Numer Algor (2016) 71:207–228
DOI 10.1007/s11075-015-9998-1

ORIGINAL PAPER

Fast, accurate and robust adaptive finite difference methods for fractional diffusion equations

Santos B. Yuste[1] · **J. Quintana-Murillo**[1]

Received: 2 January 2015 / Accepted: 21 April 2015 / Published online: 2 May 2015
© Springer Science+Business Media New York 2015

Abstract The computation time required by standard finite difference methods with fixed timesteps for solving fractional diffusion equations is usually very large because the number of operations required to find the solution scales as the square of the number of timesteps. Besides, the solutions of these problems usually involve markedly different time scales, which leads to quite inhomogeneous numerical errors. A natural way to address these difficulties is by resorting to adaptive numerical methods where the size of the timesteps is chosen according to the behaviour of the solution. A key feature of these methods is then the efficiency of the adaptive algorithm employed to *dynamically* set the size of every timestep. Here we discuss two adaptive methods based on the step-doubling technique. These methods are, in many cases, immensely faster than the corresponding standard method with fixed timesteps and they allow a tolerance level to be set for the numerical errors that turns out to be a good indicator of the actual errors.

Keywords Finite difference methods · Adaptive methods · Fractional diffusion equations · Anomalous diffusion

Mathematics Subject Classification (2010) 26A33 · 65M06 · 35R11 · 35Q82

✉ Santos B. Yuste
santos@unex.es

[1] Departamento de Física, Universidad de Extremadura, E-06071 Badajoz, Spain

1 Introduction

Interest in fractional calculus is flourishing, to a large extent, due to its usefulness as a mathematical tool for tackling an increasing variety of scientific problems usually associated with complex systems that show some kind of long-term memory. In control engineering, fractional calculus have been successfully employed for many years. In statistical physics, fractional calculus is extremely useful in the study of some systems in which the diffusive processes are anomalous. In particular, it can be rigorously proved that fractional diffusion equations are the appropriate equations to describe the spread of some classes of continuous time random walkers in the same way that normal diffusion equations describe the diffusion of Brownian walkers (see, for example, the review chapter of Metzler and Jeon in Ref. [1]).

Of course, the utility of fractional calculus is linked to our ability to extract useful information about the systems concerned from this formalism. Fortunately, many of the long-time well-established analytical methods employed to study normal diffusion equations can be adapted to fractional diffusion equations. For example, in many cases fractional and normal diffusion equations can be solved similarly in Fourier-Laplace space. However, as is the case too with non-fractional problems, it is also very convenient (or even indispensable) to have at our disposal numerical procedures for studying these equations, and thus get information about the systems they describe. The study of numerical methods for solving fractional equations has been an area of quite active research in recent years (see relatively recent accounts of the literature on this issue in Refs. [2] and [3]). As also for non-fractional differential equations, finite difference methods are one of the most important classes of numerical methods for solving fractional partial differential equations.

Usually, finite difference methods developed for fractional diffusion equations employ uniform time discretization, i.e., fixed timesteps [3]. But methods of this kind have two main drawbacks: they are slow and their accuracy is inconsistent. In fact, they become *increasingly* slower as time goes by: the CPU time required to get the solution at time t grows as the *square* of t (i.e., the arithmetic complexity of these algorithms is of the order Δ^{-2}, with Δ being the size of the timestep). This difficulty has been recognized for a long time, and some procedures have been proposed to alleviate it. The most obvious is to increase the order of accuracy of the numerical method so that larger timesteps can be used without losing the accuracy of the solutions [3]. Another approach is based on the so-called "short memory principle" [4] that, in summary, either tries to cap the number of required operations per step assuming that the influence of the previous values of the solution for times far from the present time can be neglected [5], or takes advantage of the way in which the kernel of the fractional derivative decays to get arithmetic complexity of order $\Delta^{-1}\log\Delta^{-1}$ [6]. Another problem of standard methods with fixed timesteps, one that is rarely noticed, is that the accuracy of their numerical results changes strongly (even by orders of magnitude, see the figures in Section 3) over the time interval of integration. The reason for this behaviour can be traced back to the typical behaviour of the solutions of many fractional diffusion equations. In many cases, these solutions can be written as a superposition of generalized Fourier modes that decay as Mittag-Leffler functions. But, as is well known [7], these functions decay very fast

for short times and very slowly for longer times. This behaviour is, in many cases, inherited by the full solution. Indeed, in these cases, to use fixed timesteps to deal with such different time regimes seems a poor choice.

These problems regarding the speed and accuracy of finite difference methods can be alleviated by using methods with adaptive timesteps. This kind of method has the great advantage that the size of the timesteps can be chosen according to the behaviour of the solution. Ideally, a good adaptive method, as Press et al. say for ODE integrators, "should exert some adaptive control over its own progress, making frequent changes in its stepsize ... Many small steps should tiptoe through treacherous terrain, while a few great strides should speed through smooth uninteresting countryside ... [so as to] achieve some predetermined accuracy in the solution with minimum computational effort" [8]. In order to construct this kind of method for fractional diffusion equations, two key ingredients are required: first, a finite difference method that can work with variable timesteps, and second, a procedure for choosing the size of the timesteps. Finite difference methods that can work with variable timesteps are scarce. Some examples are the matrix approach on non-equidistant grids by Podlubny et al. [10], a generalized Crank-Nicolson method by Mustapha et al. [11, 12], a non-uniform L1 time discretization [13–15], and a generalization of the original convolution quadrature method by Lubich [16]. The finite difference method we employ in this paper is an unconditionally stable implicit method discussed in Ref. [13]; the adaptive control procedure is based in the so-called step-doubling technique [8, 9].

The paper is organized as follows. In Section 2 we present an unconditionally stable finite difference scheme that is able to solve fractional diffusion equations by employing variable timesteps. In Section 3 we present two adaptive algorithms for choosing the size of the timesteps and we analyze and compare their speed and accuracy: the trial and error method (considered in Ref. [14]) and the one we call predictive method. In Section 4 we discuss four examples that show some relevant features of our adaptive methods and we provide an alternative comparison of the speeds of these methods. We end with some remarks and conclusions.

2 Algorithm with non-uniform timesteps

The equation we consider is a one-dimensional fractional diffusion equation in the Caputo form

$$\frac{\partial^\gamma u}{\partial t^\gamma} = K \frac{\partial^2 u}{\partial x^2} + f(x,t) \tag{1}$$

where $f(x,t)$ is a source term and

$$\frac{\partial^\gamma}{\partial t^\gamma} y(t) \equiv \frac{1}{\Gamma(1-\gamma)} \int_0^t d\tau \frac{1}{(t-\tau)^\gamma} \frac{dy(\tau)}{d\tau}, \quad 0 < \gamma < 1, \tag{2}$$

is the Caputo fractional derivative [5]. The extension of our procedure to other spatial dimensions and to other equations with terms involving standard non-fractional spatial derivatives (e.g., the fractional Fokker-Planck equation [17]) is straightforward.

In this paper, the Caputo time derivative is discretized by means of a direct generalization of the well-known fractional L1 formula [18] to the case of non-uniform meshes [13]. This non-uniform time discretization is a key part of our approach. Because our purpose is to study adaptive methods that tackle the difficulties associated with the fractional nature of the time-derivative operator by employing non-uniform timesteps, we limit ourselves to the simple discretization of the non-fractional part of the equation (the Laplacian operator) by means of the three-point centred formula. For the case of uniform timesteps the present method becomes the numerical scheme discussed by Liu et al. [19] and Murio [20]. It can be proved that the method is unconditionally stable *regardless* of the size of the (non-uniform) timesteps employed [13, 15]. A key aspect of the present method is the way in which the fractional derivative is discretized on a non-uniform temporal mesh; the discretization of the *non*-fractional spatial operator can be implemented straightforwardly by means of standard procedures of non-fractional finite difference methods [21]. For the sake of completeness, we shall give here the main formulas of this finite difference scheme on non-uniform temporal meshes.

Let (x_j, t_m) be the coordinates of the (j, m) node of the mesh of the space-time region where one wants to obtain the numerical solution of the fractional equation. We will denote by $U_j^{(m)}$ the numerical estimate provided by the difference methods of the exact solution $u(x_j, t_m) = u_j^{(m)}$. Next, we replace the continuous operators of the fractional equation by suitably chosen difference operators:

$$\frac{\partial^\gamma}{\partial t^\gamma} u(x, t_n) = \frac{1}{\Gamma(2-\gamma)} \sum_{m=0}^{n-1} T_{m,n}^{(\gamma)} \left[u(x, t_{m+1}) - u(x, t_m) \right] + R_{t_n}(x) \tag{3}$$

where [13]

$$T_{m,n}^{(\gamma)} = \frac{(1-\gamma) \int_{t_m}^{t_{m+1}} (t_n - \tau)^{-\gamma} \, d\tau}{t_{m+1} - t_m} = \frac{(t_n - t_m)^{1-\gamma} - (t_n - t_{m+1})^{1-\gamma}}{t_{m+1} - t_m} \tag{4}$$

and $m \leq n-1$. The truncation error $R_{t_n}(x)$ is bounded by a quantity that is the sum of one term of order $\Delta_n^{2-\gamma}$ and another of order $\Delta_{\max}^2 \Delta_n^{-\gamma}$ [15] where $\Delta_n = t_n - t_{n-1}$. The Laplacian operator is given by the three-point centred formula:

$$\frac{\partial^2}{\partial x^2} u(x_j, t) = \frac{u(x_{j+1}, t) - 2u(x_j, t) + u(x_{j+1}, t)}{(\Delta x)^2} + R_{x_j}(t). \tag{5}$$

The truncation error $R_{x_j}(t)$ is of order $(\Delta x)^2$. Neglecting the truncation errors and multiplying the equation by Δ_n^γ one gets the difference approximation to the continuous equation [13]:

$$\sum_{m=0}^{n-1} \tilde{T}_{m,n}^{(\gamma)} \left[U_j^{(m+1)} - U_j^{(m)} \right] = S_n [U_{j+1}^{(n)} - 2U_j^{(n)} + U_{j-1}^{(n)}] + F_j^{(n)} \tag{6}$$

where [13, 23]

$$S_n = \Gamma(2-\gamma)\frac{K\,\Delta_n^{\gamma}}{(\Delta x)^2}, \tag{7}$$

$$\tilde{T}_{m,n}^{(\gamma)} = \Delta_n^{\gamma}\; T_{m,n}^{(\gamma)}\,, \tag{8}$$

$$F_j^{(n)} = \Gamma(2-\gamma)\;\Delta_n^{\gamma}\; f(x_j, t_n). \tag{9}$$

Reordering (6) one gets the following (implicit) finite difference scheme:

$$-S_n\,U_{j+1}^{(n)} + (1+2S_n)U_j^{(n)} - S_n\,U_{j-1}^{(n)} = U_j^{(n-1)} - \sum_{m=0}^{n-2} \tilde{T}_{m,n}^{(\gamma)} \left[U_j^{(m+1)} - U_j^{(m)}\right] + F_j^{(n)}, \tag{10}$$

which can be written in matrix-vector form as

$$A\,U^{(n)} = G\left(U^{(n-1)}, U^{(n-2)}, \ldots, U^{(0)}, F^{(n)}, t_n\right), \tag{11}$$

where $U^{(m)}$ stands for the vector $\left\{U_j^{(m)}\right\}$. This equation, $AU = G$, is formally identical to that of the non-fractional differential equation, and its solution $U = A^{-1}G$ can be obtained very efficiently by means of the Thomas algorithm because A is a tridiagonal matrix. The key difference with respect to non-fractional algorithms is that, in order to evaluate G, the numerical solution $U^{(m)}$ for *all* the n previous time values has to be employed [see the right-hand side of (10)], while for non-fractional equations (i.e., for $\gamma = 1$) only the solution at the previous value $U^{(n-1)}$ is required. This explains why finite difference methods are increasingly slow: the computational cost of going from the solution at time t_{n-1} to the solution at time t_n grows as n, i.e., as the number of terms of the sum that defines G, which implies that the computational cost for going from t_0 to t_n grows as n^2.

3 Adaptive methods

In the previous section we have presented a finite difference method that can work with variable timesteps. This is the first key ingredient of our adaptive method. The second ingredient is a procedure for choosing the size of the timesteps according to the behaviour of the solution. In this paper, we shall consider two methods: the trial and error (T&E) step-doubling algorithm [14], and the predictive step-doubling algorithm. Both algorithms are based on the step doubling technique [8]: the numerical solution at a given time t_n is evaluated twice, first employing a full step $\Delta_n = t_n - t_{n-1}$ and, next, independently, using two half steps of size $\Delta_n/2$; the difference $\mathcal{E}^{(n)}$ between the two numerical estimates of the solution, $U_k^{(n)}$ and $\widehat{U}_k^{(n)}$, respectively (see Fig. 1), gauges the truncation error. The control algorithm, by adjusting the size of the timesteps, keeps this difference around a prefixed value τ, the

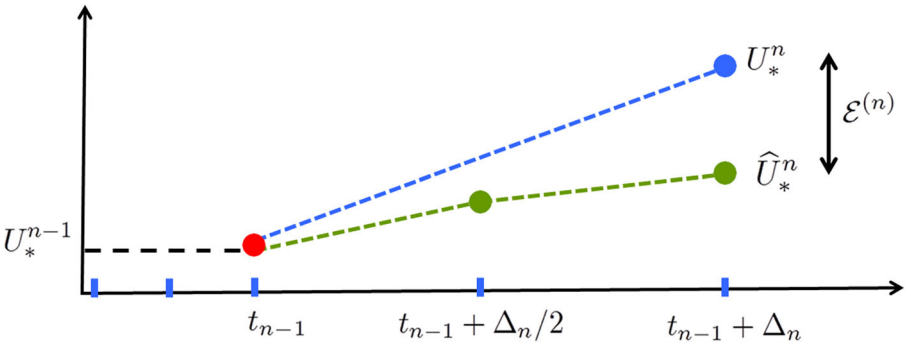

Fig. 1 Scheme of the step-doubling technique. The solution at time t_n is obtained by means of (i) a full timestep of size $t_n - t_{n-1}$ and (ii) by means of two steps of size $(t_n - t_{n-1})/2$. The difference $\mathcal{E}^{(n)}$ between both solutions is used as an indicator of the numerical error

tolerance. Hopefully, this tolerance is an indicator of the size of the actual numerical error (see the Appendix). In this paper we define the difference $\mathcal{E}^{(n)}$ in this way:

$$\mathcal{E}^{(n)} = \max_{\text{all } k} \left| \widehat{U}_k^{(n)} - U_k^{(n)} \right|. \tag{12}$$

Similarly, we define the numerical error at time t_n by

$$error(t_n) \equiv E^{(n)} = \max_{\text{all } x_j} \left| u(x_j, t_n) - U_k^{(n)} \right|. \tag{13}$$

In the next two subsections we describe in detail these two methods and discuss their main characteristics and performance. To do this, we will use the following problem as testbed:

$$\frac{\partial^\gamma u}{\partial t^\gamma} = \frac{\partial^2 u}{\partial x^2}, \quad 0 \le x \le \pi, \tag{14a}$$

$$u(x = 0, t) = u(x = \pi, t) = 0, \tag{14b}$$

$$u(x, 0) = \sin x. \tag{14c}$$

Its exact solution is specially simple

$$u(x, t) = E_\gamma(-t^\gamma) \sin(x) \tag{15}$$

with E_γ being the Mittag-Leffler function [7].

3.1 Trial and error method

In the trial and error (T&E) algorithm the procedure for choosing the size of the timesteps is as follows [14]:

1. If, initially, $\mathcal{E}^{(n)}$ is larger than the tolerance τ, then we halve the timestep Δ_n and check whether the new difference $\mathcal{E}^{(n)}$ corresponding to the new timestep (i.e., to the timestep $\Delta_n/2$) is still larger than the tolerance. We repeat this procedure

until the difference $\mathcal{E}^{(n)}$ is smaller than τ. In this case we get the last timestep as the appropriate value.

2. But if, initially, the difference $\mathcal{E}^{(n)}$ is smaller than the tolerance τ, then we double the size of the timestep. We repeat this procedure until $\mathcal{E}^{(n)}$ is larger than the tolerance. When this happens, we take as appropriate timestep the timestep *previous* to the last one.

As starting value for Δ_n we use the value of the previous timestep Δ_{n-1}. Therefore Δ_0 is undefined and its value has to be given in order to initiate the algorithm. We have always taken $\Delta_0 = 0.01$ because this seeding value is largely irrelevant since the T&E algorithm quickly finds a timestep Δ_1 that leads to a suitable $\mathcal{E}^{(1)}$.

In order to see how good the T&E algorithm is, we checked its speed (that is, its capacity to integrate the equation over large time intervals employing small CPU times) and the size of the errors that the method provides when applied to the testbed problem (14). Unless otherwise explicitly stated, the CPU times are not given in seconds but in units of $\mathcal{T}_{50}$, which is the CPU time employed by our method with fixed timesteps (i.e, without implementing any adaptive choice of the size of the timesteps) to get the solution of the problem (14) for $\gamma < 1$ when 50 timesteps are used (in our computer $\mathcal{T}_{50} \approx 1.4$ seconds). In this way the CPU-time values reported here are expected to be roughly independent of the particular computer system employed. The normalized CPU time required to evaluate the numerical solution of a given problem at time t will be denoted by $T_{\mathrm{CPU}}(t)$.

In Fig. 2, we compare $T_{\mathrm{CPU}}(t)$ for $\gamma = 1/4$ corresponding to the T&E method with tolerance $\tau = 10^{-4}$ with the T_{CPU} values for the standard (non-adaptive) method with fixed timesteps of size $\Delta_n = 0.01$. First we see that, as expected, the CPU time required by the standard method grows quadratically: $T_{\mathrm{CPU}} \propto t^2$. However, for the

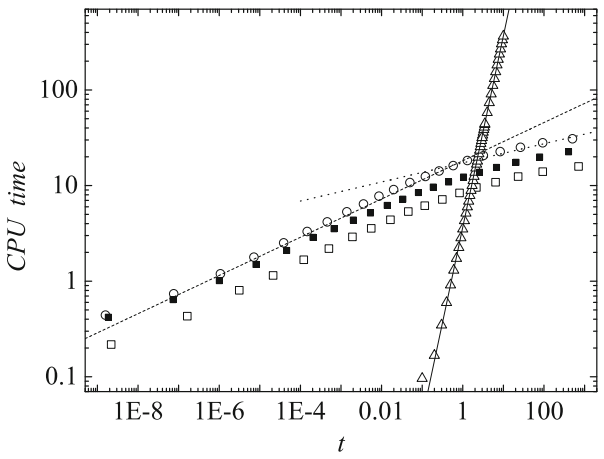

Fig. 2 Normalized computational time $T_{\mathrm{CPU}}(t)$ required by the fixed-step method with $\Delta = 0.01$ (*triangles*), by the T&E method (*circles*), and by the predictive method with $\theta = 3/2$ and $\omega = 1$ (*open squares*) and $\omega = 1/2$ (*solid squares*), to solve problem (14) with $\gamma = 1/4$ up to time t. In all cases $\Delta x = \pi/40$, $\tau = 10^{-4}$ and $\Delta_0 = 0.01$. The *lines* are guides to the eye; their slopes (0.2 for the *dashed line*, 0.1 for the *dotted line*, 2 for the *solid line*) provide estimates of the power exponent β in $T_{\mathrm{CPU}}(t) \sim t^{\beta}$

Table 1 CPU time (in *seconds*) employed to get the solution of (14) with $\gamma = 1/4$ up to time t by (i) the method with fixed timesteps (FT) with $\Delta = 0.01$, and (ii) by the T&E method with tolerance $\tau = 10^{-4}$. In both methods $\Delta x = \pi/40$. The CPU time required by the FT method to find the solution at $t = 502.4$ is an estimate obtained by extrapolation (see Fig. 2)

t	0.12	1.29	2.58	3.22	8.34	502.4
T_{CPU} (FT)	0.2	8	33	49	346	$\gtrsim 2$ weeks
T_{CPU} (T&E)	17	26	29	29	32	43

T&E method one finds that $T_{\mathrm{CPU}} \propto t^{\beta}$ with $\beta \approx 0.2$. That is, the growth of the computational time is not quadratic, not even linear, but sublinear! In fact, the growth is even slower for longer times ($\beta \approx 0.1$)! This means that, except for short times, the adaptive method is immensely faster than the standard method with fixed timesteps. In Table 1 we give some specific values of $T_{\mathrm{CPU}}(t)$ in seconds. It is clear that the standard method becomes useless for times t above a few tens when $\Delta_n = 0.01$ or, equivalently, when the number of timesteps is above a few thousands.

The CPU times of the T&E method given in Fig. 2 and Table 1 correspond to a tolerance $\tau = 10^{-4}$. In Fig. 3, we show $T_{\mathrm{CPU}}(t)$ for several values of τ. As expected, the speed of the method increases when the tolerance increases (of course, the price to be paid is that the method is then less accurate as we shall see below). It is interesting to note that Fig. 3 shows that the CPU time is, in fact, roughly proportional to $\tau^{-\eta}$ with η around 1. For example, when τ changes from $\tau = 10^{-3}$ to $\tau = 10^{-4}$, ones sees that the CPU time increases approximately by a factor of ten. For other values of γ, a very similar behaviour is found. This can be explained by the following back-of-the-envelope argument. By construction, $\tau \approx \mathcal{E}^{(n)}$, but, as will be seen in Section 3.2, $\mathcal{E}^{(n)} \sim \Delta_n^{\theta}$ where $\theta \approx 3/2$. Then $\Delta_n \sim \tau^{1/\theta}$. For a given time, t, $T_{CPU}(t) \sim n^2 \sim (t/\Delta t)^2$ where Δt is here the average value of the timesteps given until time t. But if the size of the timesteps scales roughly as $\tau^{1/\theta}$, then one expects that its average

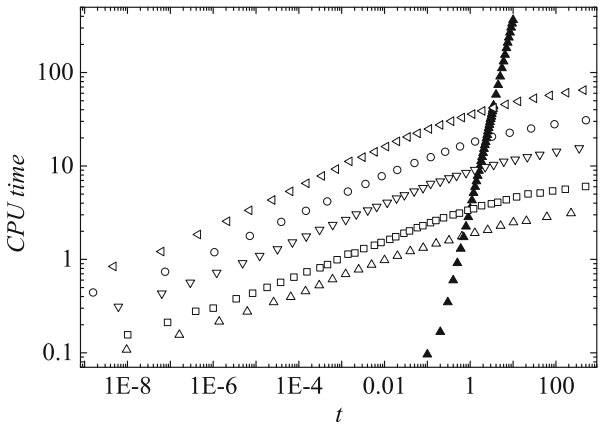

Fig. 3 Normalized computational time T_{CPU} vs t for the method with fixed timesteps with $\Delta = 0.01$ (*solid triangles*) and for the T&E method with tolerance 10^{-5} (left *triangles*), 5×10^{-4} (*circles*), 10^{-4} (down triangles), 2×10^{-4} (*squares*), 10^{-3} (*open up triangles*). In all cases $\gamma = 1/4$, $\Delta x = \pi/40$ and $\Delta_0 = 0.01$

should scale similarly, $\Delta t \sim \tau^{1/\theta}$. Therefore $T_{CPU}(t) \sim \tau^{-\eta}$ with $\eta = 2/\theta$. For $\theta \approx 3/2$ one gets $\eta \approx 1.3$, which is not far from 1.

Figure 4 shows how the computational time $T_{\rm CPU}(t)$ depends on the fractional parameter γ. One sees that, approximately, $T_{\rm CPU}(t) \sim t^{\beta}$ and that β increases when γ increases ($\beta \to 2$ when $\gamma \to 1^-$). However, the value $\gamma = 1$ is singular: there is a drastic change of the CPU times between $\gamma \to 1^-$ and $\gamma = 1$. The reason for this is clear: for $\gamma = 1$ the operator $\partial\gamma/\partial t^{\gamma}$ on the left hand side of (1) is a pure differential operator (i.e., a local operator, no longer an integro-differential operator), and then it is not necessary to carry out the sum of the right hand side of (10), which is what makes fractional finite difference methods so (increasingly) slow. For this reason, the CPU times required to integrate the normal diffusion problem ($\gamma = 1$) are far shorter than the CPU times for subdiffusion problems ($\gamma < 1$). Note that for $\gamma < 1$, no matter how close is γ to unity, one has to spend a lot of computation time evaluating this sum even though one knows that the closer γ is to unity the closer the sum is to zero.

Regarding the error, one sees in Fig. 5 that the adaptive algorithm provides quite homogeneous errors, that is, this technique has the convenient property that it keeps the errors to a desired degree of accuracy, neither too large nor too small. This should be compared with the quite uneven errors of the standard method with fixed timesteps. (In short, non-uniform timesteps lead to uniform errors while uniform timesteps lead to quite non-uniform errors.) Finally, one sees in Fig. 5 that the errors are close to the tolerance, in particular, that they are of the same order of magnitude (around three times the tolerance in this case). This example illustrates the fact that this adaptive algorithm has the nice property that the tolerance, a quantity one can fix at will, is a convenient *indicator* of the accuracy of the numerical method. In Ref. [22] we provide a MATHEMATICA code where this adaptive T&E method is implemented and some of the examples considered in this paper are solved.

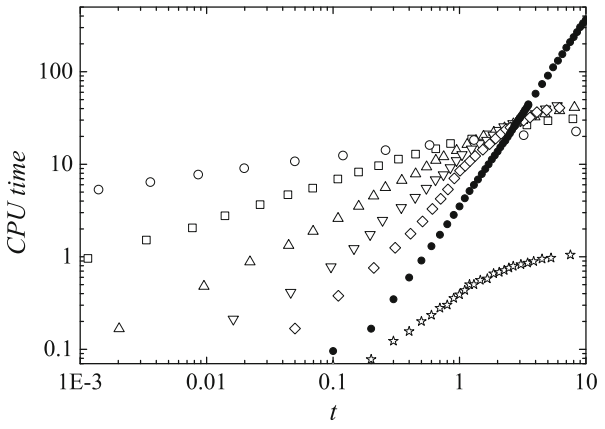

Fig. 4 Normalized computational time $T_{\rm CPU}$ vs t when problem (14) is solved by means of the T&E method with $\tau = 10^{-4}$ for $\gamma = 0.25, 0.5, 0.75, 0.9, 0.99, 1$ (*open circles*, *squares*, *up triangles*, *down triangles*, *diamonds*, *stars*, respectively) and by the method with fixed timesteps with $\Delta = 0.01$ (*solid circles*). In all cases $\Delta x = \pi/40$ and $\Delta_0 = 0.01$

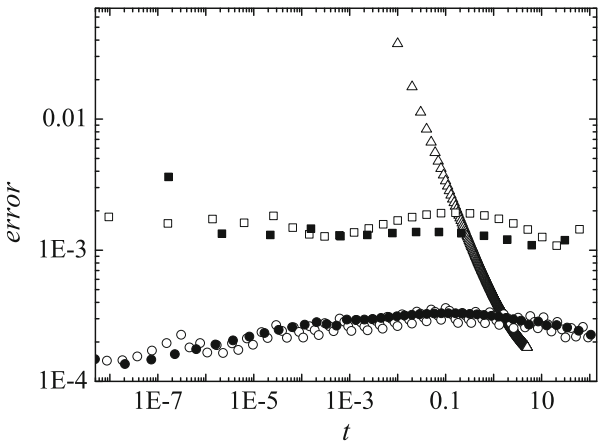

Fig. 5 Maximum absolute numerical error vs. time when problem (14) for $\gamma = 1/4$ is solved by means of the fixed step method with $\Delta_n = 0.01$ (*triangles*), the T&E method with tolerance 10^{-4} (*open circles*) and 10^{-3} (*open squares*), and the predictive method with $\omega = 1/2$, and tolerance 10^{-4} (*solid circles*) and 10^{-3} (*solid squares*). In all cases $\Delta x = \pi/40$ and $\Delta_0 = 0.01$

Indeed, this method is both fast and accurate. In the next subsection, we present another adaptive method, the predictive method, that is as accurate as the T&E method but a bit faster.

3.2 Predictive method

The adaptive predictive method is also based on the step-doubling technique. The starting point is to assume that the difference $\mathcal{E}^{(n)}$ scales as a power of the size of the timesteps:

$$\mathcal{E}^{(n)} \sim \Delta_n^{\theta}. \tag{16}$$

Provided this relationship holds, and from the value of the difference $\mathcal{E}^{(n)}_{\text{old}}$ for a given timestep Δ_n^{old}, one can easily predict the size of the timestep Δ_n^{pred} that leads to an error equal to the tolerance, $\mathcal{E}^{(n)}_{\text{pred}} = \tau$, namely,

$$\Delta_n^{\text{pred}} = \Delta_n^{\text{old}} \left[\frac{\tau}{\mathcal{E}^{(n)}_{\text{old}}} \right]^{1/\theta}. \tag{17}$$

One expects that the direct use of this timestep should spare one from wasting computer time trying to find the right timestep (the one that leads to a difference $\mathcal{E}^{(n)}$ of the order of the tolerance) by means of a blind succession of trials and errors of the size of the timestep as the T&E method does. This prompts us to propose the following predictive step-doubling algorithm:

1. If, for the initial value of Δ_n, the difference satisfies

$$\tau/2 \leq \mathcal{E}_n \leq 2\tau, \tag{18}$$

then this timestep is accepted.

2. Otherwise, we employ a new timestep given by the formula

$$\Delta_n^{\text{new}} = \omega\, \Delta_n^{\text{old}} \left[\frac{\tau}{\mathcal{E}_{\text{old}}^{(n)}} \right]^{1/\theta} + (1-\omega)\, \Delta_n^{\text{old}} \tag{19}$$

until the corresponding difference $\mathcal{E}_{\text{new}}^{(n)}$ satisfies condition (18).

The starting value for Δ_n is just Δ_{n-1}, and we take $\Delta_0 = 0.01$ in all cases. As mentioned previously, the particular initial value for Δ_0 is largely irrelevant because the above step-doubling algorithm, as does the T&E algorithm, quickly finds the right timestep Δ_0. The parameter $0 < \omega \leq 1$ is a kind of under-relaxation parameter [8, 21] that affects the speed and robustness of the algorithm. We have found that the (pure predictive) parameter $\omega = 1$ usually leads to the fastest adaptive algorithm (see Fig. 2). However, in some rare cases, the pure predictive method breaks down because the choice of the timesteps enters into an infinite loop $\Delta_n^a \to \Delta_n^b \to \Delta_n^a \ldots$ due to the fact that their corresponding differences $\mathcal{E}^{(n)}$ never fulfill the (exit) condition (18). When this happens, we have found that the use of an under-relaxation value ($0 < \omega < 1$) fixes this problem. In particular, we have never found this kind of problem for $\omega = 1/2$.

The predictive method relies on (16) so that it is pertinent to check the validity of this power-law scaling. Figure 6 shows the values of θ obtained by fitting $\mathcal{E}^{(n)}$ to (16) for several values of γ, number of timesteps n, and values of Δ_n. Specifically, we studied how $\mathcal{E}^{(n)}$ scales with the size of the the last timestep Δ_n, employing for Δ_n the values $m\Delta_{n-1}$ and also $m\Delta_{n-1}/3$ with $m = 1, 2, \ldots 10$, with Δ_{n-1} being the size of the previous timestep. The rationale for the choice of these ratios $\Delta_n/\Delta_{n-1} = m, m/3$ is that they are of order of the typical values we find in our numerical experiments (in fact, these values are usually one half, one or two). The exponent θ is always between one and two, which is remarkable if one realizes that

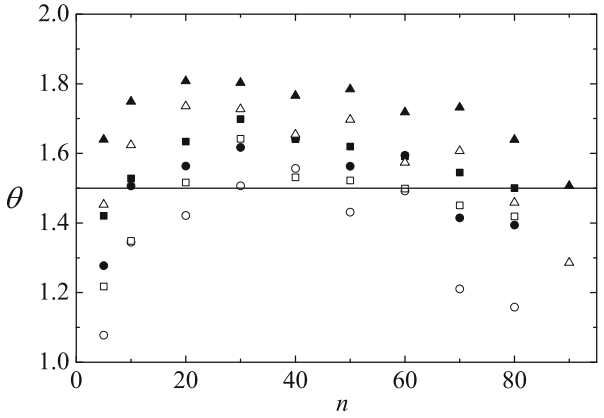

Fig. 6 Scaling exponent θ of the predictive method vs. the number of timesteps when problem (14) is solved for several values of γ (*circles*: $\gamma = 0.25$, squares: $\gamma = 0.5$, triangles: $\gamma = 0.75$) and $\Delta_n = m\Delta_{n-1}$ (*open symbols*) and $\Delta_n = m\Delta_{n-1}/3$ (*solid symbols*) with $m = 1, 2, \ldots 10$. The line marks the value $\theta = 3/2$. In all cases $\Delta x = \pi/80$, $\omega = 1$ and $\Delta_0 = 0.01$

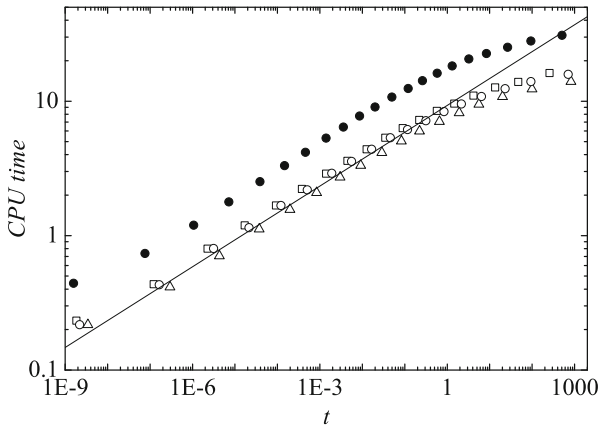

Fig. 7 Normalized computational time T_{CPU} vs. t when problem (14) with $\gamma = 1/4$ is solved by means of the T&E method (*stars*) and the predictive method with $\theta = 3/2$ (*squares*), $\theta = 5/4$ (*up triangles*), and $\theta = 5/3$ (*circles*). In all cases $\Delta x = \pi/40$, $\omega = 1$, $\Delta_0 = 0.01$ and $\tau = 10^{-4}$. The *line*, of slope 0.2, is a guide to the eye

the size of Δ_n spans several orders of magnitude, from around 10^{-8} to 10^2. As a simple overall effective value, we always use $\theta = 3/2$ in this paper. Of course, this choice would be questionable if the method were very sensitive to the specific value of θ. It turns out that this is not the case. Regarding the computational time, Fig. 7 shows that the specific value of θ is hardly relevant. However, the predictive method is faster (around three times faster) than the T&E method in this example. Regarding the errors, one sees in Fig. 5 that their behaviour is quite similar to the behaviour of the errors of the T&E method. For both methods the errors are nicely close to the prefixed tolerance. In Ref. [22] we provide a MATHEMATICA code where the predictive method is implemented and compared with the T&E method for some of the examples considered in this paper.

4 Four examples and a further comparison between the T&E and predictive algorithms

4.1 A problem with a less simple initial condition

The example we consider here is similar to the "testbed" problem of Section 3, that is, (14) but now with the initial condition

$$u(x,0) = \frac{4x}{\pi}\left(1 - \frac{x}{\pi}\right). \tag{20}$$

Its exact solution is now less simple:

$$u(x,t) = \sum_{n=0}^{\infty} \frac{32}{\pi^3(2n+1)^3} E_\gamma\left[-(2n+1)^2 t^\gamma\right] \sin\left[(2n+1)x\right] \tag{21}$$

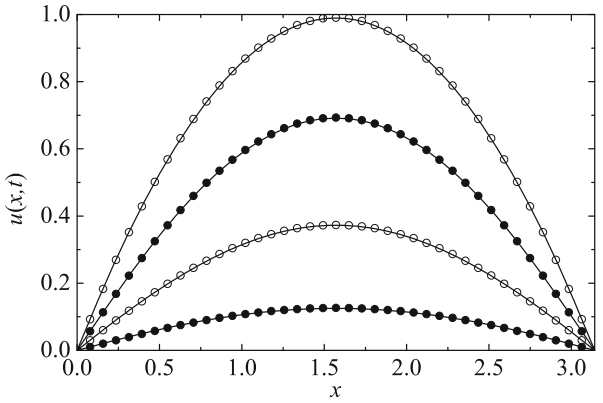

Fig. 8 Solution $u(x,t)$ vs. x for the problem of Section 4.1 with $\gamma = 1/4$ and, from top to bottom, $t = 1.91 \times 10^{-8}$, $t = 3.19 \times 10^{-2}$, $t = 5.12$, and $t = 1.26 \times 10^{3}$. The *symbols* are the numerical solution by the T&E method (*open symbols*) and predictive method with $\omega = 1/2$ (*solid symbols*). In all cases $\tau = 0.001$, $\Delta x = \pi/40$, $\Delta_0 = 0.01$. The *lines* are the exact solution (21)

In Figs. 8 and 9 we compare this exact solution with the numerical solutions provided by the T&E and predictive methods for $\gamma = 1/4$. In Fig. 8 we show the full solution $u(x,t)$ for several times and in Fig. 9 we just show the solution at the middle point, $u(\pi/2,t)$, as well as the errors of the two numerical methods. The agreement is excellent. Due to the fact that the solution of this problem changes very fast for short times and very slowly for long times, in Fig. 9 we have used a log-log plot in order to see clearly how this solution evolves for all times. The errors shown in the inset of Fig. 9 are quite similar to those of the "testbed" problem shown in

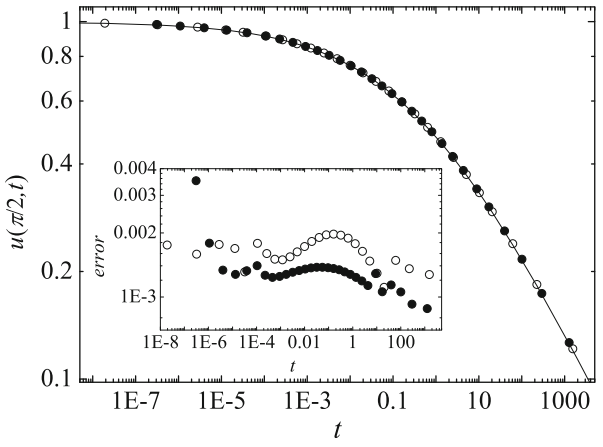

Fig. 9 Solution and numerical errors (inset) at the midpoint $u(\pi/2,t)$ vs. t for the problem of Section 4.1 with $\gamma = 1/4$. The *symbols* are the numerical solution by the T&E method (*open symbols*) and predictive method with $\omega = 1/2$ (*solid symbols*). In all cases $\tau = 0.001$, $\Delta x = \pi/40$, $\Delta_0 = 0.01$. The *line* is the exact solution (21)

Fig. 5. In fact, Figs. 2–7 would have changed little if the initial condition (20) had been used.

In Figs. 8 and 9 no results obtained by means of the method with fixed timesteps are plotted because, in order to get these solutions for long times (say for times around 10^3) employing a reasonable CPU time, one should use large timesteps (say $\Delta_n \simeq 0.1$); otherwise the number of required timesteps would be too large and impractical (see Table 1). But if timesteps of this size are used, then the evolution of the solution over a significant range is lost; for example, a timestep $\Delta_n = 0.1$ would be unable to show how the maximum of the solution $u(x,t)$ evolves over almost half its range, i.e., from $u(\pi/2, 0) = 1$ to $u(\pi/2, 0.1) \approx 0.624$ (see Fig. 9).

4.2 A problem with a steep source term

A relevant feature of the two adaptive methods we have presented above is that they can dynamically adapt the size of the timesteps according to the behaviour of the solution. For example, if at a given time we introduce an external perturbation into the system, we expect the adaptive method to be able to take care of this through the use of a temporal mesh suited to the behaviour of the perturbation. In this subsection we give a clear example of this. The problem we consider is (1) with $K = 1$, boundary conditions $u(0,t) = u(\pi,t) = 0$, and the source term

$$f(x,t) = \left[1 + \frac{\Gamma(1+p)t^{-\gamma}}{\Gamma(1+p-\gamma)}\right] a t^p \sin x. \tag{22}$$

Its exact solution is

$$u(x,t) = \left[E_\gamma(-t^\gamma) + a t^p\right] \sin x. \tag{23}$$

Here we take $a = p = 20$ because this leads to a solution with three different time regimes: the short-time regime where the solution changes very fast, the intermediate regime (roughly until times a little below $t = 1$) where the solution changes very slowly, and a final regime for longer times where again the solution changes very fast. This case is then a good example with which to test adaptive methods. The results provided by the T&E method, the predictive method with $\omega = 1/2$, and the method with fixed timesteps are shown in Fig. 10. Remarkably, we have found that, in some cases, the pure predictive method ($\omega = 1$) breaks down in this example as described in Section 3.2. For example, for $\Delta_0 = 0.01$, $\gamma = 1/4$, $a = p = 20$ and $\tau = 0.001$, we find that the exit condition $\tau/2 < \mathcal{E}^{(n)} < \tau$ for the predictive method with $\omega = 1$ never holds at step $n = 18$ because then the algorithm enters into a loop with $\left\{\Delta_n^a, \mathcal{E}_a^{(n)}\right\} \to \left\{\Delta_n^b, \mathcal{E}_b^{(n)}\right\} \to \left\{\Delta_n^a, \mathcal{E}_a^{(n)}\right\} \to \cdots$ where $\left\{\Delta_n^a, \mathcal{E}_a^{(n)}\right\} \simeq \left\{0.3458, 2.723 \times 10^{-3}\right\}$ and $\left\{\Delta_n^b, \mathcal{E}_b^{(n)}\right\} \simeq \left\{0.1773, 3.672 \times 10^{-4}\right\}$. Finally, it is also remarkable the way in which the size of the timesteps of the adaptive methods changes according to the behaviour of the solution: Fig. 10 shows that for small times and for times around $t = 1$ the solution change very fast and then the adaptive methods tiptoe in these regions keeping the numerical errors small; however, for intermediate times the solution changes very slowly, and the adaptive methods react

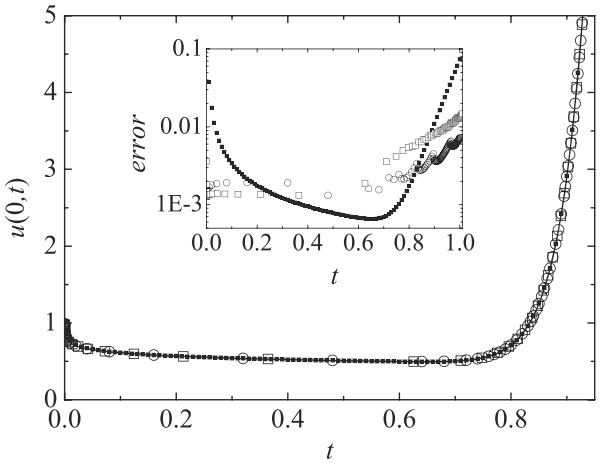

Fig. 10 Solution and numerical errors (inset) at the midpoint $u(\pi/2, t)$ of the problem described in the main text for $\gamma = 1/4$ and source term (22) with $a = p = 20$. *Solid squares*: numerical method with $\Delta_n = 0.01$; *circles*: T&E method with tolerance $\tau = 10^{-3}$; *open squares*: predictive method with $\tau = 10^{-3}$ and $\omega = 1/2$; *line*: exact solution. In all cases $\Delta x = \pi/40$ and $\Delta_0 = 0.01$

by making large strides, thus going fast forward in time although not at the expense of increasing the numerical errors.

4.3 A problem with a quasi-periodic source term

Adaptive methods are specially advantageous for problems with solutions having quite different time scales. If this is not the case, then, apart from being able to roughly pre-set their accuracy, adaptive methods are not much faster than fixed timestep algorithms. In fact, they might be even slower due to the constant computational overload of the continuous searching of nonexistent better (or marginally better) timestep sizes. Here we present an example of this. The problem we consider now is (1) with $K = 1$, boundary conditions $u(0, t) = u(\pi, t) = 0$, and source term

$$f(x, t) = a\left[\sin(\nu t) + \nu^{\gamma} \sin\left(\nu t + \gamma\pi/2\right)\right] \sin x. \tag{24}$$

Its exact solution is

$$u(x, t) = \left[E_{\gamma}(-t^{\gamma}) + a \sin(\nu t)\right] \sin x. \tag{25}$$

Its temporal part is the sum of $E_{\gamma}(-t^{\gamma})$, a function that, for small γ, changes fast for short times and is almost constant for long times, and $a \sin \nu t$, a periodic function. This implies that, except for very short times, one does not expect the adaptive algorithms to be better than the plain fixed timestep algorithm. In Fig. 11 the solution provided by the T&E method, the predictive method with $\omega = 1/2$, and the method with fixed timesteps with $\Delta_n = 0.1$ for the case with $\nu = 2\pi$ and $a = 1/10$ is shown. One sees that the size of the timesteps employed by the adaptive methods is almost constant (around 0.1) when the solution is almost periodic. In other words, the best

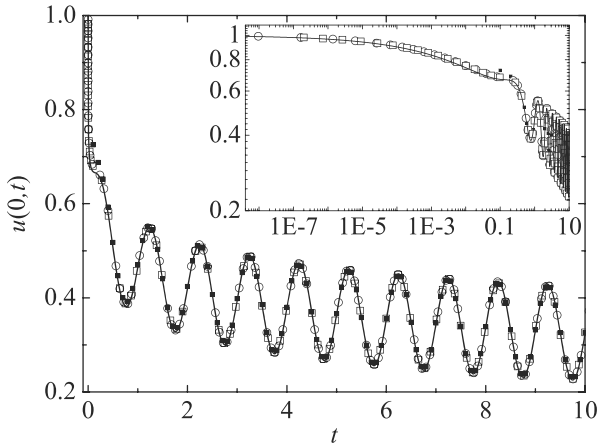

Fig. 11 Solution at the midpoint $u(\pi/2, t)$ vs. time of the problem described in the main text for $\gamma = 1/4$ and source term (24) with $\nu = 2\pi$ and $a = 1/10$. *Solid squares*: numerical method with $\Delta_n = 0.1$; *circles*: T&E method with tolerance $\tau = 10^{-3}$; *open squares*: predictive method with $\tau = 10^{-3}$ and $\omega = 1/2$; *line*: exact solution. In all cases $\Delta x = \pi/40$ and $\Delta_0 = 0.01$. Inset: detail of the solution $u(\pi/2, t)$ for short times

option in this case is to use a fixed timestep. This is what the adaptive methods do, but only after trying to find a better size in every new timestep and find out that the right size hardly changes. On the other hand, the inset of Fig. 11 shows again that the two adaptive methods are excellent when the solution changes fast.

4.4 A problem with non-homogeneous boundary conditions

We want to find the density profile at any time of a set of continuous-time random walkers moving in a one-dimensional finite medium, initially void of walkers, when there is a reservoir of walkers at one end of the medium (so that their concentration is constant there) and they are completely removed from the system at the other end. In mathematical terms, the problem we have to solve then is given by (1) with $0 \le x \le L$, boundary conditions $u(0, t) = u_0$, $u(L, t) = 0$, and initial condition $u(x, 0) = 0$. The exact solution can be obtained by solving the problem in the Laplace space, or directly by means of the method of images [17]:

$$u(x,t) = u_0 \sum_{m=0}^{M} H_{10}^{11}\left[mz_c + z \left| \begin{matrix} 1, & \gamma/2 \\ 0, & 1 \end{matrix} \right. \right] - u_0 \sum_{m=1}^{M} H_{10}^{11}\left[mz_c - z \left| \begin{matrix} 1, & \gamma/2 \\ 0, & 1 \end{matrix} \right. \right], \tag{26}$$

with $M \to \infty$, and where $z = x/(Kt^\gamma)^{1/2}$, $z_c = 2L/(Kt^\gamma)^{1/2}$, and H_{10}^{11} is a Fox H function [17, 24]. When $\gamma = 1$, the Fox function becomes the complementary error function $H_{10}^{11}\left[z \left| \begin{smallmatrix} 1,1/2 \\ 0,1 \end{smallmatrix} \right. \right] = \mathrm{erfc}(z/2)$, and the classical solution [25, Eq. (6), p. 310] is recovered.

In Fig. 12, we compare the exact solution with the numerical solution obtained with the T&E and predictive methods. This problem makes clear the immense advantage of adaptive methods over methods with fixed timesteps. In order to find the solution close to the stationary state, one has to reach times around 10^4. Therefore, in order to get this solution by means of a reasonable number of timesteps of fixed size, one has to use large timesteps, let us say 10^4 timesteps of size $\Delta_n = 1$, which means that all the changes of the system from $t = 0$ up to $t = \Delta_n = 1$ would be overlooked. In our case, see Fig. 12, this would mean overlooking a time interval in which substantial and relevant changes in the solution occurs. In other words, Fig. 12 shows that, in order to conveniently track the solution from the initial condition to the stationary state, one has to employ times that span twelve orders of magnitude (from $t \sim 10^{-8}$ to $t \sim 10^4$). No computer employing finite difference methods with fixed timesteps can handle this problem in a reasonable computation time.

4.5 A further comparison of the speed of the T&E and predictive methods

A different way of checking the efficiency of the adaptive methods is counting the number of times M that the system (10) has to be generated and solved in order to get the solution $u(x, t)$ at time t. The evaluation and solution of system (10) for increasing values of the time t_n is what makes these finite difference methods increasingly slow. Therefore, the smaller the number M required to get the solution for a given time, the better. Note that every time an adaptive algorithm tries a new timestep, this number increases by one.

In Fig. 13 we show the time t vs. M when the two adaptive methods are employed to solve the four examples considered in Sections 4.1–4.4. For comparison, the cor-

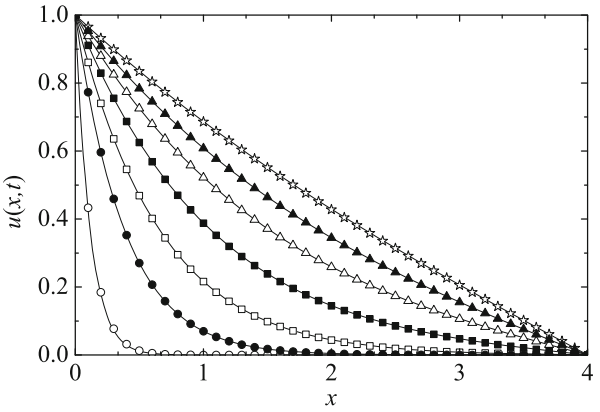

Fig. 12 Solution $u(x, t)$ vs. x of the problem with nonhomogeneous boundary condition described in the main text for $\gamma = 1/4$, $K = 1$, $L = 4$, and several values of t. The *lines* are the exact solution given by (26) with $M = 8$, the *open symbols* are the numerical solutions obtained by means of the T&E method, and the solid symbols correspond to the predictive method with $\omega = 1/2$. For both methods $\tau = 10^{-3}$. The solutions correspond to (from *left* to *right*) $t = 1.91 \times 10^{-8}, 2.67 \times 10^{-4}, 2.00 \times 10^{-2}, 8.93 \times 10^{-1}, 2.05 \times 10^{1}, 2.68 \times 10^{2}, 1.14 \times 10^{4}$. In all cases, $\Delta x = 0.1$, $\tau = 10^{-3}$ and $\Delta_0 = 0.01$

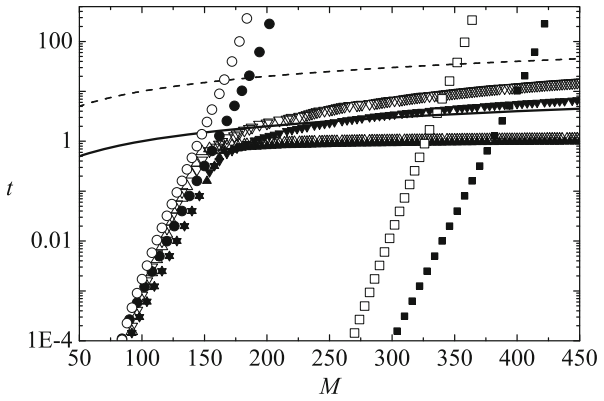

Fig. 13 Number of times M the system (10) has to be solved to get the solution $u(x,t)$ up to time t for the problem considered in Sections 4.1 (*circles*), 4.2 (*down triangles*), 4.3 (*up triangles*) and 4.4 (*squares*). *Open symbols*: T&E method; *solid symbols*: predictive method with $\omega = 1/2$. In all cases $\tau = 0.001$ and $\gamma = 1/4$. The *lines* represent M for fixed timesteps, $M = t/\Delta t$, with $\Delta t = 0.01$ (*solid line*) and $\Delta = 0.1$ (*broken line*)

responding values $M = t/\Delta t$ for fixed timesteps of size $\Delta = 0.1$ and $\Delta = 0.01$ are also provided. We see again that the two adaptive methods are far better than the method with fixed timesteps for the examples considered in Sections 4.1 and 4.4, two cases where the solution changes very slowly for long times. For the case discussed in Section 4.3 in which the solution is quasi periodic, the efficiency of the two adaptive methods is similar to the efficiency of the method with fixed timestep. This is not unexpected because, as we showed in Section 4.3, the adaptive methods end choosing almost constant timesteps for the long time regime where the solution is quasi-periodic. Finally, for the case of Section 4.2 with a steep source term, the adaptive methods get stuck for times around 1 in order to keep the accuracy of the numerical solution (see Fig. 10). We see that the convenience of using adaptive methods depends on the problem one has to solve. Regarding the speed of method, adaptive methods are not advisable when the rate of change of the solution does not vary largely along the time interval of integration because, in this case, there is no need to adapt the size of the timesteps. Finally, Fig. 13 shows that the predictive method is always faster than the T&E method for the four examples considered in Sections 4.1–4.4.

5 Summary and conclusions

A major drawback of finite difference methods for fractional equations is that obtaining the solution for every new timestep is increasingly costly in terms of computational time. This implies that the number of timesteps required to find a solution should be as small as posible while keeping a reasonable accuracy in the numerical solution. In this respect, finite difference methods with homogeneous

timesteps have the additional drawback that they typically provide numerical results with quite uneven accuracy (small in some time regions but relatively large in others).

In order to lessen these two problems, we have proposed the use of adaptive methods with adaptive timesteps. This kind of method has the great advantage that the size of the timesteps can be tailored to the behaviour of the solution. For example, one can choose small timesteps only when the solution is changing rapidly in order to keep the accuracy of the method and to track down the relevant features of the solution at these time scales. On the other hand, one can choose large timesteps if the solution changes slowly. In this way, without losing accuracy, the method can advance with large strides when feasible so that long times can be reached.

We have studied two different adaptive methods. Both employ an integration algorithm based on the L1 discretization of the Caputo fractional derivative with non-homogeneous timesteps, but differ in their adaptive algorithm, i.e., in the way in which the size of every timestep is determined. Both methods are fast (immensely faster than the corresponding non-adaptive method with fixed timesteps in some cases) and provide solutions with an accuracy that, to a large extent, is consistent throughout the integration time interval. Remarkably, this accuracy can be pre-set at will through setting up a kind of self-consistent parameter (the tolerance) that is, in many cases, an excellent indicator of the final accuracy of the numerical solution.

Acknowledgments This work was partially funded by the Ministerio de Ciencia y Tecnología (Spain) through Grant No. FIS2013-42840-P (partially financed by FEDER funds) and by the Junta de Extremadura through Grant. No. GR15104.

Appendix

We have seen that the tolerance we set in the adaptive methods of Section 3 turns out to be a reasonable indicator of the numerical error (see Figs. 5, 9 and 10). In this appendix we explore this issue from a theoretical perspective. The procedure and results are somewhat parallel to those of ordinary differential equations [8, 9], although far more involved.

Zhang et al. [15] have shown that the numerical error at point (x_j, t_n), $e_j^{(n)} = |u(x_j, t_n) - U_j^{(n)}|$, is bounded by the sum of a term coming from the time discretization, $e_{t,j}^{(n)}$ of the fractional derivative, and a term coming from the space discretization $e_{x,j}^{(n)}$ of the Laplacian: $e_j^{(n)} \leq e_{t,j}^{(n)} + e_{x,j}^{(n)}$. In our case, the Laplacian is discretized by means of the well-known three-point centered formula so that $e_{x,j}^{(n)} = C_{x,j}^{(n)} \Delta_x^2 = O(\Delta_x^2)$, $C_{x,j}^{(n)}$ being a coefficient that depends on the solution $u(x, t)$ at (x_j, t_n). The temporal bound is less simple [15]:

$$e_{t,j}^{(n)} = C_j^{(n)} \max_{1 \leq k \leq n} r^{(k)} = O\left(\max_{1 \leq k \leq n} r^{(k)} \right) \tag{27}$$

where

$$r^{(k)} = \left(\frac{\Delta_k^2}{2(1-\gamma)} + \frac{\Delta_{\max,k}^2}{8} \right) \Delta_k^{-\gamma}, \tag{28}$$

$$\Delta_{\max,k} = \max_{1 \le m \le k} \Delta_m, \tag{29}$$

and $C_j^{(n)}$ is a coefficient, proportional to $t_n^{\gamma/2}$, that depends on γ and on the maximum value of $|\partial^2 u/\partial t^2|$ at x_j in the time interval $[t_0, t_n]$. From here on we assume that $e_{t,j}^{(n)} + e_{x,j}^{(n)}$ provides an estimate of the error, that is, $e_j^{(n)} \approx e_{t,j}^{(n)} + e_{x,j}^{(n)}$. We also assume that $e_{x,j}^{(n)}$ is smaller than the tolerance τ (otherwise the adaptive method does not work because no temporal mesh can make the error smaller than τ). Next we define $\hat{e}_{t,j}^{(n)}$ as the numerical error when the solution at t_n is computed from the solution at time t_{n-1} using two timesteps of size $\Delta_n/2$, that is, $e_{t,j}^{(n)}$ is the error when the time discretization is $\{t_0, t_1, \ldots, t_{n-1}, t_n\}$ and $\hat{e}_{t,j}^{(n)}$ is the corresponding error, $\hat{e}_{t,j}^{(n)} = \left| u(x_j, t_n) - \hat{U}_j^{(n)} \right|$, when the discretization is $\{t_0, t_1, \ldots, t_{n-1}, t_{n-1/2}, t_n\}$ with $t_{n-1/2} = (t_{n-1} + t_n)/2$. It is easy to see from (27)–(29) that

$$e_{t,j}^{(1)} = O\left(\frac{\Delta_1^{2-\gamma}}{2(1-\gamma)} \right)$$

whereas

$$\hat{e}_{t,j}^{(1)} = O\left(\frac{(\Delta_1/2)^{2-\gamma}}{2(1-\gamma)} + \frac{(\Delta_1/2)^{2-\gamma}}{8} \right).$$

The difference between these two quantities, $\hat{e}_{t,j}^{(1)} - e_{t,j}^{(1)}$, is of the same order than $e_{t,j}^{(1)}$, which means that $e_{t,j}^{(1)}$ and $\mathcal{E}_j^{(1)} = |\hat{U}_j^{(1)} - U_j^{(1)}|$ are of the same order, too. In our adaptive methods the size of the timestep Δ_1 is then chosen so that $\mathcal{E}^{(1)} = \max_{\text{all } j} \mathcal{E}_j^{(1)} = \mathcal{E}_{j*}^{(1)}$ is of order of the tolerance, $\mathcal{E}^{(1)} \approx \tau$, where $j*$ denotes the index (position) for which $\mathcal{E}_j^{(n)}$ is maximum. Next we proceed by induction. Let us assume that we have evaluated the numerical solutions until time t_{n-1} in such a way that $\mathcal{E}^{(m)} \approx \tau$ for $m = 1, \ldots n-1$, $j*$ being the index for which $\mathcal{E}_{j*}^{(n-1)} \approx \tau$. In this case, from (27) one sees that

$$e_{t,j*}^{(n)} = C_{j*}^{(n)} \max \left\{ e_{t,j*}^{(n-1)} / C_{j*}^{(n-1)}, r^{(n)} \right\}. \tag{30}$$

or, equivalently,

$$e_{t,j*}^{(n)} \approx \max \left\{ \tau, C_{j*}^{(n)} \left(\frac{\Delta_n^2}{2(1-\gamma)} + \frac{\Delta_{\max,n}^2}{8} \right) \Delta_n^{-\gamma} \right\} \tag{31}$$

if one assumes that $C_{j*}^{(n)} / C_{j*}^{(n-1)} \approx 1$. Similarly,

$$\hat{e}_{t,j*}^{(n)} \approx \max \left\{ \tau, C_{j*}^{(n)} \left(\frac{(\Delta_n/2)^2}{2(1-\gamma)} + \frac{\Delta_{\max,n}^2}{8} \right) (\Delta_n/2)^{-\gamma} \right\}. \tag{32}$$

The difference between these two quantities, $\hat{e}^{(n)}_{t,j*} - e^{(n)}_{t,j*}$ is then of the same order than $e^{(n)}_{t,j*}$, which means that $e^{(n)}_{t,j*}$ and $\mathcal{E}^{(n)}_{j*} = |\hat{U}^{(n)}_{j*} - U^{(n)}_{j*}|$ are of the same order, too. We then choose Δ_n so that $\mathcal{E}^{(n)}_{j*}$ is of order of the tolerance. In the previous reasoning we have assumed implicitly that $j*$ for timestep $n-1$, j^*_{n-1}, is the same than for timestep n, j^*_n. Of course, although the value of j^* may change (typically $j^*_n = j^*_{n-1}$ and sometimes $j^*_n = j^*_{n-1} \pm 1$), we still expect that $\mathcal{E}^{(n)}_{j^*_{n-1}} \approx \mathcal{E}^{(n)}_{j^*_n} \approx \tau$ provided that $u(x_{j^*_{n-1}}, t_n)$ is similar to $u(x_{j^*_n}, t_n)$.

References

1. Klafter, J., Lim, S.C., Metzler, R. (eds.): Fractional Dynamics: Recent Advances. World Scientific, Singapore (2011)
2. Yang, Q.: Novel Analytical and Numerical Methods for Solving Fractional Dynamical Systems. Ph.D. thesis, Queensland University of Technology (2010)
3. Li, C., Zeng, F.: Finite difference methods for fractional differential equations. Int. J. Bifurcat. Chaos **22**, 1230014 (2012)
4. Deng, W.: Short memory principle and a predictor–corrector approach for fractional differential equations. J. Comput. Appl. Math. **206**, 174–188 (2007)
5. Podlubny, I. Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of their Solution and some of their Applications, 1st edn. Academic Press, San Diego (1999)
6. Ford, N., Simpson, A.: The numerical solution of fractional differential equations: speed versus accuracy. Numer. Algoritm. **26**, 333–346 (2001)
7. Mainardi, F., Gorenflo, R.: On Mittag-Leffler-type functions in fractional evolution processes. J. Comput. Appl. Math **118**, 283–299 (2000)
8. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. Numerical Recipes: The Art of Scientific Computing, 3rd edn. Cambridge University Press, New York (2007)
9. Gear, C.W.: Numerical Initial Value Problems in Ordinary Differential Equationns. Prentice-Hall, Englewood Cliffs (1971). Sect. 5.4
10. Podlubny, I., Skovranek, T., Vinagre Jara, B.M., Petras, I., Verbitsky, V., Chen, Y.: Matrix approach to discrete fractional calculus III: non-equidistant grids, variable step length and distributed orders. Philos. T. Roy. Soc. A **371**, 20120153 (2013)
11. Mustapha, K.: An implicit finite-difference time-stepping method for a sub-diffusion equation, with spatial discretization by finite elements. IMA J. Numer. Anal. **31**, 719–739 (2010)
12. Mustapha, K., AlMutawa, J.: A finite difference method for an anomalous sub-diffusion equation, theory and applications. Numer. Algoritm. **61**, 525–543 (2012)
13. Yuste, S.B., Quintana-Murillo, J.: A finite difference method with non-uniform timesteps for fractional diffusion equations. Comput. Phys. Commun. **183**, 2594–2600 (2012)
14. Quintana-Murillo, J., Yuste, S.B.: A finite difference method with non-uniform timesteps for fractional diffusion and diffusion-wave equations. Eur. Phys. J. Spec. Top. **222**, 1987–1998 (2013)
15. Zhang, Y.-N., Sun, Z.-Z., Liao, H.-L.: Finite difference methods for the time fractional diffusion equation on non-uniform meshes. J. Comput. Phys. **265**, 195–210 (2014)
16. Lopez-Fernandez, M., Sauter, S.: Generalized convolution quadrature with variable time stepping. IMA J. Numer. Anal. **33**, 1156–1175 (2013)
17. Metzler, R., Klafter, J.: The random walk's guide to anomalous diffusion: a fractional dynamics approach. Phys. Rep. **339**, 1–77 (2000)
18. Oldham, K.B., Spanier, J.: The fractional calculus; theory and applications of differentiation and integration to arbitrary order. Academic Press, New York (1974)
19. Liu, F., Zhuang, P., Anh, V., Turner, I.: A fractional-order implicit difference approximation for the space-time fractional diffusion equation. ANZIAM J. **47**, C48–C68 (2006)
20. Murio, D.A.: Implicit finite difference approximation for time fractional diffusion equations. Comput. Math. Appl. **56**, 1138–1145 (2008)

21. LeVeque, R.: Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems. Society for Industrial and Applied Mathematics, Philadelphia (2007)
22. See supplementary material at http://www.eweb.unex.es/eweb/fisteor/santos/Adap.zip where a MATHEMATICA code demonstrating how our calculations are carried out is available
23. Yuste, S.B., Quintana-Murillo, J.: Corrigendum to A finite difference method with non-uniform timesteps for fractional diffusion equations [Comput Phys. Comm. **183**, 2594–2600 (2012)]. Comput. Phys. Commun. **185**, 1192 (2014)
24. Mathai, A., Saxena, R.: The H-Function with Applications in Statistics and other Disciplines. Wiley, New York (1978)
25. Carslaw, H.S., Jaeger, J.C.: Conduction of Heat in Solids. Oxford University Press, Oxford (1959)