An introduction to fractional calculus

Fundamental ideas and numerics

Fabio Durastante
Università di Pisa
fabio.durastante@unipi.it
fdurastante.github.io

May, 2022

Where are we?

We know a general way to obtain FLMM methods of the form

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n} \omega_{n-j} f(t_j, y^{(j)}),$$

- starting from the polynomials (ρ, σ) of an implicit order p method,

Where are we?

We know a general way to obtain FLMM methods of the form

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n} \omega_{n-j} f(t_j, y^{(j)}),$$

- starting from the polynomials (ρ, σ) of an implicit order p method,
- we have seen how to compute the convolution coefficients ω_n,

Where are we?

We know a general way to obtain FLMM methods of the form

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n} \omega_{n-j} f(t_j, y^{(j)}),$$

- starting from the polynomials (ρ, σ) of an implicit order p method,
- we have seen how to compute the convolution coefficients ω_n,
- we have seen how to compute the starting nodes $w_{n,j}$,

Where are we?

We know a general way to obtain FLMM methods of the form

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n} \omega_{n-j} f(t_j, y^{(j)}),$$

- starting from the polynomials (ρ, σ) of an implicit order p method,
- we have seen how to compute the convolution coefficients ω_n,
- we have seen how to compute the starting nodes $w_{n,j}$,
- we need to discuss how we compute the starting values for a multi-step method,

Where are we?

We know a general way to obtain FLMM methods of the form

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n} \omega_{n-j} f(t_j, y^{(j)}),$$

- ✅ starting from the polynomials (ρ, σ) of an implicit order p method,
- ✅ we have seen how to compute the convolution coefficients ω_n,
- ✅ we have seen how to compute the starting nodes $w_{n,j}$,
- 📋 we need to discuss how we compute the starting values for a multi-step method,
- 📋 we still need to discuss how we can efficiently treat the memory term.

Computing the starting values

To **initialize the computation** we need the values $y^{(0)}, \ldots, y^{(s)}$, $s+1$, $s = |\mathcal{A}| = \mathcal{A}_{p-1} \cup \{p-1\}$ with p the order of convergence of the FLMM, and
$\mathcal{A}_{p-1} = \{\nu \in \mathbb{R} \mid \nu = i + j\alpha, \quad i,j \in \mathbb{N}, \nu < p-1\}$.

Computing the starting values

To **initialize the computation** we need the values $y^{(0)}, \ldots, y^{(s)}$, $s+1$, $s = |\mathcal{A}| = \mathcal{A}_{p-1} \cup \{p-1\}$ with p the order of convergence of the FLMM, and
$\mathcal{A}_{p-1} = \{\nu \in \mathbb{R} \mid \nu = i + j\alpha, \quad i,j \in \mathbb{N}, \nu < p-1\}$.

- We know $y^{(0)}$ from the initial condition, thus we have to solve for the remaining ones.

Computing the starting values

To **initialize the computation** we need the values $y^{(0)}, \dots, y^{(s)}$, $s+1$, $s = |\mathcal{A}| = \mathcal{A}_{p-1} \cup \{p-1\}$ with p the order of convergence of the FLMM, and
$\mathcal{A}_{p-1} = \{\nu \in \mathbb{R} \mid \nu = i + j\alpha, \quad i, j \in \mathbb{N}, \nu < p-1\}$.

- We know $y^{(0)}$ from the initial condition, thus we have to solve for the remaining ones.
- To avoid mixing methods we evaluate all the approximations at the same time by solving

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(s)} \end{bmatrix} = \begin{bmatrix} T_{m-1}(t_1) \\ T_{m-1}(t_2) \\ \vdots \\ T_{m-1}(t_s) \end{bmatrix} + \tau^\alpha \begin{bmatrix} (\omega_1 + w_{1,0}) f_0 \\ (\omega_2 + w_{2,0}) f_0 \\ \vdots \\ (\omega_s + w_{s,0}) f_0 \end{bmatrix} + \tau^\alpha (\Omega \otimes I + W \otimes I) \begin{bmatrix} f(t_1, y^{(1)}) \\ f(t_2, y^{(2)}) \\ \vdots \\ f(t_s, y^{(s)}) \end{bmatrix}$$

where

$$\Omega = \begin{bmatrix} \omega_0 & & & \\ \omega_1 & \omega_0 & & \\ \vdots & \vdots & \ddots & \\ \omega_{s-1} & \omega_{s-2} & \cdots & \omega_0 \end{bmatrix}, \quad W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,s} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,s} \\ \vdots & \vdots & \ddots & \vdots \\ w_{s,1} & w_{s,2} & \cdots & w_{s,s} \end{bmatrix}$$

Computing the starting values

To **initialize the computation** we need the values $y^{(0)}, \ldots, y^{(s)}$, $s+1$, $s = |\mathcal{A}| = \mathcal{A}_{p-1} \cup \{p-1\}$ with p the order of convergence of the FLMM, and
$\mathcal{A}_{p-1} = \{\nu \in \mathbb{R} \,|\, \nu = i + j\alpha, \quad i, j \in \mathbb{N}, \nu < p-1\}$.

- We know $y^{(0)}$ from the initial condition, thus we have to solve for the remaining ones.
- To avoid mixing methods we evaluate all the approximations at the same time by solving

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(s)} \end{bmatrix} = \begin{bmatrix} T_{m-1}(t_1) \\ T_{m-1}(t_2) \\ \vdots \\ T_{m-1}(t_s) \end{bmatrix} + \tau^\alpha \begin{bmatrix} (\omega_1 + w_{1,0}) f_0 \\ (\omega_2 + w_{2,0}) f_0 \\ \vdots \\ (\omega_s + w_{s,0}) f_0 \end{bmatrix} + \tau^\alpha \left(\Omega \otimes I + W \otimes I \right) \begin{bmatrix} f(t_1, y^{(1)}) \\ f(t_2, y^{(2)}) \\ \vdots \\ f(t_s, y^{(s)}) \end{bmatrix}$$

- This will be in general an $s \times \dim(y^{(j)})$ nonlinear system that we need to solve before starting the iteration.

Computing the starting values

To **initialize the computation** we need the values $y^{(0)}, \ldots, y^{(s)}$, $s+1$, $s = |\mathcal{A}| = \mathcal{A}_{p-1} \cup \{p-1\}$ with p the order of convergence of the FLMM, and
$\mathcal{A}_{p-1} = \{\nu \in \mathbb{R} \,|\, \nu = i + j\alpha, \quad i, j \in \mathbb{N}, \nu < p-1\}$.

- We know $y^{(0)}$ from the initial condition, thus we have to solve for the remaining ones.
- To avoid mixing methods we evaluate all the approximations at the same time by solving

$$\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(s)} \end{bmatrix} = \begin{bmatrix} T_{m-1}(t_1) \\ T_{m-1}(t_2) \\ \vdots \\ T_{m-1}(t_s) \end{bmatrix} + \tau^{\alpha} \begin{bmatrix} (\omega_1 + w_{1,0}) f_0 \\ (\omega_2 + w_{2,0}) f_0 \\ \vdots \\ (\omega_s + w_{s,0}) f_0 \end{bmatrix} + \tau^{\alpha} \left(\Omega \otimes I + W \otimes I \right) \begin{bmatrix} f(t_1, y^{(1)}) \\ f(t_2, y^{(2)}) \\ \vdots \\ f(t_s, y^{(s)}) \end{bmatrix}$$

- This will be in general an $s \times \dim(y^{(j)})$ nonlinear system that we need to solve before starting the iteration.
- If the value of α is not very small, viz s is moderate, and the system of ODEs is moderate this is manageable.

Treating the memory term

If we compute the sum on the coefficients ω_j naively for

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n-1} \omega_{n-j} f(t_j, y^{(j)}) + \tau^{\alpha} \omega_0 f(t_n, y^{(n)}),$$

we end up having a $O(N^2)$ cost! If we do not perform this task efficiently the numerical solution degenerates in an unworkable task as we either refine our grid or enlarge our computational domain.

Treating the memory term

If we compute the sum on the coefficients ω_j naively for

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n-1} \omega_{n-j} f(t_j, y^{(j)}) + \tau^{\alpha} \omega_0 f(t_n, y^{(n)}),$$

we end up having a $O(N^2)$ cost! If we do not perform this task efficiently the numerical solution degenerates in an unworkable task as we either refine our grid or enlarge our computational domain.

We can try to "forget" part of the lag-term,

Treating the memory term

If we compute the sum on the coefficients ω_j naively for

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n-1} \omega_{n-j} f(t_j, y^{(j)}) + \tau^{\alpha} \omega_0 f(t_n, y^{(n)}),$$

we end up having a $O(N^2)$ cost! If we do not perform this task efficiently the numerical solution degenerates in an unworkable task as we either refine our grid or enlarge our computational domain.

- We can try to "forget" part of the lag-term,
- We can consider using a stretched grid towards t_0 to reduce N,

Treating the memory term

If we compute the sum on the coefficients ω_j naively for

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n-1} \omega_{n-j} f(t_j, y^{(j)}) + \tau^{\alpha} \omega_0 f(t_n, y^{(n)}),$$

we end up having a $O(N^2)$ cost! If we do not perform this task efficiently the numerical solution degenerates in an unworkable task as we either refine our grid or enlarge our computational domain.

- We can try to "forget" part of the lag-term,
- We can consider using a stretched grid towards t_0 to reduce N,
- We can try an approach with nested meshes to reduce the load,

Treating the memory term

If we compute the sum on the coefficients ω_j naively for

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n-1} \omega_{n-j} f(t_j, y^{(j)}) + \tau^{\alpha} \omega_0 f(t_n, y^{(n)}),$$

we end up having a $O(N^2)$ cost! If we do not perform this task efficiently the numerical solution degenerates in an unworkable task as we either refine our grid or enlarge our computational domain.

- We can try to "forget" part of the lag-term,
- We can consider using a stretched grid towards t_0 to reduce N,
- We can try an approach with nested meshes to reduce the load,
- We can exploit the fact that this is a convolution and adopt some FFT tricks.

The FFT trick (Hairer, Lubich, and Schlichte 1985)

The treatment remains the same indifferently for both PI and FLMM method, let us focus here on the generic formulation

$$y^{(n)} = \phi_n + \sum_{j=0}^{n} c_{n-j} f_j.$$

The FFT trick (Hairer, Lubich, and Schlichte 1985)

The treatment remains the same indifferently for both PI and FLMM method, let us focus here on the generic formulation

$$y^{(n)} = \phi_n + \sum_{j=0}^{n} c_{n-j} f_j.$$

- Let r be a moderate number of step, e.g., $r = 2^k$ for a small k, we compute the first steps directly

$$y^{(n)} = \phi_n + \sum_{j=0}^{n} c_{n-j} f_j, \quad n = 0, 1, \ldots, r-1.$$

The FFT trick (Hairer, Lubich, and Schlichte 1985)

The treatment remains the same indifferently for both PI and FLMM method, let us focus here on the generic formulation

$$y^{(n)} = \phi_n + \sum_{j=0}^{n} c_{n-j} f_j.$$

- Let r be a moderate number of step, e.g., $r = 2^k$ for a small k, we compute the first steps directly
$$y^{(n)} = \phi_n + \sum_{j=0}^{n} c_{n-j} f_j, \quad n = 0, 1, \ldots, r-1.$$
- If we now want to compute the next r approximations we can separate the lag term as
$$y^{(n)} = \phi_n + \sum_{j=0}^{r-1} c_{n-j} f_j + \sum_{j=r}^{n} c_{n-j} f_j, \quad n \in \{r, r+1, \ldots, 2r-1\}.$$

The FFT trick (Hairer, Lubich, and Schlichte 1985)

- If we now want to compute the next r approximations we can separate the lag term as

$$y^{(n)} = \phi_n + \sum_{j=0}^{r-1} c_{n-j} f_j + \sum_{j=r}^{n} c_{n-j} f_j, \quad n \in \{r, r+1, \dots, 2r-1\}.$$

The FFT trick (Hairer, Lubich, and Schlichte 1985)

- If we now want to compute the next r approximations we can separate the lag term as

$$y^{(n)} = \phi_n + \sum_{j=0}^{r-1} c_{n-j} f_j + \sum_{j=r}^{n} c_{n-j} f_j, \quad n \in \{r, r+1, \ldots, 2r-1\}.$$

We can use FFT!

If we call $S_r(n, 0, r-1) = \sum_{j=0}^{r-1} c_{n-j} f_j$, $n \in \{r, r+1, \ldots, 2r-1\}$, the set of partial sums each of length r we can evaluate them with FFT in $O(2r \log_2(2r))$.

The FFT trick (Hairer, Lubich, and Schlichte 1985)

- If we now want to compute the next r approximations we can separate the lag term as

$$y^{(n)} = \phi_n + \sum_{j=0}^{r-1} c_{n-j} f_j + \sum_{j=r}^{n} c_{n-j} f_j, \quad n \in \{r, r+1, \dots, 2r-1\}.$$

- We can apply the same process recursively if we double every time-interval under consideration

$$\begin{aligned}
y^{(n)} &= \phi_n + \sum_{j=0}^{2r-1} c_{n-j} f_j + \sum_{j=2r}^{n} c_{n-j} f_j, \quad n \in \{2r, 2r+1, \dots, 3r-1\}, \\
y^{(n)} &= \phi_n + \sum_{j=0}^{2r-1} c_{n-j} f_j + \sum_{j=2r}^{3r-1} c_{n-j} f_j + \sum_{j=3r}^{n} c_{n-j} f_j, \quad n \in \{3r, 3r+1, \dots, 4r-1\},
\end{aligned}$$

The FFT trick (Hairer, Lubich, and Schlichte 1985)

We can use FFT!

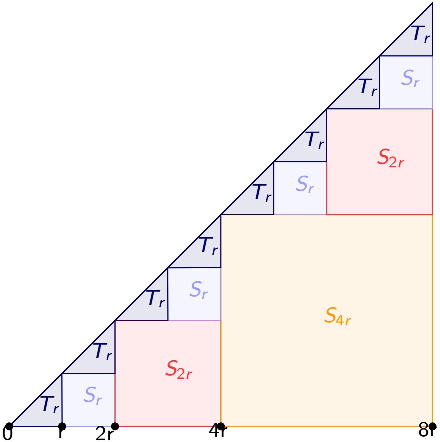
If we call $S_{2r}(n,0,2r-1) = \sum_{j=0}^{2r-1} c_{n-j} f_j$, and $S_r(n,2r,3r-1) = \sum_{j=2r}^{3r-1} c_{n-j} f_j$ the set of partial sums of lengths $2r$ and r we can evaluate them with FFT in $O(4r\log_2(4r))$ and $O(2r\log_2(2r))$ respectively.

- We can apply the same process recursively if we double every time-interval under consideration

$$y^{(n)} = \phi_n + \sum_{j=0}^{2r-1} c_{n-j} f_j + \sum_{j=2r}^{n} c_{n-j} f_j, \quad n \in \{2r, 2r+1, \dots, 3r-1\},$$

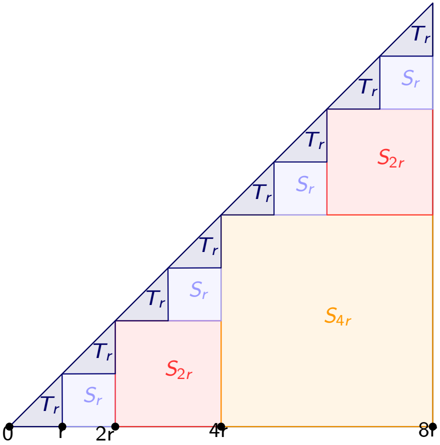
$$y^{(n)} = \phi_n + \sum_{j=0}^{2r-1} c_{n-j} f_j + \sum_{j=2r}^{3r-1} c_{n-j} f_j + \sum_{j=3r}^{n} c_{n-j} f_j, \quad n \in \{3r, 3r+1, \dots, 4r-1\},$$

The FFT trick (Hairer, Lubich, and Schlichte 1985)



- We can iterate the process for the $4r$ approximations in the interval $n \in \{4r, \ldots, 8r-1\}$, together with the partial sums $S_{4r}(n, 0, 4r-1)$, $S_{2r}(n, 4r, 6r-1)$, $S_r(n, 6r, 7r-1)$ that can be evaluated in $O(8r\log_2(8r))$, $O(4r\log_2(4r))$ and $O(2r\log_2(2r))$ respectively,

The FFT trick (Hairer, Lubich, and Schlichte 1985)



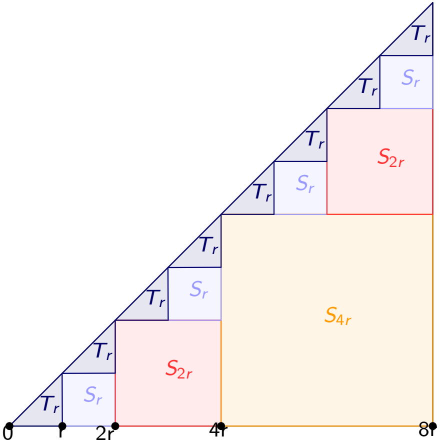
- We can iterate the process for the $4r$ approximations in the interval $n \in \{4r, \dots, 8r-1\}$, together with the partial sums $S_{4r}(n, 0, 4r-1)$, $S_{2r}(n, 4r, 6r-1)$, $S_r(n, 6r, 7r-1)$ that can be evaluated in $O(8r\log_2(8r))$, $O(4r\log_2(4r))$ and $O(2r\log_2(2r))$ respectively,
- At each level we have to complete the recursion by computing

$$T_r(p,n) = \sum_{j=p}^{n} c_{n-j} f_j, \;\; p = \ell r,$$

$$n \in \{\ell r, \ell r + 1, \dots, (\ell+1)r - 1\},$$

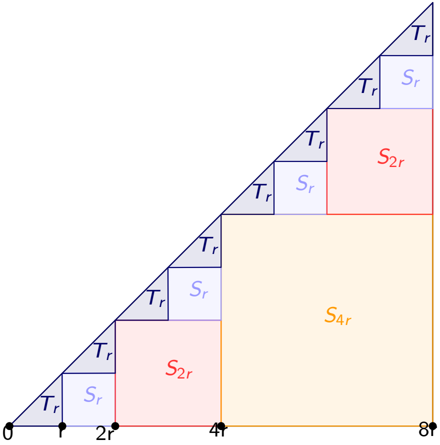
$$\ell = 0, 1, 2, \dots.$$

The FFT trick (Hairer, Lubich, and Schlichte 1985)



To determine the whole cost we just have to sum the various components

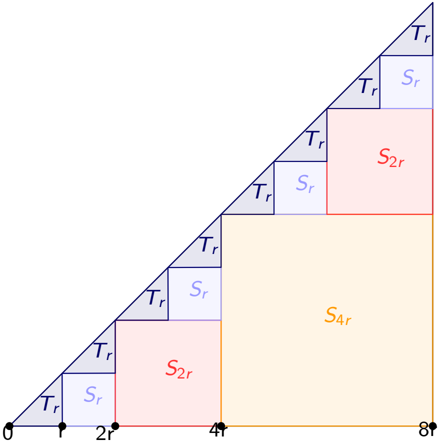
The FFT trick (Hairer, Lubich, and Schlichte 1985)



To determine the whole cost we just have to sum the various components

- Assume that $N = 2^{n_t}$

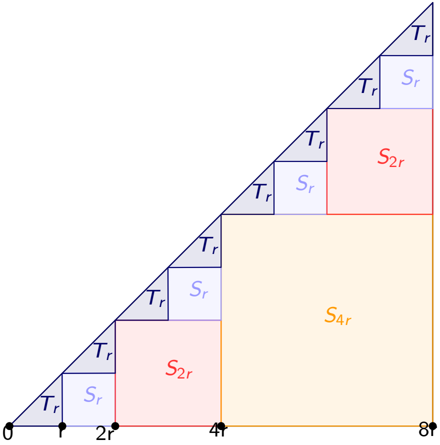
The FFT trick (Hairer, Lubich, and Schlichte 1985)



To determine the whole cost we just have to sum the various components

- Assume that $N = 2^{n_t}$
- $O(N \log_2 N)$ for S_{4r},

The FFT trick (Hairer, Lubich, and Schlichte 1985)

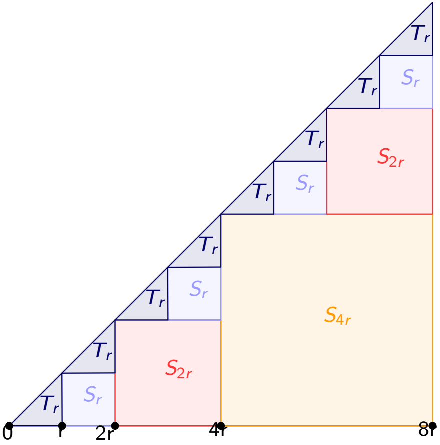


To determine the whole cost we just have to sum the various components

- Assume that $N = 2^{n_t}$
- $O(N \log_2 N)$ for S_{4r},

$+$ $O(N/2 \log_2 N/2)$ for 2 S_{2r}

The FFT trick (Hairer, Lubich, and Schlichte 1985)



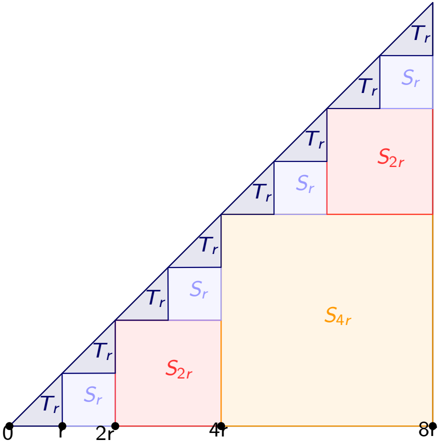
To determine the whole cost we just have to sum the various components

- Assume that $N = 2^{n_t}$
- $O(N \log_2 N)$ for S_{4r},

$+ \; O(N/2 \log_2 N/2)$ for 2 S_{2r}

$+ \; O(N/4 \log_2 N/4)$ for 4 S_r

The FFT trick (Hairer, Lubich, and Schlichte 1985)



To determine the whole cost we just have to sum the various components

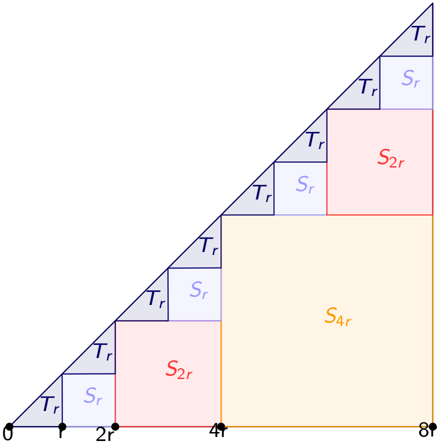
- Assume that $N = 2^{n_t}$
- $O(N \log_2 N)$ for S_{4r},

$+$ $O(N/2 \log_2 N/2)$ for 2 S_{2r}

$+$ $O(N/4 \log_2 N/4)$ for 4 S_r

$+$ $r(r+1)/2$ for the N/r convolutions T_r

The FFT trick (Hairer, Lubich, and Schlichte 1985)



To determine the whole cost we just have to sum the various components

- Assume that $N = 2^{n_t}$
- $O(N \log_2 N)$ for S_{4r},

$+$ $O(N/2 \log_2 N/2)$ for 2 S_{2r}

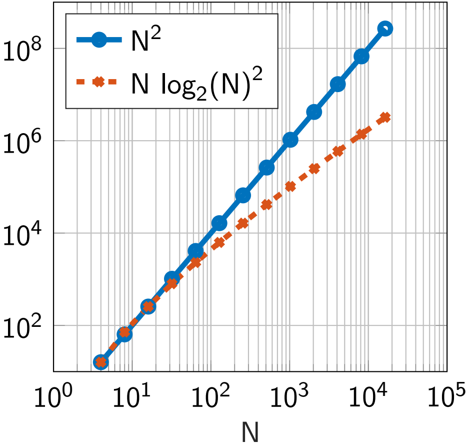
$+$ $O(N/4 \log_2 N/4)$ for 4 S_r

$+$ $r(r+1)/2$ for the N/r convolutions T_r

- In general:

$$N \log_2 N + 2\frac{N}{2}\log_2 \frac{N}{2} + 4\frac{N}{4}\log_2\frac{N}{4} + \cdots \\ + p\frac{N}{p}\log_2\frac{N}{p} + \frac{N}{r}\frac{r(r+1)}{2}, \quad p = \frac{N}{2r}$$

The FFT trick (Hairer, Lubich, and Schlichte 1985)



To determine the whole cost we just have to sum the various components

- Assume that $N = 2^{n_t}$
- $O(N \log_2 N)$ for S_{4r},

$+$ $O(N/2 \log_2 N/2)$ for 2 S_{2r}

$+$ $O(N/4 \log_2 N/4)$ for 4 S_r

$+$ $r(r+1)/2$ for the N/r convolutions T_r

- In general:

$$\sum_{j=0}^{\log_2 p} N \log_2 \frac{N}{2^j} + N\frac{r+1}{2} = O(N(\log_2 N)^2).$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a "fixed memory length" to reduce the computational (and memory) load.

$$
\begin{aligned}
y(t_{n+1}) = y(t_n) &+ \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1} - \tau)^{\alpha-1} f(\tau, x(\tau)) \,\mathrm{d}\tau \\
&+ \frac{1}{\Gamma(\alpha)} \int_{0}^{t_n} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}) f(\tau, y(\tau)) \,\mathrm{d}\tau, \quad \alpha \in (0, 1).
\end{aligned}
$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a “fixed memory length” to reduce the computational (and memory) load.

$$
\begin{aligned}
y(t_{n+1}) = y(t_n) &+ \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1}-\tau)^{\alpha-1} f(\tau, x(\tau))\,\mathrm{d}\tau \\
&+ \frac{1}{\Gamma(\alpha)} \int_0^{t_n} \left((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}\right) f(\tau, y(\tau))\,\mathrm{d}\tau, \quad \alpha \in (0,1).
\end{aligned}
$$

Let us introduce now a fixed window T_M of memory, then

$$
\begin{aligned}
E &= \left| \frac{1}{\Gamma(\alpha)} \int_0^{t_n - T_M} \left((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}\right) f(\tau, y(\tau))\,\mathrm{d}\tau \right| \\
&\leq \frac{M}{\Gamma(\alpha)} \left| \int_0^{t_n - T_M} \left((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}\right) \mathrm{d}\tau \right|
\end{aligned}
$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a “fixed memory length” to reduce the computational (and memory) load.

$$
\begin{aligned}
y(t_{n+1}) = y(t_n) &+ \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1} - \tau)^{\alpha-1} f(\tau, x(\tau))\, \mathrm{d}\tau \\
&+ \frac{1}{\Gamma(\alpha)} \int_{0}^{t_n} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}) f(\tau, y(\tau))\, \mathrm{d}\tau, \quad \alpha \in (0,1).
\end{aligned}
$$

Let us introduce now a fixed window T_M of memory, then

$$
\begin{aligned}
E =& \left| \frac{1}{\Gamma(\alpha)} \int_0^{t_n - T_M} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}) f(\tau, y(\tau))\, \mathrm{d}\tau \right| \\
\leq& \frac{M}{\Gamma(\alpha)} \left| \int_0^{t_n - T_M} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1})\, \mathrm{d}\tau \right| \\
=& \frac{M}{\alpha\Gamma(\alpha)} \left| (\tau + T_M)^{\alpha} - t_{n+1}^{\alpha} - T^{\alpha} + t_n^{\alpha} \right|
\end{aligned}
$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a "fixed memory length" to reduce the computational (and memory) load.

$$
\begin{aligned}
y(t_{n+1}) = y(t_n) &+ \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1}-\tau)^{\alpha-1} f(\tau, x(\tau))\,\mathrm{d}\tau \\
&+ \frac{1}{\Gamma(\alpha)} \int_{0}^{t_n} \left((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}\right) f(\tau, y(\tau))\,\mathrm{d}\tau, \quad \alpha \in (0,1).
\end{aligned}
$$

Let us introduce now a fixed window T_M of memory, then

$$
\begin{aligned}
E =& \left| \frac{1}{\Gamma(\alpha)} \int_{0}^{t_n - T_M} \left((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}\right) f(\tau, y(\tau))\,\mathrm{d}\tau \right| \\
\leq & \frac{M}{\Gamma(\alpha)} \left| \int_{0}^{t_n - T_M} \left((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}\right) \mathrm{d}\tau \right| \\
=& \frac{M}{\Gamma(\alpha)} \left| \int_{T_M}^{T_M+\tau} z^{\alpha-1}\,\mathrm{d}z - \int_{t_n}^{t_{n+1}} z^{\alpha-1}\,\mathrm{d}z \right|
\end{aligned}
$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a "fixed memory length" to reduce the computational (and memory) load.

$$
\begin{aligned}
y(t_{n+1}) = y(t_n) &+ \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1} - \tau)^{\alpha-1} f(\tau, x(\tau)) \, \mathrm{d}\tau \\
&+ \frac{1}{\Gamma(\alpha)} \int_0^{t_n} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}) f(\tau, y(\tau)) \, \mathrm{d}\tau, \quad \alpha \in (0,1).
\end{aligned}
$$

Let us introduce now a fixed window T_M of memory, then

$$
\begin{aligned}
E = & \left| \frac{1}{\Gamma(\alpha)} \int_0^{t_n - T_M} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}) f(\tau, y(\tau)) \, \mathrm{d}\tau \right| \\
\leq & \frac{M}{\Gamma(\alpha)} \left| \int_0^{t_n - T_M} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}) \, \mathrm{d}\tau \right| \\
= & \frac{M}{\Gamma(\alpha)} \left| \int_{T_M}^{T_M + \tau} z^{\alpha-1} \, \mathrm{d}z - \int_{t_n}^{t_{n+1}} z^{\alpha-1} \, \mathrm{d}z \right| (Mean Value Theorem)
\end{aligned}
$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a "fixed memory length" to reduce the computational (and memory) load.

$$
\begin{aligned}
y(t_{n+1}) = y(t_n) &+ \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1} - \tau)^{\alpha-1} f(\tau, x(\tau))\, \mathrm{d}\tau \\
&+ \frac{1}{\Gamma(\alpha)} \int_0^{t_n} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}) f(\tau, y(\tau))\, \mathrm{d}\tau, \quad \alpha \in (0,1).
\end{aligned}
$$

Let us introduce now a fixed window T_M of memory, then

$$
\begin{aligned}
E = & \left| \frac{1}{\Gamma(\alpha)} \int_0^{t_n - T_M} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}) f(\tau, y(\tau))\, \mathrm{d}\tau \right| \\
\leq & \frac{M}{\Gamma(\alpha)} \left| \int_0^{t_n - T_M} ((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1})\, \mathrm{d}\tau \right| \\
= & \frac{M}{\Gamma(\alpha)} \left| (z_1^*)^{\alpha-1}\tau - (z_2^*)^{\alpha-1}\tau \right| \quad z_1^* \in [T_M, T_M + \tau],\ z_2^* \in [t_n, t_{n+1}]
\end{aligned}
$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a "fixed memory length" to reduce the computational (and memory) load.

$$
\begin{aligned}
y(t_{n+1}) = y(t_n) &+ \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1} - \tau)^{\alpha-1} f(\tau, x(\tau))\, \mathrm{d}\tau \\
&+ \frac{1}{\Gamma(\alpha)} \int_0^{t_n} \left((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}\right) f(\tau, y(\tau))\, \mathrm{d}\tau, \quad \alpha \in (0,1).
\end{aligned}
$$

Let us introduce now a fixed window T_M of memory, then

$$
\begin{aligned}
E =& \left| \frac{1}{\Gamma(\alpha)} \int_0^{t_n - T_M} \left((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}\right) f(\tau, y(\tau))\, \mathrm{d}\tau \right| \\
\leq & \frac{M}{\Gamma(\alpha)} \left| \int_0^{t_n - T_M} \left((t_{n+1} - \tau)^{\alpha-1} - (t_n - \tau)^{\alpha-1}\right) \mathrm{d}\tau \right| \\
< & \frac{M}{\Gamma(\alpha)} T_M^{\alpha-1} \tau, \qquad \alpha \in (0,1).
\end{aligned}
$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a "fixed memory length" to reduce the computational (and memory) load.

$$y(t_{n+1}) = y(t_n) + \frac{1}{\Gamma(\alpha)}\int_{t_n}^{t_{n+1}} (t_{n+1}-\tau)^{\alpha-1} f(\tau, x(\tau))\,\mathrm{d}\tau$$
$$+ \frac{1}{\Gamma(\alpha)}\int_{0}^{t_n} ((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}) f(\tau, y(\tau))\,\mathrm{d}\tau, \quad \alpha \in (0,1).$$

Let us introduce now a fixed window T_M of memory, then

$$E = \left| \frac{1}{\Gamma(\alpha)}\int_{0}^{t_n - T_M} ((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}) f(\tau, y(\tau))\,\mathrm{d}\tau \right| < \frac{M}{\Gamma(\alpha)} T_M^{\alpha-1}\tau, \quad \alpha \in (0,1).$$

Short-memory principle (Ford and Simpson 2001)

We can try to use a "fixed memory length" to reduce the computational (and memory) load.

$$
\begin{aligned}
y(t_{n+1}) = y(t_n) &+ \frac{1}{\Gamma(\alpha)} \int_{t_n}^{t_{n+1}} (t_{n+1}-\tau)^{\alpha-1} f(\tau, x(\tau))\,\mathrm{d}\tau \\
&+ \frac{1}{\Gamma(\alpha)} \int_0^{t_n} ((t_{n+1}-\tau)^{\alpha-1} - (t_n-\tau)^{\alpha-1}) f(\tau, y(\tau))\,\mathrm{d}\tau, \quad \alpha \in (0,1).
\end{aligned}
$$

Let us introduce now a fixed window T_M of memory, then If we have a global error bound E_{global} with step-length τ we just need to choose

$$
T_M > \left(\frac{M}{\Gamma(\alpha) E_{\text{global}}} \right)^{1/1-\alpha}, \quad \alpha \in (0,1),
$$

while if we have a local error bound E_{local}

$$
T_M > \left(\frac{M\tau}{\Gamma(\alpha) E_{\text{local}}} \right)^{1/1-\alpha}, \quad \alpha \in (0,1).
$$

Short-memory principle (Ford and Simpson 2001)

- In case $\alpha \in (0, 1)$ the short memory method with fixed length can be effective and the length T is independent of the full interval of integration.

Short-memory principle (Ford and Simpson 2001)

- In case $\alpha \in (0,1)$ the short memory method with fixed length can be effective and the length T is independent of the full interval of integration.
- Similar bounds can be written for the case $\alpha > 1$, that is

$$E < \frac{M}{\Gamma(\alpha)}(t_{n+1}^{\alpha} - T_M^{\alpha-1})\tau, \quad \alpha > 1.$$

But now to preserve the order of accuracy, we must choose

$$T_M^{\alpha-1} > t_{n+1}^{\alpha-1} - \frac{E_{\text{global}}\Gamma(\alpha)}{M}, \quad \alpha > 1,$$

that we will make us lose all the computational gain.

Short-memory principle (Ford and Simpson 2001)

- In case $\alpha \in (0,1)$ the short memory method with fixed length can be effective and the length T is independent of the full interval of integration.
- Similar bounds can be written for the case $\alpha > 1$, that is

$$E < \frac{M}{\Gamma(\alpha)}(t_{n+1}^{\alpha} - T_M^{\alpha-1})\tau, \quad \alpha > 1.$$

  But now to preserve the order of accuracy, we must choose

$$T_M^{\alpha-1} > t_{n+1}^{\alpha-1} - \frac{E_{\text{global}}\Gamma(\alpha)}{M}, \quad \alpha > 1,$$

  that we will make us lose all the computational gain.

The idea can be refined by using *nested meshes*.

Nested meshes

Zeroing out the memory term is too drastic, we may want to relax this.

Nested meshes

Zeroing out the memory term is too drastic, we may want to relax this.

Scaling properties

$$I^{\alpha}_{[0,t]} f(t) = \int_0^t \frac{f(\tau)}{(t-\tau)^{1-\alpha}} \, \mathrm{d}\tau$$

Nested meshes

Zeroing out the memory term is too drastic, we may want to relax this.

Scaling properties

$$I^{\alpha}_{[0,t]} f(wt) = \int_0^t \frac{f(\tau)}{(wt-\tau)^{1-\alpha}} \, \mathrm{d}\tau$$

Nested meshes

Zeroing out the memory term is too drastic, we may want to relax this.

Scaling properties

$$I^{\alpha}_{[0,t]} f(wt) = w^{\alpha} \int_0^t \frac{f(w\tau)}{(t-\tau)^{1-\alpha}} \, \mathrm{d}\tau$$

Nested meshes

Zeroing out the memory term is too drastic, we may want to relax this.

Scaling properties

Given $p \in \mathbb{N}$ we then have

$$I^{\alpha}_{[0,t]} f(w^p t) = w^{p\alpha} \int_0^t \frac{f(w^p \tau)}{(t-\tau)^{1-\alpha}} \, \mathrm{d}\tau$$

Nested meshes

Zeroing out the memory term is too drastic, we may want to relax this.

Scaling properties

Given $p \in \mathbb{N}$ we then have

$$I^{\alpha}_{[0,t]} f(w^p t) = w^{p\alpha} \int_0^t \frac{f(w^p \tau)}{(t-\tau)^{1-\alpha}} \,\mathrm{d}\tau$$

- We can use the weight on the mesh

$$\Omega^{\alpha}_{\tau} f(n\tau) \approx I^{\alpha}_{[0,t]} f(n\tau), \text{ step length } \tau$$

to compute

$$\Omega^{\alpha}_{w^p\tau} f(nw^p\tau) \approx I^{\alpha}_{[0,t]} f(nw^p\tau), \text{ step length } w^p\tau$$

Nested meshes

Zeroing out the memory term is too drastic, we may want to relax this.

Scaling properties

Given $p \in \mathbb{N}$ we then have

$$I^{\alpha}_{[0,t]} f(w^p t) = w^{p\alpha} \int_0^t \frac{f(w^p \tau)}{(t-\tau)^{1-\alpha}} \, \mathrm{d}\tau$$

- We can use the weight on the mesh

$$\Omega^{\alpha}_{\tau} f(n\tau) \approx I^{\alpha}_{[0,t]} f(n\tau), \text{ step length } \tau$$

  to compute

$$\Omega^{\alpha}_{w^p\tau} f(nw^p\tau) \approx I^{\alpha}_{[0,t]} f(nw^p\tau), \text{ step length } w^p\tau$$

- In summary for any $p \in \mathbb{N}$ we get

$$\Omega^{\alpha}_{\tau} f(n\tau) = \sum_{j=0}^{n} \omega_{n-j} f(j\tau) \Leftrightarrow \Omega^{\alpha}_{w^p\tau} f(nw^p\tau) = w^{p\alpha} \sum_{j=0}^{n} \omega_{n-j} f(jw^p\tau).$$

Nested meshes (Ford and Simpson 2001)

Nested mesh

Given $\tau \in \mathbb{R}^+$, the mesh $M_\tau = \{\tau n,\ n \in \mathbb{N}\}$. Selected $w, r, p \in \mathbb{N}$, $w > 0$, $r > p$, we have $M_{w^p\tau} \supset M_{w^r\tau}$ and we decompose the interval as

$$[0, t] = [0, t - w^m T] \cup [t - w^m T, t - w^{m-1} T] \cup \cdots \cup [t - wT, t - T] \cup [t - T, t]$$

for $m \in \mathbb{N}$ the smallest integer such that $t < w^{m+1} T$.

Nested meshes (Ford and Simpson 2001)

Nested mesh

Given $\tau \in \mathbb{R}^+$, the mesh $M_\tau = \{\tau n,\ n \in \mathbb{N}\}$. Selected $w, r, p \in \mathbb{N}$, $w > 0$, $r > p$, we have $M_{w^p\tau} \supset M_{w^r\tau}$ and we decompose the interval as

$$[0, t] = [0, t - w^m T] \cup [t - w^m T, t - w^{m-1} T] \cup \cdots \cup [t - wT, t - T] \cup [t - T, t]$$

for $m \in \mathbb{N}$ the smallest integer such that $t < w^{m+1} T$.

- This links the **scaling property** with the singularity of the type $1/(t-\tau)^{1-\alpha}$ suggesting that we should distribute the computational effort logarithmically, and not uniformly.

Nested meshes (Ford and Simpson 2001)

Nested mesh

Given $\tau \in \mathbb{R}^+$, the mesh $M_\tau = \{\tau n,\ n \in \mathbb{N}\}$. Selected $w, r, p \in \mathbb{N}$, $w > 0$, $r > p$, we have $M_{w^p \tau} \supset M_{w^r \tau}$ and we decompose the interval as

$$[0, t] = [0, t - w^m T] \cup [t - w^m T, t - w^{m-1} T] \cup \cdots \cup [t - wT, t - T] \cup [t - T, t]$$

for $m \in \mathbb{N}$ the smallest integer such that $t < w^{m+1} T$.

- This links the **scaling property** with the singularity of the type $1/(t-\tau)^{1-\alpha}$ suggesting that we should distribute the computational effort logarithmically, and not uniformly.
- We rewrite our integral as

$$I^{\alpha}_{[0,t]} f(t) = I^{\alpha}_{[t-T,t]} f(t) + \sum_{i=0}^{m-1} I^{\alpha}_{[t-w^{i+1}T, t-w^i T]} f(t) + I^{\alpha}_{[0, t-w^m T]} f(t)$$

Nested meshes (Ford and Simpson 2001)

Nested mesh

Given $\tau \in \mathbb{R}^+$, the mesh $M_\tau = \{\tau n,\ n \in \mathbb{N}\}$. Selected $w, r, p \in \mathbb{N}$, $w > 0$, $r > p$, we have $M_{w^p\tau} \supset M_{w^r\tau}$ and we decompose the interval as

$$[0, t] = [0, t - w^m T] \cup [t - w^m T, t - w^{m-1} T] \cup \cdots \cup [t - wT, t - T] \cup [t - T, t]$$

for $m \in \mathbb{N}$ the smallest integer such that $t < w^{m+1} T$.

- This links the **scaling property** with the singularity of the type $1/(t-\tau)^{1-\alpha}$ suggesting that we should distribute the computational effort logarithmically, and not uniformly.
- We rewrite our integral using the scaling property as

$$I^{\alpha}_{[0,t]} f(t) = I^{\alpha}_{[t-T,t]} f(t) + \sum_{i=0}^{m-1} w^{i\alpha} I^{\alpha}_{[t-wT,t-T]} f(w^i t) + w^{m\alpha} I^{\alpha}_{[0,t-T]} f(w^m t).$$

Nested meshes (Ford and Simpson 2001)

In the discrete approximation of

$$I^{\alpha}_{[0,t]} f(t) = I^{\alpha}_{[t-T,t]} f(t) + \sum_{i=0}^{m-1} w^{i\alpha} I^{\alpha}_{[t-wT,t-T]} f(w^i t) + w^{m\alpha} I^{\alpha}_{[0,t-T]} f(w^m t).$$

we approximate

$$\Omega^{\alpha}_{\tau,[t-w^{i+1}T,t-w^iT]} f(t) \approx \Omega^{\alpha}_{w^i\tau,[t-w^{i+1}T,t-w^iT]} f(t)$$

and substitute

$$w^{i\alpha}\Omega^{\alpha}_{\tau,[t-wT,t-T]} f(t) = \Omega^{\alpha}_{w^i\tau,[t-w^{i+1}T,t-w^iT]} f(t).$$

Nested meshes (Ford and Simpson 2001)

In the discrete approximation of

$$I^{\alpha}_{[0,t]}f(t) = I^{\alpha}_{[t-T,t]}f(t) + \sum_{i=0}^{m-1} w^{i\alpha} I^{\alpha}_{[t-wT,t-T]}f(w^i t) + w^{m\alpha} I^{\alpha}_{[0,t-T]}f(w^m t).$$

we approximate

$$\Omega^{\alpha}_{\tau,[t-w^{i+1}T,t-w^iT]}f(t) \approx \Omega^{\alpha}_{w^i\tau,[t-w^{i+1}T,t-w^iT]}f(t)$$

and substitute

$$w^{i\alpha}\Omega^{\alpha}_{\tau,[t-wT,t-T]}f(t) = \Omega^{\alpha}_{w^i\tau,[t-w^{i+1}T,t-w^iT]}f(t).$$

Theorem (Ford and Simpson 2001, Theorem 1)

The nested mesh scheme preserves the order of the underlying quadrature rule on which it is based.

Nested meshes (Ford and Simpson 2001)

In the discrete approximation of

$$I^{\alpha}_{[0,t]} f(t) = I^{\alpha}_{[t-T,t]} f(t) + \sum_{i=0}^{m-1} w^{i\alpha} I^{\alpha}_{[t-wT,t-T]} f(w^i t) + w^{m\alpha} I^{\alpha}_{[0,t-T]} f(w^m t).$$

we approximate

$$\Omega^{\alpha}_{\tau,[t-w^{i+1}T,t-w^iT]} f(t) \approx \Omega^{\alpha}_{w^i\tau,[t-w^{i+1}T,t-w^iT]} f(t)$$

and substitute

$$w^{i\alpha}\Omega^{\alpha}_{\tau,[t-wT,t-T]} f(t) = \Omega^{\alpha}_{w^i\tau,[t-w^{i+1}T,t-w^iT]} f(t).$$

Theorem (Ford and Simpson 2001, Theorem 1)

The nested mesh scheme preserves the order of the underlying quadrature rule on which it is based.

Proof. For integration over a fixed interval $[0,t]$ the choice of T fixes (independent of h) the number of subranges over which the integral is evaluated, on each of them we have en error $O(h^p)$.

Nested meshes (Ford and Simpson 2001)

- The first benefit is that we evaluate a fixed number of quadrature coefficients and then re-use them on all successive intervals,

Nested meshes (Ford and Simpson 2001)

- The first benefit is that we evaluate a fixed number of quadrature coefficients and then re-use them on all successive intervals,
- This approach cost $O(w^m)$ with respect to $O(w^{2m})$ of the full method,

Nested meshes (Ford and Simpson 2001)

- The first benefit is that we evaluate a fixed number of quadrature coefficients and then re-use them on all successive intervals,
- This approach cost $O(w^m)$ with respect to $O(w^{2m})$ of the full method,
- We could use linear extrapolation techniques to improve the results.

Nested meshes (Ford and Simpson 2001)

- The first benefit is that we evaluate a fixed number of quadrature coefficients and then re-use them on all successive intervals,
- This approach cost $O(w^m)$ with respect to $O(w^{2m})$ of the full method,
- We could use linear extrapolation techniques to improve the results.
- Selecting the various parameter may need a bit of tuning.

Available codes

With respect to the ordinary case for which there exists many reliable and high-performance codes, the choices for computing the solution of fractional differential equation is much more *sparse*.

- From (Garrappa 2018)
  - `FDE_PI1_Ex.m` - Explicit Product-Integration of rectanguar type
  - `FDE_PI1_Im.m` - Implicit Product-Integration of rectanguar type
  - `FDE_PI2_Im.m` - Implicit Product-Integration of trapezoidal type
  - `FDE_PI12_PC.m` - Product-Integration with predictor-corrector
- From (Garrappa 2015)
  - FLMM2 Matlab code - Three implicit second order Fractional Linear Multistep Methods.

A remark

All these methods use direct-solver for the Newton method inside them, there is space to make improvement on the solution strategies. Furthermore, a challenge that yet remains: can we find a strategy that combines the convolution features and savings on the memory?

What do we have now

We know a general way to obtain FLMM methods of the form

$$y^{(n)} = T_{m-1}(t_n) + \tau^{\beta} \sum_{j=0}^{s} w_{n,j} f(t_j, y^{(j)}) + \tau^{\alpha} \sum_{j=0}^{n} \omega_{n-j} f(t_j, y^{(j)}),$$

- starting from the polynomials (ρ, σ) of an implicit order p method,
- we have seen how to compute the convolution coefficients ω_n,
- we have seen how to compute the starting nodes $w_{n,j}$,
- we know how we can compute the starting values for a multi-step method by solving a nonlinear system with Newton,
- we have some hints on how we can efficiently treat the memory term.

A worked out example

Let us write everything for a case, let us start from the 2^{nd} order BDF formula for ODEs

$$y^{(n+2)} - \frac{4}{3}y^{(n+1)} + \frac{1}{3}y^{(n)} = \frac{2}{3}\tau f_{n+2},$$

A worked out example

Let us write everything for a case, let us start from the 2nd order BDF formula for ODEs

$$y^{(n+2)} - \frac{4}{3}y^{(n+1)} + \frac{1}{3}y^{(n)} = \frac{2}{3}\tau f_{n+2},$$

- First of all we write down the (ρ, σ) polynomials defining the scheme:

$$\rho(\zeta) = \zeta^2 - \frac{4}{3}\zeta + \frac{1}{3}, \qquad \sigma(\zeta) = \frac{2}{3}\zeta^2.$$

A worked out example

Let us write everything for a case, let us start from the 2nd order BDF formula for ODEs

$$y^{(n+2)} - \frac{4}{3}y^{(n+1)} + \frac{1}{3}y^{(n)} = \frac{2}{3}\tau f_{n+2},$$

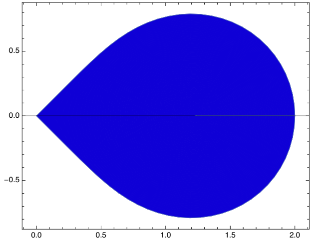
- First of all we write down the (ρ, σ) polynomials defining the scheme:

$$\rho(\zeta) = \zeta^2 - \frac{4}{3}\zeta + \frac{1}{3}, \qquad \sigma(\zeta) = \frac{2}{3}\zeta^2.$$

- Then we compute the generating function $\omega(\zeta)$

$$\omega(\zeta) = \frac{\rho(1/\zeta)}{\sigma(1/\zeta)} = \frac{2}{3\left(1 - 4\zeta/3 + \zeta^2/3\right)}.$$

$\alpha = 0.5$



$\{1/\omega(\zeta)^{\alpha} \,:\, |\zeta| \leq 1\}$

A worked out example

Now we need to expand the convolution coefficients of

$$\omega^\alpha(\zeta) = (\omega(\zeta))^\alpha = \frac{2^\alpha}{3^\alpha}(1 - {}^{4\zeta}\!/_3 + {}^{\zeta^2}\!/_3)^{-\alpha}.$$

A worked out example

Now we need to expand the convolution coefficients of

$$\omega^{\alpha}(\zeta) = (\omega(\zeta))^{\alpha} = \frac{2^{\alpha}}{3^{\alpha}}\left(1 - {}^{4\zeta}/_{3} + {}^{\zeta^2}/_{3}\right)^{-\alpha}.$$

Theorem (Henrici 1974, Theorem 1.6c, p. 42)

Let $\phi(\zeta) = 1 + \sum_{n=1}^{+\infty} a_n \zeta^n$ be a formal power series. Then for any $\alpha \in \mathbb{C}$, we have

$$(\phi(\zeta))^{\alpha} = \sum_{n=0}^{+\infty} v_n^{(\alpha)} \zeta^n,$$

where coefficients $v_n^{(\alpha)}$ can be evaluated recursively as

$$v_0^{(\alpha)} = 1, \qquad v_n^{(\alpha)} = \sum_{j=1}^{n} \left(\frac{(\alpha+1)j}{n} - 1 \right) a_j v_{n-j}^{(\alpha)}$$

A worked out example

Now we need to expand the convolution coefficients of

$$\omega^\alpha(\zeta) = (\omega(\zeta))^\alpha = \frac{2^\alpha}{3^\alpha}(1 - {}^{4\zeta}/_{3} + {}^{\zeta^2}/_{3})^{-\alpha}.$$

- $\omega_n = {}^{2^\alpha}/_{3^\alpha}\tilde{\omega}_n$,

A worked out example

Now we need to expand the convolution coefficients of

$$\omega^\alpha(\zeta) = (\omega(\zeta))^\alpha = \frac{2^\alpha}{3^\alpha}(1 - {}^{4\zeta}/_{3} + {}^{\zeta^2}/_{3})^{-\alpha}.$$

- $\omega_n = {}^{2^\alpha}/_{3^\alpha}\tilde{\omega}_n$,
- $a_1 = -{}^{4}/_{3}$, $a_2 = {}^{1}/_{3}$, $a_j = 0$ if $j \geq 3$, thus using

$$\tilde{\omega}_0^{(\alpha)} = 1, \qquad \tilde{\omega}_n^{(\alpha)} = \sum_{j=1}^{n} \left(\frac{(\alpha+1)j}{n} - 1 \right) a_j v_{n-j}^{(\alpha)}$$

A worked out example

Now we need to expand the convolution coefficients of

$$\omega^\alpha(\zeta) = (\omega(\zeta))^\alpha = \frac{2^\alpha}{3^\alpha}(1 - {}^{4\zeta}/_3 + {}^{\zeta^2}/_3)^{-\alpha}.$$

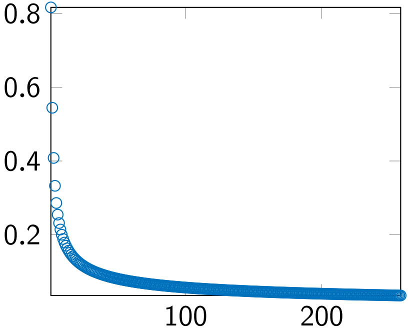
- $\omega_n = {}^{2^\alpha}/_{3^\alpha}\tilde{\omega}_n$,
- $a_1 = -{}^4/_3$, $a_2 = {}^1/_3$, $a_j = 0$ if $j \geq 3$, thus using

$$\tilde{\omega}_0^{(\alpha)} = 1, \qquad \tilde{\omega}_n^{(\alpha)} = \sum_{j=1}^{n} \left(\frac{(\alpha+1)j}{n} - 1 \right) a_j v_{n-j}^{(\alpha)}$$

- we get $\tilde{\omega}_0 = 1$, $\tilde{\omega}_1 = {}^4/_3\alpha\tilde{\omega}_0 = {}^{4\alpha}/_3$,

$$\tilde{\omega}_n = \frac{4}{3}\left(1 + \frac{\alpha-1}{n}\right)\tilde{\omega}_{n-1} + \frac{4}{3}\left(\frac{2(1-\alpha)}{n} - 1\right)\tilde{\omega}_{n-2}.$$

A worked out example

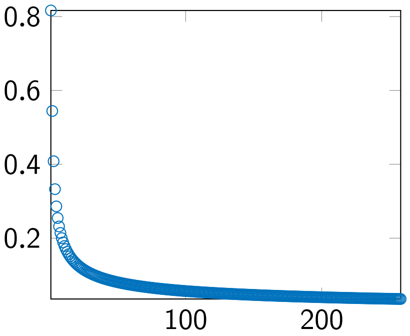


$\alpha = 0.5.$

- Since the a_j are a finite small number, we can compute the coefficients in an $O(N)$ operations,

```
omega = zeros(1,N+1) ;
onethird = 1/3 ; fourthird = 4/3;
twothird_oneminusalpha = 2/3*(1-alpha);
fourthird_oneminusalpha = 4/3*(1-alpha);
omega(1) = 1 ; omega(2) = fourthird*alpha*omega(1);
for n = 2 : N
 omega(n+1) = (fourthird -
 ↪  fourthird_oneminusalpha/n)*omega(n) + ...
 (twothird_oneminusalpha/n - onethird)*omega(n-1);
end
omega = omega*((2/3)^(alpha));
```

A worked out example

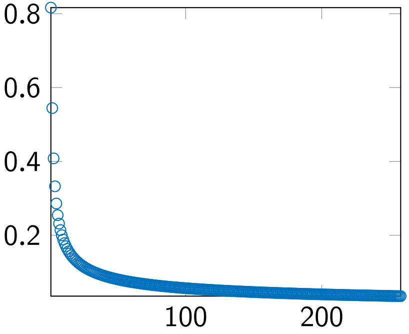


$\alpha = 0.5.$

- Since the a_j are a finite small number, we can compute the coefficients in an $O(N)$ operations,
- We can solve ${}_{CA}D^{0.5}_{[0,2]}y(t) = -2y(t)$, $y(0) = 1$

τ	$\lvert y^{(n)} - y(2)\rvert$	order
2^{-6}	1.44e-04	1.61
2^{-7}	4.42e-05	1.71
2^{-8}	1.28e-05	1.79
2^{-9}	3.57e-06	1.84
2^{-10}	9.68e-07	1.88
2^{-11}	2.85e-07	1.76
2^{-12}	8.17e-08	1.80
2^{-13}	2.29e-08	1.84
2^{-14}	6.27e-09	1.87

A worked out example



$\alpha = 0.5.$

- Since the a_j are a finite small number, we can compute the coefficients in an $O(N)$ operations,
- We can solve ${}_{CA}D^{0.5}_{[0,2]}y(t) = -2y(t)$, $y(0) = 1$
- For the starting weights we have to solve a 3×3 Vandermonde system:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & \sqrt{2} \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} w_{n,0} \\ w_{n,1} \\ w_{n,2} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

number of time-step times.

Reuse

Since we have a fixed time-grid we can reuse the same factorization for the Vandermonde system and compute all the weights in a single sweep.

Fractional Brusselator

The Brusselator is a model of the **autocatalytic chemical reaction**, it is described by

$$\begin{cases} \dot{x}_1 = a - (\mu + 1)x_1 + x_1^2 x_2, \\ \dot{x}_2 = \mu x_1 - x_1^2 x_2, \end{cases} \qquad a, \mu > 0.$$

- If $\mu > a^2 + 1$ then a single Brusselator has a *unique limit cycle*,

Fractional Brusselator

The Brusselator is a model of the **autocatalytic chemical reaction**, it is described by

$$\begin{cases} \dot{x}_1 = a - (\mu + 1)x_1 + x_1^2 x_2, \\ \dot{x}_2 = \mu x_1 - x_1^2 x_2, \end{cases} \qquad a, \mu > 0.$$

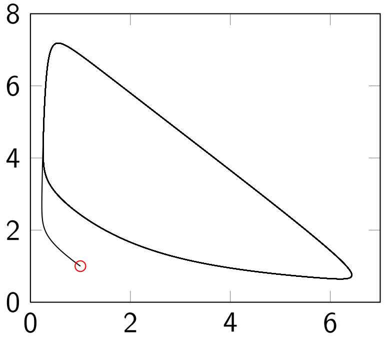
- If $\mu > a^2 + 1$ then a single Brusselator has a *unique limit cycle*,
- If $(a-1)^2 < \mu \leq a^2 + 1$ all the orbits tend to the steady state.

Fractional Brusselator

The Brusselator is a model of the **autocatalytic chemical reaction**, it is described by

$$\begin{cases} \dot{x}_1 = a - (\mu+1)x_1 + x_1^2 x_2, \\ \dot{x}_2 = \mu x_1 - x_1^2 x_2, \end{cases} \qquad a, \mu > 0.$$

```
a = 1 ; mu = 4 ;
param = [ a , mu ] ;
f_fun = @(t,y,par) [ ...
par(1) - (par(2)+1)*y(1) + y(1)^2*y(2) ; ...
par(2)*y(1) - y(1)^2*y(2) ] ;
t0 = 0 ; T = 100 ;
y0 = [ 1 ; 1] ;
[T,Y] = ode45(@(t,y)
↪  f_fun(t,y,param),[t0,T],y0);
figure(1)
plot(Y(:,1),Y(:,2),'k-',y(1,1),y(2,1),'ro')
```



Fractional Brusselator

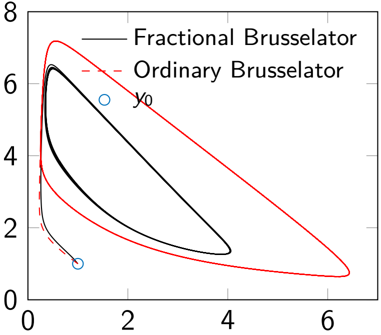
The Brusselator is a model of the **autocatalytic chemical reaction**, it is described by

$$\begin{cases} {}_{CA}D^{\alpha_1}x(t) = a - (\mu+1)x_1 + x_1^2 x_2, \\ {}_{CA}D^{\alpha_2}x(t) = \mu x_1 - x_1^2 x_2, \end{cases} \qquad a, \mu > 0.$$

```
alpha = [0.8,0.7] ;
h = 1e-2;
[t, y] =
↪  fde_pi1_ex(alpha,f_fun,t0,T,y0,h,param) ;
```

The cycle of the single fractional Brusselator is contained in the region

$$\left\{ (x_1, x_2) \,:\, \frac{a}{\mu+1} < x_1 < \frac{2a}{\mu},\ 0 < x_2 < \frac{\mu(1+\mu)}{a} \right\}$$



Fractional Brusselator

The Brusselator is a model of the **autocatalytic chemical reaction**, it is described by

$$\begin{cases} {}_{CA}D^{\alpha_1} x(t) = a - (\mu + 1)x_1 + x_1^2 x_2, \\ {}_{CA}D^{\alpha_2} x(t) = \mu x_1 - x_1^2 x_2, \end{cases} \qquad a, \mu > 0.$$

```
alpha = [0.8,0.7] ;
h = 1e-2;
[t, y] =
↪  fde_pi1_ex(alpha,f_fun,t0,T,y0,h,param) ;
```

The cycle of the single fractional Brusselator is contained in the region

$$\left\{ (x_1, x_2) \,:\, \frac{a}{\mu + 1} < x_1 < \frac{2a}{\mu},\ 0 < x_2 < \frac{\mu(1+\mu)}{a} \right\}$$

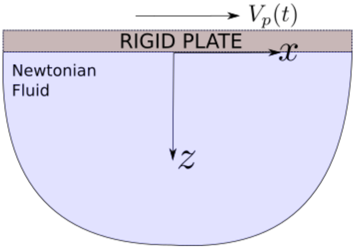
Of interest (Wang and Li 2007)

Finding the smallest values α_1, α_2 for which a limit cycle exist is of interest.

Bagley-Torvik Model (Bagley and Torvik 1986)

Stoke's Second Problem

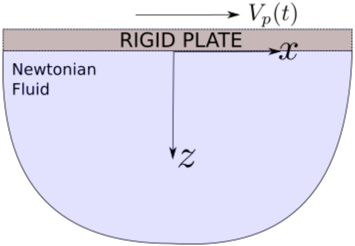
Can we determine the behavior of a half-space of Newtonian, viscous fluid undergoing the motion induced by the prescribed uniform sinusoidal motion of a plate on the surface?



Bagley-Torvik Model (Bagley and Torvik 1986)

Stoke's Second Problem

Can we determine the behavior of a half-space of Newtonian, viscous fluid undergoing the motion induced by the prescribed uniform sinusoidal motion of a plate on the surface?



If we write down the **equation of motion** we find

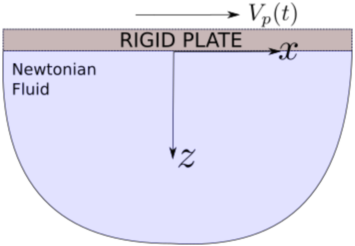
$$\rho \frac{\partial v}{\partial t} = \mu \frac{\partial^2 v}{\partial z^2}$$

- ρ is the *fluid density*, μ is the viscosity, v is the profile of the *transverse fluid velocity*.

Bagley-Torvik Model (Bagley and Torvik 1986)

Stoke's Second Problem

Can we determine the behavior of a half-space of Newtonian, viscous fluid undergoing the motion induced by the prescribed uniform sinusoidal motion of a plate on the surface?



If we write down the **equation of motion** we find

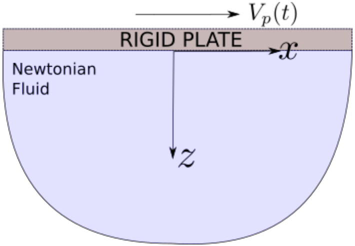
$$\rho[s\tilde{v}(s,z) - v(0,x)] = \mu \frac{d^2\tilde{v}(s,z)}{dz^2},$$

- ρ is the *fluid density*, μ is the viscosity, v is the profile of the *transverse fluid velocity*.
- We apply Laplace transform to the equation $\tilde{v} = \mathcal{L}v(s)$,

Bagley-Torvik Model (Bagley and Torvik 1986)

Stoke's Second Problem

Can we determine the behavior of a half-space of Newtonian, viscous fluid undergoing the motion induced by the prescribed uniform sinusoidal motion of a plate on the surface?



If we write down the **equation of motion** we find

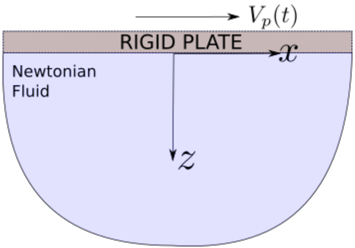
$$\tilde{v}(s,z) = \tilde{v}_p(s) \exp\left(\sqrt{\frac{\rho s}{\mu}} z\right)$$

- ρ is the *fluid density*, μ is the viscosity, v is the profile of the *transverse fluid velocity*.
- We apply Laplace transform to the equation $\tilde{v} = \mathcal{L}v(s)$,
- We solve and impose the boundary condition given by the $\tilde{v}_p = \mathcal{L}V_p(s)$,

Bagley-Torvik Model (Bagley and Torvik 1986)

Stoke's Second Problem

Can we determine the behavior of a half-space of Newtonian, viscous fluid undergoing the motion induced by the prescribed uniform sinusoidal motion of a plate on the surface?



If we write down the **equation of motion** we find

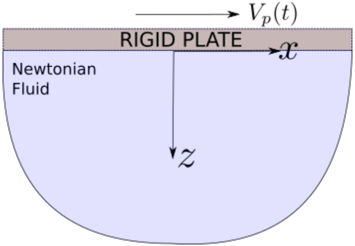
$$\tilde{\sigma}(s,z) = \sqrt{\mu\rho}\sqrt{s}\tilde{v}(s,z) = \sqrt{\mu\rho}\frac{1}{\sqrt{s}}s\tilde{v}(s,z)$$

- ρ is the *fluid density*, μ is the viscosity, v is the profile of the *transverse fluid velocity*.
- We apply Laplace transform to the equation $\tilde{v} = \mathcal{L}v(s)$,
- We solve and impose the boundary condition given by the $\tilde{v}_p = \mathcal{L}V_p(s)$,
- Since the *shear stress* is given by $\sigma(t,z) = \mu v_z(t,z)$ we can write its Laplace transform.

Bagley-Torvik Model (Bagley and Torvik 1986)

Stoke's Second Problem

Can we determine the behavior of a half-space of Newtonian, viscous fluid undergoing the motion induced by the prescribed uniform sinusoidal motion of a plate on the surface?



If we write down the **equation of motion** we find

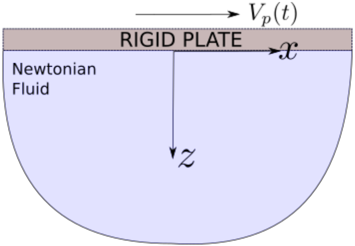
$$\tilde{\sigma}(s,z) = \sqrt{\mu\rho}\mathcal{L}\left\{\frac{1}{\Gamma(1/2)t^{1/2}}\right\} * \mathcal{L}\{v_t\}$$

- ρ is the *fluid density*, μ is the viscosity, v is the profile of the *transverse fluid velocity*.
- We apply Laplace transform to the equation $\tilde{v} = \mathcal{L}v(s)$,
- We solve and impose the boundary condition given by the $\tilde{v}_p = \mathcal{L}V_p(s)$,
- Since the *shear stress* is given by $\sigma(t,z) = \mu v_z(t,z)$ we can write its Laplace transform.
- Finally we invert it.

Bagley-Torvik Model (Bagley and Torvik 1986)

Stoke's Second Problem

Can we determine the behavior of a half-space of Newtonian, viscous fluid undergoing the motion induced by the prescribed uniform sinusoidal motion of a plate on the surface?



If we write down the **equation of motion** we find

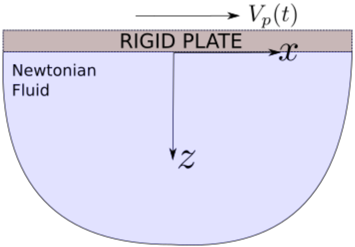
$$\sigma(t,z) = \sqrt{\mu\rho}\frac{1}{\Gamma(1/2)}\int_0^t (t-\tau)^{1/2} v_\tau(\tau,z)\,\mathrm{d}\tau.$$

- ρ is the *fluid density*, μ is the viscosity, v is the profile of the *transverse fluid velocity*.
- We apply Laplace transform to the equation $\tilde{v} = \mathcal{L}v(s)$,
- We solve and impose the boundary condition given by the $\tilde{v}_p = \mathcal{L}V_p(s)$,
- Since the *shear stress* is given by $\sigma(t,z) = \mu v_z(t,z)$ we can write its Laplace transform.
- Finally we invert it.

Bagley-Torvik Model (Bagley and Torvik 1986)

Stoke's Second Problem

Can we determine the behavior of a half-space of Newtonian, viscous fluid undergoing the motion induced by the prescribed uniform sinusoidal motion of a plate on the surface?

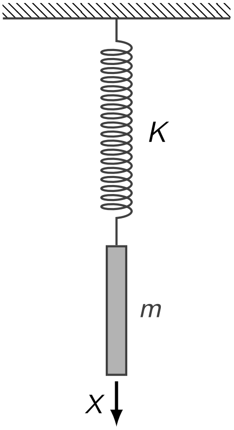


If we write down the **equation of motion** we find

$$\sigma(t,z) = \sqrt{\mu\rho}\,{}_{CA}D^{1/2}_{[0,t]} v(t,z).$$

- ρ is the *fluid density*, μ is the viscosity, v is the profile of the *transverse fluid velocity*.
- We apply Laplace transform to the equation $\tilde{v} = \mathcal{L}v(s)$,
- We solve and impose the boundary condition given by the $\tilde{v}_p = \mathcal{L}V_p(s)$,
- Since the *shear stress* is given by $\sigma(t,z) = \mu v_z(t,z)$ we can write its Laplace transform.
- Finally we invert it.

Bagley-Torvik Model (Bagley and Torvik 1986)

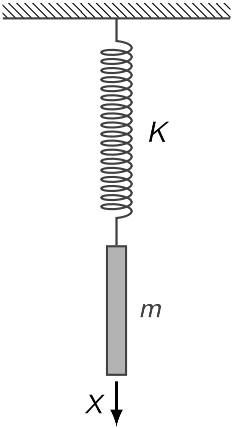


Immersed Plate

Assumptions:

- The spring is massless and its oscillations do not disturb the fluid,
- The area A of the plate is sufficiently large as to produce in the fluid adjacent to the plate the velocity field and stresses we just derived,

Bagley-Torvik Model (Bagley and Torvik 1986)



Immersed Plate

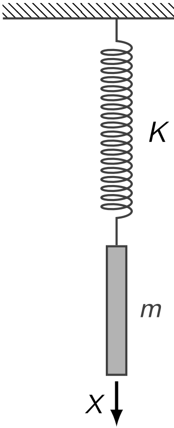
Assumptions:

- The spring is massless and its oscillations do not disturb the fluid,
- The area A of the plate is sufficiently large as to produce in the fluid adjacent to the plate the velocity field and stresses we just derived,

Deriving the equation:

$$m\ddot{X} = F_X = -KX - 2A\sigma(t, 0)$$

Bagley-Torvik Model (Bagley and Torvik 1986)



Immersed Plate

Assumptions:

- The spring is massless and its oscillations do not disturb the fluid,
- The area A of the plate is sufficiently large as to produce in the fluid adjacent to the plate the velocity field and stresses we just derived,

Deriving the equation:

$$m\ddot{X} = F_X = -KX - 2A\sigma(t,0)$$

Using the expression for the *strain* and $V_p(t,0) = \dot{X}(t)$ we find

$$m\ddot{X} + 2A\sqrt{\mu\rho}\,{}_{CA}D^{3/2}_{[0,t]}X + KX = 0.$$

Linear Multi-Term FDEs

The Bagley-Torvik model is an example of a Linear Multi-Term FDE, that is, something of the form

$$\lambda_{Q\,CA}D^{\alpha_Q}y(t) + \lambda_{Q-1\,CA}D^{\alpha_{Q-1}}y(t) + \cdots + \lambda_{2\,CA}D^{\alpha_2}y(t) + \lambda_{1\,CA}D^{\alpha_1}y(t) = f(t, y(t)),$$

with

- $\lambda_i \in \mathbb{R}\ \forall\, i = 1, \ldots, Q$,
- $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_{Q-1} < \alpha_Q$ and $\alpha_Q \neq 0$.

For this problem we have $m_Q = \max m_i$, $m_i = \lceil \alpha_i \rceil$, $i = 1, \ldots, Q$ **initial conditions**:

$$y(t_0) = y_0, y'(t_0) = y_0^{(1)}, \ldots, y^{(m_Q-1)}(t_0) = y_0^{(m_Q-1)}.$$

Linear Multi-Term FDEs

The Bagley-Torvik model is an example of a Linear Multi-Term FDE, that is, something of the form

$$\lambda_Q {}_{CA}D^{\alpha_Q} y(t) + \lambda_{Q-1} {}_{CA}D^{\alpha_{Q-1}} y(t) + \cdots + \lambda_2 {}_{CA}D^{\alpha_2} y(t) + \lambda_1 {}_{CA}D^{\alpha_1} y(t) = f(t, y(t)),$$

with

- $\lambda_i \in \mathbb{R} \; \forall\, i = 1, \ldots, Q$,
- $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_{Q-1} < \alpha_Q$ and $\alpha_Q \neq 0$.

For this problem we have $m_Q = \max m_i$, $m_i = \lceil \alpha_i \rceil$, $i = 1, \ldots, Q$ **initial conditions**:

$$y(t_0) = y_0, y'(t_0) = y_0^{(1)}, \ldots, y^{(m_Q-1)}(t_0) = y_0^{(m_Q-1)}.$$

❓ How can we solve them?

Linear Multi-Term FDEs

We need to **recall one of the properties** we have seen of the Caputo derivatives

(P_1) $I^{\alpha}_{[t_0,T]}\,{}_{CA}D^{\alpha}_{[t_0,T]}y(t) = y(t) - T_{m-1}[y,t_0](t)$,

(P_2) $I^{\beta}_{[t_0,T]}\,{}_{CA}D^{\alpha}_{[t_0,T]}y(t) = I^{\beta}_{[t_0,T]}\,{}_{RL}D^{\alpha}_{[t_0,T]}\left[y(t) - T_{m-1}[y;t_0](t)\right] = I^{\beta-\alpha}_{[t_0,T]}\left[y(t) - T_{m-1}[y;t_0](t)\right]$, $\beta > \alpha$.

Linear Multi-Term FDEs

We need to **recall one of the properties** we have seen of the Caputo derivatives

(P_1) $I^{\alpha}_{[t_0,T]}\,{}_{CA}D^{\alpha}_{[t_0,T]}y(t) = y(t) - T_{m-1}[y, t_0](t)$,

(P_2) $I^{\beta}_{[t_0,T]}\,{}_{CA}D^{\alpha}_{[t_0,T]}y(t) = I^{\beta}_{[t_0,T]}\,{}_{RL}D^{\alpha}_{[t_0,T]}\left[y(t) - T_{m-1}[y; t_0](t)\right] =$
$I^{\beta-\alpha}_{[t_0,T]}\left[y(t) - T_{m-1}[y; t_0](t)\right]$, $\beta > \alpha$.

We start from the multi-term equation

$$\lambda_Q\,{}_{CA}D^{\alpha_Q}y(t) + \lambda_{Q-1}\,{}_{CA}D^{\alpha_{Q-1}}y(t) + \cdots + \lambda_2\,{}_{CA}D^{\alpha_2}y(t) + \lambda_1\,{}_{CA}D^{\alpha_1}y(t) = f(t, y(t)),$$

Linear Multi-Term FDEs

We need to **recall one of the properties** we have seen of the Caputo derivatives

(P_1) $I^{\alpha}_{[t_0,T]}\,{}_{CA}D^{\alpha}_{[t_0,T]}y(t) = y(t) - T_{m-1}[y,t_0](t)$,

(P_2) $I^{\beta}_{[t_0,T]}\,{}_{CA}D^{\alpha}_{[t_0,T]}y(t) = I^{\beta}_{[t_0,T]}\,{}_{RL}D^{\alpha}_{[t_0,T]}\left[y(t) - T_{m-1}[y;t_0](t)\right] =$
$I^{\beta-\alpha}_{[t_0,T]}\left[y(t) - T_{m-1}[y;t_0](t)\right]$, $\beta > \alpha$.

We start from the multi-term equation

$$\lambda_Q I^{\alpha_Q}_{[t_0,T]}\left[{}_{CA}D^{\alpha_Q}y(t)\right] = -I^{\alpha_Q}_{[t_0,T]}\left[\sum_{i=1}^{Q-1}\lambda_i\,{}_{CA}D^{\alpha_i}y(t) + f(t,y(t))\right],$$

- we multiply both sides by $I^{\alpha_Q}_{[t_0,T]}$,

Linear Multi-Term FDEs

We need to **recall one of the properties** we have seen of the Caputo derivatives

(P_1) $I^{\alpha}_{[t_0,T]}\, {}_{CA}D^{\alpha}_{[t_0,T]} y(t) = y(t) - T_{m-1}[y,t_0](t)$,

(P_2) $I^{\beta}_{[t_0,T]}\, {}_{CA}D^{\alpha}_{[t_0,T]} y(t) = I^{\beta}_{[t_0,T]}\, {}_{RL}D^{\alpha}_{[t_0,T]} [y(t) - T_{m-1}[y;t_0](t)] =$
$I^{\beta-\alpha}_{[t_0,T]} [y(t) - T_{m-1}[y;t_0](t)]$, $\beta > \alpha$.

We start from the multi-term equation

$$y(t) - T_{m_Q-1}[y,t_0](t) = -\sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I^{\alpha_Q-\alpha_i}_{[t_0,t]} [y(t) - T_{m_i-1}[y;t_0](t)] + \frac{1}{\lambda_Q} I^{\alpha_Q}_{[t_0,T]} f(t,y(t))$$

- we multiply both sides by $I^{\alpha_Q}_{[t_0,T]}$,
- we use P_1 on the left-hand side, P_2 on the right-hand side,

Linear Multi-Term FDEs

We need to **recall one of the properties** we have seen of the Caputo derivatives

(P_1) $I^{\alpha}_{[t_0,T]}\, {}_{CA}D^{\alpha}_{[t_0,T]} y(t) = y(t) - T_{m-1}[y, t_0](t)$,

(P_2) $I^{\beta}_{[t_0,T]}\, {}_{CA}D^{\alpha}_{[t_0,T]} y(t) = I^{\beta}_{[t_0,T]}\, {}_{RL}D^{\alpha}_{[t_0,T]} \left[y(t) - T_{m-1}[y;t_0](t)\right] =$
$I^{\beta-\alpha}_{[t_0,T]} \left[y(t) - T_{m-1}[y;t_0](t)\right]$, $\beta > \alpha$.

We start from the multi-term equation

$$y(t) = T_{m_Q-1}[y,t_0](t) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I^{\alpha_Q-\alpha_i}_{[t_0,t]} \left[y(t) - T_{m_i-1}[y;t_0](t)\right] + \frac{1}{\lambda_Q} I^{\alpha_Q}_{[t_0,T]} f(t, y(t))$$

- we multiply both sides by $I^{\alpha_Q}_{[t_0,T]}$,
- we use P_1 on the left-hand side, P_2 on the right-hand side,
- and re-arrange to get an expression for the solution.

Linear Multi-Term FDEs: generalizing PI rules

First we do a bit of rewriting of

$$y(t) = T_{m_Q-1}[y,t_0](t) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I^{\alpha_Q-\alpha_i}_{[t_0,t]} [y(t) - T_{m_i-1}[y;t_0](t)] + \frac{1}{\lambda_Q} I^{\alpha_Q}_{[t_0,T]} f(t,y(t))$$

- we employ the usual fractional integral for polynomials:

$$I^{\alpha}_{[t_0,t]} T_{m-1}[y;t_0](t) = \sum_{k=0}^{m-1} \frac{(t-t_0)^{k+\alpha}}{\Gamma(k+\alpha)} y^{(k)}(t_0), \quad \begin{array}{l} \alpha \in \{\alpha_1,\ldots,\alpha_{Q-1}\}, \\ m \in \{m_1,\ldots,m_{Q-1}\}. \end{array}$$

Linear Multi-Term FDEs: generalizing PI rules

First we do a bit of rewriting of

$$y(t) = T_{m_Q-1}[y, t_0](t) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I^{\alpha_Q-\alpha_i}_{[t_0,t]} [y(t) - T_{m_i-1}[y; t_0](t)] + \frac{1}{\lambda_Q} I^{\alpha_Q}_{[t_0,T]} f(t, y(t))$$

- we employ the usual fractional integral for polynomials:

$$I^{\alpha}_{[t_0,t]} T_{m-1}[y; t_0](t) = \sum_{k=0}^{m-1} \frac{(t-t_0)^{k+\alpha}}{\Gamma(k+\alpha)} y^{(k)}(t_0), \quad \begin{array}{l} \alpha \in \{\alpha_1, \ldots, \alpha_{Q-1}\}, \\ m \in \{m_1, \ldots, m_{Q-1}\}. \end{array}$$

- We use it to simplify the expression

$$\tilde{T}(t) = T_{m_Q-1}[y; t_0](t) + \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} \sum_{k=0}^{m_i-1} \frac{(t-t_0)^{k+\alpha_Q-\alpha_i}}{\Gamma(k+\alpha_Q-\alpha_i+1)} y^{(k)}(t_0).$$

Linear Multi-Term FDEs: generalizing PI rules

First we do a bit of rewriting of

$$y(t) = \tilde{T}(t) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I^{\alpha_Q - \alpha_i}_{[t_0,t]} y(t) + \frac{1}{\lambda_Q} I^{\alpha_Q}_{[t_0,T]} f(t, y(t)).$$

- we employ the usual fractional integral for polynomials:

$$I^{\alpha}_{[t_0,t]} T_{m-1}[y; t_0](t) = \sum_{k=0}^{m-1} \frac{(t-t_0)^{k+\alpha}}{\Gamma(k+\alpha)} y^{(k)}(t_0), \quad \begin{array}{l} \alpha \in \{\alpha_1, \ldots, \alpha_{Q-1}\}, \\ m \in \{m_1, \ldots, m_{Q-1}\}. \end{array}$$

- We use it to simplify the expression

$$\tilde{T}(t) = T_{m_Q-1}[y; t_0](t) + \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} \sum_{k=0}^{m_i-1} \frac{(t-t_0)^{k+\alpha_Q-\alpha_i}}{\Gamma(k+\alpha_Q-\alpha_i+1)} y^{(k)}(t_0).$$

Linear Multi-Term FDEs: generalizing PI rules

Now we have an expression that we can treat by adapting one of the Product Integral rules

$$y(t) = \tilde{T}(t) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I_{[t_0,t]}^{\alpha_Q-\alpha_i} y(t) + \frac{1}{\lambda_Q} I_{[t_0,T]}^{\alpha_Q} f(t, y(t)).$$

Linear Multi-Term FDEs: generalizing PI rules

Now we have an expression that we can treat by adapting one of the Product Integral rules

$$y(t) = \tilde{T}(t) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I^{\alpha_Q-\alpha_i}_{[t_0,t]} y(t) + \frac{1}{\lambda_Q} I^{\alpha_Q}_{[t_0,T]} f(t, y(t)).$$

We can start from the **explicit rectangular product integral rule** on a uniform grid

$$y^{(n)} = \tilde{T}(t_n) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} \tau^{\alpha_Q-\alpha_i} \sum_{j=0}^{n-1} b^{(\alpha_Q-\alpha_i)}_{n-j-1} y^{(j)} + \frac{1}{\lambda_Q} \sum_{j=0}^{n-1} b^{(\alpha_Q)}_{n-j-1} f(t_j, y^{(j)}).$$

with

$$b_n^{(\alpha)} = [(n+1)^\alpha - n^\alpha]/\alpha, \quad n = 1, \ldots, N.$$

Linear Multi-Term FDEs: generalizing PI rules

Now we have an expression that we can treat by adapting one of the Product Integral rules

$$y(t) = \tilde{T}(t) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I^{\alpha_Q - \alpha_i}_{[t_0,t]} y(t) + \frac{1}{\lambda_Q} I^{\alpha_Q}_{[t_0,T]} f(t, y(t)).$$

We can start from the **implicit rectangular product integral rule** on a uniform grid

$$y^{(n)} = \tilde{T}(t_n) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} \tau^{\alpha_Q - \alpha_i} \sum_{j=0}^{n} b^{(\alpha_Q - \alpha_i)}_{n-j-1} y^{(j)} + \frac{1}{\lambda_Q} \sum_{j=0}^{n} b^{(\alpha_Q)}_{n-j-1} f(t_j, y^{(j)}).$$

with

$$b^{(\alpha)}_n = [(n+1)^\alpha - n^\alpha]/\alpha, \quad n = 1, \ldots, N.$$

Linear Multi-Term FDEs: generalizing PI rules

Now we have an expression that we can treat by adapting one of the Product Integral rules

$$y(t) = \tilde{T}(t) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} I_{[t_0,t]}^{\alpha_Q-\alpha_i} y(t) + \frac{1}{\lambda_Q} I_{[t_0,T]}^{\alpha_Q} f(t, y(t)).$$

We can start from the **implicit rectangular product integral rule** on a uniform grid

$$y^{(n)} = \tilde{T}(t_n) - \sum_{i=1}^{Q-1} \frac{\lambda_i}{\lambda_Q} \tau^{\alpha_Q-\alpha_i} \sum_{j=0}^{n} b_{n-j-1}^{(\alpha_Q-\alpha_i)} y^{(j)} + \frac{1}{\lambda_Q} \sum_{j=0}^{n} b_{n-j-1}^{(\alpha_Q)} f(t_j, y^{(j)}).$$

with

$$b_n^{(\alpha)} = [(n+1)^{\alpha} - n^{\alpha}]/\alpha, \quad n = 1, \ldots, N.$$

We can do it similarly for the **Implicit Trapezoidal Rule** and then for the **Predictor-Corrector method** (Diethelm 2003).

Linear Multi-Term FDEs: generalizing PI rules

❓ Can we do something similar for FLMMs?

Linear Multi-Term FDEs: generalizing PI rules

Can we do something similar for FLMMs?

We don't know how to determine the starting values $w_{n,j}$ for the quadrature. Thus this approach is not viable.

Linear Multi-Term FDEs: generalizing PI rules

Can we do something similar for FLMMs?

We don't know how to determine the starting values $w_{n,j}$ for the quadrature. Thus this approach is not viable.

Available codes (Garrappa 2018):

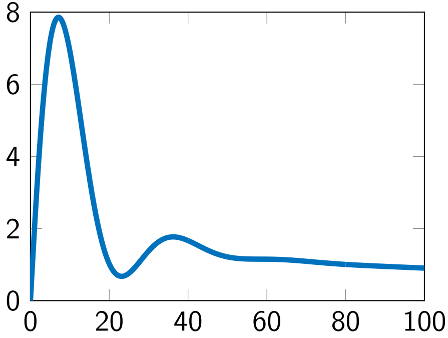
- `MT_FDE_PI1_Ex.m` - Explicit Product-Integration of rectanguar type
- `MT_FDE_PI1_Im.m` - Implicit Product-Integration of rectanguar type
- `MT_FDE_PI2_Im.m` - Implicit Product-Integration of trapezoidal type
- `MT_FDE_PI12_PC.m`- Product-Integration with predictor-corrector

Linear Multi-Term FDEs: back to Bagley-Torvik

We reached the equation

$$m\ddot{X} + 2A\sqrt{\mu\rho}\, {}_{CA}D^{3/2}_{[0,t]} X + KX = 0.$$

```
m = 10; A = 6; K = 3;
mu = 2; rho = 2;
alpha = [2 3/2] ;
lambda = [m 2*A*sqrt(mu*rho)] ;
f_fun = @(t,X) -K*X;
J_fun = @(t,X) -K;
t0 = 0 ; T = 100 ;
X0 = [0 , 2 ];
h = 1e-2;
[t, X] = mt_fde_pi1_ex(alpha, lambda, f_fun,
↪  t0, T, X0, h);
```

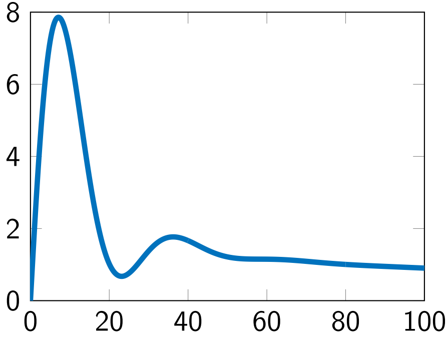


Linear Multi-Term FDEs: back to Bagley-Torvik

We reached the equation

$$m\ddot{X} + 2A\sqrt{\mu\rho}\,{}_{CA}D^{3/2}_{[0,t]}X + KX = 0.$$

```
m = 10; A = 6; K = 3;
mu = 2; rho = 2;
alpha = [2 3/2] ;
lambda = [m 2*A*sqrt(mu*rho)] ;
f_fun = @(t,X) -K*X;
J_fun = @(t,X) -K;
t0 = 0 ; T = 100 ;
X0 = [0 , 2 ];
h = 1e-2;
[t, X] = mt_fde_pi1_ex(alpha, lambda, f_fun,
↪  t0, T, X0, h);
```



But does it fit the reality?

Linear Multi-Term FDEs: back to Bagley-Torvik

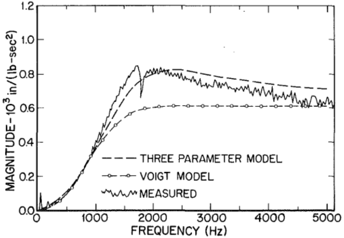


Fig. 5 The magnitude of the transfer function for Case 1

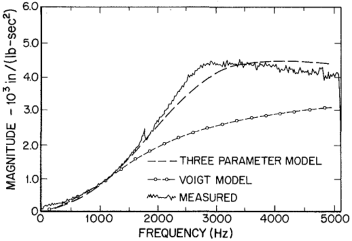


Fig. 7 The magnitude of the transfer function for Case 5

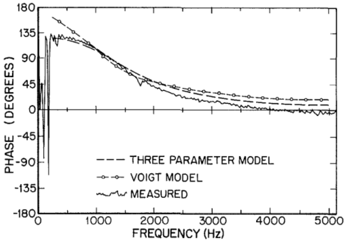


Fig. 6 The phase of the transfer function for Case 1

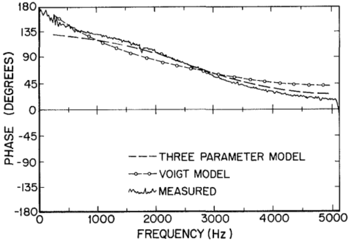


Fig. 8 The phase of the transfer function for Case 5

The model we have derived is a model of the form

$$\sigma(t) = G_0\epsilon(t) + G_1\dot{\epsilon}(t),$$
$$\epsilon(t) = \frac{x(t)}{\delta}$$
$$f(t) = m\ddot{x}(t) + f_p(t),$$
$$f_p(t) = \frac{2A}{\delta}(G_0 + G_1 {}_{CA}D^{\alpha}x(t)).$$

One can do *parameter tuning* to find the fractional order from experimental data and compare the results with the integer-order model. The results on the left by Bagley and Torvik 1986 show that the fractional model obtain a better fit with the measured data.

Equivalent formulations of the Multi-Term FDEs

In the integer-order case we know how to rewrite the equation

$$y^{(n)}(t) = f(t, y^{(n-1)}(t), \ldots, y^{(1)}(t), y(t)), \quad y^{(j)}(0) = y_0^{(j)}, \quad j = 0, 1, \ldots, n-1,$$

as a **system of first-order equations**.

Equivalent formulations of the Multi-Term FDEs

In the integer-order case we know how to rewrite the equation

$$y^{(n)}(t) = f(t, y^{(n-1)}(t), \ldots, y^{(1)}(t), y(t)), \quad y^{(j)}(0) = y_0^{(j)}, \quad j = 0, 1, \ldots, n-1,$$

as a **system of first-order equations**.

❓ Can we do something similar in the fractional case?

Equivalent formulations of the Multi-Term FDEs

In the integer-order case we know how to rewrite the equation

$$y^{(n)}(t) = f(t, y^{(n-1)}(t), \dots, y^{(1)}(t), y(t)), \quad y^{(j)}(0) = y_0^{(j)}, \quad j = 0, 1, \dots, n-1,$$

as a **system of first-order equations**.

(A1) Let us assume that our multi-term equation is of the form

$$ {}_{CA}D^{\alpha_k} y(t) = f(t, {}_{CA}D^{\alpha_{k-1}} y(t), \dots, {}_{CA}D^{\alpha_1}(t), y(t)), \quad \begin{array}{l} y^{(j)}(0) = y_0^{(j)}, \\ j = 0, 1, \dots, n-1, \end{array}$$

for $\alpha_k > \alpha_{k-1} > \cdots > \alpha_1 > 0$, $\alpha_j - \alpha_{j-1} \leq 1 \; \forall j = 1, 2, \dots, k$, $0 < \alpha_1 \leq 1$.

Equivalent formulations of the Multi-Term FDEs

In the integer-order case we know how to rewrite the equation

$$y^{(n)}(t) = f(t, y^{(n-1)}(t), \ldots, y^{(1)}(t), y(t)), \quad y^{(j)}(0) = y_0^{(j)}, \quad j = 0, 1, \ldots, n-1,$$

as a **system of first-order equations**.

(A1) Let us assume that our multi-term equation is of the form

$$ {}_{CA}D^{\alpha_k} y(t) = f(t, {}_{CA}D^{\alpha_{k-1}} y(t), \ldots, {}_{CA}D^{\alpha_1}(t), y(t)), \quad \begin{array}{l} y^{(j)}(0) = y_0^{(j)}, \\ j = 0, 1, \ldots, n-1, \end{array}$$

for $\alpha_k > \alpha_{k-1} > \cdots > \alpha_1 > 0$, $\alpha_j - \alpha_{j-1} \leq 1 \; \forall j = 1, 2, \ldots, k$, $0 < \alpha_1 \leq 1$.

(A2) Assume also that $\alpha_j \in \mathbb{Q} \; \forall j = 1, 2, \ldots, k$, and that M is the least common multiple of $\alpha_1, \alpha_2, \ldots, \alpha_k$.

Equivalent formulations of the Multi-Term FDEs

Theorem (Diethelm 2010, Theorem 8.1)

Under the assumptions (A1) and (A2), set $\gamma = {}^1\!/_M$, and $N = M\alpha_k$, then the IVP is equivalent to

$$\begin{cases} {}_{CA}D^{\gamma} y_0(t) = y_1(t), \\ {}_{CA}D^{\gamma} y_1(t) = y_2(t), \\ \vdots \\ {}_{CA}D^{\gamma} y_{N-2}(t) = y_{N-1}(t), \\ {}_{CA}D^{\gamma} y_{N-1}(t) = f(t, y_0(t), y_{\alpha_{k-1}/M}(t), \ldots, y_{\alpha_1/M}(t), y(t)) \end{cases} \qquad y_i(0) = \begin{cases} y_0^{(j/m)}, & \text{if } \frac{j}{M} \in \mathbb{N}_0, \\ 0, & \text{otherwise.} \end{cases}$$

$\Rightarrow$ whenever $\mathbf{y} = (y_0, \ldots, y_{N-1})^T$ with $y_0 \in \mathcal{C}^{\lceil \alpha_k \rceil}[0, b]$, for some $b > 0$, is a solution of the N-dimensional system, then $y \equiv y_0$ is a solution of the multi-term FDE.

Equivalent formulations of the Multi-Term FDEs

Theorem (Diethelm 2010, Theorem 8.1)

Under the assumptions (A1) and (A2), set $\gamma = 1/M$, and $N = M\alpha_k$, then the IVP is equivalent to

$$\begin{cases} {}_{CA}D^{\gamma} y_0(t) = y_1(t), \\ {}_{CA}D^{\gamma} y_1(t) = y_2(t), \\ \vdots \\ {}_{CA}D^{\gamma} y_{N-2}(t) = y_{N-1}(t), \\ {}_{CA}D^{\gamma} y_{N-1}(t) = f(t, y_0(t), y_{\alpha_{k-1/M}}(t), \ldots, y_{\alpha_{1/M}}(t), y(t)) \end{cases} \qquad y_i(0) = \begin{cases} y_0^{(j/m)}, & \text{if } \frac{j}{M} \in \mathbb{N}_0, \\ 0, & \text{otherwise.} \end{cases}$$

$\Leftarrow$ whenever $y \in \mathcal{C}^{\lceil \alpha_k \rceil}([0, b])$ is a solution of the multi-term FDE, then the vector function $\mathbf{y} = (y, {}_{CA}D^{\gamma} y, {}_{CA}D^{2\gamma} y, \ldots, {}_{CA}D^{(N-1)\gamma} y)^T$ solves the N-dimensional system.

Equivalent formulations of the Multi-Term FDEs

We can relax (A2) from the *rationality requirement* to a requirement on being commensurable[2].

(A2)' Let $1 \geq \alpha_k > \alpha_{k-1} > \ldots > \alpha_1 > 0$ and assume the equation to be *commensurate*, then we define $\tilde{\alpha}_j = {}^{\alpha_j}/_{\alpha_1}$ for $j = 1, \ldots, k$, let $\tilde{M}$ be the least common multiple of the denominators of the values $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_k$.

Theorem (Diethelm 2010, Theorem 8.2)

Under the assumption (A1) and (A2)', set $\gamma = {}^{\alpha_1}/_{\tilde{M}}$ and $N = {}^{\tilde{M}\alpha_k}/_{\alpha_1}$, then the equivalence relation of the N-dimensional system and of the multi-term FDE holds as in the previous result.

[2]Two non-zero real numbers α and β are said to be commensurable if their ratio ${}^{\alpha}/_{\beta} \in \mathbb{Q}$.

Equivalent formulations of the Multi-Term FDEs

We can relax (A2) from the *rationality requirement* to a requirement on being commensurable[2].

(A2)' Let $1 \geq \alpha_k > \alpha_{k-1} > \ldots > \alpha_1 > 0$ and assume the equation to be *commensurate*, then we define $\tilde{\alpha}_j = {}^{\alpha_j}/_{\alpha_1}$ for $j = 1, \ldots, k$, let $\tilde{M}$ be the least common multiple of the denominators of the values $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_k$.

Theorem (Diethelm 2010, Theorem 8.2)

Under the assumption (A1) and (A2)', set $\gamma = {}^{\alpha_1}/_{\tilde{M}}$ and $N = {}^{\tilde{M}\alpha_k}/_{\alpha_1}$, then the equivalence relation of the N-dimensional system and of the multi-term FDE holds as in the previous result.

- Existence and uniqueness results can be obtained for the single term reformulation,

[2] Two non-zero real numbers α and β are said to be commensurable if their ratio ${}^{\alpha}/_{\beta} \in \mathbb{Q}$.

Equivalent formulations of the Multi-Term FDEs

We can relax (A2) from the *rationality requirement* to a requirement on being commensurable[2].

(A2)' Let $1 \geq \alpha_k > \alpha_{k-1} > \ldots > \alpha_1 > 0$ and assume the equation to be *commensurate*, then we define $\tilde{\alpha}_j = {}^{\alpha_j}/_{\alpha_1}$ for $j = 1, \ldots, k$, let $\tilde{M}$ be the least common multiple of the denominators of the values $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_k$.

Theorem (Diethelm 2010, Theorem 8.2)

Under the assumption (A1) and (A2)', set $\gamma = {}^{\alpha_1}/_{\tilde{M}}$ and $N = {}^{\tilde{M}\alpha_k}/_{\alpha_1}$, then the equivalence relation of the N-dimensional system and of the multi-term FDE holds as in the previous result.

- Existence and uniqueness results can be obtained for the single term reformulation,
- See (Ford and Connolly 2009) for other reformulations and comparisons.

[2] Two non-zero real numbers α and β are said to be commensurable if their ratio ${}^{\alpha}/_{\beta} \in \mathbb{Q}$.

The Method of Lines

Consider a partial differential equations of the form

$$\text{Find } u(\mathbf{x}, t) \text{ s.t. } u_t = \mathcal{L}u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+,$$

where $\mathcal{L}$ is a differential operator, either linear or nonlinear, coupled with the opportune boundary conditions, and given suitable initial conditions.

The Method of Lines

Consider a partial differential equations of the form

$$\text{Find } u(\mathbf{x},t) \text{ s.t. } u_t = \mathcal{L}u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+,$$

where $\mathcal{L}$ is a differential operator, either linear or nonlinear, coupled with the opportune boundary conditions, and given suitable initial conditions.
A *classical way* of approaching this task is using a **Method Of Lines** (MOL) approach, that is

1. we discretize w.r.t. the *space variables* with some method (e.g., Finite Elements/Differences/Volumes, meshfree/meshless methods, spectral methods...)

$$M\mathbf{u}_t = F(t,\mathbf{u}), \quad M \in \mathbb{R}^{n_d \times n_d},\ F : \mathbb{R} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_d}, \mathbf{u} : \mathbb{R} \to \mathbb{R}^{n_d}.$$

The Method of Lines

Consider a partial differential equations of the form

$$\text{Find } u(\mathbf{x}, t) \text{ s.t. } u_t = \mathcal{L}u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+,$$

where $\mathcal{L}$ is a differential operator, either linear or nonlinear, coupled with the opportune boundary conditions, and given suitable initial conditions.
A *classical way* of approaching this task is using a **Method Of Lines** (MOL) approach, that is

1. we discretize w.r.t. the *space variables* with some method (e.g., Finite Elements/Differences/Volumes, meshfree/meshless methods, spectral methods...)

$$M\mathbf{u}_t = F(t, \mathbf{u}), \quad M \in \mathbb{R}^{n_d \times n_d}, \ F : \mathbb{R} \times \mathbb{R}^{n_d} \to \mathbb{R}^{n_d}, \mathbf{u} : \mathbb{R} \to \mathbb{R}^{n_d}.$$

2. now we have a (possibly nonlinear, non-autonomous) system of ODEs to which we can apply an integrator.

PDEs with fractional derivatives with respect to time

We can think of using the methods we have seen until now for solving PDEs in which the **derivative with respect to time** has been substituted by the **fractional derivative in the Caputo sense**

$$\text{Find } u(\mathbf{x}, t) \text{ s.t. } {}_{CA}D^{\alpha} u = \mathcal{L} u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+.$$

PDEs with fractional derivatives with respect to time

We can think of using the methods we have seen until now for solving PDEs in which the **derivative with respect to time** has been substituted by the **fractional derivative in the Caputo sense**

$$\text{Find } u(\mathbf{x},t) \text{ s.t. } {}_{CA}D^{\alpha}u = \mathcal{L}u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+.$$

Examples:

- Time-fractional diffusion equation

$${}_{CA}D_t^{\alpha}u = \operatorname{div}(p(x)\operatorname{grad} u) - q(x)u + F(x,t), \quad 0 < \alpha \leq 1.$$

PDEs with fractional derivatives with respect to time

We can think of using the methods we have seen until now for solving PDEs in which the **derivative with respect to time** has been substituted by the **fractional derivative in the Caputo sense**

$$\text{Find } u(\mathbf{x},t) \text{ s.t. } {}_{CA}D^{\alpha}u = \mathcal{L}u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+.$$

Examples:

- Time-fractional diffusion equation

$$ {}_{CA}D_t^{\alpha}u = \operatorname{div}(p(x)\operatorname{grad} u) - q(x)u + F(x,t), \quad 0 < \alpha \leq 1.$$

- Time-fractional advection-dispersion equation

$$ {}_{CA}D_t^{\alpha}u = \operatorname{div}(p(x)\operatorname{grad} u) - \nu \operatorname{grad}(u), \quad 0 < \alpha \leq 1.$$

PDEs with fractional derivatives with respect to time

We can think of using the methods we have seen until now for solving PDEs in which the **derivative with respect to time** has been substituted by the **fractional derivative in the Caputo sense**

$$\text{Find } u(\mathbf{x},t) \text{ s.t. } {}_{CA}D^{\alpha}u = \mathcal{L}u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+.$$

Examples:

1. Time-fractional diffusion equation

$$ {}_{CA}D_t^{\alpha}u = \operatorname{div}(p(x)\operatorname{grad} u) - q(x)u + F(x,t), \quad 0 < \alpha \leq 1.$$

2. Time-fractional advection-dispersion equation

$$ {}_{CA}D_t^{\alpha}u = \operatorname{div}(p(x)\operatorname{grad} u) - \nu \operatorname{grad}(u), \quad 0 < \alpha \leq 1.$$

3. Time-fractional Schrödinger equation

$$ (iT_\rho)^{\alpha}\, {}_{CA}D_t^{\alpha}\psi = -\frac{L_\rho^2}{2N_m}\nabla^2\psi + N_\nu\psi, \quad 0 < \alpha \leq 1.$$

PDEs with fractional derivatives with respect to time

We can think of using the methods we have seen until now for solving PDEs in which the **derivative with respect to time** has been substituted by the **fractional derivative in the Caputo sense**

$$\text{Find } u(\mathbf{x},t) \text{ s.t. } {}_{CA}D^{\alpha}u = \mathcal{L}u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+.$$

Examples:

- Time-fractional Burgers equation equation

$$ {}_{CA}D_t^{\alpha}u = u_{xx} + Au^p u_x, \quad 0 < \alpha \leq 1,\ p > 0.$$

PDEs with fractional derivatives with respect to time

We can think of using the methods we have seen until now for solving PDEs in which the **derivative with respect to time** has been substituted by the **fractional derivative in the Caputo sense**

$$\text{Find } u(\mathbf{x}, t) \text{ s.t. } {}_{CA}D^{\alpha} u = \mathcal{L}u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+.$$

Examples:

- Time-fractional Burgers equation equation

$$ {}_{CA}D_t^{\alpha} u = u_{xx} + A u^p u_x, \quad 0 < \alpha \leq 1,\ p > 0.$$

- Time-fractional Korteweg–de Vries equation

$$ {}_{CA}D_t^{\alpha} u = u_{xxx} + A u^p u_x, \quad 0 < \alpha \leq 1,\ p > 0.$$

PDEs with fractional derivatives with respect to time

We can think of using the methods we have seen until now for solving PDEs in which the **derivative with respect to time** has been substituted by the **fractional derivative in the Caputo sense**

$$\text{Find } u(\mathbf{x}, t) \text{ s.t. } {}_{CA}D^{\alpha} u = \mathcal{L} u, \quad \mathbf{x} \in \Omega \subseteq \mathbb{R}^d, t \in I \subseteq \mathbb{R}_+.$$

Examples:

- Time-fractional Burgers equation equation

$$ {}_{CA}D_t^{\alpha} u = u_{xx} + A u^p u_x, \quad 0 < \alpha \leq 1, \ p > 0.$$

- Time-fractional Korteweg–de Vries equation

$$ {}_{CA}D_t^{\alpha} u = u_{xxx} + A u^p u_x, \quad 0 < \alpha \leq 1, \ p > 0.$$

- Time-fractional (incompressible) Navier–Stokes equation

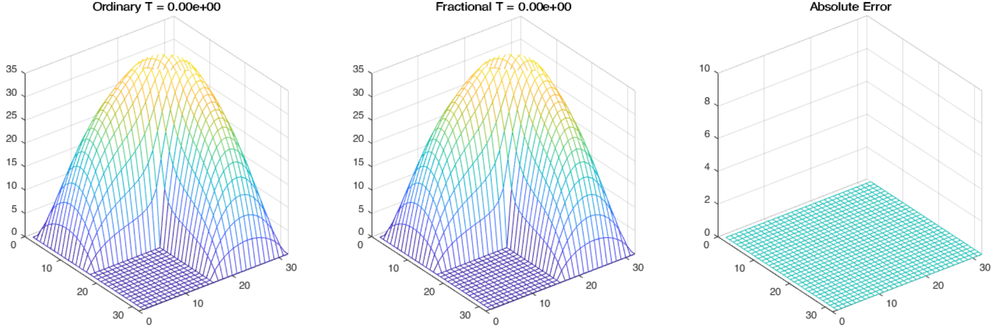
$$\begin{cases} {}_{CA}D_t^{\alpha} (u \cdot \nabla) u = \nu \nabla^2 u - \frac{1}{\rho} \nabla p + f, \\ \nabla \cdot u = 0. \end{cases} \quad 0 < \alpha \leq 1.$$

An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05 \nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

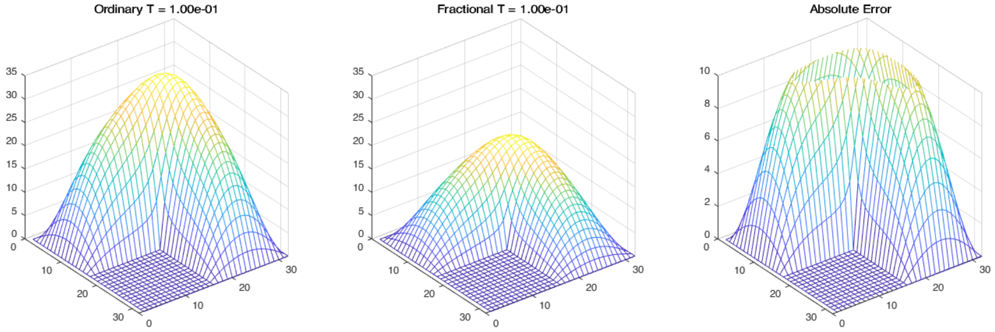


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05 \nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

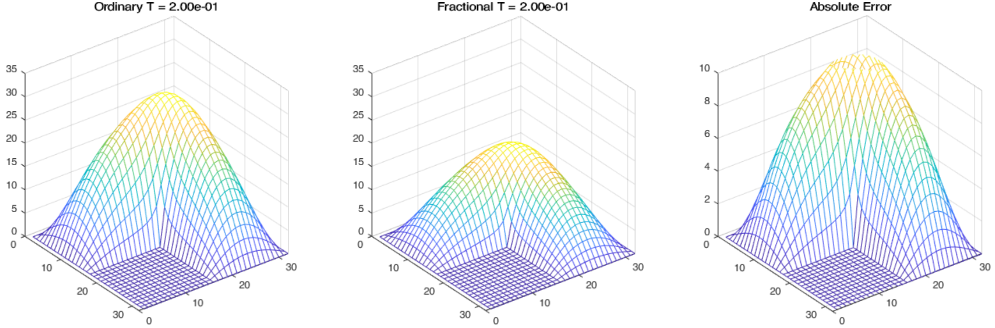


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05 \nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

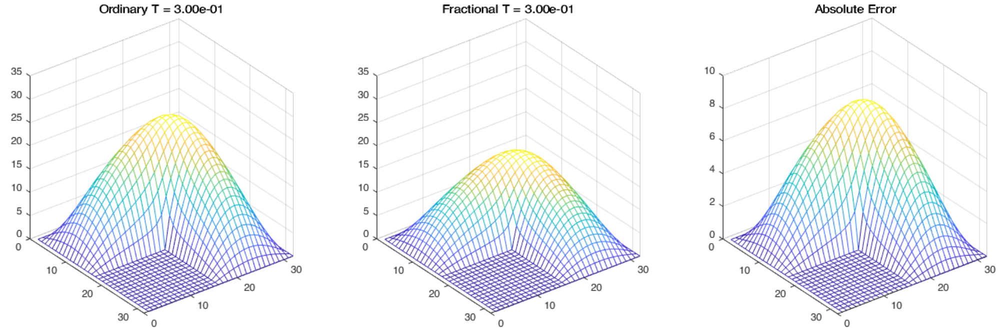


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05 \nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

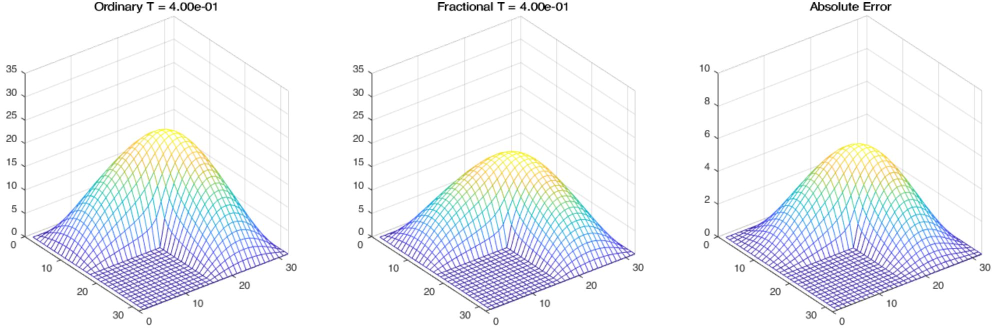


An example with diffusion

Let us consider the case of

$$_{CA}D_t^{\alpha} u = 0.05\nabla^2 u, \quad \alpha = 0.3, 1.$$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

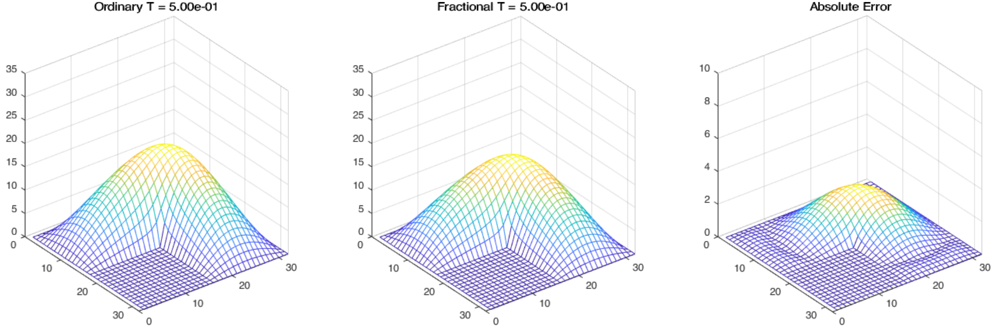


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05\nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

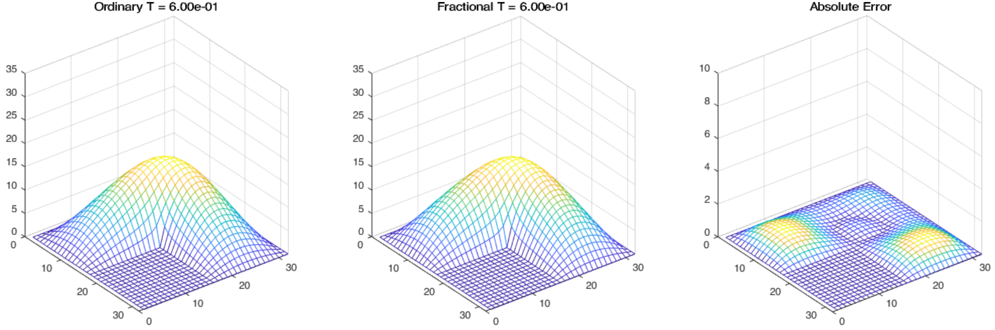


An example with diffusion

Let us consider the case of

$$_{CA}D_t^{\alpha} u = 0.05\nabla^2 u, \quad \alpha = 0.3, 1.$$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

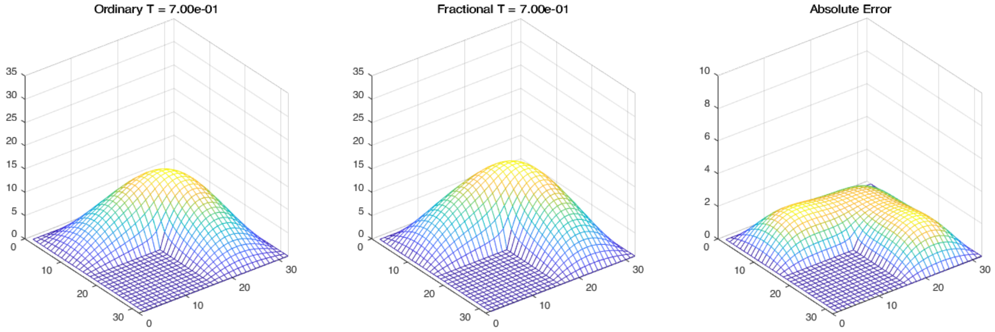


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05 \nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

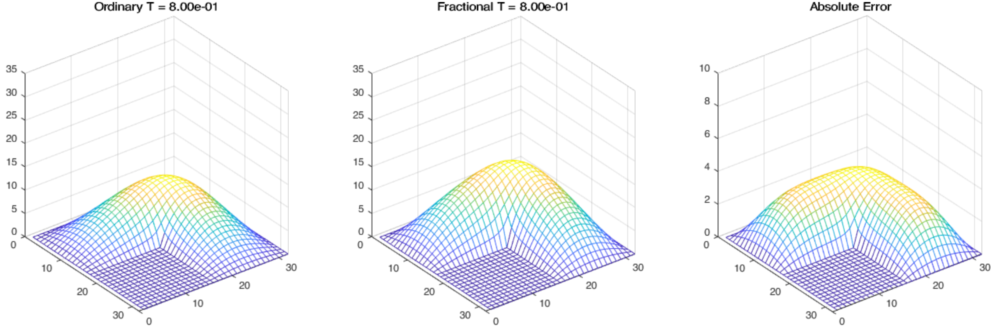


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05 \nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

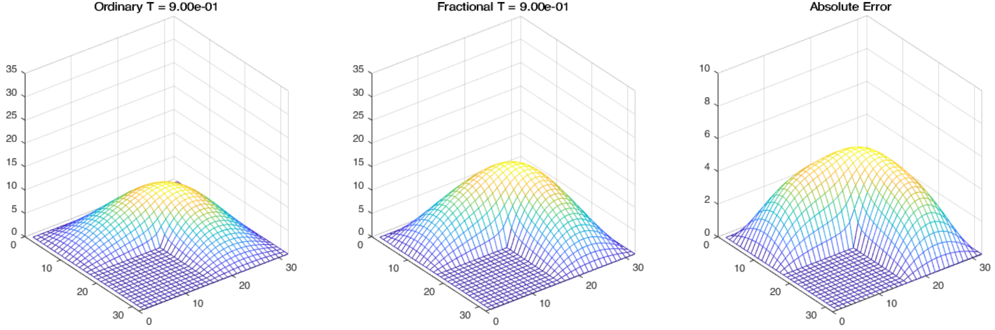


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05 \nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

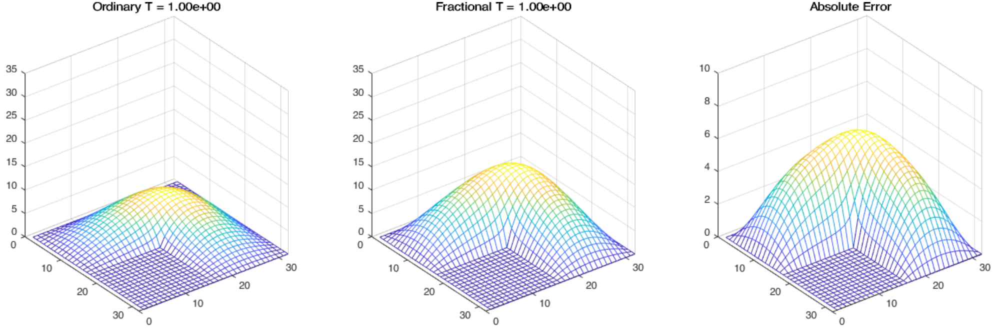


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05 \nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.

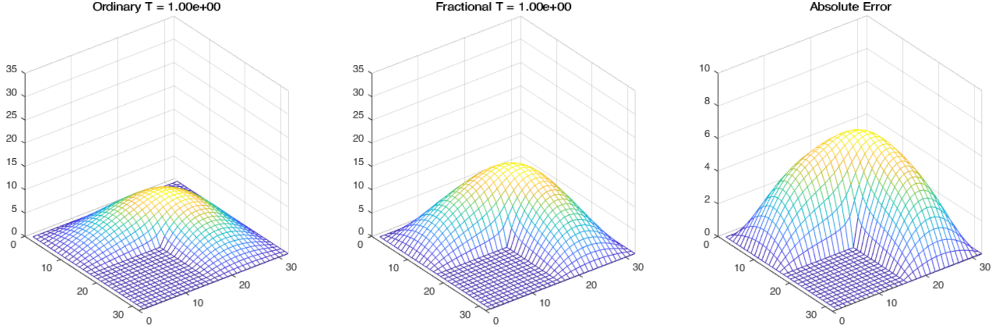


An example with diffusion

Let us consider the case of

$$ {}_{CA}D_t^{\alpha} u = 0.05\nabla^2 u, \quad \alpha = 0.3, 1. $$

and integrate it with the FBDF2 with $\tau = 10^{-2}$.



How can you describe the observed behavior?

Conclusions and next steps

- We have completed the construction of several schemes for the integration of FODEs,

Conclusions and next steps

- We have completed the construction of several schemes for the integration of FODEs,
- We have discussed the case of FODEs with multiple terms and different orders,

Conclusions and next steps

- We have completed the construction of several schemes for the integration of FODEs,
- We have discussed the case of FODEs with multiple terms and different orders,
- We started looking into some time-fractional PDEs using the Method of Lines together with our FODEs algorithms.

Conclusions and next steps

- We have completed the construction of several schemes for the integration of FODEs,
- We have discussed the case of FODEs with multiple terms and different orders,
- We started looking into some time-fractional PDEs using the Method of Lines together with our FODEs algorithms.
- Can we better describe this “subdiffusive” behavior we have observed in time-fractional diffusion equation?

Conclusions and next steps

- ✅ We have completed the construction of several schemes for the integration of FODEs,
- ✅ We have discussed the case of FODEs with multiple terms and different orders,
- ✅ We started looking into some time-fractional PDEs using the Method of Lines together with our FODEs algorithms.
- 📋 Can we better describe this "subdiffusive" behavior we have observed in time-fractional diffusion equation?
- 📋 For linear problems can we investigate the "exponential" fractional integrators?

Bibliography I

Bagley, R. L. and P. J. Torvik (1986). "On the Fractional Calculus Model of Viscoelastic Behavior". In: *Journal of Rheology* 30.1, pp. 133–155. DOI: 10.1122/1.549887.

Diethelm, K. (2003). "Efficient solution of multi-term fractional differential equations using $\mathrm{P(EC)}^m\mathrm{E}$ methods". In: *Computing* 71.4, pp. 305–319. ISSN: 0010-485X. DOI: 10.1007/s00607-003-0033-3. URL: https://doi.org/10.1007/s00607-003-0033-3.

Diethelm, K. (2010). *The analysis of fractional differential equations*. Vol. 2004. Lecture Notes in Mathematics. An application-oriented exposition using differential operators of Caputo type. Springer-Verlag, Berlin, pp. viii+247. ISBN: 978-3-642-14573-5. DOI: 10.1007/978-3-642-14574-2. URL: https://doi.org/10.1007/978-3-642-14574-2.

Ford, N. J. and J. A. Connolly (2009). "Systems-based decomposition schemes for the approximate solution of multi-term fractional differential equations". In: *J. Comput. Appl. Math.* 229.2, pp. 382–391. ISSN: 0377-0427. DOI: 10.1016/j.cam.2008.04.003. URL: https://doi.org/10.1016/j.cam.2008.04.003.

Bibliography II

- Ford, N. J. and A. C. Simpson (2001). “The numerical solution of fractional differential equations: speed versus accuracy”. In: *Numer. Algorithms* 26.4, pp. 333–346. ISSN: 1017-1398. DOI: 10.1023/A:1016601312158. URL: https://doi.org/10.1023/A:1016601312158.
- Garrappa, R. (2015). “Trapezoidal methods for fractional differential equations: theoretical and computational aspects”. In: *Math. Comput. Simulation* 110, pp. 96–112. ISSN: 0378-4754. DOI: 10.1016/j.matcom.2013.09.012. URL: https://doi.org/10.1016/j.matcom.2013.09.012.
- — (2018). “Numerical solution of fractional differential equations: A survey and a software tutorial”. In: *Mathematics* 6.2, p. 16.
- Hairer, E., C. Lubich, and M. Schlichte (1985). “Fast numerical solution of nonlinear Volterra convolution equations”. In: *SIAM J. Sci. Statist. Comput.* 6.3, pp. 532–541. ISSN: 0196-5204. DOI: 10.1137/0906037. URL: https://doi.org/10.1137/0906037.
- Henrici, P. (1974). *Applied and computational complex analysis*. Pure and Applied Mathematics. Volume 1: Power series—integration—conformal mapping—location of zeros. Wiley-Interscience [John Wiley & Sons], New York-London-Sydney, pp. xv+682.

Bibliography III

Wang, Y. and C. Li (2007). "Does the fractional Brusselator with efficient dimension less than 1 have a limit cycle?" In: *Physics Letters A* 363.5, pp. 414–419. ISSN: 0375-9601. DOI: https://doi.org/10.1016/j.physleta.2006.11.038. URL: https://www.sciencedirect.com/science/article/pii/S0375960106018020.