Increasing the Efficiency of Shooting Methods for Terminal Value Problems of Fractional Order

Kai Diethelm[a,b]

[a] *Institut Computational Mathematics, Technische Universität Braunschweig, Fallersleber-Tor-Wall 23, 38100 Braunschweig, Germany*
[b] *GNS Gesellschaft für numerische Simulation mbH, Am Gaußberg 2, 38114 Braunschweig, Germany*

Abstract

Shooting methods are a well established tool for the numerical solution of terminal value problems of fractional order. However, they can be computationally quite expensive because of their iterative nature in which (a) each single iteration may be costly, and (b) the number of iterations can be large. In this paper we propose algorithmic strategies for improving the efficiency of such methods. Our strategies are aimed at simultaneously reducing the cost of each iteration and reducing the number of required iterations.

Keywords: terminal value problem, numerical solution, shooting method, Caputo derivative
2010 MSC: 65L10, 65R20

1. Introduction

Terminal value problems involving ordinary differential equations of fractional order nowadays play an important role in the modeling of many phenomena in physics, engineering, etc., both in their own right and as sub-problems arising in connection with fractional-order processes that have been observed at a point in time other than their starting point. The class of problems that we shall consider here is of the form

$$D^{\alpha}_{*a}y(t) = f(t, y(t)), \qquad y(b) = y^*, \qquad a \le t \le b, \tag{1}$$

which is the most general form of a terminal value problem involving an explicit single-term differential equation. Following the commonly accepted notation (see, e.g., [5] and the references cited therein), we use the symbol D^{α}_{*a} in eq. (1) and throughout the remainder of this paper to denote the Caputo differential operator of order α with starting point a; in particular the subscripted asterisk is used to distinguish this operator from the Riemann-Liouville derivative that is usually designated by D^{α}_{a} and that we will not consider here. We restrict our attention to the case that $0 < \alpha < 1$ because this is the case that is relevant to the vast majority of applications, cf., e.g., [1, 2, 5, 6, 12, 19] and the

references cited therein. Thus, we are looking for a solution $y : [a, b] \to \mathbb{R}$ to a fractional differential equation involving a Caputo differential operator D^{α}_{*a} with starting point a (the left end point of the interval where the solution is sought) under the assumption that the value of the function y is known only at the point b, i.e. the right end point of the interval of interest. The question for the existence and uniqueness of solutions to such problems under the usual assumptions (essentially, continuity of f on a suitable set and a Lipschitz property of f with respect to the second variable) has recently been answered affirmatively [4, 8, 14, 15].

Problems of the form (1) arise in a number of situations, in particular when fractional differential equations are used for modeling concrete phenomena in physics, engineering, biology, etc. (cf. [2, 5, 6, 12, 19] for specific examples) and information about the state of the system is known at some point $t = b$, but not at the beginning of the process, i.e. at $t = a$. One is then often interested in finding out how the system started which means that eq. (1) needs to be solved. Depending on the specific problem at hand, it may be sufficient to find this solution and thus the behavior on the interval $[a, b]$, or one must use this information — in particular the starting value $y(a)$ — as an intermediate result to define a classical initial value problem consisting of this initial condition and the differential equation from (1) and then solve this initial value problem on the larger interval $[a, \tilde{b}]$ with $\tilde{b} > b$, thus not only reconstructing the behavior of the process between the starting point a and the point of observation b, but also predicting its development for future times $t > b$.

We shall therefore now briefly review existing approaches for numerically solving the problem (1) and then propose a technique for increasing their efficiency. In passing, we note that our methods may be used as a building block for solving the even more general problem of finding a solution to eq. (1) under the additional assumption that the starting point a is also unknown. To the best of our knowledge, this latter problem has not been addressed in the literature yet; we defer a detailed discussion to a later paper.

2. Numerical Approaches

Essentially, there are two different ways in which one can tackle the given terminal value problem (1) numerically. The first idea is based on the observation [5, Theorem 6.18] that the problem is essentially a special case of a boundary value problem and can hence be rewritten in the form of the equivalent integral equation of Fredholm type

$$y(t) = y^* + \frac{1}{\Gamma(\alpha)} \int_a^b G(t, s) f(s, y(s)) \mathrm{d}s \tag{2}$$

where

$$G(t, s) = \begin{cases} -(b - s)^{\alpha - 1} & \text{for } s > t, \\ (t - s)^{\alpha - 1} - (b - s)^{\alpha - 1} & \text{for } s \leq t. \end{cases} \tag{3}$$

This approach allows to employ classical techniques like finite elements or finite differences for the solution of such Fredholm equations in order to solve the terminal value problem. However, such an approach has in the past been used mainly for two-point boundary value problems containing differential equations of order $\alpha \in (1, 2)$; cf., e.g., [13, 18, 20].

In this paper, we shall concentrate on a different method that has been studied in detail, e.g., in [14, 16, 17]. This approach is also based on a suitable adaptation of a technique known for integer-order problems, namely the idea of a shooting method. When transferred to the fractional-order setting under consideration here, this method can be summarized as follows:

1. We are looking for a solution on the interval $[a, b]$. If the value of the solution y at the starting point a of the differential operator D^{α}_{*a} were known, we would be dealing with a classical initial value problem and could use the corresponding numerical methods.
2. As the value $y(a)$ is not known, we begin by guessing a first approximation y_{10}, say, for $y(a)$.
3. Then we go on in an iterative manner that can be described in an abstract way (that will be made concrete in the ensuing paragraphs) as follows:
   (a) In the k-th passage through the iteration loop, we determine a numerical solution y_k to the initial value problem consisting of the given differential equation from (1) combined with the initial condition $y_k(a) = y_{k0}$. The terminal condition that appears in eq. (1) is ignored at this stage.
   (b) We then look at the value $y_k(b)$ and compare it to the value y^* required by the terminal condition. If these two values are sufficiently close to each other, we exit the loop and accept y_k as the approximate solution to the complete terminal value problem (1).
   (c) Otherwise, we compute a new starting value $y_{k+1,0}$ from the previous value y_{k0}, taking into account the difference between $y_k(b)$ and y^*, and continue with the iteration.

In their detailed comparative analysis [17], Ford et al. concluded that the Adams-type predictor-corrector method from [9, 10] was a very good choice for solving the initial value problems arising in each iteration of the process outlined above. For the purposes of this paper, we shall follow their advice and combine this with the idea from [3] where it has been stated that a suitable choice of the number of corrector iterations in the predictor-corrector method improves the convergence order to $O(h^2)$ essentially without increasing the computational cost. However, it will be immediately clear that our key findings indicated below should remain valid for differently chosen numerical solvers.

In order to begin the process, we need to find an initial guess y_{10} for the starting value. Owing to the lack of other information, we here suggest to use the terminal value, i.e. to define

$$y_{10} := y^* \tag{4}$$

where y^* is the given value from the original terminal value problem (1).

Finally, to complete the description of the algorithm, we mention that the computation of the starting value $y_{k+1,0}$ for the $(k+1)$st iteration can be based on the result from [8] that the value must be smaller than y_{k0} if $y^* < y_k(b)$ and larger than y_{k0} if $y^* > y_k(b)$. As proposed in [17], for determining the precise choice a bisection method is used.

While this approach is straightforward to derive, easy to implement and known to work successfully in practice [14, 17], a closer look reveals a certain disadvantage, namely that it can be computationally rather costly, at least if a high accuracy of the numerical solution is required. Two factors contribute to this problem:

1. Usual accuracy requirements demand the step size h of the initial value problem solver to be relatively small which leads to a large number $N = (b-a)/h$ of discretization points. As the computational complexity of the solver is $O(N^2)$, cf., e.g., [11, §7.3], this means that each individual iteration of the loop can be rather expensive.
2. Since no information whatsoever is assumed to be available with respect to the correct inital value $y(a)$, one needs to start the iteration with a completely arbitrary guess, and thus one has to expect that the number of iterations required to reach a reasonable accuracy can be very large.

We shall therefore now derive a strategy for modifying the basic algorithm described above in an attempt to find a computationally cheaper and hence more efficient way of solving the terminal value problem (1).

3. Improvements of the Approach

3.1. Choice of the Step Size for the IVP Solver

Our first step towards reducing the complexity of the algorithm is based on the simple observation that the overall error $y - y_k$ of the approximate solution in the k-th iteration can be additively decomposed into two components,

$$\delta_k := y - y_k = \delta_k^{\mathrm{I}} + \delta_k^{\mathrm{N}}, \tag{5}$$

where the first component δ_k^{I} is the error that results from the fact we are actually trying to solve the wrong initial value problem because the exact initial value is unknown, and the second component δ_k^{N} comes from the numerical solution method for this initial value problem.

Now, during the early phase of the iteration, i.e. for small values of the iteration counter k, the currently used initial value y_{k0} typically is still quite far away from the true value $y(a)$, and hence the component δ_k^{I} is also likely to be rather large, i.e. the solution to the current initial value problem is quite far away from the exact solution y over the entire interval. It thus follows that using the small step size h required for an accurate final solution would lead to the relation $\delta_k^{\mathrm{N}} \ll \delta_k^{\mathrm{I}}$, and hence a moderate increase in the step size h would lead to a moderate increase in the error component δ_k^{N} and hence only to a hardly noticeable change in the overall error δ_k (recall that, as stated above,

under typical conditions we may expect $\delta_k = O(h^2)$ if the number of corrector iterations in the Adams method is properly chosen [3]).

Thus, our idea is not to use the same small step size h for the initial value problem solver for all iterations, but to use a step size h_k in the k-th iteration, and to choose this step size rather large for small k and to successively reduce it as k grows. Clearly, it follows from our considerations above that this has a significant positive influence on the computational cost, whereas the adverse effects due to the increased approximation error are negligible.

Our precise choice of h_k was rather simple: Given the total number of iterations to be performed, say K, and the mesh size h_K for the final iteration (with the finest mesh), we simply used

$$h_k = \frac{K}{k} h_K \qquad (k = 1, 2, \ldots, K).$$

The numerical examples in Section 4 indicate that the method works very well.

A much more radical, but in fact not far fetched, idea can be motivated by the observation that $\delta_k^{\mathrm{I}} = O(2^{-k})$ (this follows from [5, Theorem 6.20] in conjunction with the properties of the bisection strategy used for updating the initial value); specifically, as we attempt to balance the contributions δ_k^{I} and δ_k^{N} that together make up the entire error δ_k, we choose the auxiliary value $\tilde{h}_k := c_{\mathrm{S}} 2^{-k/2}$ where c_{S} is a user defined parameter, compute $\tilde{N}_k := (b-a)/\tilde{h}_k$, determine N_k by rounding $\tilde{N}_k$ to the nearest integer, and finally set $h_k := (b-a)/N_k$. In this way, the k-th iteration has an arithmetic complexity of

$$O(N_k^2) = O(\tilde{N}_k^2) = O(\tilde{h}_k^{-2}) = O(2^k),$$

and hence the total algorithm requires a cost of

$$O\left(\sum_{k=1}^{\ell} 2^k\right) = O(2^{\ell+1}) \leq 2C \cdot 2^{\ell},$$

where ℓ denotes the total number of iterations required, which compares favourably with the complexity of

$$O\left(\sum_{k=1}^{\ell} 2^{\ell}\right) = O(\ell 2^{\ell}) \leq \ell C \cdot 2^{\ell}$$

that we would have required if all iterations had used the finest mesh (with mesh size h_ℓ) in the initial value problem solver. However, in our numercial experiments we found that this choice was too ambitious; the resulting approximate solutions using the same number of iterations were significantly worse than those using the approach mentioned first. Increasing the number of iterations to a point that led to run times being similar to those of the first approach improved the accuracy somewhat but was still not competitive with the first approach. We do believe, however, that a refinement of this alternative mesh size selection strategy could lead to a better performance.

3.2. Choice of the Fundamental Interval for the IVP

Another idea that can be used to reduce the computational cost is based on a completely different concept. Specifically, while the method of Subsection 3.1 was based on reducing the complexity of each iteration step, we now describe an approach that is targeted at reducing the number of iterations by trying to overcome the problem caused by the unknown choice for the initial value in the first passage through the iteration loop.

The method is based on the heuristic argument that, while the known terminal value $y(b)$ is not a useful approximation for the initial value $y(a)$ if the distance between initial point a and terminal point b is large, it is likely to be reasonably close to the initial value if the interval between the two points a and b is small. We combine this argument with the observation from [7] that the solution y to the terminal value problem (1) depends on the parameter a, i.e. the location of the starting point, in a continuous manner. Thus, instead of solving the problem (1) on the potentially long interval $[a, b]$ directly, we introduce intermediate points a_j $(j = 0, 1, 2, \ldots, M)$ such that $a = a_M < a_{M-1} < a_{M-2} < \cdots < a_1 < a_0 = b$ and solve the analogous problems

$$D^{\alpha}_{*a_j} y^{[j]}(t) = f(t, y^{[j]}(t)), \qquad y^{[j]}(b) = y^*, \qquad a_j \le t \le b, \tag{6}$$

on the short intervals $[a_j, b]$ successively for $j = 1, 2, \ldots, M$. In this process, we follow the structure outlined above for each of the subintervals; the key point here is the selection of the initial values relative to the current starting point for the first iteration of the shooting procedure for this interval: When we deal with the first interval $[a_1, b]$, we begin the shooting method with $y_{10}^{[1]} = y^*$. Later, in the j-th iteration, we have already solved the problem on the previous subinterval $[a_{j-1}, b]$ and have thus computed an approximation $y_K^{[j-1]}(a_{j-1})$ for some $K \in \mathbb{N}$. We then start the j-th iteration with this value, i.e. we set $y_{10}^{[j]} = y_K^{[j-1]}(a_{j-1})$.

Our numerical experiments have indicated that the simplest conceivable way of selecting the intermediate points a_j, namely a uniform distribution over the basic interval $[a, b]$, i.e. the choice

$$a_j := b - j\frac{b-a}{M}$$

with some $m \in \mathbb{N}$ yields a satisfactory behaviour; nevertheless we believe that a more sophisticated approach might lead to even better results. The effects that we obtained by this procedure are presented in Section 4.

3.3. Combining the Two Ideas

Clearly, it is possible to combine the two approaches described above, i.e. one need not use the very fine original mesh for the numerical initial value solver for solving each of the intermediate terminal value problems arising in the method of Subsection 3.2; rather, one can use the more efficient method

described in Subsection 3.1 for this sub-task. This should lead to an even more efficient method. However, our investigations performed so far have not yet led to a strategy for selecting suitable step sizes for solving the intermediate terminal value problems on the subintervals $[a_j, b]$ that performed in a satisfactory manner. We hope to be able to develop a suitable solution to this problem in the future.

3.4. Using Different Solvers in the Various Stages

It is also possible to replace our Adams-type predictor-corrector algorithm by a different numerical method in the first iteration steps. For example, a computationally cheaper scheme could be used in the phase where an accurate approximate solution is not yet needed. A scheme that does not have the predictor-corrector structure might be useful here. However, if that idea were followed without using any of the two ideas described above, the complexity would remain at $O(N^2)$ for an N-point discretization, and so while a reduction of the constant implicitly contained in the O-term is possible, the exponent of N that is essential for the high computational cost would remain unchanged. Thus, in an asymptotic sense this idea does not lead to an improvement over the concepts suggested above. We therefore refrain from pursuing it any further and only consider the Adams method as advised by Ford et al. [17].

4. Numerical Examples

We now report the results (in particular, the run times and the approximation errors) obtained by applying our algorithms to some example problems.

The run times reported below refer only to the numerical solution of the differential equation itself; preprocessing steps like the calculation of the required weights are excluded because they are known to be of linear complexity with respect to the number of grid points and thus negligible. All results were obtained on a standard desktop PC with an AMD Athlon 64 X2 Dual Core Processor 5200+ clocked at 2.6 GHz.

4.1. A Linear Problem

In order to demonstrate the improvements obtained by our suggested changes, we now present some typical example problems and compare the results obtained by the original shooting method with our modified algorithms. To this end, we shall first look at the terminal value problem

$$D^{\alpha}_{*0}y(t) = \Gamma(2+\alpha)t + \frac{1}{4}(y(t) - w - t^{1+\alpha}), \quad y(b) = b^{\alpha+1} + w, \quad t \in [0, b], \quad (7)$$

the special case $\alpha = 1/2$ and $w = 0$ of which has already been considered in [17]. The exact solution of eq. (7) is known to be $y(t) = t^{1+\alpha} + w$. For our tests we specifically choose $\alpha = 7/10$, $w = -3$ and $b = 12$, thus obtaining a rather long interval.

In order to assess the quality of the numerical results and to indicate what sort of accuracy can at best be expected, we first give some numerical results for the initial value problem associated to (7) which amounts to replacing the terminal condition in that equation by the initial condition $y(0) = w$. Some results obtained by using the Adams method with various step sizes are given in Table 1.

step size	0.1	0.05	0.025	0.0125	0.00625
max. error	2.01E-3	3.83E-4	7.29E-5	1.38E-5	2.63E-6
run time	0.2 ms	0.5 ms	1.8 ms	6.5 ms	24.9 ms

Table 1: Numerical results for initial value problem corresponding to (7).

In particular, we conclude from Table 1 that, e.g., in order to obtain an absolute accuracy of 10^{-5} we need to use a step size of approximately 0.01. Our next step is then to solve the terminal value problem using the standard shooting method where we choose the same step sizes as in our initial value example from Table 1 and choose the number of iterations, i.e. the number of initial value problems to be solved using the given step size, such that we obtain a similar maximal error. The results of this procedure can be seen in Table 2.

step size	0.1	0.05	0.025	0.0125	0.00625
total no. of solved IVPs	20	23	24	27	27
max. error	2.83E-3	2.58E-4	1.63E-4	1.25E-5	1.26E-6
run time	1.9 ms	6.0 ms	19.0 ms	72.5 ms	265.2 ms

Table 2: Numerical results for terminal value problem (7) using classical algorithm.

Our next step is then to replace the standard shooting method by the improved variant indicated in Subsection 3.1 where we now use larger step sizes for the initial value solver in the early iterations. As indicated in Table 3, in this particular example we are in a situation where the last iteration has the same error as it had in the unmodified version even though the preceding iterations were less accurate and thus faster than in the original algorithm. Thus, as can be seen from the speedup factors shown in Table 3, our modification does indeed provide a significant performance improvement.

We conclude the considerations regarding this specific example by looking at the effects of the modification proposed in Subsection 3.2. Thus, we now do not solve the equation on the complete interval $[a, b]$ directly, but we introduce intermediate points $a_1, a_2, \ldots, a_M = a$ and work on the successively growing subintervals $[a_j, b]$. For each subinterval we then used the algorithm with the standard step size. In our numerical experiments we found the best performance for the choice $M = 2$. The results given in Table 4 indicate that we now need to solve more initial value problems than in the original algorithm, but since these problems are computationally cheaper, the overall run time does not change

minimal step size	0.1	0.05	0.025	0.0125	0.00625
total no. of solved IVPs	20	23	24	27	27
max. error	2.83E-3	2.58E-4	1.63E-4	1.04E-5	1.26E-6
run time	1.0 ms	2.6 ms	7.8 ms	27.9 ms	98.1 ms
speedup	1.9	2.3	2.4	2.6	2.7

Table 3: Numerical results for terminal value problem (7) using algorithm of Subsection 3.1 (modified step size) and speedup compared to original version of algorithm.

very much. Also, the accuracy is slightly worse than in the original case. It thus turns out that we have so far not found a satisfactory value for the parameters. Further investigations in this direction seem to be necessary.

step size	0.1	0.05	0.025	0.0125	0.00625
M	2	2	2	2	2
total no. of solved IVPs	27	31	34	36	26
max. error	2.86E-3	9.60E-4	4.41E-5	2.08E-5	7.23E-6
run time	2.1 ms	6.4 ms	20.9 ms	74.2 ms	290.3 ms

Table 4: Numerical results for terminal value problem (7) using algorithm of Subsection 3.2 (modified interval length).

For the reasons explained in Subsection 3.3, we cannot report any results for the combination of the two approaches yet. Some ideas for selecting the parameters in this case are currently under consideration.

4.2. A Nonlinear Problem

Next, we look at the nonlinear example problem

$$\begin{aligned} D^{\alpha}_{*0}y(t) &= \frac{40320}{\Gamma(9-\alpha)}t^{8-\alpha} - 3\frac{\Gamma(5+\alpha/2)}{\Gamma(5-\alpha/2)}t^{4-\alpha/2} + \frac{9}{4}\Gamma(1+\alpha) \\ &\quad + \left(\frac{3}{2}t^{\alpha/2} - t^4\right)^3 - (y(t))^{3/2}, \qquad (8) \\ y(b) &= b^8 - 3b^{4+\alpha/2} + \frac{9}{4}b^{\alpha}, \qquad 0 \le t \le b. \end{aligned}$$

This example has also been considered in [17]; the solution is

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

We report results for this example for $\alpha = 0.4$ and $b = 1.1$, and as in the previous example we first state in Table 5 some findings for the associated initial value problem, i.e. the problem consisting of the differential equation from (8) and the initial condition $y(0) = 0$.

step size	0.02	0.01	0.005	0.0025	0.00125
max. error	1.56E-3	3.94E-4	9.94E-5	2.50E-5	6.26E-6
run time	0.2 ms	0.4 ms	1.0 ms	2.5 ms	7.4 ms

Table 5: Numerical results for initial value problem corresponding to (8).

As above, we then continue with the terminal value problem that we are really interested in. The results for the original algorithm are given in Table 6, and the corresponding values for the modification indicated in Subsection 3.1 (increased step size for initial value solver in early iterations) can be seen in Table 7. As in the first example, we again observe a significant reduction in the required computing time, albeit this time at the cost of a slightly deteriorated accuracy.

step size	0.02	0.01	0.005	0.0025	0.00125
total no. of solved IVPs	10	10	10	13	14
max. error	2.94E-3	7.34E-4	1.05E-4	3.93E-5	6.55E-6
run time	1.6 ms	3.2 ms	6.9 ms	20.8 ms	58.0 ms

Table 6: Numerical results for terminal value problem (8) using classical algorithm.

minimal step size	0.02	0.01	0.005	0.0025	0.00125
total no. of solved IVPs	12	12	11	14	14
max. error	9.65E-3	2.62E-3	2.62E-4	5.90E-5	1.47E-5
run time	1.4 ms	2.1 ms	3.9 ms	11.3 ms	30.3 ms
speedup	1.1	1.5	1.8	1.8	1.9

Table 7: Numerical results for terminal value problem (8) using algorithm of Subsection 3.1 (modified step size) and speedup compared to original version of algorithm.

For the modification of Subsection 3.2, i.e. the introduction of intermediate intervals, we again found the choice $M = 2$ to give the best results, and this time we have to conclude (see Table 8) a similar accuracy as in the original algorithm can be reached but the run times are now somewhat longer, so the behaviour is still not satisfactory.

5. Conclusion

We have developed two proposals for improving the efficiency of shooting methods for terminal value problems for fractional differential equations. The first method is based on using different step sizes for the initial value solver that the shooting method uses internally, thus reducing the computational cost of each individual iteration. The second method uses the idea not to tackle the

minimal step size	0.02	0.01	0.005	0.0025	0.00125
M	2	2	2	2	2
total no. of solved IVPs	15	15	15	21	25
max. error	3.00E-3	8.18E-4	2.73E-4	4.85E-5	1.13E-5
run time	2.1 ms	4.1 ms	8.5 ms	27.5 ms	83.5 ms

Table 8: Numerical results for terminal value problem (8) using algorithm of Subsection 3.2 (modified interval length).

problem on the complete interval directly but to work on intermediate subintervals first, thus aiming at a reduction of the number of required iterations. The approaches can be used either separately or in combination with each other. They depend on certain parameters, such as the choice of the step sizes h_k or the number and location of the intermediate points a_j. Our numerical results indicate that the first method indeed performs significantly better than the underlying classical fundamental algorithms; we have made some specific proposals for practically useful values for its parameters. For the second approach, the search for a satisfactory choice of its parameters is still ongoing.

References

References

[1] D. Baleanu, K. Diethelm, E. Scalas, J. J. Trujillo, *Fractional Calculus: Models and Numerical Methods.* World Scientific, Singapore (2012).

[2] M. Caputo, *Linear models of dissipation whose Q is almost frequency independent–II.* Geophys. J. Royal Astronom. Soc. **13** (1967), 529–539; reprinted in Fract. Calc. Appl. Anal. **11** (2008), 4–14.

[3] K. Diethelm, *Efficient Solution of Multi-term Fractional Differential Equations Using P(EC)^{m}E Methods.* Computing **71** (2003), 305–319.

[4] K. Diethelm, *On the Separation of Solutions of Fractional Differential Equations.* Fract. Calc. Appl. Anal. **11** (2008), 259–268.

[5] K. Diethelm, *The Analysis of Fractional Differential Equations.* Springer, Berlin (2010).

[6] K. Diethelm, *A Fractional Calculus Based Model for the Simulation of an Outbreak of Dengue Fever.* Nonlinear Dynamics **71** (2013), 613–619.

[7] K. Diethelm, *An Extension of the Well-posedness Concept for Fractional Differential Equations of Caputo's Type.* Appl. Anal. **93** (2014), 2126–2135.

[8] K. Diethelm, N. J. Ford, *Volterra Integral Equations and Fractional Calculus: Do Neighboring Solutions Intersect?* J. Integral Equations Appl. **24** (2012), 25–37.

[9] K. Diethelm, N. J. Ford, A. D. Freed, *A Predictor-Corrector Approach for the Numerical Solution of Fractional Differential Equations.* Nonlinear Dynamics **29** (2002), 3–22.

[10] K. Diethelm, N. J. Ford, A. D. Freed, *Detailed Error Analysis for a Fractional Adams Method.* Numer. Algorithms **36** (2004), 31–52.

[11] K. Diethelm, N. J. Ford, A. D. Freed, Y. Luchko, *Algorithms for the Fractional Calculus: A Selection of Numerical Methods.* Comput. Methods Appl. Mech. Eng. **194** (2005), 743–773.

[12] K. Diethelm, A. D. Freed, *Caputo Derivatives in Viscoelasticity: A Nonlinear Finite-deformation Theory for Tissue.* Fract. Calc. Appl. Anal. **10** (2007), 219–248.

[13] G. J. Fix, J. P. Roop, *Least Squares Finite-Element Solution of a Fractional Order Two-point Boundary Value Problem.* Computers Math. Applic. **48** (2004), 1017–1033.

[14] N. J. Ford, M. L. Morgado, *Fractional Boundary Value Problems: Analysis and Numerical Methods.* Fract. Calc. Appl. Anal. **14** (2011), 554–567.

[15] N. J. Ford, M. L. Morgado, *Stability, Structural Stability and Numerical Methods for Fractional Boundary Value Problems.* Oper. Theory Adv. Appl. **229** (2013), 157–173.

[16] N. J. Ford, M. L. Morgado, M. Rebelo, *Nonpolynomial Collocation Approximation of Solutions to Fractional Differential Equations.* Fract. Calc. Appl. Anal. **16** (2013), 874–891.

[17] N. J. Ford, M. L. Morgado, M. Rebelo, *High Order Numerical Methods for Fractional Terminal Value Problems.* Comput. Meth. Appl. Math. **14** (2014), 55–70.

[18] B. Jin, R. D. Lazarov, J. E. Pasciak, W. Rundell, *Variational Formulation of Problems Involving Fractional Order Differential Operators.* Math. Comput. (2014), in press.

[19] A. Schmidt, L. Gaul, *Finite Element Formulation of Viscoelastic Constitutive Equations Using Fractional Time Derivatives.* Nonlinear Dynam. **29** (2002), 37–55.

[20] M. Zayernouri, G. E. Karniadakis, *Fractional Sturm-Liouville Eigenproblems: Theory and Numerical Approximation.* J. Comput. Phys. **252** (2013), 495–517.