The code `fhbvm`

This code is described in [1,2,3], and is available at the URL [4]. The basic idea is that of deriving a local polynomial approximation of the vector field of the problem (for sake of brevity, we have omitted t as a formal argument of f)

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

by expanding it along the orthonormal Jacobi polynomial basis

$$P_j \in \Pi_j, \quad \int_0^1 \omega(x) P_i(x) P_j(x) \mathrm{d}x = \delta_{ij}, \quad i,j = 0,1,\ldots, \quad \omega(x) = \alpha(1-x)^{\alpha-1},$$

with $\alpha > 0$ the order of the fractional derivative. In more details, given a mesh in the form

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

let us denote by

$$y_j(ch_j) := y(t_{j-1} + ch_j), \qquad c \in [0,1], \qquad j = 1, \ldots, N,$$

the restriction of the solution to the sub-interval $[t_{j-1}, t_j]$. Consequently, the problem can be rewritten as the following sequence of local problems:

$$\begin{aligned} y_j^{(\alpha)}(ch_j) &= f(y_j(ch_j)), \qquad c \in [0,1], \qquad j = 1, \ldots, N, \\ y_1^{(i)}(0) &= y_0^i, \qquad i = 0, \ldots, \lceil \alpha \rceil - 1. \end{aligned}$$

By expanding the local vector fields along the Jacobi basis,

$$f(y_j(ch)) = \sum_{\ell \geq 0} P_\ell(c) \gamma_\ell(y_j), \qquad c \in [0,1], \qquad j = 1, \ldots, N,$$

with the Fourier coefficients defined by

$$\gamma_\ell(y_j) = \int_0^1 \omega(\tau) P_\ell(\tau) f(y_j(\tau h_j)) \mathrm{d}\tau, \qquad \ell = 0, 1, \ldots,$$

one then obtains the following equivalent formulation:

$$\begin{aligned} y_j^{(\alpha)}(ch_j) &= \sum_{\ell \geq 0} P_\ell(c) \gamma_\ell(y_j), \qquad c \in [0,1], \qquad j = 1, \ldots, N, \\ y_1^{(i)}(0) &= y_0^i, \qquad i = 0, \ldots, \lceil \alpha \rceil - 1. \end{aligned}$$

Consequently, for $c \in [0,1]$, so that $t = t_{n-1} + ch_n \in [t_{n-1}, t_n]$, one obtains

$$y(t) \equiv y_n(ch_n) = \Psi_n(ch_n) + h_n^\alpha \sum_{\ell \geq 0} I^\alpha P_\ell(c) \gamma_\ell(y_n), \qquad c \in [0,1],$$

with $I^\alpha P_\ell(c)$ the Riemann-Liouville integral of $P_\ell(c)$ and, by setting

$$T_\alpha(t) = \sum_{i=0}^{\lceil \alpha \rceil - 1} \frac{t^i}{i!} y_0^i,$$

the *memory term* formally given by:

$$\Psi_n(ch_n) = T_\alpha(t_{n-1} + ch_n) + \sum_{j=1}^{n-1} h_j^\alpha \sum_{\ell \geq 0} J_\ell^\alpha \left(\frac{t_{n-1} - t_{j-1} + ch_n}{h_j} \right) \gamma_\ell(y_j), \quad c \in [0,1],$$

having set

$$J_\ell^\alpha(x) := \frac{1}{\Gamma(\alpha)} \int_0^1 (x-\tau)^{\alpha-1} P_\ell(\tau) \mathrm{d}\tau, \qquad x \geq 1.$$

A piecewise approximation

$$\sigma_j(ch_j) \approx y_j(ch_j), \qquad c \in [0,1], \qquad j = 1, \ldots, N,$$

can be obtained by truncating the infinite expansions of the local vector fields to polynomials of degree $s-1$:

$$\begin{aligned} \sigma_j^{(\alpha)}(ch_j) &= \sum_{\ell=0}^{s-1} P_\ell(c) \gamma_\ell(\sigma_j), \qquad c \in [0,1], \qquad j = 1, \ldots, N, \\ \sigma_1^{(i)}(0) &= y_0^j, \qquad i = 0, \ldots, \lceil \alpha \rceil - 1, \end{aligned}$$

with the Fourier coefficients $\gamma_\ell(\sigma_j)$ defined as above, by formally replacing y_j with σ_j. Consequently, for $t = t_{n-1} + ch_n \in [t_{n-1}, t_n]$, one obtains

$$\sigma_n(ch_n) = \Psi_n^s(ch_n) + h_n^\alpha \sum_{\ell=0}^{s-1} I^\alpha P_\ell(c) \gamma_\ell(\sigma_n), \qquad c \in [0,1],$$

with the memory term now given by

$$\Psi_n^s(ch_n) = T_\alpha(t_{n-1} + ch_n) + \sum_{j=1}^{n-1} h_j^\alpha \sum_{\ell=0}^{s-1} J_\ell^\alpha \left(\frac{t_{n-1} - t_{j-1} + ch_n}{h_j} \right) \gamma_\ell(\sigma_j), \quad c \in [0,1].$$

Next, in order to obtain a practical numerical method, the Fourier coefficients $\gamma_\ell(\sigma_j)$ are approximated by means of an interpolatory Gauss-Jacobi quadrature of order $2k$, with abscissae given by the zeros of the kth Jacobi polynomial, for a convenient $k \geq s$, $P_k(c_i) = 0$, $i = 1, \ldots, k$, and corresponding weights $b_1, \ldots, b_k$:

$$\gamma_\ell(\sigma_j) \approx \sum_{i=1}^{k} b_i P_\ell(c_i) f(\sigma_j(c_i h_j)) =: \hat{\gamma}_\ell^j, \qquad \ell = 0, \ldots, s-1, \quad j = 1, \ldots, N,$$

where, for the sake of brevity, we continue denoting σ_j the approximation of the solution in $[t_{j-1}, t_j]$, after the discretization of the Fourier coefficients. In so doing, one obtains a FHBVM(k,s) method. A noticeable feature of such methods (common to HBVMs, obtained for $\alpha = 1$ [5,6]), is that the discrete problem can be cast in terms of the s Fourier coefficients in the current sub-interval, independently of k. In fact, from the above equations one derives:

$$\hat{\gamma}_\ell^n = \sum_{i=1}^{k} b_i P_\ell(c_i) f\left(\Psi_n^s(c_i h_n) + h_n^\alpha \sum_{r=0}^{s-1} I^\alpha P_r(c_i) \hat{\gamma}_r^n \right), \qquad \ell = 0, \ldots, s-1,$$

which can be efficiently solved by means of a Newton-type iteration [2].

The code `fhbvm` implements a FHBVM(22,20) method. Moreover, in order to have manageable integrals involved in its implementation, it uses either a uniform mesh,

$$t_j = jh, \qquad j = 0, \dots, N, \qquad h = \frac{T}{N},$$

or a graded one, where, for a convenient $r > 1$,

$$t_0 = 0, \quad t_j = t_{j-1} + h_j, \quad h_j = r^{j-1} h_1, \qquad j = 1, \dots, N, \qquad h_1 \frac{r^N - 1}{r - 1} = T.$$

In fact, in such a case the argument of J_ℓ^α in the memory term becomes:

$$\frac{t_{n-1} - t_{j-1} + ch_n}{h_j} = \begin{cases} n - j + c, & \text{for the uniform mesh,} \\ \frac{r^{n-j}-1}{r-1} + cr^{n-j}, & \text{for the graded mesh.} \end{cases}$$

The code automatically detectes which mesh is needed, depending on the smoothness of the vector field at the initial point, also choosing the parameters N, r, and h_1. Moreover, also the smoothness of the vector field is automatically detected, by means of a trial and error procedure.

Remark 1. *It is worth noticing that the local truncated vector fields are equivalent to requiring that the corresponding residuals, $r_j(ch_j) := \sigma_j^{(\alpha)}(ch_j) - f(\sigma_j(ch_j))$, $c \in [0,1]$, be orthogonal, w.r.t. the weighted product defining the Jacobi polynomials. I.e., for all $j = 1, \dots, N$:*

$$\int_0^1 \omega(c) P_\ell(c) r_j(ch_j) \mathrm{d}c = 0, \qquad \ell = 0, \dots, s-1.$$

In consideration of the high value of s ($s = 20$) implemented in the code `fhbvm`, one expects a spectral accuracy in time for the numerical solution [1], similarly to what happens in the ODE case for HBVMs [7].

The calling sequence of the code `fhbvm` is:

```
[t,y,stats,err] = fhbvm( fun, y0, T, M )
```

In input:

- `fun` is the identifier of a function computing:
  - the vector field (`fun(t,y)`) also in vector mode, returning row vectors;
  - the Jacobian (`fun(t,y,1)`);
  - the order α of the fractional derivative, if called without arguments;
- `y0` is a matrix of dimension $\lceil \alpha \rceil \times m$ containing the initial conditions;
- `T` is the width of the integration interval (the initial point being set to 0);
- `M` is an optional parameter such that $h =$ `T/M` is the constant timestep, if a uniform mesh is selected, or $h_N \approx$ `T/M` is the final timestep, if the graded mesh is used. In this latter case, the parameters N, r, h_1 are automatically computed.

It is worth noticing that this code requires very few parameters be given by the user in input. In output:

- `t,y` contain the computed solution (`y` is stored by rows);
- `stats` is an optional vector containing the following four times:
  - the pre-processing time;
  - the execution time;
  - the pre-processing time for the error estimation;

  - the execution time for the error estimation.

  Consequently, the total execution time is given by their sum;
- `err` is an optional vector containing the estimated error on a doubled mesh (this is an expensive procedure, which is done only if this parameter is specified in output).

[1] Brugnano, L.; Burrage, K.; Burrage, P.; Iavernaro F. A spectrally accurate step-by-step method for the numerical solution of fractional differential equations. *J. Sci. Comput.* **2024**, *99*, 48. https://doi.org/10.1007/s10915-024-02517-1

[2] Brugnano, L.; Gurioli, G.; Iavernaro, F. Numerical solution of FDE-IVPs by using Fractional HBVMs: the fhbvm code. *Numer. Algoritms* **2025**, *99*, 463–489. https://doi.org/10.1007/s11075-024-01884-y

[3] Brugnano, L.; Gurioli, G.; Iavernaro, F. Solving FDE-IVPs by using Fractional HBVMs: Some experiments with the fhbvm code. *J. Comput. Methods Sci. Eng.* **2025**. https://doi.org/10.1177/14727978251321328

[4] https://people.dimai.unifi.it/brugnano/fhbvm/ (accessed on April 10, 2025).

[5] Brugnano, L.; Iavernaro, F. *Line Integral Methods for Conservative Problems.* Chapman et Hall/CRC, Boca Raton, FL, 2016.

[6] Brugnano, L.; Iavernaro, F. Line integral solution of differential problems. *Axioms* **2018**, *7(2)*, 36. https://doi.org/10.3390/axioms7020036

[7] Amodio, P.; Brugnano, L.; Iavernaro, F. Analysis of Spectral Hamiltonian Boundary Value Methods (SHBVMs) for the numerical solution of ODE problems. *Numer. Algorithms* **2020**, *83*, 1489–1508. https://doi.org/10.1007/s11075-019-00733-7