1 Basic FDE solvers with $0 < \alpha < 1$

This section presents numerical *explicit* solvers for fractional-order ordinary differential equations (FDEs) with order $0 < \alpha < 1$. For more general cases and higher-accuracy discretization schemes, see the subsequent sections.

The corresponding implementations are available in `caputo_solver.py` and `riemann_liouville_solver.py` at: `https://github.com/kangqiyu/torchfde/`.

0. **formulation:** We consider the following initial value problem for a fractional differential equation:

$$D^{\alpha} y(t) = f(t, y(t)), \quad y(0) = y_0. \tag{1.1}$$

where

- D^{α} denotes the fractional derivative operator (either Caputo or Riemann-Liouville (RL) type)
- $0 < \alpha < 1$ is the fractional order
- $f : [0, T] \times \mathbb{R} \to \mathbb{R}$ is the dynamic function
- y_0 is the given initial condition.
- We discretize the problem on a uniform temporal grid $\{t_j = jh : j = 0, 1, \ldots, N\}$, where $h = T/N$.

1. **fractional ABM predictor [1]:** The numerical method is based on the Volterra integral representation of (1.1):

$$y(t) = y_0 + \frac{1}{\Gamma(\alpha)} \int_0^t (t - \tau)^{\alpha - 1} f(\tau, y(\tau)) d\tau \tag{1.2}$$

Here D^{α} takes the Caputo type. The fractional Adams-Bashforth predictor is given by the following iterations:

$$y_{k+1} = y_0 + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{k} b_{j,k+1} f\left(t_j, y_j\right) \tag{1.3}$$

with weights

$$b_{j,k+1} = \frac{h^{\alpha}}{\alpha} \left[(k + 1 - j)^{\alpha} - (k - j)^{\alpha}\right], \quad j = 0, 1, \ldots, k. \tag{1.4}$$

- **short memory version:** We truncate the convolution tail to a memory length M and get:

$$y_{k+1} = y_0 + \frac{1}{\Gamma(\alpha)} \sum_{j=\max\{0, k-M+1\}}^{k} b_{j,k+1} f\left(t_j, y_j\right) \tag{1.5}$$

2. **L1 solver**: Here D^{α} takes the Caputo type. From [2] and **??**, we have the following approximate

$$D^{\alpha} y\left(t_{k+1}\right) \approx \frac{h^{-\alpha}}{\Gamma(2 - \alpha)} \left(y_{k+1} + \sum_{j=1}^{k} \left[(k - j + 2)^{1-\alpha} - 2(k - j + 1)^{1-\alpha} + (k - j)^{1-\alpha}\right] y_j - \left[(k + 1)^{1-\alpha} - k^{1-\alpha}\right] y_0 \right).$$

We have the following iteration:

$$y_{k+1} = h^{\alpha} \Gamma(2 - \alpha) f\left(t_k, y_k\right) - \sum_{j=0}^{k} c_j^{(k)} y_j \tag{1.6}$$

where

$$c_j^{(k)} = \begin{cases} -\left[(k + 1)^{1-\alpha} - k^{1-\alpha}\right] & \text{if } j = 0 \\ (k - j + 2)^{1-\alpha} - 2(k - j + 1)^{1-\alpha} + (k - j)^{1-\alpha} & \text{if } 1 \le j \le k \end{cases} \tag{1.7}$$

- **short memory version:** We truncate the convolution tail to a memory length M and get:

$$y_{k+1} = h^{\alpha} \Gamma(2 - \alpha) f\left(t_k, y_k\right) - \sum_{j=\max\{0, k-M+1\}}^{k} c_j^{(k)} y_j \tag{1.8}$$

3. **GL solver [3]:** For this method, the fractional operator D^α refers to the RL fractional derivative, which we approximate using the Grünwald-Letnikov (GL) finite difference scheme:

$$D^\alpha g(t) = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{j=0}^{\left\lfloor \frac{t}{h} \right\rfloor} (-1)^j \binom{\alpha}{j} g(t-jh), \tag{1.9}$$

The GL weights can be computed efficiently through the following recursive relation:

$$c_0^{(\alpha)} = 1, \quad c_j^{(\alpha)} = \left(1 - \frac{1+\alpha}{j}\right) c_{j-1}^{(\alpha)} \quad (j \geq 1) \tag{1.10}$$

so that $c_j^{(\alpha)} = (-1)^j \binom{\alpha}{j}$. Approximating $D^\alpha y\left(t_k\right)$ by the GL sum gives

$$\sum_{j=0}^{k} c_j^{(\alpha)} y_{k-j} = h^\alpha f\left(t_{k-1}, y_{k-1}\right).$$

Solving for y_k (and using $c_0^{(\alpha)} = 1$) yields the itertation:

$$y_k = h^\alpha f\left(t_{k-1}, y_{k-1}\right) - \sum_{j=1}^{k} c_j^{(\alpha)} y_{k-j}, \quad k = 1, 2, \ldots$$

Equivalently, to align with the index using in (1.4), we use

$$y_{k+1} = h^\alpha f\left(t_k, y_k\right) - \sum_{j=1}^{k+1} c_j^{(\alpha)} y_{k+1-j}, \quad k = 0, 1, \ldots \tag{1.11}$$

- **short memory version:** We truncate the convolution tail to a memory length M and get:

$$\begin{aligned} y_{k+1} &= h^\alpha f\left(t_k, y_k\right) - \sum_{j=1}^{\min\{k+1, M\}} c_j^{(\alpha)} y_{k+1-j} \\ &= h^\alpha f\left(t_k, y_k\right) - \sum_{j=\max\{0, k+1-M\}}^{k} c_{k+1-j}^{(\alpha)} y_j. \end{aligned} \tag{1.12}$$

4. **Product Trap solver:** For this method [4], the fractional operator D^α refers to the RL fractional derivative, which we approximate using the product trapezoidal method:

$$D^\alpha g\left(x_k\right) \approx \frac{h^{-\alpha}}{\Gamma(2-\alpha)} \sum_{j=0}^{k} A_{j,k} g\left(x_j\right)$$

with

$$A_{j,k} = \begin{cases} \left((k-1)^{1-\alpha} - (k+\alpha-1)k^{-\alpha}\right) & \text{if } j = 0 \\ (k-j+1)^{1-\alpha} + (k-j-1)^{1-\alpha} - 2(k-j)^{1-\alpha} & \text{if } 1 \leq j \leq k-1 \\ 1 & \text{if } j = k \end{cases}$$

This leads to an algorithm of the form

$$\frac{h^{-\alpha}}{\Gamma(2-\alpha)} \sum_{j=0}^{k} A_{j,k} y_j = f\left(x_{k-1}, y_{k-1}\right) \quad (k = 1, 2, \ldots, N)$$

Since $A_{k,k} = 1$, we may rewrite this identity in the form

$$y_k = \Gamma(2-\alpha) h^\alpha f\left(x_{k-1}, y_{k-1}\right) - \sum_{j=0}^{k-1} A_{j,k} y_j$$

Equivalently, to align with the index using in (1.4), we use

$$y_{k+1} = \Gamma(2-\alpha) h^\alpha f\left(x_k, y_k\right) - \sum_{j=0}^{k} A_{j,k+1} y_j \tag{1.13}$$

- **short memory version:** We truncate the convolution tail to a memory length M and get:

$$y_{k+1} = \Gamma(2-\alpha)h^{\alpha} f\left(x_k, y_k\right) - \sum_{j=\max\{0,k+1-M\}}^{k} A_{j,k+1} y_j \tag{1.14}$$

2 Distributed FDE Solver

We consider the distributed-order system governed by the following integro-differential equation:

$$\int_a^b D^{\alpha} y(t)\, \mathrm{d}\mu(\alpha) = f(t, y(t)), \tag{2.1}$$

where μ represents a measure over the fractional order α (which may be parameterized as a learnable distribution). A standard numerical approach approximates this integral using a quadrature rule, resulting in a multi-term fractional differential equation (FDE):

$$\sum_{j=0}^{n} w_j D^{\alpha_j} y(t) = f(t, y(t)), \tag{2.2}$$

subject to the initial condition $y(0) = y_0$. Here, w_j denotes the discretization weights (coefficients) and $\alpha_j \in (0, 1)$ represents the discrete fractional orders.

We work on a uniform grid $t_k = kh$ with step size $h > 0$. For each fractional order α_j we approximate the RL derivative at t_k by the GL formula

$$D^{\alpha_j} y(t_k) \approx h^{-\alpha_j} \sum_{m=0}^{k} c_m^{(\alpha_j)}\, y_{k-m}, \qquad k = 1, \ldots, N. \tag{2.3}$$

Substituting this into the multi-term FDE

$$\sum_{j=0}^{n} w_j D^{\alpha_j} y(t) = f(t, y(t)),$$

and evaluating at t_k with an explicit right-hand side $f(t_{k-1}, y_{k-1})$ gives

$$\sum_{j=0}^{n} w_j h^{-\alpha_j} \sum_{m=0}^{k} c_m^{(\alpha_j)} y_{k-m} = f(t_{k-1}, y_{k-1}), \qquad k = 1, \ldots, N. \tag{2.4}$$

Interchanging the sums, we obtain a single convolution with *distributed GL weights*

$$\sum_{m=0}^{k} \tilde{c}_m\, y_{k-m} = f(t_{k-1}, y_{k-1}), \qquad \tilde{c}_m := \sum_{j=0}^{n} w_j h^{-\alpha_j} c_m^{(\alpha_j)}, \quad m \geq 0. \tag{2.5}$$

Using $c_0^{(\alpha_j)} = 1$ for all j from (1.10), we have

$$\tilde{c}_0 = \sum_{j=0}^{n} w_j h^{-\alpha_j} > 0, \tag{2.6}$$

so we can solve for y_k as

$$y_k = \frac{1}{\tilde{c}_0} \left(f(t_{k-1}, y_{k-1}) - \sum_{m=1}^{k} \tilde{c}_m\, y_{k-m} \right), \qquad k = 1, 2, \ldots, N. \tag{2.7}$$

Equivalently, reindexing to align with the notation in (1.4), we write

$$y_{k+1} = \frac{1}{\tilde{c}_0} \left(f(t_k, y_k) - \sum_{m=1}^{k+1} \tilde{c}_m\, y_{k+1-m} \right), \qquad k = 0, 1, \ldots, N-1. \tag{2.8}$$

In particular, when μ collapses to a single atom (i.e., $n = 0$, $w_0 = 1$, $\alpha_0 = \alpha$), we have $\tilde{c}_m = h^{-\alpha} c_m^{(\alpha)}$ and (2.8) reduces to the single-order GL scheme in the previous paragraph (up to the trivial scaling by $\tilde{c}_0 = h^{-\alpha}$).

- **short memory version:** As in the single-order case, we may truncate the convolution to a finite memory length M and obtain a short-memory distributed GL solver. For $k+1 \le M$, (2.8) is unchanged. For $k+1 > M$, we keep only the M most recent history terms, which yields

$$y_{k+1} = \frac{1}{\tilde{c}_0}\left(f(t_k, y_k) - \sum_{m=1}^{\min\{k+1,M\}} \tilde{c}_m \, y_{k+1-m} \right) \tag{2.9}$$

$$= \frac{1}{\tilde{c}_0}\left(f(t_k, y_k) - \sum_{j=\max\{0,k+1-M\}}^{k} \tilde{c}_{k+1-j} \, y_j \right), \qquad k = 0, 1, \ldots, N-1, \tag{2.10}$$

which mirrors the short-memory GL scheme but with the aggregated weights $\{\tilde{c}_m\}_{m\ge 0}$ encoding the distributed-order measure μ through the quadrature coefficients $\{(w_j, \alpha_j)\}_{j=0}^{n}$.

References

[1] K. Diethelm, N. J. Ford, and A. D. Freed, "Detailed error analysis for a fractional adams method," *Numerical algorithms*, vol. 36, no. 1, pp. 31–52, 2004.

[2] G.-h. Gao and Z.-z. Sun, "A compact finite difference scheme for the fractional sub-diffusion equations," *Journal of Computational Physics*, vol. 230, no. 3, pp. 586–595, 2011.

[3] O. Tacha, J. Munoz-Pacheco, E. Zambrano-Serrano, I. Stouboulos, and V.-T. Pham, "Determining the chaotic behavior in a fractional-order finance system with negative parameters," *Nonlinear Dynamics*, vol. 94, no. 2, pp. 1303–1317, 2018.

[4] D. Baleanu, K. Diethelm, E. Scalas, and J. J. Trujillo, *Fractional calculus: models and numerical methods*. World Scientific, 2012, vol. 3.