Numerical methods of the fractional calculus

Some notation

Notice: backwards numbering is convenient.

The G1 algorithm

Recall the Grunwald-Letnikov definition:

\[ D^\alpha_t f(x) \approx \left( \begin{array}{c} x-t \cr N \end{array} \right) \sum_{j=0}^{N} \frac{\Gamma(j+1)}{\Gamma(-\alpha)} (x-t)^{j} \left( x-t \frac{N}{N} \right) \]

Omitting the limit gives the simplest approximation:

\[ D^\alpha_t f(x) \approx \left( \begin{array}{c} x-t \cr N \end{array} \right) \sum_{j=0}^{N} \frac{\Gamma(j+1)}{\Gamma(-\alpha)} (x-t)^{j} \left( x-t \frac{N}{N} \right) \]

For \( \alpha = 0 \) we have:

\[ D^\alpha_t f(x) \approx x^{-Nt} \sum_{j=0}^{N} \frac{\Gamma(j+1)}{\Gamma(-\alpha)} x^{j} \left( x-t \frac{N}{N} \right) \]

Computation of binomial coefficients

For the implementation we need to compute

\[ w_k^{(0)} = \binom{-1}{k} \frac{1}{k}, \quad k = 0, 1, 2, \ldots \]

The recurrence relationships can be used:

\[ w_k^{(n+1)} = \frac{1}{k+1} w_k^{(n)} + \frac{1}{k+1} w_{k+1}^{(n)}, \quad k = 1, 2, 3, \ldots \]

Computation of binomial coefficients

Using FFT (fast Fourier transform)

The binomial coefficients can be considered as the coefficients of the power series expansion

\[ (1 - 2z)^k = \sum_{n=0}^{\infty} \binom{k}{n} z^n = \sum_{n=0}^{\infty} w_k^{(n)} z^n \]

Taking \( z = e^{-i\omega} \) we obtain

\[ (1 - e^{-i\omega})^k = \sum_{n=0}^{\infty} w_k^{(n)} e^{-i\omega n} \]
Computation of binomial coefficients
Using FFT (fast Fourier transform)

We have (from the previous slide):
\[
(1 - e^{-\lambda t})^n = \sum_{k=0}^{\infty} \binom{n}{k} e^{-k \lambda t}
\]

Therefore, the coefficients \( \binom{n}{k} \) can be considered as Fourier transforms:
\[
\binom{n}{k} = \frac{1}{2\pi} \int_{0}^{2\pi} f_\alpha(\phi) e^{ik\lambda t} d\phi, \quad f_\alpha(\phi) = (1 - e^{-\lambda t}).
\]

The fast Fourier transform (FFT) can be used here!

Using G1 for computations

Notice: infinities at t=0 are removed to make the rest of the picture visible.

Higher-order approximations

We have seen the first-order approximation:
\[
\partial^\alpha x f(x) \approx \lim_{h \to 0} \frac{1}{h} \sum_{k=1}^{n} \binom{n}{k} f(x - kh).
\]

The weights \( \binom{n}{k} \) (\( k = 0, 1, 2, \ldots, n \)) assigned to the values \( f(x - kh) \) are the first \( n+1 \) coefficients of the Taylor series expansion of the function
\[
\binom{n}{k} (x) = (1 - z)^n = \left( \binom{n}{k}(x) \right)^{\alpha}
\]
Higher-order approximations

Christian Lubich's formulas: $d^p_t(x) = x^{p-1}$

Expand these functions in Taylor series and use the coefficients as weights in the formula:

$0 D^\alpha_t f(t) \approx \sum \frac{f^{(i)}(0)}{i!} h^i f(t - i h)$

The G2 algorithm

Oldham and Spanier (1974) observed that the approximations give faster convergence, and suggested “fractional central differences”

$0 D^q_t f(x) = \lim_{h \to 0} \frac{1}{h} \left( f(x) - \sum_{k=0}^{N-1} f(x - k h) \frac{x-k}{N} \right)$

This formula uses the function values other than at nodes, so we have to interpolate (Langrange three point interpolation):

$0 D^q_t f(x) = \frac{x^{q-1}}{\Gamma(q)} \sum_{k=0}^{N-1} \frac{f(x - k h)}{(x-k)^q} \left( x-k \right)$

The R1 algorithm

G1 and G2 are based on fractional differences. R1 and R2 are based on approximation of integration.

Take $q < 0$ (fractional integration):

$0 D^q_t f(x) = \frac{f(x)}{\Gamma(-q)} \sum_{j=0}^{2} \frac{f(x-j h)}{(x-j)^{-q}} \left( x-j \right)$

Using the approximation

$\int_{x}^{x+\Delta x} f(y) dy = \frac{f(x) + f(x+\Delta x)}{2} \Delta x + \frac{f(x) - f(x+\Delta x)}{2} \Delta x^{1-q}$

we obtain the R1 algorithm:

$0 D^q_t f(x) = \frac{x^{q-1}}{\Gamma(1-q)} \sum_{j=0}^{2} \frac{f(x-j h)}{(x-j)^{1-q}} \left( x-j \right)$

The G2 algorithm

Taking lower terminal $a = 0$ we have

$a D^q_t f(x) = \lim_{h \to 0} \frac{1}{h} \left( f(x) - \sum_{k=0}^{N-1} f(x - k h) \frac{x-k}{N} \right)$

This formula uses the function values other than at nodes, so we have to interpolate (Langrange three point interpolation):

This gives the G2 algorithm:

$a D^q_t f(x) = \frac{x^{q-1}}{\Gamma(q)} \sum_{j=0}^{2} \frac{f(x-j h)}{(x-j)^q} \left( x-j \right)$

The R1 algorithm

In the previous slide we have used the piecewise constant approximation of the function using the function values in the middle of the subintervals:
The R2 algorithm

A better function approximation can be achieved using piecewise linear continuous function:

\[ f(x) = \begin{cases} \frac{x^2}{x^2 + 1} & \text{if } x > 0 \\ \frac{x^2}{x^2 + 1} & \text{if } x < 0 \end{cases} \]

The L1 algorithm

Now let us approximate each term using

\[ \int_{a}^{b} x^q f(x) \, dx = \left[ \frac{x^{q+1}}{q+1} f(x) \right]_{a}^{b} + \frac{q}{q+1} \int_{a}^{b} x^{q} f(x) \, dx \]

This leads to the L1 algorithm:

\[ a \frac{D^q f(x)}{D x^q} = \frac{x^{q-1}}{\Gamma(q+q)} \left[ \frac{1}{x} f(x) + \sum_{n=2}^{N} \frac{1}{n!} \frac{f^{(n)}(x)}{x^n} \right] \]

The L2 algorithm

Similarly, taking \( 1 \leq q < 2 \) we can write

\[ a \frac{D^q f(x)}{D x^q} = \frac{x^{q-1}}{\Gamma(q+q)} \left[ \frac{1}{x} f(x) + \sum_{n=2}^{N} \frac{1}{n!} \frac{f^{(n)}(x)}{x^n} \right] \]

We need some approximations for the first and second order derivatives here.

The R2 algorithm

Considering the piecewise linear approximation we have:

\[ \int_{a}^{b} x^q f(x) \, dx = \left[ \frac{x^{q+1}}{q+1} f(x) \right]_{a}^{b} + \frac{q}{q+1} \int_{a}^{b} x^{q-1} f(x) \, dx \]

and this gives the R2 algorithm:

\[ a \frac{D^q f(x)}{D x^q} = \frac{x^{q-1}}{\Gamma(q+q)} \left[ \frac{1}{x} f(x) + \sum_{n=2}^{N} \frac{1}{n!} \frac{f^{(n)}(x)}{x^n} \right] \]

Notice: both R1 and R2 algorithms allow consideration of non-equidistant discretization nodes (derive the formulas)!

The L1 algorithm

Now let us approximate each term using

\[ \int_{a}^{b} x^q f(x) \, dx = \left[ \frac{x^{q+1}}{q+1} f(x) \right]_{a}^{b} + \frac{q}{q+1} \int_{a}^{b} x^{q} f(x) \, dx \]

This leads to the L1 algorithm:

\[ a \frac{D^q f(x)}{D x^q} = \frac{x^{q-1}}{\Gamma(q+q)} \left[ \frac{1}{x} f(x) + \sum_{n=2}^{N} \frac{1}{n!} \frac{f^{(n)}(x)}{x^n} \right] \]

The L2 algorithm

Similarly, taking \( 1 \leq q < 2 \) we can write

\[ a \frac{D^q f(x)}{D x^q} = \frac{x^{q-1}}{\Gamma(q+q)} \left[ \frac{1}{x} f(x) + \sum_{n=2}^{N} \frac{1}{n!} \frac{f^{(n)}(x)}{x^n} \right] \]

We need some approximations for the first and second order derivatives here.

The R2 algorithm

Considering the piecewise linear approximation we have:

\[ \int_{a}^{b} x^q f(x) \, dx = \left[ \frac{x^{q+1}}{q+1} f(x) \right]_{a}^{b} + \frac{q}{q+1} \int_{a}^{b} x^{q} f(x) \, dx \]

and this gives the R2 algorithm:

\[ a \frac{D^q f(x)}{D x^q} = \frac{x^{q-1}}{\Gamma(q+q)} \left[ \frac{1}{x} f(x) + \sum_{n=2}^{N} \frac{1}{n!} \frac{f^{(n)}(x)}{x^n} \right] \]

Notice: both R1 and R2 algorithms allow consideration of non-equidistant discretization nodes (derive the formulas)!

The L1 algorithm

Now let us approximate each term using

\[ \int_{a}^{b} x^q f(x) \, dx = \left[ \frac{x^{q+1}}{q+1} f(x) \right]_{a}^{b} + \frac{q}{q+1} \int_{a}^{b} x^{q} f(x) \, dx \]

This leads to the L1 algorithm:

\[ a \frac{D^q f(x)}{D x^q} = \frac{x^{q-1}}{\Gamma(q+q)} \left[ \frac{1}{x} f(x) + \sum_{n=2}^{N} \frac{1}{n!} \frac{f^{(n)}(x)}{x^n} \right] \]

The L2 algorithm

Similarly, taking \( 1 \leq q < 2 \) we can write

\[ a \frac{D^q f(x)}{D x^q} = \frac{x^{q-1}}{\Gamma(q+q)} \left[ \frac{1}{x} f(x) + \sum_{n=2}^{N} \frac{1}{n!} \frac{f^{(n)}(x)}{x^n} \right] \]

We need some approximations for the first and second order derivatives here.
The D algorithm
Kai Diethelm suggested (1997) numerical evaluation of fractional derivatives using quadrature formulas for finite-part integrals. First, use change of variables to transform the interval \([0, T]\) to \([0, 1]\) and introduce an equidistant grid with nodes \(t_j = j/m\). Then
\[
\frac{d^q}{dx^q}f(0) = \frac{1}{\Gamma(-q)} \int_0^T f(t) t^{-q}dt = \frac{\Gamma(q)}{\Gamma(-q)} \int_0^T \frac{(t^q - t_j^q)df}{t_j^q}
\]
and the D algorithm is:
\[
\frac{d^q}{dx^q}f(0) \approx \frac{\Gamma(q)}{\Gamma(-q)} \sum_{j \in \mathbb{Z}} w_j f(t_j)
\]

A general framework
All considered algorithms (G1, G2, R1, R2, L1, L2, D) can be written in the same form as
\[
\frac{d^q}{dx^q}f(x) = \frac{\Gamma(q)}{\Gamma(-q)} \sum_{j = -1}^N w_j(q) f_j
\]

G1 in brief:
- Typical \(w_j(q)\) \(\frac{\Gamma(q)\Gamma(q-j+1)}{\Gamma(-q)}\)
- Range of typicality \(0 \leq j \leq N-1\)
- Values of \(j\) \(-1, N\) for which \(w_j(q) = 0\)
- Atypical values none

G2 in brief:
- Typical \(w_j(q)\) \(\frac{\Gamma(q)\Gamma(q-j+1)}{\Gamma(-q)}\)
- Range of typicality \(0 \leq j \leq N-1\)
- Values of \(j\) \(-1, N\) for which \(w_j(q) = 0\)
- Atypical values \(w_j(q) = \frac{q^2 + 2q}{8\Gamma(-q)}\)

R1 in brief:
- Typical \(w_j(q)\) \(\frac{1}{\Gamma(q)}\)
- Range of typicality \(1 \leq j \leq N-1\)
- Values of \(j\) \(-1\) for which \(w_j(q) = 0\)
- Atypical values \(w_j(q) = \frac{1}{2\Gamma(-q)}\)

R2 in brief:
- Typical \(w_j(q)\) \(\frac{1}{\Gamma(q)}\)
- Range of typicality \(1 \leq j \leq N-1\)
- Values of \(j\) \(-1\) for which \(w_j(q) = 0\)
- Atypical values \(w_j(q) = \frac{1}{2\Gamma(-q)}\)
Matrix approach to discretization of fractional integrals and derivatives

**L1 in brief:**
- Typical \( w_1(p) \): \( \frac{1}{1 \times p^k - (1 - 1)^k} \)
- Range of typicality: \( 1 \leq j \leq N - 1 \)
- Values of \( j \) for which \( w_1(p) = 0 \): none
- Atypical values: \( w_1(p) = 1/(1 - q) \)
  - \( \mu_1(q) = 1/(1 - q) \)
  - \( \gamma(q) = (1 - q)/(1 - q) \)

**L2 in brief:**
- Typical \( w_2(p) \): \( \frac{1}{1 \times p^k - (1 - 1)^k} \)
- Range of typicality: \( 1 \leq j \leq N - 2 \)
- Values of \( j \) for which \( w_2(p) = 0 \): none
- Atypical values: \( w_2(p) = 1/(1 - q) \)
  - \( \mu_2(q) = 1/(1 - q) \)
  - \( \gamma(q) = (1 - q)/(1 - q) \)

**Triangular strip matrices (TSM)**
- **Lower TSM:**
  \[
  L_N = \begin{bmatrix}
  \mu_1 & 0 & 0 & \cdots & 0 \\
  \mu_2 & \mu_1 & 0 & \cdots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  \mu_{N-1} & \mu_{N-2} & \mu_{N-3} & \cdots & \mu_1
  \end{bmatrix}
  \]
- **Upper TSM:**
  \[
  U_N = \begin{bmatrix}
  \mu_1 & 0 & 0 & \cdots & 0 \\
  0 & \mu_1 & 0 & \cdots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & \cdots & \mu_1
  \end{bmatrix}
  \]

If two TSMs are of the same type, then: \( CD = DC \).

**Truncation operation**
- \( g(z) = \sum_{n=0}^{\infty} \omega_n z^n \rightarrow \text{trunc}_N(g(z)) = \sum_{n=0}^{N} \omega_n z^n = g_N(z) \)

Function \( g(z) \) generates a sequence of lower TSMs:
- \( L_N, \quad N = 1, 2, \ldots \)
or upper TSMs:
- \( U_N, \quad N = 1, 2, \ldots \)

**Properties:**
- \( \text{trunc}_N(\gamma \lambda(z)) = \gamma \text{trunc}_N(\lambda(z)) \)
- \( \text{trunc}_N(\lambda(z) + \mu(z)) = \text{trunc}_N(\lambda(z)) + \text{trunc}_N(\mu(z)) \)
- \( \text{trunc}_N(\lambda(2)\mu(z)) = \text{trunc}_N(\lambda(z)) \text{trunc}_N(\mu(z)) \)

**Operations with TSMs**
- \( A_N = \sum_{n=0}^{N} \lambda_n(E_1^n) = \lambda(E_1^N) \), \( B_N = \sum_{n=0}^{N} \mu_n(E_1^n) = \mu(E_1^N) \)
- \( \lambda_N(z) = \text{trunc}_N(\lambda(z)) \), \( \mu_N(z) = \text{trunc}_N(\mu(z)) \)

**Addition and subtraction:**
- \( A_N + B_N \rightarrow \text{trunc}_N(\lambda(z) + \mu(z)) \)

**Multiplication by a constant:**
- \( \gamma A_N \rightarrow \gamma \text{trunc}_N(\lambda(z)) \)

**Product of TSMs:**
- \( A_N B_N \rightarrow \text{trunc}_N(\lambda(z)) \mu(z) \)

**Matrix inversion:**
- \( (A_N)^{-1} \rightarrow \text{trunc}_N(\lambda^{-1}(z)) \)
Left-sided R-L derivatives

\[ L^k \frac{f(t)}{t^k} + \sum_{j \geq 0} \left( \begin{array}{c} j-k \\theta \end{array} \right) \frac{f_t(t)}{t^{j-k}} \quad k = 0, 1, \ldots \}

\[ k = 0, 1, \ldots \]

Example: Riesz kernel

\[ \frac{1}{\Gamma(1-\alpha) \sqrt{2\pi t^{1-\alpha}}} = 1, \quad (-1 < t < 1), \]

Exact solution:

\[ y(t) = \frac{1}{\sqrt{2\pi \tau}} \left( 1 - \epsilon^2 \right)^{1-\alpha/2}. \]

Numerical solution:

\[ \int_0^1 y(t) \, \frac{\sinh(t)}{\cosh(t)} \, dt = 1, \]

\[ (B_x(t) + C_x(t))(t) = F_x. \]

Example: Riesz kernel

\[ \alpha = 0.8 \]

Example (Caputo derivatives)

\[ y^{(n)}(t) + y(t) = 1, \quad y(0) = 0, \quad y'(0) = 0. \]

Exact solution:

\[ y(t) = e^{-t} \left( e^t \right)^{-\alpha}. \]

Numerical solution:

\[ (B_{xx} + C_{xx})(t) \int_0^t y(t) \, dt = S(t)F. \]

and from the initial conditions we have:

\[ y_0 = y_1 = 0. \]
Example (Caputo derivatives)

\[ \alpha = 1.8 \]

\[ \alpha = 1.8 \]

The “short memory” principle

\[ \dot{D}_\tau f(t) \approx \int_0^t \dot{D}_\tau f(\tau) \, D^n t^c, \quad (t > \alpha + L) \]

"Memory length" depends on required accuracy

The “short memory” principle

If \( f(t) \leq M \) for \( a \leq t \leq b \), then it can be shown that

\[ \Delta(t) = \int_a^t \dot{D}_\tau f(\tau) - \int_a^t \dot{D}_\tau f(\tau) \leq \frac{M}{(t-a)^\alpha} \quad (a+L \leq t \leq b) \]

Therefore,

\[ \Delta(t) \leq \epsilon, \quad (a+L \leq t \leq b), \quad \text{if} \quad L \geq \left( \frac{M}{(t-a)^\alpha} \right)^{1/\alpha} \]