The code `flmm2`

This code is described in [1], and is available at the URL [2]. It implements three implicit methods for approximating the solution of the problem (for sake of brevity, we omit t as an argument of the vector field):

$$y^{(\alpha)} = f(y), \qquad t \in [0,T], \qquad y^{(i)}(0) = y_0^i \in \mathbb{R}^m, \qquad i = 0, \dots, \lceil \alpha \rceil - 1,$$

at the grid points $t_j = jh$, $j = 0, 1, \dots, N$, with $h = T/N$ the used timestep. More in detail, by setting, as is usual, $y_j \approx y(t_j)$ and $f_j = f(y_j)$, one then obtains that the methods implemented in the code are in the form [3]:

$$y_n = T_\alpha(t_n) + h^\alpha \sum_{j=0}^{s} w_{n,j} f_j + h^\alpha \sum_{j=0}^{n} \omega_{n-j} f_j, \qquad n = 1, \dots, N,$$

where the coefficients $\{w_{n,j}\}$ of the *correction term*,

$$h^\alpha \sum_{j=0}^{s} w_{n,j} f_j,$$

are defined in order for the approximation of the Riemann-Liouville integral be exact for the functions

$$t^\nu, \qquad \nu \in \{\nu = i + j\alpha < 1,\, i,j \in \mathbb{N}\} \cup \{1\}.$$

Consequently, at each timestep, if the above allowed values of ν are $s+1$, the weights $\{w_{n,j}\}$ are obtained by solving the linear system of equations:

$$\sum_{j=0}^{s} w_{n,j} j^\nu = -\sum_{j=0}^{n} \omega_{n-j} j^\nu + \frac{\Gamma(\nu+1)}{\Gamma(1+\nu+\alpha)} n^{\nu+\alpha},$$

Instead, the coefficients ω_j, $j = 0, \dots, n$, are the first $n+1$ coefficients of the formal power series

$$\omega_\alpha(\xi) \equiv \sum_{j \ge 0} \omega_j \xi^j,$$

with $\omega_\alpha(\xi)$ depending on the chosen method. In particular [1,3]:

- $\omega_\alpha(\xi) = (1+\xi)^\alpha (2(1-\xi))^{-\alpha}$ gives the coefficients of the fractional trapezoidal rule;
- $\omega_\alpha(\xi) = (1-\xi)^{-\alpha}\left(1 - \frac{\alpha}{2}(1-\xi)\right)$ gives a so called fractional Newton-Gregory formula;
- $\omega_\alpha(\xi) = 2^\alpha (3 - 4\xi + \xi^2)^{-\alpha}$ gives the fractional BDF2 method.

It is worth noticing that, according to the general procedure given in [3], in the case of the fractional trapezoidal and BDF2 methods,

$$\omega_\alpha(\xi) = \left(\frac{\sigma(\xi^{-1})}{\rho(\xi^{-1})} \right)^\alpha,$$

with $\rho(z)$ and $\sigma(z)$ the first and second characteristic polynomials of the corresponding linear multistep method used in the ODE case. Moreover, the convolution products are evaluated via a fast FFT algorithm [4].

As is clear, all the above methods are implicit, since at the nth time-step y_n is obtained by solving a nonlinear system in the form

$$y_n = g_n + h^\alpha \omega_0 f(y_n),$$

with g_n a known vector. The above nonlinear system is solved by means of a straight Newton method, with a safeguard on the maximum number of iterations.

The calling sequence of the function `flmm2` is:

```
[t,y] = flmm2(alpha,fdefun,Jdefun,t0,tfinal,y0,h,param,method,tol,itmax)
```

In output `t and y` contain the computed solution (`y` is stored by columns), whereas, in input:

- `alpha` is the order α of the derivative;
- `fdefun` is the identifier of the function evaluating the vector field (`fdefun(t,y)`) which returns a column vector;
- `Jfdefun` is the identifier of the function evaluating the Jacobian (`Jfdefun(t,y)`);
- `[t0,tfinal]` is the integration interval;
- `y0` is a matrix of dimension $m \times \lceil \alpha \rceil$ with the initial conditions;
- `h` is the used stepsize, assumed constant;
- `param` contains possible parameters needed by `fdefun` and `Jfdefun`;
- `method` denotes the used fractional linear multistep method: 1 for the trapezoidal rule; 2 for the Newton-Gregory formula; 3 for the BDF2;
- `tol` is the tolerance for the convergence of the Newton iteration (the default value is 10^{-6});
- `itmax` is the maximum number of Newton iterations (the default value is 100).

[1] Garrappa, R. Trapezoidal methods for fractional differential equations: theoretical and computational aspects. *Math. Comput. Simulation* **2015**, *110*, 96–112. https://doi.org/10.1016/j.matcom.2013.09.012

[2] http://www.mathworks.com/matlabcentral/fileexchange/47081-flmm2 (accessed on March 24, 2025).

[3] Lubich, Ch. Discretized fractional calculus. *SIAM J. Math. Anal.* **1986**, *17(3)*, 704–719. https://epubs.siam.org/doi/pdf/10.1137/0517050

[4] Hairer, E.; Lubich, Ch.; Schlichte, S. Fast numerical solution of nonlinear Volterra convolution equations *SIAM J. Sci. Statist. Comput.* **1985**, *6*, 532–541. https://doi.org/10.1137/0906037