A local truncation error estimation for a SubIval solver

M. SOWA*

Silesian University of Technology, 2a Akademicka St., 44-100 Gliwice, Poland

Abstract. The paper concerns an analysis for SubIval (the subinterval-based method for fractional derivative computations in initial value problems). A time step size adaptive solver is discussed, for which the formula of a local truncation error is derived. A general form for a system of linear equations is given for the considered class of problems (for which the analysis is performed in the paper). Two circuit examples are introduced to display the usefulness of the SubIval solver. For the examples that have been chosen it is possible to obtain referential solutions through completely different methods. The results obtained through the numerical solver are compared with evaluations of the referential solutions. The error estimation results obtained for the time steps of the SubIval solver are compared with the actual errors, being the differences between the numerical solutions and the referential solutions. The paper also contains a comparison of the accuracy of results obtained through the SubIval solver with the accuracies of other solvers.

Key words: fractional calculus, numerical method, adaptive step size, local truncation error, circuit analysis.

1. Introduction

The first major developments in the field of fractional calculus date back to the 19th century [1, 2]. Since then, many mathematical considerations have been carried out. Apart from it being a challenging mathematical topic, it has also become popular of late because of its many applications, e.g. in heat conduction analyses [3–5], the design and implementation of fractional order filters [6, 7], in the design and analysis of fractional order controllers [8, 9] (including coupled analyses of control of synchronous machines [10, 11]), electric and magnetic field analyses (when materials of complex properties are considered [12–14]), inverse kinematics [15], continuum mechanics [16], viscoelasticity [17, 18].

Fractional calculus is also applied in circuit analyses (computational examples of this field have been applied in later parts of this paper). In circuit analyses, fractional capacitors are applied in modeling supercapacitors [19–22] and fractional coils have shown to accurately resemble the responses of some coils with ferromagnetic cores [23, 24].

Elements of fractional calculus include definitions of fractional derivatives and integrals (or generally differintegrals) with respect to both time (more commonly studied) and space [25]. The study presented in this paper concerns the Caputo time derivative of order \( \alpha \in [0,1] \) [26]:

\[
D_t^\alpha x(t) = \frac{1}{\Gamma(1-\alpha)} \int_{t_0}^{t} \frac{x(t-\tau)}{(t-\tau)^\alpha} d\tau.
\] (1)

The Caputo definition is equivalent to the oldest definition (which is that of Riemann and Liouville [27]) if zero initial conditions are imposed [28].

The increasing interest in fractional calculus can also be attributed to the fact that many studies are performed concerning the methods allowing to solve problems with fractional derivatives and integrals.

The possibility of solving a problem with fractional derivatives by means of a selected method depends on the form of the problem. For linear problems that can be expressed in the form of a system of fractional state equations, for some known source time functions one can apply analytical solutions based on the Mittag-Leffler function [29, 30]. For other problems one can apply semi-analytical methods (the general form of the solution is assumed as a series of terms whose coefficients are computed), which are especially well described and tested for problems which do not require many equations (although they may contain nonlinear dependencies). These are e.g. the Adomian decomposition method [31, 32] and the differential transform method [33]. For a more general approach, which can deal with the widest variety of problems, one can apply time stepping solvers, using numerical methods. Some of the most common are fractional linear multistep methods (backward difference methods, explained in general in [34]), product integration (PI) rule methods [35], collocation methods [36, 37] and methods using the Grünwald-Letnikov definition of the fractional derivative [38].

The study in this paper concerns the application of a method called SubIval (the subinterval-based numerical method for fractional derivative computations in initial value problems, first discussed in [39]). It is a numerical method that can be applied in time stepping solvers.

2. Application of SubIval

A special notation relating to intervals is used for the differ-integration:

\[
d^\alpha_\Xi x(t) = D_t^\alpha x(t),
\] (2)

where \( \Xi \in [t_a, t_b] \). SubIval relies on a partition of the differ-integration interval \( \Xi_{tot} = [t_0, t_{now}] \) into subintervals (with...
where \( t_0 \) being the initial time instance and \( t_{\text{now}} \) being the computed time instance:

\[
d^0_{\Xi_M} x(t) = \sum_{s=1}^{M} d^0_{\Xi_s} x(t). \tag{3}
\]

For each subinterval \( \Xi_s = [t_{s\text{start}}, t_{s\text{end}}] \) if \( s > 1 \) then \( t_{s\text{start}} = t_{s-1\text{end}} \). In the first subinterval \( t_1\text{start} = t_0 \), while in the last one \( t_{M\text{end}} = t_{\text{now}} \).

In each of these subintervals the solution is approximated by means of a polynomial:

\[
d^p_{\Xi_s} x(t) = \sum_{s=1}^{M} d^p_{\Xi_s} x_s(t). \tag{4}
\]

Each approximation \( x_s(t) \) is built upon nodes denoted by \( t_{s,1}, t_{s,2}, \ldots t_{s,n_s} \) (the order of each polynomial \( x_s(t) \) can, therefore, be given by \( q_s = n_s - 1 \)). These are covered by subintervals denoted by \( \Theta_s = [t_{s,1}, t_{s,n_s}] \), where for each subinterval pair \( \Theta_s \supseteq \Xi_s \). The subintervals are formed and modified according to a subinterval dynamics algorithm described with great detail (with examples) in [40].

Each approximation can be given by:

\[
\tilde{x}_s(t) = \sum_{j=1}^{n_s} x_{s,j} L_{s,j}(t) \tag{5}
\]

where \( L_{s,j} \) denote Lagrange basis polynomials:

\[
L_{s,j}(t_{s,k}) = \begin{cases} 0, & \text{if } k \neq j, \\ 1, & \text{if } k = j, \end{cases} \tag{6}
\]

while \( x_{s,j} \) are the values of the considered variable computed at their respective \( t_{s,j} \) nodes.

SubVal can be used in a typical time stepping solver, in which for each subsequently computed time instance \( t = t_{\text{now}} \) the only values treated as unknowns are the variables selected at this time instance. Values of previous time instances are not modified anymore and treated as known values.

Through the subinterval dynamics and local polynomial differintegrations (applying analytical formulae for monomials [42, 43]) SubVal leads to a convenient implicit formula:

\[
d^0_{\Xi_M} x(t) \approx a x_{M,n_M} + b, \tag{7}
\]

where \( a \) and \( b \) are computed for each variable separately every time when a new time instance \( t_{\text{now}} \) is selected. \( a \) results from the differintegrations of the Lagrange basis polynomial being the multiplier of the still unknown value \( x_{M,n_M} \):

\[
a = d^0_{\Xi_M} L_{M,n_M}(t). \tag{8}
\]

One can distinguish two terms in \( b \):

\[
b = b_M + b_{\text{prev}}, \tag{9}
\]

where \( b_M \) depends on the differintegrals in the \( \Xi_M \) subinterval:

\[
b_M = d^0_{\Xi_M} \sum_{j=1}^{n_M-1} x_{M,j} L_{M,j}(t), \tag{10}
\]

while \( b_{\text{prev}} \) depends on the differintegrals in the remaining subintervals with indices \( s = 1, 2, \ldots M - 1 \):

\[
b_{\text{prev}} = \sum_{s=1}^{M-1} d^0_{\Xi_s} \sum_{j=1}^{n_s} x_{s,j} L_{s,j}(t). \tag{11}
\]

3. Approximation of the local truncation error

One of the properties of SubVal and its subinterval dynamics is that they make only the \( M \)-th subinterval (being the rightmost subinterval on the time axis) cover the node with unknown variables. Also, in the typical time stepping solver it is designed for, one only controls the time step size between \( t_{\text{now}} \) and the previous node (this step size is further on denoted by \( \Delta t_{\text{now}} \)).

Assuming the solution \( x(t) \) is infinitely differentiable in \( \Theta_M \) then it can be expressed by means of a power series:

\[
x(t_{\text{loc}}) = c_0 + \sum_{k=1}^{\infty} c_k t^{\lambda_k}_{\text{loc}}, \tag{12}
\]

where \( t_{\text{loc}} = t - t_{M,1} \).

The error of the fractional derivative approximation resulting from SubVal, in the subinterval \( \Xi_M \), is given by the formula:

\[
e_{\text{True}} M = d^0_{\Xi_M} x(t_{\text{loc}}) - d^0_{\Xi_M} \tilde{x}_M(t_{\text{loc}}). \tag{13}
\]

One of the purposes of estimating the local truncation error is to be able to control the error of the numerical computations. If one can only control the error by changing \( \Delta t_{\text{now}} \) then \( e_{\text{True}} M \) must be given in terms of this variable.

Because the considerations concern only the subinterval \( \Xi_M \) and nodes covered by \( \Theta_M \) a simpler notation: \( t_j = t_{M,j} \) will be used in this section along with the number of nodes \( n = n_M \).

The step sizes of \( \Theta_M \) are denoted by:

\[
\Delta t_j = t_j - t_{j-1}, \tag{14}
\]

with their indices starting from \( j = 2 \). Hence, the time instances can be given by:

\[
t_j = \left\{ \begin{array}{ll} 0, & \text{if } j = 1, \\ \sum_{k=2}^{j} \Delta t_k, & \text{if } j = 2, \ldots n. \end{array} \right. \tag{15}
\]

Each subsequent step size will be modified by multiplying the previous one by a computed coefficient (denoted by \( \eta_j \)):

\[
\Delta t_j = \eta_j \Delta t_j-1, \tag{16}
\]

with the indices of \( \eta_j \) starting from \( j = 3 \). By introducing \( \mu_j = \eta_j^{-1} \) the step sizes can be given by:

\[
\Delta t_j = \left\{ \begin{array}{ll} \prod_{r=j+1}^{n} \mu_r \Delta t_{\text{now}}, & \text{if } j = 2, \ldots n - 1, \\ \Delta t_{\text{now}}, & \text{if } j = n. \end{array} \right. \tag{17}
\]
Finally, the time instances are given by:

\[ t_j = \begin{cases} 
0, & \text{if } j = 0, \\
\Delta t_{\text{now}} \sum_{k=2}^{j} \prod_{r=k+1}^{n} \mu_r, & \text{if } j = 2, \ldots n-1, \\
\Delta t_{\text{now}} \left( 1 + \sum_{k=2}^{j} \prod_{r=k+1}^{n} \mu_r \right), & \text{if } j = n.
\]  

(18)

The following steps are involved in obtaining a formula for \( e_{\text{True M}} \):

- as it is done in a typical local truncation error estimation [41] the errors of the computed variable at the selected time instances are omitted and each \( x_j \) is substituted by the respective \( x(t_j) \),
- Lagrange basis polynomials are given in terms of dependencies on \( \Delta t_{\text{now}} \) and the \( \mu \) coefficients,
- \( x(t_{\text{loc}}) \) is truncated into a sum of finite terms, where the remaining polynomial is at least one order higher than that of the approximation \( \tilde{x}_M(t_{\text{loc}}) \),
- the fractional derivative of \( \tilde{x}(t_{\text{loc}}) \) and that of the finite sum representation of \( x(t_{\text{loc}}) \) are computed (for the interval \( \mathcal{Z}_M \)) and the difference between them is obtained.

In the last step – the differintegration of both the exact solution and the approximating polynomial can be done by means of the already mentioned analytical formulæ for monomials (which SubIval uses). For a monomial \( e^k_{\text{loc}} \) with \( e \in \mathbb{R} \) and \( k \in \mathbb{Z}_+ \) the differintegration in the interval \( \mathcal{Z} = [t_{\text{loc}a}, t_{\text{loc}b}] \) results in [42,43]:

\[ \frac{\alpha^k e}{k!} e^k_{\text{loc}} = \frac{k e}{\Gamma(1-\alpha)} \frac{k-\alpha}{\Gamma(k-1+\alpha)} \left( B_{\text{num}}(k, 1-\alpha) - B_{\text{num}}(k, 1-\alpha) \right), \]

(19)

where

\[ B_{\mu}(k, 1-\alpha) = \frac{\Gamma(k) \Gamma(1-\alpha)}{\Gamma(k+1-\alpha)} \left( 1 - (1-\rho) \sum_{j=0}^{k-1} \prod_{i=0}^{j-1} (1-\alpha+i) \right). \]

(20)

In order to avoid human error – the tiresome derivations of the truncated form of \( e_{\text{True M}} \) can be done by means of a selected computer algebra system allowing for symbolic computation. The SymPy Python library [44] has been used for this purpose. Initially the symbolic form of the truncated \( e_{\text{True M}} \) has been studied for the polynomial orders \( q \) from 1 to 5 with \( \mathcal{Z}_M = \Theta_M \). The obtained formulæ have the general form:

\[ e_{\text{True M}} = \frac{\nu_q}{\Gamma(q+2-\alpha)} e^{q+1} \Delta t_{\text{now}}^{q+1-\alpha} + O(\Delta t_{\text{now}}^{q+2-\alpha}). \]

(21)

The \( \nu_q \) coefficients obtained for \( q = 1, 2 \) and 3 are given in Table 1 (for higher polynomial orders the expressions are very long and therefore not have been presented in the paper). The obtained formulæ suggest that the reduction of the step sizes leads to smaller errors as is the case when using BDF (backward differentiation formulæ) for the derivative order \( \alpha = 1 \).

Moreover, the above formula represents the BDF error for \( \alpha = 1 \).

In order to obtain an error estimation in a single step one must compare at least two approximations. A two stage approach is proposed in the study where the first one (called stage A) applies a polynomial of order \( q - 1 \) to obtain \( \tilde{x}_M(t) \), while the second stage (called stage B) applies a polynomial of order \( q \). Obviously, the theoretically more accurate results of the stage B will be selected in the final solution, however the error estimation:

\[ e_{\text{stages}} = d^2_{\text{2 M}} \tilde{x}_M B(t_{\text{loc}}) - d^2_{\text{2 M}} \tilde{x}_M A(t_{\text{loc}}) \]

(22)

will take into account a term lacking in stage A (the interval \( \mathcal{Z}_M A \) is \( \mathcal{Z}_M \) from stage A). In order to obtain a general formula for this term (and the remainder appearing as a consequence of terms not included in both stage approximations), like before, the SymPy library has been applied. The obtained general form is:

\[ e_{\text{stages}} = e^{q-1}_q \Delta t_{\text{now}}^{q-\alpha} + O(\Delta t_{\text{now}}^{q+1-\alpha}). \]

(23)

where a resemblance to \( e_{\text{True M}} \) for the polynomial order \( q - 1 \) is evident.

### 4. Step size adaptivity

For the considerations in this section an error approximation is assumed by considering only the first term of \( e_{\text{stages}} \):

\[ e = \frac{\nu_q - 1}{\Gamma(q+1-\alpha)} e^{q+1} \Delta t_{\text{now}}^{q+1-\alpha}. \]

(24)

Assuming \( e_{\text{ctrl}} \) represents an acceptable error then the step size for which \( e \) takes this value is given by:

\[ \Delta t_{\text{new}} = \sqrt{\frac{e_{\text{ctrl}}}{e}} e^{q+1} \Delta t_{\text{now}}^{q+1-\alpha}. \]

(25)

where for simplicity the \( c \) coefficient has been introduced:

\[ e = \frac{\nu_q - 1}{\Gamma(q+1-\alpha)} c. \]

(26)

The denominator of (25) can be obtained from the relation between the computed error estimation \( e \) and the step size \( \Delta t_{\text{now}} \) applied at that moment:

\[ e = \frac{\nu_q - 1}{\Gamma(q+1-\alpha)} \sqrt{c \Delta t_{\text{now}}^{q+1-\alpha}}. \]

(27)

which leads to the relation:

\[ \eta = \frac{\Delta t_{\text{new}}}{\Delta t_{\text{now}}} = e^{-\alpha} \sqrt{\frac{e_{\text{ctrl}}}{e}}. \]

(28)
meaning that an applied step size $\Delta t_{\text{next}}$ should be multiplied by $\eta$ in order for the error to tend to $e_{\text{ctrl}}$.

A few remarks can be made at this stage of the analysis:

- the obtained error estimation $e$ appears in the denominator, hence if the obtained value is 0 then $\eta$ can be set to 1, thus not modifying the step size,
- in some cases when a user sets $e_{\text{ctrl}}$ too low the SubVal solver could take a great amount of time to obtain the solution, hence, for extreme cases a minimum step size value $\Delta t_{\min}$ can be set,
- for very small error values, a rational approach could be not to greatly increase the step size (even for very small errors, as that might change), hence, a maximum step size $\Delta t_{\max}$ can also be set,
- not knowing the scale of a variable it is better to compute the errors as relative values (also, $e_{\text{ctrl}}$ would be given in percentages),
- for several variables under fractional derivatives (state variables) the smallest $\eta$ can be selected, following a pessimistic approach,
- the error estimation is performed after an approximation of the derivative (and the solution) had been obtained, the time step size modification is, hence, performed for the next step size; however, if the obtained error value is too large (exceeding a previously set parameter $e_{\text{max}}$) then the time step should be repeated with a new $\Delta t_{\text{next}}$ which leads to a new time instance $t_{\text{next}}$.

When a new time instance $t_{\text{next}}$ is selected then a new set of parameters $a$, $b$, and $b_M$ can be obtained for each variable in both stages. In stage $A$ one can apply $a$ and $b_M$ to obtain:

$$d_A = d_{\Xi, M, A}^a \bar{x}_{M, A}(t_{\text{loc}})$$

appearing in (22). However, in stage $B$ the coefficients $a$ and $b_M$ are obtained from a differintegration in a new interval $\Xi_M \neq \Xi_A$, hence an additional differintegration must be performed to obtain the term:

$$d_B = d_{\Xi, M, B}^a \bar{x}_{M, B}(t_{\text{loc}}).$$

In practice, the approximations of the derivative and the error estimation that follows can be performed using various strategies, which have been explained in Sec. 6.

5. Considered class of problems

The problems for which the local truncation error estimation is considered can be described by the following system of equations:

$$\begin{cases}
M_1 y(t) + M_2 x(t) = T v(t), \\
d_{\Xi}^a x(t) + M_{III} y(t) + M_{IV} x(t) = 0_{n_x},
\end{cases}$$

where

- $x(t)$ is the vector of state variables (its length is denoted by $n_x$),
- $y(t)$ is a vector of the remaining variables introduced to the solution (the length of the vector is denoted by $n_y$),
- $v(t)$ contains the source time functions (the vector length is given by $n_v$),
- $d_{\Xi}^a x(t)$ is a vector of fractional derivatives of the state variables:

$$d_{\Xi}^a x(t) = \left[ d_{\Xi}^a x_1(t) \ d_{\Xi}^a x_2(t) \ldots d_{\Xi}^a x_{n_x}(t) \right]^T$$

with $\alpha$ representing a vector of the derivative orders $\alpha_1, \alpha_2, \ldots, \alpha_{n_x}$; with the interval $\Xi = [t_0, t]$, $0_{n_x}$ is a vector of $n_x$ zeros,

- $M_1$ is an $n_y \times n_y$ matrix,
- $M_{II}$ is an $n_y \times n_x$ matrix,
- $M_{III}$ is an $n_x \times n_y$ matrix,
- $M_{IV}$ is an $n_x \times n_x$ matrix,
- $T$ is an $n_y \times n_v$ matrix.

Also, for further consideration, the vector:

$$w(t) = [ y(t) \ x(t) ]^T$$

is defined, containing all the variables placed in the solution. For each stage of the time stepping solver one can obtain the $a$ and $b$ coefficient pair. All of the $a$ coefficients can be placed in a vector with indices corresponding to those of the state variables:

$$a = [ a_1 \ a_2 \ldots \ a_{n_x} ]^T.$$ 

The same can be done for the $b$ coefficients:

$$b = [ b_1 \ b_2 \ldots \ b_{n_y} ]^T.$$ 

By applying the SubVal approximation (7) at the time instance $t = t_{\text{next}}$ one obtains the system of equations:

$$\begin{bmatrix}
M_1 & M_{II} \\
M_{III} & M_{IV} + \text{diag}(a)
\end{bmatrix} w = \begin{bmatrix} T v(t_{\text{next}}) \\
- b
\end{bmatrix},$$

with $w$ being the numerical solution for $w(t_{\text{next}})$.

6. Error computation strategies

The most obvious approach to the error estimation is to take the following steps:

(a) compute $a$ and $b$ for both stages, along with $b_M$ for stage A, while for stage B compute the additional coefficients:

$$a_M = d_{\Xi, M, A}^a L_{M, n_M}(t)$$

and

$$b_M = d_{\Xi, M, A}^a \sum_{j=1}^{n_M-1} x_{M,j} L_{M,j}(t),$$

(b) compute the solution of (36) at both stages (for a selected state variable $x$ the solution is further on denoted by $x_{\text{now}}$ for stage A and $x_{\text{now}}$ for stage B),

(c) obtain $d_A$ mentioned in (29) through:

$$d_A = a x_{\text{now}} + b_M,$$

and $d_B$ mentioned in (30) through:

$$d_B = a_M x_{\text{now}} + b_M A.$$
(d) finally, a relative difference between $d_A$ and $d_B$ is taken into account to evaluate the error estimation $e$ (this is done for each variable).

The above is further on referred to as the $d$ approach.

A slightly modified approach would be to compute step (a) but only obtain the solution for stage B. $d_A$ would be then computed through (39) using $x_{now \ B}$ instead of $x_{now \ A}$. $d_B$ is still computed using (40). This simplified approach is further on called the $d_s$ approach.

Another approach is tested in the study (being an extension to the $d$-based approach), where in addition to the differences between the variable approximations – selected variables from the solution vector $w$ are compared for both stages. This is further on called the $w$ approach.

Finally, an approach is tested, which also adds error computations of $w$, but for the derivative error computations – the component $d_A$ is obtained from the solution of stage B (like in the $d_s$ approach). This is further on called the $w_s$ approach.

7. Computational examples

Two circuit examples have been selected, which can be described by the system of equations given by (31). For the selected examples one can provide referential solutions that can be obtained through completely different methods (not involving the approximation of fractional derivatives).

7.1. Circuit in periodic steady-state. The first considered example is presented in Fig. 1. The problem has a relatively easily obtainable referential steady-state solution using a complex number approach. As it is a linear problem then for the source frequencies $f = 50, 150$ Hz the solutions can be obtained separately and then added using the superposition principle.

For a single frequency a complex number approach can be applied where one solves the system:

$$
\begin{bmatrix}
M_1 & M_{11} \\
M_{1 \ 1} & M_{IV} + \text{diag}(g)
\end{bmatrix} \begin{bmatrix}
w_1 \\
0_{n_w}
\end{bmatrix} = \begin{bmatrix}
T^w \\
0_{n_w}
\end{bmatrix},
$$

with

- $w$ being a vector of the complex number representations of the variables in $w(t)$ (for the considered frequency),
- $w$ is the complex number representation of the source vector for the considered frequency,
- $g$ results from the periodic steady-state differintegration and is given by:

$$
g = \begin{bmatrix}
\hat{\omega}_{1} \\
\hat{\omega}_{2} \\
\vdots \\
\hat{\omega}_{n_w}
\end{bmatrix}^T,
$$

where for the indices $i = 1, 2, \ldots, n_w$:

$$
\hat{\omega}_{i} = \exp \left( j \omega_{i} \frac{\pi}{2} \right).
$$

The options for the numerical solver are given in Table 2.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>$p_{mov}$</th>
<th>$e_{max}$</th>
<th>$e_{rel}$</th>
<th>$\Delta q_{\min}$</th>
<th>$\Delta q_{\max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>4</td>
<td>0.1 %</td>
<td>10^{-2} %</td>
<td>$T/20$</td>
<td>$T/10^3$</td>
</tr>
</tbody>
</table>

The SubIval solver allows to obtain a transient solution. For each period $t \in [(k - 1)T, kT]$ the value has been evaluated:

$$
e_{\text{steady}} = \max_{i=1,2,\ldots,n_w} \frac{kT}{T} \int_{t=(k-1)T}^{t} \left| \frac{\sum_{i=1}^{n_w} \hat{w}_i(t) T}{\hat{w}_i \text{ abs max}} \right| dt
$$

representing a measure determining whether the circuit is in a steady-state ($w_i(t)$ is the $i$-th variable in the $w(t)$ vector and $w_i \text{ abs max}$ is the maximum of absolute values of $w_i(t)$ in the considered period). The transient solution has been obtained for $n_T = 20$ periods, which was enough for $e_{\text{steady}}$ to become lower than $10^{-3}$%.

A comparison between the results (for the first and last periods) obtained through the SubIval solver and the referential (periodic steady-state) solution is given in Fig. 2. The variables selected for the comparison are the state variables:

$$
x(t) = [i_\alpha \ i_\gamma \ u_\beta \ u_\kappa]^T.
$$

So far one can observe that the numerical solver is performing properly as the results are very close in the steady-state. An analysis of both the estimated and actual error of the derivative approximation (for the discussed example) is presented in Sec. 8. The results depicted in the figure have been obtained with an application of the $d_s$ approach. However, the other approaches yielded results that are not visibly different.
7.2. Transient example. The second example that has been studied is one concerning the transient response of the parallel $RLC$ circuit presented in Fig. 3.

![Circuit with unit step excitation (each fractional element is marked by parentheses and the order of the element)](image)

For this simple example it is possible to obtain a referential analytical solution. It can be obtained similarly as in [45], where a fractional series $RLC$ circuit was studied. According to this solution – the capacitor voltage $u_C$ is:

$$u_C(t) = J_0 \frac{\lambda_1 E_{\alpha,\alpha+1}(\lambda_1 t^\alpha) - \lambda_2 E_{\alpha,\alpha+1}(\lambda_2 t^\alpha)}{C(\lambda_1 - \lambda_2)} t^\alpha, \quad (46)$$

while the current through the coil $i_L$ is given by:

$$i_L(t) = J_0 \frac{E_{\alpha,\alpha+1}(\lambda_1 t^\alpha) - E_{\alpha,\alpha+1}(\lambda_2 t^\alpha)}{2LC\sqrt{\Delta}} t^\alpha, \quad (47)$$

where

$$\Delta = \frac{1}{4R^2L^2} - \frac{1}{LC}, \quad (48)$$

and

$$\lambda_1 = -\frac{1}{2RL} + \sqrt{\Delta} \quad \text{and} \quad \lambda_2 = -\frac{1}{2RL} - \sqrt{\Delta}. \quad (50)$$

In (46) and (47) the notation:

$$E_{\alpha,\beta}(z) = \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + \beta)} \quad (51)$$

is used for the two parameter Mittag-Leffler function. The solution has been obtained for the interval $t \in [0, T_{\text{max}}]$, where $T_{\text{max}} = 0.14 \text{s}$ was selected. A comparison of the results obtained through the SubIval solver and through evaluations of the analytical solution is presented in Fig. 4.

![Comparison of time functions obtained by the SubIval solver and those of the referential (periodic steady-state) solution](image)

Fig. 4. Comparison of time functions obtained by the SubIval solver and those of the referential solution

8. Error analysis

In the SubIval solver the error estimation is computed for each fractional derivative, which for simplicity can denoted by:

$$d_i(t) = d_{L_\alpha} x_i(t). \quad (52)$$
A local truncation error estimation for a SubIval solver

A measure of the actual accuracy of the SubIval solver is then computed by means of the following formula for each $d_i$:

$$ e_i = 100 \cdot \frac{\max_{j=1,2,\ldots,n_2} |d_i(t_j) - d_i(\text{num } j)|}{|d_i(t_j)|} \% , \quad (53) $$

where $t_1, t_2, \ldots, t_{n_1}$ are the nodes on the time axis that have been selected by the SubIval solver, $d_i(t_j)$ is the value obtained for $t_j$ by means of the referential solution and $d_i(\text{num } j)$ is the value of $d_i(t_j)$ obtained by the SubIval solver.

A comparison of the maximum of errors (for each selected time instance) computed through the above formula with the maximum of estimated error values is depicted in Fig. 5 for the steady-state periodic example discussed in Subsec. 7.1. The figure presents results for each of the considered error computation approaches given in Sec. 6.

![Fig. 5. Comparison of the maximum of actual and estimated error values for the periodic steady-state example](image)

There are differences between the maximum values of the estimated and actual error. The pattern is, however, similar and both errors tend to a value near that of the selected $e_{\text{ctrl}}$ (where, for both examples, the value $10^{-2} \%$ was selected for this parameter). This can be observed in Table 3, where the average values of the actual and estimated errors are given for each error estimation approach.

<table>
<thead>
<tr>
<th>approach</th>
<th>periodical example</th>
<th>transient example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$</td>
<td>actual</td>
<td>estimated</td>
</tr>
<tr>
<td></td>
<td>$1.59 \cdot 10^{-2}$</td>
<td>$1.14 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>$d_s$</td>
<td>$1.24 \cdot 10^{-2}$</td>
<td>$1.02 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>$w$</td>
<td>$1.34 \cdot 10^{-2}$</td>
<td>$1.14 \cdot 10^{-2}$</td>
</tr>
<tr>
<td>$w_s$</td>
<td>$1.13 \cdot 10^{-2}$</td>
<td>$1.08 \cdot 10^{-2}$</td>
</tr>
</tbody>
</table>

The $d_s$ approach seems to yield the best results as the error is kept closest to $e_{\text{ctrl}}$ throughout the solving process. The $w_s$ approach works similarly, with no surprise as it also takes into account the same error values (though adding its own – basing on other variables, as explained previously in Sec. 6.

Figure 6 displays the maximum values of the actual and estimated error for the transient example of Subsec. 7.2.

![Fig. 6. Comparison of the maximum of actual and estimated error values for the transient circuit example](image)

This time one can clearly see an advantage of the $d_s$ and $w_s$ approaches. It can also be observed in the average values of the error in Table 3.

9. Remarks on computation tools

The computations of SubIval and the adaptive step size solver have been performed with own C# programs applying a DLL available at [46]. The library uses a part of the code given in [47] for the computation of the gamma function.

The SubIval step size adaptive solver required a system of equations to be solved for each iteration. In these computations classes from the MathNet Numerics library [48] have been applied for matrix and vector operations.

10. SubIval in comparison to other methods

SubIval is a method that can be applied in time stepping solvers for the approximation of the fractional derivative in the Caputo definition of order $\alpha \in (0, 1]$. A comparison has been performed with methods also belonging to this category. The methods can also be those that can be applied for the Riemann-Liouville fractional derivative as for the examples used in this paper zero initial conditions are applied.

There are not many methods whose implementations are available so that one can build a FDAE (fractional differential algebraic equations) solver upon them.
Some rare cases of solvers applicable in Matlab [49] (or GNU Octave [50]) are described in [51, 52], which can be found at [53]. They are based on PI (product integration) rule methods and FLMM (fractional linear multistep methods). The FLMM solvers have been implemented only for the case of a common fractional derivative order, hence these solvers cannot be applied for the periodic example. The available solvers are based on the following methods (the abbreviations in parentheses are for further reference):

- explicit product integration of rectangular type (PI1Ex solver),
- implicit product integration of rectangular type (PI1Im solver),
- implicit product integration of trapezoidal type (PI2Im solver),
- product integration with predictor-corrector (PI12PC solver),
- FLMM applying trapezoidal rule integration (FLMM_T solver),
- FLMM applying the Newton-Gregory formula for integration (FLMM_NG solver),
- FLMM applying the second order backward differentiation formula (FLMM_BDF2 solver).

Another possibility is the implementation of a method basing on the Grünwald-Letnikov approximation [54]. The Grünwald-Letnikov definition is equivalent with the Riemann-Liouville definition [55]. For the Caputo derivative it can be described by:

\[
\begin{align*}
\text{t}_0^\alpha \mathcal{D}_t^\alpha x(t) &= \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{j=N} (-1)^j \frac{\Gamma(\alpha + 1)}{j! \Gamma(\alpha - j + 1)} x(t - j \Delta t) \\
&- \frac{x(t_0)(t - t_0)^{-\alpha}}{\Gamma(1 - \alpha)},
\end{align*}
\]

with \( N \) being the index of the current time step. The solver basing on this method has been implemented as an implicit solver in GNU Octave. It is further on called the GL solver.

Both of the computational examples of Section 7 have been solved with the PI solvers, the FLMM solvers and the GL solver. As their implementations are performed in a Computer Algebra System (as opposed to the SubIval solver, which has been written in C#) it will take much longer for them to complete the computations, hence the computation times are not compared. What can be reliably compared is the accuracy of the results. The referential solutions have been implemented in GNU Octave – an auxiliary function [56, 57] has been used for the computations of the Mittag-Leffler function in (46) and (47).

In order to verify the result, error computations have been performed just like for the SubIval solver, i.e. applying (53). For all state variables, the average of errors for all time instances have been computed. The maximum of those values, for all solvers and both examples, is given in Table 4.

The mentioned solvers are only implemented for constant step sizes, hence in all cases the minimum step size applied for the SubIval solver \( \Delta t_{\text{min}} \) has been applied initially and further tests have been performed to observe if an increase of this value resulted in an increase in accuracy. Ultimately, the lowest error has been recorded. In the case of SubIval – the results are presented for the time step size adaptive solver with the \( d_s \) approach.

| Table 4 | Solver comparison: average error values (maximums for all variables) for both considered examples (*the solvers suffered from stability issues) |
|-----------------|------------------|------------------|------------------|------------------|------------------|------------------|
|                | periodic example | transient example |
| SubIval solver  | 1.24 · 10^{-2}   | 7.09 · 10^{-3}   |
| PI1Ex solver    | failed*          | 1.29             |
| PI1Im solver    | 8.02 · 10^{-1}   | 3.44 · 10^{-1}   |
| PI2Im solver    | 4.66 · 10^{-2}   | 1.10 · 10^{-1}   |
| PI12PC solver   | failed*          | 1.10 · 10^{-1}   |
| FLMM_T solver   | –                | 1.11 · 10^{-1}   |
| FLMM_NG solver  | –                | 1.11 · 10^{-1}   |
| FLMM_BDF2 solver| –                | 1.08 · 10^{-1}   |
| GL solver       | 7.23 · 10^{-1}   | 1.24 · 10^{-1}   |

The following observations can be made:

- the most accurate results have been obtained through SubIval solver,
- as for the other solvers – in general the best performance has been observed for the PI2Im solver,
- the explicit solvers failed to properly solve the periodic example (stability issues have been observed).

11. Summary and concluding remarks

The local truncation error estimation for a solver based on SubIval has been studied.

The details on the core computations of SubIval have been first presented (Sec. 2) and an approximate formula for a local truncation error has been derived (Sec. 3). Then the step size adaptivity in the SubIval solver has been discussed (Sec. 4).

A class of linear problems has been defined (Sec. 5) for which the research of this paper has been conducted.

For the error formula – various approaches have been proposed (Sec. 6) for the computation of the estimated error.

Two circuit problems with known referential solutions have been introduced (Subsec. 7.1 and 7.2 respectively) to check the correctness of the error estimation. The error estimation results (in comparison to the actual differences between the numerical and referential solutions) have been presented in Sec. 8.

A few additional remarks can be added as to the error estimation results. All strategies for the error estimation performed similarly – keeping the error within range of the \( e_{\text{ctrl}} \) value for the periodic steady-state example, while the \( d_s \) and \( w_s \) approaches performed significantly better for the transient example. This could be caused by the fact that the local truncation error formula is actually derived from the beginning with an assumption of errors omitted for the values computed at each of the time instances. The \( d_s \) and \( w_s \) approaches do take the same solutions for the computations of the fractional
derivatives in both stages (called stage A and stage B), which could be the reason for the more accurate resemblance of the actual error when using these approaches.

One source of inaccuracies in the error estimation that is worth mentioning is the fact that the step size is always adapted using results from preceding time steps (unless the time step is repeated, although such an occurrence was not necessary as the computed error values were all below \( \varepsilon_{\text{max}} \)).

Another source of inaccuracies is that the estimation is done by computing the differentials for \( \Xi_{\text{M}} \) and the actual error has been obtained for the entire interval.

Judging by the results presented in the paper – the \( d_{\text{L}} \) approach should be used for the considered class of problems as it yields the best results and requires only one solution to be computed at each time step. Additionally, no advantages have been observed when adding errors of selected variables in \( \text{w}(t) \) (which resulted in the \( w_{\text{L}} \) approach).

The \( w_{\text{L}} \) approach could provide a safeguard in nonlinear problems if the derivative error causes much greater errors for other variables. This, however, is a topic reserved for future studies.

REFERENCES


[43] http://functions.wolfram.com/GammaBetaErf/Beta3/03/01/02/0003/


[50] https://www.gnu.org/software/octave/


