COMMUNICATIONS ON
PURE AND APPLIED ANALYSIS
Volume **5**, Number **2**, June **2006**

Website: http://AIMsciences.org

pp. **289–307**

COMPARISON OF NUMERICAL METHODS FOR FRACTIONAL DIFFERENTIAL EQUATIONS

Neville J. Ford and Joseph A. Connolly

Department of Mathematics
University of Chester
Parkgate Road, Chester CH1 4BJ, UK

Abstract. In this paper we present a comparison of numerical methods for the solution of single term fractional differential equations. We review five available methods and use a graphical technique to compare their relative merits. We conclude by giving recommendations on the choice of efficient methods for any given single term fractional differential equation.

1. **Introduction.** Over the past decade the development of numerical methods for the solution of equations containing derivatives and integrals of non-integer order has become a hot topic. There have been several simple algorithms published for producing approximate solutions for fractional differential equations. Despite the interest these algorithms have generated among mathematical modellers and applied scientists from several fields there seems to have been no systematic comparison of their relative merits from an applications viewpoint to date. Much theoretical work has been completed already, but as the recent paper [9] indicates, there can be a considerable gap between methods that perform well in theory and those whose practical implementations are effective. For existing theoretical work we refer to the recent works by authors such as Diethelm, Ford, Luchko and Podlubny (see [7], [10], [11], [12], [14], [21], [23].)

The difficulty of transferring the theoretical good properties of a method to actual performance in practice is reviewed more thoroughly later. In this paper we have chosen to assess the performance of the methods through actual implementation to solve certain fractional differential equations (whose exact solutions are known, thus facilitating the desired comparisons to be made). In a sequel to the present paper we shall present corresponding conclusions relating to multi-term equations. For the present we simply draw attention to the following conclusion from our experiments: it is *not* reasonable to assume that a method that is best for a single term equation is necessarily also best for solving an *apparently similar* multi-term problem.

To be precise, this paper uses the single-term fractional differential equation

$$D^{\alpha}y(t) = \lambda y(t) + g(t) \tag{1.1}$$

as the basis for our experiments. In line with recent work [10], [11], the differential operator D is assumed to be of the Caputo type, because this seems to be more widely applicable in modern modelling applications. Inhomogeneous initial conditions then can be specified, if desired, in terms of integer order derivatives.

2000 *Mathematics Subject Classification.* Primary: 65L05, 65R20; Secondary: 65L70.
Key words and phrases. Numerical methods, efficiency, fractional differential equations.

2. **Basic Ideas and Definitions.** Single term fractional differential equations take the general form

$$D^\alpha y(t) = f(t, y(t)), \qquad D^{(k)}(y_0) = y_0^{(k)} \quad (k = 0, 1, \ldots, \lceil \alpha \rceil - 1), \tag{2.1}$$

where α is some positive non-integer number. Here the notation D^α is used for the Caputo type fractional derivative, defined by

$$D^\alpha y(t) := \frac{1}{\Gamma(m-\alpha)} \int_0^t (t-\tau)^{m-\alpha-1} y^{(m)}(\tau)\, \mathrm{d}\tau$$

where $m := \lceil \alpha \rceil$.

One can also define the Caputo fractional derivative based on classical Riemann-Liouville differential operators of fractional order $\alpha > 0$ which are defined by,

$$D_*^\alpha y(t) := \frac{1}{\Gamma(m-\alpha)} \frac{d^m}{dt^m} \int_0^t \frac{y(u)}{(t-u)^{\alpha-m+1}} du$$

where m is the integer defined by $m - 1 < \alpha < m$. The standard (Riemann-Liouville) approach, is then to define the initial conditions for solving the fractional differential equation in the form

$$\frac{d^{\alpha-k}}{dt^{\alpha-k}} y(t)|_{t=0+} = b_k, \qquad k = 1, 2, \ldots, m = \lfloor \alpha + 1 \rfloor,$$

with given values b_k. This means we need to specify initial values in terms of (inconvenient) fractional derivatives of the function y. To avoid this inconvenience, Caputo [4] suggested incorporating the classical derivatives (of integer order) of the function y, as they are commonly used in initial value problems with integer-order equations, into the fractional-order equation, giving the alternative (equivalent) formulation of the Caputo fractional differential equation as

$$D^\alpha y(t) := D_*^\alpha (y - T_{m-1}[y])(t) = f(t, y(t)), \tag{2.2}$$

where $T_{m-1}[y]$ is the Taylor polynomial of order $(m-1)$ for y, centered at 0.

3. **How Should We Judge a Good Numerical Method?** The *effectiveness* of numerical methods is classically considered using ideas such as the convergence, consistency and stability of the method. Of these, the consistency *order* of the method is often regarded as a benchmark for comparing methods. The so-called order of the method (we talk about methods of order p when the error can be shown to have $\mathcal{O}(h^p)$ as $h \to 0$ for step length $h > 0$) helps inform the user about the rate at which the error in an application of a fixed step length method over a fixed interval $[0, T]$ will decrease as a result of decreasing the step length. Thus a method of order 1 should (speaking loosely) see a halving of the error when the step length is halved, while the use of a method of order 2 would lead to the error being multiplied by a factor of $\frac{1}{4}$ as the step length is halved. However the orders quoted are asymptotic results and apply only as the step length $h \to 0$ so the errors observed in practice may be better or worse than the errors predicted. As a further observation one needs to understand that the order of the method gives an error *bound* over *some fixed time interval*. There is a constant that depends on the method and the equation (and that grows rapidly as the length T of the interval increases) that determines the size of the actual error. Therefore two methods with the same order could have extremely different errors when they are applied. (This would happen if the two constants were quite different). There is also a further complication: the order of a

method will typically only be obtained subject to sufficient smoothness conditions imposed either on the function $f(t, y(t))$ on the right hand side of the equation, or on the solution y itself. Obviously it is more convenient for the conditions to be imposed on the equation, rather than on its solution, but the analysis often becomes tractable only with conditions on the smoothness of the solution. See [12] for a fuller discussion of this.

In this paper we shall use examples whose solutions are twice continuously differentiable. This is motivated by the desire to choose problems for analysis where the standard convergence results already available apply for each of the algorithms under comparison.

Alongside the effectiveness of the numerical method is a requirement to keep the computational *cost* low. Traditionally computer time itself was expensive. However now that is no longer the case and cost needs to be measured more in terms of how long it will take for an algorithm to be implemented (programmed) and then (usually more significantly) how long one must wait for the computations before the solution is available. This last point can be really important if the application involves a lot of computation. Sometimes the results are available too late to be of use.

One measures the *computational cost* of a method in terms either of the actual time spent completing the calculation or in terms of the number of *FLOPS* (Floating Point Operations) needed to complete the calculation. Both methods of calculating have their advantages and disadvantages. Measurement of time depends on other loading on the computer system and therefore may vary from run to run, while the counting of FLOPS does not (for the same reason) allow a direct conversion to predict the time that the program will take to run. In this paper we have chosen to use FLOPS as the more consistent method of counting the work required in running the program. We talk about a method having computational cost $\mathcal{O}(n^q)$ to show the effect of changing $n = \frac{1}{h}$ on the work required by the algorithm. Of course it comes as no surprise to find that high order methods typically also require more work!

Surprisingly little has been done in the past to try to calculate the true *efficiency* of the numerical methods. Obviously the user wants a reliable method that produces the lowest errors in the shortest run time. In other words we need to compare the errors with the computational cost.

We introduce a graphical representation of efficiency by plotting the 'log of Error' against the 'log of FLOP count', for a sequence of decreasing step lengths. This produces a line for each method, which we have found can be extrapolated with reasonable accuracy. Where the lines representing two methods cross, this highlights a critical combination of error and work where the more efficient method for the problem changes.

The results of our experiments show that the critical values where these changes occur depend upon the order of the problem α and the function g. We show how, for a particular equation, one could use our approach to ensure the appropriate method was selected. However, probably more usefully, we are able to make general recommendations on the choice of *usually optimal* or *nearly optimal* methods.

4. **The Methods.**

We have chosen to compare the merits of 5 methods that have attracted attention recently. None of the methods are high order and this might

arouse some suspicion. However the conclusions of the recent paper [9] cast considerable doubt on the value of implementing high order methods for problems such as these. Indeed, in our experience, applications experts generally apply one of the numerical schemes we have considered here in preference to any available higher order scheme. We therefore consider the following methods:

1. The Implicit Quadrature method, introduced by Diethelm [7] (see also [5])
2. The Predictor Corrector method, more fully discussed by Diethelm, Ford and Freed [11]
3. The Approximate Mittag-Leffler method, considered by Diethelm and Luchko [14]
4. A Collocation method, described by Blank [1]
5. The Finite Differences method, discussed by Gorenflo [15]

The pseudo-code implementation of methods 1-3 can be found in [13]. All calculations were performed using double precision arithmetic in Matlab.

Remark 4.1. *There have been several alternative methods proposed over the years for the solution of fractional differential equations. Two such approaches are the use of product quadratures (see* [3]*,pp. 366 ff. and the references contained therein) and Galerkin-type methods (see, for example,* [16]*). For our purposes here we have confined ourselves to those methods that we know have been adopted recently in applications. For further reading and a discussion of several alternative approaches, we would recommend* [25, 17, 20, 2, 26]

4.1. **Implicit Quadrature, K. Diethelm** [7]**.** Diethelm [7] introduced a numerical scheme which appears to use the Riemann-Liouville fractional derivative in defining a backward difference formula method. Upon closer inspection, by incorporating the initial condition $y(0) = y_0$ into the equation, the problem has been transformed into a fractional differential equation of Caputo type (2.2).

By interchanging differentiation and integration of the Riemann-Liouville Fractional Derivative we obtain,

$$D^{\alpha}y(t) = \frac{1}{\Gamma(-\alpha)}\int_0^t \frac{y(u)}{(t-u)^{\alpha+1}}du, \tag{4.1}$$

where the integral is interpreted in a Hadamard finite-part sense.

We introduce some notation that we shall use for each numerical scheme. We set a fixed step length $h > 0$ and use y_j as a representation of the approximation of $y(jh)$ at integer multiples of the step length. An equispaced grid $t_j = j/n = nh$ on the solution interval is set. For $j = 1, 2, \ldots, n$, we obtain,

$$g(t_j) + \lambda y(t_j) = \frac{1}{\Gamma(-\alpha)}\int_0^{t_j} \frac{y(u)-y(0)}{(t_j-u)^{\alpha+1}}du = \frac{t_j^{-\alpha}}{\Gamma(-\alpha)}\int_0^1 \frac{y(t_j - t_j w)-y(0)}{w^{\alpha+1}}dw.$$

Replacing the integral by a compound quadrature formula [8] with equispaced nodes $0, 1/j, 2/j, \ldots, 1$ for each j, gives

$$Q_j[v] = \sum_{i=0}^{j} w_{ij} v(i/j) \approx \int_0^1 v(u)u^{-\alpha-1}du$$

with remainder term

$$R_j[v] = \int_0^1 v(u)u^{-\alpha-1}du - Q_j[v].$$

The following implicit formula gives Diethelm's numerical scheme for the solution of equation (1.1):

$$y_j = \frac{1}{w_{0j} - (j/n)^{\alpha}\Gamma(-\alpha)\lambda}\left(\left(\frac{j}{n}\right)^{\alpha}\Gamma(-\alpha)\,g(t_j) - \sum_{i=1}^{j} w_{ij}y_{j-i} - \frac{1}{\alpha}y_0\right), \quad (4.2)$$

where the weights w_{ij} are given by

$$\alpha(1-\alpha)\,j^{-\alpha}w_{ij} = \begin{cases} -1 & \text{for } i = 0, \\ 2i^{1-\alpha} - (i-1)^{1-\alpha} - (i+1)^{1-\alpha} & \text{for } i = 1, 2, \ldots, \text{j-1}, \\ (\alpha-1)i^{\alpha} - (i-1)^{1-\alpha} + i^{1-\alpha} & \text{for } i = j. \end{cases}$$

Diethelm gives the result that when using functions that are sufficiently smooth the error behaves as $O(h^{2-\alpha})$ (see, for example, [7], [11]).

4.2. **Predictor Corrector, K. Diethelm, N. J. Ford and A. D. Freed** [11]**.** Diethelm et al. [11] introduced an algorithm of $P(EC)^M E$ (Predict-Evaluate-Correct-Evaluate) type, for the solution of the non-linear FDE,

$$D^{\alpha}y(t) = f(t, y(t)), \qquad 0 \le t \le 1, \quad (4.3)$$

where $0 < \alpha < m, m = \lceil \alpha \rceil$ subject to the initial conditions,

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

The algorithm can be used to solve equation (1.1), although obviously these predictor-corrector schemes are particularly appropriate for nonlinear equations. Later we will compare the computational efficiency of the $P(EC)^M E$ algorithm directly with the other single-term FDE methods.

The method is derived as follows: Equation (4.3) is equivalent to the Volterra Integral Equation

$$y(t) = \sum_{k=0}^{m-1} \frac{t^k}{k!}y_0^{(k)} + \frac{1}{\Gamma(\alpha)}\int_0^t (t-u)^{\alpha-1}f(u, y(u))du \quad (4.4)$$

We can replace the integral in equation (4.4) by the product rectangle rule

$$\int_0^{t_{j+1}} (t_{j+1} - u)^{\alpha-1}f(u)du \approx \sum_{i=0}^{j} b_{i,j+1}f(t_i)$$

where,

$$b_{i,j+1} = \frac{h^{\alpha}}{\alpha}((j+1-i)^{\alpha} - (j-i)^{\alpha}). \quad (4.5)$$

The predictor y_{j+1}^P is determined by the fractional Adams-Bashforth method,

$$y_{j+1}^P = \sum_{k=0}^{m-1} \frac{t_{j+1}^k}{k!}y_0^{(k)} + \frac{1}{\Gamma(\alpha)}\sum_{i=0}^{j} b_{i,j+1}f(t_i, y_i). \quad (4.6)$$

with weights (4.5).

The corrector formula is the fractional version of the Adams-Moulton method, which can be single or multi-step. Using standard results from quadrature theory the integral in equation (4.4) can be approximated by

$$\int_0^{t_{j+1}} (t_{j+1} - u)^{\alpha-1}f(u)du \approx \frac{h^{\alpha}}{\alpha(\alpha+1)}\sum_{i=0}^{j+1} a_{i,j+1}f(t_i)$$

where,

$$a_{i,j+1} = \begin{cases} (j^{\alpha+1} - (j-\alpha)(j+1)^{\alpha}) & \text{if } i = 0, \\ \left((j-i+2)^{\alpha+1} + (j-i)^{\alpha+1} - 2(j-i+1)^{\alpha+1}\right) & \text{if } 1 \le i \le j, \\ 1 & \text{if } i = j+1. \end{cases} \quad (4.7)$$

Thus the corrector y_{j+1} is determined by

$$y_{j+1} = \sum_{k=0}^{m-1} \frac{t_{j+1}^k}{k!} x_0^{(k)} + \frac{1}{\Gamma(\alpha)} \left(\sum_{i=0}^{j} a_{i,j+1} f(t_i, y_i) + a_{j+1,j+1} f(t_{i+1}, y_{i+1}^P) \right). \quad (4.8)$$

with weights (4.7).

An approximate solution to equation (1.1) or (4.3) can be obtained using the predictor (4.6) and the corrector (4.8) with the relevant weights.

4.2.1. *A note on multiple corrector iterations.* The Predictor Corrector order of convergence depends first upon the Predictor order of convergence and then on the number of corrector iterations, which generally raise the order towards the order of the Corrector. In this case the Predictor is a fractional generalization of the Euler method, and it is widely known that the error of the Euler method behaves (for ordinary differential equations) as $O(h)$. The error of the Adams-Bashford algorithm for ordinary differential equations is well known to be $O(h^2)$. Indeed, it is known that, in the ordinary differential equation case, if the corrector is iterated repeatedly to convergence then the scheme has order $O(h^2)$.

For the Fractional Differential Equation, when $\alpha < 1$ and no additional corrector iterations are applied Diethelm and Ford [12] proved that the error in the fractional Predictor Corrector algorithm behaves as

$$\max_{j=0,1,\ldots,n} |y(t_j) - y_j| = O(h^p)$$

where $p = \min(2, 1+\alpha)$.

4.3. **Approximate Mittag-Leffler, K. Diethelm and Y. Luchko** [14]**.** Diethelm and Luchko use the observation that a linear fractional differential equation has an exact solution, which can be expressed in terms of Mittag-Leffler type functions. They then use convolution quadrature [8] and discretized operational calculus [18] to produce an approximation to this expression.

The numerical scheme for the solution of equations of the form (1.1) is given by,

$$y_j = \tilde{y}_j + \sum_{i=1}^{N} \hat{y}_j. \quad (4.9)$$

This is achieved by considering the solution as being made up of two parts: the homogeneous problem (i.e. $g = 0$) with non-zero initial conditions, and the inhomogeneous problem (i.e. $g \neq 0$) with zero initial conditions.

For the homogeneous equation we define

$$\tilde{y}_j := \sum_{k=0}^{m-1} x_0^{(k)} \left(\frac{(t_j)^k}{k!} + u_h(k; t_j) \right), j = 1, \ldots, n, \quad (4.10)$$

where

$$u_h(k; t_j) := \omega_j(k; h)/h.$$

Here $\omega_i(k;h), k = 0, 1, \ldots, m-1, i = 0, 1, \ldots, n$ are the coefficients of the power series

$$F_k(\delta(\zeta)/h) = \sum_{i=0}^{\infty} \omega_i(k;h)\zeta^i, \tag{4.11}$$

and $\delta(\zeta) = \sum_{i=0}^{\infty} \delta_i \zeta^i$ is the quotient of the generating polynomials of a linear multistep method,

$$F_k(s) := \frac{\lambda s^{\alpha-k-1}}{s^\alpha - \lambda s^\alpha}$$

and the natural numbers $l_k, k = 0, \ldots, m-1$ are determined from the condition

$$\begin{cases} m_{l_k} \geq k+1 \\ m_{l_k+1} \leq k \end{cases}.$$

For the inhomogeneous equation Luchko and Gorenflo [19] showed that the analytical solution $\tilde{y}$ of equation (1.1) could be represented as

$$\tilde{y} = \int_0^t E(\mu;t)g(t-\tau)d\tau. \tag{4.12}$$

Then using the numerical scheme described in Lubich [18], the convolution integral (4.12) at the point t_j, is approximated by,

$$\tilde{y}_j = \sum_{0 \leq i \leq j} \omega_j(h)g(t_{j-i}), j = 1, \ldots n. \tag{4.13}$$

Where $\omega_i(h)$ is given by

$$F(\alpha;\delta(\zeta)/h) = \sum_{i=0}^{\infty} \omega_i(h)\zeta^i.$$

A simple modification obtains the required convergence order, thus the inhomogeneous case is given by,

$$\tilde{y}_j = \hat{y}_j + \sum_{i=i_0}^{q-1} w_{ji}(h)g(t_i) \quad , q-1 < p-\gamma \leq q \in \mathbb{N}. \tag{4.14}$$

Here the quadrature weights and starting weights need to be calculated in accordance with the methods described in [9], [18]. This means that the weights $w_{ji}(h)$, $i = i_0, \ldots, q-1$ are determined from the ill-conditioned Vandermonde system discussed in [9] which restricts the order of method that can be applied successfully.

In principle one can use a convolution quadrature of arbitrarily high order by the methods in [18]. In our practical experiments we have restricted ourselves to orders 1 and 2. $p = 1$, represents a method based upon Euler's rule. $p = 2$ represents a method based upon a trapezoidal rule. The recent paper [9] indicates that a third order method might also be competitive.

In the following example we can see how the experimental orders of convergence of the method match the theoretical orders.

We let the exact analytical solution, for a fixed interval T be $y(T)$ and the approximation at T using n step lengths be $y_n(T)$. Here, and subsequently, EOC, represents the experimental order of convergence evaluated using the formula

$$EOC = \log_2 \left(\frac{\mid y(T) - y_n(T) \mid}{\mid y(T) - y_{2n}(T) \mid} \right). \tag{4.15}$$

For our experiment, we solved

$$D^{0.5}y + y = t^2 + \frac{2t^{1.5}}{\Gamma(2.5)}, \tag{4.16}$$

with,

$$y(0) = 0.$$

All time values are given in seconds. KFlops = 1000× Flops.

	p=1				p=2			
h	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	3.43e-02	5.7	0.00		1.90e-03	5.9	0.00	
1/16	1.73e-02	21.4	0.00	0.99	5.00e-04	22.3	0.00	1.93
1/32	8.66e-03	89.7	0.05	1.00	1.23e-04	92.9	0.05	1.96
1/64	4.34e-03	395.8	0.11	1.00	3.29e-05	404.8	0.11	1.97
1/128	2.17e-03	1758.4	0.22	1.00	8.40e-06	1797.9	0.22	1.98

TABLE 1. How the order of the convolution quadrature method effects convergence

It can be seen from Table 1 that the second order method involves no extra work and produces much smaller errors. In this paper we shall compare the order 2 method with the competitors.

4.4. **Collocation, L. Blank** [1]**.** Blank used the Riemann-Liouville definition of the fractional derivative D_*. However, as we remarked in Section 2, by subtracting the leading terms of the Taylor series expansion of the solution about zero from the function y, the Riemann-Liouville definition becomes equivalent to the Caputo definition.

Assuming an equidistant mesh $t_j = jh$, and a r-times globally continuously differentiable polynomial spline solution y, a collocation method with polynomial splines can be implemented. For the reader unfamiliar with collocation techniques, a spline on each subinterval $[t_j, t_{j+1}]$ is described by,

$$y(t_j + vh) = \sum_{l=0}^{s+r} a_l^{(j)} v^l, \qquad v \in [0,1],$$

then by choosing s collocation parameters, $0 < c_1 < \ldots < c_s < 1, y$ is constructed to satisfy (1.1) at the collocation points $t_{j,i} = t_j + c_i h$ for $i = 1, \ldots, s$. In other words

$$D^\alpha \left(y(t_j, i) - \sum_{k=0}^{m-1} \frac{1}{k!} t_{j,i}^k y^{(k)}(0) \right) = \lambda y(t_{j,i}) + g(t_{j,i}) \tag{4.17}$$

It was shown by Blank that if $(m-1)$-times continuously differentiable splines (i.e. $r = m-1$) were chosen, the above approximation can be expressed as,

$$\left[D^\alpha \left(y(t) - \sum_{k=0}^{m-1} \frac{1}{k!} t^k y^{(k)}(0) \right) |_{t=t_{j,i}} \right]_{i=1,\ldots,s} = \frac{h^{-\alpha}}{\Gamma(1-\alpha)} \sum_{i=0}^{j} V^{(j-i)} b^{(i)} \tag{4.18}$$

with suitable weight matrices $V^{(i)}$.

The numerical solution y of (4.17), given by the collocation approximation, is determined by the following systems of equations. We use the notation of [1] to which we refer for further details.

On the first subinterval $[0, t_1]$,

$$a^{(0)} = \begin{pmatrix} \ddots & & 0 \\ & h^k \frac{1}{k!} & \\ 0 & & \ddots \end{pmatrix}_{k=0,\ldots,m-1} \left(y^{(k)}(0)\right)_{k=0,\ldots,m-1}$$

$$b^{(0)} = \mathrm{INV}\left\{\lambda C a^{(0)} + g^{(0)}\right\} \tag{4.19}$$

while on $[t_j, t_{j+1}]$ for $j \geq 1$,

$$a^{(j)} = \mathrm{Diff}\begin{pmatrix} a^{(j-1)} \\ b^{(j-1)} \end{pmatrix}$$

$$b^{(j)} = \mathrm{INV}\left(\lambda^{1/\alpha} C a^{(j)} + g^{(j)} - \frac{h^{-\alpha}}{\Gamma(1-\alpha)} \sum_{i=0}^{j-1} V^{(j-i)} b^{(i)}\right)$$

with the matrices $K, V^{(j)}, V, P$, defined by,

$$K^{(j)} = \left((j + c_i)^{m-1+l-\alpha}\right)_{\substack{i=1,\ldots,s \\ l=1,\ldots,s}}$$

$$P_{k,l} = \begin{cases} \frac{(l+m-1)!}{(l-k)!} \Pi_{p=1}^{m-1+k} \frac{1}{p-\alpha} & \text{for } k \leq l \\ 0 & \text{for } k > l \end{cases}$$

$$P = (P_{k,l})$$

$$\begin{aligned} V^{(0)} &= K^{(0)} P_{\mathrm{diag}} \\ V^{(j)} &= K^{(j)} P_{\mathrm{diag}} - K^{(j-1)} P, \qquad (j \geq 1) \end{aligned}$$

Also,

$$\begin{aligned} g^{(j)} &= (g(t_j + c_i h))_{i=1,\ldots,s}, \\ \mathrm{Diff} &= (\mathrm{Diff}_{k,l})_{\substack{k=1,\ldots,s \\ l=1,\ldots,s}} \end{aligned} \tag{4.20}$$

where

$$\mathrm{Diff}_{k,l} = \begin{cases} \frac{l!}{k!(l-k)!} & \text{for } k \leq l \\ 0 & \text{for } k > l \end{cases} \tag{4.21}$$

and

$$\mathrm{INV} = \left[\frac{h^{-\alpha}}{\Gamma(1-\alpha)} K^{(0)} P_{\mathrm{diag}} - \lambda (c_i^{m-1+l})_{\substack{i=1,\ldots,s \\ l=1,\ldots,s}}\right]^{-1} \tag{4.22}$$

In our experiments, reported here, we solved

$$D^{0.5} y + y = t^2 + \frac{2t^{1.5}}{\Gamma(2.5)}, \tag{4.23}$$

with,

$$y(0) = 0.$$

4.4.1. *Choice of collocation points for Blank's method.* In order to implement the numerical scheme one needs to select the collocation points. In the work by Blank the aim was to reflect the qualitative behaviour of the solutions rather than to attain a high accuracy and therefore this was not a consideration there. In fact, there has been no formal consideration of the order of convergence of the method. In our experiments, we considered various choices of collocation points and found that all gave orders of convergence close to 1 as $h \to 0$. However, as can be seen in the table below, for finite values of $h > 0$ there may be quite a difference in the apparent order of convergence and for some choices, the apparent order approaches 1 only for extremely small step lengths. We used the results from this table to assist in our choice of collocation points. The recent book by Brunner ([3]) gives, on pages 361 ff., details (including convergence theory) for collocation methods for weakly singular Volterra integral equations. In that book (see also [24]) there is discussion of how graded meshes can help make the convergence order optimal. The conclusions of Theorem 6.2.9 of [3] indicate that the convergence order of collocation methods for the present example will be bounded below by 0.5 and will vary according to the mesh grading. Our numerical experiments are consistent with this result.

	Collocation points 0.1, 0.5, 1.0				Collocation points 0.4, 0.5, 1.0			
h	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.06e-01	5.2	0.06		6.14e-02	5.2	0.05	
1/16	5.24e-02	11.2	0.11	1.02	3.38e-02	11.2	0.11	0.86
1/32	2.56e-02	27.8	0.16	1.04	1.81e-02	27.8	0.17	0.90
1/64	1.25e-02	79.5	0.33	1.03	9.54e-03	79.5	0.39	0.93
1/128	6.13e-03	256.6	0.93	1.03	4.95e-03	256.6	1.10	0.95
	Collocation points 0.1, 0.4, 0.5, 1.0				Collocation points 0.1, 0.4, 0.5, 0.6, 1.0			
1/8	3.31e-04	8.6	0.05		1.25e-01	13.4	0.06	
1/16	3.54e-04	18.4	0.11	-0.10	6.07e-02	28.3	0.11	1.04
1/32	2.65e-04	45.5	0.22	0.42	2.94e-02	69.4	0.22	1.05
1/64	1.83e-04	129.7	0.50	0.54	1.42e-02	196.0	0.60	1.04
1/128	1.16e-04	417.9	1.32	0.66	6.94e-03	627.5	1.32	1.04

TABLE 2. Effect of collocation point change

4.5. **Finite Differences, R. Gorenflo** [15]**.** The finite difference approach to fractional integration can be attributed to Grünwald and Letnikov. Podlubny [22] used the first order finite difference method and confirmed that this leads to a solver of order 1. Gorenflo [15] introduced a second order difference method (accuracy $O(h^2)$). He described sufficiently smooth zero extension conditions to achieve the desired accuracy.

The approach is a generalization of the well-known backward difference operator for ODEs. By substituting the fractional derivative α for the integer order derivative, the approximation of the α-th derivative is

$$h^{-\alpha} \bigtriangledown_h^{\alpha} y(t) = \tilde{D}^{\alpha} y(t) = h^{-\alpha} \sum_{i=0}^{[t/h]} (-1)^i \begin{pmatrix} \alpha \\ i \end{pmatrix} y(t - ih). \tag{4.24}$$

Thus, for example, the first order approximation to the single term problem (1.1) can be represented by the difference scheme,

$$y_j = -\lambda h^\alpha y_{j-1} - \sum_{i=1}^{j} w_i^{(\alpha)} y(t_{j-i}) + h^\alpha g(t_j) \tag{4.25}$$

where,

$$w_i^{(\alpha)} = (-1)^i \left(\begin{array}{c} \alpha \\ i \end{array} \right).$$

5. **Linear Test Equations.** To illustrate the convergence behaviour of the various single-term methods we vary the value of α of the test equation,

$$D^\alpha y + y = t^2 + \frac{2t^{2-\alpha}}{\Gamma(3-\alpha)} \tag{5.1}$$

$y(0) = 0$, which has the exact solution $y = t^2$. Different values of α are chosen to highlight that the convergence order can depend upon the value of α. In [7] and [11] Diethelm and Ford proved that the convergence of the Implicit Quadrature and Predictor Corrector methods are of the order $O(h^{2-\alpha})$ and $O(h^{1+\alpha})$ respectively. As usual we use M to denote the number of corrector iterations. The Approximate Mittag-Leffler method [14] (with $p = 2$) is of order $O(h^2)$, while the Collocation [1] and Finite Differences [15] methods we have implemented are both of order $O(h)$.

From a user's perspective, what is of more interest is how much time is taken to achieve a solution of a particular accuracy. Tables 3-8 give the information to assess this. For example, in Table 3, the Approximate Mittag-Leffler method with $h = 1/128$, has an absolute error of 4.71e-006, compared to the absolute error of 2.73e-05 for the Implicit Quadrature method, however the number of KFlops required (1796.4 compared to 42.1) is much greater for the Approximate Mittag-Leffler method. It turns out that neither of these methods is the most economical because, if we look at Table 4, we can see that the Predictor Corrector method with $(M = 8)$ is much more efficient.

Remark 5.1. *One could consider also nonlinear fractional differential equations and make a similar comparison of the efficiency of the methods. Here it would be necessary (except for the predictor-corrector method) to use some form of nonlinear solver (such as a Newton iteration) at each step to solve the nonlinear equations. We have considered this matter elsewhere (see* [6]*) and we concluded that the extra work involved in using the predictor-corrector method for a nonlinear problem compared to its use for a linear problem was minimal. As we shall see in our later comparison for linear equations, the extra overhead of a Newton solver for solving nonlinear equations means that the alternative methods we have considered are certainly not competitive with the predictor-corrector scheme.*

6. **Comparison of efficiency of Single-term Fractional Differential Equation solvers.** We plot, for each method, the logarithm or the absolute error at time T against the logarithm of the FLOP count. The 'ideal' numerical method's path will lie close to the bottom left of the graph. This represents a small error for a small computational cost. Points where the paths of the methods cross represent critical values where one method becomes more numerically efficient than the other. We have shown experimentally that the paths of the various methods can be extrapolated reasonably successfully to explore whether, for smaller step

h	Predictor Corrector (M=1)				Implicit Quadrature			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	9.84e-02	2.5	0.05		2.88e-03	1.2	0.00	
1/16	3.50e-02	5.2	0.11	1.49	9.15e-04	2.6	0.00	1.65
1/32	1.27e-02	11.5	0.16	1.46	2.87e-04	5.9	0.05	1.68
1/64	4.72e-03	27.0	0.28	1.43	8.88e-05	14.9	0.11	1.69
1/128	1.79e-03	70.4	0.55	1.40	2.73e-05	42.1	0.22	1.70
	Approximate Mittag-Leffler				Collocation			
1/8	1.19e-03	5.9	0.06		1.25e-01	5.2	0.05	
1/16	2.99e-04	22.3	0.06	1.99	6.24e-02	11.2	0.11	1.00
1/32	7.50e-05	92.9	0.06	2.00	3.08e-02	27.8	0.17	1.02
1/64	1.88e-05	405.9	0.11	2.00	1.51e-02	79.5	0.33	1.03
1/128	4.71e-06	1796.4	0.27	2.00	7.40e-03	256.6	0.93	1.03
	Finite Differences							
1/8	1.63e-02	0.9	0.00					
1/16	8.19e-03	2.3	0.05	0.993				
1/32	4.11e-03	6.9	0.06	0.996				
1/64	2.06e-03	23.1	0.11	0.998				
1/128	1.03e-03	83.0	0.44	0.999				

TABLE 3. Test equation $D^{1/4}y + y = t^2 + \frac{2t^{1.75}}{\Gamma(2.75)}$

h	Predictor Corrector (M=2)				Predictor Corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	3.34e-02	1.9	0.06		9.39e-03	2.9	0.05	
1/16	1.14e-02	4.0	0.06	1.56	2.16e-03	6.4	0.11	2.12
1/32	3.81e-03	9.6	0.11	1.58	4.93e-04	15.4	0.17	2.13
1/64	1.27e-03	25.2	0.22	1.58	1.13e-04	40.9	0.39	2.13
1/128	4.25e-04	74.9	0.44	1.58	2.59e-05	122.8	0.77	2.12
	Predictor Corrector (M=8)				Predictor Corrector (M=16)			
1/8	1.06e-03	5.1	0.06		1.06e-03	9.3	0.16	
1/16	2.68e-04	11.2	0.16	1.98	2.68e-04	20.8	0.33	1.98
1/32	6.78e-05	27.0	0.33	1.98	6.78e-05	50.3	0.66	1.99
1/64	1.71e-05	72.4	0.66	1.99	1.71e-05	135.4	1.26	1.99
1/128	4.29e-06	218.5	1.37	1.99	4.29e-06	410.0	2.53	1.99

TABLE 4. Test equation $D^{1/4}y + y = t^2 + \frac{2t^{1.75}}{\Gamma(2.75)}$

sizes (or equivalently, long run times) any paths cross for step lengths outside the ranges we tried. The gradient of the path represents (in some sense) the *order of numerical efficiency*. Thus two paths which are parallel will possess the same *order of numerical efficiency* but different levels of absolute error.

The graphical technique was used to compare methods for the test equation

$$D^{\alpha}y + y = t^2 + \frac{2t^{2-\alpha}}{\Gamma(3-\alpha)}, \quad y(0) = 0,$$

h	Predictor Corrector (M=1)				Implicit Quadrature			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	2.97e-02	1.3	0.00		1.07e-02	1.2	0.00	
1/16	9.18e-03	2.8	0.05	1.69	3.90e-03	2.6	0.05	1.45
1/32	2.94e-03	6.6	0.05	1.64	1.41e-03	5.9	0.06	1.47
1/64	9.71e-04	17.3	0.16	1.60	5.07e-04	14.9	0.11	1.48
1/128	3.27e-04	51.0	0.33	1.57	1.81e-04	42.1	0.22	1.49
	Approximate Mittag-Leffler				Collocation			
1/8	1.90e-03	5.9	0.06		1.06e-01	5.2	0.06	
1/16	5.00e-04	22.3	0.06	1.99	5.24e-02	11.2	0.11	1.02
1/32	1.29e-04	92.9	0.06	2.00	2.56e-02	27.8	0.16	1.04
1/64	3.29e-05	404.7	0.11	2.00	1.25e-02	79.5	0.33	1.03
1/128	8.36e-06	1797.7	0.27	2.00	6.13e-03	256.6	0.93	1.03
	Finite Differences							
1/8	3.43e-02	0.9	0.00					
1/16	1.73e-02	2.3	0.05	0.991				
1/32	8.66e-03	6.9	0.06	0.996				
1/64	4.34e-03	23.1	0.11	0.998				
1/128	2.17e-03	83.0	0.44	0.999				

TABLE 5. Test equation $D^{1/2}y + y = t^2 + \frac{2t^{1.5}}{\Gamma(2.5)}$

h	Predictor Corrector (M=2)				Predictor Corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	5.99e-03	1.9	0.05		5.53e-04	2.9	0.06	
1/16	1.34e-03	4.0	0.06	2.16	2.11e-04	6.4	0.05	1.39
1/32	3.08e-04	9.6	0.11	2.13	6.12e-05	15.4	0.17	1.79
1/64	7.18e-05	25.2	0.22	2.10	1.63e-05	40.9	0.38	1.91
1/128	1.70e-05	74.9	0.38	2.08	4.20e-06	122.8	0.77	1.96
	Predictor Corrector (M=8)				Predictor Corrector (M=16)			
1/8	1.06e-03	5.1	0.05		1.06e-03	9.3	0.17	
1/16	2.68e-04	11.2	0.22	1.98	2.68e-04	20.8	0.28	1.98
1/32	6.78e-05	27.0	0.33	1.98	6.78e-05	50.3	0.60	1.99
1/64	1.71e-05	72.4	0.66	1.99	1.71e-05	135.4	1.27	1.99
1/128	4.29e-06	218.5	1.37	1.99	4.29e-06	410.0	2.58	1.99

TABLE 6. Test equation $D^{1/2}y + y = t^2 + \frac{2t^{1.5}}{\Gamma(2.5)}$

exact solution $y = t^2$, for various values of α.

On a first reading, Figure 1 seems to show that the Implicit Quadrature method performs most economically for $\alpha = 1/4$. Based on further experiments, this would also appear to be the case for all $0 < \alpha < 1/2$. After closer examination however, one can see that the slopes of the Implicit Quadrature and the Predictor Corrector ($M = 8$) paths could possibly cross for smaller h. Extrapolating the paths as in Figure 2 shows that the paths cross when $\log_2(\text{error}) = -18$, which corresponds to an error of 2.5×10^{-6}. This size of error could be achieved using a step size of

h	Predictor Corrector (M=1)				Implicit Quadrature			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	1.23e-02	1.3	0.00		3.03e-02	0.9	0.00	
1/16	3.34e-03	2.8	0.06	1.88	1.30e-02	1.9	0.00	1.22
1/32	9.63e-04	6.6	0.05	1.79	5.55e-03	4.5	0.05	1.23
1/64	2.77e-04	17.3	0.17	1.80	2.35e-03	11.9	0.11	1.24
1/128	8.07e-05	51.0	0.27	1.80	9.92e-04	36.1	0.22	1.24
	Approximate Mittag-Leffler				Collocation			
1/8	1.27e-03	6.0	0.00		8.94e-02	5.1	0.05	
1/16	4.02e-04	22.2	0.06	1.66	4.55e-02	11.1	0.11	0.97
1/32	1.16e-04	93.3	0.05	1.79	2.28e-02	27.6	0.22	1.00
1/64	3.20e-005	408.5	0.17	1.86	1.13e-02	79.1	0.44	1.01
1/128	8.62e-006	1799.0	0.28	1.89	5.65e-03	255.8	1.15	1.00
	Finite Differences							
1/8	5.43e-02	0.7	0.00					
1/16	2.74e-02	1.8	0.00	0.99				
1/32	1.38e-02	5.8	0.06	0.99				
1/64	6.91e-03	14.7	0.11	1.00				
1/128	3.45e-03	49.9	0.38	1.00				

TABLE 7. Test equation $D^{3/4}y + y = t^2 + \frac{2t^{1.25}}{\Gamma(2.25)}$

h	Predictor Corrector (M=2)				Predictor Corrector (M=4)			
	error y(1)	KFlops	Time	EOC	error y(1)	KFlops	Time	EOC
1/8	6.33e-04	1.9	0.00		8.21e-04	2.9	0.05	
1/16	2.81e-05	4.0	0.05	4.49	2.16e-04	6.4	0.11	1.93
1/32	1.41e-05	9.6	0.17	0.99	5.58e-05	15.4	0.16	1.95
1/64	7.06e-06	25.2	0.22	1.00	1.43e-05	40.9	0.38	1.96
1/128	2.38e-06	74.9	0.44	1.75	3.64e-06	122.8	0.77	1.97
	Predictor Corrector (M=8)				Predictor Corrector (M=16)			
1/8	8.46e-04	5.1	0.05		8.46e-04	9.3	0.11	
1/16	2.18e-04	11.2	0.17	1.96	2.18e-04	20.8	0.27	1.96
1/32	5.59e-05	27.0	0.33	1.96	5.59e-05	50.3	0.66	1.96
1/64	1.43e-05	72.4	0.72	1.97	1.43e-05	135.4	1.26	1.97
1/128	3.64e-06	218.5	1.37	1.97	3.64e-06	410.0	2.53	1.97

TABLE 8. Test equation $D^{3/4}y + y = t^2 + \frac{2t^{1.25}}{\Gamma(2.25)}$

$h = 1/165$. In Figure 3 the step size for the Implicit Quadrature method has been reduced to $h = 1/1024$ and the Predictor Corrector $M = 8$ method to $h = 1/256$. The figure shows that the paths these methods do intersect at the predicted value of $\log_2(\text{error}) = -18$.

7. **Dependence on the Smoothness of the Solution.** Several of the earlier papers rely on the smoothness of the solution to prove results on the rates of convergence. This is the classical approach from ordinary differential equations. However, for fractional equations even polynomial solutions may become non-smooth following fractional order differentiation. Therefore we explore briefly whether the form of the solution affects the performance of the method. Table 9 shows how the exact solution y determines which method is most economical for a fixed value of α. This indicates that the differentiability of the function directly affects the relative performance of the methods. This is supported by the experimental data in Tables 9 - 11. Based on our experiments it seems that the higher the powers of t appearing in the solution, the more efficient, relatively, the Predictor-Corrector method becomes. Since this method has always been either optimal or quite near to the optimal method, we would propose that it should be the method of choice in most applications.

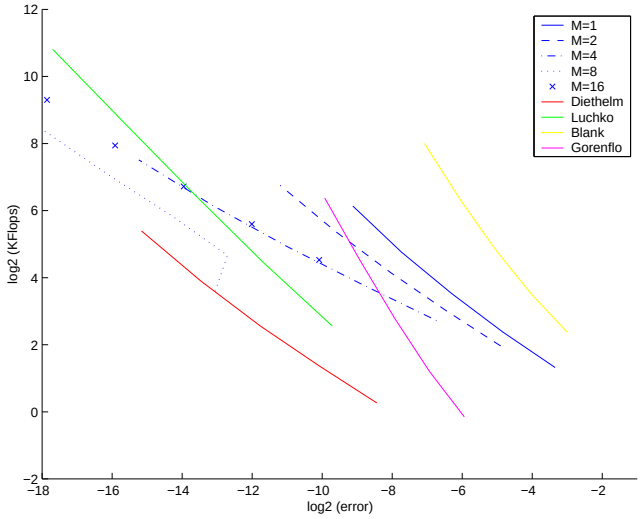

FIGURE 1. $\alpha = 1/4$

Figures 4 and 5 give similar graphs for other values of α.

	Shows the number of steps (n_D) with $m = 8$ corrections at which the Adams method outperforms the Diethelm method, for Single term equations with the following exact solutions.					
value of α	$y = t^2$	$y = t^3$	$y = t^4$	$y = t^5$	$y = t^7$	$y = t^{10}$
$\alpha = 0.25$	$n_D = 165$	$n_D = 260$	$n_D = 360$	$n_D = 425$	$n_D = 555$	$n_D = 750$

TABLE 9. Power of t

This still leaves us to choose the number of corrector iterations: Table 10 shows how the optimum number of corrector iterations varies with α in our experiments. In the Table, the value m represents the optimum number of corrector iterations

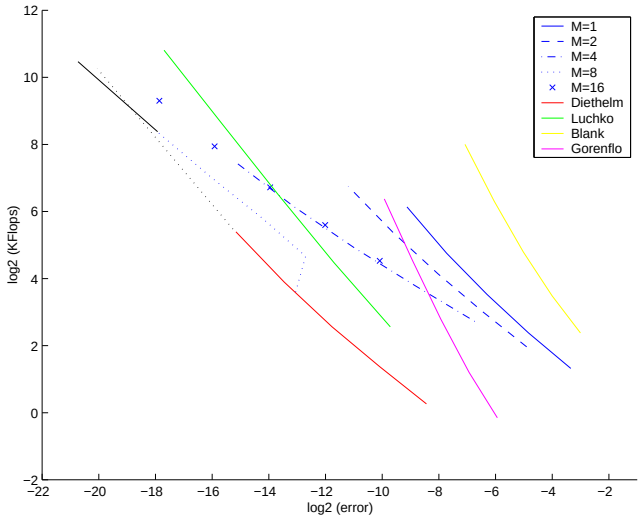

FIGURE 2. $\alpha = 1/4$ extrapolated

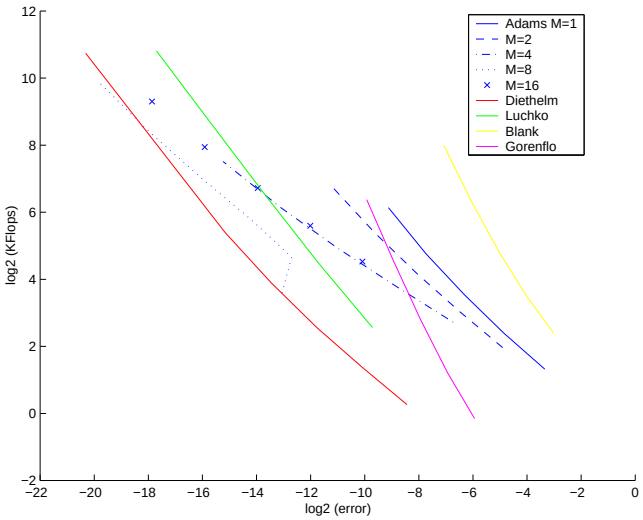

FIGURE 3. $\alpha = 1/4$ extended

and the value (n_D) is the number of steps at which there is a change in the optimum number of corrector iterations.

Table 11 shows how varying the coefficient of t^2 has little effect on the optimum number of steps n_D.

8. **Conclusions and Further Work.** Based on our experiments, the Predictor-Corrector method [11] is recommended in most situations, however the Implicit Quadrature method [7] performs well for linear problems with $\alpha < 1/2$. As we pointed out previously, the Predictor-Corrector method has the advantage of easy implementation for non-linear problems too and also is *nearly optimal* even when

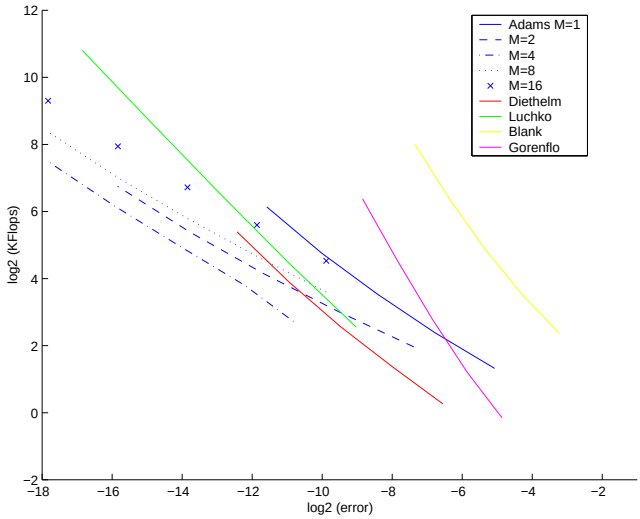

FIGURE 4. $\alpha = 1/2$

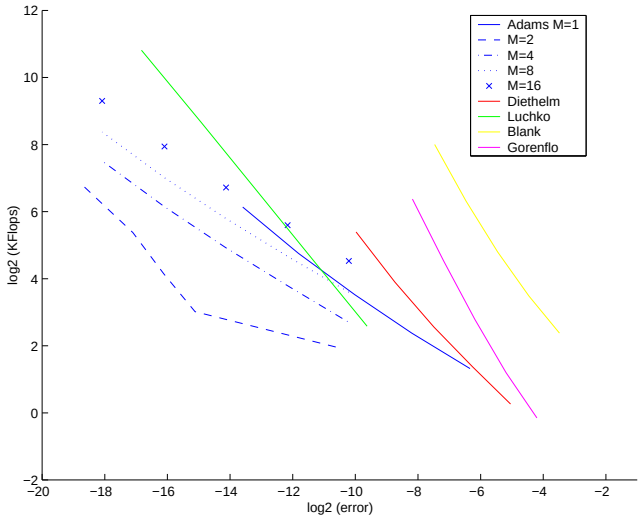

FIGURE 5. $\alpha = 3/4$

the Implicit Quadrature is better. Therefore in our opinion it should be the method of choice for those who want to use a single method.

As we remarked earlier, the recent paper [9] indicates that a convolution quadrature method based on a third order backward difference scheme might be competitive and this would be worthy of experimentation in the future. Both time and space constraints have prevented its consideration here.

	Single term equation which has the exact solution.		
value of α	Exact $y = t^2$	Exact $y = t^3$	
$\alpha > 1$	$m = 1$	$m = 1$	
$1 > \alpha > 0.6$	$m = 2$	$m = 2$	
$\alpha = 0.575$	$m = 4 \rightarrow 2$ at $n_D = 20$	$m = 2$	
$\alpha = 0.55$	$m = 4 \rightarrow 2$ at $n_D = 128$	$n_D = 6$ $m = 2$	
$\alpha = 0.525$	$m = 4$	$n_D = 9$ $m = 2$	
$\alpha = 0.5$	$m = 4$	$n_D = 13$ $m = 2$	
$\alpha = 0.45$	$m = 4$	$n_D = 10$ $m = 4$	
$\alpha = 0.4$	$m = 4$	$m = 4$	
$\alpha = 0.35$	$n_D = 18$ $m = 4$	$n_D = 10$ $m = 4$	
$\alpha = 0.3$	$n_D = 75$ $m = 4$	$n_D = 34$ $m = 4$	
	$n_D = 90$ $m = 8$	$n_D = 135$ $m = 8$	
$\alpha = 0.25$	$n_D = 165$ $m = 8$	$n_D = 260$ $m = 8$	
$\alpha = 0.2$	$n_D = 260$ $m = 8$	$n_D = 700$ $m = 8$	
$\alpha = 0.15$	$n_D = 1500$ $m = 8$	$n_D \simeq 400$ $m = 8$	

TABLE 10. Varying α

	shows the optimum number of steps (n_D) and corrections (m) at which the Adams method outperforms the Diethelm method.				
value of α	Exact $y = 2t^2$	Exact $y = 10t^2$	Exact $y = \frac{t^2}{2}$		
$\alpha = 0.25$	$n_D = 172$ $m = 8$	$n_D = 172$ $m = 8$	$n_D = 170$ $m = 8$		

TABLE 11. Solution is a multiple of t^2

REFERENCES

[1] L. Blank. Numerical treatment of differential equations of fractional order. *Nonlinear World*, 4:473–490, 1997.

[2] L. Boyadjiev, S. L. Kalla, and H. G. Khajah. Analytical and numerical treatment of a fractional integro-differential equation of Volterra type. *Math. Comput. Modelling*, 25:1–9, 1997.

[3] H. Brunner. *Collocation methods for Volterra integral and related functional equations*. Cambridge University Press, 2004.

[4] M. Caputo. Linear models of dissipation whose (q) is almost frequency independent-2. *Geophys. J. R. Astr. Soc.*, 13:529–539, 1967.

[5] J. T. Chern. *Finite element modelling of viscoelastic materials on the theory of fractional calculus*. PhD thesis, Pennsylvania State University, 1993.

[6] J. Connolly. *The numerical solution of fractional and distributed order differential equations*. PhD thesis, University of Liverpool, 2005.

[7] K. Diethelm. An algorithm for the numerical solution of differential equations of fractional order. *Electronic Transactions on Numerical Analysis*, 5:1–6, 1997.

[8] K. Diethelm. Generalized compound quadrature formula for finite-part integrals. *Journal of Numerical Analysis*, 17:479–493, 1997.

[9] K. Diethelm, J. M. Ford, N. J. Ford, and M. Weilbeer. Pitfalls in fast numerical solvers for fractional differential equations. *J. Comput. Appl. Math.*, 186:482–503, 2005.

[10] K. Diethelm and N. J. Ford. Analysis of fractional differential equations. *Journal of Mathematical Analysis and Applications*, 265:229–248, 2002.

[11] K. Diethelm, N. J. Ford, and A. D. Freed. A predictor-corrector approach for the numerical solution of fractional differential equations. *Nonlinear Dynamics*, 29:3–22, 2002.

[12] K. Diethelm, N. J. Ford, and A. D. Freed. Detailed error analysis for a fractional Adams method. *Numerical Algorithms*, 36:31–52, 2004.

[13] K. Diethelm, N. J. Ford, A. D. Freed, and Y. Luchko. Algorithms for the fractional calculus: A selection of numerical methods. *Computer Methods in Applied Mechanics and Engineering*, 194:743–773, 2005.

[14] K. Diethelm and Y. Luchko. Numerical solution of linear multi-term differential equations of fractional order. Technical report, TU Braunschweig, 2001.

[15] R. Gorenflo. Fracional calculus: Some numerical methods. *CISM Courses and Lectures, no. 378, Springer, Wien*, pages 277–290, 1997.

[16] M. Enelund K. Adolfsson and S. Larsson. Adaptive discretization of an integro-differential equation with a weakly singular convolution kernel. *Comp. Meth. Appl. Mech. Engrg.*, 192:5285–5304, 2003.

[17] J. Leszczynski and M. Ciesielski. A numerical method for solution of ordinary differential equations of fractional order. In *Parallel processing and applied mathematics*, pages 695–702. Springer Lecture Notes Comput. Sci. 2328, 2002.

[18] C. Lubich. Convolution quadrature and discretized operational calculus 2. *Numerical Mathematics*, 52:413–425, 1988.

[19] Y. Luchko and R. Gorenflo. The initial value problem for some fractional differential equations with the Caputo derivatives. Technical report, University of Berlin, 1997.

[20] S. Momani and S. Hadid. An algorithm for numerical solutions of fractional order differential equations. *J. Fractional Calculus*, 15:61–66, 1999.

[21] I. Podlubny. Numerical solution of ordinary fractional differential equations by the fractional difference method. In *Proceedings of the Second International Conference on Difference Equations*. Gordon and Breach Science Publishers, 1995.

[22] I. Podlubny. *Fractional Differential Equations*. Academic Press, San Diego, 1999.

[23] I. Podlubny. Geometric and physical interpretation of fractional integration and fractional differentiation. *Fractional Calculus and Applied Analysis*, 5:367–386, 2002.

[24] T. Tang. Superconvergence of numerical solutions to weakly singular volterra integro-differential equations. *Numer. Math.*, 61:373–382, 1992.

[25] M. Weilbeer. *Efficient Numerical Methods for Fractional Differential Equations and their Analytical Background*. PhD thesis, T-U Braunschweig, 2005.

[26] Z. Zhu, G. Li, and C. Cheng. A numerical method for fractional integral with applications. *Appl. Math. Mech. Eng. Ed.*, 24:373–384, 2003.

Received March 2005; revised June 2005.

E-mail address: njford@chester.ac.uk