Revista Ciencia en Desarrollo. Vol. 15 Num. 2. pp. 43 - 52
julio - diciembre 2024

https://doi.org/10.19053/uptc.01217488.v15.n2.2024.17346

Artículo de investigación

Numfracpy, Técnicas del Cálculo Fraccionario en *Python*

Numfracpy, Fractional Calculus Techniques with *Python*

Jorge H. Lopez[1] ✉ y Alejandro P. Riascos[2],

[1]Departamento de Física, Universidad de Nariño, San Juan de Pasto, Colombia.
[2]Departamento de Física, Universidad Nacional de Colombia, Bogotá, Colombia.

Recepción: 12-mar-2024 **Aceptado:** 13-may-2024 **Publicado:** 23-jul-2024
Cómo citar: López Melo, J. H., & P. Riascos, A. (2024). Numfracpy, Técnicas del Cálculo Fraccionario en Python. Ciencia en Desarrollo, 15(2). https://doi.org/10.19053/uptc.01217488.v15.n2.2024.17346.

Resumen

En este trabajo se introduce una librería en el lenguaje *Python* que implementa técnicas propias del cálculo fraccionario. Este tipo de cálculo ha visto un incremento notable de sus aplicaciones en diversas áreas de las ciencias en las últimas décadas. Sin embargo, el tipo de cálculos que se necesitan para su desarrollo no son simples y no hay muchas ayudas computacionales para su implementación, especialmente en *Python*. *Numfracpy* se encuentra disponible al público en el índice de paquetes PyPI (*Python* Package Index) e implementa diversos conceptos del cálculo fraccionario como lo son: La integral y la derivada de Riemann-Liouville, la derivada de Caputo, la derivada de Grünwald-Letnikov, las funciones de Mittag-Leffler, la solución numérica de un tipo de ecuación diferencial en derivadas fraccionarias y un sistema de tales ecuaciones diferenciales. En este trabajo se presentan varios algoritmos implementados y los resultados obtenidos se comparan con aquellos reportados en la literatura, encontrando una buena aproximación en los diferentes ejemplos ilustrados.

Palabras Clave: Cálculo Fraccionario, Caputo, Grüwald-Letnikov, Métodos Numéricos, Python, Riemann-Liouville.

Abstract

This work introduces a *Python* library that implements techniques of fractional calculus. This type of calculus has seen a significant increase in its applications across various scientific areas in recent decades. However, the calculations required for its development are not straightforward, and there are limited computational tools available, especially in *Python*. *Numfracpy* is publicly available on the *Python* Package Index (PyPI) and incorporates various fractional calculus concepts, such as Riemann-Liouville integral and derivative, Caputo derivative, Grünwald-Letnikov derivative, Mittag-Leffler functions, numerical solution of a specific fractional differential equation, and a system of such differential equations. The paper presents several implemented algorithms, and the obtained results are compared with those reported in the literature, demonstrating good approximations in the various illustrated examples.

Keywords: Fractional Calculus, Caputo, Grüwald-Letnikov, Numerical Methods, Python, Riemann-Liouville.

E-ISSN: 2462-7658

1. Introducción

El cálculo fraccionario surge para introducir derivadas de orden no entero y permite unificar los conceptos de integral y derivada. Ya desde los orígenes del cálculo clásico se especuló acerca del significado de una derivada de orden 1/2, de tal manera que $D^{1/2}D^{1/2}f = Df$, como se evidencia en una carta de Gottfried Wilhelm Leibniz a Guillaume de L'Höpital en 1695 [1]. Sin embargo, su desarrollo se dió en los siglos XIX y XX, principalmente. Fue Niels Henrik Abel quien da un primer desarrollo de las nociones fundamentales de la integral y la derivada de orden *no* entero y su unificación [2, 3], trabajo análogo que desarrolló paralelamente Joseph Liouville [4, 5]. A lo largo de la historia se han dado múltiples definiciones para la integral y derivada de orden no entero, lo cual ha obstaculizado, en cierta medida la asimilación de estas técnicas en las diferentes áreas de las ciencias. Sin embargo, en la literatura moderna [6, 7] la mayoría de las definiciones suelen fundamentarse en la integral de Riemann-Liouville, y aunque hay diversas definiciones para la derivada [8, 9, 10] como: la derivada de Riemann-Liouville, de Caputo, de Grüwald-Letnikov, y de Riesz; la mayoría de las aplicaciones que usan ecuaciones diferenciales en derivadas fraccionarias lo hacen aplicando la derivada de Caputo pues las condiciones iniciales replican las que se tienen para las ecuaciones diferenciales en derivadas ordinarias. Es importante destacar, que además del interés teórico natural que recibe el cálculo fraccionario, se ha logrado aplicarlo en áreas tan diversas como: el estudio de la viscoelasticidad [11], la difusión anómala en sistemas biológicos [12], propagación ondulatoria [13], imágenes médicas por ultrasonido y elastografía [14], la espectroscopía de impedancia eléctrica [15], la teoría de control [16], el tratamiento de imágenes [17], entre muchos otros [18, 19]. Sin embargo, los cálculos para resolver modelos sencillos no son simples y con este fin se disponen de algunas ayudas computacionales. En el software *MatLab* se dispone de paquetes como [20] *FOTF*, *NINTEGER*, *CRONE*, *FOMCON*, *DFOD*, y los mencionados en [21], entre otros. En *Fortran* se tiene el código [22] denominado *AstroFracTool*. En cuanto a *Python* se dispone del paquete [23] *FOMCOpy* (extensión de *FOMCO* en *Matlab*) aplicado en el Internet de las Cosas, *fracdiff* para derivadas [24] y su extensión a integrales [25], como también *differint* [26] para el cálculo de derivadas. Sin embargo, no conocemos de un paquete más completo en *Python* que, además de calcular derivadas e integrales de orden no entero, pueda desarrollar la solución numérica de una ecuación en derivadas fraccionarias y un sistema de tales ecuaciones. Precisamente con este objetivo hemos desarrollado *Numfracpy*. A continuación, se describen los algoritmos empleados, se presentan los métodos implementados en la librería, seguido por su aplicación con ejemplos en varios contextos.

2. Metodología

A continuación indicamos los algoritmos implementados en la resolución numérica de integrales, derivadas y ecuaciones diferenciales.

2.1. Integral de Riemann-Liouville

La integral fraccionaria de Riemann-Liouville de orden α, con $\alpha > 0$, de una función $f(t)$, está definida por [6, 7, 28]

$$D_{a,t}^{-\alpha} = {}_{RL}D_{a,t}^{-\alpha} f(t) = \frac{1}{\Gamma(\alpha)} \int_a^t (t-s)^{\alpha-1} f(s)\, ds. \qquad (1)$$

Esta definición es la noción más generalizada de integral en el cálculo fraccionario y se constituye en la piedra angular del mismo. Permite unificar los conceptos de derivada e integral del cálculo clásico y además, extenderlos a órdenes no enteros. En particular, nótese que ${}_{RL}D_{a,t}^{-1} f(t) = \int_a^t f(s)\, ds$. Usualmente se toma el valor $a = 0$, lo cual seguiremos en este trabajo. También puede notarse que para $\alpha = 0$ no está definida esta integral por la presencia del factor $\Gamma(\alpha)$ en el denominador. Obsérvese además que la misma notación indica que una integral puede verse como una derivada de orden negativo. Numéricamente la integral de Riemann-Liouville puede evaluarse usando algoritmos clásicos para evaluar una integral definida [29, 30, 31]. Por ejemplo, se puede hacer uso de una interpolación polinomial y usar fórmulas como la rectangular, trapezoidal, de Simpson y de Newton-Cotes [29]. También puede hacerse una interpolación por splines cúbicos, una interpolación gaussiana, y aún, usar el método lineal de multipaso [32]. Nosotros calculamos la integral de Riemann-Liouville haciendo uso de la función *quad* del paquete *scipy* [33]. Esta función basa su funcionamiento en rutinas de la librería *QUADPACK* de *FORTRAN* [34], la cual contiene numerosas funcionalidades para el cálculo de una integral definida y ha sido usada ampliamente durante décadas.

2.2. Derivada de Caputo

La derivada fraccionaria de Caputo de orden α de una función $f(t)$, se fundamenta en la integral de Riemann-Liouville y está dada por [6, 7]

$$\begin{aligned} {}_{C}D_{a,t}^{\alpha} &= D_{a,t}^{-(m-\alpha)} \left[f^{(m)}(t) \right] \\ &= \frac{1}{\Gamma(m-\alpha)} \int_a^t (t-s)^{m-\alpha-1} f^{(m)}(s)\, ds, \end{aligned} \qquad (2)$$

donde m es un entero positivo tal que $m-1 < \alpha < m$. Por tanto, para calcular la derivada de Caputo de una función f se deriva primero m veces y luego se aplica la integral de Riemann-Liouville de orden $m-\alpha$. Asumimos, como es costumbre, que $a = 0$. En nuestra librería implementamos dos algoritmos denominados *L1* y *L2* [35] para calcular numéricamente la derivada de Caputo para los casos $m = 1$ y $m = 2$ respectivamente. Estos algoritmos tienen muy buena estabilidad [36, 37] y se aproxima de una manera simple la derivada de la función f.

2.2.1. Método L1, caso 0 <α <1

Para $0 < \alpha < 1$ la derivada fraccionaria de Caputo está dada por

$$\begin{aligned} {}_{C}D_{0,t}^{\alpha} &= D_{0,t}^{-(1-\alpha)} \left[f'(t) \right] \\ &= \frac{1}{\Gamma(1-\alpha)} \int_0^t (t-s)^{-\alpha} f'(s)\, ds \\ &= \sum_{k=0}^{n-1} b_{n-k-1} (f(t_{k+1}) - f(t_k)). \end{aligned} \qquad (3)$$

En este método se realiza una discretización uniforme del intervalo temporal $(0, t = t_n)$ en la forma $[0, \Delta t, 2\Delta t, \ldots, n\Delta t]$, de manera que $t_k = k\Delta t$. La derivada $f'(t)$ se aproxima por una diferencia finita

hacia adelante y se obtiene

$$\begin{aligned}&\left[{}_{C}D^{\alpha}_{0,t}f(t)\right]_{t=t_n} \approx \\ &\frac{1}{\Gamma(1-\alpha)}\sum_{k=0}^{n-1}\int_{t_k}^{t_{k+1}}(t_n-s)^{-\alpha}\frac{f(t_{k+1})-f(t_k)}{\Delta t}\,ds \\ &=\sum_{k=0}^{n-1}b_{n-k-1}(f(t_{k+1})-f(t_k)),\end{aligned} \tag{4}$$

donde

$$b_k=\frac{\Delta t^{-\alpha}}{\Gamma(2-\alpha)}\left[(k+1)^{1-\alpha}-k^{1-\alpha}\right]. \tag{5}$$

2.2.2. Método L2, caso $1<\alpha<2$

Para $1<\alpha<2$ la derivada fraccionaria de Caputo está dada por

$$\begin{aligned}{}_{C}D^{\alpha}_{0,t}&=D^{-(2-\alpha)}_{0,t}\left[f''(t)\right]\\ &=\frac{1}{\Gamma(2-\alpha)}\int_0^t(t-s)^{1-\alpha}f''(s)\,ds.\end{aligned} \tag{6}$$

Haciendo una discretización similar a la del método *L1* se cumple

$$\begin{aligned}{}_{C}D^{\alpha}_{0,t}&=\frac{1}{\Gamma(2-\alpha)}\int_0^{t_n}(t_n-s)^{1-\alpha}f''(s)\,ds\\ &=\frac{1}{\Gamma(2-\alpha)}\sum_{k=0}^{n-1}\int_{t_k}^{t_{k+1}}(t_n-s)^{1-\alpha}f''(s)\,ds\\ &=\frac{1}{\Gamma(2-\alpha)}\sum_{k=0}^{n-1}\int_{t_k}^{t_{k+1}}s^{1-\alpha}f''(t_n-s)\,ds.\end{aligned} \tag{7}$$

En el último paso se hizo un cambio de variable y se tuvo en cuenta que el proceso se hace sobre todo el recorrido del sumatorio indicado. Aproximando la segunda derivada $f''(t_n-s)$ por

$$\frac{f(t_n-t_{k+1})-2f(t_n-t_k)+f(t_n-t_{k-1})}{\Delta t^2}, \tag{8}$$

después de algo de álgebra elemental se obtiene

$$\begin{aligned}&{}_{C}D^{\alpha}_{0,t}\\ &\approx\frac{1}{\Gamma(2-\alpha)}\sum_{k=0}^{n-1}\frac{f(t_n-t_{k+1})-2f(t_n-t_k)+f(t_n-t_{k-1})}{\Delta t^2}\\ &\int_{t_k}^{t_{k+1}}s^{1-\alpha}\,ds=\sum_{k=-1}^{n}W_k f(t_{n-k}),\end{aligned} \tag{9}$$

donde

$$W_k=\frac{\Delta t^{-\alpha}}{\Gamma(3-\alpha)}\begin{cases}1, & k=-1\\ 2^{2-\alpha}-3, & k=0\\ (k+2)^{2-\alpha}-3(k+1)^{2-\alpha}+3k^{2-\alpha}-(k-1)^{2-\alpha}, & 1\le k\le n-2\\ -2n^{2-\alpha}+3(n-1)^{2-\alpha}-(n-2)^{2-\alpha}, & k=n-1\\ n^{2-\alpha}-(n-1)^{2-\alpha}, & k=n\end{cases} \tag{10}$$

2.3. Derivada de Riemann-Liouville

La derivada fraccionaria de Riemann-Liouville de orden α para una función $f(t)$, está dada por [6, 7]

$$\begin{aligned}{}_{RL}D^{\alpha}_{a,t}&=\frac{d^m}{dt^m}\left[D^{-(m-\alpha)}_{a,t}f(t)\right]\\ &=\frac{1}{\Gamma(m-\alpha)}\frac{d^m}{dt^m}\int_a^t(t-s)^{m-\alpha-1}f(s)\,ds,\end{aligned} \tag{11}$$

donde $m=\lceil\alpha\rceil$, y $\lceil\,\rceil$ representa la función techo. En particular, para $0<\alpha<1$ se tiene $m=1$ y

$${}_{RL}D^{\alpha}_{a,t}=\frac{1}{\Gamma(1-\alpha)}\frac{d}{dt}\int_a^t(t-s)^{-\alpha}f(s)\,ds. \tag{12}$$

Para una función f suficientemente suave, es decir $f\in C^m[0,t]$, se satisface la siguiente relación [35]

$${}_{RL}D^{\alpha}_{0,t}f(t)={}_{C}D^{\alpha}_{0,t}f(t)+\sum_{k=0}^{m-1}\frac{f^{(k)}(0)t^{k-\alpha}}{\Gamma(k+1-\alpha)}. \tag{13}$$

Por tanto, los métodos *L1* y *L2* para los casos $0<\alpha<1$ y $1<\alpha<2$ respectivamente, para la derivada de Caputo se pueden utilizar directamente para aproximar la derivada de Riemann-Liouville. Esta es la manera en que calculamos la derivada de Riemann-Liouville en la librería *Numfracpy*.

2.4. Derivada de Grünwald-Letnikov

La derivada de Grünwald-Letnikov [6, 7] de orden $\alpha>0$ de la función $f(t)$, está definida por

$${}_{GL}D^{\alpha}_{a,t}f(t)=\lim_{\substack{h\to 0\\ Nh=t-a}}h^{-\alpha}\sum_{j=0}^{N}(-1)^j\binom{\alpha}{j}f(t-jh). \tag{14}$$

Esta derivada puede verse como una aproximación por sumas de Riemann de la integral de Riemann-Liouville de orden $-\alpha$ [38]. Nótese que una integral de orden negativo se concibe como una derivada en el cálculo fraccionario. Se asume que el intervalo en la variable independiente se discretiza con un paso h y se toma $a=0$ como de costumbre. En nuestra librería aproximamos la derivada de Grünwald-Letnikov directamente por su definición para $0<\alpha<1$. Esta definición puede escribirse en la forma

$$\left[{}_{GL}D^{\alpha}_{0,t}\right]_{t=t_n}\approx\frac{1}{\Delta t^{\alpha}}\sum_{j=0}^{n}\omega_j^{(\alpha)}f(t_{n-j}), \tag{15}$$

con $\omega_j^{(\alpha)}=(-1)^j\binom{\alpha}{j}$. Para el caso $1<\alpha<2$, el esquema anterior puede llevar a inestabilidades numéricas. Por lo cual, hacemos uso de la siguiente fórmula modificada de Grünwald-Letnikov, con $p=1$ de acuerdo con [35]

$$\left[{}_{GL}D^{\alpha}_{0,t}\right]_{t=t_n}\approx\frac{1}{\Delta t^{\alpha}}\sum_{j=0}^{n+p}\omega_j^{(\alpha)}f(t_{n-j+p}). \tag{16}$$

2.5. Solución numérica de una ecuación diferencial

Consideramos ahora el siguiente problema de valor inicial

$$\begin{cases}{}_{C}D^{\alpha}_{0,t}u(t)=f(t,u(t)), & m-1<\alpha<m\in\mathbb{Z}^+\\ u^j(0)=u_0^j, & j=0,1,\cdots,m-1\end{cases} \tag{17}$$

donde ${}_{C}D^{\alpha}_{0,t}$ es la derivada de Caputo y u_0^j es la derivada de u de orden j en $t=0$. Este problema es equivalente a la siguiente ecuación integral de Volterra [39]

$$\begin{aligned}u(t)&=\sum_{j=0}^{m-1}\frac{t^j}{j!}u_0^j+\frac{1}{\Gamma(\alpha)}\int_0^t(t-s)^{\alpha-1}f(s,u(s))ds\\ &=\sum_{j=0}^{m-1}\frac{t^j}{j!}u_0^j+{}_{RL}D^{-\alpha}_{0,t}f(t,u(t)).\end{aligned} \tag{18}$$

Para resolver este problema, se implementa el método *fraccional de Adams* basado en las siguientes relaciones [35].

$$u^P_{n+1} = \sum_{j=0}^{m-1} \frac{t^j_{n+1}}{j!} u^j_0 + \Delta t^\alpha \sum_{j=0}^{n} b_{j,n+1} f(t_j, u_j)$$
$$u_{n+1} = \sum_{j=0}^{m-1} \frac{t^j_{n+1}}{j!} u^j_0 + \sum_{j=0}^{n} a_{j,n+1} f(t_j, u_j) + a_{n+1,n+1} f(t_{n+1}, u^P_{n+1}), \tag{19}$$

donde

$$b_{j,n} = \frac{1}{\Gamma(\alpha+1)} \left[(n-j)^\alpha - (n-1-j)^\alpha\right] \tag{20}$$

$$a_{j,n} = \frac{\Delta t^\alpha}{\Gamma(\alpha+2)} \begin{cases} (n-1)^{\alpha+1} - (n-1-\alpha)n^\alpha, & j=0 \\ (n-j+1)^{\alpha+1} - 2(n-j)^{\alpha+1} + (n-1-j)^{\alpha+1}, & 1 \le j \le n-1 \\ 1, & j=n \end{cases} \tag{21}$$

2.6. Sistema de ecuaciones diferenciales

Algunos métodos, como los de Runge-Kutta pueden extrapolarse en forma natural para considerar un sistema de ecuaciones diferenciales en el cálculo clásico. Para el caso de un sistema de ecuaciones diferenciales en derivadas fraccionarias podemos hacer algo similar con el método fraccional de Adams. En este punto, conviene hacer distinción entre una ecuación multi-término y un sistema multi-orden [40] como se explica a continuación.

2.6.1. Sistema multi-orden

Se define un sistema multi-orden a un sistema de ecuaciones diferenciales de la forma

$$\begin{aligned} {}_cD^{\alpha_1} y_1(x) &= f_1(x, y_1, \ldots, y_n) \\ &\vdots \\ {}_cD^{\alpha_n} y_n(x) &= f_n(x, y_1, \ldots, y_n). \end{aligned} \tag{22}$$

Asumiendo $0 < \alpha_i < 1$, las condiciones iniciales serán de la forma $y_i(0) = y_{i,0}$ con $i = 1, 2, \ldots, n$. En caso que para alguna i, $\alpha_i > 1$, se deben proporcionar además los valores para $y_i'(0), y_i'(0), \ldots, y_i^{m-1}(0)$, donde $m = \lceil \alpha_i \rceil$. En este caso el método de Adams se extiende en forma natural y las relaciones para resolver el sistema de ecs. (22) está dado por

$$\begin{aligned} y^P_{i,n+1} &= \sum_{j=0}^{m-1} \frac{x^j_{n+1}}{j!} y^j_{i,0} \\ &\quad + \Delta t^\alpha \sum_{j=0}^{n} b_{j,n+1} f_i(x_j, y_{1,j}, \ldots, y_{n,j}) \\ y_{i,n+1} &= \sum_{j=0}^{m-1} \frac{x^j_{n+1}}{j!} y^j_{i,0} + \sum_{j=0}^{n} a_{j,n+1} f_i(x_j, y_{1,j}, \ldots, y_{n,j}) \\ &\quad + a_{n+1,n+1} f_i(x_j, y^P_{1,j}, \ldots, y^P_{n,j}), \end{aligned} \tag{23}$$

con $b_{i,n}$ y $a_{i,n}$ están dados por las ecs. (20) y (21) respectivamente.

2.6.2. Ecuación multi-término

Una ecuación multi-término en derivadas fraccionarias se define formalmente por la expresión

$${}_cD^{\alpha_n} y(x) = f(x, y(x), {}_cD^{\alpha_1(x)}, \ldots, {}_cD^{\alpha_{n-1}(x)}), \tag{24}$$

con condiciones iniciales

$$y^{(i)}(0) = y_0^{(i)}, \quad i = 0, 1, \ldots, m-1, \tag{25}$$

donde $m = \lceil \alpha_n \rceil$. Se asume, sin pérdida de generalidad, que $0 < \alpha_1 < \alpha_2 < \cdots < \alpha_n$ y los enteros contenidos en el intervalo $(0, \alpha_n)$ se encuentra en la secuencia finita $\{\alpha_k\}_{k=1}^n$. Por tanto, se cumple $0 < \alpha_{j+1} - \alpha_j \le 1$ para $j = 1, 2, \ldots, n-1$. Para resolver numéricamente una ecuación multi-orden tenemos en cuenta el siguiente teorema (ver [40]) que nos permite usar el sistema multi-orden asociado a tal ecuación.

Teorema 1 *La ecuación multi-término (24) con condiciones iniciales (25) es equivalente al siguiente sistema multi-orden*

$$\begin{aligned} {}_cD^{\beta_1} y_1(x) &= y_2(x) \\ {}_cD^{\beta_2} y_2(x) &= y_3(x) \\ &\vdots \\ {}_cD^{\beta_{n-1}} y_{n-1}(x) &= y_n(x) \\ {}_cD^{\beta_n} y_n &= f(x, y_1, y_2, \ldots, y_n), \end{aligned} \tag{26}$$

con las siguientes condiciones iniciales

$$y_j(0) = \begin{cases} y_0^{(0)} & \text{si} \quad j = 1 \\ y_0^{(k)} & \text{si} \quad \alpha_{j-1} = k \in \mathbb{N} \\ 0 & \text{otro caso} \end{cases} \tag{27}$$

donde $\beta_1 = \alpha_1$ y $\beta_j = \alpha_j - \alpha_{j-1}$ para $j = 2, 3, \ldots, n$. Nótese que $0 < \beta_j \le 1$ para todo j.

3. Resultados y discusión

Numfracpy contiene los métodos que se enuncian en la tabla 2.

Es importante mencionar que los métodos asociados a las funciones de Mittag-Leffler fueron tomados de la fuente [41]. A su vez, este código se basa en el importante artículo de Garrappa [27]. Las funciones de Mittag-Leffler son esenciales para el cálculo fraccionario, y su papel es análogo al de la función exponencial en el cálculo clásico. La función de Mittag-Leffler de dos parámetros se define por la expresión [27]

$$E_{\alpha,\beta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + \beta)}. \tag{28}$$

La función de Mittag-Leffler de un parámetro está dada por $E_\alpha(z) = E_{\alpha,1}(z)$. Adicionalmente, la función de Mittag-Leffler de tres parámetros está dada por [27]

$$E^\gamma_{\alpha,\beta}(z) = \frac{1}{\Gamma(\gamma)} \sum_{k=0}^{\infty} \frac{\Gamma(\gamma+k) z^k}{k! \Gamma(\alpha k + \beta)}. \tag{29}$$

3.1. Comparación de resultados entre derivadas

Ilustraremos los resultados obtenidos para diversas derivadas en nuestra librería comparándolos con los siguientes resultados teóricos [26, 35]

$$\begin{aligned} {}_{RL}D^{1/2}_{0,x}(\sqrt{x})|_{x=1} &= {}_{C}D^{1/2}_{0,x}(\sqrt{x})|_{x=1} \\ &= \frac{\sqrt{\pi}}{2} \approx 0.886226925453 \end{aligned} \tag{30}$$

$${}_{RL}D^{1/2}_{0,x}(x^2-1)|_{x=1} = \frac{5}{3\sqrt{\pi}} \approx 0.94031597258 \tag{31}$$

$${}_{C}D^{1/2}_{0,x}(x^2-1)|_{x=1} = \frac{8}{3\sqrt{\pi}} \approx 1.504505556127 \tag{32}$$

$$\begin{aligned} {}_{RL}D^{1/2}_{0,x}(e^x)|_{x=1} &= e\cdot \mathtt{erf}(1) + \frac{1}{\sqrt{\pi}} \\ &\approx 2.854887836 \end{aligned} \tag{33}$$

$${}_{C}D^{1/2}_{0,x}(e^x)|_{x=1} = e\cdot \mathtt{erf}(1) \approx 2.2906982523, \tag{34}$$

donde $\mathtt{erf}(x)$ es la función de error. Recordemos que la relación entre la derivada de Riemann-Liouville y Caputo está dada por (ec. (13) con $m=1$)

$${}_{RL}D^{\alpha}_{0,t}f(t) = {}_{C}D^{\alpha}_{0,t}f(t) + \frac{f(0)t^{-\alpha}}{\Gamma(1-\alpha)}. \tag{35}$$

En la tabla 1 se ilustran los resultados obtenidos para varias derivadas, en la última columna se muestra el error relativo. Se consideró una discretización del intervalo $[0,1]$ con 120 puntos y $\alpha = 1/2$.

Tabla 1: Comparación de las distintas derivadas para $\alpha = 1/2$.

Función	Derivada	Valor	Error Rel.
$\sqrt{x}$	RL1	0.88631629	1.008e-4
	Caputo	0.88631629	1.008e-4
	GL1	0.88627175	5.057e-5
x^2 - 1	RL1	0.93996561	3.726e-4
	Caputo	1.50415519	2.329e-4
	GL1	0.93620587	4.371e-3
e^x	RL1	2.85441978	1.640e-4
	Caputo	2.29023020	2.043e-4
	GL1	2.84836369	2.285e-3

Estos resultados deben compararse con los obtenidos usando la librería *differint* [26] en *Python*.

3.2. Solución numérica de una ecuación diferencial

La siguiente es una ecuación básica que ha sido empleada para modelar distintos comportamientos [42]

$${}_{C}D^{\alpha}_{0,x}u(t) = 1-u(t). \tag{36}$$

Para $0<\alpha<1$, $u(t)$ presenta un comportamiento monótono aproximándose a una asíntota horizontal como se ilustra en la figura 1. Este comportamiento puede modelar, por ejemplo, la evolución del esfuerzo en la viscoplasticidad [43].

Este resultado, en *numfracpy*, se obtiene usando el comando FODE(f, Initial, Interv, dx, alpha) con $f(t,u) = 1-u$, Initial = [0], Interv = [0,5], $dx = 0.01$ y alpha recorre los valores 0.25, 0.50, 0.75 y 1.00 para obtener cada curva.

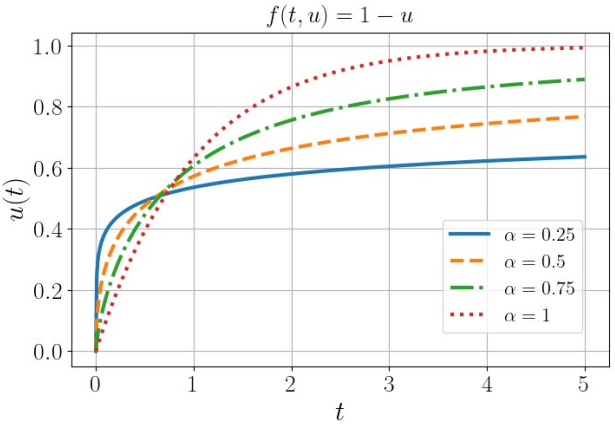

Figura 1: Solución de la ec. (36) para $0<\alpha\leq 1$ para $u(0)=0$ y $dx = 0.01$.

Análogamente se obtiene la solución a la ec. (36) para $\alpha = 1.00$, 1.25, 1.50 y 1.75, como se ilustra en la figura 2. En este caso la situación física se equipara a la de un oscilador amortiguado.

Cabe resaltar que los resultados ilustrados en las gráficas 1 y 2 están en claro acuerdo con lo reportado en [42].

3.3. Solución de un sistema de ecuaciones diferenciales

Consideraremos ahora la ecuación de Bagley-Torvik, la cual ha sido usada ampliamente en la literatura [44, 45, 46] para ilustrar la aplicación de técnicas computacionales en una ecuación multi-término. Mediante la ecuación de Bagley-Torvik, que se ilustra a continuación, se modela el movimiento de una placa rígida conectada a un resorte con un punto fijo e inmersa en un fluido Newtoniano.

$$AD^2y(x) + B\,{}_{C}D^{3/2}y(x) + Cy(x) = f(x). \tag{37}$$

Donde $y(x)$ representa el desplazamiento de la placa y las constantes A,B,C dependen de la viscosidad, la densidad del fluido, la rigidez del resorte y el área de la placa. Además, $f(x)$ representa la fuerza externa. En esta ecuación, D^2 representa naturalmente la segunda derivada. Se cumple entonces, por la sección 2.6.1, que $\alpha_1 = 1$, $\alpha_2 = 3/2$ y $\alpha_3 = 2$, y por Teorema 1 $\beta_1 = 1$, $\beta_2 = 1/2$ y $\beta_3 = 1/2$. Por tanto, el sistema multi-orden asociado es

$$\begin{aligned} y_1(x) &= y(x) \\ {}_{C}D^{1}y_1(x) &= y_2(x) \\ {}_{C}D^{1/2}y_2(x) &= y_3(x) \\ {}_{C}D^{1/2}y_3(x) &= A^{-1}\left(f(x) - By_3(x) - Cy_1(x)\right). \end{aligned} \tag{38}$$

Las condiciones iniciales, conforme al Teorema 1, están dadas por

$$\begin{aligned} &y_1(0) = y(0); \quad y_2(0) = {}_{C}D^{1}y_1(x)|_{x=0} = y'(0); \\ &y_3(0) = {}_{C}D^{3/2}y_1(x)|_{x=0} = 0, \end{aligned} \tag{39}$$

Tabla 2: Métodos implementados en *numfracpy*.

Nombre del Método	**Retorna**	**Parámetros**
RL_integral(f,a,t,alpha)	${}_{RL}D^{-\alpha}_{a,t} f(t)$	alpha representa α.
Caputo_der1(f,t,dt,alpha)	${}_{C}D^{\alpha}_{a,t}$ con $0<\alpha<1$	dt es el tamaño
Caputo_der2(f,t,dt,alpha)	${}_{C}D^{\alpha}_{a,t}$ con $1<\alpha<2$	de la discretización.
RL_der1(f,t,dt,alpha)	${}_{RL}D^{\alpha}_{a,t}$ con $0<\alpha<1$	
RL_der2(f,t,dt,alpha)	${}_{RL}D^{\alpha}_{a,t}$ con $1<\alpha<2$	
GLDer1(f,t,dt,alpha)	${}_{GL}D^{\alpha}_{a,t} f(t)$ con $0<\alpha<1$	
GLDer2(f,t,dt,alpha)	${}_{GL}D^{\alpha}_{a,t} f(t)$ con $1<\alpha<2$	
FODE(f,Initial,Interv,dx,alpha)	Una tupla (x,y) con x la discretización de $[a,b]$ con espacio dx y y el valor de f en cada punto de x.	Initial es una lista con las condiciones iniciales $[f(0),f'(0),f''(0),\ldots]$. Interv es una lista con los valores a y b.
SystemFODEs(f,Initial,Interv,dx ,alpha)	Resuelve un sistema multi-orden de la forma ${}_{C}D^{\alpha_i}u_i(t)=f_i(t,[u])$ con $i=1,2,\ldots,n$ y $[u]=[u_1,u_2,\ldots,u_n]$. Retorna una tupla $(x,ysol)$ donde x es una discretización de $[a,b]$ y $ysol$ es una doble lista $ysol[i][j]$ en la que i representa u_i y j es su valor en el punto $j$ésimo en la discretización de $[a,b]$.	f es una lista de funciones $[f_1,f_2,\ldots,f_n]$ con condiciones iniciales Initial = $[u_1(0),u_2(0),\ldots,u_n(0)]$. alpha representa la tupla $\alpha_1,\alpha_2,\ldots,\alpha_n$. Interv y dx se toman como en FODE.
Mittag_Leffler_one(z,alpha)	Valor función de Mittag-Leffler de un parámetro usando el algoritmo.	alpha, beta y gamma son los parámetros α, β y γ. z punto en que se evalúa la función.
Mittag_Leffler_two(z,alpha,beta)	Valor función de Mittag-Leffler de dos parámetros usando el algoritmo.	
Mittag_Leffler_three(z,alpha beta,gamma)	Valor función de Mittag-Leffler de tres parámetros usando el algoritmo.	
MittagLeffler_one_fsum(z,alpha, Nmax)	Valor función de Mittag-Leffler de un parámetro por la definición.	
MittagLeffler_two_fsum(z,alpha, beta,Nmax)	Valor función de Mittag-Leffler de dos parámetros por la definición.	

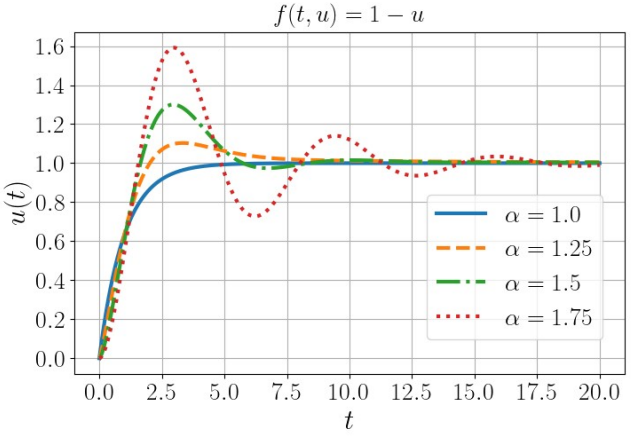

Figura 2: Solución de ec. (36) para $1\leq\alpha<2$ para $u(0)=0$ y $dx=0.01$.

pues estas son análogas a las condiciones iniciales clásicas e iguales a cero para las derivadas no enteras.

3.3.1. Ecuación Bagley-Torvik, caso A

Consideremos la ecuación de Bagley-Torvik con los parámetros $A=B=C=1$, la función $f(x)=x+1$, condiciones iniciales $y(0)=1$, $y'(0)=1$ y $x\in[0,5]$. La solución analítica está dada precisamente por [45]

$$y(x)=x+1. \tag{40}$$

Para obtener el resultado con *numfracpy* utilizamos el comando SystemFODEs(f,Initial,Interv,dx, alpha) para f = $[f_1,f_2,f_3]$, donde $f_1(x,y_1,y_2,y_3)=y_2$, $f_2(x,y_1,y_2,y_3)=y_3$ y $f_3(x,y_1,y_2,y_3)=f(x)-y_3-y_1$, Initial = [1,1,0], Interv = [0,5], dx = 0.001 y alpha = [1,1/2,1/2].

Nuestro resultado, ver figura 3, presenta una aceptable aproximación a la solución analítica con diferencias del orden de centésimas, este resultado puede contrastarse con el obtenido en [46], el cual desarrolla técnicas especiales para problemas con valor en la frontera. Cabe anotar que nuestro algoritmo utiliza un valor del paso dx bajo y puede mejorarse optimizando los algoritmos codificados. Sin embargo, son realmente escasas las librerías de software que pueden resolver un sistema de ecuaciones diferenciales en derivadas fraccionarias.

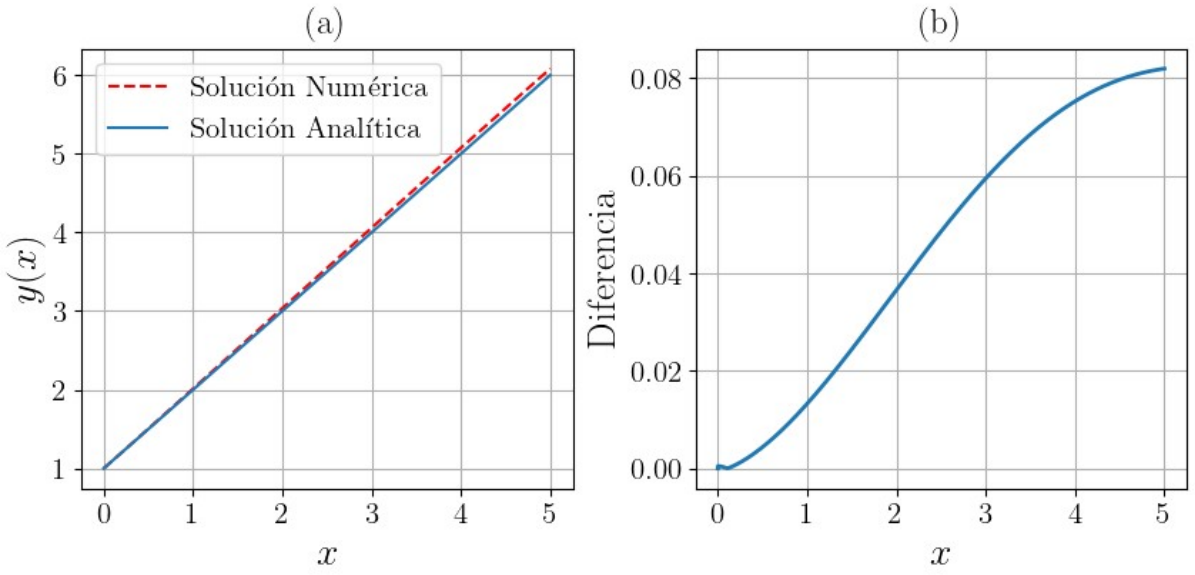

Figura 3: Solución de la ecuación Bagley-Torvik con $A = B = C = 1$, $y(0) = 1$, $y'(0) = 1$, $x \in [0,5]$ y $dx = 0.001$. (a) Comparación entre las soluciones analítica y numérica. (b) Valor absoluto de la diferencia entre las soluciones numérica y analítica.

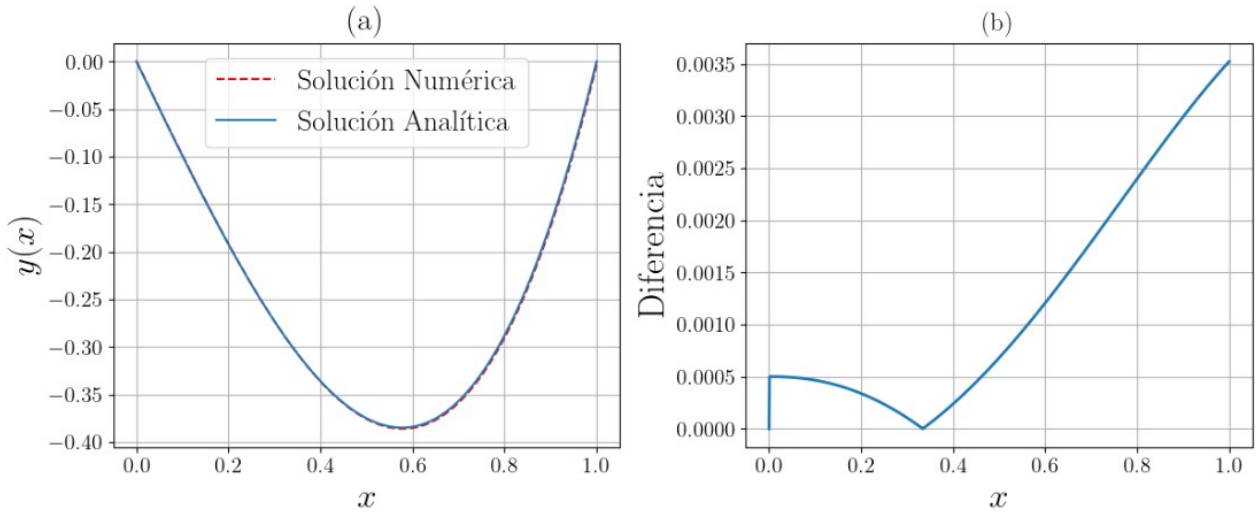

Figura 4: Solución de la ecuación Bagley-Torvik con $A = B = C = 1$, $y(0) = 0$, $y'(0) = -1$, $x \in [0,1]$ y $dx = 0.001$. (a) Comparación entre las soluciones analítica y numérica. (b) Valor absoluto de la diferencia entre las soluciones numérica y analítica.

3.3.2. Ecuación Bagley-Torvik, caso B

Ahora consideraremos la ecuación de Bagley-Torvik con los parámetros $A = B = C = 1$, la función $f(x) = x^3 + 5x + \frac{8}{\sqrt{\pi}}x^{3/2}$, condiciones iniciales $y(0) = 0$, $y'(0) = -1$ y $x \in [0,1]$. En este caso la solución analítica está dada por [46]

$$y(x) = x^3 - x. \tag{41}$$

El comando usado es análogo al que se empleó para el caso A y con el mismo valor para dx. La figura 4 (a) ilustra la comparación entre la solución analítica y la numérica, la cual muestra un muy buen ajuste entre ambos resultados. En la figura 4 (b) se ilustra la diferencia, en valor absoluto, entre los resultados analítico y numérico; la cual resulta ser del orden de milésimas.

4. Conclusiones

La librería de *Python*, *Numfracpy*, implementa la integral de Riemann-Liouville y tres diferentes derivadas fraccionarias: Derivada de Caputo, de Riemann-Liouville y de Grünwald-Letnikov; las cuales basan su definición en la primera integral. De acuerdo al desempeño mostrado para las funciones de la tabla 1 podemos observar diferencias relativas del orden de 10^{-4} con respecto al valor teórico. En comparación con la librería *differint* en [26] se obtienen mejores aproximaciones. Pocas librerías desarrolladas en *Python* existen actualmente [47], y es tal vez, *differint* la única otra librería que desarrolla diversas definiciones de la derivada fraccionaria.

Además, *numfracpy* logra desarrollar soluciones numéricas para ecuaciones diferenciales de la forma ${}_CD^{\alpha}u(t) = f(t,u(t))$. Este proceso fue ilustrado con la solución numérica de la ecuación (36) que puede visualizarse en las figuras 1 y 2. Estos resultados están en claro acuerdo con lo reportado en la literatura, ver por ejemplo [42]. Otra característica notable de *numfracpy* es que puede resolver numéricamente un sistema de ecuaciones diferenciales en derivadas fraccionarias, lo cual no es nada común en los paquetes de software que implementan el cálculo fraccionario. En particular, analizamos la ecuación de Bagley-Torvik que permite modelar el movimiento de una placa conectada a un resorte bajo la acción de una fuerza externa e inmersa en un fluido newtoniano. Nuestro estudio consideró dos tipos de fuerza distintas y los resultados fueron contrastados con los analíticos en las figuras 3 y 4 obteniendo resultados aceptables y con diferencias respecto a la solución numérica en el orden las centésimas y milésimas. En este caso extrapolamos directamente el método fraccional de Adams que fue utilizado para el caso de una ecuación diferencial fraccionaria. Aunque en la literatura se han desarrollado algoritmos con mejor desempeño en casos particulares, ver por ejemplo [46], es notable tener una librería de acceso libre que pueda resolver un sistema de ecuaciones en derivadas fraccionarias. Es de anotar que nuestra librería solo implementa ecuaciones diferenciales usando la derivada de Caputo, como es costumbre en la literatura, pues las condiciones iniciales reflejan las de las derivadas clásicas con una relación física directa.

En resumen, *Numfracpy* es una librería en *Python* que desarrolla aproximaciones numéricas para calcular integrales, derivadas y ecuaciones diferenciales propias del cálculo fraccionario, y aunque ciertamente, los algoritmos implementados pueden optimizarse, es tal vez la única librería de *Python* multipropósito de su clase. Además, la adición de nuestra librería de cálculo fraccionario a las herramientas ya existentes en *Python* abre el camino a su uso como apoyo en cursos de matemáticas, ecuaciones diferenciales y ciencia en general, siendo una oportunidad para motivar el estudio del cálculo fraccionario y sus diferentes aplicaciones a nivel de estudiantes de pregrado, posgrado y su implementación en temas de investigación.

Implicaciones Éticas

Este estudio no implicó intervención de seres vivos y se respetaron los derechos de autor. No conocemos ninguna contravención a alguna norma ética.

Contribuciones de los autores

Los autores confirman su contribución al artículo de la siguiente manera:

Ambos autores analizaron los fundamentos teóricos y algorítmicos de los métodos mencionados. Alejandro P. Riascos manejó mayormente la definición de algunas derivadas e integrales, así como las funciones de Mittag-Leffler. Jorge H. López estuvo mayormente involucrado con la definición e introducción de algunos conceptos básicos del cálculo fraccionario, las ecuaciones diferenciales en derivadas fraccionarias y la elaboración final del manuscrito. Ambos autores participaron escribiendo el código en Python de la librería Numfracpy, también ambos revisaron y aprobaron el manuscrito final.

Declaración de fuentes de financiación

No hubo un rubro específico para la realización de este proyecto. Sin embargo, se reconoce que las instituciones a las que están afiliados los autores fomentan la participación de los autores en la creación de dichos proyectos y en sus planes de trabajo tienen horas destinadas a la creación y desarrollo de estos.

Conflicto de interés

Los autores declaran que no existe ningún conflicto de interés en relación con la publicación de este artículo.

Referencias

[1] B. Ross, “The development of fractional calculus 1695 - 1900”, Historia Mathematica, vol. 4, no. 1, pp. 75 - 89, 1977. doi:10.1016/0315-0860(77)90039-8.

[2] N.H. Abel, L. Sylow and S. Lie “Solution de quelques problémes á láide díntégrales définies”, Oeuvres complétes de Niels Henrik Abel, pp. 11 - 27, 2012. doi:10.1017/cbo9781139245807.003.

[3] I. Podlubny, R. L. Magin, and I. Trymorush, “Niels Henrik Abel and the birth of fractional calculus”, Fractional Calculus and Applied Analysis, vol. 20, no. 5, pp. 1068-1075, 2017. doi:10.1515/fca-2017-0057.

[4] J. Liouville, “Mémoire sur quelques questions de géométrie et de mécanique, et sur un nouveau genre de calcul pour résoudre ces questions”, Journal de l’École Polytechnique, Paris, 13: 1-69. 1832.

[5] J. Liouville, “Mémoire sur le calcul des différentielles á indices quelconques”, Journal de l’École Polytechnique, Paris, 13: 71-162. 1832.

[6] K. Oldham and J. Spanier, “The fractional calculus theory and applications of differentiation and integration to arbitrary order”. Elsevier, 1974.

[7] I. Podlubny, “Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications”. Elsevier, 1998.

[8] G. Sales Teodoro, J. A. Tenreiro Machado, and E. Capelas de Oliveira, “A review of definitions of fractional derivatives and other operators”, Journal of Computational Physics, vol. 388, pp. 195-208, 2019. doi:10.1016/j.jcp.2019.03.008.

[9] J. T. Machado, V. Kiryakova, and F. Mainardi, “Recent history of fractional calculus”, Communications in Nonlinear Science and Numerical Simulation, vol. 16, no. 3, pp. 1140-1153, 2011. doi:10.1016/j.cnsns.2010.05.027.

[10] J. A. Tenreiro Machado et al., “Some applications of fractional calculus in engineering”, Mathematical Problems in Engineering, vol. 2010, pp. 1-34, 2010. doi:10.1155/2010/639801.

[11] R. C. Koeller, “Applications of fractional calculus to the theory of viscoelasticity”, Journal of Applied Mechanics, vol. 51, no. 2, pp. 299-307, 1984. doi:10.1115/1.3167616.

[12] J. F. Reverey et al., "Superdiffusion dominates intracellular particle motion in the supercrowded cytoplasm of pathogenic Acanthamoeba castellanii" Scientific Reports, vol. 5, no. 1, 2015. doi:10.1038/srep11690.

[13] N. Challamel and T. M. Atanackovic, "Fractional Calculus with Applications in Mechanics: Wave Propagation, Impact and Variational Principles". John Wiley and Sons Incorporated, 2014.

[14] S. Holm and S.P. Näsholm. "A causal and fractional all-frequency wave equation for lossy media". The Journal of the Acoustical Society of America, 130(4), pp. 2195-2202, 2011.

[15] F. Ciuchi, A. Mazzulla, N. Scaramuzza, E.K. Lenzi and L.R. Evangelista. "Fractional diffusion equation and the electrical impedance: Experimental evidence in liquid-crystalline cells". The Journal of Physical Chemistry C, 116(15), 8773-8777, 2012.

[16] P.D. Mandić, T.B. Sekara, M.P. Lazarević, and m. Bosković. "Dominant pole placement with fractional order PID controllers: D-decomposition approach". ISA transactions, 67, 76-86, 2017.

[17] J. Zhang, Z. Wei, and L. Xiao, L. "Adaptive fractional-order multi-scale method for image denoising". Journal of Mathematical Imaging and Vision, 43, 39-49, 2012.

[18] H. Sun, Y. Zhang, D. Baleanu, W. Chen, and Y. Chen, "A new collection of real world applications of fractional calculus in Science and Engineering", Communications in Nonlinear Science and Numerical Simulation, vol. 64, pp. 213-231, 2018. doi:10.1016/j.cnsns.2018.04.019.

[19] V. Tarasov, "On history of mathematical economics: Application of fractional calculus", Mathematics, vol. 7, no. 6, p. 509, 2019. doi:10.3390/math7060509.

[20] Z. Li, L. Liu, S. Dehghan, Y. Chen, and D. Xue, "A review and evaluation of numerical tools for fractional calculus and Fractional Order controls", International Journal of Control, vol. 90, no. 6, pp. 1165-1181, 2016. doi:10.1080/00207179.2015.1124290.

[21] R. Garrappa, "Numerical solution of fractional differential equations: A survey and a software tutorial", Mathematics, vol. 6, no. 2, p. 16, 2018. doi:10.3390/math6020016

[22] R. Marazzato and A. C. Sparavigna. "Astronomical image processing based on fractional calculus: the AstroFracTool." arXiv preprint arXiv:0910.4637, 2009.

[23] T. Onyedi, A. Tepljakov, and E. Petlenkov, "Fomconpy: Fractional-order modelling and Control Library for *python*", 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), 2020. doi:10.1109/tsp49548.2020.9163581.

[24] T. Dasgupta and M. Maitra, "An extremely fast and accurate fractional order differentiator" in 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp. 1-4, July, 2017.

[25] T. Midya, D. Garai, and T. Dasgupta, "A fast and accurate module for calculating fractional order derivatives and integrals in *Python*", 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018. doi:10.1109/icccnt.2018.8494055.

[26] M. Adams, "*differint*: A *Python* Package for Numerical Fractional Calculus", arXiv:1912.05303 [cs.MS], December, 2019, doi:10.48550/arXiv.1912.05303.

[27] R. Garrappa, "Numerical evaluation of two and three parameter Mittag-Leffler functions", *SIAM Journal on Numerical Analysis*, 53(3), pp. 1350-1369, 2015, doi:10.1137/140971191.

[28] R. Herrmann, "Fractional calculus: an introduction for physicists". World Scientific Publishing, 2018.

[29] S. Chapra, "Numerical methods for engineers". Mcgraw-hill, 2010.

[30] R. Hamming, "Numerical methods for scientists and engineers". Courier Corporation, 2012.

[31] E. Isaacson and H. B. Keller, "Analysis of numerical methods". Dover Publications, 1994.

[32] P. J. Davis and P. Rabinowitz, "Methods of numerical integration". Dover Publications, 2007.

[33] Scipy, Disponible en https://scipy.org/ (accessed March 7, 2024).

[34] QUADPACK, Disponible en https://en.wikipedia.org /wiki/QUADPACK (accessed March 7, 2024)

[35] C. Li and F. Zeng. "Numerical methods for fractional calculus". CRC Press, 2015.

[36] T. A. M. Langlands, and B. I. Henry. "The accuracy and stability of an implicit solution method for the fractional diffusion equation." Journal of Computational Physics 205.2, pp 719-736, 2005, doi: 10.1016/j.jcp.2004.11.025

[37] F. Zeng and C. "A new Crank-Nicolson finite element method for the time-fractional subdiffusion equation". Applied Numerical Mathematics 121, pp: 82-95, doi: 10.1016/j.apnum.2017.06.011

[38] Saeed. Fractional calculus 03 Riemann Liouville fractional integral dr saeed, YouTube. Disponible en https://www.youtube.com/watch?v=lHMScG219P4&list=RDCMU COjjhlMi0O2WSIPEn9TDZ5Q&index=2 (Accessed: March 7 2024).

[39] K. Diethelm and N. J. Ford, "Analysis of fractional differential equations", Journal of Mathematical Analysis and Applications, 265(2), pp. 229-248, 2002, doi:10.1006/jmaa.2000.7194.

[40] D. Baleanu, K. Diethelm, E. Scalas and J. J. Trujillo. "Fractional calculus: models and numerical methods". Vol. 3. World Scientific, 2012.

[41] K. Khinsen. Khinsen/Mittag-Leffler: The generalized mittag-leffler in *Python*, GitHub. Disponible en https://github.com/khinsen/mittag-leffler (Accessed: March 7 2024).

[42] K. Diethelm, N. J. Ford, A. D. Freed, and Y. Luchko, "Algorithms for the fractional calculus: A selection of numerical methods", Computer Methods in Applied Mechanics and Engineering, vol. 194, no. 6-8, pp. 743-773, Feb. 2005, doi: 10.1016/j.cma.2004.06.006.

[43] K. Diethelm and A. D. Freed, "On the Solution of Nonlinear Fractional-Order Differential Equations Used in the Modeling of Viscoplasticity", in Springer eBooks, 1999, pp. 217-224. doi: 10.1007/978-3-642-60185-9_24.

[44] P. J. Torvik and R. L. Bagley (1984) “On the appearance of the fractional derivative in the behavior of real materials”, Journal of Applied Mechanics, 51(2), pp. 294-298, Jun. 1984, doi:10.1115/1.3167615.

[45] K. Diethelm and J. Ford, “Numerical solution of the Bagley-Torvik equation”, BIT Numerical Mathematics, vol. 42, no. 3, pp. 490-507, 2002. doi:10.1023/a:1021973025166.

[46] M.G. Sakar, O. Saldir and A. Akgül, A, “Novel Technique for Fractional Bagley-Torvik Equation”. Proc. Natl. Acad. Sci., India, Sect. A Phys. Sci. 89, pp. 539-545, 2019. doi:10.1007/s40010-018-0488-4.

[47] “Build software better, together”, GitHub. Disponible en https://github.com/topics/fractional-calculus (accessed March 7 2024).