The code `fcoll`

This code is described and published in [1]. It relies on a collocation method for solving scalar problems only:

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

In more details, given the mesh

$$t_0 = 0, \qquad t_j = t_{j-1} + h_j, \qquad j = 1,\ldots,N,$$

and q collocation points

$$t_{ji} \equiv t_{i-1} + \eta_i h_j \in [t_{j-1}, t_j], \qquad i = 1,\ldots,q,$$

the used method looks for approximations

$$z_{ji} \approx D^{(\alpha)}y(t_{ji}), \qquad i = 1,\ldots,q,$$

such that, by setting

$$\sum_{i=1}^{q} z_{ji}\ell_i^j(t), \qquad \ell_i^j(t) = \prod_{\nu=1,\nu\neq i}^{q} \frac{t - t_{j\nu}}{t_{ji} - t_{j\nu}}, \qquad t \in [t_{j-1}, t_j], \qquad j = 1,\ldots,N,$$

the interpolating polynomial of degree $q-1$, one has:

$$z_{jk} = f\left(t_{jk},\, \phi_j(t_{jk}) + \sum_{i=1}^{q} z_{ji} I^{\alpha}\ell_i^j(t_{jk})\right), \qquad k = 1,\ldots,q,$$

with $I^{\alpha}\ell_i^j(t)$ the Riemann-Liouville integral of $\ell_i^j(t)$, and the *memory term* given by

$$\phi_j(t) \;:=\; \sum_{i=0}^{\lceil\alpha\rceil-1} \frac{t^i}{i!} y_0^i \;+\; \sum_{\mu=1}^{j-1}\sum_{i=1}^{q} z_{\mu i} I^{\alpha}\ell_i^{\mu}(t), \qquad t \in [t_{j-1}, t_j].$$

The obtained discrete problem is solved via the Matlab© function `fsolve` by using very tiny tolerances. The code implements an efficient computation of the involved Riemann-Liouville integrals of the Lagrange polynomials $\ell_i^j(t)$ which, in turn, allow to effectively solve the local discrete problem, thus providing the approximation at t_j as:

$$y_j \;:=\; \phi_j(t_j) \;+\; \sum_{i=1}^{q} z_{ji} I^{\alpha}\ell_i^j(t_j).$$

Further, in order to ensure a fast error decay, a graded mesh in the form

$$t_j = \left(\frac{j}{N}\right)^r, \qquad j = 0,1,\ldots,N, \qquad \text{with} \quad r \geq 1,$$

may be adopted, in case the vector field is not smooth at the origin. Clearly, when $r = 1$, a uniform mesh is obtained. The calling sequence is:

```
[t,y] = fcoll(f,b,gam,alpha,eta,r,N)
```

In output `t` and `y` contain the computed solution, whereas, in input:

- `f` is the identifier of the function implementing the vector field (`f(t,y)`);
- `b` is the final integration time (the starting time is 0);

- `gam` is a column vector containing the initial conditions;
- `alpha` is the order of the fractional derivative;
- `eta` is a column vector of the normalized collocation abscissae defined above,

$$\eta_i = \frac{t_{ji} - t_{j-1}}{h_j}, \qquad i = 1 \ldots, q;$$

- `r` is the parameter for the graded mesh;
- `N` is the number of mesh points.

[1] Cardone, A; Conte, D.; Paternoster, B. A MATLAB implementation of spline collocation methods for fractional differential equations. In: *Gervasi O. et al. (eds.) Computational Science and Its Applications – ICCSA 2021*. Lecture Notes in Computer Science *12949*,**2021**, pp. 387–401. https://doi.org/10.1007/978-3-030-86653-2_29