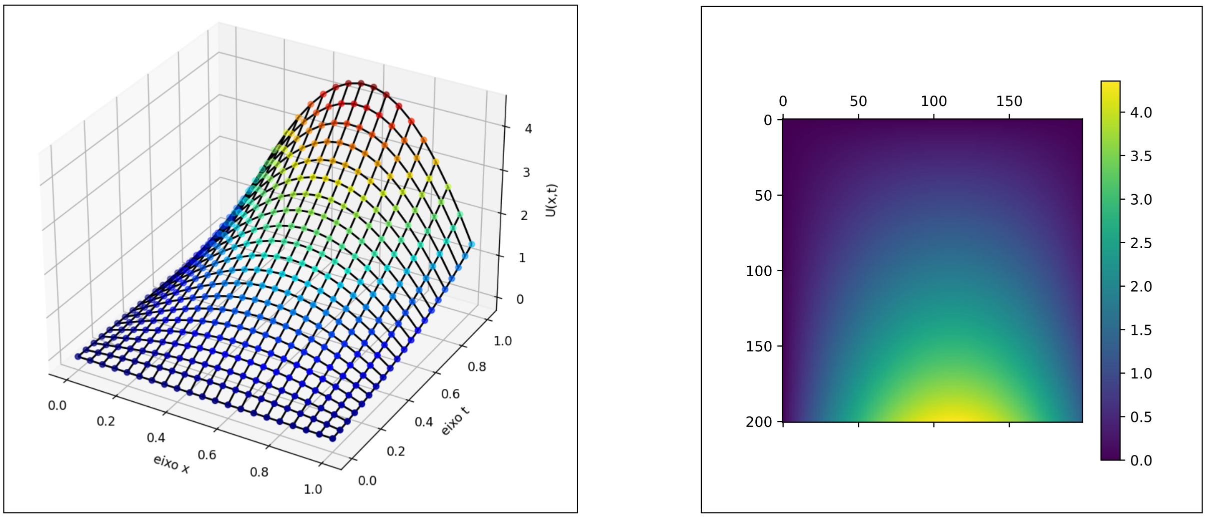


(a) (b)

Legenda: As Figuras (23a) e (23b) exibem a solução da EDF com a derivada de Caputo. Os parâmetros utilizados foram: $\Delta x = 1/200$, $\Delta t = 1/200$, $\mathcal{K} = 0,5$, $\alpha = 0,5$ e $\tau = 0,5$.

Fonte: O autor, 2023.

Figura 24 - Gráfico de superfície da EDF com a derivada de Chen.

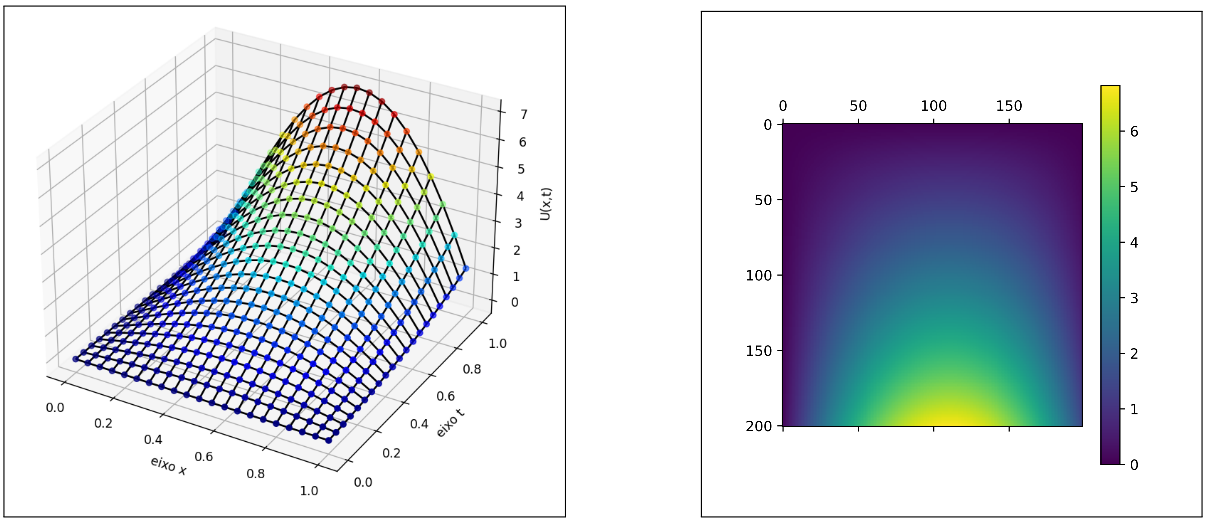


(a) (b)

Legenda: As Figuras (24a) e (24b) exibem a solução da EDF com a derivada de Chen. Os parâmetros utilizados foram: $\Delta x = 1/200$, $\Delta t = 1/200$, $\mathcal{K} = 0,5$, $\alpha = 0,5$ e $\tau = 0,5$.

Fonte: O autor, 2023.

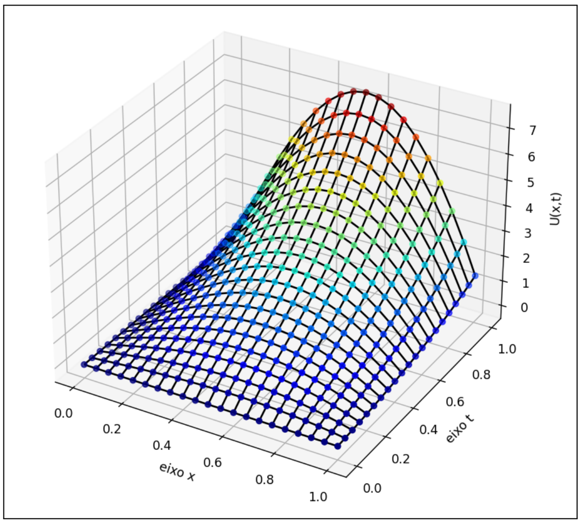
Figura 25 - Gráfico de superfície da EDF com a derivada de Katugampola.



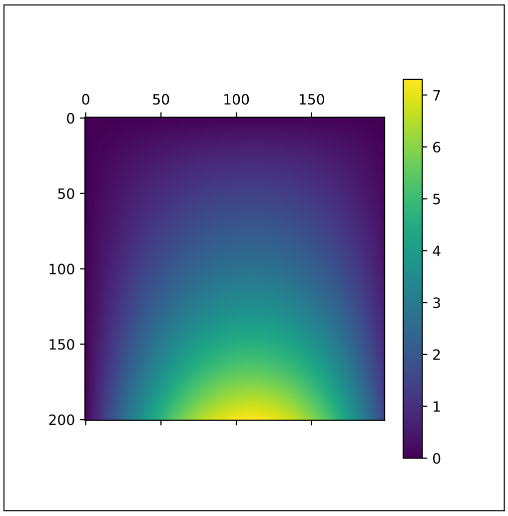
(a) (b)

Legenda: As Figuras (25a) e (25b) exibem a solução da EDF com a derivada de Katugampola. Os parâmetros utilizados foram: $\Delta x = 1/200$, $\Delta t = 1/200$, $\mathcal{K} = 0,5$, $\alpha = 0,5$ e $\tau = 0,5$.

Figura 26 - Gráfico de superfície da EDF com a derivada de Caputo-Fabrizio.



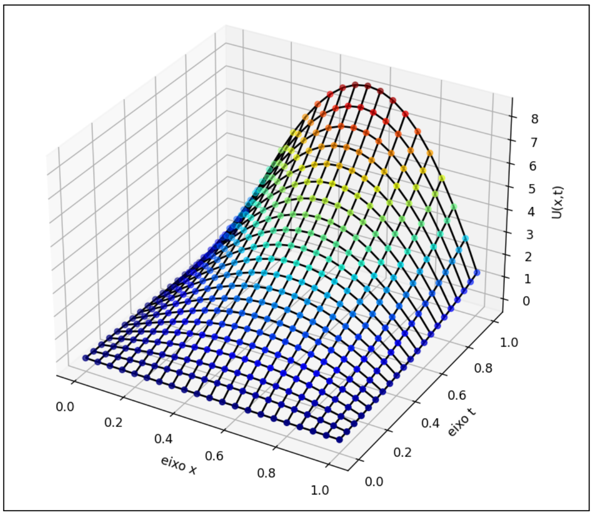
(a)



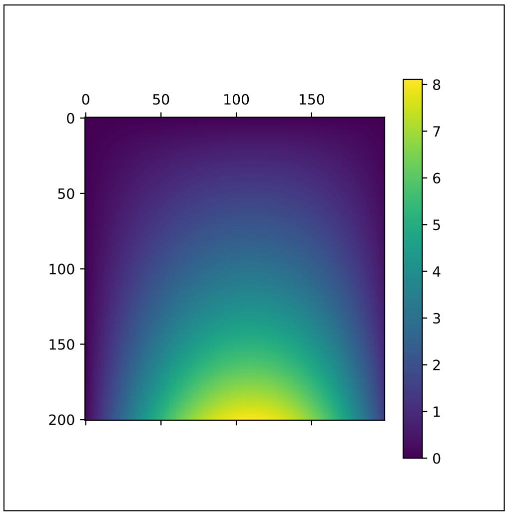
(b)

Legenda: As Figuras (26a) e (26b) exibem a solução da EDF com a derivada de Caputo. Os parâmetros utilizados foram: $\Delta x = 1/200$, $\Delta t = 1/200$, $\mathcal{K} = 0,5$, $\alpha = 0,5$ e $\tau = 0,5$.

Figura 27 - Gráfico de superfície da EDF com a derivada de Atangana-Baleanu.



(a)



(b)

Legenda: As Figuras (27a) e (27b) exibem a solução da EDF com a derivada de Atangana-Baleanu do tipo Caputo. Os parâmetros utilizados foram: $\Delta x = 1/200$, $\Delta t = 1/200$, $\mathcal{K} = 0,5$, $\alpha = 0,5$ e $\tau = 0,5$.

Neste experimento, com os parâmetros utilizados, as soluções numéricas do problema da difusão fracionária com as derivadas de Riemann-Liouville, Caputo e Caputo-Fabrizio estiveram bem próximas, corroborando diversos testes realizados ao longo deste trabalho. Os resultados com os operadores de Katugampola e Atangana-Baleanu mantiveram conformidade com as três derivadas fracionárias mencionadas. Contudo, a derivada de Chen gerou um resultado bem distinto das demais derivadas fracionárias.

CONCLUSÃO

Tendo em vista o número crescente de derivadas fracionárias propostas nos últimos anos, torna-se necessário compreender sua atuação na modelagem de problemas aos quais são aplicadas. Neste sentido, a proposta do trabalho foi investigar a influência de seis derivadas fracionárias quando aplicadas à equação da difusão fracionária. Além disso, visando abordar o problema da escassez de métodos numéricos para o cálculo fracionário, seguimos nesta direção, realizando o estudo via métodos numéricos.

No decorrer deste trabalho, viu-se como construir esquemas numéricos para a equação da difusão fracionária utilizando diferentes derivadas fracionárias, a saber, derivadas fracionárias de Riemann-Liouville, Caputo, Chen, Katugampola, Caputo-Fabrizio e Atangana-Baleanu. Esses operadores foram selecionados a partir da classificação realizada por (TEODORO, 2019) em sua tese de doutorado, que por sua vez, teve como base, o critério proposto em 2015 por Ortigueira e Machado (ORTIGUEIRA; MACHADO, 2015). Esse critério é composto por cinco propriedades.

Ao conduzirmos testes para o esquema explícito, especificamente utilizando o método tipo Euler progressivo com a derivada fracionária segundo Riemann-Liouville, confirmamos os resultados obtidos pela análise numérica que revelou que o método é condicionalmente estável, e a condição para estabilidade está relacionada a $\mathcal{K}$, Δt, Δx, α e τ. No entanto, a necessidade de escolher valores muito pequenos para Δt implica em um aumento significativo no tempo de processamento do código e no uso da memória requerida. Ao tomarmos $\mathcal{K}$ e τ iguais a um, $\Delta x = 1/10$, e $\alpha = 0,9$, seria necessário um Δt da ordem de 10^{-25} para que haja convergência no método. Mesmo para esse valor relativamente pequeno de t, o método se torna inviável. Isso ocorre porque, para avançar um passo na variável t, é necessário revisitar todos os passos já calculados, devido à influência do efeito de memória associado à derivada fracionária.

Visto que a condição de estabilidade torna o método explícito impraticável, optamos por abandonar essa abordagem. A partir desta etapa, consideramos apenas os métodos implícitos, especificamente o método tipo Euler regressivo. Para o operador de Riemann-Liouville, os testes computacionais confirmam os resultados teóricos obtidos pela análise numérica, demonstrando que o método é incondicionalmente estável e sua taxa de convergência é compatível com a do método de Euler regressivo clássico quando aplicado à equação de ordem inteira.

Levando em conta os resultados alcançados pelo método tipo Euler regressivo com Riemann-Liouville, propomos esquemas semelhantes para as outras derivadas fracionárias. Os experimentos numéricos realizados para o método tipo Euler regressivo com todas as outras derivadas fracionárias indicam que esses esquemas são incondicionalmente estáveis. Esses métodos produziram boas aproximações, e suas taxas de convergência estiveram em concordância com a precisão local das aproximações das derivadas fracionárias. Os ajustes que o método incondicionalmente estável permite no dimensionamento de sua malha são de suma importância para sua viabilidade, visto que o número elevado de passos na variável a qual a derivada fracionária está sendo aplicada pode torná-lo inviável.

Ao analisarmos o efeito da ordem α, os resultados numéricos da equação da difusão com os diferentes operadores foram compatíveis para os α's investigados. Contudo, quando acrescentamos na análise as derivadas inteiras de ordem zero e de ordem um, as derivadas fracionárias locais, por não satisfazerem a segunda condição do critério de Ortigueira e Machado, apresentam uma inconsistência com os resultados, o que não ocorre com os demais operadores, por satisfazerem tal critério. As mudanças nas soluções numéricas para os diferentes valores do parâmetro de correção dimensional τ foram significativas para alguns valores de α e não tiveram um impacto significativo para outros. Além disso, não considerar τ na equação da difusão equivale a fazermos τ igual a um.

A análise da complexidade de tempo dos códigos mostrou que os códigos com as derivadas locais, Chem e Katugampola, tiveram um aumento linear do tempo de execução com o tamanho da entrada, o que corresponde a uma complexidade linear, $O(n)$. Já os códigos com as derivadas de Riemann-Liouville, Caputo, Caputo-Fabrizio e Atangana Baleanu, que são operadores não locais, tiveram uma complexidade quadrática, $O(n^2)$. Ao realizar o estudo sobre a influência dos operadores na modelagem da difusão fracionária, os resultados com a derivada fracionária de Riemann-Liouville, Caputo e Caputo-Fabrizio ficaram muito próximos, consolidando diversos outros experimentos ao longo desta tese. Os resultados obtidos com os operadores de Katugampola e Atangana-Baleanu foram consistentes com as três derivadas fracionárias mencionadas. No entanto, a derivada de Chen gerou um resultado que divergiu dos resultados das demais derivadas fracionárias.

Uma continuação natural do presente trabalho é realizar a análise teórica da estabilidade para os cinco métodos, visando buscar aproximações para as derivadas fracionárias que possibilitem melhores desempenhos e, particularmente, a construção de um método semelhante ao de Crank-Nicolson. A inserção do parâmetro de correção de dimensão τ na modelagem gera novas possibilidades, mas também nos impõe lidar com um novo desafio, qual seja, um problema de otimização a ser ainda formulado (e resolvido): como escolher o melhor valor para τ? Este certamente representa uma importante direção na pesquisa, pois um suporte teórico também deverá ser considerado, conforme feito no estudo da estabilidade.

Uma outra direção na qual se pretende avançar é a aplicação da equação da difusão fracionária ao problema da difusão anômala, consistindo na verificação natural de qual operador seria mais adequado para essa simulação.

REFERÊNCIAS

ABRO, K. A.; MEMON, A. A.; UQAILI, M. A. A comparative mathematical analysis of RL and RC electrical circuits via Atangana-Baleanu and Caputo-Fabrizio fractional derivatives. *The European Physical Journal Plus*, Springer, v. 133, n. 3, p. 1–9, 2018.

ANDERS, W. Über den Fundamentalsatz in der Theorie der Funktionen Ea(z). *Acta Math*, v. 29, n. 1, p. 191–201, 1905.

ATANGANA, A.; BALEANU, D. New fractional derivatives with nonlocal and non-singular kernel: theory and application to heat transfer model. *arXiv preprint arXiv:1602.03408*, 2016.

BALEANU, D. et al. *Fractional Calculus: Models and Numerical Methods*. 2. ed. Singapore: World Scientific, 2012. 448 p.

BERRYMAN, J. G. Evolution of a stable profile for a class of nonlinear diffusion equations with fixed boundaries. *Journal of Mathematical Physics*, American Institute of Physics, v. 18, n. 11, p. 2108–2115, 1977.

BURDEN, R. L. *Numerical Analysis*. [S.l.]: Brooks/Cole Cengage Learning, 2011.

CAMARGO, R. F.; OLIVEIRA, E. C. *Cálculo Fracionário*. 1. ed. São Paulo: Livraria da Física, 2015. 183 p.

CANNON, J. R. *The One-Dimensional Heat Equation*. 1. ed. Cambridge: Cambridge University Press, 1984. 512 p.

CAPUTO, M. *Elasticità e Dissipazione*. [S.l.]: Zanichelli, 1969.

CAPUTO, M.; FABRIZIO, M. A new definition of fractional derivative without singular kernel. *Progr. Fract. Differ. Appl*, v. 1, n. 2, p. 1–13, 2015.

CAPUTO, M.; FABRIZIO, M. Applications of new time and spatial fractional derivatives with exponential kernels. *Progr. Fract. Differ. Appl*, v. 2, n. 2, p. 1–11, 2016.

CARVER, H. C.; O'TOOLE, A.; RAIFORD, T. *The Annals of Mathematical Statistics*. [S.l.]: Edwards Bros., 1930.

CHARNEY, J. G.; FJÖRTOFT, R.; NEUMANN, J. v. Numerical integration of the barotropic vorticity equation. *Tellus*, Taylor & Francis, v. 2, n. 4, p. 237–254, 1950.

CHEN, C.-M. et al. A Fourier method for the fractional diffusion equation describing sub-diffusion. *Journal of Computational Physics*, Elsevier, v. 227, n. 2, p. 886–897, 2007.

CHEN, W. Fractional and fractal derivatives modeling of turbulence. *arXiv preprint nlin/0511066*, 2005.

CHEN, W. Time–space fabric underlying anomalous diffusion. *Chaos, Solitons & Fractals*, Elsevier, v. 28, n. 4, p. 923–929, 2006.

CHEN, W. et al. Anomalous diffusion modeling by fractal and fractional derivatives. *Computers & Mathematics with Applications*, Elsevier, v. 59, n. 5, p. 1754–1758, 2010.

COURANT, R. *Cálculo Diferencial e Integral II*. 1. ed. Porto Alegre: Editora Globo, 1963.

CUMINATO, J. A.; MENEGUETTE, M. *Discretização de Equações Diferenciais Parciais: Técnicas de Diferenças Finitas*. [S.l.]: Sociedade Brasileira de Matemática, 2013.

EDDY, R. *Stability in the Numerical Solution of Initial Value Problems in Partial Differential Equations:(Project NR 044-049)*. [S.l.]: US Naval Ordinance Laboratory, 1958.

FARIA, C. O.; MOURA, C. A. de; NEGREIROS, J. P. de. Uma abordagem numérica sobre o efeito dos operadores fracionários na derivação temporal para a equação de difusão fracionária. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 10, n. 1, p. 2–7, 2023.

FARIA, C. O. de; MOURA, C. A. de; NEGREIROS, J. P. de. Algoritmos numéricos para a equação de difusão linear de ordem fracionária. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 8, n. 1, 2021.

FOMIN, S.; CHUGUNOV, V.; HASHIDA, T. Application of fractional differential equations for modeling the anomalous diffusion of contaminant from fracture into porous rock matrix with bordering alteration zone. *Transport in Porous Media*, Springer, v. 81, p. 187–205, 2010.

FOURIER, J. B. J. Propagation de la chaleur dans un solide rectangulaire infini. In: ______. *Théorie Analytique de la Chaleur*. [S.l.]: Cambridge University Press, 2009. (Cambridge Library Collection – Mathematics), p. 159–265.

GAO, G.-h.; SUN, Z.-z. A compact finite difference scheme for the fractional sub-diffusion equations. *Journal of Computational Physics*, Elsevier, v. 230, n. 3, p. 586–595, 2011.

GÓMEZ-AGUILAR, J. et al. Modeling diffusive transport with a fractional derivative without singular kernel. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 447, p. 467–481, 2016.

GÓMEZ-AGUILAR, J. F.; RAZO-HERNÁNDEZ, R.; GRANADOS-LIEBERMAN, D. A physical interpretation of fractional calculus in observable terms: analysis of the fractional time constant and the transitory response. *Revista Mexicana de Física*, Scielo, v. 60, n. 1, p. 32–38, 2014.

GÓMEZ-AGUILAR, J. F. et al. Fractional mechanical oscillators. *Revista Mexicana de Física*, Scielo, v. 58, n. 4, p. 348–352, 2012.

GRÜNWALD, A. Über “begrenzte” derivation und deren anwendung. *Z. Angew Math. und Phys*, v. 12, p. 441–480, 1867.

HARRIS, C.; SELLS, R. L.; GUTH, E. Random walk theory of one-dimensional gases. *The Journal of Chemical Physics*, American Institute of Physics, v. 21, n. 9, p. 1617–1618, 1953.

HE, J.-H. A tutorial review on fractal spacetime and fractional calculus. *International Journal of Theoretical Physics*, Springer, v. 53, p. 3698–3718, 2014.

KATUGAMPOLA, U. N. A new fractional derivative with classical properties. *arXiv preprint arXiv:1410.6535*, 2014.

LANGLANDS, T. A. M.; HENRY, B. I. The accuracy and stability of an implicit solution method for the fractional diffusion equation. *Journal of Computational Physics*, Elsevier, v. 205, n. 2, p. 719–736, 2005.

LAX, P. D.; RICHTMYER, R. D. Survey of the stability of linear finite difference equations. *Communications on Pure and Applied Mathematics*, Wiley Online Library, v. 9, n. 2, p. 267–293, 1956.

LETNIKOV, A. V. Theory of differentiation with an arbitrary index. *Matem. Sbornik*, v. 3, p. 1–66, 1868.

LEVEQUE, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations*: Steady-state and time-dependent problems. Philadelphia: SIAM, 2007. 341 p.

LIU, F. et al. Analysis of a discrete non-Markovian random walk approximation for the time fractional diffusion equation. *Anziam Journal*, v. 46, p. C488–C504, 2004.

MAINARDI, F. *Fractional Calculus and Waves in Linear Viscoelasticity: an Introduction to Mathematical Models*. [S.l.]: World Scientific, 2022.

MEERSCHAERT, M. M.; TADJERAN, C. Finite difference approximations for fractional advection–dispersion flow equations. *Journal of Computational and Applied Mathematics*, Elsevier, v. 172, n. 1, p. 65–77, 2004.

METZLER, R.; KLAFTER, J. The random walk's guide to anomalous diffusion: a fractional dynamics approach. *Physics reports*, Elsevier, v. 339, n. 1, p. 1–77, 2000.

METZLER, R.; KLAFTER, J. The restaurant at the end of the random walk: recent developments in the description of anomalous transport by fractional dynamics. *Journal of Physics A: Mathematical and General*, IOP Publishing, v. 37, n. 31, p. R161, 2004.

MOSCHEN, I. D. C. *Sobre as funções Mittag-Leffler e o modelo fracionário de materiais viscoelásticos*. 2006. 130 f. Tese (Doutorado em Engenharia Mecânica) — Programa de Pós-Graduação em Mecânica. Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2006.

MOURA, C. A. de; KUBRUSLY, C. S. *The Courant–Friedrichs–Lewy (CFL) Condition*. [S.l.]: Springer, 2013. v. 10.

NARASIMHAN, T. N. Fourier's heat conduction equation: History, influence, and connections. *Reviews of Geophysics*, Wiley Online Library, v. 37, n. 1, p. 151–172, 1999.

NEGREIROS, J. P. de. *Um problema inverso na modelagem da difusão do calor*. Dissertação (Mestrado) — Universidade do Estado do Rio de Janeiro, 2010.

NEGREIROS, J. P. de; FARIA, C. O.; MOURA, C. A. de. Um estudo de estabilidade por Fourier para a equação de difusão fracionária com correção dimensional. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 9, n. 1, 2022.

OLDHAM, K.; SPANIER, J. *The Fractional Calculus Theory and Applications of Differentiation and Integration to Arbitrary Order*. 1. ed. New York: Dover Publications, 1974. 234 p.

ORTIGUEIRA, M. D.; MACHADO, J. T. What is a fractional derivative? *Journal of Computational Physics*, Elsevier, v. 293, p. 4–13, 2015.

PETER, O. J. et al. Analysis and dynamics of fractional order mathematical model of covid-19 in nigeria using atangana-baleanu operator. *Comput. Mater. Con.*, v. 66, p. 1823–1848, 2021.

PICK, A. Über Diffusion. *Poggendorff Ann. Physic.(Ser. 2)*, v. 94, p. 59–86, 1855.

PODLUBNY, I. *Fractional Differential Equations*: An introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications. 1. ed. San Diego: Academic Press, 1998. 340 p.

PODLUBNY, I. Geometric and physical interpretation of fractional integration and fractional differentiation. Fractional Calculus and Applied Analysis, v. 5, n. 4, p. 367–386, 2002.

PRABHAKAR, T. R. et al. A singular integral equation with a generalized Mittag-Leffler function in the kernel. *Yokohama Math. J*, v. 19, n. 1, p. 7–15, 1971.

QUARTERONI, A.; SACCO, R.; SALERI, F. *Numerical Mathematics*. [S.l.]: Springer Science & Business Media, 2010. v. 37.

QURESHI, S.; RANGAIG, N. A.; BALEANU, D. New numerical aspects of Caputo-Fabrizio fractional derivative operator. *Mathematics*, Multidisciplinary Digital Publishing Institute, v. 7, n. 4, p. 374, 2019.

RICE, S. A.; FRISCH, H. L. Dynamical theory of diffusion in crystals. IV. Some aspects of the introduction of irreversibility. *The Journal of Chemical Physics*, American Institute of Physics, v. 32, n. 4, p. 1026–1034, 1960.

RICHARDSON, L. F. Atmospheric diffusion shown on a distance-neighbour graph. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, The Royal Society London, v. 110, n. 756, p. 709–737, 1926.

RICHTMYER, R. D.; MORTON, K. *Different Methods for Initial Value Problems*. [S.l.]: Interscience, 1967.

ROSS, B. A brief history and exposition of the fundamental theory of fractional calculus. In: *Fractional calculus and its applications*. [S.l.]: Springer, 1975. p. 1–36.

RUDIN, W. et al. *Principles of Mathematical Analysis*. [S.l.]: McGraw-hill New York, 1976. v. 3.

SCHEIDEGGER, A. The random-walk model with autocorrelation of flow through porous media. *Canadian Journal of Physics*, NRC Research Press Ottawa, Canada, v. 36, n. 6, p. 649–658, 1958.

SCHER, H.; MONTROLL, E. W. Anomalous transit-time dispersion in amorphous solids. *Phys. Rev. B*, American Physical Society, v. 12, p. 2455–2477, Sep 1975. Disponível em: ⟨https://link.aps.org/doi/10.1103/PhysRevB.12.2455⟩.

SHEIKH, N. A. et al. Comparison and analysis of the Atangana–Baleanu and Caputo–Fabrizio fractional derivatives for generalized casson fluid model with heat generation and chemical reaction. *Results in Physics*, Elsevier, v. 7, p. 789–800, 2017.

SO, F.; LIU, K. L. A study of the subdiffusive fractional Fokker–Planck equation of bi-stable systems. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 331, n. 3-4, p. 378–390, 2004.

SONIN, N. Y. On differentiation with arbitrary index. *Moscow Matem. Sbornik*, v. 6, n. 1, p. 1–38, 1869.

SPOHN, H. Surface dynamics below the roughening transition. *Journal de Physique I*, EDP Sciences, v. 3, n. 1, p. 69–81, 1993.

STEPHENSON, J. Some non-linear diffusion equations and fractal diffusion. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 222, n. 1-4, p. 234–247, 1995.

STRIKWERDA, J. C. *Finite Difference Schemes and Partial Differential Equations.* [S.l.]: SIAM, 2004.

TEODORO, G.; OLIVEIRA, D.; OLIVEIRA, E. Sobre derivadas fracionárias. *Revista Brasileira de Ensino de Física*, SciELO Brasil, v. 40, 2017.

TEODORO, G. S. *Derivadas fracionárias: tipos e critérios de validade.* 2019. 182 f. Tese (Doutorado em Matemática Aplicada) — Instituto de Matemática, Estatística e Computação Científica, Universidade Estadual de Campinas, São Paulo, 2019.

VIEIRA, D. S. *Equações de difusão e o cálculo fracionário.* Dissertação (Mestrado) — Universidade Estadual de Maringá, 2015.

YADAV, S.; PANDEY, R. K.; SHUKLA, A. K. Numerical approximations of Atangana–Baleanu Caputo derivative and its application. *Chaos, Solitons & Fractals*, Elsevier, v. 118, p. 58–64, 2019.

YANG, X.-J.; SRIVASTAVA, H. M.; MACHADO, J. A new fractional derivative without singular kernel: application to the modelling of the steady heat flow. *arXiv preprint arXiv:1601.01623*, 2015.

YUSTE, S. B.; ACEDO, L. An explicit finite difference method and a new von Neumann-type stability analysis for fractional diffusion equations. *SIAM Journal on Numerical Analysis*, SIAM, v. 42, n. 5, p. 1862–1874, 2005.

ZHOKH, A.; STRIZHAK, P. Non-Fickian transport in porous media: Always temporally anomalous? *Transport in Porous Media*, Springer, v. 124, p. 309–323, 2018.

APÊNDICE A – Códigos em Python – Difusão Usual

Aqui, apresentaremos os códigos em linguagem Python correspondentes aos métodos de diferenças progressivas, regressivas e de Crank-Nicolson quando aplicados à equação da difusão usual, que foram discutidos na Seção 2.5.

A.1 Diferenças Progressivas

```
# DIFUSAO CLASSICA - METODO EULER PROGRESSIVO

import numpy as np
from sympy import gamma, exp, sin
import time


#  FUNCOES DEFINIDAS PELO PROGRAMADOR

def CondicaoInicial(limite_x, npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = sin(np.pi * (i*delta_x))
    return condicao_inicial


def CondicaoFronteira(limite_x, limite_t, npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = 0
        front_na_direita[n] = 0
    return front_na_esquerda, front_na_direita


def TermoFonte(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
```

```
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = 0
    return termo


def MatrizM(limite_x, npassos_x, limite_t, npassos_t, K):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    v = K * delta_t / delta_x ** 2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = 1 - 2 * v
        matriz[i][i+1] = v
        matriz[i+1][i] = v
    matriz[npassos_x-2][npassos_x-2] = 1 - 2 * v
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
   limite_t, npassos_t):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
       npassos_t)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max


def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
```

```
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = exp(-np.pi ** 2 * (n*delta_t))
              * sin(np.pi * (i*delta_x))
    return sanalitica


# ENTRADA DOS DADOS

limite_x = float(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = float(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Difusividade termica K '))


# PROGRAMA PRINCIPAL

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
   npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t)

matriz_M = MatrizM(limite_x, npassos_x, limite_t, npassos_t, K)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
v = K * delta_t / delta_x ** 2

n = 0
while n < npassos_t:
    produto = np.dot(matriz_M, U[n])
```

```
        produto[0] = produto[0] + v * fesquerda[n]
        produto[-1] = produto[-1] + v * fdireita[n]
        U[n+1] = produto + delta_t * fonte[n]
        n = n + 1

fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
print()


# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t)

fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60

print(f'O pos-processamento foi executado em {segundospos:.1f}
    segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

A.2 Diferenças Regressivas

```
# DIFUSAO CLASSICA - DIFERENCAS REGRESSIVAS

import numpy as np
from sympy import gamma, exp, sin
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR

def CondicaoInicial(limite_x, npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = sin(np.pi * (i*delta_x))
    return condicao_inicial


def CondicaoFronteira(limite_x, limite_t, npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = 0
        front_na_direita[n] = 0
    return front_na_esquerda, front_na_direita


def TermoFonte(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = 0
    return termo
```

```
def MatrizN(limite_x, npassos_x, limite_t, npassos_t, K):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    v = K * delta_t / delta_x ** 2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = 1 + 2 * v
        matriz[i][i+1] = - v
        matriz[i+1][i] = - v
    matriz[npassos_x-2][npassos_x-2] = 1 + 2 * v
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
   limite_t, npassos_t):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
       npassos_t)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max


def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = exp(-np.pi ** 2 * (n*delta_t))
               * sin(np.pi * (i*delta_x))
    return sanalitica
```

```
# ENTRADA DOS DADOS

limite_x = float(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = float(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Difusividade termica K '))


# PROGRAMA PRINCIPAL

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
    npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t)

matriz_N = MatrizN(limite_x, npassos_x, limite_t, npassos_t, K)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
v = K * delta_t / delta_x ** 2

n = 1
while n <= npassos_t:
    U[n-1][0] = U[n-1][0] - v * fesquerda[n]
    U[n-1][-1] = U[n-1][-1] - v * fdireita[n]
    U[n] = np.linalg.solve(matriz_N, U[n-1] + delta_t * fonte[n
        ])
    n = n + 1

fim = time.time()
segundos = fim - inicio
```

```
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
print()


# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t)

fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60

print(f'O pos-processamento foi executado em {segundospos:.1f}
    segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

A.3 Crank-Nicolson

```
# DIFUSAO CLASSICA - CRANK-NICOLSON

import numpy as np
from sympy import gamma, exp, sin
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR

def CondicaoInicial(limite_x, npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = sin(np.pi * (i*delta_x))
    return condicao_inicial


def CondicaoFronteira(limite_x, limite_t, npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = 0
        front_na_direita[n] = 0
    return front_na_esquerda, front_na_direita


def TermoFonte(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = 0
    return termo
```

```
def MatrizM(limite_x, npassos_x, limite_t, npassos_t, K):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    v = K * delta_t / delta_x ** 2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = 1 - v
        matriz[i][i+1] = 0.5 * v
        matriz[i+1][i] = 0.5 * v
    matriz[npassos_x-2][npassos_x-2] = 1 - v
    return matriz


def MatrizN(limite_x, npassos_x, limite_t, npassos_t, K):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    v = K * delta_t / delta_x ** 2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = 1 + v
        matriz[i][i+1] = - 0.5 * v
        matriz[i+1][i] = - 0.5 * v
    matriz[npassos_x-2][npassos_x-2] = 1 + v
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
  limite_t, npassos_t):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
      npassos_t)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max
```

```
def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = exp(-np.pi ** 2 * (n*delta_t))
              * sin(np.pi * (i*delta_x))
    return sanalitica


# ENTRADA DOS DADOS

limite_x = float(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = float(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Difusividade termica K '))


# PROGRAMA PRINCIPAL

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
    npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t)

matriz_M = MatrizM(limite_x, npassos_x, limite_t, npassos_t, K)
matriz_N = MatrizN(limite_x, npassos_x, limite_t, npassos_t, K)

delta_x = limite_x / npassos_x
```

```
delta_t = limite_t / npassos_t
v = K * delta_t / delta_x ** 2

n = 0
while n < npassos_t:
    produto = np.dot(matriz_M, U[n])
    produto[0] = produto[0] + v * (fesquerda[n] - fesquerda[n
        +1])
    produto[-1] = produto[-1] + v * (fdireita[n] - fdireita[n
        +1])
    U[n+1] = np.linalg.solve(matriz_N, produto + delta_t * (
        fonte[n] + fonte[n+1]) / 2)
    n = n + 1

print(U[npassos_t])
W = SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t)
print(W[npassos_t])

fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
print()


# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t)

fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60
```

```python
print(f'O pos-processamento foi executado em {segundospos:.1f} segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

APÊNDICE B – Códigos em Python – Difusão Fracionária

Apresentamos os códigos em linguagem Python para os métodos numéricos quando aplicados à equação da difusão fracionária. Esses códigos são resultados desta tese, mais precisamente os algoritmos dos Experimentos 3.1, 3.2, 3.4, 4.1, 4.3, 5.1 e 5.3.

B.1 Método tipo Euler progressivo – Derivada de Riemann-Liouville

```
import numpy as np
from sympy import gamma, exp
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR

def CondicaoInicial(limite_x, npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = 0
    return condicao_inicial


def CondicaoFronteira(limite_x, limite_t, npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = (n * delta_t) ** 1.5 * exp(np.pi
            * (0 - alpha))
        front_na_direita[n] = (n * delta_t) ** 1.5 * exp(np.pi *
            (limite_x - alpha))
    return front_na_esquerda, front_na_direita


def TermoFonte(limite_x, npassos_x, limite_t, npassos_t):
```

```
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = (1.5 * (n * delta_t) ** 0.5 - (np.pi
                ** 2) * (gamma(2.5) *
             (n * delta_t) ** (1.5 - alpha)) / gamma(2.5 - alpha
                )) * exp(np.pi * (i * delta_x - alpha))
    return termo


def CoeficienteOmega(npassos_t, alpha):
    coeficiente = np.zeros(npassos_t+1)
    for k in range(0, npassos_t+1):
        coeficiente[k] = (-1) ** k * (gamma(alpha + 1)/(gamma(k
            + 1)*gamma(alpha - k + 1)))
    return coeficiente


def MatrizA(limite_x, npassos_x, limite_t, npassos_t, K, tau,
   alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    r = (tau ** alpha) * K * (delta_t**(1 - alpha)) / delta_x **
        2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = -2 * r
        matriz[i][i+1] = r
        matriz[i+1][i] = r
    matriz[npassos_x-2][npassos_x-2] = -2 * r
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
   limite_t, npassos_t, alpha):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
       npassos_t, alpha)
```

```
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max


def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t,
    alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = (n * delta_t) ** 1.5 * exp(np.
                pi * (i * delta_x - alpha))
    return sanalitica


# ENTRADA DOS DADOS

limite_x = int(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = int(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Coeficiente de difusao '))
alpha = float(input('Ordem da derivada fracionaria '))
tau = float(input('Parametro de correcao dimensional '))


# PROGRAMA PRINCIPAL

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))
```

```
U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
    npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t)

omega = CoeficienteOmega(npassos_t, alpha)

matriz_A = MatrizA(limite_x, npassos_x, limite_t, npassos_t, K,
    tau, alpha)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
r = (tau ** alpha) * K * (delta_t ** (1 - alpha)) / delta_x ** 2

n = 0
while n < npassos_t:
    auxiliar = np.zeros(npassos_x-1)
    for k in range(0, n+1):
        produto = np.dot(matriz_A, U[n-k])
        produto[0] = produto[0] + r * fesquerda[n-k]
        produto[-1] = produto[-1] + r * fdireita[n-k]
        auxiliar = auxiliar + omega[k] * produto

    U[n+1] = U[n] + auxiliar + delta_t * fonte[n]
    n = n + 1

fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
print()
```

```
# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t, alpha)

fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60

print(f'O pos-processamento foi executado em {segundospos:.1f}
    segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

B.2 Método tipo Euler regressivo – Derivada de Riemann-Liouville

```
import numpy as np
from sympy import gamma, exp
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR

def CondicaoInicial(limite_x, npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = 0
    return condicao_inicial


def CondicaoFronteira(limite_x, limite_t, npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = (n * delta_t) ** 1.5 * exp(np.pi
            * (0 - alpha))
        front_na_direita[n] = (n * delta_t) ** 1.5 * exp(np.pi *
            (limite_x - alpha))
    return front_na_esquerda, front_na_direita


def TermoFonte(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = (1.5 * (n * delta_t) ** 0.5 - (np.pi
                ** 2) * (gamma(2.5) *
             (n * delta_t) ** (1.5 - alpha)) / gamma(2.5 - alpha
                )) * exp(np.pi * (i * delta_x - alpha))
```

```
    return termo


def CoeficienteOmega(npassos_t, alpha):
    coeficiente = np.zeros(npassos_t+1)
    for k in range(0, npassos_t+1):
        coeficiente[k] = (-1) ** k * (gamma(alpha + 1)/(gamma(k + 1)*gamma(alpha - k + 1)))
    return coeficiente


def MatrizA(limite_x, npassos_x, limite_t, npassos_t, tau, alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    r = (tau ** alpha) * K * (delta_t**(1 - alpha)) / delta_x ** 2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = -2 * r
        matriz[i][i+1] = r
        matriz[i+1][i] = r
    matriz[npassos_x-2][npassos_x-2] = -2 * r
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x, limite_t, npassos_t, alpha):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t, alpha)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max
```

```
def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t,
    alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = (n * delta_t) ** 1.5 * exp(np.
                pi * (i * delta_x - alpha))
    return sanalitica


# ENTRADA DOS DADOS

limite_x = int(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = int(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Coeficiente de difusao '))
alpha = float(input('Ordem da derivada fracionaria '))
tau = float(input('Parametro de correcao dimensional '))


# PROGRAMA PRINCIPAlimite_x

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
    npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t)

omega = CoeficienteOmega(npassos_t, alpha)
```

```
matriz_A = MatrizA(limite_x, npassos_x, limite_t, npassos_t, tau
    , alpha)

matriz_I = np.identity(npassos_x-1)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
r = (tau ** alpha) * K * (delta_t**(1-alpha)) / delta_x ** 2

n = 1
while n <= npassos_t:
    auxiliar = np.zeros(npassos_x-1)
    for k in range(1, n+1):
        produto = np.dot(matriz_A, U[n-k])
        produto[0] = produto[0] + r * fesquerda[n-k]
        produto[-1] = produto[-1] + r * fdireita[n-k]
        auxiliar = auxiliar + omega[k] * produto

    auxiliar[0] = auxiliar[0] + r * fesquerda[n]
    auxiliar[-1] = auxiliar[-1] + r * fdireita[n]
    U[n] = np.linalg.solve(matriz_I - omega[0] * matriz_A, U[n
        -1] + auxiliar + delta_t * fonte[n])
    n = n + 1

fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
print()


#  POS-PROCESSAMENTO

iniciopos = time.time()
```

```
desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t, alpha)

fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60

print(f'O pos-processamento foi executado em {segundospos:.1f}
    segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

B.3 Método tipo Euler regressivo – Derivada de Caputo

```
import numpy as np
from sympy import gamma, exp
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR

def CondicaoInicial(limite_x, npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = 0
    return condicao_inicial


def CondicaoFronteira(limite_x, limite_t, npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = (n * delta_t) ** 1.5 * exp(np.pi
            * (0 - alpha))
        front_na_direita[n] = (n * delta_t) ** 1.5 * exp(np.pi *
             (limite_x - alpha))
    return front_na_esquerda, front_na_direita


def TermoFonte(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = (1.5 * (n * delta_t) ** 0.5 - (np.pi
                ** 2) * (gamma(2.5) *
             (n * delta_t) ** (1.5 - alpha)) / gamma(2.5 - alpha
                )) * exp(np.pi * (i * delta_x - alpha))
```

```
    return termo


def CoeficienteB(npassos_t, alpha):
    coeficiente = np.zeros(npassos_t+1)
    for k in range(0, npassos_t+1):
        coeficiente[k] = ((k + 1) ** (1 - alpha) - k ** (1 -
            alpha)) / gamma(2 - alpha)
    return coeficiente


def MatrizA(limite_x, npassos_x, limite_t, npassos_t, K, tau,
   alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    r = (tau ** alpha) * K * (delta_t**(1 - alpha)) / delta_x **
        2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = -2 * r
        matriz[i][i+1] = r
        matriz[i+1][i] = r
    matriz[npassos_x-2][npassos_x-2] = -2 * r
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
   limite_t, npassos_t, alpha):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
       npassos_t, alpha)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max
```

```
def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t,
    alpha):
     delta_x = limite_x / npassos_x
     delta_t = limite_t / npassos_t
     sanalitica = np.zeros((npassos_t+1, npassos_x-1))
     for n in range(0, npassos_t+1):
         for i in range(1, npassos_x):
             sanalitica[n][i-1] = (n * delta_t) ** 1.5 * exp(np.
                pi * (i * delta_x - alpha))
     return sanalitica


# ENTRADA DOS DADOS

limite_x = int(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = int(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Coeficiente de difusao '))
alpha = float(input('Ordem da derivada fracionaria '))
tau = float(input('Parametro de correcao dimensional '))


# PROGRAMA PRINCIPAlimite_x

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
    npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t)

coeficiente_B = CoeficienteB(npassos_t, alpha)
```

```
matriz_A = MatrizA(limite_x, npassos_x, limite_t, npassos_t, K,
    tau, alpha)

matriz_I = np.identity(npassos_x-1)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
r = (tau ** alpha) * K * (delta_t**(1-alpha)) / delta_x ** 2

n = 1
while n <= npassos_t:
    auxiliar = np.zeros(npassos_x-1)
    for k in range(0, n-1):
        produto = np.dot(matriz_A, U[k+1] - U[k])
        produto[0] = produto[0] + r * (fesquerda[k+1] -
            fesquerda[k])
        produto[-1] = produto[-1] + r * (fdireita[k+1] -
            fdireita[k])
        auxiliar = auxiliar + coeficiente_B[n-k-1] * produto

    vetor = np.dot(matriz_I - coeficiente_B[0] * matriz_A, U[n
        -1])
    vetor[0] = vetor[0] + coeficiente_B[0] * r * (fesquerda[n] -
        fesquerda[n-1])
    vetor[-1] = vetor[-1] + coeficiente_B[0] * r * (fdireita[n]
        - fdireita[n-1])
    U[n] = np.linalg.solve(matriz_I - coeficiente_B[0] *
        matriz_A, vetor + auxiliar + delta_t * fonte[n])
    n = n + 1


fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
```

```
print()


# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t, alpha)

fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60

print(f'O pos-processamento foi executado em {segundospos:.1f}
    segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

B.4 Método tipo Euler regressivo – Derivada de Chen

```
import numpy as np
from sympy import gamma, sin , exp
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR


def CondicaoInicial(limite_x , npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = 0
    return condicao_inicial


def CondicaoFronteira(limite_x , limite_t , npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = 0
        front_na_direita[n] = 0
    return front_na_esquerda , front_na_direita


def TermoFonte(limite_x , npassos_x , limite_t , npassos_t , alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(1, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = (1.5 + alpha) * (n*delta_t)**(0.5 +
                alpha) * (1 + np.pi**2 / (alpha * (n*delta_t)**(
                alpha-1))) * sin(np.pi * (i*delta_x))
    return termo
```

```
def CoeficienteP(npassos_t, alpha):
    coeficiente = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        coeficiente[n] = (n**(1-alpha)) / alpha
    return coeficiente


def MatrizA(limite_x, npassos_x, limite_t, npassos_t, K, tau,
   alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    r = (tau ** alpha) * K * (delta_t**(1 - alpha)) / delta_x **
        2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = -2 * r
        matriz[i][i+1] = r
        matriz[i+1][i] = r
    matriz[npassos_x-2][npassos_x-2] = -2 * r
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
   limite_t, npassos_t, alpha):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
       npassos_t, alpha)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max


def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t,
```

```
    alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = (n * delta_t) ** (1.5 + alpha)
                * sin(np.pi * (i * delta_x))
    return sanalitica


# ENTRADA DOS DADOS

limite_x = int(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = int(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Coeficiente de difusao '))
alpha = float(input('Ordem da derivada fracionaria '))
tau = float(input('Parametro de correcao dimensional '))


# PROGRAMA PRINCIPAL

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
    npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t,
    alpha)

coefP = CoeficienteP(npassos_t, alpha)

matriz_A = MatrizA(limite_x, npassos_x, limite_t, npassos_t, K,
```

```
    tau, alpha)

matriz_I = np.identity(npassos_x-1)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
r = (tau ** alpha) * K * (delta_t**(1-alpha)) / delta_x ** 2



n = 1
while n <= npassos_t:
    vprod = np.dot(matriz_I - coefP[n] * matriz_A, U[n - 1])

    vprod[0] = vprod[0] + coefP[n] * r * (fesquerda[n] -
        fesquerda[n - 1])
    vprod[-1] = vprod[-1] + coefP[n] * r * (fdireita[n] -
        fdireita[n - 1])

    U[n] = np.linalg.solve(matriz_I - coefP[n] * matriz_A, vprod
        + delta_t * fonte[n])
    n = n + 1


fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
print()


# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t, alpha)
```

```
fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60

print(f'O pos-processamento foi executado em {segundospos:.1f} segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

B.5 Método tipo Euler regressivo – Derivada de Katugampola

```
import numpy as np
from sympy import gamma, sin, exp
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR


def CondicaoInicial(limite_x, npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = 0
    return condicao_inicial


def CondicaoFronteira(limite_x, limite_t, npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = (n * delta_t) ** 1.5 * exp(np.pi
            * (0 - alpha))
        front_na_direita[n] = (n * delta_t) ** 1.5 * exp(np.pi *
            (limite_x - alpha))
    return front_na_esquerda, front_na_direita


def TermoFonte(limite_x, npassos_x, limite_t, npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(1, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = 1.5 * (n*delta_t)**0.5 * (1 - (n*
                delta_t)**(1 - alpha) * np.pi**2) * exp(np.pi *
                ((i*delta_x) - alpha))
```

```
    return termo


def CoeficienteT(npassos_t, alpha):
    coeficiente = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        coeficiente[n] = n**(1-alpha)
    return coeficiente


def MatrizA(limite_x, npassos_x, limite_t, npassos_t, K, tau,
   alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    r = (tau ** alpha) * K * (delta_t**(1 - alpha)) / delta_x **
        2
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = -2 * r
        matriz[i][i+1] = r
        matriz[i+1][i] = r
    matriz[npassos_x-2][npassos_x-2] = -2 * r
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
   limite_t, npassos_t, alpha):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
       npassos_t, alpha)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max
```

```
def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t,
    alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = (n * delta_t) ** 1.5 * exp(np.
                pi * (i * delta_x - alpha))
    return sanalitica


# ENTRADA DOS DADOS

limite_x = int(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = int(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Coeficiente de difusao '))
alpha = float(input('Ordem da derivada fracionaria '))
tau = float(input('Parametro de correcao dimensional '))


# PROGRAMA PRINCIPAL

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
    npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t)

coefT = CoeficienteT(npassos_t, alpha)
```

```
matriz_A = MatrizA(limite_x, npassos_x, limite_t, npassos_t, K,
    tau, alpha)

matriz_I = np.identity(npassos_x-1)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
r = (tau ** alpha) * K * (delta_t**(1-alpha)) / delta_x ** 2

n = 1
while n <= npassos_t:
    vprod = np.dot(matriz_I - coefT[n] * matriz_A, U[n - 1])

    vprod[0] = vprod[0] + coefT[n] * r * (fesquerda[n] -
        fesquerda[n - 1])
    vprod[-1] = vprod[-1] + coefT[n] * r * (fdireita[n] -
        fdireita[n - 1])

    U[n] = np.linalg.solve(matriz_I - coefT[n] * matriz_A, vprod
         + delta_t * fonte[n])
    n = n + 1

fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
print()


# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t, alpha)
```

```
fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60

print(f'O pos-processamento foi executado em {segundospos:.1f} 
    segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

B.6 Método tipo Euler regressivo – Derivada de Caputo-Fabrizio

```
import numpy as np
from sympy import gamma, sin , exp
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR

def CondicaoInicial(limite_x , npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x -1)
    for i in range(1, npassos_x):
        condicao_inicial[i -1] = 0
    return condicao_inicial


def CondicaoFronteira(limite_x , limite_t , npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = (n * delta_t) * exp(n * delta_t)
           * sin(alpha * np.pi * 0)
        front_na_direita[n] = (n * delta_t) * exp(n * delta_t) *
            sin(alpha * np.pi * 1)
    return front_na_esquerda , front_na_direita


def TermoFonte(limite_x , npassos_x , limite_t , npassos_t):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo_fonte = np.zeros((npassos_t+1, npassos_x -1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            termo_fonte[n][i -1] = (((np.pi**2 * alpha**2 + 1) *
               (n*delta_t) + np.pi**2 * alpha**3 + 1)
            * exp(n*delta_t) - np.pi**2 * alpha**3 * exp(alpha *
               (n*delta_t)/(alpha -1))) * sin(alpha * np.pi * (i
```

```
            *delta_x))
    return termo_fonte


def CoeficienteE(limite_t, npassos_t, alpha):
    delta_t = limite_t / npassos_t
    coeficiente = np.zeros(npassos_t+1)
    for k in range(0, npassos_t+1):
        coeficiente[k] = exp(-(alpha * k * delta_t) / (1 - alpha
            ))
    return coeficiente


def MatrizB(limite_x, npassos_x, limite_t, npassos_t, K, tau,
   alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    s = (tau ** alpha) * K * (exp((alpha * delta_t)/(1 - alpha))
        -1)/(alpha * delta_x**2)
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = -2 * s
        matriz[i][i+1] = s
        matriz[i+1][i] = s
    matriz[npassos_x-2][npassos_x-2] = -2 * s
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
   limite_t, npassos_t, alpha):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
        npassos_t, alpha)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
```

```
    max = np.amax(desvio)
    return max


def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t,
   alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = (n * delta_t) * exp(n * delta_t
               ) * sin(alpha * np.pi * (i * delta_x))
    return sanalitica


# ENTRADA DOS DADOS

limite_x = int(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = int(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Coeficiente de difusao '))
alpha = float(input('Ordem da derivada fracionaria '))
tau = float(input('Parametro de correcao dimensional '))


# PROGRAMA PRINCIPAlimite_x

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
   npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t)
```

```
coeficiente_E = CoeficienteE(limite_t, npassos_t, alpha)

matriz_B = MatrizB(limite_x, npassos_x, limite_t, npassos_t, K,
    tau, alpha)

matriz_I = np.identity(npassos_x-1)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
s = (tau ** alpha) * K * (exp((alpha * delta_t)/(1 - alpha))-1)
    /(alpha * delta_x**2)

n = 1
while n <= npassos_t:
    auxiliar = np.zeros(npassos_x-1)
    for k in range(2, n+1):
        produto = np.dot(matriz_B, U[n-k+1] - U[n-k])
        produto[0] = produto[0] + s * (fesquerda[n-k+1] -
            fesquerda[n-k])
        produto[-1] = produto[-1] + s * (fdireita[n-k+1] -
            fdireita[n-k])
        auxiliar = auxiliar + coeficiente_E[k] * produto

    auxiliar = auxiliar + np.dot(matriz_I - coeficiente_E[1] *
        matriz_B, U[n - 1])
    auxiliar[0] = auxiliar[0] + s * coeficiente_E[1] * (
        fesquerda[n] - fesquerda[n-1])
    auxiliar[-1] = auxiliar[-1] + s * coeficiente_E[1] * (
        fdireita[n] - fdireita[n-1])
    U[n] = np.linalg.solve(matriz_I - coeficiente_E[1] *
        matriz_B, auxiliar + delta_t * fonte[n])
    n = n + 1

fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
```

```
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
    minutos:.1f} minutos')
print()


# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
    npassos_x, limite_t, npassos_t, alpha)

fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60

print(f'O pos-processamento foi executado em {segundospos:.1f}
    segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```

B.7 Método tipo Euler regressivo – Derivada de Atagana-Baleanu

```
import numpy as np
from sympy import gamma, exp, sin
import time


# FUNCOES DEFINIDAS PELO PROGRAMADOR

def CondicaoInicial(limite_x, npassos_x):
    delta_x = limite_x / npassos_x
    condicao_inicial = np.zeros(npassos_x-1)
    for i in range(1, npassos_x):
        condicao_inicial[i-1] = 0
    return condicao_inicial


def CondicaoFronteira(limite_x, limite_t, npassos_t):
    delta_t = limite_t / npassos_t
    front_na_esquerda = np.zeros(npassos_t+1)
    front_na_direita = np.zeros(npassos_t+1)
    for n in range(0, npassos_t+1):
        front_na_esquerda[n] = 0
        front_na_direita[n] = 0
    return front_na_esquerda, front_na_direita


def TermoFonte(limite_x, npassos_x, limite_t, npassos_t, alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    termo = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            termo[n][i-1] = (4 * (n * delta_t) ** 3 + ((24 * (np
                .pi ** 2) * (n * delta_t) ** 4) / (1-alpha)) *
             MittagLeffler(alpha, 5, - alpha * ((n * delta_t)**
               alpha) / (1-alpha) ) ) * sin(np.pi * (i *
               delta_x))
    return termo
```

```
def MittagLeffler(alpha, beta, parametro):
    variavel_auxiliar = 0
    for k in range(0, 50):
        mta = parametro**k / gamma(alpha * k + beta)
        variavel_auxiliar = variavel_auxiliar + mta
    return variavel_auxiliar


def MittagLeffler2(alpha, npassos_t, limite_t):
    delta_t = limite_t / npassos_t
    k = 0
    vetorML = np.zeros(npassos_t + 1)
    while k <= npassos_t:
        variavel_auxiliar = 0
        for j in range(0, 50):
            mta = ((- alpha / (1 - alpha)) * (npassos_t *
               delta_t - k * delta_t) ** alpha)**j / gamma(alpha
                * j + 2)
            variavel_auxiliar = variavel_auxiliar + mta
        vetorML[k] = variavel_auxiliar
        k = k + 1
    return vetorML


def CoeficienteC(n, npassos_t, vetor_E):
    vetorC = np.zeros(n+1)
    vetorC[0] = (n-1) * vetor_E[npassos_t - n + 1] - n * vetor_E
       [npassos_t - n + 0]
    for j in range(1, n):
        vetorC[j] = (n - j + 1) * vetor_E[npassos_t - n + j - 1]
            - 2*(n-j) * vetor_E[npassos_t - n + j] + (n-j-1) *
            vetor_E[npassos_t - n + j + 1]
    vetorC[n] = vetor_E[npassos_t - n + n - 1]
    return vetorC


def MatrizD(limite_x, npassos_x, limite_t, npassos_t, K, tau,
```

```
    alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    v = (tau ** alpha) * K * delta_t / ((1 - alpha) * delta_x **
        2)
    matriz = np.zeros((npassos_x-1, npassos_x-1))
    for i in range(0, npassos_x-2):
        matriz[i][i] = -2 * v
        matriz[i][i+1] = v
        matriz[i+1][i] = v
    matriz[npassos_x-2][npassos_x-2] = -2 * v
    return matriz


def PosProcessamento(SolucaoAnalitica, U, limite_x, npassos_x,
    limite_t, npassos_t, alpha):
    solucao = SolucaoAnalitica(limite_x, npassos_x, limite_t,
        npassos_t, alpha)
    delta_x = limite_x / npassos_x
    desvio = np.zeros(npassos_t+1)
    for n in range(1, npassos_t+1):
        L2 = 0
        for i in range(1, npassos_x):
            L2 = L2 + (solucao[n][i-1] - U[n][i-1]) ** 2
        desvio[n] = (delta_x * L2) ** 0.5
    max = np.amax(desvio)
    return max


def SolucaoAnalitica(limite_x, npassos_x, limite_t, npassos_t,
    alpha):
    delta_x = limite_x / npassos_x
    delta_t = limite_t / npassos_t
    sanalitica = np.zeros((npassos_t+1, npassos_x-1))
    for n in range(0, npassos_t+1):
        for i in range(1, npassos_x):
            sanalitica[n][i-1] = (n * delta_t) ** 4 * sin(np.pi
                * (i * delta_x))
    return sanalitica
```

```
# ENTRADA DOS DADOS

limite_x = int(input('Limite do eixo x '))
npassos_x = int(input('Numero de passos do eixo x '))
limite_t = int(input('Tempo maximo no eixo t '))
npassos_t = int(input('Numero de passos do eixo t '))
K = float(input('Coeficiente de difusao '))
alpha = float(input('Ordem da derivada fracionaria '))
tau = float(input('Parametro de correcao dimensional '))


# PROGRAMA PRINCIPAlimite_x

inicio = time.time()

U = np.zeros((npassos_t+1, npassos_x-1))

U[0] = CondicaoInicial(limite_x, npassos_x)

fesquerda, fdireita = CondicaoFronteira(limite_x, limite_t,
    npassos_t)

fonte = TermoFonte(limite_x, npassos_x, limite_t, npassos_t,
    alpha)

matriz_D = MatrizD(limite_x, npassos_x, limite_t, npassos_t, K,
    tau, alpha)

matriz_I = np.identity(npassos_x-1)

delta_x = limite_x / npassos_x
delta_t = limite_t / npassos_t
v = (tau ** alpha) * K * delta_t / ((1 - alpha) * delta_x ** 2)
vetor_E = MittagLeffler2(alpha, npassos_t, limite_t)


n = 1
```

```
while n <= npassos_t:
    auxiliar = np.zeros(npassos_x-1)
    coeficiente_C = CoeficienteC(n, npassos_t, vetor_E)
    for k in range(0, n):
        produto = np.dot(matriz_D, U[k])
        produto[0] = produto[0] + v * fesquerda[k]
        produto[-1] = produto[-1] + v * fdireita[k]
        auxiliar = auxiliar + coeficiente_C[k] * produto

    auxiliar[0] = auxiliar[0] + v * coeficiente_C[n] * fesquerda
       [n]
    auxiliar[-1] = auxiliar[-1] + v * coeficiente_C[n] *
       fdireita[n]
    U[n] = np.linalg.solve(matriz_I - coeficiente_C[n] *
       matriz_D, U[n-1] + auxiliar + delta_t * fonte[n])
    n = n + 1


fim = time.time()
segundos = fim - inicio
minutos = segundos / 60

print()
print(f'O codigo foi executado em {segundos:.1f} segundos ou {
   minutos:.1f} minutos')
print()


# POS-PROCESSAMENTO

iniciopos = time.time()

desvio_L2 = PosProcessamento(SolucaoAnalitica, U, limite_x,
   npassos_x, limite_t, npassos_t, alpha)

fimpos = time.time()
segundospos = fimpos - iniciopos
minutospos = segundospos / 60
```

```
print(f'O pos-processamento foi executado em {segundospos:.1f} segundos ou {minutospos:.1f} minutos')
print()
print(f' O desvio na norma L2 e: {desvio_L2:.4e}')
```