Computing 71, 305–319 (2003)
Digital Object Identifier (DOI) 10.1007/s00607-003-0033-3

Computing
Printed in Austria

Efficient Solution of Multi-Term Fractional Differential Equations Using P(EC)^{m}E Methods

K. Diethelm, Braunschweig

Received January 24, 2003; revised September 2, 2003
Published online: December 1, 2003

Abstract

We investigate strategies for the numerical solution of the initial value problem $y^{(\alpha_v)}(x) = f(x, y(x), y^{(\alpha_1)}(x), \ldots, y^{(\alpha_{v-1})}(x))$ with initial conditions $y^{(k)}(0) = y_0^{(k)} (k = 0, 1, \ldots, \lceil \alpha_v \rceil - 1)$, where $0 < \alpha_1 < \alpha_2 < \cdots < \alpha_v$. Here $y^{(\alpha_j)}$ denotes the derivative of order $\alpha_j > 0$ (not necessarily $\alpha_j \in \mathbb{N}$) in the sense of Caputo. The methods are based on numerical integration techniques applied to an equivalent nonlinear and weakly singular Volterra integral equation. The classical approach leads to an algorithm with very high arithmetic complexity. Therefore we derive an alternative that leads to lower complexity without sacrificing too much precision.

Mathematics Subject Classification: Primary 65L05; Secondary 65L06, 65L20

Key words: Fractional differential equation, multi-term equation, predictor-corrector method.

1 Introduction

We investigate strategies for the numerical solution of the initial value problem

$$y^{(\alpha_v)}(x) = f(x, y(x), y^{(\alpha_1)}(x), \ldots, y^{(\alpha_{v-1})}(x)), \tag{1}$$

$$y^{(k)}(0) = y_0^{(k)} \qquad (k = 0, 1, \ldots, \lceil \alpha_v \rceil - 1), \tag{2}$$

where $0 < \alpha_1 < \alpha_2 < \cdots < \alpha_v$. Here,

$$y^{(\alpha_j)}(x) := D_*^{\alpha_j} y(x) := J^{\lceil \alpha \rceil - \alpha_j} D^{\lceil \alpha_j \rceil} y(x) \tag{3}$$

denotes the derivative of order $\alpha_j > 0$ (not necessarily $\alpha_j \in \mathbb{N}$) in the sense of Caputo, where, for $q > 0$,

$$J^q y(x) := \frac{1}{\Gamma(q)} \int_0^x (x - t)^{q-1} y(t) \mathrm{d}t$$

is the Riemann-Liouville fractional integral of order q, and where for $m \in \mathbb{N}$, D^m denotes the usual differential operator of order m. Since we have a total of

v differential operators, we call this problem a v-term differential equation. Applications for such equations arise, e.g., in various areas of mechanics [15, 19, 21].

In §2, we review the main properties of the fundamental algorithm by looking at the simplest case, $v = 1$. The algorithm is a predictor-corrector (more precisely, PECE) method introduced in [8, 9] and investigated in a more detailed way in [6, 7]. It can be interpreted in the spirit of the classical Adams-Bashforth-Moulton schemes for first-order equations. Specifically we analyse the discretization error of this approach under various assumptions on the given data. It must be emphasized that other algorithms for certain fractional differential equations are available, but these (like the ones from [2, 10, 18]) typically have a restricted applicability in the sense that they normally encounter difficulties when handling non-linear equations. The Adams method is capable of handling any sort of right-hand side in eq. (1).

On the basis of these results we then (in §3) look at the general problem for arbitrary v. Here we first try to generalize the classical approach for higher order differential equations (with integer α_j). It turns out that this is possible only under some additional number-theoretical assumptions on the α_j. Moreover the resulting systems often have a very large dimension d (whose precise value depends on the exact number-theoretic properties of $\alpha_1, \ldots, \alpha_v$). The computational complexity that is due to this high dimensionality is increased even more by the fact that the special structure of the system forces us to use a very small step size h for the numerical algorithm. As a rule of thumb we find that a reasonable choice for the step size is $h = O(1/d)$.

As an alternative we follow [3] and suggest in §4 to construct an approximate solution in two steps. First we replace the given equation (1) by a different v-term fractional differential equation without changing the initial conditions. This second differential equation is constructed in such a way that it has two main properties:

- Its solution $\tilde{y}$ does not differ significantly from the solution y of the original problem.
- Its structure is such that it can be converted into an equivalent system of one-term fractional differential equations of order α whose dimension is comparatively small.

Then we solve this system numerically with our predictor-corrector algorithm. In view of the properties of the system that are guaranteed by our construction, we find that the computational effort remains reasonably small without sacrificing too much precision. We shall see that it may sometimes be useful to replace the PECE method by a P(EC)^{m}E method with $m \geq 2$. In particular we compare various different choices for the parameter m.

2 The Fundamental Predictor-Corrector Algorithm

In this section we recall the fundamental algorithm that we shall later use to solve our differential equations. It has been introduced in [8] and [9] and investigated further in [6] and [7]. As we shall see in the later sections, it will be sufficient to consider only the case $\nu = 1$ in the initial value problem stated in eqs. (1) and (2). Thus we momentarily concentrate our attention on the differential equation

$$D_*^\alpha y(x) = f(x, y(x)), \tag{4}$$

equipped with initial conditions

$$y^{(k)}(0) = y_0^{(k)}, \qquad k = 0, 1, \ldots, m-1, \tag{5}$$

where $m = \lceil \alpha \rceil$ and the real numbers $y_0^{(k)}$, $k = 0, 1, \ldots, m-1$, are assumed to be given.

The algorithm will be a generalization of the well known second-order Adams-Bashforth-Moulton method for first-order initial value problems [12, 13].

Our approach is based on the analytical property that the initial value problem (4), (5) is equivalent to the Volterra integral equation

$$y(x) = \sum_{k=0}^{\lceil \alpha \rceil - 1} y_0^{(k)} \frac{x^k}{k!} + \frac{1}{\Gamma(\alpha)} \int_0^x (x-t)^{\alpha-1} f(t, y(t)) \mathrm{d}t \tag{6}$$

in the sense that a continuous function is a solution of the initial value problem if and only if it is a solution of (6). For a brief derivation of this equivalence we refer to [4, Lemma 2.3]. Note that the sum outside the integral on the right-hand side is completely determined by the initial values and hence is known. Therefore, instead of solving the originally given initial value problem, we now try to solve this Volterra equation. In typical situations (in particular in the situations that we shall encounter below, i.e. when this equation is constructed from a given multi-term equation) one usually has $0 < \alpha < 1$, and hence the Volterra equation (6) is weakly singular. Moreover we want to admit a very large class of possible right-hand side functions f, and therefore we do not want to assume that the equation is linear.

In order to construct a reasonable approximate solution of this nonlinear and weakly singular Volterra equation, we introduce the equispaced nodes $t_j = jh$ with some given $h > 0$ and simply use the product trapezoidal quadrature formula with these nodes to replace the integral in (6), which of course yields an implicit method. Thus we use it as a corrector in a predictor-corrector pair just as one would do for a first-order equation [12, 13]. In other words, this method can be considered to be an analogue of the classical one-step Adams-Moulton algorithm. Hence our corrector formula is given by

$$y_h(t_{n+1}) = \sum_{k=0}^{\lceil\alpha\rceil-1} \frac{t_{n+1}^k}{k!} y_0^{(k)} + \frac{h^\alpha}{\Gamma(\alpha+2)} f(t_{n+1}, y_h^{\mathrm{P}}(t_{n+1})) + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{j=0}^{n} a_{j,n+1} f(t_j, y_h(t_j)), \tag{7}$$

where

$$a_{0,n+1} = n^{\alpha+1} - (n-\alpha)(n+1)^\alpha,$$
$$a_{j,n+1} = (n-j+2)^{\alpha+1} + (n-j)^{\alpha+1} - 2(n-j+1)^{\alpha+1} \quad (1 \le j \le n). \tag{8}$$

The remaining problem is the determination of the predictor formula that we require to calculate the value $y_h^{\mathrm{P}}(t_{n+1})$. A natural choice for the predictor of a one-step Adams-Moulton rule is a one-step Adams-Bashforth rule, i.e. we replace the integral in eq. (6) by the product rectangle rule with the same nodes as before. This approach gives us the predictor $y_h^{\mathrm{P}}(t_{n+1})$ as

$$y_h^{\mathrm{P}}(t_{n+1}) = \sum_{k=0}^{\lceil\alpha\rceil-1} \frac{t_{n+1}^k}{k!} y_0^{(k)} + \frac{1}{\Gamma(\alpha)} \sum_{j=0}^{n} b_{j,n+1} f(t_n, y_h(t_j)) \tag{9}$$

where now

$$b_{j,n+1} = \frac{h^\alpha}{\alpha}((n+1-j)^\alpha - (n-j)^\alpha) \qquad (0 \le j \le n) \tag{10}$$

(see also [6, 9] for a more detailed derivation of the method). Our basic algorithm, the fractional Adams-Bashforth-Moulton method, is completely described now by eqs. (9) and (7) with the weights $b_{j,n+1}$ and $a_{j,n+1}$ being defined according to (10) and (8), respectively.

The mathematical analysis of this method in [7] shows that we may expect the error to behave as

$$\max_{j=0,1,\ldots,N} |y(t_j) - y_h(t_j)| = O(h^p) \qquad \text{where } p = \min(2, 1+\alpha) \tag{11}$$

and the quantities h and N are related according to $h = T/N$, with T being the upper bound of the interval on which we are looking for the solution. In view of the fact that $1 < p \le 2$ we have a satisfactory globally valid error bound. We note that, quite in contrast to the behaviour of the algorithm described in [2], the convergence order p of the Adams-Bashforth-Moulton scheme increases as α, the order of the differential equation, increases. However, it must be noted that other methods exist (see, e.g., [2]), that have an error bound of the form $O(h^{2-\alpha})$. This is better than the Adams method for $0 < \alpha < 1/2$, and we shall see below that it is very likely that our multi-term equations like (1) give rise to systems with α being very small (like $\alpha = 0.1$ or even smaller). From this point of view the method of [2]

seems preferable, but it has the significant disadvantage of not being readily applicable to non-linear problems. Moreover we shall see below that there are certain other attractive aspects of a predictor-corrector scheme.

We also note that Lubich [16] has proposed to solve the integral equation (6) by a fractional multistep method. This idea gives, in theory, a class of methods with high convergence order; however there are various practical disadvantages associated with this concept. For example, high-order methods require the starting values to be determined with sufficient accuracy, and the construction of suitable numerical methods for this problem is by no means evident. Moreover they require the construction of certain starting weights. For the classical case $\alpha = 1/2$ this is rather simple, but for general α this may be a highly ill-conditioned problem, and for α close to 0 (which is the case that we need to deal with; see below) it is also computationally very expensive because a linear system of dimension $O(1/\alpha)$ needs to be solved at every step. Therefore we prefer the Adams scheme described above in the situation at hand here.

3 Extension to Multi-Term Equations

In order to apply such a method to a multi-term equation, i.e. to an equation of the form (1) with $\nu \geq 2$, we first have to transform the given initial value problem into a system of equations, all of which have the same order. In other words we must construct an initial value problem of the form

$$D_*^q Y(x) = g(x, Y(x)), \qquad Y(0) = Y_0 \tag{12}$$

where now Y_0 is a vector of a suitable dimension d, say, the function Y maps an interval $[0, T]$ to $\mathbb{R}^d$, and $g : [0, T] \times \mathbb{R}^d \to \mathbb{R}^d$. Having done this, we need to establish the connection between the solution Y of this problem and the solution y of the original problem. It turns out (see, e.g., [1, 3, 5, 14]) that such a construction is possible only under certain assumptions on the parameters $\alpha_1, \ldots, \alpha_\nu$; indeed it can be seen that we can perform this conversion

- in the case $\alpha_\nu \geq 1$ if and only if $\alpha_j \in \mathbb{Q}$ for all j, and
- in the case $\alpha_\nu < 1$ if and only if the quotients $\alpha_j/\alpha_k \in \mathbb{Q}$ for all j and k.

(Note that the condition in the latter case is weaker than in the former.) The resulting system is then, as derived in [3], of the form (12), where

$$Y(x) = (y_0(x), y_1(x), \ldots, y_{d-1}(x))^{\mathrm{T}} \tag{13}$$

with $d = \alpha_\nu/q$, where

$$q = \begin{cases} \gcd(\alpha_1, \alpha_2, \ldots, \alpha_\nu) & \text{if } \alpha_\nu < 1, \\ \gcd(1, \alpha_1, \alpha_2, \ldots, \alpha_\nu) & \text{if } \alpha_\nu \geq 1. \end{cases}$$

The gcd terminology has to be used in the generalized sense here because the arguments may be non-integer numbers, i.e. we set

$$\gcd(z_1, \ldots, z_n) := \max\{z \in \mathbb{R} : z_j/z \in \mathbb{N} \text{ for all } j\}.$$

The assumptions above make sure that the required greatest common divisors exist. The function g on the right-hand side of (12) is then given by

$$g(x, Y(x)) = \begin{pmatrix} y_1(x) \\ y_2(x) \\ \vdots \\ y_{d-1}(x) \\ f(x, y_0(x), y_{\alpha_1/q}(x), \ldots, y_{\alpha_{\nu-1}/q}(x)) \end{pmatrix} \tag{14}$$

and the initial conditions have the form

$$y_j(0) = \begin{cases} y_0^{(jq)} & \text{for } jq \in \mathbb{N}_0, \\ 0 & \text{else.} \end{cases} \tag{15}$$

The connection between the given equation (1) and this system is then simply established by noting that the exact solution y of the multi-term equation is identical to the component y_0 of the solution vector of the system.

The analytical approach itself is rather straightforward and does not present major difficulties. However, from the numerical point of view, there are some problems associated with this approach. These problems are essentially created by the structure of the initial conditions as described in (15). In particular, we note that the vector containing the initial conditions consists of the entry $y_0^{(0)}$ in the first component, followed by $1/q - 1$ zeros. Only then the next non-zero entry may appear; its value comes from the given initial values of the original problem, i.e. from eq. (2). In practical applications one often has very small values of q and hence a very large number of zeros in the vector Y_0. The Predict-Evaluate-Correct-Evaluate (PECE) form of the Adams algorithm means that the non-zero elements are propagated by two rows in each step, and hence the algorithm needs to take, roughly speaking, $1/(2q)$ steps before it can produce an approximation for y_0 that differs form the initial value. In other words, if q is small then we need to take a very large number of steps before the numerical solution can leave the initial value and start to follow the exact solution. Assuming that we want to work on a given fixed interval, this means that we need to run the scheme with a very small step size just to overcome this problem, whereas a much larger step size might be sufficient in order to achieve a certain accuracy.

As mentioned at the end of §2, it is possible to solve eq. (12) by Lubich's multistep technique [16], but here we must note that in addition to the practical disadvantages mentioned above, one would encounter the need to solve a nonlinear system of dimension q (a potentially very large number) at every step. Therefore

we shall devote the next sections to some ideas that improve the performance of the predictor-corrector scheme.

4 The Two-Stage Approach

The problem described above is due to the large dimension of the system, and this in turn is a consequence of the size of the greatest common divisor of the orders of the differential operators in the given system. Therefore we have suggested [3] an alternative approach that is based on a two-stage strategy.

The first stage consists of replacing the given equation (1) with initial conditions (2) by a new differential equation

$$\tilde{y}^{(\tilde{\alpha}_v)}(x) = f(x, \tilde{y}(x), \tilde{y}^{(\tilde{\alpha}_1)}(x), \ldots, \tilde{y}^{(\tilde{\alpha}_{v-1})}(x)) \tag{16}$$

with identical initial conditions. We thus perturb the orders of the differential operators, but all other parameters of the given problem (the function f on the right-hand side and the initial conditions) remain unchanged.

The essence of this idea is that, according to the results of [3], the exact solution $\tilde{y}$ of this new initial value problem and the exact solution y of the original problem differ only by

$$\|y - \tilde{y}\|_\infty = O\left(\max_{j=1,2,\ldots,v} |\alpha_j - \tilde{\alpha}_j|\right). \tag{17}$$

Here by $\|\cdot\|_\infty$ we denote the Chebyshev norm taken over a suitable finite interval $[0, T]$, say, where both problems have a solution.

In order to exploit the capabilities of this approach, we need to choose the new parameters $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_v$ in such a way that they have the following three properties:

(a) $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_v \in \mathbb{Q}$,

(b) $\gcd(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_v)$ is large,

(c) $\max_j |\alpha_j - \tilde{\alpha}_j|$ is small.

Here condition (a) asserts that a conversion of eq. (16) to a single-term system (as described in §3) is possible. Specifically, since only the new values $\tilde{\alpha}_j$ enter the later stages of the scheme, such a conversion is always possible, without any restrictions on the original values α_j. Condition (b) makes sure that the dimension d of this system is small (remember that in §3 we had seen that essentially $d \sim 1/q$ where q is the greatest common divisor mentioned in condition (b)). The significance of this property lies in the reduction of the computational complexity. Condition (c) finally makes sure that the error introduced by this perturbation remains small, cf. eq. (17).

It must be noted of course that there is a conflict between conditions (b) and (c): In many cases it will be possible to improve the approximation required in (c) at

the price of decreasing the gcd mentioned in (b). A proper compromise must be found in this case. It seems to be impossible however to state a generally valid strategy for the solution of this conflict; a good compromise will likely depend on the specific parameters of the equation under consideration.

This completes the first stage of the algorithm. At the end of this stage we have found a new initial value problem that consists of the perturbed differential equation (16) together with the original (unperturbed) initial conditions (2).

The second stage of the algorithm is then the stage where the initial value problem that was constructed in stage 1 will be solved numerically. In practice we will first use the approach of §3 to convert the new initial value problem into a single-term system, and then we will solve this system numerically (for example by means of the Adams method described in §2). As pointed out at the end of §3 the large number of zeros in the initial condition of the resulting system may force us to use a very small step size. Alternatively it may be useful to replace the plain PECE structure by a $P(EC)^mE$ method, i.e. by introducing additional corrector iterations. This would allow for a quicker propagation of the non-zero elements, and it may be possible to avoid the use of excessively small step sizes. We shall provide some numerical examples in §5. This flexibility in the number of corrector steps is actually one of the main reasons why we suggest the Adams scheme and not, e.g., the method of [2].

5 A Numerical Example

We want to illustrate the properties of our scheme by using the example

$$y^{(1.455)}(x) = -x^{0.1}\frac{E_{1.545}(-x)}{E_{1.445}(-x)}e^{x}y(x)y^{(0.555)}(x) + e^{-2x} - [y^{(1)}(x)]^2, \tag{18}$$

$$y(0) = 1, \quad y'(0) = -1, \tag{19}$$

where E_μ denotes the Mittag-Leffler function of order μ, defined by

$$E_\mu(x) = \sum_{j=0}^{\infty}\frac{x^j}{\Gamma(j\mu+1)}.$$

Obviously this is a nonlinear three-term equation with $\alpha_1 = 0.555$, $\alpha_2 = 1$ and $\alpha_3 = 1.455$. The initial conditions are chosen such that the exact solution is

$$y(x) = e^{-x}.$$

We want to look at this equation on the interval [0, 1]. All calculations were done on a 500 MHz Pentium based PC in double precision arithmetic.

As a first attempt to solve the problem with the two-stage strategy of §4, we have approximated this equation by

$$\tilde{y}^{(1.5)}(x) = -x^{0.1}\frac{E_{1.545}(-x)}{E_{1.445}(-x)}e^{x}\tilde{y}(x)\tilde{y}^{(0.5)}(x) + e^{-2x} - [\tilde{y}^{(1)}(x)]^2, \tag{20}$$

converted (20) to a three-dimensional system of order $q = 0.5$, and solved this system numerically with the Adams method in its plain PECE form using various step sizes. The results are described in Table 1.

We can see that there is almost no improvement when we change the step size from 1/20 to 1/40. This indicates that the error of the Adams scheme (i.e. the error introduced in the second stage) is already very small compared to the error of the first stage. Therefore there is no need to look for an improved scheme for the solution of this simple system. Note in particular that (see Figure 1) even the crudest of these three approximations (the dashed line) gives a *qualitatively* correct picture of the exact solution (the solid line).

In order to obtain a better approximation with our method we must now reduce the error of stage 1, i.e. we need to introduce smaller perturbations in the orders of the differential operators. We thus try to approximate the given equation (18) not by (20) but by

$$\tilde{y}^{(1.45)}(x) = -x^{0.1}\frac{E_{1.545}(-x)}{E_{1.445}(-x)}e^{x}\tilde{y}(x)\tilde{y}^{(0.55)}(x) + e^{-2x} - [\tilde{y}^{(1)}(x)]^2 \tag{21}$$

and proceed as above. Consequently we find that we have to solve a 29-dimensional system of order 0.05 numerically. This task is (in particular due to the nature of the initial conditions) much more difficult than the previous one, and

Table 1. Numerical results of first approximation ($d = 3$, $q = 0.5$).

Step size	Maximal error	Run time
1/10	0.136	0.07 s
1/20	0.124	0.18 s
1/40	0.118	0.56 s

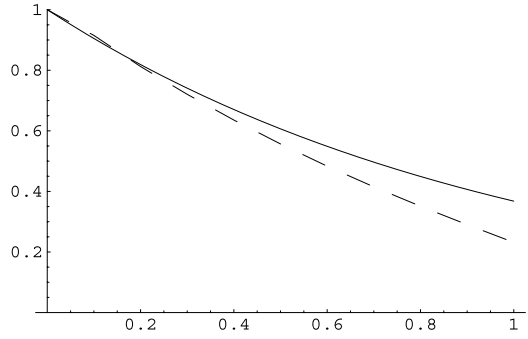

Fig. 1. Exact solution and first approximation ($d = 3$, $q = 0.5$, step size $h = 0.1$)

therefore we need to put more effort into the numerical scheme. This means that we have to use a smaller step size or more corrector iterations. The results for smaller step sizes are given in Table 2.

For the purpose of comparison with the previous example we have included the case of a step size of 1/40. As can be seen by comparing Tables 1 and 2, the error is much larger now than it was before. The reason is the problem that we mentioned above: Since the dimension of the system has been increased, the numerical solution needs more time to get away from the initial value. An even more obvious picture of the situation appears when we look at the graphical data provided in Figure 2. Here again the solid line is the exact solution, the other lines correspond to the numerical solutions (dashed line: $h = 1/40$; dash-dotted line: $h = 1/100$; dotted line: $h = 1/200$). We thus have to say that the graph for $h = 1/40$ does *not* give a qualitatively correct picture of the true solution.

Alternatively we may replace the PECE scheme by a $\mathrm{P(EC)}^m\mathrm{E}$ scheme, thus introducing more corrector iterations. Then we can reduce the error without using smaller step sizes. The point here is that by, for example, doubling the number of corrector iterations, we essentially leave the computational complexity unchanged: A corrector iteration is of the form

$$y_h^{[\ell]}(t_{n+1}) = \sum_{k=0}^{\lceil\alpha\rceil-1} \frac{t_{n+1}^k}{k!} y_0^{(k)} + \frac{h^\alpha}{\Gamma(\alpha+2)} f(t_{n+1}, y_h^{[\ell-1]}(t_{n+1})) + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{j=0}^{n} a_{j,n+1} f(t_j, y_h(t_j)),$$

Table 2. Numerical results of second approximation ($d = 29$, $q = 0.05$).

step size	maximal error	run time
1/40	0.2015	0.9 s
1/100	0.0861	5.5 s
1/200	0.0440	21.5 s
1/400	0.0222	82.4 s

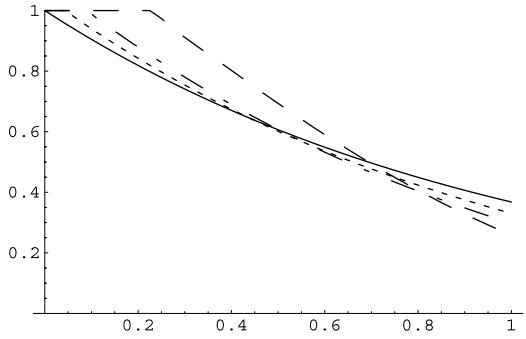

Fig. 2. Exact solution and second approximation ($d = 29$, $q = 0.05$, various step sizes)

cf. eq. (7). Here $y_h^{[\ell]}(t_{n+1})$ denotes the approximation after ℓ corrector steps, $y_h^{[0]}(t_{n+1}) = y_h^{\mathrm{P}}(t_{n+1})$ is the predictor, and $y_h(t_{n+1}) := y_h^{[m]}(t_{n+1})$ is the final approximation after m corrector steps that we actually use. We can rewrite this as

$$y_h^{[\ell]}(t_{n+1}) = \gamma_{n+1} + \frac{h^\alpha}{\Gamma(\alpha+2)} f(t_{n+1}, y_h^{[\ell-1]}(t_{n+1}))$$

where

$$\gamma_{n+1} = \sum_{k=0}^{\lceil\alpha\rceil-1} \frac{t_{n+1}^k}{k!} y_0^{(k)} + \frac{h^\alpha}{\Gamma(\alpha+2)} \sum_{j=0}^{n} a_{j,n+1} f(t_j, y_h(t_j))$$

is independent of ℓ. Thus the total arithmetic complexity of the corrector part of the nth step (taking us from t_{n-1} to t_n) is $O(n)$ for the calculation of γ_n plus $O(m)$ for the m corrector steps, which (since m is constant) is asymptotically the same as the complexity in the case $m = 1$.

If we would use the other option and reduce the step size by a factor of two, then the run time would increase by a factor of four because the complexity of the algorithm is $O(h^{-2})$. Both approaches would reduce the size of the initial interval where the numerical solution gets stuck at the initial value by a factor of $1/2$.

It is possible to use the approach of Ford and Simpson [11] to reduce the complexity of the basic algorithm to $O(h^{-1} \ln h^{-1})$, but not more; we have refrained from doing so because (a) this is still slower than an increase of the number of corrector steps, and (b) it would have introduced an additional computational overhead.

The data obtained by our P(EC)^{m}E approach are given in Table 3. Note that the data of Table 2 correspond to this method with $m = 1$.

It is clearly seen that there is a significant advantage in this approach: By choosing $m = 10$ and $h = 1/40$ for example, we obtain an absolute error that is about 25% smaller than in the case $m = 1$ and $h = 1/200$, and at the same time the run time is 75% shorter. The reason is the following. In the case $m = 1$ the numerical solution gets stuck at the initial value for a rather long interval. At the end of this interval the true solution has moved away significantly from the initial value, and here the error attains its maximum. Over the remainder of the interval $[0, 1]$ the numerical solution then has to creep towards the exact solution, and the error gets smaller. If we choose a larger value for m, we make the problematic initial interval smaller,

Table 3. Numerical results of second approximation ($d = 29$, $q = 0.05$), with P(EC)^{m}E algorithm.

Number of corrector iterations	Step size	Maximal error	Run time
10	1/40	0.03175	1.3 s
10	1/100	0.01174	6.5 s
20	1/40	0.00989	1.6 s
20	1/100	0.00379	7.2 s

and therefore we also diminish the error attained over this interval. This is apparent from Figure 3 where we have compared the absolute errors for $m = 1$, $h = 1/200$ (solid line) and $m = 10$, $h = 1/40$ (dashed line).

Finally we note that all orders of the differential operators in eq. (18) are rational, and so we may skip stage 1 of our two-stage process (thus effectively approximating the given equation by itself) and continue with stage 2 as usual. This gives rise to a 291-dimensional system of order $q = 0.005$. Some numerical results for the plain PECE method are given in Table 4.

It is clear that the run times are not competitive. Therefore we once again revert to the $P(EC)^mE$ structure with larger values for m and larger step sizes as before. Some results are stated in Table 5.

Comparing Tables 4 and 5 we once again find a significant run time advantage in the $P(EC)^mE$ method as compared to the PECE method without losing accuracy, but even the approximations obtained by the faster $P(EC)^mE$ approach are less

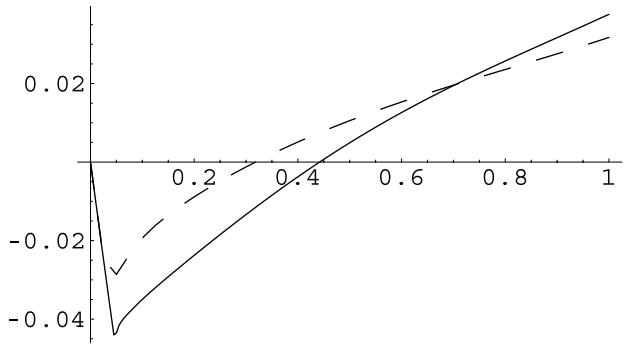

Fig. 3. Errors for second approximation ($d = 29$, $q = 0.05$, various combinations of step size and number of corrector steps)

Table 4. Numerical results for unperturbed equation ($d = 291$, $q = 0.005$), with PECE algorithm.

Step size	Maximal error	Run time
1/200	0.3904	101.2 s
1/400	0.2193	368.4 s
1/800	0.1164	1358.0 s
1/1600	0.0600	5017.4 s

Table 5. Numerical results for unperturbed equation ($d = 291$, $q = 0.005$), with $P(EC)^mE$ algorithm.

Number of corrector iterations	Step size	Maximal error	Run time
10	1/100	0.16473	24.6 s
10	1/200	0.08607	85.4 s
20	1/100	0.08607	25.4 s
20	1/200	0.04400	86.1 s
200	1/120	0.03880	9.5 s
200	1/100	0.00430	58.5 s
200	1/200	0.00226	154.8 s

accurate and more time consuming than the results presented in Table 3 where we had used a simpler differential equation system.

6 Full Description of a Possible Algorithm

Based on our theoretical considerations above and on heuristical arguments coming from the numerical results, we now give a complete description of our algorithm for the approximate solution of the initial value problem (1), (2). The algorithm will follow the basic ideas outlined above. The fundamental concept is that we assume a bound on the complexity to be given (expressed in terms of the gcd of the orders $\tilde{\alpha}_j$) and that we try to achieve a high accuracy in the solution without exceeding the complexity limit.

Specifically, we assume that the user specifies a parameter $q \in \mathbb{Q}$ which we interpret as a lower bound for $\gcd(1, \tilde{\alpha}_1, \ldots, \tilde{\alpha}_v)$. Since the dimension d of the system that we shall construct in stage 2 of the algorithm is given by $d = \tilde{\alpha}_v/q \approx \alpha_v/q$, this data gives us an upper bound on the dimension and hence an upper bound on the arithmetic complexity.

We begin by constructing the perturbations required for the first stage. This is very simple; for $j = 1, 2, \ldots, v$ we only have to set $\tilde{\alpha}_j := \beta_j q$ where $\beta_j \in \mathbb{N}$ is chosen to be the natural number closest to α_j/q (i.e. $\beta_j = \lfloor \alpha_j/q + 0.5 \rfloor$). In this way we make sure that, for every single j, the quantity $|\tilde{\alpha}_j - \alpha_j|$ is minimized under the condition that $\gcd(1, \tilde{\alpha}_1, \ldots, \tilde{\alpha}_v) \geq q$. This essentially completes the first stage.

The second stage begins by rewriting the perturbed equations as a system of order q and dimension d as described in §3. This system is solved by the P(EC)^{m}E scheme indicated in §§4 and 5. To avoid the problems caused by the large number of zeros in the new initial condition, we choose the parameter m in a way that depends on the number of zeros (i.e. on q); specifically we set $m := \lceil 1/q \rceil$. Note that it follows from the considerations in §5 that it is neither necessary nor helpful to introduce additional flexibility by choosing different values for the parameter m in each step. The choice that we propose here is sufficient to avoid the problems caused by the (possibly) large number of zeros in the initial condition. Choosing m larger than this would not give a better order of accuracy, so there is no point in doing that (cf. the considerations on eq. (22) below). Choosing m smaller (permanently or temporarily) would mean that the problem cannot be avoided totally, so one would have to assume a deterioration of the approximation quality, but on the other hand it would not lead to a significantly faster algorithm because the arithmetic complexity of the entire scheme is (asymptotically) independent of m.

Another advantage of the P(EC)^{m}E scheme with the choice of m indicated above can be explained by a careful look at the error analysis of the plain PECE scheme in [7] combined with the standard error analysis of P(EC)^{m}E schemes for first-order equations (cf., e.g., [20]): The algorithm converges to the true solution of the perturbed equation with an error of

$$\max_{j=0,1,\ldots,N} |y(t_j) - y_h(t_j)| = O(h^p) \quad \text{where } p = \min(2, 1 + qm). \tag{22}$$

In the special case of the PECE scheme (i.e. $m = 1$) this reduces to the observation of eq. (11). Since we advocate to use $m = \lceil 1/q \rceil \geq 1/q$, this means that we actually have $p = 2$ in every case, so we find slightly better convergence behaviour than in the simple PECE approach; indeed this is the maximum order than one can possibly obtain by an algorithm that uses the approximation method underlying our scheme. As noted above, in many applications of interest our approach is likely to give q very close to 0, and then the $O(h^2)$ convergence obtained in this way is a significant advantage compared to the $O(h^{1+q})$ behaviour of the plain PECE method.

Our approach is particularly useful when one is looking for a computationally inexpensive but still reasonably accurate approximation. In many applications this will be what is desired because often one needs to solve a great number of such initial value problems whose solutions are then required as input data for other problems. Additionally, high accuracy is frequently impossible to obtain anyway because the given data (in particular the orders α_j of the differential operators) are something like material constants known only up to a certain (usually moderate) precision.

7 Conclusion

We have shown that the two-stage method of Diethelm and Ford [3] can be a very useful and efficient tool for the approximate solution of initial value problems involving multi-term fractional differential equations. In order to fully exploit the capabilities of the approach it seems to be useful to tune the parameters according to the specific requirements of the problem under consideration. In particular, one should combine the approach with a $\mathrm{P(EC)}^m\mathrm{E}$ algorithm, i.e. an algorithm that uses more than just one corrector step, if the order of the single-term system constructed in the process is small. The precise choice of the parameters of the second stage (step size and number of corrector steps) thus must depend on the choice of the parameters in the first stage (the approximate orders of the differential operators).

References

[1] Bagley, R. L., Calico, R. A.: Fractional order state equations for the control of viscoelastically damped structures. J. of Guidance, Control and Dynamics *14*, 304–311 (1991).
[2] Diethelm, K.: An algorithm for the numerical solution of differential equations of fractional order. Elec. Transact. Numer. Anal. *5*, 1–6 (1997).
[3] Diethelm, K., Ford, N. J.: Multi-order fractional differential equations and their numerical solution. Appl. Math. Comput., to appear
[4] Diethelm, K., Ford, N. J.: Analysis of fractional differential equations. J. Math. Anal. Appl. *265*, 229–248 (2002).
[5] Diethelm, K., Ford, N. J.: Numerical solution of the Bagley-Torvik equation, *BIT 42*, 490–507 (2002).

[6] Diethelm, K., Ford, N. J., Freed, A. D.: A predictor-corrector approach for the numerical solution of fractional differential equations. Nonlinear Dynamics *29*, 3–22 (2002).
[7] Diethelm, K., Ford, N. J., Freed, A. D.: Detailed error analysis for a fractional Adams method, submitted for publication.
[8] Diethelm, K., Freed, A. D.: On the solution of nonlinear fractional differential equations used in the modeling of viscoplasticity. In Keil, F., Mackens, W., Voß, H., Werther, J., (eds.) "Scientific Computing in Chemical Engineering II – Computational Fluid Dynamics, Reaction Engineering, and Molecular Properties", pp. 217–224, Heidelberg: Springer 1999.
[9] Diethelm, K., Freed, A. D.: The FracPECE subroutine for the numerical solution of differential equations of fractional order. In Heinzel, S., Plesser, T., (eds.) "Forschung und wissenschaftliches Rechnen 1998", pp. 57–71, Göttingen: Gesellschaft für wissenschaftliche Datenverarbeitung 1999.
[10] Diethelm, K., Luchko, Y.: Numerical solution of linear multi-term initial value problems of fractional order. J. Comput. Anal. Appl., to appear.
[11] Ford, N. J., Simpson, A. C.: The numerical solution of fractional differential equations: Speed versus accuracy. Numer. Algorithms *26*, 333–346 (2001).
[12] Hairer, E., Nørsett S. P., Wanner, G.: "Solving ordinary differential equations I: Nonstiff problems". 2nd revised ed, Berlin: Springer 1993.
[13] Hairer, E., Wanner, G.: "Solving ordinary differential equations II: Stiff and differential-algebraic problems". Berlin: Springer 1991.
[14] Hartley, T. T., Lorenzo, C. F.: The vector linear fractional initialization problem, NASA Technical Publication 1999-208919, NASA Glenn Research Center, Cleveland, 1999.
[15] Koeller, R. C.: Polynomial operators, Stieltjes convolution, and fractional calculus in hereditary mechanics. Acta Mech. *58*, 251–264 (1986).
[16] Lubich, C.: Fractional linear multistep methods for Abel-Volterra integral equations of the second kind. Math. Comp. *45*, 463–469 (1985).
[17] Luchko, Y., Gorenflo, R.: An operational method for solving fractional differential equations with the Caputo derivatives. Acta Math. Vietnamica *24*, 207–233 (1999).
[18] Nkamnang, A. R.: "Diskretisierung von mehrgliedrigen Abelschen Integralgleichungen und gewöhnlichen Differentialgleichungen gebrochener Ordnung". Ph.D. thesis, Berlin: Freie Universität Berlin 1999.
[19] Podlubny, I.: "Fractional differential equations". San Diego: Academic Press 1999.
[20] Strehmel, K., Weiner, R.: "Numerik gewöhnlicher Differentialgleichungen". Stuttgart: Teubner 1995.
[21] Torvik P. J., Bagley R. L.: On the appearance of the fractional derivative in the behavior of real materials. J. Appl. Mech. 51, 294–298 (1984).

K. Diethelm
Institut für Angewandte Mathematik
Technische Universität Braunschweig
Pockelsstraße 14
38106 Braunschweig
Germany
e-mail: k.diethelm@tu-bs.de