Journal of Scientific Computing (2026) 107:94
https://doi.org/10.1007/s10915-026-03315-7

A Multi-Order Extension of Fractional HBVMs (FHBVMs)

Luigi Brugnano[1] · Gianmarco Gurioli[1] · Felice Iavernaro[2] · Mikk Vikerpuur[3]

Received: 4 December 2025 / Revised: 6 March 2026 / Accepted: 27 April 2026

Abstract
The efficient numerical solution of fractional differential equations has been recently tackled through the definition of Fractional HBVMs (FHBVMs), a class of Runge-Kutta type methods. Corresponding Matlab© codes have been also made available on the internet, proving to be very competitive w.r.t. existing ones. However, so far, FHBVMs have been given for solving systems of fractional differential equations with the same order of fractional derivative, whereas the numerical solution of multi-order problems (i.e., problems in which different orders of fractional derivatives occur) has not been handled, yet. Due to their relevance in applications, in this paper we propose an extension of FHBVMs for addressing fractional multi-order problems, providing full details for such an approach. A corresponding Matlab© code, handling the case of two different fractional orders, is also made available, proving very effective for numerically solving these problems.

Keywords Fractional Differential Equations · FDEs · Caputo derivative · Fractional HBVMs · FHBVMs · Fractional multi-order problems.

Mathematics Subject Classification 34A08 · 65R20 · 65-04.

1 Introduction

The idea of extending the definition of derivatives to the case of a non-integer order of differentiation dates back to 1695, to the correspondence between Leibniz and de l'Hôpital [56], when the meaning of the derivative of order one half was discussed. Although its theoretical

✉ Luigi Brugnano
luigi.brugnano@unifi.it

Gianmarco Gurioli
gianmarco.gurioli@unifi.it

Felice Iavernaro
felice.iavernaro@uniba.it

Mikk Vikerpuur
mikk.vikerpuur@ut.ee

1 Dipartimento di Matematica e Informatica "U. Dini", Università di Firenze, Florence, Italy

2 Dipartimento di Matematica, Università di Bari "Aldo Moro", Bari, Italy

3 Institute of Mathematics and Statistics, University of Tartu, Tartu, Estonia

Published online: 09 May 2026

foundation is centuries old, the concept of fractional derivative has recently evolved from a purely mathematical curiosity into a vital tool for modeling across a variety of scientific fields (see, e.g., [37, 39, 47, 51, 53, 58, 59], to mention a few), due to its ability to encompass memory terms. In fact, unless the ODE case, the differential operator here considered is of a nonlocal nature [24, 56]. For this reason, the numerical solution of FDEs has been the subject of many researches in the last years, starting from the pioneering work of Lubich [46], where fractional versions of linear multistep formulae are considered: we refer, e.g., to [25, 26, 30, 31, 33, 42, 55], to mention a few of them. More recently, the class of *Fractional HBVMs (FHBVMs)* [11] has been introduced for the efficient numerical solution of initial value problems of fractional differential equations (FDE-IVPs) in the Caputo form, with one fractional derivative.

Remark 1 The acronym HBVMs stands for *"Hamiltonian Boundary Value Methods"*, a class of energy-conserving Runge-Kutta methods for the efficient numerical solution of Hamiltonian ODE problems (see, e.g., the monograph [16]). As matter of fact, when the fractional order of the equation becomes 1 (i.e., the FDE reduces to an integer-order ODE), FHBVMs do coincide with HBVMs.

The basic steps leading to the definition of FHBVMs are the following ones:

- local expansion of the vector field along a suitable Jacobi polynomial basis, naturally induced by the fractional problem at hand (when the equation is an ordinary one, the Jacobi basis reduces to the Legendre polynomial basis, thus resulting into HBVMs);
- discretization of the corresponding Fourier coefficients, by using a corresponding Gauss-Jacobi quadrature;
- solution of the generated discrete problems, defined on a suitable discrete mesh.

FHBVMs have been efficiently implemented in the two Matlab© codes:

- `fhbvm`, [12, 13] using either a graded or a uniform mesh (depending on the features of the problem at hand);
- `fhbvm2`, [14] possibly also using a mixed mesh (initially graded and subsequently uniform).

An extensive experimentation [15] has proved that the above codes are extremely competitive w.r.t. publicly available ones, based on different numerical methods, due to the fact that the implemented FHBVM methods can achieve, alike in the ODE case [2], a spectrally accurate solution.

Nevertheless, multi-order systems of equations, i.e., systems where different fractional orders appear, are of interest in a number of applications, including material science [56, 60]), population dynamics [5], epidemic modelling [19, 43], pharmacokinetics [63], and chaotic dynamics [7, 36, 44, 57]. In this respect, the extension of FHBVMs for such problems poses important challenges. To mention a few of them: the handling of different Jacobi quadratures; a more involved structure and solution of the generated discrete problems; a more involved theoretical analysis of the methods. Their study is precisely the goal of this paper.

Consequently, we shall hereafter consider multi-order problems in the following general form, for $\alpha_i \in (\ell-1, \ell)$, $i = 1 \dots, \nu$, $\ell \in \mathbb{N}$, $\ell \geq 1$:

$$\begin{aligned} y_i^{(\alpha_i)}(t) &= f_i(y(t)), \qquad t \in [0, T], \\ y_i^{(\iota)}(0) &= y_{i0}^{\iota} \in \mathbb{R}^{m_i}, \qquad \iota = 0, \dots, \ell-1, \qquad i = 1, \dots, \nu, \\ y &= (y_1^\top, \dots, y_\nu^\top)^\top \in \mathbb{R}^m, \qquad m = \sum_{i=1}^{\nu} m_i. \end{aligned} \tag{1}$$

Here, for all $i = 1, \ldots, \nu$, assuming $y_i \in A^\ell([0, T])$ (i.e., $y_i^{(\ell-1)}$ is absolutely continuous),

$$y_i^{(\alpha_i)}(t) := \frac{1}{\Gamma(\ell - \alpha_i)} \int_0^t (t - x)^{\ell - \alpha_i - 1} y_i^{(\ell)}(x) \mathrm{d}x \tag{2}$$

is the Caputo derivative of y_i and, for the sake of brevity, in (1) we have omitted t as a formal argument of f_i. The approach described in the following sections reduces to the usual one defining FHBVMs, when the $\{\alpha_i\}$ are all equal (or, equivalently, $\nu = 1$), but differs significantly both in the metodology and the implementations strategies, when they are different from each other.

While the numerical solution of linear or nonlinear multi-term FDEs [1] involving Caputo-type fractional derivatives has, over the years, attracted considerable attention by a large number of authors, the general multi-order problem in the form (1), has received significantly less study. Nevertheless, we can point the reader interested in the numerical analysis of such problems to the Adomian decomposition and the variational iteration method studied in [50], the differential transform technique used in [28], the Chebyshev matrix approach implemented in [6, 22], the application of the fractional Jacobi-Picard iteration method in [4], wavelet-type solutions in [20, 64], and the trapezoidal and product integration methods in [33]. We also note that there exist some prior numerical methods implementing a spectral-type approach, with basis functions consisting of fractional-order Jacobi functions [9, 29] or Chelyshkov polynomials [1].

With this premise, the structure of the paper is as follows. At first, in Section 2 we explain the possible approaches for extending FHBVMs to handle problem (1). Later on, in Sections 3 and 4, we provide full details for the corresponding discretization procedure, emphasizing the major difference in the choice of the discrete abscissae and quadratures. The practical solution of the generated discrete problems is then detailed in Section 5, and the stability and accuracy analysis of the methods is given in Section 6. A new version of the code `fhbvm2` in [14], named `fhbvm2_2`, is then presented in Section 7, handling at the moment the case $\nu = 2$. This latter code is later used in the numerical tests reported in Section 8. At last, some conclusions are given in Section 9.

2 Fractional HBVMs (FHBVMs)

To begin with, we recall that, under suitable regularity assumptions on the vector fields f_i, the solution of problem (1) can be formally written (see, e.g., [27]) as:

$$\begin{aligned} y_i(t) &= \sum_{\iota=0}^{\ell-1} \frac{y_{i0}^{\iota}}{\iota!} t^\iota + \frac{1}{\Gamma(\alpha_i)} \int_0^t (t - x)^{\alpha_i - 1} y_i^{(\alpha_i)}(x) \mathrm{d}x \\ &= \sum_{\iota=0}^{\ell-1} \frac{y_{i0}^{\iota}}{\iota!} t^\iota + \frac{1}{\Gamma(\alpha_i)} \int_0^t (t - x)^{\alpha_i - 1} f_i(y(x)) \mathrm{d}x \\ &\equiv T_{i\ell}(t) + I^{\alpha_i} f_i(y(t)), \qquad t \in [0, T], \qquad i = 1, \ldots, \nu, \end{aligned} \tag{3}$$

with y defined according to (1), and

$$I^{\alpha_i} f_i(y(t)) = \frac{1}{\Gamma(\alpha_i)} \int_0^t (t - x)^{\alpha_i - 1} f_i(y(x)) \mathrm{d}x$$

[1] I.e., FDEs with different fractional derivatives appearing in the same equation.

the Riemann-Liouville integral corresponding to (2).

Let us now consider, for explanation purposes, the case where problem (1) is, at first, solved over the reference interval $[0, h]$. To this end, we use the families of Jacobi polynomials $\{P_j^i\}_{j\geq 0}$, $i = 1, \ldots, \nu$, respectively orthonormal w.r.t. the weighting functions:

$$\omega_i(c) = \alpha_i(1-c)^{\alpha_i - 1}, \quad c \in [0,1], \quad \Longrightarrow \quad \int_0^1 \omega_i(c) P_r^i(c) P_j^i(c) \mathrm{d}c = \delta_{rj}, \\ r, j = 0, 1, \ldots, \tag{4}$$

with, as is usual, δ_{rj} denoting the Kronecker symbol. Next, we expand each vector field f_i in (1) along the corresponding family of polynomials, thus obtaining the equivalent problem

$$y_i^{(\alpha_i)}(ch) = f_i(y(ch)) \equiv \sum_{j\geq 0} P_j^i(c)\gamma_{ij}(y), \qquad c \in [0,1], \\ y_i^{(\iota)}(0) = y_{i0}^{\iota} \in \mathbb{R}^{m_i}, \qquad \iota = 0, \ldots, \ell - 1, \qquad i = 1, \ldots, \nu, \tag{5}$$

with the corresponding Fourier coefficients defined as

$$\gamma_{ij}(y) = \int_0^1 \omega_i(\tau) P_j^i(\tau) f_i(y(\tau h)) \mathrm{d}\tau, \qquad j = 0, 1, \ldots. \tag{6}$$

Integrating (5) side by side, and imposing the initial conditions, one then obtains

$$y_i(ch) = T_{i\ell}(ch) + h^{\alpha_i} \sum_{j\geq 0} I^{\alpha_i} P_j^i(c)\gamma_{ij}(y), \qquad c \in [0,1], \qquad i = 1, \ldots, \nu, \tag{7}$$

which is equivalent to (3), for $t \in [0, h]$. Moreover, by considering that

$$P_0^i(c) \equiv 1 \qquad \text{and} \qquad I^{\alpha_i} P_j^i(1) = \frac{\delta_{j0}}{\Gamma(\alpha_i + 1)},$$

from (4), (6), and (7) one derives:

$$y_i(h) = T_{i\ell}(h) + \frac{h^{\alpha_i}}{\Gamma(\alpha_i+1)}\gamma_{i0}(y) = T_{i\ell}(h) + \frac{h^{\alpha_i}}{\Gamma(\alpha_i)\alpha_i}\gamma_{i0}(y) \\ = T_{i\ell}(h) + \frac{1}{\Gamma(\alpha_i)}\int_0^h (h-x)^{\alpha_i - 1} f_i(y(x))\mathrm{d}x \equiv T_{i\ell}(h) + I^{\alpha_i} f_i(y(h)),$$

that is, (3) at $t = h$. Polynomial approximations of degree $s - 1$ to (5) are obtained by truncating the infinite series to finite sums with s terms:

$$\sigma_i^{(\alpha_i)}(ch) = \sum_{j=0}^{s-1} P_j^i(c)\gamma_{ij}(\sigma), \qquad c \in [0,1], \\ \sigma_i^{(\iota)}(0) = y_{i0}^{\iota} \in \mathbb{R}^{m_i}, \qquad \iota = 0, \ldots, \ell - 1, \qquad i = 1, \ldots, \nu, \\ \sigma = (\sigma_1^\top, \ldots, \sigma_\nu^\top)^\top \in \mathbb{R}^m, \tag{8}$$

with the Fourier coefficients $\gamma_{ij}(\sigma)$ defined according to (6), by formally replacing y with σ. As is clear, this is equivalent to requiring the residuals be orthogonal to all polynomials

of degree $s-1$, w.r.t. the inner product defined by (4):[2]

$$\int_0^1 \omega_i(c) P_j^i(c) \left[\sigma_i^{(\alpha_i)}(ch) - f_i(\sigma(ch))\right] \mathrm{d}c = 0, \qquad j = 0, \ldots, s-1, \qquad i = 1, \ldots, \nu. \tag{9}$$

Similarly to (7), the approximations σ_i are then given by integrating (8) side by side and imposing the initial conditions:

$$\begin{aligned}\sigma_i(ch) &= T_{i\ell}(ch) + \frac{h^{\alpha_i}}{\Gamma(\alpha_i)} \int_0^c (c-\tau)^{\alpha_i - 1} \sigma_i^{(\alpha_i)}(\tau h) \mathrm{d}\tau \;\equiv\; T_{i\ell}(ch) + I^{\alpha_i} \sigma_i^{(\alpha_i)}(ch) \\ &= T_{i\ell}(ch) + h^{\alpha_i} \sum_{j=0}^{s-1} I^{\alpha_i} P_j^i(c) \gamma_{ij}(\sigma), \qquad c \in [0,1], \qquad i = 1, \ldots, \nu. \end{aligned} \tag{10}$$

In particular, at $t = h$ one derives, similarly as in (8):

$$\begin{aligned}\sigma_i(h) &= T_{i\ell}(h) + \frac{h^{\alpha_i}}{\Gamma(\alpha_i + 1)} \gamma_{i0}(\sigma) \;=\; T_{i\ell}(h) + \frac{1}{\Gamma(\alpha_i)} \int_0^h (h-x)^{\alpha_i - 1} f_i(\sigma(x)) \mathrm{d}x \\ &\equiv T_{i\ell}(h) + I^{\alpha_i} f_i(\sigma(h)), \qquad i = 1, \ldots, \nu. \end{aligned}$$

Next, let us return to the interval $[0, T]$, and consider the case where a mesh

$$t_n = t_{n-1} + h_n, \qquad n = 1, \ldots, N, \qquad t_0 = 0, \qquad t_N = T, \tag{11}$$

is given. Consequently, now $\sigma(t)$ becomes a piecewise approximation to $y(t)$ on the whole interval. Moreover, let us denote:

$$\begin{aligned} \sigma_{in}(ch_n) &= \sigma_i(t_{n-1} + ch_n), \\ \sigma_{in}^{(\alpha_i)}(ch_n) &= \sum_{j=0}^{s-1} P_j^i(c) \gamma_{ij}(\sigma^n), \\ \gamma_{ij}(\sigma^n) &= \int_0^1 \omega_i(c) P_j^i(c) f_i(\sigma^n(ch_n)) \mathrm{d}c, \qquad j = 0, \ldots, s-1, \\ T_{i\ell}^n(ch_n) &= T_{i\ell}(t_{n-1} + ch_n), \qquad i = 1, \ldots, \nu, \\ \sigma^n(ch_n) &= \sigma(t_{n-1} + ch_n), \qquad c \in [0,1], \qquad n = 1, \ldots, N. \end{aligned} \tag{12}$$

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

$$\begin{aligned} \sigma_i(t) \equiv \sigma_{in}(ch_n) \;&=\; T_{i\ell}(t) + \frac{1}{\Gamma(\alpha_i)} \int_0^t (t-x)^{\alpha_i - 1} \sigma_i^{(\alpha_i)}(x) \mathrm{d}x \\ &= T_{i\ell}(t_{n-1} + ch_n) + \frac{1}{\Gamma(\alpha_i)} \int_0^{t_{n-1} + ch_n} (t_{n-1} + ch_n - x)^{\alpha_i - 1} \sigma_i^{(\alpha_i)}(x) \mathrm{d}x \\ &= T_{i\ell}^n(ch_n) + \frac{1}{\Gamma(\alpha_i)} \left[\sum_{\mu=1}^{n-1} \int_{t_{\mu-1}}^{t_\mu} (t_{n-1} + ch_n - x)^{\alpha_i - 1} \sigma_i^{(\alpha_i)}(x) \mathrm{d}x \right. \\ &\qquad \left. + \int_{t_{n-1}}^{t_{n-1} + ch_n} (t_{n-1} + ch_n - x)^{\alpha_i - 1} \sigma_i^{(\alpha_i)}(x) \mathrm{d}x \right] \end{aligned}$$

[2] This feature is common to HBVMs, obtained when $\alpha_i = 1$ [3, 16].

$$
\begin{aligned}
&= T_{i\ell}^n(ch_n) + \left[\sum_{\mu=1}^{n-1} \frac{h_\mu^{\alpha_i}}{\Gamma(\alpha_i)} \int_0^1 \left(\frac{t_{n-1}+ch_n-t_{\mu-1}}{h_\mu} - \tau\right)^{\alpha_i-1} \sigma_{i\mu}^{(\alpha_i)}(\tau h_\mu)\mathrm{d}\tau \right.\\
&\quad \left. + \frac{h_n^{\alpha_i}}{\Gamma(\alpha_i)} \int_0^c (c-\tau)^{\alpha_i-1} \sigma_{in}^{(\alpha_i)}(\tau h_n)\mathrm{d}\tau\right]\\
&= T_{i\ell}^n(ch_n) + \left[\sum_{\mu=1}^{n-1} \frac{h_\mu^{\alpha_i}}{\Gamma(\alpha_i)} \int_0^1 \left(\frac{t_{n-1}+ch_n-t_{\mu-1}}{h_\mu} - \tau\right)^{\alpha_i-1} \sum_{j=0}^{s-1} P_j^i(\tau)\gamma_{ij}(\sigma^\mu)\mathrm{d}\tau \right.\\
&\quad \left. + \frac{h_n^{\alpha_i}}{\Gamma(\alpha_i)} \int_0^c (c-\tau)^{\alpha_i-1} \sum_{j=0}^{s-1} P_j^i(\tau)\gamma_{ij}(\sigma^n)\mathrm{d}\tau\right]\\
&= T_{i\ell}^n(ch_n) + \sum_{\mu=1}^{n-1} h_\mu^{\alpha_i} \sum_{j=0}^{s-1} J_j^i\left(\frac{t_{n-1}+ch_n-t_{\mu-1}}{h_\mu}\right)\gamma_{ij}(\sigma^\mu) + h_n^{\alpha_i} \sum_{j=0}^{s-1} I^{\alpha_i} P_j^i(c)\gamma_{ij}(\sigma^n)\\
&\equiv \phi_i^n(ch_n) + h_n^{\alpha_i} \sum_{j=0}^{s-1} I^{\alpha_i} P_j^i(c)\gamma_{ij}(\sigma^n), \qquad c \in [0,1], \qquad i = 1,\dots,\nu, \qquad (13)
\end{aligned}
$$

having set, for $x \geq 1$,

$$
J_j^i(x) := \frac{1}{\Gamma(\alpha_i)} \int_0^1 (x-\tau)^{\alpha_i-1} P_j^i(\tau)\mathrm{d}\tau, \qquad j = 0,\dots,s-1, \qquad i = 1,\dots,\nu, \qquad (14)
$$

and the *memory* term

$$
\begin{aligned}
\phi_i^n(ch_n) := T_{i\ell}^n(ch_n) + \sum_{\mu=1}^{n-1} h_\mu^{\alpha_i} \sum_{j=0}^{s-1} J_j^i\left(\frac{t_{n-1}+ch_n-t_{\mu-1}}{h_\mu}\right)\gamma_{ij}(\sigma^\mu), \quad c \in [0,1],\\
i = 1,\dots,\nu. \qquad (15)
\end{aligned}
$$

Remark 2 For the efficient numerical evaluation of the integrals $I^{\alpha_i} P_j^i(c)$ in (7), and $J_j^i(x)$ in (14), we refer to [12, Section 3.2].

It must be emphasized that, since the memory term only depends on the past integration steps, according to (12)–(15) the discrete problem at step n consists in finding the $s\nu$ Fourier coefficients:

$$
\gamma_{ij}(\sigma^n) = \int_0^1 \omega_i(c) P_j^i(c) f_i(\sigma^n(ch_n))\mathrm{d}c, \qquad j = 0,\dots,s-1, \qquad i = 1,\dots,\nu. \qquad (16)
$$

In more details, by setting hereafter $I_r \in \mathbb{R}^{r\times r}$ the identity matrix,

$$
\mathbf{1}_r = (1,\ \dots,\ 1)^\top \in \mathbb{R}^r, \qquad \boldsymbol{\alpha} = (\alpha_1,\dots,\alpha_\nu),
$$

introducing the vectors, with $\otimes$ denoting the Kronecker product,

$$
\gamma_i(\sigma^n) = \begin{pmatrix} \gamma_{i0}(\sigma^n) \\ \vdots \\ \gamma_{i,s-1}(\sigma^n) \end{pmatrix} \in \mathbb{R}^{sm_i}, \qquad i = 1,\dots,\nu,
$$

$$\boldsymbol{\gamma}(\sigma^n) = \begin{pmatrix} \gamma_1(\sigma^n) \\ \vdots \\ \gamma_\nu(\sigma^n) \end{pmatrix} \in \mathbb{R}^{sm}, \tag{17}$$

the matrices and functions

$$P^i(c) = \begin{pmatrix} P_0^i(c) & & \\ & \ddots & \\ & & P_{s-1}^i(c) \end{pmatrix} \otimes I_{m_i} \in \mathbb{R}^{sm_i \times sm_i}, \qquad i = 1, \ldots, \nu,$$

$$P(c) = \begin{pmatrix} P^1(c) & & \\ & \ddots & \\ & & P^\nu(c) \end{pmatrix} \in \mathbb{R}^{sm \times sm},$$

$$I^{\alpha_i} P^i(c) = \left(I^{\alpha_i} P_0^i(c), \ \ldots, \ I^{\alpha_i} P_{s-1}^i(c) \right) \otimes I_{m_i} \in \mathbb{R}^{m_i \times sm_i}, \qquad i = 1, \ldots, \nu,$$

$$h_n^{\boldsymbol{\alpha}} = \begin{pmatrix} h_n^{\alpha_1} I_{m_1} & & \\ & \ddots & \\ & & h_n^{\alpha_\nu} I_{m_\nu} \end{pmatrix} \in \mathbb{R}^{m \times m},$$

$$I^{\boldsymbol{\alpha}} P(c) = \begin{pmatrix} I^{\alpha_1} P^1(c) & & \\ & \ddots & \\ & & I^{\alpha_\nu} P^\nu(c) \end{pmatrix} \in \mathbb{R}^{m \times sm},$$

$$\phi^n(ch_n) = \begin{pmatrix} \phi_1^n(ch_n) \\ \vdots \\ \phi_\nu^n(ch_n) \end{pmatrix} \in \mathbb{R}^m, \tag{18}$$

and taking into account that

$$\sigma^n(ch_n) = \phi^n(ch_n) + h_n^{\boldsymbol{\alpha}} I^{\boldsymbol{\alpha}} P(c) \boldsymbol{\gamma}(\sigma^n), \tag{19}$$

one has that (16) can be rewritten as:

$$\gamma_{ij}(\sigma^n) = \int_0^1 \omega_i(c) P_j^i(c) f_i \left(\phi^n(ch_n) + h_n^{\boldsymbol{\alpha}} I^{\boldsymbol{\alpha}} P(c) \boldsymbol{\gamma}(\sigma^n) \right) \mathrm{d}c,$$
$$j = 0, \ldots, s-1, \qquad i = 1, \ldots, \nu. \tag{20}$$

This problem, in turn, can be cast in vector form as

$$\boldsymbol{\gamma}(\sigma^n) = \int_0^1 \boldsymbol{\omega}(c) P(c) \boldsymbol{f} \left(\phi^n(ch_n) + h_n^{\boldsymbol{\alpha}} I^{\boldsymbol{\alpha}} P(c) \boldsymbol{\gamma}(\sigma^n) \right) \mathrm{d}c, \tag{21}$$

having set

$$\boldsymbol{f}(\sigma) = \begin{pmatrix} f_1(\sigma) \\ \vdots \\ f_\nu(\sigma) \end{pmatrix} \otimes \mathbf{1}_s \in \mathbb{R}^{sm},$$

$$\boldsymbol{\omega}(c) = \begin{pmatrix} \omega_1(c) I_{m_1} & & \\ & \ddots & \\ & & \omega_\nu(c) I_{m_\nu} \end{pmatrix} \otimes I_s \in \mathbb{R}^{sm \times sm}. \tag{22}$$

However, in order to derive a practical numerical method, the $s\nu$ Fourier coefficients (20) (i.e., the block vector (21)) need to be approximated by suitable quadrature rules. In the case where in (1) $\nu = 1$, this can be efficiently done by using one Gauss-Jacobi quadrature, defined at the zeros of the corresponding kth Jacobi polynomial: this results into a FHBVM(k, s) method. In the following sections, we study the extension to the case $\nu > 1$.

3 Extending FHBVMs to the Multi-Order Case

The straightforward extension of FHBVMs to cope with problem (1), consists in discretizing the integrals involved in the Fourier coefficients (20) at the nth integration step, by means of a corresponding Gauss-Jacobi quadrature. In more details, for a suitable $k \geq s$, one approximates (20) with the order $2k$ interpolatory quadrature rule defined at the zeros c_ρ^i, $\rho = 1, \dots, k$, of $P_k^i(c)$, with corresponding weights $\{b_\rho^i\}$:

$$\gamma_{ij}(\sigma^n) \approx \gamma_{ij}^n = \sum_{\rho=1}^{k} b_\rho^i P_j^i(c_\rho^i) f_i \left(\phi^n(c_\rho^i h_n) + h_n^{\boldsymbol{\alpha}} I^{\boldsymbol{\alpha}} P(c_\rho^i) \boldsymbol{\gamma}^n \right),$$
$$j = 0, \dots, s-1, \qquad i = 1, \dots, \nu, \tag{23}$$

having set (compare with (17))

$$\boldsymbol{\gamma}^n = \begin{pmatrix} \gamma_1^n \\ \vdots \\ \gamma_\nu^n \end{pmatrix}, \qquad \gamma_i^n = \begin{pmatrix} \gamma_{i0}^n \\ \vdots \\ \gamma_{i,s-1}^n \end{pmatrix}, \qquad i = 1, \dots, \nu. \tag{24}$$

According to the analysis in Section 6.2, this results into a FHBVM(k, s) method (see also [11]), having order at least s. In particular, the choice $k = s$, provides a collocation method. Moreover, a spectrally accurate solution can be achieved, by choosing $s \gg 1$, in view of (9). This is exactly the strategy used in the two Matlab© codes `fhbvm` and `fhbvm2`, implementing a FHBVM(22,20) and a FHBVM(22,22) method, respectively, designed for the case $\nu = 1$. In the case $\nu > 1$, however, one needs to evaluate the memory term, $\phi^n(ch_n)$, at the $k\nu$ abscissae

$$c_\rho^i, \qquad \rho = 1, \dots, k, \qquad i = 1, \dots, \nu. \tag{25}$$

This is a quite expensive part of the computational cost since, as it can be guessed from (15), all the information *ab initio* has to be computed for each abscissa in (25). Consequently, having ν different sets of abscissae is not a welcome news.

A different approach consists in looking for a set of suitable abscissae, $c_1, \dots, c_k$, such that the interpolatory quadrature rules defined over them are exact for polynomials of degree $2s-1$ (as it happens for the Gauss-Jacobi quadratures with $k = s$), for all the given weighting functions appearing in (20). I.e.,[3]

$$\forall g \in \Pi_{2s-1} \; : \; \int_0^1 \omega_i(c) g(c) \mathrm{d}c = \sum_{\rho=1}^{k} b_\rho^i \, g(c_\rho), \quad i = 1, \dots, \nu, \tag{26}$$

[3] Hereafter, Π_m will denote the vector space of polynomials of degree at most m.

where, for the sake of brevity, we continue denoting by $\{b_\rho^i\}$ the weights of the new quadratures. This is equivalent to find a family of polynomials $\{\pi_k\}$, such that, for a suitable $q \in \mathbb{N}$,

$$\pi_k(c_\rho)=0, \quad \rho=1,\dots,k, \quad \int_0^1 \omega_i(c)\pi_k(c)c^j \mathrm{d}c = 0, \quad j=0,\dots,q-1, \quad i=1,\dots,\nu. \tag{27}$$

This is a particular instance of a family of so-called *multiple orthogonal polynomials (MOPs)* [21, 62], which we shall briefly sketch in the next section. In particular, in the case of multiple Jacobi weights, as in our case, such abscissae are known as *Jacobi-Piñeiro abscissae* [40].

We observe that, in case (26)–(27) hold true, then we can compute [4]

$$\begin{aligned}\gamma_{ij}(\sigma^n) \approx \gamma_{ij}^n \;=\; & \sum_{\rho=1}^{k} b_\rho^i P_j^i(c_\rho) f_i\left(\phi^n(c_\rho h_n) + h_n^{\boldsymbol{\alpha}} I^{\boldsymbol{\alpha}} P(c_\rho)\boldsymbol{\gamma}^n\right), \\ & j=0,\dots,s-1, \qquad i=1,\dots,\nu,\end{aligned} \tag{28}$$

in place of (23). In such a way, we need to compute the memory term $\phi^n(ch_n)$ only at *one set* of k abscissae, instead of ν sets.

Remark 3 We observe that, when using (28) in place of (23), also the polynomials $P_j^i(c)$, and the corresponding integrals $I^{\alpha_i} P_j^i(c)$, $j=0,\dots,s-1$, $i=1,\dots,\nu$, are evaluated at only *one set* of abscissae, $c_1,\dots,c_k$. Conversely, in the case of (23), they need to be evaluated at ν different sets of abscissae, and this impacts unfavorably also in the implementation of the nonlinear iteration, as we shall sketch later in Section 5.

4 Multiple Orthogonal Polynomials (MOPs)

To begin with, the aim is that of computing a set of abscissae $c_1,\dots,c_k$, satisying (27), for a suitable k to be determined. To this end, the following result holds true.

Theorem 1 *Assume (27) holds true. Then the interpolatory quadratures with weights*

$$b_\rho^i = \int_0^1 \omega_i(c)L_{\rho k}(c)\mathrm{d}c, \qquad L_{\rho k}(c) = \prod_{\iota\neq\rho}\frac{c-c_\iota}{c_\rho-c_\iota}, \qquad \rho=1,\dots,k, \qquad i=1,\dots,\nu,$$

satisfy

$$\int_0^1 \omega_i(c)g(c)\mathrm{d}c = \sum_{\rho=1}^{k} b_\rho^i g(c_\rho), \qquad i=1,\dots,\nu, \qquad \forall g\in\Pi_{k+q-1}.$$

Proof See [8, Theorem 1]. □

Consequently, in order for this approach to reduce to the standard one [11, 14], when $\nu=1$, hereafter we require:

$$k=\nu q, \qquad k+q-1\geq 2s-1, \tag{29}$$

so that all the quadratures (c_ρ, b_ρ^i), $i=1,\dots,\nu$, have at least order $2s$. Therefore, the following result is proved.

[4] Of course, the weights $\{b_\rho^i\}$ are not the same as those in (23) but, instead, are those in (26) .

Table 1 Values of k according to (30) for increasing values of ν, for $s = 22$

ν	1	2	3	4	5
k	22	30	33	36	40

Corollary 1 *The conditions (29) are satisfied, provided that*[5]

$$q = \left\lceil \frac{2s}{\nu+1} \right\rceil \quad \Rightarrow \quad k = \nu \left\lceil \frac{2s}{\nu+1} \right\rceil. \tag{30}$$

Table 1 shows the values of k for $s = 22$, which is the value considered in the code `fhbvm2` [14].

Consequently, the problem is that of finding the zeros of the polynomial $\pi_k(c)$ in (27). A straightforward approach, to determine such a polynomial, consists in expressing it w.r.t. the power basis,

$$\pi_k(c) = \sum_{\iota=0}^{k} a_\iota c^{k-\iota}, \qquad a_0 = 1,$$

and impose the k orthogonality conditions (27) (with $k = \nu q$, according to (29)). These orthogonality conditions, by setting hereafter

$$(f(c), g(c))_i = \int_0^1 \omega_i(c) f(c) g(c) \mathrm{d}c, \qquad i = 1, \ldots, \nu, \tag{31}$$

read

$$\begin{pmatrix} (c^0, c^0)_1 & \ldots & (c^0, c^{k-1})_1 \\ \vdots & & \vdots \\ (c^{q-1}, c^0)_1 & \ldots & (c^{q-1}, c^{k-1})_1 \\ \vdots & & \vdots \\ (c^0, c^0)_\nu & \ldots & (c^0, c^{k-1})_\nu \\ \vdots & & \vdots \\ (c^{q-1}, c^0)_\nu & \ldots & (c^{q-1}, c^{k-1})_\nu \end{pmatrix} \begin{pmatrix} a_k \\ \vdots \\ a_1 \end{pmatrix} = - \begin{pmatrix} (c^0, c^k)_1 \\ \vdots \\ (c^{q-1}, c^k)_1 \\ \vdots \\ (c^0, c^k)_\nu \\ \vdots \\ (c^{q-1}, c^k)_\nu \end{pmatrix}.$$

Nevertheless, such linear system of k equations has the pitfall of having a badly conditioned coefficient matrix and is, therefore, not practically useful to recover $\pi_k(c)$. A different approach consists in deriving this polynomial as the kth one in the sequence $\{\pi_j\}_{j\geq 0}$ of monic polynomials satifying the recurrence [49][6]

$$\pi_j(c) = c\pi_{j-1}(c) - \sum_{i=\max(1, j-\nu)}^{j} a_{ji}\pi_{i-1}(c), \quad j = 1, 2, \ldots, k, \qquad \pi_0(c) \equiv 1. \tag{32}$$

[5] Hereafter, $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ will denote the ceiling and floor functions, respectively.

[6] The notation has been slightly modified, w.r.t. [49], in order to have a more "familiar" enumeration of the entries of matrix H_k in (34).

This can be cast in vector form as

$$H_k \begin{pmatrix} \pi_0(c) \\ \pi_1(c) \\ \vdots \\ \vdots \\ \pi_{k-1}(c) \end{pmatrix} = c \begin{pmatrix} \pi_0(c) \\ \pi_1(c) \\ \vdots \\ \vdots \\ \pi_{k-1}(c) \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ \pi_k(c) \end{pmatrix}, \tag{33}$$

with $H_k \in \mathbb{R}^{k\times k}$ a lower Hessenberg matrix,

$$H_k = \begin{pmatrix} a_{11} & 1 & & & & \\ \vdots & \ddots & \ddots & & & \\ a_{\nu+1,1} & \dots & a_{\nu+1,\nu+1} & 1 & & \\ & \ddots & & \ddots & \ddots & \\ & & a_{k-1,k-1-\nu} & \dots & a_{k-1,k-1} & 1 \\ & & & a_{k,k-\nu} & \dots & a_{kk} \end{pmatrix}. \tag{34}$$

From (33) one deduces that $\pi_k(c_\rho) = 0$, $\rho = 1, \dots, k$, iff $c_\rho \in \sigma(H_k)$, with (right) eigenvectors respectively given by:

$$\begin{pmatrix} \pi_0(c_\rho) \\ \vdots \\ \pi_{k-1}(c_\rho) \end{pmatrix}, \qquad \rho = 1, \dots, k. \tag{35}$$

The coefficients $\{a_{ji}\}$, in turn, if $j \in \{1, \dots, k\}$ is such that

$$j = r\nu + \mu, \qquad \text{with} \qquad \mu \in \{0, \dots, \nu - 1\}, \tag{36}$$

are computed by imposing that, by using the notation (31),[7]

$$(\pi_j(c), c^\rho)_i = 0, \qquad \rho = 0, \dots, r, \qquad i = 1, \dots, \mu,$$

$$(\pi_j(c), c^\rho)_i = 0, \qquad \rho = 0, \dots, r-1, \qquad i = \mu+1, \dots, \nu,$$

which is clearly equivalent to require:

$$(\pi_j(c), \pi_\rho(c))_i = 0, \qquad \rho = 0, \dots, r, \qquad i = 1, \dots, \mu,$$

$$(\pi_j(c), \pi_\rho(c))_i = 0, \qquad \rho = 0, \dots, r-1, \qquad i = \mu+1, \dots, \nu. \tag{37}$$

By setting, with reference to (36),

$$\xi(j) = \begin{cases} \mu, \text{ if } \mu > 0, \\ \nu, \text{ if } \mu = 0, \end{cases} \tag{38}$$

this results into [49]:

$$a_{ji} = \frac{\left(c\pi_{j-1}(c) - \sum_{\iota=\max(1,j-\nu)}^{i-1} a_{j\iota}\pi_{\iota-1}(c),\ \pi_{\left\lfloor \frac{i-1}{\nu} \right\rfloor}(c) \right)_{\xi(i)}}{\left(\pi_{i-1}(c), \pi_{\left\lfloor \frac{i-1}{\nu} \right\rfloor}(c) \right)_{\xi(i)}},$$

[7] Of course, when $r - 1 < 0$, it is meant that the corresponding orthogonality condition is void.

$$i = \max(1, j - \nu), \ldots, j, \qquad j = 1, \ldots, k. \tag{39}$$

In fact, the following result holds true.

Theorem 2 *By considering $k = q\nu$, according to (29), the family of polynomials $\{\pi_1, \ldots, \pi_k\}$, defined by (32)–(36) and (38)–(39), satisfies the orthogonality conditions (37) (see (31)).*

Proof Following an induction argument on the index j varying on ν consecutive values, for $j = 1$, one has

$$\pi_1(c) = c - a_{11}, \qquad a_{11} = \frac{(c, 1)_1}{(1, 1)_1},$$

so that (37) holds true with $r = 0$ and $\mu = 1$. A straightforward induction argument, shows that for $j = 2, \ldots, \nu$, from (32) one obtains:

$$(\pi_j(c), \pi_0(c))_i = 0 \qquad \Rightarrow \qquad a_{ji} = \frac{\left(c\pi_{j-1}(c) - \sum_{\iota=1}^{i-1} a_{j\iota}\pi_{\iota-1}(c), \pi_0(c)\right)_i}{(\pi_{i-1}(c), \pi_0(c))_i},$$
$$i = 1, \ldots, j,$$

which matches (39), as required. Next, let us suppose (37) holds true up to $j = r\nu$, $r \geq 1$, and prove for

$$j = r\nu + \mu, \qquad \mu = 1, \ldots, \nu. \tag{40}$$

Let us start with $\mu = 1$. Clearly,

$$j = r\nu + 1 \qquad \Rightarrow \qquad \xi(i) = 1, \ldots, \nu, 1, \qquad \text{as} \qquad i = j - \nu, \ldots, j - 1, j. \tag{41}$$

Moreover, we know that (see (32))

$$i - 1 \in \{j - \nu - 1, \ldots, j - 1\} = \{(r - 1)\nu, \ldots, r\nu\},$$

so that, by induction hypothesis, if $i - 1 = (r - 1)\nu + \eta$, $\eta = 0, \ldots, \nu$, then the following orthogonality conditions hold true:[8]

$$(\pi_{j-\nu}(c), \pi_\rho(c))_\iota = 0, \qquad \rho = 0, \ldots, r - 2, \qquad \iota = 1, \ldots, \nu, \qquad \text{(i.e., } \eta = 0),$$

$$(\pi_{i-1}(c), \pi_\rho(c))_\iota = 0, \qquad \rho = 0, \ldots, r - 1, \qquad \iota = 1, \ldots, \eta,$$
$$(\pi_{i-1}(c), \pi_\rho(c))_\iota = 0, \qquad \rho = 0, \ldots, r - 2, \qquad \iota = \eta + 1, \ldots, \nu, \qquad \eta = 1, \ldots, \nu - 1,$$

$$(\pi_{j-1}(c), \pi_\rho(c))_\iota = 0, \qquad \rho = 0, \ldots, r - 1, \qquad \iota = 1, \ldots, \nu, \qquad \text{(i.e., } \eta = \nu). \tag{42}$$

Next, according to (37), let us impose that π_j satisfies the following orthogonality conditions,

$$(\pi_j(c), \pi_\rho(c))_i = 0, \qquad \rho = 0, \ldots, r - 1, \qquad i = 1, \ldots, \nu,$$
$$(\pi_j(c), \pi_r(c))_1 = 0,$$

which, by virtue of (42), reduce to

$$(\pi_j(c), \pi_{r-1}(c))_i = 0, \qquad i = 1, \ldots, \nu,$$

[8] Clearly, if $r - 2 < 0$, the corresponding ortogonality condition is void.

$$(\pi_j(c), \pi_r(c))_1 = 0. \tag{43}$$

Imposing such conditions in the given order, then gives, according to (41),

$$a_{j,j-\nu+\eta} = \frac{\left(c\pi_{j-1}(c) - \sum_{\iota=j-\nu}^{j-\nu+\eta-1} a_{j\iota}\pi_{\iota-1}(c),\ \pi_{r-1}(c)\right)_{\eta+1}}{\left(\pi_{j-\nu+\eta-1}(c), \pi_{r-1}(c)\right)_{\eta+1}}, \qquad \eta = 0, \ldots, \nu-1,$$

and

$$a_{jj} = \frac{\left(c\pi_{j-1}(c) - \sum_{\iota=j-\nu}^{j-1} a_{j\iota}\pi_{\iota-1}(c),\ \pi_r(c)\right)_1}{\left(\pi_{j-1}(c), \pi_r(c)\right)_1},$$

respectively, which match (39) when $j = r\nu + 1$. The subsequent values of μ in (40) can be handled by means of similar arguments. In fact, for $\mu = 2, \ldots, \nu$, one needs to impose (compare with (43)):

$$(\pi_j(c), \pi_{r-1}(c))_i = 0, \qquad i = \mu, \ldots, \nu,$$

$$(\pi_j(c), \pi_r(c))_i = 0, \qquad i = 1 \ldots, \mu.$$

From these orthogonality conditions, by considering that, for $i = j-\nu, \ldots, j, \quad \xi(i) = \mu, \ldots, \nu, 1, \ldots, \mu$, one eventually obtains:

$$a_{j,j-\nu+\eta} = \frac{\left(c\pi_{j-1}(c) - \sum_{\iota=j-\nu}^{j-\nu+\eta-1} a_{j\iota}\pi_{\iota-1}(c),\ \pi_{r-1}(c)\right)_{\eta+\mu}}{\left(\pi_{j-\nu+\eta-1}(c), \pi_{r-1}(c)\right)_{\eta+\mu}}, \qquad \eta = 0, \ldots, \nu-\mu,$$

and

$$a_{j,j-\nu+\eta} = \frac{\left(c\pi_{j-1}(c) - \sum_{\iota=j-\nu}^{j-\nu+\eta-1} a_{j\iota}\pi_{\iota-1}(c),\ \pi_r(c)\right)_{\eta-\nu+\mu}}{\left(\pi_{j-\nu+\eta-1}(c), \pi_r(c)\right)_{\eta-\nu+\mu}}, \qquad \eta = \nu-\mu+1, \ldots, \nu,$$

which, again, are consistent with (39). Consequently, the statement follows. □

Remark 4 The proof of Theorem 2 has been reported for a twofold reason:

1. a formal proof, in the general case, was still lacking in the literature (usually, only the case $\nu = 2$ is worked out);
2. the proof itself provides operative guidelines for efficiently deriving the entries of matrix H_k in (34).

4.1 Some Computational Remarks

In order to derive a practical numerical method, one needs:

1. at first, compute the integrands involved in (39) (see (31)). By taking into account (29)–(30), one has that the integrands are polynomials of degree at most

   $$k-1+\left\lfloor \frac{k-1}{\nu} \right\rfloor = \nu q-1+\left\lfloor \frac{\nu q-1}{\nu} \right\rfloor = \nu q+q-2 = (\nu+1)q-2 = (\nu+1)\left\lceil \frac{2s}{\nu+1} \right\rceil -2.$$

   Consequently, they can be computed exactly (up to round-off errors) by using a corresponding Gauss-Jacobi quadrature over

   $$\varphi = \left\lceil \frac{\nu+1}{2}\left\lceil \frac{2s}{\nu+1} \right\rceil - \frac{1}{2} \right\rceil \tag{44}$$

Table 2 Values of φ in (44) for increasing values of ν, for $s = 22$.

ν	1	2	3	4	5
φ	22	22	22	22	24

abscissae. For sake of completeness, in Table 2 we list a few values of φ, for the case $s = 22$ considered in the code `fhbvm2`, for increasing values of ν;

2. secondly, matrix H_k in (33)–(34) must be balanced, in order to improve accuracy in numerically finding its eigenvalues. For this purpose, a recent strategy, proposed in [40], consists in a diagonal scaling that symmetrizes the tridiagonal part of H_k. Additional computational aspects have been also considered in [41, 61].

5 Solving the Discrete Problems

As sketched above (see (28)), the discrete problem to be solved at the nth integration step consists in:

$$\gamma_{ij}^n = \sum_{\rho=1}^{k} b_\rho^i P_j^i(c_\rho) f_i\left(\phi^n(c_\rho h_n) + h_n^{\boldsymbol{\alpha}} I^{\boldsymbol{\alpha}} P(c_\rho)\boldsymbol{\gamma}^n\right), \quad j = 0, \ldots, s-1, \quad i = 1, \ldots, \nu, \tag{45}$$

where the notations (17)–(18) and (24) have been used. By using notations (17)–(18) and (24) again, we can recast (45) as:

$$\gamma_i^n = (\mathcal{P}_s^i)^\top \Omega_i \cdot f_i\left(\phi^n(\boldsymbol{c}h_n) + \boldsymbol{h}_n^{\boldsymbol{\alpha}} \mathcal{I}_s^{\boldsymbol{\alpha}} \boldsymbol{\gamma}^n\right), \qquad i = 1, \ldots, \nu, \tag{46}$$

where we have set

$$\boldsymbol{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix} \in \mathbb{R}^k, \qquad \phi^n(\boldsymbol{c}h_n) = \begin{pmatrix} \phi^n(c_1 h_n) \\ \vdots \\ \phi^n(c_k h_n) \end{pmatrix} \in \mathbb{R}^{km},$$

$$\boldsymbol{h}_n^{\boldsymbol{\alpha}} = I_k \otimes h_n^{\boldsymbol{\alpha}} \in \mathbb{R}^{km \times km}, \tag{47}$$

additionally,

$$\Omega_i = \begin{pmatrix} b_1^i & & \\ & \ddots & \\ & & b_k^i \end{pmatrix} \otimes I_{m_i} \in \mathbb{R}^{km_i \times km_i},$$

$$\mathcal{P}_s^i = \begin{pmatrix} P_0^i(c_1) & \ldots & P_{s-1}^i(c_1) \\ \vdots & & \vdots \\ P_0^i(c_k) & \ldots & P_{s-1}^i(c_k) \end{pmatrix} \otimes I_{m_i} \in \mathbb{R}^{km_i \times sm_i},$$

$$\mathcal{I}_s^{\boldsymbol{\alpha}} = \begin{pmatrix} I^{\boldsymbol{\alpha}} P(c_1) \\ \vdots \\ I^{\boldsymbol{\alpha}} P(c_k) \end{pmatrix} \in \mathbb{R}^{km \times sm}, \tag{48}$$

and

$$f_i\left(\phi^n(\boldsymbol{c}h_n)+\boldsymbol{h}_n^{\boldsymbol{\alpha}}\mathcal{I}_s^{\boldsymbol{\alpha}}\boldsymbol{\gamma}^n\right)=\begin{pmatrix} f_i\left(\phi^n(c_1h_n)+h_n^{\boldsymbol{\alpha}}I^{\boldsymbol{\alpha}}P(c_1)\boldsymbol{\gamma}^n\right)\\ \vdots\\ f_i\left(\phi^n(c_kh_n)+h_n^{\boldsymbol{\alpha}}I^{\boldsymbol{\alpha}}P(c_k)\boldsymbol{\gamma}^n\right)\end{pmatrix},\qquad i=1,\dots,\nu. \tag{49}$$

By further introducing the notation

$$\mathcal{P}_s=\begin{pmatrix}\mathcal{P}_s^1 & & \\ & \ddots & \\ & & \mathcal{P}_s^\nu\end{pmatrix}\in\mathbb{R}^{km\times sm},\qquad \Omega=\begin{pmatrix}\Omega_1 & & \\ & \ddots & \\ & & \Omega_\nu\end{pmatrix}\in\mathbb{R}^{km\times km}, \tag{50}$$

and

$$f\left(\phi^n(\boldsymbol{c}h_n)+\boldsymbol{h}_n^{\boldsymbol{\alpha}}\mathcal{I}_s^{\boldsymbol{\alpha}}\boldsymbol{\gamma}^n\right)=\begin{pmatrix} f_1\left(\phi^n(\boldsymbol{c}h_n)+\boldsymbol{h}_n^{\boldsymbol{\alpha}}\mathcal{I}_s^{\boldsymbol{\alpha}}\boldsymbol{\gamma}^n\right)\\ \vdots\\ f_\nu\left(\phi^n(\boldsymbol{c}h_n)+\boldsymbol{h}_n^{\boldsymbol{\alpha}}\mathcal{I}_s^{\boldsymbol{\alpha}}\boldsymbol{\gamma}^n\right)\end{pmatrix}\in\mathbb{R}^{km}, \tag{51}$$

one eventually obtains that (46) can be compactly written as:

$$\boldsymbol{\gamma}^n=\mathcal{P}_s^\top\Omega\cdot f\left(\phi^n(\boldsymbol{c}h_n)+\boldsymbol{h}_n^{\boldsymbol{\alpha}}\mathcal{I}_s^{\boldsymbol{\alpha}}\boldsymbol{\gamma}^n\right). \tag{52}$$

From this formulation, the following result follows.

Theorem 3 *Assume f is Lipschitz continuous. Then, for all sufficiently small $h_n>0$ problem (52) has a unique solution.*

Proof In fact, from (17)–(18) and (50), it follows that, provided that, for any suitable matrix norm,

$$h_n^{\min_i\alpha_i}L\|\mathcal{P}_s^\top\Omega\|\|\mathcal{I}_s^{\boldsymbol{\alpha}}\|<1, \tag{53}$$

with L the local Lipschitz constant, the iteration

$$\boldsymbol{\gamma}^{n,r+1}=\mathcal{P}_s^\top\Omega\cdot f\left(\phi^n(\boldsymbol{c}h_n)+\boldsymbol{h}_n^{\boldsymbol{\alpha}}\mathcal{I}_s^{\boldsymbol{\alpha}}\boldsymbol{\gamma}^{n,r}\right),\qquad r=0,1,\dots,\qquad \boldsymbol{\gamma}^{n,0}=\mathbf{0}, \tag{54}$$

is a contraction. □

Remark 5 Even though the fixed-point iteration (54) is straightforward and easy to implement, nevertheless, the condition (53) may be too restrictive, if $L\gg 1$: in such a case, a Newton-type iteration has to be used. This aspect will be studied in the next two subsections 5.1 and 5.2.

5.1 The Case $\nu=1$

In the case $\nu=1$, i.e., when only one fractional derivative occurs, a very efficient Newton-type iteration can be considered, i.e., the so-called *blended iteration*, at first derived in the ODE case [16–18], and then adapted for the FDE case [12]. We briefly sketch it, for sake of completeness. In such a case, from (17)–(18), (47)–(48), and (50), with $\nu=1$, considering that $m_1=m$, it follows that (52) becomes

$$\gamma_1^n=(\mathcal{P}_s^1)^\top\Omega_1\cdot f_1\left(\phi^n(\boldsymbol{c}h_n)+h_n^{\alpha_1}\mathcal{I}_s^{\alpha_1}\gamma_1^n\right).$$

Consequently, we need to solve the problem

$$G(\gamma):=\gamma-(\mathcal{P}_s^1)^\top\Omega_1\cdot f_1\left(\phi^n(\boldsymbol{c}h_n)+h_n^{\alpha_1}\mathcal{I}_s^{\alpha_1}\gamma\right)=0.$$

Some algebra shows that the simplified Newton iteration, for its solution, reads:[9]

$$\begin{aligned} &\text{solve: } \left[I_{sm} - h_n^{\alpha_1} X_{11} \otimes f_1'(\phi^n(0))\right] \delta^j = -G(\gamma^j), \\ &\qquad\qquad\qquad\qquad \text{update: } \gamma^{j+1} = \gamma^j + \delta^j, \qquad j = 0, 1, \ldots, \end{aligned} \tag{55}$$

with

$$\begin{aligned} X_{11} = &\begin{pmatrix} P_0^1(c_1) & \ldots & P_0^1(c_k) \\ \vdots & & \vdots \\ P_{s-1}^1(c_1) & \ldots & P_{s-1}^1(c_k) \end{pmatrix} \begin{pmatrix} b_1^1 & & \\ & \ddots & \\ & & b_k^1 \end{pmatrix} \begin{pmatrix} I^{\alpha_1} P_0^1(c_1) & \ldots & I^{\alpha_1} P_{s-1}^1(c_1) \\ \vdots & & \vdots \\ I^{\alpha_1} P_0^1(c_k) & \ldots & I^{\alpha_1} P_{s-1}^1(c_k) \end{pmatrix} \\ &\in \mathbb{R}^{s\times s}, \end{aligned} \tag{56}$$

where the initial condition $\gamma^0 = 0$ can be conveniently used. The iteration (55)–(56), though straightforward, requires to factor the matrix

$$\left[I_{sm} - h_n^{\alpha_1} X_{11} \otimes f_1'(\phi^n(0))\right] \in \mathbb{R}^{sm\times sm}. \tag{57}$$

The associated *blended iteration*, formally obtained by combining (55) with the equivalent formulation

$$\rho_{11}\left[X_{11}^{-1} \otimes I_m - h_n^{\alpha_1} I_s \otimes f_1'(\phi^n(0))\right] \delta_j = -\rho_{11}(X_{11}^{-1} \otimes I_m) G(\gamma^j),$$

with weights θ^n and $I_{sm} - \theta^n$, respectively, where

$$\theta^n = I_s \otimes \left[I_m - \rho_{11} h_n^{\alpha_1} f_1'(\phi^n(0))\right]^{-1}$$

and $\rho_{11} > 0$ is a suitable parameter depending on the spectrum of matrix X_{11} [12], reads, as follows:

$$\begin{aligned} \text{set:} \qquad \eta_j &= -G(\gamma^j) \\ \hat{\eta}_j &= \rho_{11}(X_{11}^{-1} \otimes I_m)\eta_j \\ \gamma^{j+1} &= \gamma^j - \theta^n\left[\hat{\eta}_j + \theta^n\left(\eta_j - \hat{\eta}_j\right)\right], \qquad j = 0, 1, \ldots. \end{aligned} \tag{58}$$

Also in this case, one conveniently starts at $\gamma^0 = 0$. Clearly, this iteration only requires to factor the matrix

$$I_m - \rho_{11} h_n^{\alpha_1} f_1'(\phi^n(0)) \in \mathbb{R}^{m\times m},$$

instead of (57). Unfortunately, this very effective approach, implemented in the Matlab© codes `fhbvm` [12] and `fhbbm2` [14], does not extend to the case $\nu > 1$.

5.2 The Case $\nu > 1$

To cope with this case, we need to rely on the straight simplified Newton iteration for solving (52), i.e.,

$$G(\boldsymbol{\gamma}) := \boldsymbol{\gamma} - \mathcal{P}_s^\top \Omega \cdot f\left(\phi^n(\boldsymbol{c}h_n) + \boldsymbol{h}_n^{\boldsymbol{\alpha}} \mathcal{I}_s^{\boldsymbol{\alpha}} \boldsymbol{\gamma}\right) = \mathbf{0},$$

which now reads:

[9] As is usual, f_1' denotes the Jacobian matrix of f_1.

$$\text{solve: } \left[I_{sm} - \mathcal{P}_s^\top \Omega \cdot (I_k \otimes f'(\phi^n(0))) \cdot \boldsymbol{h}_n^{\boldsymbol{\alpha}} \mathcal{I}_s \right] \boldsymbol{\delta}^j = -G(\boldsymbol{\gamma}^j),$$

$$\text{update: } \boldsymbol{\gamma}^{j+1} = \boldsymbol{\gamma}^j + \boldsymbol{\delta}^j, \qquad j = 0, 1, \ldots, \tag{59}$$

where, again, $\boldsymbol{\gamma}^0 = \mathbf{0}$ is conveniently used. For sake of clarity, let us explain in detail the (block) structure of the matrix

$$\mathcal{P}_s^\top \Omega \cdot (I_k \otimes f'(\phi^n(0))) \cdot \boldsymbol{h}_n^{\boldsymbol{\alpha}} \mathcal{I}_s \equiv \begin{pmatrix} h_n^{\alpha_1} X_{11} \otimes f_{11}^n & \ldots & h_n^{\alpha_\nu} X_{1\nu} \otimes f_{1\nu}^n \\ \vdots & & \vdots \\ h_n^{\alpha_1} X_{\nu 1} \otimes f_{\nu 1}^n & \ldots & h_n^{\alpha_\nu} X_{\nu\nu} \otimes f_{\nu\nu}^n \end{pmatrix} \in \mathbb{R}^{sm \times sm},$$

where, with reference to (1) and (51), for all $i, j = 1, \ldots, \nu$:

$$f_{ij}^n = \frac{\partial}{\partial y_j} f_i(\phi^n(0)) \in \mathbb{R}^{m_i \times m_j},$$

and, generalizing (56),

$$X_{ij} = \begin{pmatrix} P_0^i(c_1) & \ldots & P_0^i(c_k) \\ \vdots & & \vdots \\ P_{s-1}^i(c_1) & \ldots & P_{s-1}^i(c_k) \end{pmatrix} \begin{pmatrix} b_1^i & & \\ & \ddots & \\ & & b_k^i \end{pmatrix} \begin{pmatrix} I^{\alpha_j} P_0^j(c_1) & \ldots & I^{\alpha_j} P_{s-1}^j(c_1) \\ \vdots & & \vdots \\ I^{\alpha_j} P_0^j(c_k) & \ldots & I^{\alpha_j} P_{s-1}^j(c_k) \end{pmatrix} \in \mathbb{R}^{s \times s}. \tag{60}$$

Clearly, in this case the $sm \times sm$ coefficient matrix in (59), i.e.,

$$I_{sm} - \begin{pmatrix} h_n^{\alpha_1} X_{11} \otimes f_{11}^n & \ldots & h_n^{\alpha_\nu} X_{1\nu} \otimes f_{1\nu}^n \\ \vdots & & \vdots \\ h_n^{\alpha_1} X_{\nu 1} \otimes f_{\nu 1}^n & \ldots & h_n^{\alpha_\nu} X_{\nu\nu} \otimes f_{\nu\nu}^n \end{pmatrix} \in \mathbb{R}^{sm \times sm}, \tag{61}$$

needs to be factored and, therefore, the computational cost is higher than in the case $\nu = 1$, where the more efficient *blended iteration* (58) can be used.

Remark 6 In case one uses the approach (23), instead of (28), for approximating the Fourier coefficients, the simplified Newton iteration would require to factor the matrix

$$I_{sm} - \begin{pmatrix} h_n^{\alpha_1} X_{11}^1 \otimes f_{11}^n & \ldots & h_n^{\alpha_\nu} X_{1\nu}^1 \otimes f_{1\nu}^n \\ \vdots & & \vdots \\ h_n^{\alpha_1} X_{\nu 1}^\nu \otimes f_{\nu 1}^n & \ldots & h_n^{\alpha_\nu} X_{\nu\nu}^\nu \otimes f_{\nu\nu}^n \end{pmatrix} \in \mathbb{R}^{sm \times sm},$$

in place of (61), with

$$X_{ij}^i = \begin{pmatrix} P_0^i(c_1^i) & \ldots & P_0^i(c_k^i) \\ \vdots & & \vdots \\ P_{s-1}^i(c_1^i) & \ldots & P_{s-1}^i(c_k^i) \end{pmatrix} \begin{pmatrix} b_1^i & & \\ & \ddots & \\ & & b_k^i \end{pmatrix} \begin{pmatrix} I^{\alpha_j} P_0^j(c_1^i) & \ldots & I^{\alpha_j} P_{s-1}^j(c_1^i) \\ \vdots & & \vdots \\ I^{\alpha_j} P_0^j(c_k^i) & \ldots & I^{\alpha_j} P_{s-1}^j(c_k^i) \end{pmatrix} \in \mathbb{R}^{s \times s}.$$

Consequently, for all $i = 1, \ldots, \nu$, the ith block row depends on the corrresponding set of Jacobi abscissae (i.e., the zeros of $P_k^i(c)$).

6 Analysis of the Methods

In this section we sketch a linear stability analysis of FHBVMs and study their convergence order. We also report a few numerical tests to assess the theoretical findings.

6.1 Linear Stability Analysis

We consider problem (1) in the linear and autonomous case, assuming, without losing generality, that $\nu = m$:

$$y_i^{(\alpha_i)}(t) = \sum_{j=1}^{m} a_{ij} y_j(t), \qquad t > 0, \qquad y_i(0) = y_{i0} \in \mathbb{R}, \qquad 0 < \alpha_i < 1, \qquad i = 1, \dots, m. \tag{62}$$

We also consider the corresponding shifted orthonormal Jacobi polynomial bases (4). Then, fixing a suitable $s \geq 1$, according to (8)–(9), a FHBVM, used with timestep t, basically determines approximations:

$$\sigma_i^{(\alpha_i)}(ct) = \sum_{\nu=0}^{s-1} P_\nu^i(c)\gamma_{i\nu}, \qquad \sigma_i(ct) = y_{i0} + t^{\alpha_i} \sum_{\nu=0}^{s-1} I^{\alpha_i} P_\nu^i(c)\gamma_{i\nu}, \qquad c \in [0,1], \tag{63}$$

with the coefficients $\{\gamma_{i\nu}\}$ determined such that

$$\int_0^1 \omega_i(c) P_\iota^i(c) \left[\sigma_i^{(\alpha_i)}(ct) - \sum_{j=1}^{m} a_{ij} \sigma_j(ct) \right] \mathrm{d}c = 0, \qquad \iota = 0, \dots, s-1, \quad i = 1, \dots, m. \tag{64}$$

From (4) and (63), one then obtains that, recalling that $\delta_{\iota 0}$ denotes the Kronecker symbol,

$$\gamma_{i\iota} - \sum_{j=1}^{m} a_{ij} \left[\delta_{\iota 0} y_{j0} + t^{\alpha_j} \sum_{\nu=0}^{s-1} \left(\int_0^1 \omega_i(c) P_\iota^i(c) I^{\alpha_j} P_\nu^j(c) \mathrm{d}c \right) \gamma_{j\nu} \right] = 0, \quad \iota = 0, \dots, s-1, \quad i = 1, \dots, m. \tag{65}$$

More compactly, by setting

$$x_{ij}^{\iota\nu} = \int_0^1 \omega_i(c) P_\iota^i(c) I^{\alpha_j} P_\nu^j(c) \mathrm{d}c, \qquad i, j = 1, \dots, m, \quad \iota, \nu = 0, \dots, s-1, \tag{66}$$

and

$$\psi_i = \sum_{j=1}^{m} a_{ij} y_{j0}, \qquad i = 1, \dots, m, \tag{67}$$

one derives that (65) is equivalent to:

$$\gamma_{i\iota} - \sum_{j=1}^{m} a_{ij} t^{\alpha_j} \sum_{\nu=0}^{s-1} x_{ij}^{\iota\nu} \gamma_\nu^j = \delta_{\iota 0} \psi_i, \qquad \iota = 0, \dots, s-1, \quad i = 1, \dots, m. \tag{68}$$

Setting $e_1 \in \mathbb{R}^s$ the first unit vector, the vectors

$$\gamma^i = \begin{pmatrix} \gamma_{i0} \\ \vdots \\ \gamma_{i,s-1} \end{pmatrix} \in \mathbb{R}^s, \quad i = 1, \dots, m, \tag{69}$$

the matrices

$$X_{ij} = \begin{pmatrix} x_{ij}^{0,0} & \dots & x_{ij}^{0,s-1} \\ \vdots & & \vdots \\ x_{ij}^{s-1,0} & \dots & x_{ij}^{s-1,s-1} \end{pmatrix}, \quad B_{ij} = a_{ij} X_{ij} \in \mathbb{R}^{s\times s}, \qquad i, j = 1, \dots, m, \tag{70}$$

and recalling $I_s \in \mathbb{R}^{s\times s}$ the identity matrix, one has that (68) can be cast in vector form as

$$\begin{pmatrix} I_s - B_{11}t^{\alpha_1} & -B_{12}t^{\alpha_2} & \dots & -B_{1m}t^{\alpha_m} \\ -B_{21}t^{\alpha_1} & I_s - B_{22}t^{\alpha_2} & \dots & -B_{2m}t^{\alpha_m} \\ \vdots & & \ddots & \vdots \\ -B_{m1}t^{\alpha_1} & -B_{m2}t^{\alpha_2} & \dots & I_s - B_{mm}t^{\alpha_m} \end{pmatrix} \begin{pmatrix} \gamma^1 \\ \gamma^2 \\ \vdots \\ \gamma^m \end{pmatrix} = \begin{pmatrix} e_1 \otimes \psi_1 \\ e_1 \otimes \psi_2 \\ \vdots \\ e_1 \otimes \psi_m \end{pmatrix}. \tag{71}$$

Remark 7 Discretizing the integrals in (66) with a quadrature over k points (e.g., k Gauss-Piñeiro abscissae corresponding to the given $\{\alpha_i\}$) results into a FHBVM(k, s) method. In such a case, the matrices X_{ij} defined in (70) do coincide with the matrices X_{ij} defined in (60). Conversely, if a different Gauss-Jacobi quadrature is used for each equation, then X_{ij} reduce to X_{ij}^i, as defined in Remark 6.

Considering that the functions $\sigma_i(ct)$ in (63) can be rewritten as

$$\sigma_i(ct) = y_{i0} + \sum_{\nu=0}^{s-1} I^{\alpha_i} P_\nu^i(c) \left(t^{\alpha_i}\gamma_{i\nu}\right), \qquad c \in [0, 1], \qquad i = 1, \dots, m,$$

it makes sense to reformulate problem (71) in the equivalent form:

$$M_s^{\boldsymbol{\alpha}}(t) \begin{pmatrix} t^{\alpha_1}\gamma^1 \\ t^{\alpha_2}\gamma^2 \\ \vdots \\ t^{\alpha_m}\gamma^m \end{pmatrix} = \begin{pmatrix} e_1 \otimes \psi_1 \\ e_1 \otimes \psi_2 \\ \vdots \\ e_1 \otimes \psi_m \end{pmatrix}, \tag{72}$$

with $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)$ and

$$M_s^{\boldsymbol{\alpha}}(t) = \begin{pmatrix} t^{-\alpha_1}I_s - B_{11} & -B_{12} & \dots & -B_{1m} \\ -B_{21} & t^{-\alpha_2}I_s - B_{22} & \dots & -B_{2m} \\ \vdots & & \ddots & \vdots \\ -B_{m1} & -B_{m2} & \dots & t^{-\alpha_m}I_s - B_{mm} \end{pmatrix}. \tag{73}$$

Consequently,

$$\begin{pmatrix} t^{\alpha_1}\gamma^1 \\ t^{\alpha_2}\gamma^2 \\ \vdots \\ t^{\alpha_m}\gamma^m \end{pmatrix} = M_s^{\boldsymbol{\alpha}}(t)^{-1} \begin{pmatrix} e_1 \otimes \psi_1 \\ e_1 \otimes \psi_2 \\ \vdots \\ e_1 \otimes \psi_m \end{pmatrix}, \tag{74}$$

Considering that the r.h.s. in (72) does not depend on t, from (74) one deduces that a FHBVM(k, s) method can be called *A-stable* if, for all matrices $A = (a_{ij}) \in \mathbb{R}^{m\times m}$ and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_m)$ such that the solution of problem (62) is asymptotically stable, one has that matrix

$$M_s^{\boldsymbol{\alpha}}(t)^{-1} D_s, \qquad \text{with} \qquad D_s = \operatorname{diag}(u \otimes e_1), \qquad u = (1, \dots, 1)^\top \in \mathbb{R}^m,$$

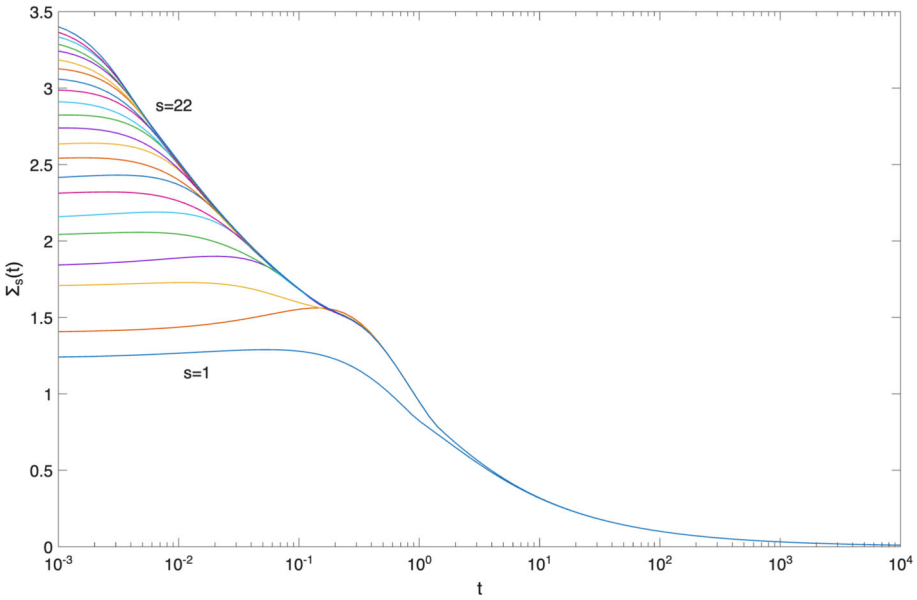

Fig. 1 Plot of the function $\Sigma_s(t)$ defined in (76), $s = 1, \ldots, 22$.

has a bounded norm for all $t > 0$ (the reason for introducing matrix D_s is due to the fact that, because of the structure of the r.h.s., only the columns $1, s+1, 2s+1, \ldots, (m-1)s+1$ of the inverse are actually involved). In this case, in fact, it follows that:

$$\exists\, K > 0\ s.t.\ \forall\, i = 1, \ldots, m,\ t > 0,\ \text{and}\ c \in [0,1] :\ |\sigma_i(ct)| \leq K.$$

As is clear, in general it is not possible to directly assess the A-stability of FHBVMs (as it has been done in the case of only one fractional order in [14]). Instead, we provide an example showing that FHBVM$(30, s)$, $s = 1, \ldots, 22$, behaves correctly when solving the linear problem

$$y_1^{(0.5)}(t) = 0.1 y_1(t) + y_2(t), \quad y_2^{(0.7)}(t) = -y_1(t) + 0.1 y_2(t), \quad t > 0, \qquad y_1(0) = y_2(0) = 1, \tag{75}$$

whose solution, according to the criteria given, e.g., in [23] (see also [10]), is asymptotically stable.[10] In more detail, in Figure 1 we plot the function

$$\Sigma_s(t) := \|(M_s^{(0.5,0.7)}(t))^{-1} D_s\|, \qquad t \in [10^{-3}, 10^4], \qquad s = 1, \ldots, 22. \tag{76}$$

From the plots, one deduces that all methods behave in the correct way.

6.2 Convergence Analysis

In the sequel (see also [14]), we shall consider the following meshes:

[10] In fact, in the multi-order case, the much simpler criterion in [48], valid in the single-order case, cannot be used.

- either a *graded mesh*, where

$$h_n = h_1 r^{n-1}, \qquad n = 1, \dots, N, \qquad \text{with} \qquad r > 1 \quad \text{s.t.} \quad h_1 \frac{r^N - 1}{r - 1} = T, \tag{77}$$

or
- a *uniform mesh*

$$h_n = h \equiv \frac{T}{N}, \qquad n = 1, \dots, N, \tag{78}$$

or
- a *mixed-mesh* where, for a suitably chosen $\rho \in \mathbb{N},\ \rho \geq 1$, and setting

$$r := \frac{\max\{2, \rho\}}{\max\{2, \rho\} - 1}, \tag{79}$$

one has, for a suitable $\mu \in \mathbb{N},\ \mu \geq 1$, and denoting $M := N - \mu + \rho$:

$$\begin{aligned} h_n &= h_1 r^{n-1}, \qquad n = 1, \dots, \mu, \qquad \text{with} \qquad h_1 \frac{r^\mu - 1}{r - 1} = \rho h, \\ h_n &= h \equiv \frac{T}{M}, \qquad n = \mu + 1, \dots, N. \end{aligned} \tag{80}$$

Of course, (80) reduces to either (78), when $\mu = \rho = 1$, or to (77), when $\mu = N > 1$. Moreover, as is clear [14]:

- (77) is appropriate when the vector field is nonsmooth at $t = 0$;
- (78) is appropriate when the vector field is suitably regular everywhere;
- (80) is appropriate when the vector field is nonsmooth at $t = 0$, but the solution becomes eventually periodic.

In the sequel we shall report a unified convergence analysis for a HBVM(k, s) method assuming, without loss of generality, that $T \leq 1$ in (1), in order to simplify the arguments.[11] By taking into account (5), (8), (12), (20), (23), and (28), and considering that, for all $n = 1, \dots, N$,

$$f_i(\sigma^n(ch_n)) = \sum_{j \geq 0} P_j^i(c) \gamma_{ij}(\sigma^n), \qquad c \in [0, 1], \qquad i = 1, \dots, \nu,$$

one obtains that the discrete solution exactly solves the sequence of perturbed problems

$$\begin{aligned} \sigma_{in}^{(\alpha_i)}(ch_n) &= f_i(\sigma^n(ch_n)) - \psi_{in}(ch_n), \qquad c \in [0, 1], \qquad n = 1, \dots, N, \\ \sigma_{i1}^{(\iota)}(0) &= y_{i0}^\iota \in \mathbb{R}^{m_i}, \qquad \iota = 0, \dots, \ell - 1, \qquad i = 1, \dots, \nu, \\ \sigma^n &= (\sigma_{1n}^\top, \dots, \sigma_{\nu n}^\top)^\top \in \mathbb{R}^m, \end{aligned} \tag{81}$$

in place of (setting $y_{in}(ch_n) = y_i(t_{n-1} + ch_n),\ c \in [0, 1]$):

$$\begin{aligned} y_{in}^{(\alpha_i)}(ch_n) &= f_i(y^n(ch_n)), \qquad c \in [0, 1], \qquad n = 1, \dots, N, \\ y_{i1}^{(\iota)}(0) &= y_{i0}^\iota \in \mathbb{R}^{m_i}, \qquad \iota = 0, \dots, \ell - 1, \qquad i = 1, \dots, \nu, \\ y^n &= (y_{1n}^\top, \dots, y_{\nu n}^\top)^\top \in \mathbb{R}^m. \end{aligned} \tag{82}$$

[11] Differently, a time scaling can be done.

In (81), we have denoted

$$\psi_{in}(ch_n) \;=\; \sum_{j\geq s} P_j^i(c)\gamma_{ij}(\sigma^n) \;+\; \sum_{j=0}^{s-1} P_j^i(c)\left[\gamma_{ij}(\sigma^n) - \gamma_{ij}^n\right], \tag{83}$$

the perturbation term, which is given by the sum of the truncation error in the expansion of $f(\sigma^n(ch_n))$ and of the quadrature error in the approximation of the Fourier coefficients $\gamma_{ij}(\sigma^n)$ (see (12)). In this respect, hereafter we shall assume that k is large enough so that the quadrature implemented in the FHBVM(k, s) method has order at least $2s$. In such a case, following arguments similar to those in [11, Section 3], it can be shown that, under regularity assumptions on the functions f_i, one has that [12]

$$\Psi_{in} := \max_{c\in[0,1]} \|\psi_{in}(ch_n)\|, \tag{84}$$

satisfies

- for (77):
$$\Psi_{in} \;\leq\; \begin{cases} O(h_1^{\alpha_i}), & \text{if } n = 1, \\ O(h_n^s), & \text{otherwise;} \end{cases}$$
- for (78):
$$\Psi_{in} \;\leq\; O(h^s);$$
- for (80):
$$\Psi_{in} \;\leq\; \begin{cases} O(h_1^{\alpha_i}), & \text{if } n = 1, \\ O(h_n^s), & \text{if } n = 2, \ldots, \mu, \\ O(h^s), & \text{otherwise.} \end{cases}$$

Setting $\Psi^n = \max_i \Psi_{in}$,

$$e_{in}(ch_n) = \|y_{in}(ch_n) - \sigma_{in}(ch_n)\|, \qquad e^n(ch_n) = \max_i e_{in}(ch_n) \qquad c \in [0, 1],$$
$$n = 1, \ldots, N,$$

the piecewise functions

$$e(t_{n-1} + ch_n) \equiv e^n(ch_n), \qquad \Psi(t_{n-1} + ch_n) \equiv \Psi^n, \qquad c \in [0, 1], \qquad n = 1, \ldots, N,$$

$\alpha = \min_i \alpha_i$, L the maximum of the Lipschitz constants of the f_i, and considering that $\Gamma(x)^{-1} < 1.2$, for $x > 0$, one eventually arrives at the inequality:

$$e(t) \;\leq\; 1.2 \int_0^t (t-\tau)^{\alpha-1} \left[Le(\tau) + \Psi(\tau)\right] \mathrm{d}\tau, \qquad t \in [0, T].$$

We observe that, because of the choices (77)–(80) one deduces that:

$$\forall_n = 2, \ldots, N : h_{n-1} \leq h_n.$$

Consequently, we can assume that $\Psi(t)$ is non decreasing and therefore, by using the fractional version of the Gronwall Lemma [66, Corollary 2] (see also [65]), one deduces that:

$$e(t) \;\leq\; K \int_0^t (t-\tau)^{\alpha-1} \Psi(\tau) \mathrm{d}\tau, \tag{85}$$

[12] Hereafter, $\|\cdot\|$ will denote the infinity norm.

with $K = 1.2\, E_\alpha(1.2\Gamma(\alpha)LT^\alpha)$, being $E_\alpha(z) = \sum_{j\geq 0} \frac{z^j}{\Gamma(\alpha j+1)}$ the one-parameter Mittag-Leffler function. Furthermore, considering that [13]

$$\forall\, n = 1, 2, \ldots, N, \quad \text{and} \quad t \geq t_n \ : \ \int_{t_{n-1}}^{t_n} (t-\tau)^{\alpha-1} \mathrm{d}\tau \ \leq \ \frac{h_n^{\tilde{\alpha}}}{\tilde{\alpha}},$$

with

$$\tilde{\alpha} = \min\{\alpha, 1\}, \tag{86}$$

and taking into account that $\Psi(t)$ is piecewise constant, one obtains:

$$K \int_0^t (t-\tau)^{\alpha-1} \Psi(\tau) \mathrm{d}\tau \ \leq \ \frac{K}{\tilde{\alpha}} \sum_{k=1}^{N} h_k^{\tilde{\alpha}} \Psi^k.$$

Moreover, by considering that for the graded mesh (77) and the graded part in (80)

$$\sum_{k=1}^{n} h_k^{s+\tilde{\alpha}} = h_n^{s+\tilde{\alpha}} \sum_{k=0}^{n-1} r^{-k(s+\tilde{\alpha})} < \frac{h_n^{s+\tilde{\alpha}}}{1 - r^{-s-\tilde{\alpha}}},$$

whereas for the uniform mesh $nh^{s+\tilde{\alpha}} \leq h^{s+\tilde{\alpha}-1}$, from (84) one eventually obtains that:

$$\max_{0\leq t\leq T} |e(t)| \ \leq \ \begin{cases} O(h_1^{\alpha+\tilde{\alpha}}) + O(h_N^{s+\tilde{\alpha}}), & \text{for (77)}, \\ O(h^{s+\tilde{\alpha}-1}), & \text{for (78)}, \\ O(h_1^{\alpha+\tilde{\alpha}}) + O(h_\mu^{s+\tilde{\alpha}}) + O(h^{s+\tilde{\alpha}-1}), & \text{for (80)}. \end{cases} \tag{87}$$

In Section 8 we report a few numerical tests, to assess (86)–(87) in the case (77).

Remark 8 From (87) one concludes that one can achieve arbitrarily high-order methods (though, suitably choosing the initial timestep, in the case of the graded or mixed meshes): as matter of fact, if $s \gg 1$, a *spectrally accurate* solution can be gained, as it happens in the ODE case [2].

7 The Code `fhbvm2_2`

The Matlab© code `fhbvm2_2` implements a FHBVM$(k, 22)$ method, using the approach described in the previous sections, in the case $\nu \in \{1, 2\}$ in (1). The code may use one of the meshes (77)–(80) described in Section 6.2. Concerning the evaluation of the abscissae and the weights for the quadrature(s):

- when $\nu = 1$, $k = 22$ and the Jacobi abscissae and weights are computed by adapting the companion Matlab© code of [35];
- when $\nu = 2$, $k = 30$ and the Jacobi-Piñeiro abscissae and weights are computed by adapting the companion Matlab© code of [40].

Remark 9 We emphasize that the code `fhbvm2_2` currently handles only the cases $\nu = 1, 2$, because the software in [40] only implements the case of two weighting functions. On the other hand, the extension of such software is not trivial, and is beyond the scope of this paper. Consequently, when an extended version of the code in [40] will be available, we shall adapt the code `fhbvm2_2` accordingly, using the arguments explained in Sections 4 and 5.

[13] Recall that $h_n = t_n - t_{n-1}$ and, moreover, $(t - t_{n-1}) \leq T \leq 1$.

Concerning the nonlinear iteration:

- when $\nu = 1$, the more efficient *blended iteration* (58) is used (as an alternative to the fixed-point iteration (54), used whenever possible);
- when $\nu = 2$, the straight simplified-Newton iteration (59) is used (as an alternative to the fixed-point iteration (54), used whenever possible).

All the remaining implementation details are similar to those of the codes `fhbvm` and `fhbvm2`, as explained in [12] and [14], respectively. The new code `fhbvm2_2` can be downloaded at the same URL [67], where the previous codes are available as well.

The calling sequence of the code is

$$[\texttt{t}, \texttt{y}, \texttt{etim}] = \texttt{fhbvm2\_2}(\texttt{fun}, \texttt{y0}, \texttt{T}, \texttt{M}, \texttt{mu}, \texttt{rho}) \tag{88}$$

In input, with reference to (1) and (80):[14]

- `T` is the final time T;
- `M` contains the value M in (80) (this parameter is named `N`, in the actual interface of the code);
- `mu` contains μ (this parameter is named `nu` in the actual interface of the code);
- `rho` contains ρ (this parameter is named `n` in the actual interface of the code);
- `y0` is a $\ell \times m$ matrix with the initial conditions (row vectors);
- `fun` is the identifier of a function implementing the problem (1). In more detail:
  - `[alpha,m1] = fun()` returns α_1, α_2 and m_1. In case $\alpha_1 = \alpha_2$ or $\alpha_2 = 0$, m_1 is not used;
  - `fun(t,y)` returns the vector field (also in vector mode);
  - `fun(t,y,1)` returns the Jacobian matrix of f.

In output:

- `t` is the used mesh (11);
- `y` contains the computed approximations (row-wise);
- `etim` is the elapsed time (in sec).

8 Numerical Tests

Here, we report some comparisons among the following Matlab© codes, publicly available on the internet:

- `fde12` [33], implementing a second-order predictor-corrector method, based on fractional versions of the Adams methods, which can handle only the case $\nu = 1$ in (1). We shall consider the PECE implementation of the methods (`fde12`) and the one with 10 corrector iterations (`fde12-10`);
- `flmm2` [33], implementing fractional versions of the trapezoidal rule (`flmm2-1`), of the Newton-Gregory formula (`flmm2-2`), and of the BFD2 method (`flmm2-3`). This code can handle only the case $\nu = 1$ in (1), and will be used with the following parameters:
  - tolerance for the nonlinear iteration: `tol=1e-15`,

[14] Hereafter, in order to have the same calling sequence for both `fhbvm2` and `fhbvm2_2`, for this latter code, with reference to (88): `M` is renamed `N`, `rho` is renamed `n`, and `mu` is renamed `nu`. As matter of fact, when possible, the two codes will be used with the same input arguments. See [15] for more details.

  - maximum number of nonlinear iterations per step: `maxit=1000`;

- `fde_pi12_pc` [33], implementing a second-order product-integration with predictor-corrector, which can handle multi-order FDEs. As with `fde12`, we shall consider the PECE implementation, and that with 10 iterations of the corrector (`fde_pi12_pc-10`);
- `fde_pi2_im` [33], implementing a second-order implicit product-integration of trapezoidal type, which can handle multi-order FDEs. It is used with the same parameters `tol` and `itmax` considered for `flmm2`;
- `fhbvm` [12], based on a FHBVM(22,20) method, which can handle only the case $\nu = 1$ in (1);
- `fhbvm2` [14], based on a FHBVM(22,22) method, which can handle only the case $\nu = 1$ in (1);
- the code `fhbvm2_2` here described, based on either a FHBVM(22,22) or a FHBVM(30,22) method, as explained above, which can handle the cases $\nu = 1$ and $\nu = 2$ in (1). In the case $\nu = 1$, it will be used with the same parameters used for `fhbvm2`.

We refer to [15] for a quick introduction to the codes `fde12`, `flmm2`, `fhbvm`, and `fhbvm2`. Moreover, the code `fde_pi12_pc` is a kind of multi-order extension of `fde12`, and `fde_pi2_im` is a kind of multi-order extension of `flmm2`, both couples of codes having a similar interface.[15]

All numerical tests have been done on a 12-core M4-pro Silicon based computer with 64GB of memory, using Matlab© Rel. 2025a.

When necessary, comparisons among the codes, on a given test problem, will be done by using a corresponding *work-precision diagram (WPD)*, where the execution time (measured in sec) is plotted againts accuracy, measured in terms of *mixed-error significant computed digits (mescd)*, defined as follows:

$$\text{mescd} \;=\; \max\left\{0,\; -\log_{10} \max_n \|(y(t_n) - y_n)./(1 + |y(t_n)|)\|\right\}, \tag{89}$$

with $y_n \approx y(t_n)$, this latter the reference solution, $|\cdot|$ the vector of the absolute values, and $./$ the component-wise division. As is clear, higher and higher accuracies are obtained by suitably tuning the parameters of the various codes.

The problems with $\nu = 1$ are taken from the FDE-Testset [15]: in particular, at the URL [68] Matlab© codes implementing them, and providing reference solutions, are available.

Remark 10 For the sake of completeness, we recall that in the definition (89) the absolute error $|y(t_n) - y_n|$ is scaled, component-wise, by $(1 + |y(t_n)|)$, so that each entry of the vector $|y(t_n) - y_n|./(1 + |y(t_n)|)$ is approximately the absolute error, if the corresponding entry of $|y(t_n)|$ is much smaller than 1, or the relative error, if much larger that 1. This measure is widely used in the numerical comparison of numerical codes (see, e.g., the Test-Set for IVP Solvers [70]).

Problem 1

This is a *stiffly oscillatory* linear problem with only one fractional derivative. It is taken from [15, Problem 4] and, at the URL [68], a corresponding Matlab© code, implementing the

[15] These codes are available at the URL [69].

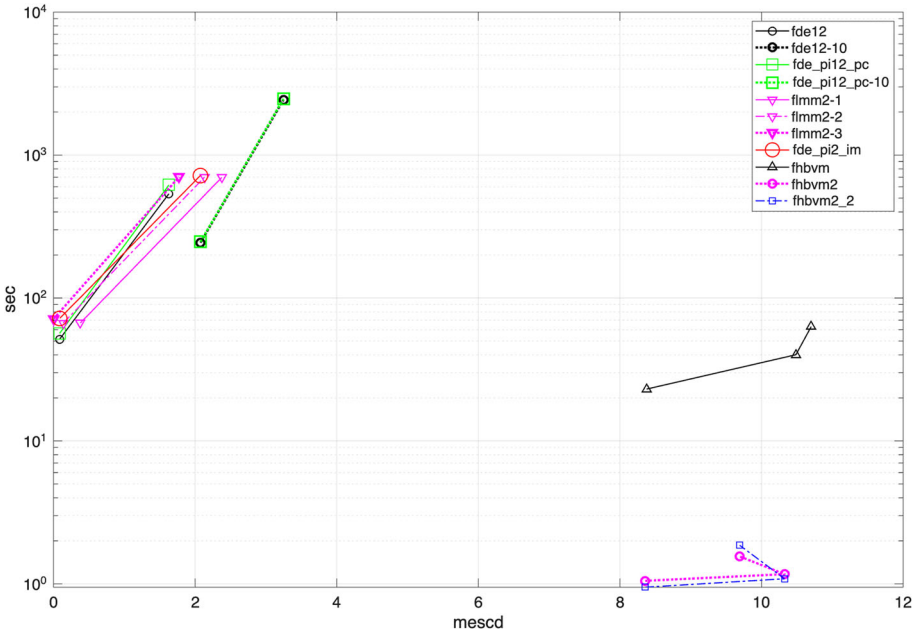

Fig. 2 WPD for Problem (90).

problem and providing the reference solution, is available:

$$y^{(0.5)} = \frac{1}{8}\begin{pmatrix} 41 & 41 & -38 & 40 & -2 \\ -79 & 81 & 2 & 0 & -2 \\ 20 & -60 & 20 & -20 & -8 \\ -22 & 58 & -24 & 20 & -4 \\ 1 & 1 & -2 & -4 & -2 \end{pmatrix} y, \qquad t \in [0, 20], \qquad y(0) = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}. \tag{90}$$

Based on [15], the following parameters are used for the various codes, to construct the corresponding WPD:

- `fde12`, `fde12-10`, `fde_pi12_pc`, `fde_pi12_pc-10`: $\mathtt{h} = 10^{-i}$, $i = 5, 6$;
- `flmm2-1`, `flmm2-2`, `flmm2-3`, `fde_pi2_im`: $\mathtt{h} = 10^{-i}$, $i = 4, 5$;
- `fhbvm`: $\mathtt{M} = 100i$, $i = 3, 4, 5$;
- `fhbvm2`, `fhbvm2_2`: $\mathtt{nu} = 50, \mathtt{n} = 1, \mathtt{N} = 100i$, $i = 3, 4, 5$.

The corresponding WPD is reported in Figure 2, from which one obtains that:

- `fde12` and `fde_pi12_pc` reach an accuracy of less than 2 mescd in more than 500 sec. Increasing the number of corrector iterations improves accuracy to about 3 mescd, but with a much higher execution time (about 2500 sec) for both codes;
- `flmm2` (all versions) and `fde_pi2_im` can achieve about 2 mescd in about 800 sec;
- `fhbvm` is the most accurate code, with about 11 mescd, with an execution time of about 60 sec;
- `fhbvm2` and `fhbvm2_2` do have a comparable performance (over 10 mescd in about 1 sec), as expected, due to the oscillatory nature of the solution [15].

Table 3 Convergence results for FHBVM(30, s) used with a graded mesh with $h_N = 2/N$

N	$s=2$		$s=3$		$s=4$	
	error	order	error	order	error	order
10	2.88e-02	–	4.36e-04	–	2.07e-06	–
20	6.89e-03	2.06	5.08e-05	3.10	1.17e-07	4.14
40	1.57e-03	2.13	5.68e-06	3.16	6.41e-09	4.19
80	3.48e-04	2.17	6.21e-07	3.19	3.43e-10	4.22

Problem 2

The next problem is a multi-order variant of [15, Problem 7]:

$$\begin{aligned} y_1^{(\alpha_1)} &= s(t,\alpha_2,\beta)^2 - y_2^2 + g(t,\alpha_1,\beta), \\ y_2^{(\alpha_2)} &= -s(t,\alpha_1,\beta)^2 + y_1^2 + g(t,\alpha_2,\beta), \qquad t \in [0,2], \\ y_1(0) &= y_2(0) = 1, \end{aligned} \tag{91}$$

having solution $y_i(t) = s(t,\alpha_i,\beta)$, $i = 1, 2$, with

$$s(t,\alpha,\beta) = (1-t^2)^2 + 4t^\alpha + (2 - 3t^{0.2})t^{\alpha+\beta},$$

and

$$g(t,\alpha,\beta) = \frac{24t^{4-\alpha}}{\Gamma(5-\alpha)} - \frac{4t^{2-\alpha}}{\Gamma(3-\alpha)} - 3t^{0.2+\beta}\frac{\Gamma(1.2+\alpha+\beta)}{\Gamma(1.2+\beta)} + 2t^\beta \frac{\Gamma(1+\alpha+\beta)}{\Gamma(1+\beta)} + 4\Gamma(1+\alpha).$$

We shall consider the values $\alpha_1 = 0.2$, $\alpha_2 = 0.4$, $\beta = 0.1$. The solution, and the corresponding vector field, are depicted in Figure 3.

The following parameters are used for the codes `fde12`, `flmm2`, and `fhbvm2_2`, which are the only ones that can be used, in this case:

- `fde_pi12_pc`, `fde_pi12_pc-10`, `fde_pi2_im`: $\mathtt{h} = 10^{-i}$, $i = 5, 6, 7$;
- `fhbvm2_2`: $\mathtt{n} = 2$, $\mathtt{nu} = 100$, $\mathtt{N} = 5i$, $i = 2, 3, 4, 5, 6$.

The obtained results are reported in the WPD in Figure 4, from which one may infer that:

- `fde_pi12_pc` and `fde_pi12_pc-10` can reach an accuracy of about 3 mescd in about 250 sec and 700 sec, respectively. Consequently, increasing the corrector iterations does not improve accuracy but only affects the execution time;
- `fde_pi2_im` reaches the same accuracy of about 3 mescd in about 10^4 sec;
- `fhbvm2_2` reaches full machine accuracy (more than 14 mescd) in about 0.3 sec.

We use problem (91) also for assessing the convergence order (87), when using a graded mesh (77). For this purpose, we use a modification of the code `fhbvm2_2` implementing a FHBVM(30, s) method. We use the parameters $\mathtt{n} = \mathtt{N} = N$, so that $h_N = 2/N$, whereas the parameter `nu` is chosen so that $h_N^{s+\alpha} = h_1^{2\alpha}$. In fact, in such case (see (86)) $\tilde{\alpha} = \alpha = 0.2$, and the upper bound in (87) (graded case) turns out to be optimal. In doing this, we also consider that the parameter r defining the graded mesh is given by (79), i.e., $r = N/(N-1)$. As one may see, the numerical results listed in Table 3 are in good agreement with (87), since the convergence order appears to be approximately given by $s + 0.2$, as expected, for the considered values $s = 2, 3, 4$.

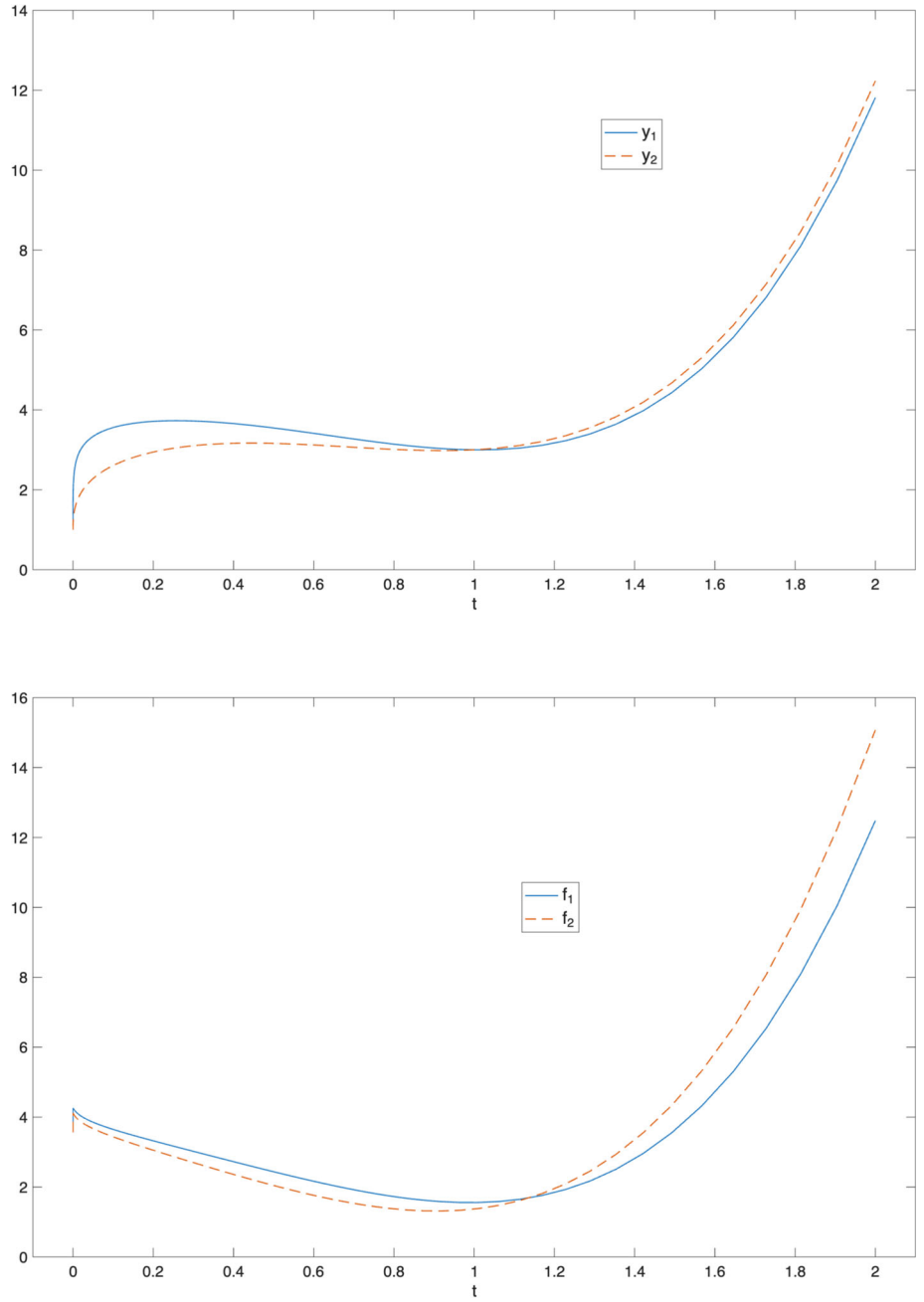

Fig. 3 Solution (upper-plot) and vector field (lower-plot) for Problem (91).

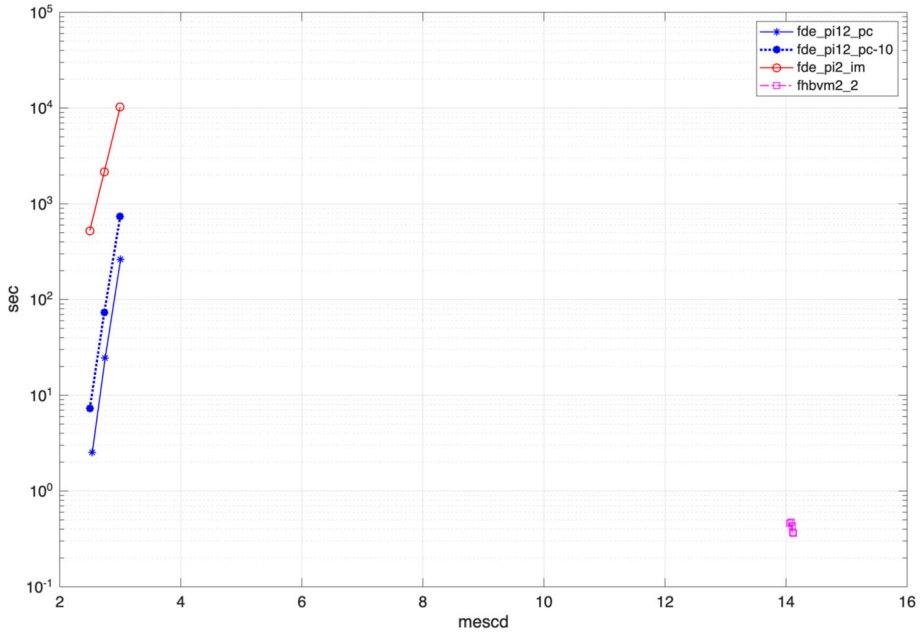

Fig. 4 WPD for Problem (91).

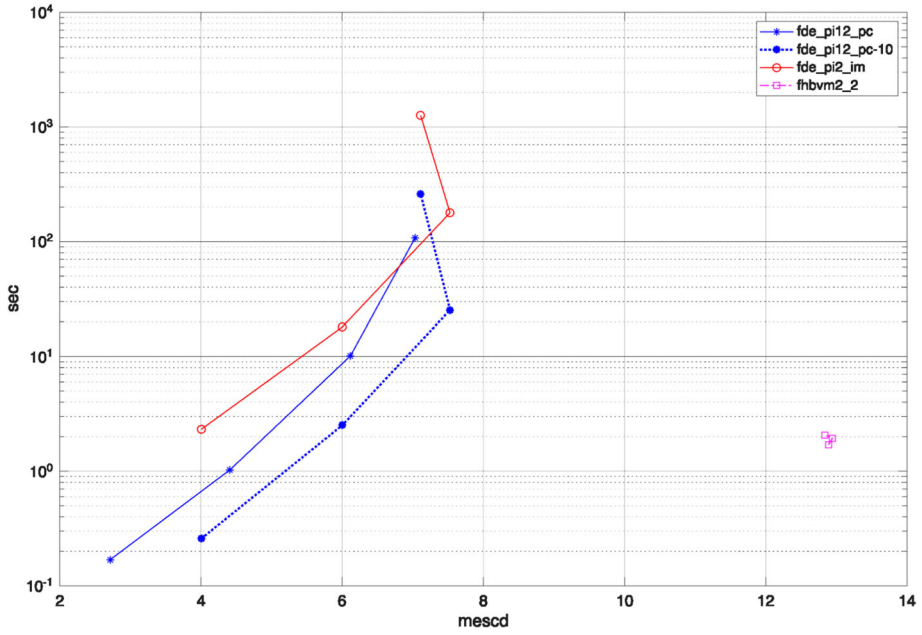

Fig. 5 WPD for Problem (92), $\alpha_1 = 0.7, \alpha_2 = 0.8$.

Problem 3

This problem is a multi-order fractional version of the Brusselator problem, adapted from [33, Problem (32)]:

$$\begin{aligned} y_1^{(\alpha_1)} &= A - (B+1)y_1 + y_1^2 y_2, \\ y_2^{(\alpha_2)} &= B y_1 - y_1^2 y_2, \qquad\qquad t \in [0, 100], \\ y_1(0) &= 1.2, \qquad y_2(0) = 2.8, \end{aligned} \tag{92}$$

where we use the same parameters as in [33]: $A = 1$, $B = 3$, $\alpha_1 = 0.8$, and $\alpha_2 = 0.7$. The solution is depicted in Figure 6, both in the phase space and versus time, from which one infers that it approaches a limit cicle, thus becoming of periodic type: in the first plot, the circle denotes the initial point of the trajectory.
The following parameters are used for the codes `fde_pi12_pc`, `fde_pi2_im`, and `fhbvm2_2`, which are the only ones that can be used, in this case:

- `fde_pi12_pc`, `fde_pi12_pc-10`, `fde_pi2_im`: $\mathtt{h} = 10^{-i}$, $i = 2, 3, 4, 5$;
- `fhbvm2_2`: $\mathtt{n} = 1$, $\mathtt{nu} = 50$, $\mathtt{N} = 50i$, $i = 4, 5, 6$.

The obtained results are reported in Figure 5, where the accuracy w.r.t. a reference solution at the end point is reported. This latter has been evaluated numerically by using `fhbvm2_2` on 3 consecutive doubled meshes, resulting into:

$$y_1(100) \approx 1.706502172199, \qquad y_2(100) \approx 1.940414058005.$$

From the obtained results, one concludes that:

- `fde_pi12_pc-10` reaches an accuracy of 7.5 mescd in about 25 sec, whereas `fde_pi12_pc` reaches about 7 mescd in 100 sec;
- `fde_pi2_im` reaches an accuracy of 7.5 sec in approximately 180 sec;
- `fhbvm2_2` reaches 13 mescd in less than 2 sec, independently of the stepsize used, as one expects from a spectrally accurate method.

Problem 4

As stressed in Section 5.1, the *blended iteration*, which can be used in the case $\nu = 1$ in (1), only requires to factor one matrix having the same size as that of the continuous problem. It has, therefore, a lower computational cost, w.r.t. the simplified Newton iteration, to be used in the case $\nu > 1$. The next example is aimed at assessing this fact. For this purpose, we shall use the code `fhbvm2_2` for solving problem (92) with the same parameters $A = 1$ and $B = 3$ used before, but with α_1, and α_2 chosen as follows:

- $\alpha_1 = \alpha_2 = 0.7$;
- $\alpha_1 = 0.7$, $\alpha_2 = 0.7 + \epsilon$, with $\epsilon = 10^{-4}$.

As is clear, in the first case only one fractional derivative order occurs (i.e., $\nu = 1$ in (1)), whereas they are two in the second case (i.e., $\nu = 2$ in (1)). Despite this fact, the computed solutions are very similar, as is shown in Figure 7. Consequently, we shall assume that the code provides a comparable accuracy, when used with the same parameters, in both cases. In particular, similarly as done for Problem 3, we shall use the parameters $\mathtt{n} = 1$, and $\mathtt{nu} = 50$. Instead, the values of `N` vary as reported in Table 4, where we also list:

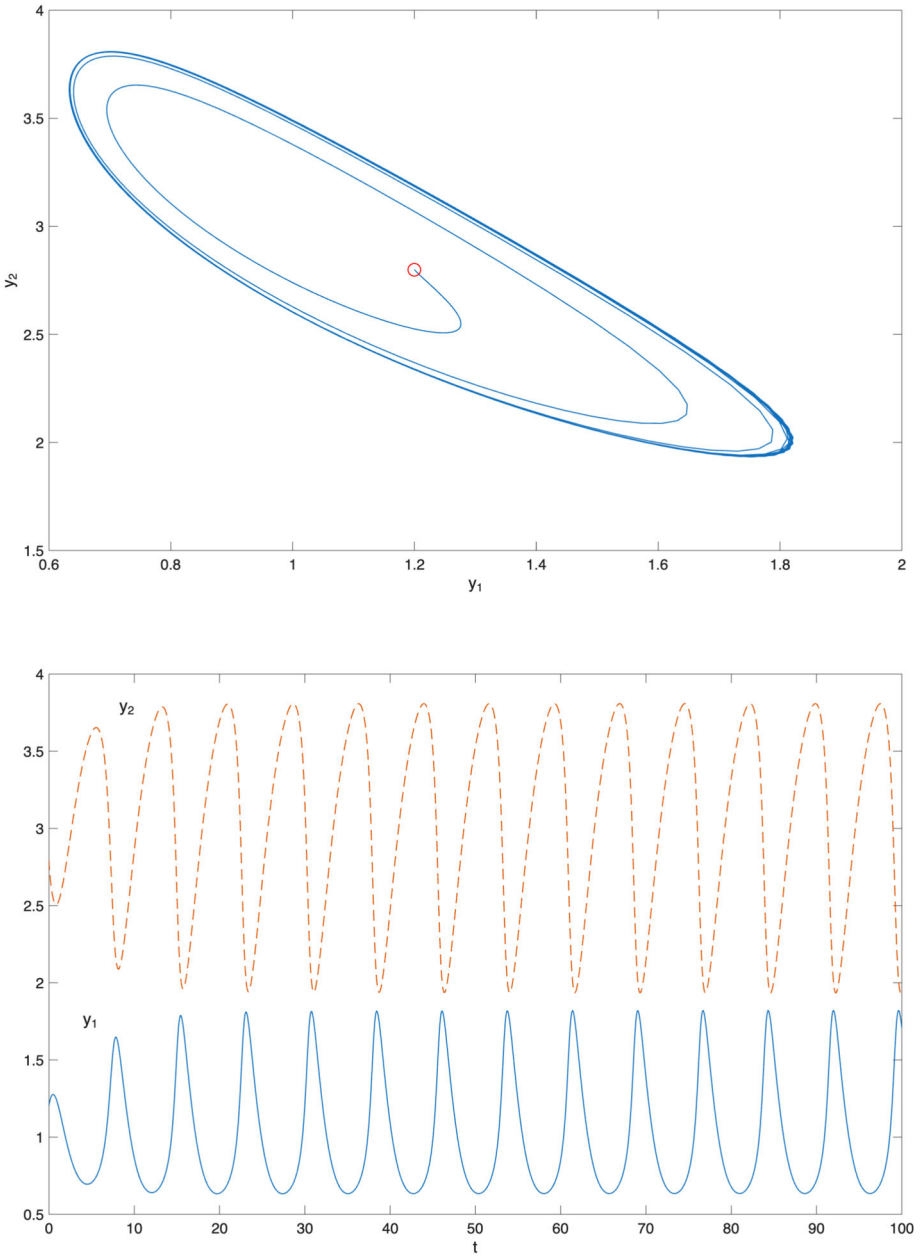

Fig. 6 Solution in the phase space (upper-plot) and versus time (lower-plot) for Problem (92).

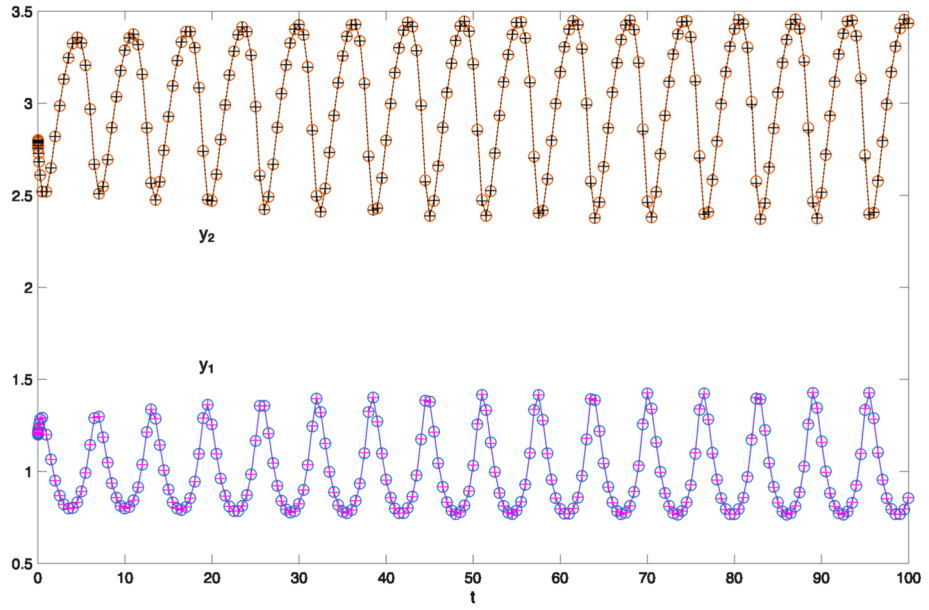

Fig. 7 Solution of Problem (92), $\alpha_1 = \alpha_2 = 0.7$ (circles) and $\alpha_1 = 0.7, \alpha_2 = 0.7 + 10^{-4}$ (crosses).

- the required number of fixed-point iterations (54) (basically, used in the initial graded mesh, where the timesteps are very small);
- the number of blended iterations (58), in the case $\nu = 1$, or simplified Newton iterations (59), in the case $\nu = 2$;
- the execution times (in sec).

As expected, the execution times, when using the blended iteration, are lower than those required by the simplified Newton iteration, despite the fact that the latter requires less steps to reach convergence (approximately 30% less). In particular, when using the blended iteration, the code is more than 3.5 times faster than when using the simplified Newton iteration.

Problem 5

At last, we consider the following multi-order fractional version of the predator-prey model described in [52], including intra-species competition:

$$\begin{aligned} y_1^{(\alpha_1)} &= r_1 y_1 - a_{11} y_1^2 - a_{12} y_1 y_2 - a_{13} y_1 y_3, \\ y_2^{(\alpha_2)} &= a_{21} y_1 y_2 - a_{22} y_2^2 - \frac{a_{23} y_2 y_3}{1 + \beta y_2} - r_2 y_2, \qquad t \in [0, T], \\ y_3^{(\alpha_2)} &= a_{31} y_1 y_3 + \frac{a_{32} y_2 y_3}{1 + \beta y_2} - a_{33} y_3^2 - r_3 y_3, \qquad y(0) = y^0 \in \mathbb{R}^3. \end{aligned} \tag{93}$$

Here:

- y_1 represents the density of the preys;
- y_2 is the density of the *intermediate* predators, which cannot prey the *top* predators;

Table 4 Execution times (in sec) and nonlinear iterations for `fhbvm2_2`, when used for solving (92) with $\alpha_1 = \alpha_2 = 0.7$ (case 1) and $\alpha_1 = 0.7$, $\alpha_2 = 0.7 + 10^{-4}$ (case 2).

	case 1			case 2		
N	Fixed-point	Blended	Execution	Fixed-point	Newton	Execution
200	150	2679	0.40	142	1950	1.51
250	149	3193	0.45	141	2238	1.65
300	146	3644	0.54	146	2529	2.05
350	146	4115	0.64	146	2822	2.37
400	152	4627	0.73	144	3084	2.80
450	151	5116	0.85	143	3368	3.14
500	151	5617	0.99	143	3667	3.54
550	150	5969	1.07	150	3934	3.83
600	148	6423	1.19	148	4146	4.22

- y_3 denotes the density of the *top* predators, which can prey also the *intermediate* ones.

We use the following parameters: $\alpha_1 = 0.99, \alpha_2 = 0.8; r_1 = 5, r_2 = 1, r_3 = 0.1; a_{11} = 0.01$, $a_{12} = 1$, $a_{13} = 35$, $a_{21} = 1$, $a_{22} = 0.2$, $a_{23} = 1$, $a_{31} = 0.1$, $a_{32} = 1$, $a_{33} = 0.3$; $\beta = 0.01$. Moreover, we use the initial point $y^0 = (0.7,\ 0.2,\ 0.1)^\top$, and integrate up to $T = 500$: as one may infer from the plots in Figure 8, the solution tends to become of periodic type, with a period $\tau \approx 11.8$. Since a reference solution is not available for this problem, to obtain corresponding accuracy results we run each code on consecutive doubled meshes, as below specified:

- `fde_pi12_pc`, `fde_pi12_pc-10`, `fde_pi2_im`: $\mathtt{h} = 10^{-2} \cdot 2^{1-\ell}$, $\ell = 1, \ldots, 7$;
- `fhbvm2_2`: $\mathtt{n} = 1$, $\mathtt{nu} = 50$, $\mathtt{N} = 500 \cdot 2^{\ell-1}$, $\ell = 1, 2, 3, 4$.[16]

Then, for each code, the accuracy on the ℓth mesh is estimated through the solution on the subsequent $(\ell + 1)$st mesh. The obtained results are listed in Table 5, from which one may conclude that:

- `fde_pi12_pc` reaches 3.5 mescd in about 24 sec;
- `fde_pi12_pc-10` and `fde_pi2_im` do have a comparable accuracy, and can reach a higher accuracy (slightly larger than 4 mescd) but with a higher execution time. Moreover, that of `fde_pi2_im` is much larger than that of `fde_pi12_pc-10`;
- `fhbvm2_2` reaches a very high accuracy (almost 12 mescd) in about 22 sec. Moreover, it is able to achieve more than 10 mescd by using a timestep as large as 1, in the uniform part of the mesh: this reflects the capability, of the underlying FHBVM(30,22) method, of obtaining a spectrally accurate solution.

Remark 11 It is worth mentioning that the considered large width of the integration interval ($T = 500$) of problem (93) confirms the good stability properties of the FHBVM(30,22) method implemented in the code `fhbvm2_2`. However, to further confirm this fact, we consider $T = 5000$ and, by using the coarsest mesh, among those used for the numerical tests ($\mathtt{n} = 1$, $\mathtt{nu} = 50$, $\mathtt{N} = T$), we obtain the plot in Figure 9, which still exhibits the correct periodic behavior.

[16] Consequently, the timestep in the uniform part of the mesh is $h = 2^{1-\ell}$, $\ell = 1, 2, 3, 4$.

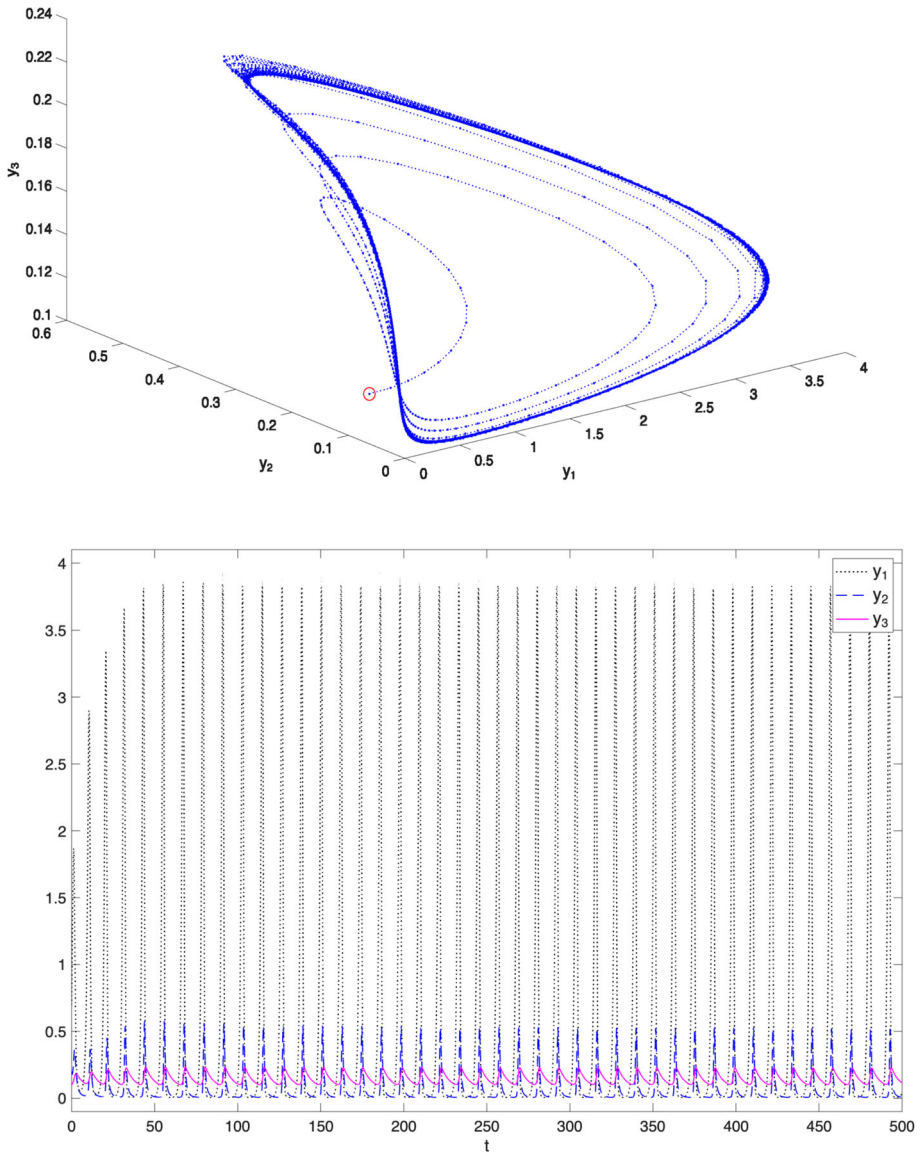

Fig. 8 Solution of Problem (93) in the phase space (upper plot) and versus time (lower plot). In the upper plot, the circle denotes the initial point of the trajectory.

9 Conclusions

In this paper, we have extended the class of methods named *Fractional HBVMs (FHBVMs)* for numerically solving multi-order FDEs. The implementation details of the methods have been given in the general case: in particular, a comprehensive review of Jacobi-Pinẽiro quadratures has been presented, with the aim of providing algorithmic insights. As a result, a new Matlab© code, named `fhbvm2_2`, has been released. The code implements the case of 2

Table 5 Estimated accuracy (in mescd) and execution times when solving (93) on consecutive doubled meshes (see text for details).

	`fde_pi12_pc`		`fde_pi12_pc-10`		`fde_pi2_im`		`fhbvm2_2`	
ℓ	mescd	sec	mescd	sec	mescd	sec	mescd	sec
1	0.59	0.76	1.83	2.21	1.83	21.41	10.22	4.41
2	1.16	1.54	2.43	4.40	2.43	40.73	11.35	9.29
3	1.77	3.13	3.03	8.91	3.03	78.52	11.68	22.50
4	2.40	6.30	3.64	18.15	3.64	161.37		
5	3.09	12.12	4.22	36.71	4.22	309.53		
6	3.55	24.16	4.06	74.04	4.06	596.44		

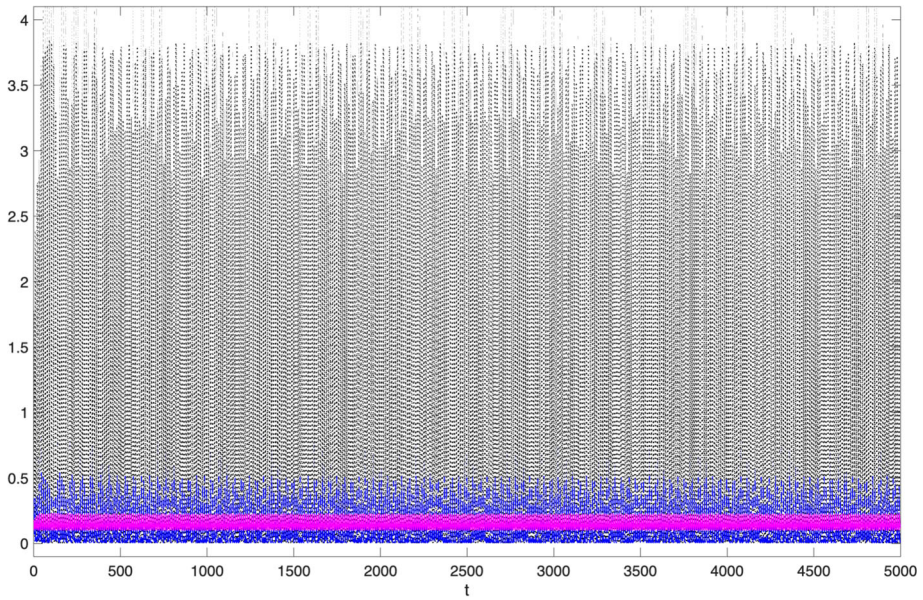

Fig. 9 Numerical solution of Problem (93), $T = 5000$ by using timestep $h = 1$ (except the first 50 initial steps).

possible different fractional orders, and relies on either FHBVM(30,22) or FHBVM(22,22): this latter method had already been used in the code `fhbvm2`, handling the single-order case. Alike the latter code, also the new one is able to gain spectrally accurate solutions, on a suitably chosen mesh. Numerical tests taken from the literature duly confirm this fact, also showing its superior performance over existing numerical codes, designed for solving multi-order FDEs, available on the internet. We plan to extend the code `fhbvm2_2` to the case of more fractional orders, as soon as an effective code for computing the Jacobi-Piñeiro weights and abscissae, in the general case, will be available.

Acknowledgements The first three authors are members of the "Gruppo Nazionale per il Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM)". The last author is supported by the project n.ro PUTJD1275 of the Estonian Research Council. The paper has been written during a period of visit of the last author at the University of Florence. The authors wish to thank the reviewers, for their helpful comments.

Funding Open access funding provided by Università degli Studi di Firenze within the CRUI-CARE Agreement.

Data Availability The Matlab© code `fhbvm2_2` is available at the URL [67].

Declarations

Conflict of interest The authors declare no conflict of interests.

References

1. Ahmed, A.I., Al-Ahmary, T.A.: Fractional-order Chelyshkov collocation method for solving systems of fractional differential equations. Math. Probl. Eng. **2022**, 4862650 (2022). https://doi.org/10.1155/2022/4862650
2. Amodio, P., Brugnano, L., Iavernaro, F.: Analysis of Spectral Hamiltonian Boundary Value Methods (SHBVMs) for the numerical solution of ODE problems. Numer. Algorithms **83**, 1489–1508 (2020). https://doi.org/10.1007/s11075-019-00733-7
3. Amodio, P., Brugnano, L., Iavernaro, F.: (Spectral) Chebyshev collocation methods for solving differential equations. Numer. Algorithms **93**, 1613–1638 (2023). https://doi.org/10.1007/s11075-022-01482-w
4. Ansari, S., Akrami, M.H.: Fractional Jacobi-Picard iteration method using Gauss-Seidel technique for solving a system of nonlinear fractional differential equations. Alexandria Eng. J. **108**, 261–272 (2024). https://doi.org/10.1016/j.aej.2024.07.091
5. Arciga-Alejandre, M.P., Sanchez-Ortiz, J., Ariza-Hernandez, F.J., Catelan-Angeles, G.: A multi-stage homotopy perturbation method for the fractional Lotka-Volterra model. Symmetry **11**, 1330 (2019). https://doi.org/10.3390/sym11111330
6. Atabakzadeh, M.H., Akrami, M.H., Erjaee, G.H.: Chebyshev operational matrix method for solving multi-order fractional ordinary differential equations. Appl. Math. Model. **37**, 8903–8911 (2013). https://doi.org/10.1016/j.apm.2013.04.019
7. Baleanu, D., Sajjadi, S.S., Asad, J.H., Jajarmi, A., Estiri, E.: Hyperchaotic behaviors, optimal control, and synchronization of a nonautonomous cardiac conduction system. Adv. Difference Equ. (2021) paper no. 157. https://doi.org/10.1186/s13662-021-03320-0
8. Borges, C.F.: On a class of Gauss-like quadrature rules. Numer. Math. **67**, 271–288 (1994). https://doi.org/10.1007/s002110050028
9. Bhrawy, A.H., Zaky, M.A.: Shifted fractional-order Jacobi orthogonal functions: Application to a system of fractional differential equations. Appl. Math. Model. **40**, 832–845 (2016). https://doi.org/10.1016/j.apm.2015.06.012
10. Brandibur, O., Garrappa, R., Kaslik, E.: Stability of Systems of Fractional-Order Differential Equations with Caputo Derivatives. Mathematics **9**(8), 914 (2021). https://doi.org/10.3390/math9080914
11. 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. **99**, 48 (2024). https://doi.org/10.1007/s10915-024-02517-1
12. Brugnano, L., Gurioli, G., Iavernaro, F.: Numerical solution of FDE-IVPs by using Fractional HBVMs: the fhbvm code. Numer. Algor. **99**, 463–489 (2025).https://doi.org/10.1007/s11075-024-01884-y
13. Brugnano, L., Gurioli, G., Iavernaro, F.: Solving FDE-IVPs by using Fractional HBVMs: Some experiments with the fhbvm code. J. Comput. Methods Sci. Eng. **25**(1), 1030–1038 (2025). https://doi.org/10.1177/14727978251321328

14. Brugnano, L., Gurioli, G., Iavernaro, F., Vikerpuur, M.: Analysis and implementation of collocation methods for fractional differential equations. J. Sci. Comput. **104**, 92 (2025). https://doi.org/10.1007/s10915-025-03006-9
15. Brugnano, L., Gurioli, G., Iavernaro, F., Vikerpuur, M.: FDE-Testset: comparing Matlab codes for solving fractional differential equations of Caputo type. Fractal Fract. **9**(5), 312 (2025). https://doi.org/10.3390/fractalfract9050312
16. Brugnano, L., Iavernaro, F.: Line Integral Methods for Conservative Problems. CRC Press, Boca Raton, FL (2016)
17. Brugnano, L., Iavernaro, F., Trigiante, D.: A note on the efficient implementation of Hamiltonian BVMs. J. Comput. Appl. Math. **236**, 375–383 (2011). https://doi.org/10.1016/j.cam.2011.07.022
18. Brugnano, L., Magherini, C.: Blended Implementation of Block Implicit Methods for ODEs. Appl. Numer. Math. **42**, 29–45 (2002). https://doi.org/10.1016/S0168-9274(01)00140-4
19. Chen, Y., Liu, F., Yu, Q., Li, T.: Review of fractional epidemic models. Appl. Math. Model. **97**, 281–307 (2021). https://doi.org/10.1016/j.apm.2021.03.044
20. Chen, Y., Ke, X., Wei, Y.: Numerical algorithm to solve system of nonlinear fractional differential equations based on wavelets method and the error analysis. Appl. Math. Comput. **251**, 475–488 (2015). https://doi.org/10.1016/j.amc.2014.11.079
21. Coussement, J., Van Assche, W.: Gaussian quadrature for multiple orthogonal polynomials. J. Compt. Appl. Math. **178**, 131–145 (2005). https://doi.org/10.1016/j.cam.2004.04.016
22. Dabiri, A., Butcher, E.A.: Stable fractional Chebyshev differentiation matrix for the numerical solution of multi-order fractional differential equations. Nonlinear Dyn. **90**, 185–201 (2017). https://doi.org/10.1007/s11071-017-3654-3
23. Deng, W., Li, C., Lü, J.: Stability analysis of linear fractional differential system with multiple time delays. Nonlinear Dyn. **48**, 409–416 (2007). https://doi.org/10.1007/s11071-006-9094-0
24. Diethelm, K.: The Analysis of Fractional Differential Equations. An Application-oriented Exposition using Differential Operators of Caputo Type. Lecture Notes in Math., Springer, Berlin, 2010. https://doi.org/10.1007/978-3-642-14574-2
25. Diethelm, K., Ford, N.J., Freed, A.D.: Detailed error analysis for a fractional Adams method. Numer. Algorithms **36**, 31–52 (2004). https://doi.org/10.1023/B:NUMA.0000027736.85078.be
26. Diethelm, K., Ford, N.J., Freed, A.D., Luchko, Y.: Algorithms for the fractional calculus: a selection of numerical methods. Comput. Methods Appl. Mech. Eng. **194**(6–8), 743–773 (2005). https://doi.org/10.1016/j.cma.2004.06.006
27. Diethelm, K., Garrappa, R., Stynes, M.: Good (and Not So Good) Practices in Computational Methods for Fractional Calculus. Mathematics **8**, 324 (2020). https://doi.org/10.3390/math8030324
28. Ertürk, V.S., Momani, S.: Solving systems of fractional differential equations using differential transform method. J. Comput. Appl. Math. **215**, 142–151 (2008). https://doi.org/10.1016/j.cam.2007.03.029
29. Faghih, A., Mokhtary, P.: Non-linear System of Multi-Order Fractional Differential Equations: Theoretical Analysis and a Robust Fractional Galerkin Implementation. J. Sci. Comput. **91**, 35 (2022). https://doi.org/10.1007/s10915-022-01814-x
30. Ford, N.J., Pedas, A., Vikerpuur, M.: High order approximations of solutions to initial value problems for linear fractional integro-differential equations. Fract. Calc. Appl. Anal. **26**, 2069–2100 (2023). https://doi.org/10.1007/s13540-023-00186-9
31. Garrappa, R.: Trapezoidal methods for fractional differential equations: theoretical and computational aspects. Math. Comput. Simul. **110**, 96–112 (2015). https://doi.org/10.1016/j.matcom.2013.09.012
32. Garrappa, R.: Numerical evaluation of two and three parameter Mittag-Leffler functions. SIAM Jour. Numer. Anal. **53**(3), 1350–1369 (2015). https://doi.org/10.1137/140971191
33. Garrappa, R.: Numerical solution of fractional differential equations: a survey and a software tutorial. Mathematics **6**(2), 16 (2018). https://doi.org/10.3390/math6020016
34. Gautschi, W.: Orthogonal polynomials: applications and computations. Acta Numer **5**, 45–119 (1996). https://doi.org/10.1017/S0962492900002622
35. Gautschi, W.: Orthogonal Polynomials Computation and Approximation. Oxford University Press (2004)
36. Grigorenko, I., Grigorenko, E.: Chaotic dynamics of the fractional Lorenz system. Phys. Rev. Letters **91**, 034101 (2003). https://doi.org/10.1103/PhysRevLett.91.034101
37. Henry, B.I., Langlands, T.A.M.: Fractional cable models for spiny neuronal dendrites. Phys. Rev. Letts. **100**(12), 128103 (2008). https://doi.org/10.1103/PhysRevLett.100.128103
38. Hori, M., Fukunaga, I., Masutani, V., Taoka, T., Kamagata, K., Suzuki, Y., et al.: Visualizing non-Gaussian diffusion: clinical application of q-space imaging and diffusional kurtosis imaging of the brain, and spine. Magn. Reson. Med. Sc. **11**, 221–233 (2012). https://doi.org/10.2463/mrms.11.221

39. Koh, C.G., Kelly, J.M.: Application of fractional derivatives to seismic analysis of base-isolated models. Earthquake engineering & structural dynamics **19**(2), 229–241 (1990). https://doi.org/10.1002/eqe.4290190207
40. Laudadio, T., Mastronardi, N., Van Assche, W., Van Dooren, P.: A Matlab package computing simultaneous Gaussian quadrature rules for multiple orthogonal polynomials. J. Compt. Appl. Math. **451**, 116109 (2024). https://doi.org/10.1016/j.cam.2024.116109
41. Laudadio, T., Mastronardi, N., Van Dooren, P.: Computational aspects of simultaneous Gaussian quadrature. Numer. Algorithms **100**, 621–643 (2025). https://doi.org/10.1007/s11075-024-01785-0
42. Li, C., Yi, Q., Chen, A.: Finite difference methods with non-uniform meshes for nonlinear fractional differential equations. J. Comput. Phys. **316**, 614–631 (2016). https://doi.org/10.1016/j.jcp.2016.04.039
43. Li, T., Wang, Y., Liu, F., Turner, I.: Novel parameter estimation techniques for a multi-order fractional dynamical epidemic model of dengue fever. Numer. Algorithms **82**, 1467–1495 (2019). https://doi.org/10.1007/s11075-019-00665-2
44. Li, Z., Chen, D., Zhu, J., Liu, Y.: Nonlinear dynamics of fractional order Duffing system. Chaos, Solitons Fractals **81**, 111–116 (2015). https://doi.org/10.1016/j.chaos.2015.09.012
45. Liang, H., Stynes, M.: Collocation methods for general Caputo two-point boundary value problems. J. Sci. Comput. **76**, 390–425 (2018). https://doi.org/10.1007/s10915-017-0622-5
46. Lubich, Ch.: Fractional linear multistep methods for Abel-Volterra integral equations of the second kind. Math. Comput. **45**(172), 463–469 (1985). https://doi.org/10.1090/S0025-5718-1985-0804935-7
47. Magin, R.L.: Fractional calculus models of complex dynamics in biological tissues. Comput. Math. Appl. **59**(5), 1586–1593 (2010). https://doi.org/10.1016/j.camwa.2009.08.039
48. Matignon, D.: Stability results for fractional differential equations with applications to control processing. Computational Engineering in Systems Applications. Lille, France: IMACS, IEEE- SMC, 1996, 2: 963–968
49. Milovanović, G.V., Stanić, M.: Construction of multiple orthogonal polynomials by discretized Stieltjes-Gautschi procedure and corresponding Gaussian quadratures. Facta Universitatis (Niš) Ser. Math. Inform. **18**, 9–29 (2003)
50. Momani, S., Odibat, Z.: Numerical approach to differential equations of fractional order. J. Comput. Appl. Math. **207**, 96–110 (2007). https://doi.org/10.1016/j.cam.2006.07.015
51. Nigmatullin, R.R., Arbuzov, A.A., Salehli, F., Giz, A., Bayrak, I., Catalgil-Giz, H.: The first experimental confirmation of the fractional kinetics containing the complexpower-law exponents: Dielectric measurements of polymerization reactions. Phys. B **388**, 418–434 (2007). https://doi.org/10.1016/j.physb.2006.06.153
52. Panja, P.: Dynamics of a fractional order predator-prey model with intraguild predation. Int. J. Math. Model. Simul. Appl. **39**(4), 256–268 (2019). https://doi.org/10.1080/02286203.2019.1611311
53. Petráš, I.: Fractional-order nonlinear systems. Modeling, analysis and simulation. Nonlinear Physical Science Ser., Springer, Heidelberg, (2011)
54. Pedas, A., Tamme, E.: Numerical solution of nonlinear fractional differential equations by spline collocation methods. J. Comput. Appl. Math. **255**, 216–230 (2014). https://doi.org/10.1016/j.cam.2013.04.049
55. Pedas, A., Tamme, E., Vikerpuur, M.: Smoothing transformation and spline collocation for nonlinear fractional initial and boundary value problems. J. Comput. Appl. Math. **317**, 1–16 (2017). https://doi.org/10.1016/j.cam.2016.11.022
56. Podlubny, I.: Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications. Academic Press Inc, San Diego, CA (1999)
57. Sheu, L.-J., Chen, H.-K., Chen, J.-H., Tam, L.-M., Chen, W.-C., Lin, K.-T., Kang, Y.: Chaos in the Newton-Leipnik system with fractional order. Chaos, Solitons Fractals **36**, 98–103 (2008). https://doi.org/10.1016/j.chaos.2006.06.013
58. Sun, H., Zhang, Y., Baleanu, D., Chen, W., Chen, Y.: A new collection of real world applications of fractional calculus in science and engineering. Commun. Nonlinear Sci. Numer. Simul. **64**, 213–231 (2018). https://doi.org/10.1016/j.cnsns.2018.04.019
59. Uchaikin, V., Kozhemiakina, E.: Non-local seismo-dynamics: A Fractional Approach. Fractal and Fractional **6**(9), 513 (2022). https://doi.org/10.3390/fractalfract6090513
60. Torvik, P.J., Bagley, R.L.: On the appearance of the fractional derivative in the behavior of real materials. J. Appl. Mech. **51**(2), 294–298 (1984). https://doi.org/10.1115/1.3167615
61. Van Assche, W.: A Golub-Welsch version for simultaneous Gaussian quadrature. Numer. Algorithms **100**, 599–620 (2025). https://doi.org/10.1007/s11075-024-01767-2
62. Van Assche, W., Coussement, J.: Some classical multiple orthogonal polynomials. J. Compt. Appl. Math. **127**, 317–347 (2001). https://doi.org/10.1016/S0377-0427(00)00503-3

63. Verotta, D.: Fractional compartmental models and multi-order Mittag-Leffler response functions. J. Pharmacokinet Pharmacodyn. **37**(2), 209–215 (2010). https://doi.org/10.1007/s10928-010-9155-3
64. Wang, J., Xu, T.-Z., Wei, Y.-Q., Xie, J.-Q.: Numerical solutions for systems of fractional order differential equations with Bernoulli wavelets. Int. J. Comput. Math. **96**, 317–336 (2019). https://doi.org/10.1080/00207160.2018.1438604
65. Webb, J.R.L.: A fractional Gronwall inequality and the asymptotic behaviour of global solutions of Caputo fractional problems. Electron. J. Differential Equations **2021**(80), 1–22 (2021)
66. Ye, H., Gao, J., Ding, Y.: A generalized Gronwall inequality and its application to a fractional differential equation. J. Math. Anal. Appl. **328**, 1075–1081 (2007). https://doi.org/10.1016/j.jmaa.2006.05.061
67. https://people.dimai.unifi.it/brugnano/fhbvm/
68. https://people.dimai.unifi.it/brugnano/FDEtestset/
69. https://www.dm.uniba.it/it/members/garrappa/software
70. https://archimede.uniba.it/~testset/testsetivpsolvers/