The FracPECE Subroutine for the Numerical Solution of Differential Equations of Fractional Order

[*]Kai Diethelm and [**]Alan D. Freed
[*]Institut für Angewandte Mathematik, TU Braunschweig
[**]Polymers Branch, NASA Lewis Research Center, Cleveland

Abstract

We present and discuss an algorithm for the numerical solution of nonlinear differential equations of fractional (i.e., non-integer) order. This algorithm allows us to analyze in an efficient way a mathematical model for the description of the behaviour of viscoplastic materials. The model contains a nonlinear differential equation of order β, where β is a material constant typically in the range $0 < \beta < 1$. This equation is coupled with a first-order differential equation. The algorithm for the numerical solution of these equations is based on a PECE-type approach.

1 Introduction

In this paper we introduce an algorithm for the numerical solution of nonlinear differential equations of fractional order β, where the main emphasis is placed on the case $0 < \beta < 1$. The development of this algorithm is motivated by a concrete application, namely the problem of modeling the behaviour of viscoplastic materials under certain conditions. In particular, we are interested in describing the strain and internal state reacting in response to changes in stress and temperature. In the theory of viscoelasticity, it is well

known [4] that the constitutive equations governing these phenomena involve differential equations of fractional order. Since the theory of viscoelasticity is essentially a linear theory, these differential equations are also linear, and therefore they may be solved using rather simple methods. We shall see that, in our situation, we have to replace these linear equations by nonlinear ones. The standard solution methods for the linear equations usually fail in the nonlinear case. Thus we now present a new algorithm, called *FracPECE*, that allows us to solve the differential equations and thus analyze the model in an efficient way. As the name *FracPECE* indicates, the code is based on the classical PECE (Predict, Evaluate, Correct, Evaluate) type approach, appropriately modified to be able to handle the fractional differential operators. In the derivation of the algorithm, we have taken particular care of the fact that the model does not consist of fractional differential equations only, but that first-order differential equations are contained too. This has imposed the requirement to find coherent schemes for both types of differential equations. The subroutine package has been implemented in the programming language Oberon [8, 22] in an object-oriented manner.

The structure of this paper is as follows. We begin by introducing the mathematical model of viscoplasticity that we want to analyze. In particular, we state the fractional differential equations that we want to solve. Together with some necessary mathematical preliminaries, this is done in §2. The algorithm itself is then presented in detail in §3. In particular, we shall mention its most important properties and discuss possible modifications in order to enhance the performance.

2 The Mathematical Model

In order to motivate the algorithm that we developed, we give a short description of (a slightly simplified version of) our mathematical model, thus explaining the equations we want to solve. For this description of our model of viscoplastic materials, it is useful to first briefly recall some key facts from fractional calculus.

The Riemann-Liouville derivative of order $\beta > 0$ of a function f with respect to the point t_0 is denoted and defined by [18, p. 59]

$$D_{t_0}^{\beta} f(t) = \frac{1}{\Gamma(m-\beta)} \frac{d^m}{dt^m} \int_{t_0}^{t} f(u)(t-u)^{m-\beta-1} du \qquad (1)$$

where m is the integer defined by the relation $m-1 < \beta \leq m$. If β is a natural number, then we recover the classical definition of the derivative. This form of a derivative has got very nice and useful mathematical properties [18, 20], but a direct application to physical problems frequently leads to difficulties

when attempting to handle initial conditions in a meaningful way. The reason is that the fractional derivative of a constant is not identically zero. Therefore, it is usual to look at Caputo's [3] variant of the Riemann-Liouville derivative, i.e.

$$d_{t_0}^{\beta} f(t) = D_{t_0}^{\beta} \left(f - \sum_{k=0}^{m} \frac{(\cdot - t_0)^k}{k!} y^{(k)}(t_0+) \right)(t), \tag{2}$$

where m is as above; cf. also the survey article [16] and the references cited therein. There is one particularly important property that both these (and indeed all other) fractional derivatives share, namely the fact that these operators are not local. This means that the value of $d_{t_0}^{\beta} f(t)$ depends on all the values of f in the interval $[t_0, t]$ (i.e. on the entire history of the function f). In sharp contrast to this, differential operators of integer order are always local, i.e. they can be evaluated from functions values of f in an arbitrarily small neighborhood of t. This hereditary behavior makes fractional differential operators a natural choice when it comes to modeling path-dependent phenomena. On the other hand, the property has got an influence on the numerical scheme: The fact that we cannot neglect the past history of f implies an increased arithemtic complexity of the algorithm compared to similar methods for integer-order equations.

In our model presented in [6] that we shall now briefly review, we have essentially set up three sets of equations. The first set describes the hydrostatic response (i.e. the dilatational kinetics), the second set deals with deviatoric response (i.e. the distortional kinetics), and the last set covers the evolution of an internal stress.

Our description starts with the first group of relations. We have decomposed the dilatational strain response e additively into an elastic (reversible) part e^{e} and a plastic (irreversible) part e^{p}, i.e.,

$$e = e^{\mathsf{e}} + e^{\mathsf{p}}, \tag{3}$$

where the individual parts fulfil the relations

$$e^{\mathsf{e}} = 3\alpha(T - T_0) - \frac{p}{\kappa^{\mathsf{e}}}, \tag{4}$$

$$\dot{e}^{\mathsf{p}} = -\frac{1}{\tau_0 \exp(Q/RT)} \frac{\xi p - x}{\kappa^{\mathsf{p}}}. \tag{5}$$

Here α is the mean coefficient of thermal expansion (and thus a material constant), T is the current temperature, T_0 is the temperature at the beginning of the experiment, p is the pressure, κ^{e} and κ^{p} are the elastic and plastic bulk moduli, τ_0 is a characteristic time, $R = 8.3145$ J/mol K is the universal gas constant, Q is the activation energy, and x is the hydrostatic component of

the internal stress. These relations describe the dilatational kinetics of the model. Formulae (3–5) constitute the standard linear solid of viscolelasticity in disguise, at least when $x = -\kappa^{\mathsf{p}}[e - 3\alpha(T - T_0)]$. This choice for a constitutive relation was made because, based on our present knowledge of hydrostatic data, this seems to be the best model currently available for the problem at hand.

For the distortional response $\mathbf{E} = \mathbf{E}^{\mathsf{e}} + \mathbf{E}^{\mathsf{p}}$ (also decomposed into elastic and plastic components), we have found

$$\mathbf{E}^{\mathsf{e}} = \frac{1}{2\mu^{\mathsf{e}}}\mathbf{S} \tag{6}$$

and

$$\dot{\mathbf{E}}^{\mathsf{p}} = \frac{1}{\tau}\left(\frac{\|\mathbf{S} - \mathbf{X}\|}{\mu^{\mathsf{p}}(1-\xi)}\right)^{\mathfrak{n}} \frac{\mathbf{S} - \mathbf{X}}{2\|\mathbf{S} - \mathbf{X}\|}. \tag{7}$$

In these equations, μ^{e} and μ^{p} are are the elastic and plastic shear moduli, $\mathbf{S}$ is the deviatoric stress, $\tau = \tau_0 \exp(Q/RT)$, ξ quantifies the ratio $\|\mathbf{S}\|/\|\mathbf{X}\|$ at saturation satisfying the restriction $0 < \xi < 1$, and $\mathfrak{n}$ establishes the stress dependence. These relations are motivated by corresponding models from nonlinear viscoelasticity [12]. Note that the nonlinearity of our set of equations is evident here since the material constant $\mathfrak{n}$ typically lies in a range between 4 and 8.

Whereas fractional derivatives are a very natural and common tool in the classical models of viscoelasticity [4], it seems that in the rather young area of viscoplasticity, the only such model seems to have been proposed by Lion [14]. Our approach distinguishes itself from his in that we consider a nonlinear material model that is, we believe, more keeping in line with dislocation physics.

There is abundant metallographic and experimental evidence implying that microstructure evolves in a path-dependent manner and, as a consequence, the state of stress that a microstructure generates will be hereditary in nature. Therefore, as mentioned above, the internal stress can be considered as a viable candidate for postulating fractional-order equations for its evolution.

Thus, for the evolution of the internal stress $\chi = \mathbf{X} - x\mathbf{I}$ ($\mathbf{I}$ being the identity tensor), we demand in the model that

$$x = -\kappa^{\mathsf{p}}\left(e - 3\alpha(T - T_0)\right) \tag{8}$$

$$d^{\beta}_{t_0}\mathbf{X} = \frac{2\mu^{\mathsf{e}}}{\tau_0^{\beta-1}}\left(\dot{\mathbf{E}}^{\mathsf{p}} - \frac{1}{\tau}\left(\frac{\|\mathbf{S} - \mathbf{X}\|}{\mu^{\mathsf{p}}(1-\xi)}\right)^{\mathfrak{n}} \frac{\mathbf{X}}{2L}\right), \tag{9}$$

$$d^{\beta}_{t_0}L = \frac{\mu^{\mathsf{p}}}{\tau\tau_0^{\beta-1}}\left(\left(\frac{\|\mathbf{S} - \mathbf{X}\|}{\mu^{\mathsf{p}}(1-\xi)}\right)^{\mathfrak{n}} - \left(\frac{L}{\xi\mu^{\mathsf{p}}}\right)^{\mathfrak{n}}\right), \tag{10}$$

where β is a material constant. As in the first-order differential equation (7), we see that the fractional-order differential equations (9) and (10) are also highly nonlinear.

These equations fully describe our model for the viscoplastic material, expressing the unknown quantities χ (internal pressure) and L (limit strength of χ) in terms of the control variables T (temperature), $\mathbf{S}$ (deviatoric stress), and p (pressure). Of course, we have to assume that initial values for the unknown functions at time $t = t_0$ are given.

The reader is referred to [6] for further details.

3 The Numerical Algorithm

3.1 Preliminary Remarks

The viscoplastic material model just presented, which we believe has the potential to apply to plastics and metallics alike for a broad class of applications, requires the simultaneous solution of a linear first-order differential equation for the plastic dilatation e^{p}, a nonlinear first-order differential equation for the plastic distortion $\mathbf{E}^{\mathrm{p}}$, and two nonlinear fractional-order differential equations for the internal stress $\mathbf{X}$ and its limit strength L. It is imperative that one has a coherent numerical scheme for handling these various types of differential equations simultaneously, which is what we now turn our attention to. For the sake of simplicity, we restrict ourselves to the case important for the application we have in mind, viz. $0 < \beta < 1$, and we only state here that the considerations below can be generalized to arbitrary positive β.

The definition of the fractional derivative and some well known results of fractional calculus (cf., e.g., [5]) tell us that we can interpret a fractional differential equation

$$D_{t_0}^{\beta}(y - y_0)(t) = f(t, y(t)), \qquad y(t_0) = y_0 \tag{11}$$

as a Volterra integral equation of the second kind with a strongly singular kernel,

$$\frac{1}{\Gamma(-\beta)} \int_{t_0}^{t} \frac{y(u) - y_0}{(t-u)^{\beta+1}} du = f(t, y(t)), \qquad y(t_0) = y_0 \tag{12}$$

thus forcing us to regularize the integral in Hadamard's finite-part sense. The numerical methods developed for this purpose, however, are currently able to cope with linear equations only [5, 7, 9]. Alternatively, we can apply a fractional integral operator to the differential equation and incorporate the

initial conditions, thus converting the equation into the equivalent equation

$$y(t) = y(t_0) + \frac{1}{\Gamma(\beta)} \int_{t_0}^{t} (t-u)^{\beta-1} f(u, y(u)) du \tag{13}$$

which also is a Volterra equation of the second kind, but now with a weakly singular kernel, such that a regularization is not necessary any more. It seems [17] that there exists only a very small number of software packages for nonlinear Volterra equations [1, 2]. Moreover, these routines are tailored specifically for equations with smooth kernels, and it is known (cf., e.g., [2, p. 63]) that they fail to produce reliable results (and return error flags instead) even if the kernels are continuous but not differentiable. In our situation the kernel is not continuous (actually it even is unbounded), and therefore these classical numerical algorithms are unable to handle our equation.

3.2 Description of the Algorithm

In the following, we shall now present our scheme for the numerical solution of the general fractional differential equation (11) that may of course be used for the special case discussed in the previous section, viz., eqs. (9) and (10). In the development we have in mind that these fractional differential equations are coupled with the first-order differential equations (4)–(7), and thus we need to combine the fractional-order algorithm with a classical method. The results of [15] give us the general advice to choose these two algorithms in such a way that both methods are based on very similar construction principles. We thus chose an Adams-Bashforth-Moulton approach for both integrators. Whereas this approach is very well known for first-order equations [10, 11], we shall give some more details for the fractional variant.

The key to the derivation of the method is to replace the original fractional differential equation (11) by the equivalent weakly singular Volterra equation (13) and to implement a product integration method for the latter. What we do is simply use the product trapezoidal quadrature formula with nodes t_j ($j = 0, 1, \ldots, n+1$), taken with respect to the weight function $(t_{n+1} - \cdot)^{\beta-1}$, to replace the integral. In other words, we apply the approximation

$$\int_{t_0}^{t_{n+1}} (t_{n+1} - u)^{\beta-1} g(u) du \approx \int_{t_0}^{t_{n+1}} (t_{n+1} - u)^{\beta-1} g_{n+1}(u) du \tag{14}$$

where g_{n+1} is the piecewise linear interpolant for g whose nodes and knots are chosen at the t_j, $j = 0, 1, 2, \ldots, n+1$. An explicit calculation yields that we can write the integral on the right-hand side of eq. (14) as

$$\int_{t_0}^{t_{n+1}} (t_{n+1} - u)^{\beta-1} g_{n+1}(u) du = \sum_{j=0}^{n+1} a_{j,n+1} g(t_j), \tag{15}$$

where

$$a_{j,n+1} = \int_{t_0}^{t_{n+1}} (t_{n+1} - u)^{\beta-1} \phi_{j,n+1}(u) du, \tag{16}$$

and

$$\phi_{j,n+1}(u) = \begin{cases} (u - t_{j-1})/(t_j - t_{j-1}) & \text{if } t_{j-1} < u < t_j, \\ (t_{j+1} - u)/(t_{j+1} - t_j) & \text{if } t_j < u < t_{j+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

In the case of equispaced nodes $t_j = t_0 + jh$ with some fixed h, the relations of eq. (16) reduce to

$$a_{j,n+1} = \begin{cases} \dfrac{h^\beta}{\beta(\beta+1)} \left(n^{\beta+1} - (n-\beta)(n+1)^\beta \right) & \text{if } j = 0, \\ \dfrac{h^\beta}{\beta(\beta+1)} & \text{if } j = n+1, \end{cases} \tag{18}$$

whereas for $1 \leq j \leq n$, we have

$$a_{j,n+1} = \frac{h^\beta}{\beta(\beta+1)} \left((n-j+2)^{\beta+1} - 2(n-j+1)^{\beta+1} + (n-j)^{\beta+1} \right). \tag{19}$$

This then gives us our corrector formula, i.e. the fractional variant of the one-step Adams-Moulton method, which is

$$y_{n+1} = y_0 + \frac{1}{\Gamma(\beta)} \left(\sum_{j=0}^{n} a_{j,n+1} f(t_j, y_j) + a_{n+1,n+1} f(t_{n+1}, y_{n+1}^P) \right). \tag{20}$$

The remaining problem is the determination of the predictor formula that we require to calculate the value y_{n+1}^P. The idea we use to generalize the one-step Adams-Bashforth method is the same as the one described above for the Adams-Moulton technique: We replace the integral on the right-hand side of eq. (13) by the product rectangle rule, i.e.

$$\int_{t_0}^{t_{n+1}} (t_{n+1} - u)^{\beta-1} g(u) du \approx \sum_{j=0}^{n} b_{j,n+1} g(t_j), \tag{21}$$

where now

$$b_{j,n+1} = \int_{t_j}^{t_{j+1}} (t_{n+1} - u)^{\beta-1} du = \frac{1}{\beta} \left((t_{n+1} - t_j)^\beta - (t_{n+1} - t_{j+1})^\beta \right). \tag{22}$$

Again, in the equispaced case, we have the simpler expression

$$b_{j,n+1} = \frac{h^\beta}{\beta}\left((n+1-j)^\beta - (n-j)^\beta\right). \tag{23}$$

Thus, the predictor y_{n+1}^P is determined by

$$y_{n+1}^P = y_0 + \frac{1}{\Gamma(\beta)}\sum_{j=0}^{n} b_{j,n+1} f(t_j, y_j). \tag{24}$$

This completes the description of our basic algorithm, which is the fractional version of the one-step Adams-Bashforth-Moulton method. Recapitulating, we see that we first have to calculate the predictor y_{n+1}^P according to eq. (24), then we evaluate $f(t_{n+1}, y_{n+1}^P)$, use this to determine the corrector y_{n+1} by means of eq. (20), and finally evaluate $f(t_{n+1}, y_{n+1})$ which is then used in the next integration step. Therefore, methods of this type are frequently called predictor-corrector or, more precisely, PECE (Predict, Evaluate, Correct, Evaluate) methods.

3.3 Main Properties of the Algorithm

We shall now describe the main properties of the algorithm. In particular, we find that, with respect to the most important questions, the behaviour of the method is independent of the parameter β and that it behaves in a way that is very similar to the classical one-step Adams-Bashforth-Moulton method (i.e. the case $\beta = 1$). Therefore, a combination of the fractional Adams-Bashforth-Moulton scheme outlined in the previous subsection with its classical version is a very natural idea when the set of equations to be solved consists, as in our application, of first-order differential equations combined with fractional-order differential equations. Moreover, it is no problem to extend the concept to include more fractional order equations, even if the order of the differential operators involved varies from equation to equation.

3.3.1 Stability

The issue of stability is very important when implementing the method on a computer in finite-precision arithmetic because we must take into account the effects introduced by rounding errors. It is known [11, Chapter IV] that the classical Adams-Bashforth-Moulton method (for first-order equations) is a reasonable and practically useful compromise in the sense that its stability properties allow for a safe application to mildly stiff equations without undue propagation of rounding errors, whereas the implementation does not require extremely time consuming elements. From the results of [15] we can see that

these properties remain unchanged when we look at the fractional version of the algorithm instead of the classical one, and therefore it is also clear that the behaviour does not depend on the order of the differential operators involved.

3.3.2 Convergence

Of course, stability alone is not sufficient in practice to make sure that the numerical solution is a good approximation to the exact solution. We must also address the problem of error estimates, i.e. the question of convergence. In this context, we can use some of the standard analysis techniques [13, §§8.2 & 8.3] to derive that (assuming sufficient smoothness of the functions involved) the convergence order of the scheme is 2, i.e. we have an error bound of the form

$$\max_j |y(t_j) - y_j| = O(h^2) \tag{25}$$

where $h = \max_j(t_{j+1} - t_j)$ and where we assume that all $t_j \in [t_0, t_0 + t^*]$ with some fixed $t^* > 0$.

Second order convergence may seem somewhat slow, but one must keep in mind that most of the parameters of our model (including β, the order of the differential equation itself), and thus also the input values of our algorithm, are material constants that are usually known only up to a very limited accuracy (typically two or three decimal digits). Thus it does not make sense to apply a high order method, especially when we bear in mind that high-order methods frequently show inferior stability properties when compared to low-order methods.

3.3.3 Arithmetic Complexity

As a final remark in this subsection, we point out that the arithmetic complexity (i.e. the number of arithmetic operations) of the algorithm in the form presented so far is not $O(\max_j(t_{j+1} - t_j)^{-1})$ as in a comparable method for first order equations. Instead of this, we find an operation count of $O(\max_j(t_{j+1} - t_j)^{-2})$, as one would expect for a general algorithm for Volterra integral equations. The reason for this high complexity is, as mentioned above, the nature of the differential operator $D^\beta_{t_0}$: When β is not an integer, then we see from eq. (1) that the differential operator is not a local operator, i.e. in order to evaluate $D^\beta_{t_0} y(t)$ it is not sufficient to know the values of f in a small neighbourhood of t. Rather, we need to have information about the entire "history" of the function f on the interval $[t_0, t]$. Although this property stands in sharp contrast to the behaviour of classical (integer-order) differential operators and gives rise to higher complexity (and thus longer run time), it is a highly desirable property for the application we have in mind because it represents, in a mathematical form, the fact that the processes under consideration are history-dependent.

3.4 Modified Versions of the Algorithm

In principle, the algorithm described so far is able to handle the problem under consideration. However, we found it useful to introduce a few modifications in order to improve the performance of the code. It is easily seen that all these modifications (that we shall describe now) are independent of each other. This gives the user the option to select any desired combination of these enhancements when calling the subroutine.

3.4.1 Fading Memory

Probably the simplest and most benifitial modification that one can choose to implement to enhance the overall performance of our numerical integrator is to take advantage of a property known as fading memory. Volterra [21, p. 188] precisely defined the notion of a fading memory, and called it "the principle of the dissipation of hereditary action". More recently, Podlubný [19] referred to Volterra's principle as the "short-memory principle", and has applied it to the first-order Grünwald-Letnikov definition of a fractional-order difference equation. The idea of a fading memory is the simple premise that the contributions to a solution vector coming from the far past in an integration can oftentimes be neglected when compared with the additional contributions arising from the near past; specifically, fading memory supposes that a length L exists such that

$$D_{t_0}^{\beta} y(t) \approx D_{t-L}^{\beta} y(t), \qquad 0 < L \leq t - t_0, \tag{26}$$

where Podlubný refers to L as the memory length.

An examination of the first-order quadrature weights listed in eq. (22) indicates that these weights monotonically decrease (for fixed step size), provided that $0 < \beta < 1$, as the independent variable of integration is made to approach the initial state at t_0 from above. In contrast, these weights monotonically increase, provided that $\beta > 1$, when the independent variable shrinks towards the initial state at t_0. For this reason, the concept of a fading memory has no role to play in fractional-order differential equations whenever $\beta > 1$, since the beginning of the integration interval is weighted more heavily than the end of the interval.

In what follows, the interval of integration $[t_0, t]$ is considered to be broken up into $n + 1$ evenly spaced subintervals, each of length $h > 0$, such that $n \gg 1$. The memory length L will then contain ℓ segments, such that $L = \ell h$ with $0 < \ell \leq n + 1$. A first-order estimate for the error Δ resulting from the fading-memory assumption of eq. (26), and based on the quadrature formula

(24), can be defined by

$$\Delta(t) = |D^{\beta}_{t_0} y(t) - D^{\beta}_{t-\ell h} y(t)| \approx \frac{1}{\Gamma(\beta)} \sum_{j=0}^{n-\ell} b_{j,n+1} \, |f(t_j, y_j)|. \qquad (27)$$

Because of the monotonic nature of these quadrature weights, this error will be bounded from above by the relation

$$\Delta(t) \leq \frac{n-\ell}{\Gamma(\beta)} b_{n-\ell,n+1} M, \qquad 0 < \beta < 1,$$

wherein

$$M = \max_{j=0}^{n-\ell} |f(t_j, y_j)|.$$

Evaluating eq. (23) produces the approximation $b_{\,n-\ell,n+1} \approx h^{\beta}/\ell$ given that $\ell \gg 1$. By constraining this bounding error to be smaller than some specified error tolerance, say $\varepsilon > 0$, the following bound has been obtained for the purpose of quantifying the number of integration subintervals defining a memory length for our integrator, viz.,

$$\frac{\ell}{n} \geq \frac{h^{\beta} M}{h^{\beta} M + \varepsilon \Gamma(\beta)}, \qquad 0 < \beta < 1. \qquad (28)$$

Because $h^{\beta} M$ and $\varepsilon \Gamma(\beta)$ are both positive valued, the concept of a fading memory will always be applicable, in principle, for those cases where β satisfies $0 < \beta < 1$.

The principle of fading memory can be implemented by simply replacing the sum $\sum_{j=0}^{n}$ present in both eqs. (20 & 24) with the sum $\sum_{j=n-\ell+1}^{n}$, where ℓ must satisfy the constraint in the formula above. Although this will result in some loss of accuracy in the solution, due to a truncation of the series, as Podlubný [19] has noted, such a truncation also suppresses the accumulation of roundoff errors, which could otherwise adversely affect the accuracy of solution due to an excessive number of unnecessary addends.

We note, however, that even though the error induced by the truncation is likely to be small from the *numerical* (i.e. practical) point of view, the *analytical* (theoretical) consequence of truncation is that even the very simple equation

$$D^{\beta}_{t_0} y(t) = c, \qquad y(t_0) = y_0, \qquad (29)$$

where c is a given non-zero constant, will not be solved exactly any more. From this consequence, it follows that theoretical error bounds of the form derived in subsection 3.3.2 cannot hold any more, not even if we are willing to accept bounds involving smaller powers of h. In practical calculations, this

not a real problem in view of the fact that these error bounds were derived under the assumption that all calculations are done in exact arithmetic, which is usually violated anyway.

Moreover, a short analysis of eq. (28) shows that, for fixed ϵ and $h \to 0$, the memory length ℓ behaves as $\mathrm{const} \cdot h^{\beta-1}$. Therefore, the overall arithmetic complexity of the unmodified scheme, which is $O(h^{-2})$, is reduced to $O(h^{\beta-2})$. Even though this still is larger than the $O(h^{-1})$ complexity present in similar algorithms for integer-order differential equations, we can speed up the algorithm somewhat.

3.4.2 Additional Corrector Iterations

Recall that in the case of a very stiff equation, we mentioned that the stability properties of the Adams-Bashforth-Moulton integrator may not be sufficient. However, it is well known [10, 11] that the pure one-step Adams-Moulton method (i.e. the trapezoidal method) possesses extremely good stability properties. These are spoiled in the Adams-Bashforth-Moulton approach only by the fact that, in eq. (20) we cannot replace the predictor approximation y^P_{n+1} on the right-hand side by the corrector value y_{n+1} because, in general, we cannot solve that equation exactly any more. The idea is now to find a better approximation for the exact solution of that equation than the rather simple one obtained by applying just one functional corrector iteration with the predictor as a starting value.

There are two main ways to achieve this goal. The first one is to use the value obtained by the first iteration (the first corrector step) as a new predictor and apply another corrector step. Obviously, this procedure can be iterated any given number of times, M say; the resulting method is called a P(EC)^{M}E algorithm. In this way, we find a method that is "closer" to the pure Adams-Moulton technique, and therefore, its stability properties are also closer to the better properties of this method. Taking this idea to the extreme, we could even refrain from stopping after M iterations and continue to iterate until convergence is achieved. This would (theoretically) lead to even better stability, but the computational cost could be prohibitive, and it may even happen that (due to rounding errors) convergence could not be achieved numerically in finite precision. It is possible to make the P(EC)^{M}E approach more efficient by not using the same number M of corrector iterations in every step. In regions of higher stiffness, more steps may be taken in order to retain stability and to keep the error under control; whereas, in regions where stiffness is not a problem, high accuracy may be achieved already with a small choice of M, thus speeding up the algorithm.

In practice, we have implemented this feature in such a way that the user can supply an upper bound for the number M of corrector steps to be taken.

Setting this bound to 1 is then equivalent to switching off this modification completely. Moreover, the user can specify a tolerance $\varepsilon > 0$ to the effect that the iteration is stopped if two consecutive steps give results that differ by less than this tolerance even if the maximum number of iterations is not reached yet.

As mentioned above, this is not the only possible approach to get a better approximation to the solution of the corrector equation. The second idea that we may use to enhance the stability of the method is to find another way of solving eq. (20) with y_{n+1}^P replaced by y_{n+1}. The most obvious way to do this would be to use a Newton iteration. As a starting value, we can still use the Adams-Bashforth predictor. Since it is known that Newton iteration converges faster (locally) than the simple functional iteration of the PECE process, we may expect to come very close to the pure Adams-Moulton method in just a few iterations. However, in order to implement Newton's method, we need to work with the Jacobian of the right-hand side of the differential equation. This can also be a source of numerical problems, and may even lead to very long run times. At present, this feature is not implemented in our code, but a future extension in this way is possible.

Note that, since we keep the second-order Adams-Moulton formula as the basis of eq. (20), the convergence order of these modified algorithms remains at 2. Moreover, this modification of the algorithm does not alter the order of the arithmetic complexity in terms of the step size h. Only the stability is affected by these modifications.

3.4.3 Mesh Spacing

In the standard version of the algorithm, we have implemented a uniformly spaced mesh, i.e. we have chosen the discretization of the basic interval such that the points t_j are given by $t_j = t_0 + jh$ with some fixed h (to be provided by the user). As noted in §3.2, this leads to some simplifications in the formulae, thus reducing the administrative overhead. In particular, the weights $a_{j,n+1}$ and $b_{j,n+1}$ can be expressed in a rather simple manner, allowing us to store them without using much memory or to recalculate them quickly. The use of a non-uniform grid means that, in the $(n+1)$st step, many of the weights used in the nth step cannot be reused and need to be calculated anew at run-time (thus slowing down the method) or a priori (thus increasing the memory requirements). However, it may be that the user is willing to accept these disadvantages because an equispaced grid is undesirable. This is possible, e.g., if some properties of the solution of the differential equation or the right-hand side are known, such as highly non-smooth behaviour, rapid variations or higher stiffness in some, but not in all, parts of the interval of

integration. Therefore, we allow the user to specify an arbitrary mesh and pass it on to the subroutine to replace the default uniform mesh.

References

[1] Blom J. G., Brunner H. (1991) Algorithm 689: Discretized collocation and iterated collocation for nonlinear Volterra integral equations of the second kind. *ACM Trans. Math. Software* **17**, 167–177

[2] Bownds J. M., Appelbaum L. (1985) Algorithm 627: A FORTRAN subroutine for solving Volterra integral equations. *ACM Trans. Math. Software* **11**, 58–65

[3] Caputo M. (1967) Linear models of dissipation whose Q is almost frequency independent, II. *Geophys. J. Royal Astronom. Soc.* **13**, 529–539

[4] Caputo M., Mainardi F. (1971) A new dissipation model based on memory mechanism. *Pure and Applied Geophysics* **91**, 134–147

[5] Diethelm K. (1997) An algorithm for the numerical solution of differential equations of fractional order. *Elec. Transact. Numer. Anal.* **5**, 1–6.

[6] Diethelm K., Freed A. D. (1998) Viscoelastic/viscoplastic material modeling using the fractional calculus. Submitted for publication

[7] Diethelm K., Walz G. (1997) Numerical solution of fractional order differential equations by extrapolation. *Numer. Algorithms* **16**, 231–253

[8] Fischer A., Marais H. (1996) *The Oberon Companion: A Guide to Using and Programming Oberon System 3.* vdf Verlag, Zürich

[9] Gorenflo R. (1997) Fractional calculus: Some numerical methods. In: Carpinteri A., Mainardi F. (Eds.) *Fractals and Fractional Calculus in Continuum Mechanics.* Springer, Wien, 277–290

[10] Hairer E., Nørsett S. P., Wanner G. (1993) *Solving Ordinary Differential Equations I: Nonstiff Problems*, 2nd edn. Springer, Berlin

[11] Hairer E., Wanner G. (1991) *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems.* Springer, Berlin

[12] Krempl E., Bordonaro C. M. (1995) A state variable model for high strength polymers. *Polymer Engineering and Science* **35**, 310–316

[13] Linz P. (1985) *Analytical and Numerical Methods for Volterra Equations.* SIAM, Philadelphia, PA

[14] Lion A. (1997) Constitutive modelling of viscoplastic material behaviour based on evolution equations of fractional order. In: Khan A. S. (Ed.) *Physics and Mechanics of Finite Plastic and Viscoplastic Deformation.* Neat Press, Fulton, MD, 61–62

[15] Lubich C. (1986) Discretized fractional calculus. *SIAM J. Math. Anal.* **17**, 704–719

[16] Mainardi F. (1997) Fractional calculus: Some basic problems in continuum and statistical mechanics. In: Carpinteri A., Mainardi F. (Eds.) *Fractals and Fractional Calculus in Continuum Mechanics.* Springer, Wien, 291–348

[17] National Institute of Standards and Technology: Guide to Available Mathematical Software. http://gams.nist.gov

[18] Oldham K. B., Spanier J. (1974) *The Fractional Calculus.* Academic Press, New York, NY

[19] Podlubný I. (1997) Numerical solution of ordinary fractional differential equations by the fractional difference method. In Elaydi S., Győri I., Ladas G. (Eds.) *Advances in Difference Equations.* Gordon and Breach, Amsterdam, 507–516

[20] Samko S. G., Kilbas A. A., Marichev O. I. (1993) *Fractional Integrals and Derivatives: Theory and Applications.* Gordon and Breach, Yverdon

[21] Volterra V. (1931) *Theory of Functionals: And of integral and integro-differential equations.* Blackie & Son, London
[22] Wirth N., Reiser M. (1992) *Programming in Oberon — Steps Beyond Pascal and Modula-2.* Addison-Wesley, Reading