An introduction to fractional calculus

Fundamental ideas and numerics

Fabio Durastante

Università di Pisa

fabio.durastante@unipi.it

fdurastante.github.io

May, 2022

The Numerical Integration of FODEs

We want to find a **numerical solution** of the differential equation written in terms of Caputo Derivatives

$$\alpha > 0, \quad m = \lceil \alpha \rceil, \qquad \begin{cases} {}_{C}D^{\alpha}_{[0,t]}\mathbf{y}(t) = f(t, \mathbf{y}(t)), & t \in [0, T], \\ \dfrac{d^k \mathbf{y}(0)}{dt^k} = \mathbf{y}_0^{(k)}, & k = 0, 1, \ldots, m-1. \end{cases} \tag{FODE}$$

Caputo fractional derivative (Caputo 2008)

Let $\alpha \geq 0$, and $m = \lceil \alpha \rceil$. Then, we define the operator

$${}_{C}D^{\alpha}_{[a,t]} y = I^{m-\alpha}_{[a,t]} \frac{d^m}{dt^m} y,$$

whenever $\frac{d^m}{dt^m} y \in \mathbb{L}^1([a, b])$.

The Numerical Integration of ODEs

What methods do we know for ODEs?

The Numerical Integration of ODEs

What methods do we know for ODEs?

Given a grid $\{t_j = j\tau\}_{j=1}^{N}$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

The Numerical Integration of ODEs

What methods do we know for ODEs?

Given a grid $\{t_j = j\tau\}_{j=1}^{N}$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

Explicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k < j$

The Numerical Integration of ODEs

What methods do we know for ODEs?

Given a grid $\{t_j = j\tau\}_{j=1}^N$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

Explicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k < j$

Implicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k \leq j$

The Numerical Integration of ODEs

What methods do we know for ODEs?
Given a grid $\{t_j = j\tau\}_{j=1}^{N}$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

Explicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k < j$

Implicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k \leq j$

Exponential integrators: Compute directly $y^{(N)}$ without any $y^{(j)}$ for $j < N$.

The Numerical Integration of ODEs

What methods do we know for ODEs?

Given a grid $\{t_j = j\tau\}_{j=1}^{N}$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

Explicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k < j$

Implicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k \leq j$

Exponential integrators: Compute directly $y^{(N)}$ without any $y^{(j)}$ for $j < N$.

The Numerical Integration of ODEs

What methods do we know for ODEs?
Given a grid $\{t_j = j\tau\}_{j=1}^{N}$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

Explicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k < j$

- One Step Methods: Explicit Runge-Kutta Methods (ERK)
- Linear Multistep Methods: Adams–Bashforth, Predictor-Corrector

Implicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k \leq j$

Exponential integrators: Compute directly $y^{(N)}$ without any $y^{(j)}$ for $j < N$.

The Numerical Integration of ODEs

What methods do we know for ODEs?
Given a grid $\{t_j = j\tau\}_{j=1}^{N}$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

Explicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k < j$

- One Step Methods: Explicit Runge-Kutta Methods (ERK)
- Linear Multistep Methods: Adams–Bashforth, Predictor-Corrector

Implicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k \leq j$

- One Step Methods: Implicit Runge-Kutta Methods (IRK,DIRK,SDIRK)
- Linear Multistep Methods: Adams–Bashforth, Backward Differentiation Formulas (BDFs), Numerical Differentiation Formulas (NDFs),...

Exponential integrators: Compute directly $y^{(N)}$ without any $y^{(j)}$ for $j < N$.

The Numerical Integration of ODEs

What methods do we know for ODEs?

Given a grid $\{t_j = j\tau\}_{j=1}^{N}$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

Explicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k < j$

- One Step Methods: Explicit Runge-Kutta Methods (ERK)
- Linear Multistep Methods: Adams–Bashforth, Predictor-Corrector

Implicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k \leq j$

- One Step Methods: Implicit Runge-Kutta Methods (IRK,DIRK,SDIRK)
- Linear Multistep Methods: Adams–Bashforth, Backward Differentiation Formulas (BDFs), Numerical Differentiation Formulas (NDFs),...

Exponential integrators: Compute directly $y^{(N)}$ without any $y^{(j)}$ for $j < N$.

For both *implicit* and *explicit* methods we have also all-at-once formulations, and a middle-ground represented by Implicit-Explicit (IMEX) methods.

The Numerical Integration of ODEs

What methods do we know for ODEs?
Given a grid $\{t_j = j\tau\}_{j=1}^{N}$ and $\tau = T/N$, approximating $\mathbf{y}(t_j) \approx y^{(j)}$.

Explicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k < j$

- One Step Methods: Explicit Runge-Kutta Methods (ERK)
- Linear Multistep Methods: Adams–Bashforth, Predictor-Corrector

Implicit methods: Compute $y^{(j)}$ using only values $y^{(k)}$ for $k \leq j$

- One Step Methods: Implicit Runge-Kutta Methods (IRK,DIRK,SDIRK)
- Linear Multistep Methods: Adams–Bashforth, Backward Differentiation Formulas (BDFs), Numerical Differentiation Formulas (NDFs),...

Exponential integrators: Compute directly $y^{(N)}$ without any $y^{(j)}$ for $j < N$.

For both *implicit* and *explicit* methods we have also all-at-once formulations, and a middle-ground represented by Implicit-Explicit (IMEX) methods.
Our *objective* is to transport what we can for the solution of (FODE).

Product Integration Rules

Product **I**ntegration rules were introduced in the work (Young 1954) for Integral Equations.

Product Integration Rules

Product **I**ntegration rules were introduced in the work (Young 1954) for Integral Equations. From the *existence results* we know that a solution to (FODE) is a solution to the Integral Equation

$$y(t) = \sum_{k=0}^{m-1} \frac{t^k}{k!} y_0^{(k)} + \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(\tau, y(\tau))\,\mathrm{d}\tau, \quad m = \lceil \alpha \rceil.$$

Product Integration Rules

Product **I**ntegration rules were introduced in the work (Young 1954) for Integral Equations. From the *existence results* we know that a solution to (FODE) is a solution to the Integral Equation

$$y(t) = \sum_{k=0}^{m-1} \frac{t^k}{k!} y_0^{(k)} + \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(\tau, y(\tau))\, \mathrm{d}\tau, \quad m = \lceil \alpha \rceil.$$

- Adams-Bashforth-Moulton methods are obtained by applying a *quadrature formula to the integral*,

Product Integration Rules

Product **I**ntegration rules were introduced in the work (Young 1954) for Integral Equations. From the *existence results* we know that a solution to (FODE) is a solution to the Integral Equation

$$y(t) = \sum_{k=0}^{m-1} \frac{t^k}{k!} y_0^{(k)} + \frac{1}{\Gamma(\alpha)} \int_0^t (t-\tau)^{\alpha-1} f(\tau, y(\tau))\, \mathrm{d}\tau, \qquad m = \lceil \alpha \rceil.$$

- Adams-Bashforth-Moulton methods are obtained by applying a *quadrature formula to the integral*,
- We can use, e.g.,
  - the **fractional rectangular formula** with nodes $\{t_j = j\tau\}_{j=1}^{n-1}$,
  - or the **product trapezoidal quadrature formula** with nodes $\{t_j = j\tau\}_{j=1}^{n}$.

  To obtain a predictor-corrector method.

Product Integral Rules

The main idea behind PI rules is to approximate the integral

$$\int_0^t (t-\tau)^{\alpha-1} f(s, y(s))\,\mathrm{d}s$$

by approximating the vector field f with suitable polynomials..

Product Integral Rules

The main idea behind PI rules is to approximate the integral

$$\int_0^t (t-\tau)^{\alpha-1} f(s, y(s))\,\mathrm{d}s$$

by approximating the vector field f with suitable polynomials. We build an Adams-type method. If we consider a grid $\{t_0, t_1, \dots, t_N\}$ on the whole $[t_0, T]$ we can decompose the integral as

$$y(t) = T_{m-1}(t) + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} (t-s)^{\alpha-1} f(\tau, y(\tau))\,\mathrm{d}\tau, \qquad t \geq t_n.$$

Product Integral Rules

The main idea behind PI rules is to approximate the integral

$$\int_0^t (t-\tau)^{\alpha-1} f(s, y(s)) \,\mathrm{d}s$$

by approximating the vector field f with suitable polynomials. We build an Adams-type method. If we consider a grid $\{t_0, t_1, \dots, t_N\}$ on the whole $[t_0, T]$ we can decompose the integral as

$$y(t) = T_{m-1}(t) + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} (t-s)^{\alpha-1} f(\tau, y(\tau)) \,\mathrm{d}\tau, \qquad t \geq t_n.$$

- Replace f in each sub-interval by the first–degree polynomial interpolant

$$p_j(\tau) = f_{j+1} + \frac{s - t_{j+1}}{\tau_j}, \quad s \in [t_j, tj+1], \quad \tau_j = t_{j+1} - t_j, \quad f_j = f(t_j, y_j).$$

Product Integral Rules

The main idea behind PI rules is to approximate the integral

$$\int_0^t (t-\tau)^{\alpha-1} f(s, y(s))\,\mathrm{d}s$$

by approximating the vector field f with suitable polynomials. We build an Adams-type method. If we consider a grid $\{t_0, t_1, \ldots, t_N\}$ on the whole $[t_0, T]$ we can decompose the integral as

$$y(t) = T_{m-1}(t) + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n-1} \int_{t_j}^{t_{j+1}} (t-s)^{\alpha-1} f(\tau, y(\tau))\,\mathrm{d}\tau, \qquad t \geq t_n.$$

- Replace f in each sub-interval by the first–degree polynomial interpolant
- These produce the usual fractional integral that we now how to solve

$$I_{n,j}^{(k)} = \frac{1}{\Gamma(\alpha)} \int_{t_j}^{t_n} (t_n-\tau)^{\alpha-1} (\tau - t_j)^k\,\mathrm{d}\tau = \frac{(t_n - t_j)^{\alpha+k}}{\Gamma(\alpha+k+1)}.$$

Product Integral Rules

The main idea behind PI rules is to approximate the integral

$$\int_0^t (t-\tau)^{\alpha-1} f(s, y(s))\,\mathrm{d}s$$

by approximating the vector field f with suitable polynomials. We build an Adams-type method. If we consider a grid $\{t_0, t_1, \ldots, t_N\}$ on the whole $[t_0, T]$ we can decompose the integral as

$$y^{(n)} = T_{m-1}(t_n) + w_n f^{(0)} + \sum_{j=1}^{n} b_{n,j} f^{(j)},$$

- Replace f in each sub-interval by the first–degree polynomial interpolant
- These produce the usual fractional integral that we now how to solve
- We plug everything in our expression using that:

$$w_n = I_{n,0}^{(0)} - \frac{I_{n,0}^{(1)}}{\tau_0} + \frac{I_{n,1}^{(1)}}{\tau_0}, \quad b_{n_j} = \frac{I_{n,j-1}^{(1)} - I_{n,j}^{(1)}}{h_{j-1}} - \frac{I_{n,j}^{(1)} - I_{n,j+1}^{(1)}}{\tau_j}, j \leq n-1, \quad b_{n,n} = \frac{I_{n,n-1}^{(1)}}{\tau_{n-1}}.$$

Product Integral Rules - Convergence

To discuss **convergence properties** we can piggyback on the theory of Abel's and Volterra's fractional integral equations.

Product Integral Rules - Convergence

To discuss **convergence properties** we can piggyback on the theory of Abel's and Volterra's fractional integral equations.

$$\int_0^t (t-s)^{-\alpha} K(t,s) y(s)\,\mathrm{d}s = f(t), \quad 0 < \alpha < 1 \quad \text{(Volterra's Integral Eq.)}$$

Product Integral Rules - Convergence

To discuss **convergence properties** we can piggyback on the theory of Abel's and Volterra's fractional integral equations.

$$\int_0^t (t-s)^{-\alpha} y(s)\,\mathrm{d}s = f(t), \quad 0 < \alpha < 1 \qquad \text{(Abel's Integral Eq.)}$$

If we discretize everything as before we get

$$[B_N \odot K_N]\mathbf{y} = \mathbf{g}, \quad B_N = \tau^{1-\alpha}[b_{i,j}], \quad K_N = [k(t_i, t_j)], \quad \odot \text{ Hadamard product.}$$

where $\mathbf{y} = (y_0, \ldots, y_N)^T$ and g contains the **initial conditions** and the **evaluations** of f.

Convergence analysis for (Cameron and McKee 1985)

"[Consistency of order p] demands that $f(t) \in \mathcal{C}^{1-\alpha}[0, T]$ which is necessary in any case for $y(t)$ to be a smooth function $\ldots$ $|y(t_i) - y_i| \leq C\tau^p$, $i = 0, 1, \ldots, m-1$."

Product Integral Rules - Convergence

The requirements from the standard theory are **far too strong** for what we can reasonably expect from the analysis on the solution regularity we did in the last lecture.

Theorem (Dixon 1985)

Let f be Lipschitz continuous with respect to the second variable and y_n be the numerical approximation obtained by applying the PI trapezoidal rule on the interval $[t_0, T]$. There exist a constant $C = C_1(T - t_0)$, which does not depend on h, such that

$$\|y(t_n) - y_n\| \leq C(t_n^{\alpha-1}\tau^{1+\alpha} + \tau^2), \qquad \tau = \max_{j=0,\dots,n-1} \tau_j.$$

Product Integral Rules - Convergence

The requirements from the standard theory are **far too strong** for what we can reasonably expect from the analysis on the solution regularity we did in the last lecture.

Theorem (Dixon 1985)

Let f be Lipschitz continuous with respect to the second variable and y_n be the numerical approximation obtained by applying the PI trapezoidal rule on the interval $[t_0, T]$. There exist a constant $C = C_1(T - t_0)$, which does not depend on h, such that

$$\|y(t_n) - y_n\| \leq C(t_n^{\alpha-1}\tau^{1+\alpha} + \tau^2), \qquad \tau = \max_{j=0,\ldots,n-1} \tau_j.$$

- The same drop in the convergence order occurs also when higher degree polynomials are employed,

Product Integral Rules - Convergence

The requirements from the standard theory are **far too strong** for what we can reasonably expect from the analysis on the solution regularity we did in the last lecture.

Theorem (Dixon 1985)

Let f be Lipschitz continuous with respect to the second variable and y_n be the numerical approximation obtained by applying the PI trapezoidal rule on the interval $[t_0, T]$. There exist a constant $C = C_1(T - t_0)$, which does not depend on h, such that

$$\|y(t_n) - y_n\| \leq C(t_n^{\alpha-1}\tau^{1+\alpha} + \tau^2), \qquad \tau = \max_{j=0,\dots,n-1} \tau_j.$$

- The same drop in the convergence order occurs also when higher degree polynomials are employed,
- When $\alpha > 1$ convergence order 2 is obtained.

Product Integral Rules - Convergence

The requirements from the standard theory are **far too strong** for what we can reasonably expect from the analysis on the solution regularity we did in the last lecture.

Theorem (Dixon 1985)

Let f be Lipschitz continuous with respect to the second variable and y_n be the numerical approximation obtained by applying the PI trapezoidal rule on the interval $[t_0, T]$. There exist a constant $C = C_1(T - t_0)$, which does not depend on h, such that

$$\|y(t_n) - y_n\| \leq C(t_n^{\alpha-1}\tau^{1+\alpha} + \tau^2), \qquad \tau = \max_{j=0,\ldots,n-1} \tau_j.$$

- The same drop in the convergence order occurs also when higher degree polynomials are employed,
- When $\alpha > 1$ convergence order 2 is obtained.

It doesn't make much sense to use higher-degree PI rules if $0 < \alpha < 1$.

The Fractional Rectangular Formula

Let us reduce to the case with $\alpha \in (0,1)$, $m = 1$, and a *uniform mesh*.
To build it we need to *approximate the integral* with the rectangule rule

$$\int_0^t (t-\tau)^{\alpha-1} f(\tau, y(\tau))\, \mathrm{d}\tau$$

on the grid $\{t_j = t_0 + j\tau\}_{j=1}^N$ with *uniform* grid spacing τ, we denote

$$f^{(j)} = f(t_j, y^{(j)}) \text{ for } y^{(j)} \approx y(t_j),$$

and write it as

$$y^{(n)} = y_0 + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n-1} b_{n-j-1} f^{(j)}, \quad b_n = [(n+1)^\alpha - n^\alpha]/\alpha, \quad n = 1, \ldots, N.$$

The Fractional Rectangular Formula

Let us reduce to the case with $\alpha \in (0,1)$, $m = 1$, and a *uniform mesh*.
To build it we need to *approximate the integral* with the rectangule rule

$$\int_0^t (t-\tau)^{\alpha-1} f(\tau, y(\tau))\,\mathrm{d}\tau$$

on the grid $\{t_j = t_0 + j\tau\}_{j=1}^N$ with *uniform* grid spacing τ, we denote

$$f^{(j)} = f(t_j, y^{(j)}) \text{ for } y^{(j)} \approx y(t_j),$$

and write it as

$$y^{(n)} = y_0 + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n-1} b_{n-j-1} f^{(j)}, \quad b_n = [(n+1)^\alpha - n^\alpha]/\alpha, \quad n = 1, \ldots, N.$$

- This is an explicit method,

The Fractional Rectangular Formula

Let us reduce to the case with $\alpha \in (0,1)$, $m = 1$, and a *uniform mesh*.
To build it we need to *approximate the integral* with the rectangule rule

$$\int_0^t (t-\tau)^{\alpha-1} f(\tau, y(\tau))\,\mathrm{d}\tau$$

on the grid $\{t_j = t_0 + j\tau\}_{j=1}^N$ with *uniform* grid spacing τ, we denote

$$f^{(j)} = f(t_j, y^{(j)}) \text{ for } y^{(j)} \approx y(t_j),$$

and write it as

$$y^{(n)} = y_0 + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n-1} b_{n-j-1} f^{(j)}, \quad b_n = [(n+1)^\alpha - n^\alpha]/\alpha, \quad n = 1, \ldots, N.$$

- This is an explicit method,
- By construction, this is a 1-step method...

The Fractional Rectangular Formula

Let us reduce to the case with $\alpha \in (0,1)$, $m = 1$, and a *uniform mesh*.
To build it we need to *approximate the integral* with the rectangule rule

$$\int_0^t (t-\tau)^{\alpha-1} f(\tau, y(\tau))\,\mathrm{d}\tau$$

on the grid $\{t_j = t_0 + j\tau\}_{j=1}^N$ with *uniform* grid spacing τ, we denote

$$f^{(j)} = f(t_j, y^{(j)}) \text{ for } y^{(j)} \approx y(t_j),$$

and write it as

$$y^{(n)} = y_0 + \frac{\tau^\alpha}{\Gamma(\alpha)} \left(b_0 f^{(n-1)} + \sum_{j=0}^{n-2} b_{n-j-1} f^{(j)} \right) \quad b_n = [(n+1)^\alpha - n^\alpha]/\alpha, \quad n = 1, \ldots, N.$$

- This is an explicit method,
- By construction, this is a 1-step method... but in reality we need all the previous steps!

Some observations

- To build the solution we have to keep in memory either the **previous solutions** or the **function evaluations**,

Some observations

- To build the solution we have to keep in memory either the **previous solutions** or the **function evaluations**,
- Using a uniform mesh the evaluation of the weights just involve the computation of real powers of integer numbers, we can simplify also the fractional trapezoidal formula

$$
\begin{aligned}
y^{(n)} =& T_{m-1}(t_n) + \frac{\tau^\alpha}{\Gamma(\alpha+2)} \left(w_n f^{(0)} + \sum_{j=1}^{n} b_{n-j} f^{(j)} \right), \\
w_n =& (\alpha + 1 - n) n^\alpha + (n-1)^{\alpha+1}, \\
b_0 =& 1,\ b_n = (n-1)^{\alpha+1} - 2n^{\alpha+1} + (n+1)^{\alpha+1},\ n \geq 1.
\end{aligned}
$$

Some observations

- To build the solution we have to keep in memory either the **previous solutions** or the **function evaluations**,
- Using a uniform mesh the evaluation of the weights just involve the computation of real powers of integer numbers,
- We have to compute:
$$\sum_{j=0}^{n-2} b_{n-j-1} f^{(j)},$$
this is a quadratic cost with N, but there is a **convolution structure**, so we can expect that some FFT-based trick could come to the rescue.

Some observations

- To build the solution we have to keep in memory either the **previous solutions** or the **function evaluations**,
- Using a uniform mesh the evaluation of the weights just involve the computation of real powers of integer numbers,
- We have to compute:
$$\sum_{j=0}^{n-2} b_{n-j-1} f^{(j)},$$
this is a quadratic cost with N, but there is a **convolution structure**, so we can expect that some FFT-based trick could come to the rescue.
- The 1-step name is related to the number of initial values to start the computation.

Some observations

- To build the solution we have to keep in memory either the **previous solutions** or the **function evaluations**,
- Using a uniform mesh the evaluation of the weights just involve the computation of real powers of integer numbers,
- We have to compute:

$$\sum_{j=0}^{n-2} b_{n-j-1} f^{(j)},$$

  this is a quadratic cost with N, but there is a **convolution structure**, so we can expect that some FFT-based trick could come to the rescue.
- The 1-step name is related to the number of initial values to start the computation.

Predictor-Corrector algorithms

Now that we have two schemes we can think of using them together to build a **predictor-corrector** algorithm.

Fractional Predictor-Corrector Scheme (Diethelm 1997)

We are going to write it again for $0 < \alpha < 1$ on a uniform mesh

1. In the *prediction step* we use the fractional rectangular formula

$$y_P^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n} b_{j,n+1} f(t_j, y^{(j)}), \quad b_{j,n+1} = \frac{(n+1-j)^\alpha - (n-j)^\alpha}{\alpha}$$

Fractional Predictor-Corrector Scheme (Diethelm 1997)

We are going to write it again for $0 < \alpha < 1$ on a uniform mesh

1. In the *prediction step* we use the fractional rectangular formula

$$y_P^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n} b_{j,n+1} f(t_j, y^{(j)}), \quad b_{j,n+1} = \frac{(n+1-j)^\alpha - (n-j)^\alpha}{\alpha}$$

2. In the *correction step* we use the fractional trapezoidal formula

$$y^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \left(\sum_{j=0}^{n} a_{j,n+1} f(t_j, y^{(j)}) + a_{n+1,n+1} f(t_{n+1}, y_P^{(n+1)}) \right)$$

   where

$$a_{j,n+1} = \begin{cases} {(n^{\alpha+1} - (n-\alpha)(n+1)^\alpha}/{\alpha(\alpha+1)}, & j = 0, \\ {(n-j+2)^{\alpha+1} - 2(n-j+1)^{\alpha+1} + (n-j)^{\alpha+1}}/{\alpha(\alpha+1)}, & j = 1, 2, \ldots, n, \\ {1}/{\alpha(\alpha+1)}, & j = n+1. \end{cases}$$

A Fractional Predictor-Corrector Scheme

- Predictor-Corrector schemes are of interest because they represent a good **compromise** between **accuracy** and **ease of implementation**.

A Fractional Predictor-Corrector Scheme

- Predictor-Corrector schemes are of interest because they represent a good **compromise** between **accuracy** and **ease of implementation**.
- To investigate the convergence we need to look deeper into the convergence results of the two PI integral rules (Diethelm, Ford, and Freed 2004).

Theorem (Diethelm, Ford, and Freed 2004, Theorem 2.4)

(a) Let $z \in \mathcal{C}^1([0, T])$. Then

$$\left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\alpha-1} z(t) \, \mathrm{d}t - \sum_{j=0}^{k} b_{j,k+1} z(t_j) \right| \leq \frac{1}{\alpha} \|z'\|_\infty t_{k+1}^{\alpha} \tau.$$

(b) Let $z(t) = t^p$ for some $p \in (0, 1)$. Then,

$$\left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\alpha-1} z(t) \, \mathrm{d}t - \sum_{j=0}^{k} b_{j,k+1} z(t_j) \right| \leq C_{\alpha,p}^{Re} t_{k+1}^{\alpha+p-1} \tau.$$

A Fractional Predictor-Corrector Scheme

And analogously for the product trapezoidal formula.

Theorem (Diethelm, Ford, and Freed 2004, Theorem 2.5).

(a) If $z \in \mathcal{C}^2([0, T])$, then there exist a constant C_α^{Tr} depending only on α such that

$$\left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\alpha - 1} z(t) \,\mathrm{d}t - \sum_{j=0}^{k+1} a_{j,k+1} z(t_j) \right| \leq C_\alpha^{Tr} \|z''\|_\infty t_{k+1}^\alpha \tau^2.$$

(b) Let $z \in \mathcal{C}^1([0, T])$ and assume that z' fulfills a Lipschitz condition of order $\mu \in (0, 1)$. Then, there exists positive constants $B_{\alpha,\mu}^{Tr}$ and $M_{z,\mu}$ such that

$$\left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\alpha - 1} z(t) \,\mathrm{d}t - \sum_{j=0}^{k+1} a_{j,k+1} z(t_j) \right| \leq B_{\alpha,\mu}^{Tr} M_{z,\mu} t_{k+1}^\alpha \tau^{1+\mu}.$$

A Fractional Predictor-Corrector Scheme

And analogously for the product trapezoidal formula.

Theorem (Diethelm, Ford, and Freed 2004, Theorem 2.5).

(a) Let $z \in \mathcal{C}^1([0, T])$ and assume that z' fulfills a Lipschitz condition of order $\mu \in (0, 1)$. Then, there exists positive constants $B_{\alpha,\mu}^{Tr}$ and $M_{z,\mu}$ such that

$$\left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\alpha-1} z(t) \, \mathrm{d}t - \sum_{j=0}^{k+1} a_{j,k+1} z(t_j) \right| \leq B_{\alpha,\mu}^{Tr} M_{z,\mu} t_{k+1}^{\alpha} \tau^{1+\mu}.$$

(b) Let $z(t) = t^p$ for some $p \in (0, 2)$ and $\rho = \min(2, p + 1)$. Then

$$\left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\alpha-1} z(t) \, \mathrm{d}t - \sum_{j=0}^{k+1} a_{j,k+1} z(t_j) \right| \leq C_{\alpha,p}^{Tr} t_{k+1}^{\alpha+p-\rho} \tau^{\rho}.$$

A Fractional Predictor-Corrector Scheme

Observe that for the fractional rectangular case (b) the bound contains

$$t_{k+1}^{\alpha+p-1},$$

if $\alpha + p < 1$ then we get that the overall integration error becomes larger if the size of the interval of integration becomes smaller!
Similarly for the case (c) for the fractional trapezoidal rule

$$\alpha < 1, \; p < 1, \; \rho = p + 1, \quad t_{k+1}^{\alpha+p-\rho},$$

has the same explosive behavior.

Smaller intervals for harder integrals

By making t_{k+1} smaller we have two effects

1. We reduce the length of the integration interval,
2. We change the weight function in a way that makes the integral more difficult.

A Fractional Predictor-Corrector Scheme

Lemma (Diethelm, Ford, and Freed 2004, Lemma 3.1)

Assume that the solution y of the initial value problem is such that

$$\left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\alpha-1} {}_{CA}D^{\alpha}_{[0,t]} y(t)\,\mathrm{d}t - \sum_{j=0}^{k} b_{j,k+1}\, {}_{CA}D^{\alpha}_{[0,t]} y(t) \right| \leq C_1 t_{k+1}^{\gamma_1} \tau^{\delta_1},$$

and

$$\left| \int_0^{t_{k+1}} (t_{k+1} - t)^{\alpha-1} {}_{CA}D^{\alpha}_{[0,t]} y(t)\,\mathrm{d}t - \sum_{j=0}^{k+1} a_{j,k+1}\, {}_{CA}D^{\alpha}_{[0,t]} y(t) \right| \leq C_2 t_{k+1}^{\gamma_2} \tau^{\delta_2},$$

with some $\gamma_1, \gamma_2 \geq 0$ and $\delta_1, \delta_2 > 0$. Then, for some suitably chosen $T > 0$, we have

$$\max_{0 \leq j \leq N} |y(t_j) - y^{(j)}| = O(\tau^q), \quad q = \min\{\delta_1 + \alpha, \delta_2\}, \quad N = \lceil T/\tau \rceil.$$

Error bounds

Theorem (Diethelm, Ford, and Freed 2004, Theorem 3.2)

Let $0 < \alpha$ and assume ${}_{CA}D^{\alpha}_{[0,t]}y(t) \in \mathcal{C}^2([O, T])$ for some suitable T. Then,

$$\max_{0 \leq j \leq N} |y(t_j) - y^{(j)}| = \begin{cases} O(\tau^2), & \text{if } \alpha \geq 1, \\ O(\tau^{1+\alpha}), & \text{if } \alpha < 1. \end{cases}$$

Proof. In view of the two bounds for the Fractional Rectangular and Trapezoidal forms we can apply the previous Lemma with $\gamma_1 = \gamma_2 = \alpha > 0$, $\delta_1 = 1$, $\delta_2 = 2$. Therefore we find a bound of order $O(\tau^q)$ where

$$q = \min\{1+\alpha, 2\} = \begin{cases} 2, & \text{if } \alpha \geq 1, \\ 1+\alpha, & \text{if } \alpha < 1. \end{cases}$$

Error bounds

Theorem (Diethelm, Ford, and Freed 2004, Theorem 3.2)

Let $0 < \alpha$ and assume ${}_{CA}D^{\alpha}_{[0,t]}y(t) \in \mathcal{C}^2([O, T])$ for some suitable T. Then,

$$\max_{0 \le j \le N} |y(t_j) - y^{(j)}| = \begin{cases} O(\tau^2), & \text{if } \alpha \ge 1, \\ O(\tau^{1+\alpha}), & \text{if } \alpha < 1. \end{cases}$$

- Order of convergence is a non-decreasing function of α,
- Hypotheses are stated in terms of the αth Caputo derivative of the solution,

Error bounds

Theorem (Diethelm, Ford, and Freed 2004, Theorem 3.2)

Let $0 < \alpha$ and assume ${}_{CA}D^{\alpha}_{[0,t]}y(t) \in \mathcal{C}^2([O, T])$ for some suitable T. Then,

$$\max_{0 \le j \le N} |y(t_j) - y^{(j)}| = \begin{cases} O(\tau^2), & \text{if } \alpha \ge 1, \\ O(\tau^{1+\alpha}), & \text{if } \alpha < 1. \end{cases}$$

- Order of convergence is a non-decreasing function of α,
- Hypotheses are stated in terms of the αth Caputo derivative of the solution,
- Can we replace them by similar assumptions on y itself?

Theorem Diethelm, Ford, and Freed 2004, Theorem 3.3

Let $\alpha > 1$ and assume $y \in \mathcal{C}^{1+\lceil \alpha \rceil}([0, T])$ for some suitable T, then

$$\max_{0 \le j \le N} |y(t_j) - y^{(j)}| = O(\tau^{1+\lceil \alpha \rceil - \alpha}).$$

Error bounds

Theorem (Diethelm, Ford, and Freed 2004, Theorem 3.2)

Let $0 < \alpha$ and assume ${}_{CA}D^{\alpha}_{[0,t]}y(t) \in \mathcal{C}^2([O, T])$ for some suitable T. Then,

$$\max_{0\leq j\leq N} |y(t_j) - y^{(j)}| = \begin{cases} O(\tau^2), & \text{if } \alpha \geq 1, \\ O(\tau^{1+\alpha}), & \text{if } \alpha < 1. \end{cases}$$

Theorem Diethelm, Ford, and Freed 2004, Theorem 3.3

Let $\alpha > 1$ and assume $y \in \mathcal{C}^{1+\lceil\alpha\rceil}([0, T])$ for some suitable T, then

$$\max_{0\leq j\leq N} |y(t_j) - y^{(j)}| = O(\tau^{1+\lceil\alpha\rceil-\alpha}).$$

Proof. We need to use the characterization of Caputo's derivative

$${}_{CA}D^{\alpha}_{[0,t]}y(t) = \sum_{\ell=0}^{m-\lceil\alpha\rceil-1} \frac{y^{(l+\lceil\alpha\rceil)}(0)}{\Gamma(\lceil\alpha\rceil-\alpha+\ell+1)} t^{\lceil\alpha\rceil-\alpha+\ell} + g(t), \qquad \begin{array}{l} g \in \mathcal{C}^{m-\lceil\alpha\rceil}([O, T]), \\ g^{(m-\lceil\alpha\rceil)} \in \mathrm{Lip}(\lceil\alpha\rceil-\alpha). \end{array}$$

Error bounds

Theorem (Diethelm, Ford, and Freed 2004, Theorem 3.2)

Let $0 < \alpha$ and assume ${}_{CA}D^{\alpha}_{[0,t]}y(t) \in \mathcal{C}^2([O, T])$ for some suitable T. Then,

$$\max_{0 \le j \le N} |y(t_j) - y^{(j)}| = \begin{cases} O(\tau^2), & \text{if } \alpha \ge 1, \\ O(\tau^{1+\alpha}), & \text{if } \alpha < 1. \end{cases}$$

Theorem Diethelm, Ford, and Freed 2004, Theorem 3.3

Let $\alpha > 1$ and assume $y \in \mathcal{C}^{1+\lceil \alpha \rceil}([0, T])$ for some suitable T, then

$$\max_{0 \le j \le N} |y(t_j) - y^{(j)}| = O(\tau^{1+\lceil \alpha \rceil - \alpha}).$$

Proof. Then for $\alpha > 1$, we can apply the Lemma with $\gamma_1 = 0$, $\gamma_2 = \alpha - 1 > 0$, $\delta_1 = 1$, $\delta_2 = 1 + \lceil \alpha \rceil - \alpha$ and thus $\delta_1 + \alpha = 1 + \alpha > 2 > \delta_2$, $\min\{\delta_1 + \alpha, \delta_2\} = \delta_2$. The overall order is then $O(\tau^{\delta_2}) = O(\tau^{1+\lceil \alpha \rceil - \alpha})$.

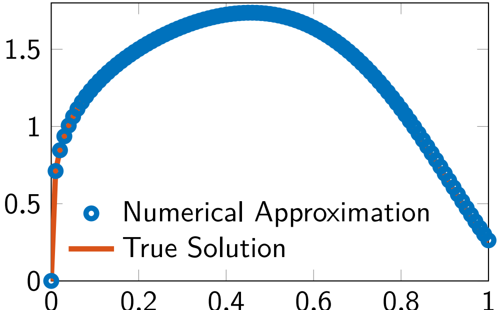
An example

Example

$$\begin{cases} {}_{CA}D^{\alpha}_{[0,t]}y(t) = \frac{40320}{\Gamma(9-\alpha)}t^{8-\alpha} - 3\frac{\Gamma(5+\alpha/2)}{\Gamma(5-\alpha/2)}t^{4-\alpha/2} + \frac{9}{4}\Gamma(\alpha+1) + \left({}^{3t^{\alpha/2}}/_{2} - t^4\right)^3 - y(t)^{3/2}, \\ y(0) = 0. \end{cases}$$

Solution: $y(t) = t^8 - 3t^{4+\alpha/2} + \frac{9}{4}t^{\alpha}$.

```
tauval = 2.^(-(1:6));
for i=1:length(hval)
 tau   = tauval(i);
 t0 = 0; T = 1;
 alpha    = 0.25;
 [T, Y] = fde_pi1_ex(alpha, f_fun, t0,
 ↪  T, y0, tau);
 err(i) = norm(Y - ye(T),'inf');
end
```



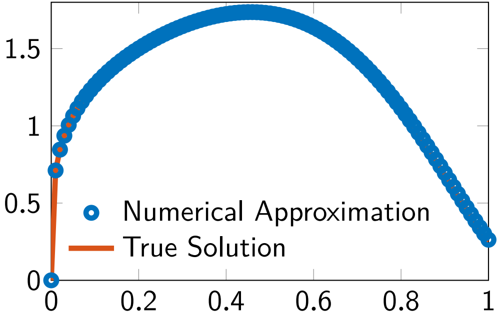
An example

Example

$$\begin{cases} {}_{CA}D^{\alpha}_{[0,t]}y(t) = \frac{40320}{\Gamma(9-\alpha)}t^{8-\alpha} - 3\frac{\Gamma(5+\alpha/2)}{\Gamma(5-\alpha/2)}t^{4-\alpha/2} + \frac{9}{4}\Gamma(\alpha+1) + \left(3t^{\alpha/2}/2 - t^4\right)^3 - y(t)^{3/2}, \\ y(0) = 0. \end{cases}$$

Solution: $y(t) = t^8 - 3t^{4+\alpha/2} + \frac{9}{4}t^{\alpha}$.

α	τ	E	q
0.25	5.00e-01	1.42e+00	
	2.50e-01	4.17e-01	1.77
	1.25e-01	2.13e-01	0.97
	6.25e-02	1.03e-01	1.05
	3.12e-02	5.04e-02	1.03
	1.56e-02	2.44e-02	1.05



An example

Example

$$\begin{cases} {}_{CA}D^{\alpha}_{[0,t]} y(t) = \frac{40320}{\Gamma(9-\alpha)} t^{8-\alpha} - 3\frac{\Gamma(5+\alpha/2)}{\Gamma(5-\alpha/2)} t^{4-\alpha/2} + \frac{9}{4}\Gamma(\alpha+1) + \left(3t^{\alpha/2}/2 - t^4\right)^3 - y(t)^{3/2}, \\ y(0) = 0. \end{cases}$$

Solution: $y(t) = t^8 - 3t^{4+\alpha/2} + \frac{9}{4}t^{\alpha}$.

α	τ	E	q
0.25	5.00e-01	1.42e+00	
	2.50e-01	4.17e-01	1.77
	1.25e-01	2.13e-01	0.97
	6.25e-02	1.03e-01	1.05
	3.12e-02	5.04e-02	1.03
	1.56e-02	2.44e-02	1.05

```
hval = 2.^(-(1:6));
for i=1:length(hval)
 h = hval(i);
 t0 = 0; T = 1;
 [T, Y] = fde_pi12_pc(alpha, f_fun,
 ↪  t0, T, y0, h, [], 1);
 err(i) = norm(Y - ye(T),'inf');
end
```

An example

Example

$$\begin{cases} {}_{CA}D^{\alpha}_{[0,t]}y(t) = \frac{40320}{\Gamma(9-\alpha)}t^{8-\alpha} - 3\frac{\Gamma(5+\alpha/2)}{\Gamma(5-\alpha/2)}t^{4-\alpha/2} + \frac{9}{4}\Gamma(\alpha+1) + \left({}^{3t^{\alpha/2}}/{}_{2} - t^4\right)^3 - y(t)^{3/2}, \\ y(0) = 0. \end{cases}$$

Solution: $y(t) = t^8 - 3t^{4+\alpha/2} + \frac{9}{4}t^{\alpha}$.

α	τ	E	q
0.25	5.00e-01	1.42e+00	
	2.50e-01	4.17e-01	1.77
	1.25e-01	2.13e-01	0.97
	6.25e-02	1.03e-01	1.05
	3.12e-02	5.04e-02	1.03
	1.56e-02	2.44e-02	1.05

α	τ	E	q
0.25	5.00e-01	2.75e+00	
	2.50e-01	1.80e+00	0.61
	1.25e-01	8.37e-01	1.10
	6.25e-02	2.45e-01	1.77
	3.12e-02	6.57e-02	1.90
	1.56e-02	2.02e-02	1.70

An example

Example

$$\begin{cases} {}_{CA}D^{\alpha}_{[0,t]}y(t) = \frac{40320}{\Gamma(9-\alpha)}t^{8-\alpha} - 3\frac{\Gamma(5+\alpha/2)}{\Gamma(5-\alpha/2)}t^{4-\alpha/2} + \frac{9}{4}\Gamma(\alpha+1) + \left(\frac{3}{2}t^{\alpha/2} - t^4\right)^3 - y(t)^{3/2}, \\ y(0) = 0. \end{cases}$$

Solution: $y(t) = t^8 - 3t^{4+\alpha/2} + \frac{9}{4}t^{\alpha}$.

α	τ	E	q
0.25	5.00e-01	1.42e+00	
	2.50e-01	4.17e-01	1.77
	1.25e-01	2.13e-01	0.97
	6.25e-02	1.03e-01	1.05
	3.12e-02	5.04e-02	1.03
	1.56e-02	2.44e-02	1.05

α	τ	E	q
	1.95e-03	9.33e-04	1.42
	9.77e-04	3.58e-04	1.38
	4.88e-04	1.40e-04	1.35
	2.44e-04	5.56e-05	1.33
	1.22e-04	2.23e-05	1.32
	6.10e-05	9.00e-06	1.31

More than one correction step

One can think of improving convergence by performing **more than one correction step** in the algorithm (Diethelm, Ford, and Freed 2002).
Let us call $\mu \in \mathbb{N}$ the number of correction steps:

$$\begin{cases} y_{[0]}^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n} b_{j,n+1} f(t_j, y^{(j)}), & \text{Prediction step,} \\ y_{[\ell]}^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \left(\sum_{j=0}^{n} a_{j,n+1} f(t_j, y^{(j)}) + a_{n+1,n+1} f(t_{n+1}, y_{[\ell-1]}^{(n+1)}) \right), & \text{Correction steps} \\ & \ell = 1, \ldots, \mu. \\ y^{(n+1)} \equiv y_{[\mu]}^{(n+1)}. & \end{cases}$$

More than one correction step

One can think of improving convergence by performing **more than one correction step** in the algorithm (Diethelm, Ford, and Freed 2002).
Let us call $\mu \in \mathbb{N}$ the number of correction steps:

$$
\begin{cases}
y_{[0]}^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n} b_{j,n+1} f(t_j, y^{(j)}), & \text{Prediction step,} \\
y_{[\ell]}^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \left(\sum_{j=0}^{n} a_{j,n+1} f(t_j, y^{(j)}) + a_{n+1,n+1} f(t_{n+1}, y_{[\ell-1]}^{(n+1)}) \right), & \text{Correction steps} \\
 & \ell = 1, \ldots, \mu. \\
y^{(n+1)} \equiv y_{[\mu]}^{(n+1)}. &
\end{cases}
$$

Each iteration is expected to increase the order of convergence of a fraction α from order 1 ($\mu = 0$) representing the fractional rectangular rule,

More than one correction step

One can think of improving convergence by performing **more than one correction step** in the algorithm (Diethelm, Ford, and Freed 2002).
Let us call $\mu \in \mathbb{N}$ the number of correction steps:

$$\begin{cases} y_{[0]}^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n} b_{j,n+1} f(t_j, y^{(j)}), & \text{Prediction step,} \\ y_{[\ell]}^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \left(\sum_{j=0}^{n} a_{j,n+1} f(t_j, y^{(j)}) + a_{n+1,n+1} f(t_{n+1}, y_{[\ell-1]}^{(n+1)}) \right), & \text{Correction steps} \\ & \ell = 1, \ldots, \mu. \\ y^{(n+1)} \equiv y_{[\mu]}^{(n+1)}. \end{cases}$$

- Each iteration is expected to increase the order of convergence of a fraction α from order 1 ($\mu = 0$) representing the fractional rectangular rule,
- The standard predictor corrector method is obtained for $\mu = 1$.

Convergence behavior

The convergence behavior can be described by using repeatedly the result from (Diethelm, Ford, and Freed 2004, Lemma 3.1) that we have used to obtain the other convergence bounds.

Corollary

$$\max_{0\le n\le N} |y(t_n) - y^{(n)}| = \begin{cases} O(\tau^{\min(1+\mu\alpha,2)}), & \text{if } {}_{CA}D^{\alpha}_{[t_0,t]}y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{\min(1+\mu\alpha,2-\alpha)}), & \text{if } y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{1+\alpha}), & \text{if } f(t,y) \in \mathcal{C}^2([O,T]\times\mathbb{D}). \end{cases}$$

Convergence behavior

The convergence behavior can be described by using repeatedly the result from (Diethelm, Ford, and Freed 2004, Lemma 3.1) that we have used to obtain the other convergence bounds.

Corollary

$$\max_{0\leq n\leq N} |y(t_n) - y^{(n)}| = \begin{cases} O(\tau^{\min(1+\mu\alpha,2)}), & \text{if } {}_{CA}D^{\alpha}_{[t_0,t]} y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{\min(1+\mu\alpha,2-\alpha)}), & \text{if } y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{1+\alpha}), & \text{if } f(t,y) \in \mathcal{C}^2([O,T]\times\mathbb{D}). \end{cases}$$

- The maximum order of convergence for ${}_{CA}D^{\alpha}_{[t_0,t]} y(t) \in \mathcal{C}^2([0,T])$ is obtained for $\mu = \lceil 1/\alpha \rceil$,

Convergence behavior

The convergence behavior can be described by using repeatedly the result from (Diethelm, Ford, and Freed 2004, Lemma 3.1) that we have used to obtain the other convergence bounds.

Corollary

$$\max_{0 \leq n \leq N} |y(t_n) - y^{(n)}| = \begin{cases} O(\tau^{\min(1+\mu\alpha,2)}), & \text{if } {}_{CA}D^{\alpha}_{[t_0,t]} y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{\min(1+\mu\alpha,2-\alpha)}), & \text{if } y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{1+\alpha}), & \text{if } f(t,y) \in \mathcal{C}^2([O,T] \times \mathbb{D}). \end{cases}$$

- The maximum order of convergence for ${}_{CA}D^{\alpha}_{[t_0,t]} y(t) \in \mathcal{C}^2([0,T])$ is obtained for $\mu = \lceil 1/\alpha \rceil$,
- The maximum order of convergence for $y(t) \in \mathcal{C}^2([0,T])$ is obtained for $\mu = \lceil 1-\alpha/\alpha \rceil$,

Convergence behavior

The convergence behavior can be described by using repeatedly the result from (Diethelm, Ford, and Freed 2004, Lemma 3.1) that we have used to obtain the other convergence bounds.

Corollary

$$\max_{0\le n\le N} |y(t_n) - y^{(n)}| = \begin{cases} O(\tau^{\min(1+\mu\alpha,2)}), & \text{if } {}_{CA}D^{\alpha}_{[t_0,t]} y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{\min(1+\mu\alpha,2-\alpha)}), & \text{if } y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{1+\alpha}), & \text{if } f(t,y) \in \mathcal{C}^2([O,T]\times\mathbb{D}). \end{cases}$$

- The maximum order of convergence for ${}_{CA}D^{\alpha}_{[t_0,t]} y(t) \in \mathcal{C}^2([0,T])$ is obtained for $\mu = \lceil 1/\alpha \rceil$,
- The maximum order of convergence for $y(t) \in \mathcal{C}^2([0,T])$ is obtained for $\mu = \lceil 1-\alpha/\alpha \rceil$,
- In the third case with a single corrector step, and no improvement is possible.

Convergence behavior

The convergence behavior can be described by using repeatedly the result from (Diethelm, Ford, and Freed 2004, Lemma 3.1) that we have used to obtain the other convergence bounds.

Corollary

$$\max_{0\leq n\leq N} |y(t_n) - y^{(n)}| = \begin{cases} O(\tau^{\min(1+\mu\alpha,2)}), & \text{if } {}_{CA}D^{\alpha}_{[t_0,t]}y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{\min(1+\mu\alpha,2-\alpha)}), & \text{if } y(t) \in \mathcal{C}^2([0,T]), \\ O(\tau^{1+\alpha}), & \text{if } f(t,y) \in \mathcal{C}^2([O,T]\times\mathbb{D}). \end{cases}$$

- The maximum order of convergence for ${}_{CA}D^{\alpha}_{[t_0,t]}y(t) \in \mathcal{C}^2([0,T])$ is obtained for $\mu = \lceil 1/\alpha \rceil$,
- The maximum order of convergence for $y(t) \in \mathcal{C}^2([0,T])$ is obtained for $\mu = \lceil 1-\alpha/\alpha \rceil$,
- In the third case with a single corrector step, and no improvement is possible.
- In general we could fix a maximum number of steps μ and halt the procedure when the error is under a certain tolerance.

Absolute stability

Let us focus on the **test problem**

$$ {}_{CA}D^{\alpha}_{[t_0,t]}y(t) = \lambda y(t), \quad y(0) = y_0, \quad \lambda \in \mathbb{C}, \quad 0 < \alpha < 1. $$

In the last lecture we have seen that the solution of this problem can be expressed as

$$ y(t) = E_\alpha(\lambda(t - t_0)^\alpha)y_0. $$

Absolute stability

Let us focus on the **test problem**

$$ {}_{CA}D^{\alpha}_{[t_0,t]}y(t) = \lambda y(t), \quad y(0) = y_0, \quad \lambda \in \mathbb{C}, \quad 0 < \alpha < 1. $$

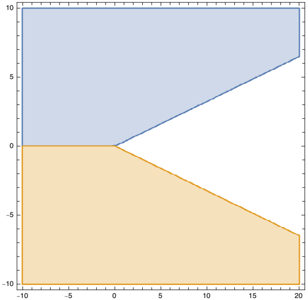
In the last lecture we have seen that the solution of this problem can be expressed as

$$ y(t) = E_{\alpha}(\lambda(t-t_0)^{\alpha})y_0. $$

Asymptotic behavior

The solution $y(t)$ asymptotically vanishes as $t \to +\infty$ for

$$ \lambda \in S^{*} = \{z \in \mathbb{C} : |\arg(z) - \pi| < (1 - \alpha/2)\pi.\} $$



Absolute stability

Let us focus on the **test problem**

$$ {}_{CA}D^{\alpha}_{[t_0,t]} y(t) = \lambda y(t), \quad y(0) = y_0, \quad \lambda \in \mathbb{C}, \quad 0 < \alpha < 1. $$

In the last lecture we have seen that the solution of this problem can be expressed as

$$ y(t) = E_{\alpha}(\lambda (t - t_0)^{\alpha}) y_0. $$

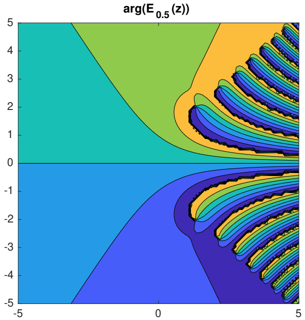
Asymptotic behavior

The solution $y(t)$ asymptotically vanishes as $t \to +\infty$ for

$$ \lambda \in S^* = \{ z \in \mathbb{C} \, : \, |\arg(z) - \pi| < (1 - \alpha/2)\pi. \} $$

The application of PI rule leads to a non-homogeneous difference equation

$$ y^{(n)} = g^{(n)} + \sum_{j=k}^{n} c_{n-j} y^{(j)}, \quad n \geq k, $$



Absolute stability

Let us focus on the **test problem**

$$ {}_{CA}D^{\alpha}_{[t_0,t]}y(t) = \lambda y(t), \quad y(0) = y_0, \quad \lambda \in \mathbb{C}, \quad 0 < \alpha < 1. $$

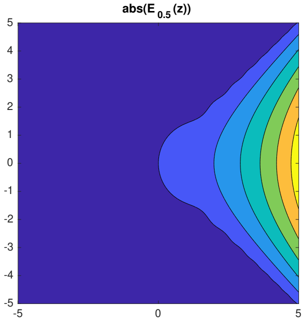
In the last lecture we have seen that the solution of this problem can be expressed as

$$ y(t) = E_{\alpha}(\lambda(t - t_0)^{\alpha})y_0. $$

Asymptotic behavior

The solution $y(t)$ asymptotically vanishes as $t \to +\infty$ for

$$ \lambda \in S^* = \{z \in \mathbb{C} : |\arg(z) - \pi| < (1 - \alpha/2)\pi.\} $$



Absolute stability

Informally

The stability region of the various PI formulas can be described as the set of all $z = \tau^\alpha \lambda$ for which the numerical solution $\{y^{(n)}\}_n$ behaves as the true solution and tends to 0 as $n \to +\infty$.

Absolute stability

Informally

The stability region of the various PI formulas can be described as the set of all $z = \tau^\alpha\lambda$ for which the numerical solution $\{y^{(n)}\}_n$ behaves as the true solution and tends to 0 as $n \to +\infty$.

As for the other theoretical result we are going to leverage information on the associated Volterra integral equation (Lubich 1986a).

- First we rewrite our non-homogeneous difference equation (in which we simplify the notation assuming to work with scalars) as

$$\begin{cases} y_n = f_n + \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(y_j), & n \geq 0 \\ f_n = f(t_n) + \tau^\alpha \sum_{j=-m}^{-1} w_{n,j} g(y_j), & t_n = t_0 + n\tau, \quad t_0 = mh. \end{cases}$$

- Then we assume that $h^\alpha w_{n,j} g(y_j) = O((n\tau)^{\alpha-1}\tau g(y_j))$, i.e., $w_{n,j} = O(n^{\alpha-1})$ as $n \to +\infty$, $j = -M, \ldots, -1$.

Absolute stability

A connection to the classical theory

In the classical case $\alpha = 1$, if we can express the term

$$\sum_{n=0}^{+\infty} \omega_n \zeta^n = \frac{\sigma(\zeta^{-1})}{\rho(\zeta^{-1})}$$

as a rational function, then we have found a standard Linear Multistep Method.

Absolute stability

A connection to the classical theory

In the classical case $\alpha = 1$, if we can express the term

$$\sum_{n=0}^{+\infty} \omega_n \zeta^n = \frac{\sigma(\zeta^{-1})}{\rho(\zeta^{-1})}$$

as a rational function, then we have found a standard Linear Multistep Method.

A-stable method

A convolution quadrature $\{\omega\}_n$ for the Abel equation

$$y(t) = f(t) + \frac{1}{\Gamma(\alpha)} \int_0^t (t-s)^{\alpha-1} g[y(s)]\, \mathrm{d}s, \quad t \geq 0,\ 0 < \alpha \leq 1,$$

is called *A*-stable if the solution $\{y_n\}_n$ given by the convolution quadrature satisfies

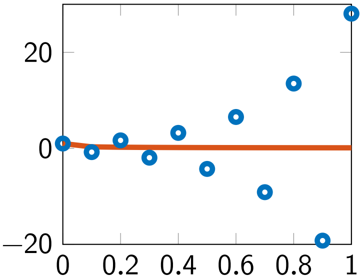
$$y_n \to 0 \text{ as } n \to +\infty \text{ whenever } \{f_n\}_n \text{ has a finite limit } \forall \tau > 0,\ \forall \lambda \in S^*.$$

Stability region

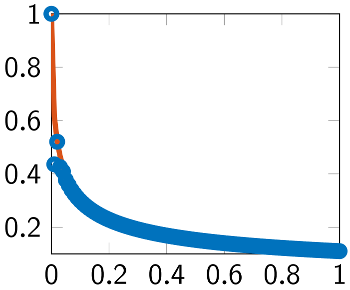
In general we cannot expect to have stability for every $\lambda \in S^*$, consider, e.g.

$$ {}_{CA}D^{\alpha}_{[t_0,t]} y(t) = -5y(t), \quad y(0) = 1, \quad T = 1. $$

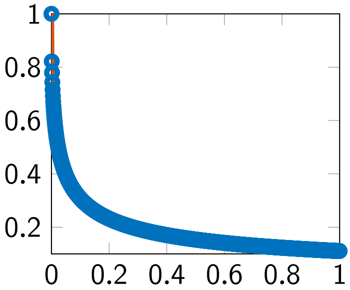
integrated with the explicit fractional rectangular rule



$h = 10^{-1}$



$h = 10^{-2}$



$h = 10^{-3}$

Stability region

Stability region

The *stability region* S of a convolution quadrature $\{\omega_m\}$ is the set of all complex $z = \tau^\alpha \lambda$ for which the numerical solution $\{y_n\}_n$ satisfies

$$y_n \to 0 \text{ as } n \to +\infty \text{ whenever } \{f_n\}_n \text{ has a finite limit.}$$

The method is called *strongly stable*, if for any $\lambda \in S^*$ there exists $\tau_0(\lambda) > 0$ such that $\tau^\alpha \lambda \in S$ for all $0 < \tau < \tau_0(\lambda)$. The method is called $A(\theta)$-stable if S contains the sector $|\arg(z) - \pi| < \theta$.

Stability region

Stability region

The *stability region* S of a convolution quadrature $\{\omega_m\}$ is the set of all complex $z = \tau^\alpha \lambda$ for which the numerical solution $\{y_n\}_n$ satisfies

$$y_n \to 0 \text{ as } n \to +\infty \text{ whenever } \{f_n\}_n \text{ has a finite limit.}$$

The method is called *strongly stable*, if for any $\lambda \in S^*$ there exists $\tau_0(\lambda) > 0$ such that $\tau^\alpha \lambda \in S$ for all $0 < \tau < \tau_0(\lambda)$. The method is called $A(\theta)$-stable if S contains the sector $|\arg(z) - \pi| < \theta$.

To obtain the characterization we need, we consider weights

$$\omega_n = (-1)^n \binom{-\alpha}{n} + v_n, \quad n \geq 0, \{v_n\}_n \in \ell^1, \tag{H_1}$$

to which corresponds

$$\omega(\zeta) = (1-\zeta)^{-\alpha} + v(\zeta) \text{ continuous in } \{\zeta \in \mathbb{C} : |\zeta| \leq 1,\ \zeta \neq 1\},\ \lim_{\zeta \to 1^-} w(\zeta) = +\infty.$$

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Proof. Let $z = \tau^\alpha \lambda$. Since 0 is neither contained in S^* nor in S, we can assume $z \neq 0$. We can rewrite our difference equation as

$$y(\zeta) = f(\zeta) + z\omega(\zeta)y(\zeta) \;\Leftrightarrow\; y(\zeta) = \frac{f(\zeta)}{1 - z\omega(\zeta)} = \frac{(1-\zeta)^\alpha f(\zeta)}{(1-\zeta)^\alpha[1 - z\omega(\zeta)]}.$$

We first prove that $S \subseteq S^*$.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \;:\; |\zeta| \leq 1\}.$$

Proof. Let $z = \tau^\alpha \lambda$. Since 0 is neither contained in S^* nor in S, we can assume $z \neq 0$. We can rewrite our difference equation as

$$y(\zeta) = f(\zeta) + z\omega(\zeta)y(\zeta) \;\Leftrightarrow\; y(\zeta) = \frac{f(\zeta)}{1 - z\omega(\zeta)} = \frac{(1-\zeta)^\alpha f(\zeta)}{(1-\zeta)^\alpha[1 - z\omega(\zeta)]}.$$

We first prove that $S \subseteq S^*$.

- The coefficient sequence $(1-\zeta)^\alpha[1 - z\omega(\zeta)]$ is in ℓ^1, indeed $v(\zeta)$ and $(1-\zeta)^\alpha$ are in ℓ^1 by using (H_1) (for the first one with $-\alpha$ instead of α), hence also $1 + (1-\zeta)^\alpha v(\zeta) = (1-\zeta)^\alpha \omega(\zeta)$, since for any two sequences in ℓ^1 we have $\sum_n |\sum_i a_{n-i} b_i| \leq \sum |a_i||b_i|$.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{{}^1\!/\omega(\zeta) \;:\; |\zeta| \leq 1\}.$$

Proof. Let $z = \tau^\alpha \lambda$. Since 0 is neither contained in S^* nor in S, we can assume $z \neq 0$. We can rewrite our difference equation as

$$y(\zeta) = f(\zeta) + z\omega(\zeta)y(\zeta) \;\Leftrightarrow\; y(\zeta) = \frac{f(\zeta)}{1 - z\omega(\zeta)} = \frac{(1-\zeta)^\alpha f(\zeta)}{(1-\zeta)^\alpha[1 - z\omega(\zeta)]}.$$

We first prove that $S \subseteq S^*$.

- The coefficient sequence $(1-\zeta)^\alpha[1 - z\omega(\zeta)]$ is in ℓ^1,
- If $z \in S$ then $1 - z\omega(\zeta) \neq 0$ for $|\zeta| \leq 1$ with $\zeta \neq 1$.

Stability region

Wiener inversion Theorem

$f(\zeta) = \sum_{n=0}^{+\infty} a_n \zeta^n$ with $\|f\|_1 < +\infty$, $\zeta = e^{in\theta}$, then $1/f(\theta) \in \ell^1$ iff $f(\theta) \neq 0$ for all θ.

Proof. Let $z = \tau^\alpha \lambda$. Since 0 is neither contained in S^* nor in S, we can assume $z \neq 0$. We can rewrite our difference equation as

$$y(\zeta) = f(\zeta) + z\omega(\zeta)y(\zeta) \;\Leftrightarrow\; y(\zeta) = \frac{f(\zeta)}{1 - z\omega(\zeta)} = \frac{(1-\zeta)^\alpha f(\zeta)}{(1-\zeta)^\alpha[1 - z\omega(\zeta)]}.$$

We first prove that $S \subseteq S^*$.

- The coefficient sequence $(1-\zeta)^\alpha[1 - z\omega(\zeta)]$ is in ℓ^1,
- If $z \in S$ then $1 - z\omega(\zeta) \neq 0$ for $|\zeta| \leq 1$ with $\zeta \neq 1$.

(H$_1$) $(1-\zeta)^\alpha[1 - z\omega(\zeta)] = (1-\zeta)^\alpha[1 - zv(\zeta)] - z$ and thus

$$(1-\zeta)^\alpha[1 - z\omega(\zeta)] \neq 0 \text{ for } |\zeta| \leq 1$$

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{{}^1/_{\omega(\zeta)} \,:\, |\zeta| \leq 1\}.$$

Proof. Let $z = \tau^\alpha \lambda$. Since 0 is neither contained in S^* nor in S, we can assume $z \neq 0$. We can rewrite our difference equation as

$$y(\zeta) = f(\zeta) + z\omega(\zeta)y(\zeta) \;\Leftrightarrow\; y(\zeta) = \frac{f(\zeta)}{1 - z\omega(\zeta)} = \frac{(1-\zeta)^\alpha f(\zeta)}{(1-\zeta)^\alpha[1 - z\omega(\zeta)]}.$$

We first prove that $S \subseteq S^*$.

- The coefficient sequence $(1-\zeta)^\alpha[1 - z\omega(\zeta)]$ is in ℓ^1,
- If $z \in S$ then $1 - z\omega(\zeta) \neq 0$ for $|\zeta| \leq 1$ with $\zeta \neq 1$.

(H_1) $(1-\zeta)^\alpha[1 - z\omega(\zeta)] = (1-\zeta)^\alpha[1 - zv(\zeta)] - z$ and thus

$$(1-\zeta)^\alpha[1 - z\omega(\zeta)] \neq 0 \text{ for } |\zeta| \leq 1 \;\Rightarrow\; {}^1/_{(1-\zeta)^\alpha[1-z\omega(\zeta)]} \in \ell^1.$$

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \;:\; |\zeta| \leq 1\}.$$

Proof. We first prove that $S \subseteq S^*$. Let $\tilde{f}_n = f_n - f_\infty \overset{n\to+\infty}{0}$ so that we can write

$$f(\zeta) = \frac{f_\infty}{1-\zeta} + \tilde{f}(\zeta) \;\Rightarrow\; (1-\zeta)^\alpha f(\zeta) = (1-\zeta)^{\alpha-1} f_\infty + (1-\zeta)^\alpha \tilde{f}(\zeta) \text{ has coefficients } \to 0.$$

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \;:\; |\zeta| \leq 1\}.$$

Proof. We first prove that $S \subseteq S^*$. Let $\tilde{f}_n = f_n - f_\infty \overset{n \to +\infty}{0}$ so that we can write

$$f(\zeta) = \frac{f_\infty}{1-\zeta} + \tilde{f}(\zeta) \;\Rightarrow\; (1-\zeta)^\alpha f(\zeta) = (1-\zeta)^{\alpha-1} f_\infty + (1-\zeta)^\alpha \tilde{f}(\zeta) \text{ has coefficients } \to 0.$$

By (H_1) the coefficient sequence of $(1-\zeta)^{\alpha-1} \to 0$.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \le 1\}.$$

Proof. We first prove that $S \subseteq S^*$. Let $\tilde{f}_n = f_n - f_\infty \xrightarrow{n\to+\infty} 0$ so that we can write

$$f(\zeta) = \frac{f_\infty}{1-\zeta} + \tilde{f}(\zeta) \;\Rightarrow\; (1-\zeta)^\alpha f(\zeta) = (1-\zeta)^{\alpha-1} f_\infty + (1-\zeta)^\alpha \tilde{f}(\zeta) \text{ has coefficients } \to 0.$$

By (H_1) the coefficient sequence of $(1-\zeta)^{\alpha-1} \to 0$. The coefficient sequence of $(1-\zeta)^\alpha \tilde{f}(\zeta) \to 0$ since $(1-\zeta)^\alpha \in \ell_1$ and $\ell_1 * c_0 \subset c_0$ for $*$ the convolution operator, and c_0 the space of zero sequences

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{{}^1\!/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Proof. We first prove that $S \subseteq S^*$. Let $\tilde{f}_n = f_n - f_\infty \overset{n\to+\infty}{0}$ so that we can write

$$f(\zeta) = \frac{f_\infty}{1-\zeta} + \tilde{f}(\zeta) \;\Rightarrow\; (1-\zeta)^\alpha f(\zeta) = (1-\zeta)^{\alpha-1} f_\infty + (1-\zeta)^\alpha \tilde{f}(\zeta) \text{ has coefficients } \to 0.$$

By (H_1) the coefficient sequence of $(1-\zeta)^{\alpha-1} \to 0$. The coefficient sequence of $(1-\zeta)^\alpha \tilde{f}(\zeta) \to 0$ since $(1-\zeta)^\alpha \in \ell_1$ and $\ell_1 * c_0 \subset c_0$ for $*$ the convolution operator, and c_0 the space of zero sequences $\Rightarrow$ the sequence $\{y_n\}_n$ of $y(\zeta)$ is in c_0. Hence we have proved that if $z \in S$ then $z \in S^*$.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Proof. To conclude we need to prove that S^* is exhausted by S, we assume that

$$1 - z\omega(\zeta_0) = 0 \text{ for some } |\zeta_0| \leq 1 \text{ and by } (\mathrm{H}_1)\ \zeta_0 \neq 1,$$

and show that then $z \notin S^*$.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Proof. To conclude we need to prove that S^* is exhausted by S, we assume that

$$1 - z\omega(\zeta_0) = 0 \text{ for some } |\zeta_0| \leq 1 \text{ and by } (\mathrm{H}_1)\ \zeta_0 \neq 1,$$

and show that then $z \notin S^*$.We select

$$y(\zeta) = \frac{(1-\zeta)^\alpha}{\zeta - \zeta_0} = \frac{(1-\zeta)^\alpha - (1-\zeta_0)^\alpha}{\zeta - \zeta_0} + (1-\zeta_0)^\alpha \frac{1}{\zeta - \zeta_0}.$$

Stability region

Lemma (Lubich 1986a, Lemma 2.1)

Assume that the coefficient sequence of $a(\zeta)$ is in ℓ^1. Let $|\zeta_0| \leq 1$. Then the coefficient sequence of

$$\frac{a(\zeta) - a(\zeta_0)}{\zeta - \zeta_0} \text{ converges to zero.}$$

Proof. To conclude we need to prove that S^* is exhausted by S, we assume that

$$1 - z\omega(\zeta_0) = 0 \text{ for some } |\zeta_0| \leq 1 \text{ and by } (\mathrm{H}_1)\ \zeta_0 \neq 1,$$

and show that then $z \notin S^*$.We select

$$y(\zeta) = \frac{(1-\zeta)^\alpha}{\zeta - \zeta_0} = \underbrace{\frac{(1-\zeta)^\alpha - (1-\zeta_0)^\alpha}{\zeta - \zeta_0}}_{=0} + (1-\zeta_0)^\alpha \frac{1}{\zeta - \zeta_0}.$$

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Proof. To conclude we need to prove that S^* is exhausted by S, we assume that

$$1 - z\omega(\zeta_0) = 0 \text{ for some } |\zeta_0| \leq 1 \text{ and by } (\mathrm{H}_1)\ \zeta_0 \neq 1,$$

and show that then $z \notin S^*$.We select

$$y(\zeta) = \frac{(1-\zeta)^\alpha}{\zeta - \zeta_0} = +(1-\zeta_0)^\alpha \frac{1}{\zeta - \zeta_0}.$$

On the other hand, $1/\zeta - \zeta_0 = -\sum_{n=0}^{+\infty} {\zeta_0}^{-n-1} \zeta^n$ diverges! Hence also the sequence associated to $y(\zeta)$ diverges.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Proof. We can now collect the various parts together

$$\begin{aligned} f(\zeta) = [1 - z\omega(\zeta)]y(\zeta) =& (1-\zeta)^{\alpha}[1 - z\omega(\zeta)](1-\zeta)^{-\alpha}y(\zeta) \\ =& \frac{(1-\zeta)^{\alpha}(1 - z\omega(\zeta)) - (1-\zeta_0)(1 - z\omega(\zeta_0))}{\zeta - \zeta_0} \end{aligned}$$

using again the lemma we get that $\{f_n\}_n$ goes to zero, but, $\{y_n\}_n$ does not, hence $z \notin S^*$.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Corollary

If a convolution quadrature satisfying (H_1) is applied to the Volterra equation and if $\tau^\alpha \lambda \in S$, then $\{y_n\}_n$ is bounded whenever $\{f_n\}_n$ is bounded. Conversely, if $\{y_n\}_n$ is bounded whenever $\{f_n\}_n$ is bounded then $\tau^\alpha \lambda \in \overline{S}$.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) : |\zeta| \leq 1\}.$$

Corollary

The stability region of an explicit convolution quadrature ($\omega_0 = 0$) satisfying (H_1) is bounded.

Proof. By the open mapping theorem $\omega(\zeta)$ maps neighborhood of 0 into neighborhood of 0. Hence S^* is a neighborhood of ∞, and the result follows from the Theorem.

Stability region

Theorem (Lubich 1986a, Theorem 2.1)

The stability region of a convolution quadrature under the condition (H_1) is

$$S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}.$$

Corollary

The stability region of an explicit convolution quadrature $(\omega_0 = 0)$ satisfying (H_1) is bounded.

Corollary

Every convolution quadrature satisfying (H_1) is strongly stable.

- Using these results we can recover the stability regions for the different methods,
- Often PI rules do not possess analytical representation of $\omega(\zeta)$ we can just use numerical approximations.

Stability region: predictor corrector method

For the Predictor-Corrector method we have

$$\begin{cases} y_P^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \sum_{j=0}^{n} b_{j,n+1} f(t_j, y^{(j)}), \\ y^{(n+1)} = y^{(0)} + \frac{\tau^\alpha}{\Gamma(\alpha)} \left(\sum_{j=0}^{n} a_{j,n+1} f(t_j, y^{(j)}) + a_{n+1,n+1} f(t_{n+1}, y_P^{(n+1)}) \right) \end{cases}$$

where

$$b_{j,n+1} = \frac{(n+1-j)^\alpha - (n-j)^\alpha}{\alpha}$$

$$a_{j,n+1} = \begin{cases} {(n^{\alpha+1} - (n-\alpha)(n+1)^\alpha} / {\alpha(\alpha+1)}, & j = 0, \\ {(n-j+2)^{\alpha+1} - 2(n-j+1)^{\alpha+1} + (n-j)^{\alpha+1}} / {\alpha(\alpha+1)}, & j = 1, 2, \dots, n, \\ {1}/{\alpha(\alpha+1)}, & j = n+1. \end{cases}$$

Stability region: predictor corrector method

For the Predictor-Corrector method we have

$$\begin{cases} y_P^{(n+1)} = y^{(0)} + \tau^\alpha \sum_{j=0}^{n} b_{n-j-1} f(t_j, y^{(j)}), \\ y^{(n+1)} = y^{(0)} + \tau^\alpha a_{n,0} f^{(0)} + \tau^\alpha \sum_{j=1}^{n} a_{n-j} f(t_n, y_P^{(n+1)}) \end{cases}$$

where

$$b_n = \frac{(n+1)^\alpha - n^\alpha}{\Gamma(\alpha+1)}$$

$$a_{n,0} = {}^{(n-1)^{\alpha+1} - n^\alpha(n-\alpha-1)}\!/\!{}_{\Gamma(\alpha+2)},$$

$$a_n = \begin{cases} {}^{1}\!/\!{}_{\Gamma(\alpha+2)}, & n = 0, \\ {}^{(n-1)^{\alpha+1} - 2n^{\alpha+1} + (n+1)^{\alpha+1}}\!/\!{}_{\Gamma(\alpha+2)}, & n \geq 1. \end{cases}$$

Stability region: predictor corrector method

For the Predictor-Corrector method we have

$$y^{(n)} = g^{(n)} + \sum_{j=k}^{n} c_{n-j} y^{(j)}, \quad n \geq k,$$

where

$$\begin{cases} g^{(n)} = (1 + za_{n,0} + za_0 + z^2 a_0 b_{n-1}) y^{(0)}, \\ c_0 = 0, \ c_n = za_n + z^2 a_0 b_{n-1}, & n \geq 1. \end{cases}$$

Stability region: predictor corrector method

For the Predictor-Corrector method we have

$$y^{(n)} = g^{(n)} + \sum_{j=k}^{n} c_{n-j} y^{(j)}, \quad n \geq k,$$

where

$$\begin{cases} g^{(n)} = (1 + z a_{n,0} + z a_0 + z^2 a_0 b_{n-1}) y^{(0)}, \\ c_0 = 0,\; c_n = z a_n + z^2 a_0 b_{n-1}, & n \geq 1. \end{cases}$$

⚙ To apply the stability region Theorem we have then to investigate the quantity $1 - c(\zeta)$ for $|\zeta| \leq 1$, and $c(\zeta) = \sum_{n=0}^{+\infty} c_n \zeta^n$.

Stability region: predictor corrector method

For the Predictor-Corrector method we have

$$y^{(n)} = g^{(n)} + \sum_{j=k}^{n} c_{n-j} y^{(j)}, \quad n \geq k,$$

where

$$\begin{cases} g^{(n)} = (1 + z a_{n,0} + z a_0 + z^2 a_0 b_{n-1}) y^{(0)}, & \\ c_0 = 0, \; c_n = z a_n + z^2 a_0 b_{n-1}, & n \geq 1. \end{cases}$$

⚙ To apply the stability region Theorem we have then to investigate the quantity $1 - c(\zeta)$ for $|\zeta| \leq 1$, and $c(\zeta) = \sum_{n=0}^{+\infty} c_n \zeta^n$.

Proposition

The stability region of the Predictor-Corrector method is

$$S = \{z \in \mathbb{C} \mid 1 - z(\alpha(\zeta) - a_0) - z^2 a_0 \zeta b(\zeta) \neq 0 \; : \; |\zeta| \leq 1\}.$$

Stability region: predictor corrector method

Proposition

The stability region of the Predictor-Corrector method is

$$S = \{z \in \mathbb{C} \mid 1 - z(\alpha(\zeta) - a_0) - z^2 a_0 \zeta b(\zeta) \neq 0 \ : \ |\zeta| \leq 1\}.$$

Proof. To apply the Theorem we need to prove (H_1), we use the binomial series to write

$$(n-1)^p = n^p - pn^{p-1} + \frac{p(p-1)}{2} n^{p-2} + \frac{p(p-1)(p-2)}{6} n^{p-3} + O(n^{p-4}),$$

and similarly for $(n+1)^p$, from which we obtain

$$b_n = \frac{1}{\Gamma(\alpha)} n^{\alpha-1} + O(n^{\alpha-2}),\ a_{n,0} = \frac{1}{2\Gamma(\alpha)} n^{\alpha-1} + O(n^{\alpha-2}),\ \alpha_n = \frac{1}{\Gamma(\alpha)} n^{\alpha-1} + O(n^{\alpha-3}),$$

and the expression we need for $c(\zeta)$ as

$$c(\zeta) = z(\alpha(\zeta) - \alpha_0) + z^2 \alpha_0 \zeta b(\zeta). \qquad \square$$

Stability region: predictor corrector method

Proposition

The stability region of the Predictor-Corrector method is

$$S = \{z \in \mathbb{C} \mid 1 - z(\alpha(\zeta) - a_0) - z^2 a_0 \zeta b(\zeta) \neq 0 \;:\; |\zeta| \leq 1\}.$$

Proof. To apply the Theorem we need to prove (H_1), we use the binomial series to write

$$(n-1)^p = n^p - pn^{p-1} + \frac{p(p-1)}{2} n^{p-2} + \frac{p(p-1)(p-2)}{6} n^{p-3} + O(n^{p-4}),$$

and similarly for $(n+1)^p$, from which we obtain

$$b_n = \frac{1}{\Gamma(\alpha)} n^{\alpha-1} + O(n^{\alpha-2}),\; a_{n,0} = \frac{1}{2\Gamma(\alpha)} n^{\alpha-1} + O(n^{\alpha-2}),\; \alpha_n = \frac{1}{\Gamma(\alpha)} n^{\alpha-1} + O(n^{\alpha-3}),$$

and the expression we need for $c(\zeta)$ as

$$c(\zeta) = z(\alpha(\zeta) - \alpha_0) + z^2 \alpha_0 \zeta b(\zeta). \qquad \square$$

- The expression can be evaluated only numerically.

A research idea?

We have written a predictor-method in an explicit form, we can write and analyze in a similar way also a predictor-corrector made of two *implicit methods*.

- We have now to solve a (possibly) non-linear problem at each step, thus things don't seem to good...
- But we can expect better stability and convergence properties.

A research idea?

We have written a predictor-method in an explicit form, we can write and analyze in a similar way also a predictor-corrector made of two *implicit methods*.

- We have now to solve a (possibly) non-linear problem at each step, thus things don't seem to good...
- But we can expect better stability and convergence properties.

What if we decide to solve the nonlinear problem in reduced precision?

A research idea?

We have written a predictor-method in an explicit form, we can write and analyze in a similar way also a predictor-corrector made of two *implicit methods*.

- We have now to solve a (possibly) non-linear problem at each step, thus things don't seem to good...
- But we can expect better stability and convergence properties.

What if we decide to solve the nonlinear problem in reduced precision?

Multiprecision algorithms on specialized hardware can give both an acceleration and maintain the overall accuracy. This idea has already been partially explored for the ODE case, but not yet for FODEs:

B. Burnett et al. (2021). "Performance Evaluation of Mixed-Precision Runge-Kutta Methods". In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, pp. 1–6

A research idea?

We have written a predictor-method in an explicit form, we can write and analyze in a similar way also a predictor-corrector made of two *implicit methods*.

- We have now to solve a (possibly) non-linear problem at each step, thus things don't seem to good...
- But we can expect better stability and convergence properties.

What if we decide to solve the nonlinear problem in reduced precision?

Multiprecision algorithms on specialized hardware can give both an acceleration and maintain the overall accuracy. This idea has already been partially explored for the ODE case, but not yet for FODEs:

B. Burnett et al. (2021). "Performance Evaluation of Mixed-Precision Runge-Kutta Methods". In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, pp. 1–6

Further analyses

One can investigate also stability regions, effects of multiple correction steps, tolerances and step-size selections...

Fractional Linear Multistep Method

To obtain methods that can be analyzed we can move to *Linear Multistep Methods*.

Fractional Linear Multistep Method

To obtain methods that can be analyzed we can move to *Linear Multistep Methods*.

- For an ODE a FLMM with k step is a method of the form:

$$\sum_{j=0}^{k} a_j y_{n+j} = \tau \sum_{j=0}^{k} b_j f_{n+j}, \quad n = 0, \ldots, s. \tag{1}$$

  for $t_j = t_0 + j\tau$, for $j = 0, \ldots, N$, $\tau = (T - t_0)/N$,

Fractional Linear Multistep Method

To obtain methods that can be analyzed we can move to *Linear Multistep Methods*.

- For an ODE a FLMM with k step is a method of the form:

$$\sum_{j=0}^{k} a_j y_{n+j} = \tau \sum_{j=0}^{k} b_j f_{n+j}, \quad n = 0, \ldots, s. \tag{1}$$

  for $t_j = t_0 + j\tau$, for $j = 0, \ldots, N$, $\tau = (T - t_0)/N$,
- They are associated with the polynomials $\rho(z) = \sum_{j=0}^{k} a_j z^j$, $\sigma(z) = \sum_{j=0}^{k} b_j z^j$,

Fractional Linear Multistep Method

To obtain methods that can be analyzed we can move to *Linear Multistep Methods*.

- For an ODE a FLMM with k step is a method of the form:

$$\sum_{j=0}^{k} a_j y_{n+j} = \tau \sum_{j=0}^{k} b_j f_{n+j}, \quad n = 0, \ldots, s. \tag{1}$$

  for $t_j = t_0 + j\tau$, for $j = 0, \ldots, N$, $\tau = (T - t_0)/N$,
- They are associated with the polynomials $\rho(z) = \sum_{j=0}^{k} a_j z^j$, $\sigma(z) = \sum_{j=0}^{k} b_j z^j$,
- The fractional version has been introduced in the pioneering work (Lubich 1986b)

Theorem (Lubich 1986b, Theorem 2.6)

Let (ρ, σ) denote an implicit linear multistep method which is stable and consistent of order p. Assume that the zeros of $\sigma(\zeta)$ have absolute values less than 1. Let $w(\zeta) = {}^{\sigma(\zeta^{-1})}/_{\rho(\zeta^{-1})}$ denote the generating power series of the corresponding convolution quadrature ω. We define $\omega^\alpha = \{\omega_n^{(\alpha)}\}_{n=0}^{+\infty}$ by $\omega^\alpha(\zeta) = \omega(\zeta)^\alpha$, then the convolution quadrature ω^α is convergent of order p.

Fractional Linear Multistep Method

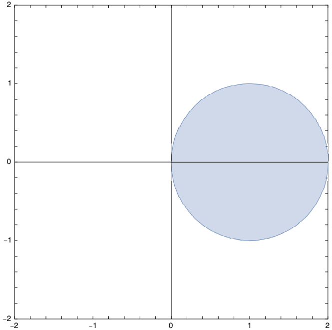
An example is represented by **B**ackward **D**ifferentiation **F**ormula**s**, for which we have

p	$\omega^{\alpha}(\zeta)$
1	$(1-\zeta)^{-\alpha}$
2	$({}^{3}\!/_{2} - 2\zeta + {}^{1}\!/_{2}\zeta^2)^{-\alpha}$
3	$({}^{11}\!/_{6} - 3\zeta + {}^{3}\!/_{2}\zeta^2 - {}^{1}\!/_{3}\zeta^3)^{-\alpha}$
4	$({}^{25}\!/_{12} - 4\zeta + 4\zeta^2 - {}^{4}\!/_{3}\zeta^3 + {}^{1}\!/_{4}\zeta^4)^{-\alpha}$
5	$({}^{137}\!/_{60} - 5\zeta + 5\zeta^2 - {}^{10}\!/_{3}\zeta^3 + {}^{5}\!/_{4}\zeta^4 - {}^{1}\!/_{5}\zeta^5)^{-\alpha}$
6	$({}^{147}\!/_{60} - 6\zeta + {}^{15}\!/_{2}\zeta^2 - {}^{20}\!/_{3}\zeta^3 + {}^{15}\!/_{4}\zeta^4 - {}^{6}\!/_{5}\zeta^5 + {}^{1}\!/_{6}\zeta^6)^{-\alpha}$

Fractional Linear Multistep Method

An example is represented by **B**ackward **D**ifferentiation **F**ormulas, for which we have

p	$\omega^\alpha(\zeta)$
1	$(1-\zeta)^{-\alpha}$
2	$(3/2 - 2\zeta + 1/2\zeta^2)^{-\alpha}$
3	$(11/6 - 3\zeta + 3/2\zeta^2 - 1/3\zeta^3)^{-\alpha}$
4	$(25/12 - 4\zeta + 4\zeta^2 - 4/3\zeta^3 + 1/4\zeta^4)^{-\alpha}$
5	$(137/60 - 5\zeta + 5\zeta^2 - 10/3\zeta^3 + 5/4\zeta^4 - 1/5\zeta^5)^{-\alpha}$
6	$(147/60 - 6\zeta + 15/2\zeta^2 - 20/3\zeta^3 + 15/4\zeta^4 - 6/5\zeta^5 + 1/6\zeta^6)^{-\alpha}$



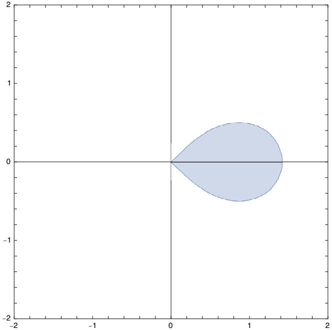
$\alpha = 1$

For $p = 1$ we can consider the stability region $S = \mathbb{C} \setminus \{1/\omega(\zeta) \, : \, |\zeta| \leq 1\}$ and plot the part of the $\mathbb{C}$-plane we have to remove.

Fractional Linear Multistep Method

An example is represented by **B**ackward **D**ifferentiation **F**ormula**s**, for which we have

p	$\omega^{\alpha}(\zeta)$
1	$(1-\zeta)^{-\alpha}$
2	$(3/2-2\zeta+1/2\zeta^2)^{-\alpha}$
3	$(11/6-3\zeta+3/2\zeta^2-1/3\zeta^3)^{-\alpha}$
4	$(25/12-4\zeta+4\zeta^2-4/3\zeta^3+1/4\zeta^4)^{-\alpha}$
5	$(137/60-5\zeta+5\zeta^2-10/3\zeta^3+5/4\zeta^4-1/5\zeta^5)^{-\alpha}$
6	$(147/60-6\zeta+15/2\zeta^2-20/3\zeta^3+15/4\zeta^4-6/5\zeta^5+1/6\zeta^6)^{-\alpha}$



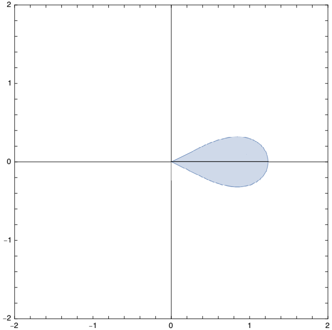
$\alpha = 0.5$

For $p=1$ we can consider the stability region $S = \mathbb{C} \setminus \{1/\omega(\zeta) \ : \ |\zeta| \leq 1\}$ and plot the part of the $\mathbb{C}$-plane we have to remove.

Fractional Linear Multistep Method

An example is represented by **B**ackward **D**ifferentiation **F**ormulas, for which we have

p	$\omega^\alpha(\zeta)$
1	$(1-\zeta)^{-\alpha}$
2	$(3/2 - 2\zeta + 1/2\zeta^2)^{-\alpha}$
3	$(11/6 - 3\zeta + 3/2\zeta^2 - 1/3\zeta^3)^{-\alpha}$
4	$(25/12 - 4\zeta + 4\zeta^2 - 4/3\zeta^3 + 1/4\zeta^4)^{-\alpha}$
5	$(137/60 - 5\zeta + 5\zeta^2 - 10/3\zeta^3 + 5/4\zeta^4 - 1/5\zeta^5)^{-\alpha}$
6	$(147/60 - 6\zeta + 15/2\zeta^2 - 20/3\zeta^3 + 15/4\zeta^4 - 6/5\zeta^5 + 1/6\zeta^6)^{-\alpha}$



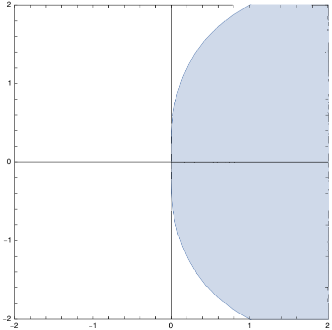
$\alpha = 0.3$

For $p = 1$ we can consider the stability region $S = \mathbb{C} \setminus \{1/\omega(\zeta) \; : \; |\zeta| \leq 1\}$ and plot the part of the $\mathbb{C}$-plane we have to remove.

Fractional Linear Multistep Method

An example is represented by **B**ackward **D**ifferentiation **F**ormulas, for which we have

p	$\omega^\alpha(\zeta)$
1	$(1-\zeta)^{-\alpha}$
2	$(3/2 - 2\zeta + 1/2\zeta^2)^{-\alpha}$
3	$(11/6 - 3\zeta + 3/2\zeta^2 - 1/3\zeta^3)^{-\alpha}$
4	$(25/12 - 4\zeta + 4\zeta^2 - 4/3\zeta^3 + 1/4\zeta^4)^{-\alpha}$
5	$(137/60 - 5\zeta + 5\zeta^2 - 10/3\zeta^3 + 5/4\zeta^4 - 1/5\zeta^5)^{-\alpha}$
6	$(147/60 - 6\zeta + 15/2\zeta^2 - 20/3\zeta^3 + 15/4\zeta^4 - 6/5\zeta^5 + 1/6\zeta^6)^{-\alpha}$



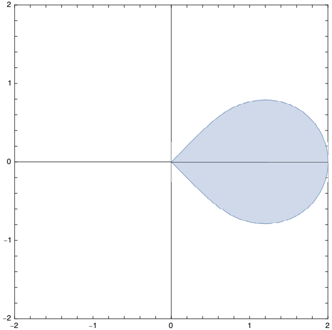
$\alpha = 1$

For $p = 1$ we can consider the stability region $S = \mathbb{C} \setminus \{1/\omega(\zeta) \ : \ |\zeta| \leq 1\}$ and plot the part of the $\mathbb{C}$-plane we have to remove.

Fractional Linear Multistep Method

An example is represented by **B**ackward **D**ifferentiation **F**ormula**s**, for which we have

p	$\omega^\alpha(\zeta)$
1	$(1-\zeta)^{-\alpha}$
2	$(3/2 - 2\zeta + 1/2\zeta^2)^{-\alpha}$
3	$(11/6 - 3\zeta + 3/2\zeta^2 - 1/3\zeta^3)^{-\alpha}$
4	$(25/12 - 4\zeta + 4\zeta^2 - 4/3\zeta^3 + 1/4\zeta^4)^{-\alpha}$
5	$(137/60 - 5\zeta + 5\zeta^2 - 10/3\zeta^3 + 5/4\zeta^4 - 1/5\zeta^5)^{-\alpha}$
6	$(147/60 - 6\zeta + 15/2\zeta^2 - 20/3\zeta^3 + 15/4\zeta^4 - 6/5\zeta^5 + 1/6\zeta^6)^{-\alpha}$



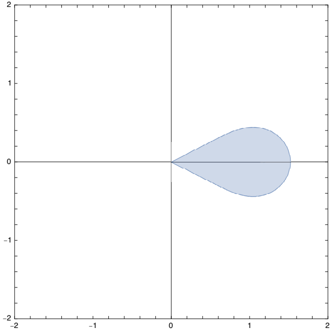
$\alpha = 0.5$

For $p = 1$ we can consider the stability region $S = \mathbb{C} \setminus \{1/\omega(\zeta) \;:\; |\zeta| \leq 1\}$ and plot the part of the $\mathbb{C}$-plane we have to remove.

Fractional Linear Multistep Method

An example is represented by **B**ackward **D**ifferentiation **F**ormulas, for which we have

p	$\omega^{\alpha}(\zeta)$
1	$(1-\zeta)^{-\alpha}$
2	$(3/2 - 2\zeta + 1/2\zeta^2)^{-\alpha}$
3	$(11/6 - 3\zeta + 3/2\zeta^2 - 1/3\zeta^3)^{-\alpha}$
4	$(25/12 - 4\zeta + 4\zeta^2 - 4/3\zeta^3 + 1/4\zeta^4)^{-\alpha}$
5	$(137/60 - 5\zeta + 5\zeta^2 - 10/3\zeta^3 + 5/4\zeta^4 - 1/5\zeta^5)^{-\alpha}$
6	$(147/60 - 6\zeta + 15/2\zeta^2 - 20/3\zeta^3 + 15/4\zeta^4 - 6/5\zeta^5 + 1/6\zeta^6)^{-\alpha}$



$\alpha = 0.3$

For $p = 1$ we can consider the stability region $S = \mathbb{C} \setminus \{1/\omega(\zeta) \,:\, |\zeta| \leq 1\}$ and plot the part of the $\mathbb{C}$-plane we have to remove.

Fractional Linear Multistep Method

An example is represented by **B**ackward **D**ifferentiation **F**ormula**s**, for which we have

p	$\omega^\alpha(\zeta)$
1	$(1-\zeta)^{-\alpha}$
2	$(3/2 - 2\zeta + 1/2\zeta^2)^{-\alpha}$
3	$(11/6 - 3\zeta + 3/2\zeta^2 - 1/3\zeta^3)^{-\alpha}$
4	$(25/12 - 4\zeta + 4\zeta^2 - 4/3\zeta^3 + 1/4\zeta^4)^{-\alpha}$
5	$(137/60 - 5\zeta + 5\zeta^2 - 10/3\zeta^3 + 5/4\zeta^4 - 1/5\zeta^5)^{-\alpha}$
6	$(147/60 - 6\zeta + 15/2\zeta^2 - 20/3\zeta^3 + 15/4\zeta^4 - 6/5\zeta^5 + 1/6\zeta^6)^{-\alpha}$

How do we obtain the coefficients?

How can we obtain the coefficient describing the method?

Computing the FLMM coefficients

We have now the converse of the previous problem, we have a closed expression for $\omega(\zeta)$, and now we need the coefficients to write

$$I_\tau^\alpha g(t_n) = \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(t_j) + \tau^\beta \sum_{j=0}^{s} w_{n,j} g(t_j),$$

- $\{\omega_j\}_{j=0}^n$ convolution coefficients from $\omega(\zeta)$,
- $\{w_{n,j}\}_{j=0}^k$ starting quadrature weights.

Computing the FLMM coefficients

We have now the converse of the previous problem, we have a closed expression for $\omega(\zeta)$, and now we need the coefficients to write

$$I_\tau^\alpha g(t_n) = \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(t_j) + \tau^\beta \sum_{j=0}^{s} w_{n,j} g(t_j),$$

- $\{\omega_j\}_{j=0}^{n}$ convolution coefficients from $\omega(\zeta)$,
- $\{w_{n,j}\}_{j=0}^{k}$ starting quadrature weights.
- For the convolution coefficients we can use:

Computing the FLMM coefficients

We have now the converse of the previous problem, we have a closed expression for $\omega(\zeta)$, and now we need the coefficients to write

$$I_\tau^\alpha g(t_n) = \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(t_j) + \tau^\beta \sum_{j=0}^{s} w_{n,j} g(t_j),$$

- $\{\omega_j\}_{j=0}^n$ convolution coefficients from $\omega(\zeta)$,
- $\{w_{n,j}\}_{j=0}^k$ starting quadrature weights.
- For the convolution coefficients we can use:
  - Fast Fourier Transform (FFT) techniques for formal power series,

Computing the FLMM coefficients

We have now the converse of the previous problem, we have a closed expression for $\omega(\zeta)$, and now we need the coefficients to write

$$I_\tau^\alpha g(t_n) = \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(t_j) + \tau^\beta \sum_{j=0}^{s} w_{n,j} g(t_j),$$

- $\{\omega_j\}_{j=0}^{n}$ convolution coefficients from $\omega(\zeta)$,
- $\{w_{n,j}\}_{j=0}^{k}$ starting quadrature weights.
- For the convolution coefficients we can use:
  - Fast Fourier Transform (FFT) techniques for formal power series,
  - A recursion technique for complex binomial series.

Computing the FLMM coefficients

We have now the converse of the previous problem, we have a closed expression for $\omega(\zeta)$, and now we need the coefficients to write

$$I_\tau^\alpha g(t_n) = \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(t_j) + \tau^\beta \sum_{j=0}^{s} w_{n,j} g(t_j),$$

- $\{\omega_j\}_{j=0}^{n}$ convolution coefficients from $\omega(\zeta)$,
- $\{w_{n,j}\}_{j=0}^{k}$ starting quadrature weights.
- For the convolution coefficients we can use:
  - Fast Fourier Transform (FFT) techniques for formal power series,
  - A recursion technique for complex binomial series.
- Solving a small $k \times k$ Vandermonde system.

The Newton Method for Power Series (Henrici 1979)

Let us suppose that $\alpha = 1/2$ and that we have a power series of the form

$$\omega(\zeta) = \sum_{j=0}^{+\infty} \omega_j \zeta^j,$$

for which we want to compute for a generic pth degree BDF

$$\omega(\zeta)^{-2} = q(\zeta) \text{ with } q(\zeta) = \sum_{k=1}^{p} \frac{1}{k}(1-\zeta)^k,$$

The Newton Method for Power Series (Henrici 1979)

Let us suppose that $\alpha = 1/2$ and that we have a power series of the form

$$\omega(\zeta) = \sum_{j=0}^{+\infty} \omega_j \zeta^j,$$

for which we want to compute for a generic pth degree BDF

$$F(\omega(\zeta)) = 0 \text{ with } F(w) = w^{-2} - q(\zeta).$$

The Newton Method for Power Series (Henrici 1979)

Let us suppose that $\alpha = 1/2$ and that we have a power series of the form

$$\omega(\zeta) = \sum_{j=0}^{+\infty} \omega_j \zeta^j,$$

for which we want to compute for a generic pth degree BDF
To which we can apply the Newton's method for power series

$$\begin{cases} \omega^{(0)}(\zeta) = \omega_0, \\ \omega^{(m+1)}(\zeta) = \left[\omega^{(m)}(\zeta) - F'(\omega^{(m)}(\zeta))^{-1} F(\omega^{(m)}(\zeta))\right]_{2^{m+1}}, \end{cases}$$

for $[\cdot]_k$ the truncation operator for a power series, i.e., $\left[\sum_{j=0}^{+\infty} a_j \zeta^j\right]_k = \sum_{j=0}^{k} a_j \zeta^j$, and ω_0 the solution of $[F(\omega_0)]_1 = 0$.

The Newton Method for Power Series (Henrici 1979)

Let us suppose that $\alpha = 1/2$ and that we have a power series of the form

$$\omega(\zeta) = \sum_{j=0}^{+\infty} \omega_j \zeta^j,$$

for which we want to compute for a generic pth degree BDF
To which we can apply the Newton's method for power series

$$\begin{cases} \omega^{(0)}(\zeta) = \omega_0 = q(0)^{-1/2}, \\ \omega^{(m+1)}(\zeta) = \left[3/2\,\omega^{(m)}(\zeta) - 1/2\left(\omega^{(m)}(\zeta)\right)^3 q(\zeta)\right]_{2^{m+1}}, \end{cases}$$

for $[\cdot]_k$ the truncation operator for a power series, i.e., $\left[\sum_{j=0}^{+\infty} a_j \zeta^j\right]_k = \sum_{j=0}^{k} a_j \zeta^j$.
After m step we have that

$$\omega^{(m)}(\zeta) = [\omega(\zeta)]_{2^m} = \sum_{j=0}^{2^m-1} \omega_j \zeta^j \quad \forall m \geq 0 \text{ and cost } O(2^m \log(2^m)).$$

Recurrence relation

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 $\alpha \in \mathcal{C}$, we have

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

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

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

- This approach costs an $O(N^2)$ in general, but can be simplified, e.g., when $a_1 = \pm 1$, and $a_i > 0$ for $i > 1$ it involves only $2N$ multiplications and N additions.

Computing the starting weights

The starting weights $w_{n,j}$ in

$$I_\tau^\alpha g(t_n) = \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(t_j) + \tau^\beta \sum_{j=0}^{s} w_{n,j} g(t_j),$$

are introduced to deal with the singular behavior of the solution close to the left endpoint of the integration interval.

Computing the starting weights

The starting weights $w_{n,j}$ in

$$I_\tau^\alpha g(t_n) = \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(t_j) + \tau^\beta \sum_{j=0}^{s} w_{n,j} g(t_j),$$

are introduced to deal with the singular behavior of the solution close to the left endpoint of the integration interval.

Starting weight selection

We fix them by imposing that $I_\tau^\alpha t^\nu$ is exact for $\nu \in \mathcal{A} = \mathcal{A}_{p-1} \cup \{p-1\}$ with p the order of convergence of the FLMM, and $\mathcal{A}_{p-1} = \{\nu \in \mathbb{R} \,|\, \nu = i + j\alpha, \quad i, j \in \mathbb{N}, \nu < p-1\}$.

Computing the starting weights

The starting weights $w_{n,j}$ in

$$I_\tau^\alpha g(t_n) = \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} g(t_j) + \tau^\beta \sum_{j=0}^{s} w_{n,j} g(t_j),$$

are introduced to deal with the singular behavior of the solution close to the left endpoint of the integration interval.

Starting weight selection

We fix them by imposing that $I_\tau^\alpha t^\nu$ is exact for $\nu \in \mathcal{A} = \mathcal{A}_{p-1} \cup \{p-1\}$ with p the order of convergence of the FLMM, and $\mathcal{A}_{p-1} = \{\nu \in \mathbb{R} \,|\, \nu = i + j\alpha, \quad i, j \in \mathbb{N}, \nu < p-1\}$.

$$\tau^\alpha \sum_{j=0}^{s} w_{n,j} (jh)^\nu = \frac{1}{\Gamma(\alpha)} \int_0^{n\tau} (n\tau - \xi)^{\alpha-1} \chi^\nu \,\mathrm{d}\chi - \tau^\alpha \sum_{j=0}^{n} \omega_{n-j} (jh)^\nu, \quad \nu \in \mathcal{A}.$$

Solving the Vandermonde system

The resulting linear system is of “real” Vandermonde type, i.e.,

$$(A)_{j,\nu_i=1}^{s} = (jh)^{\nu_i}, \qquad \nu_i \in A, \quad s = |\mathcal{A}|.$$

- The condition number depends on α!

Solving the Vandermonde system

The resulting linear system is of "real" Vandermonde type, i.e.,

$$(A)_{j,\nu_i=1}^{s} = (jh)^{\nu_i}, \qquad \nu_i \in A, \quad s = |\mathcal{A}|.$$

- The condition number depends on α!
- If $\alpha = {}^{1}/_{M}$ for some integer M then we can rewrite the system in the "integer" Vandermonde form, thus *mildly* ill-conditioned,

Solving the Vandermonde system

The resulting linear system is of "real" Vandermonde type, i.e.,

$$(A)^{s}_{j,\nu_i=1} = (jh)^{\nu_i}, \qquad \nu_i \in A, \quad s = |\mathcal{A}|.$$

- The condition number depends on α!
- If $\alpha = {}^{1}/_{M}$ for some integer M then we can rewrite the system in the "integer" Vandermonde form, thus *mildly* ill-conditioned,
- If $\alpha = {}^{1}/_{M} - \epsilon$ and $p \geq 2$, then $\mathcal{A}$ will contain 1 and $M\alpha = 1 - M\epsilon$, hence the matrix will have two almost identical columns, thus a *bad* ill-conditioning.

Solving the Vandermonde system

The resulting linear system is of "real" Vandermonde type, i.e.,

$$(A)_{j,\nu_i=1}^{s} = (jh)^{\nu_i}, \qquad \nu_i \in A, \quad s = |\mathcal{A}|.$$

- The condition number depends on α!
- If $\alpha = {}^1\!/_M$ for some integer M then we can rewrite the system in the "integer" Vandermonde form, thus *mildly* ill-conditioned,
- If $\alpha = {}^1\!/_M - \epsilon$ and $p \geq 2$, then $\mathcal{A}$ will contain 1 and $M\alpha = 1 - M\epsilon$, hence the matrix will have two almost identical columns, thus a *bad* ill-conditioning.
- The right-hand side

$$\frac{1}{\Gamma(\alpha)} \int_0^{n\tau} (n\tau - \xi)^{\alpha-1} \chi^{\nu} \,\mathrm{d}\chi - \tau^{\alpha} \sum_{j=0}^{n} \omega_{n-j} (jh)^{\nu}$$

can suffer from cancellation of digits!

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 $y^{(0)}, \dots, y^{(s)}$, $s+1$, $s = |\mathcal{A}|$.

Computing the starting values

To initialize the computation we need the values $y^{(0)}, \dots, y^{(s)}$, $s+1$, $s = |\mathcal{A}|$.

- 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 $y^{(0)}, \ldots, y^{(s)}$, $s+1$, $s = |\mathcal{A}|$.

- 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 $y^{(0)}, \ldots, y^{(s)}$, $s+1$, $s = |\mathcal{A}|$.

- 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

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

- 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 $y^{(0)}, \ldots, y^{(s)}$, $s+1$, $s = |\mathcal{A}|$.

- 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

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

- 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

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

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

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

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

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

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

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

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

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

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 step 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 step 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

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

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

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

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

We can use FFT!

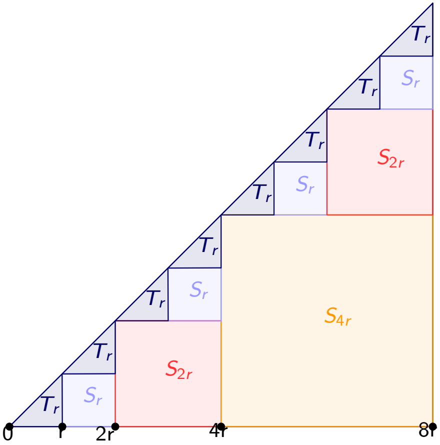
If we call $S_{2r}(n,0,2r-1) = \sum_{j=0}^{2r-1} c_{n-j} f_j$, and $S_r(n,2r,3r-1) = \sum_{j=2r}^{3r-1} c_{n-j} f_j$ the set of partial sums of lengths $2r$ and r we can evaluate them with FFT in $O(4r\log_2(4r))$ and $O(2r\log_2(2r))$ respectively.

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

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

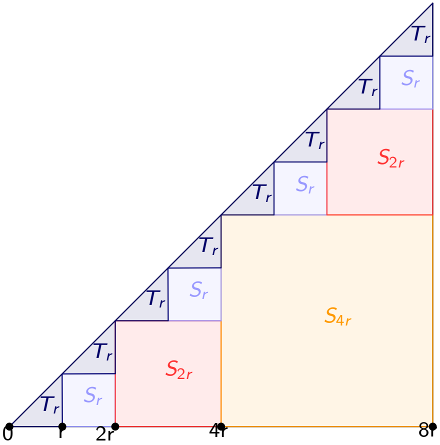
$$y^{(n)} = \phi_n + \sum_{j=0}^{2r-1} c_{n-j} f_j + \sum_{j=2r}^{3r-1} c_{n-j} f_j + \sum_{j=3r}^{n} c_{n-j} f_j, \quad n \in \{3r, 3r+1, \ldots, 4r-1\},$$

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)



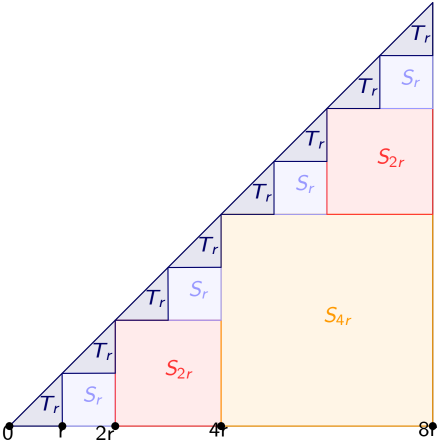
- We can iterate the process for the $4r$ approximations in the interval $n \in \{4r, \ldots, 8r-1\}$, together with the partial sums $S_{4r}(n, 0, 4r-1)$, $S_{2r}(n, 4r, 6r-1)$, $S_r(n, 6r, 7r-1)$ that can be evaluated in $O(8r\log_2(8r))$, $O(4r\log_2(4r))$ and $O(2r\log_2(2r))$ respectively,
- At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

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

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

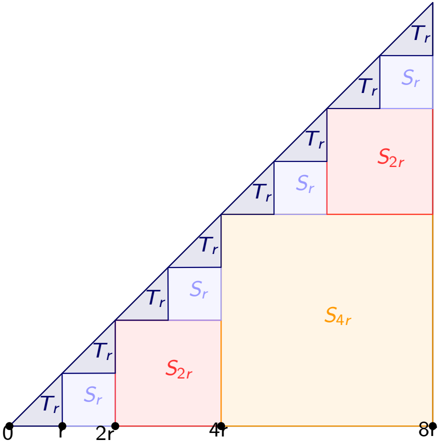
$$\ell = 0, 1, 2, \ldots.$$

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



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

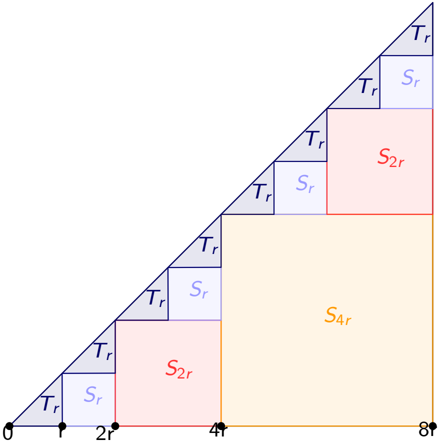
The FFT trick (Hairer, Lubich, and Schlichte 1985)



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

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

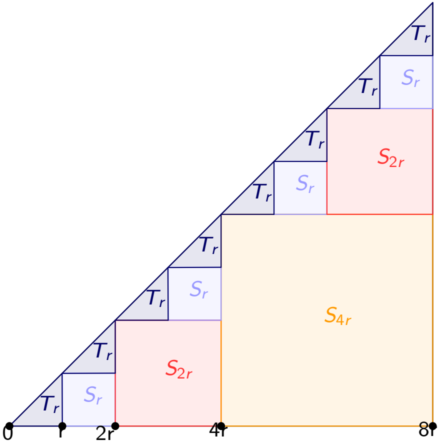
The FFT trick (Hairer, Lubich, and Schlichte 1985)



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)

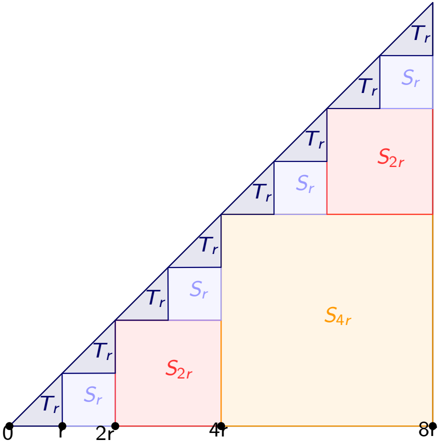


To determine the whole cost we just have to sum the various components

- 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)



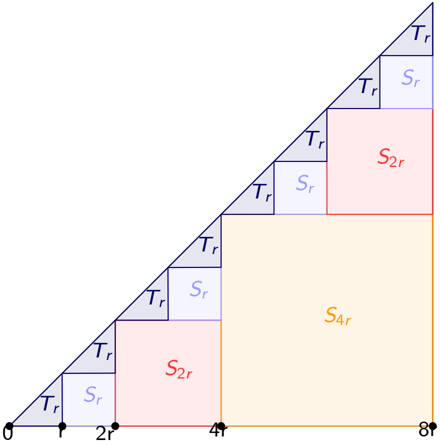
To determine the whole cost we just have to sum the various components

- 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

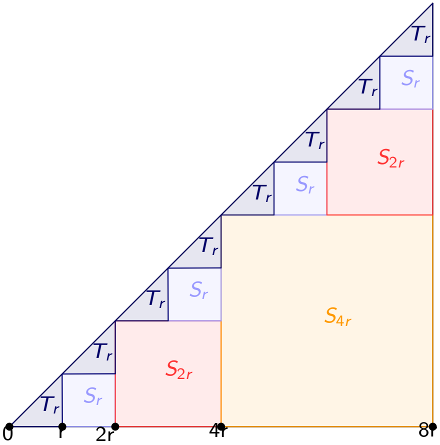
- Assume that $N = 2^{n_t}$
- $O(N \log_2 N)$ for S_{4r},

$+$ $O(N/2 \log_2 N/2)$ 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)

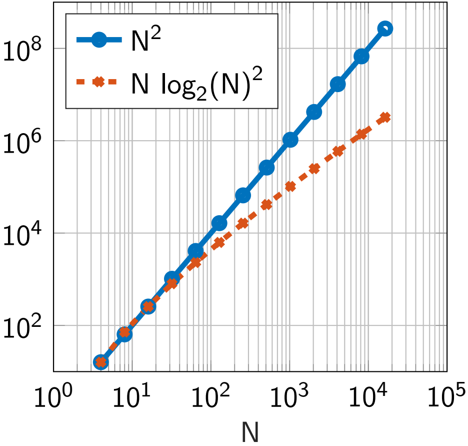


To determine the whole cost we just have to sum the various components

- 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.

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

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.

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

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

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

Short-memory principle (Ford and Simpson 2001)

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

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

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

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

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.

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

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

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

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

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

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

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

- 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?

Conclusions

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.

Conclusions

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.

Things we didn't pack...

There have been some effort in devising methods of Runge-Kutta type (Fischer 2019), collocation methods, and exponential type integrators.

Bibliography I

- Burnett, B. et al. (2021). "Performance Evaluation of Mixed-Precision Runge-Kutta Methods". In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, pp. 1–6.
- Cameron, R. F. and S. McKee (July 1985). "The Analysis of Product Integration Methods for Abel's Equation using Discrete Fractional Differentiation". In: *IMA Journal of Numerical Analysis* 5.3, pp. 339–353. ISSN: 0272-4979. DOI: 10.1093/imanum/5.3.339. eprint: https://academic.oup.com/imajna/article-pdf/5/3/339/2612709/5-3-339.pdf. URL: https://doi.org/10.1093/imanum/5.3.339.
- Caputo, M. (2008). "Linear models of dissipation whose Q is almost frequency independent. II". In: *Fract. Calc. Appl. Anal.* 11.1. Reprinted from Geophys. J. R. Astr. Soc. **1**3 (1967), no. 5, 529–539, pp. 4–14. ISSN: 1311-0454.
- Diethelm, K. (1997). "An algorithm for the numerical solution of differential equations of fractional order". In: *Electron. Trans. Numer. Anal.* 5.Mar. Pp. 1–6.

Bibliography II

- Diethelm, K., N. J. Ford, and A. D. Freed (2002). "A predictor-corrector approach for the numerical solution of fractional differential equations". In: vol. 29. 1-4. Fractional order calculus and its applications, pp. 3–22. DOI: 10.1023/A:1016592219341. URL: https://doi.org/10.1023/A:1016592219341.
- — (2004). "Detailed error analysis for a fractional Adams method". In: *Numer. Algorithms* 36.1, pp. 31–52. ISSN: 1017-1398. DOI: 10.1023/B:NUMA.0000027736.85078.be. URL: https://doi.org/10.1023/B:NUMA.0000027736.85078.be.
- Dixon, J. (1985). "On the order of the error in discretization methods for weakly singular second kind Volterra integral equations with nonsmooth solutions". In: *BIT* 25.4, pp. 624–634. ISSN: 0006-3835. DOI: 10.1007/BF01936141. URL: https://doi.org/10.1007/BF01936141.
- Fischer, M. (2019). "Fast and parallel Runge-Kutta approximation of fractional evolution equations". In: *SIAM J. Sci. Comput.* 41.2, A927–A947. ISSN: 1064-8275. DOI: 10.1137/18M1175616. URL: https://doi.org/10.1137/18M1175616.

Bibliography III

- 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 IV

Henrici, P. (1979). “Fast Fourier methods in computational complex analysis”. In: *SIAM Rev.* 21.4, pp. 481–527. ISSN: 0036-1445. DOI: 10.1137/1021093. URL: https://doi.org/10.1137/1021093.

Lubich, C. (1986a). “A stability analysis of convolution quadratures for Abel-Volterra integral equations”. In: *IMA J. Numer. Anal.* 6.1, pp. 87–101. ISSN: 0272-4979. DOI: 10.1093/imanum/6.1.87. URL: https://doi.org/10.1093/imanum/6.1.87.

— (1986b). “Discretized fractional calculus”. In: *SIAM J. Math. Anal.* 17.3, pp. 704–719. ISSN: 0036-1410. DOI: 10.1137/0517050. URL: https://doi.org/10.1137/0517050.

Young, A. (1954). “The application of approximate product integration to the numerical solution of integral equations”. In: *Proc. Roy. Soc. London Ser. A* 224, pp. 561–573. ISSN: 0962-8444. DOI: 10.1098/rspa.1954.0180. URL: https://doi.org/10.1098/rspa.1954.0180.