A Stable Fast Time-Stepping Method for Fractional Integral and Derivative Operators

Fanhai Zeng · Ian Turner1,2 · Kevin Burrage1,3

Received: 7 November 2017 / Revised: 19 February 2018 / Accepted: 4 April 2018 / Published online: 26 April 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract A unified fast time-stepping method for both fractional integral and derivative operators is proposed. The fractional operator is decomposed into a local part with memory length $\Delta T$ and a history part, where the local part is approximated by the direct convolution method and the history part is approximated by a fast memory-saving method. The fast method has $O(n_0 + \sum L q_\alpha(N_\ell))$ active memory and $O(n_0 n_T + (n_T - n_0) \sum L q_\alpha(N_\ell))$ operations, where $L = \log(n_T - n_0)$, $n_0 = \Delta T / \tau$, $n_T = T / \tau$, $\tau$ is the stepsize, $T$ is the final time, and $q_\alpha(N_\ell)$ is the number of quadrature points used in the truncated Laguerre–Gauss (LG) quadrature. The error bound of the present fast method is analyzed. It is shown that the error from the truncated LG quadrature is independent of the stepsize, and can be made arbitrarily small by choosing suitable parameters that are given explicitly. Numerical examples are presented to verify the effectiveness of the current fast method.

Keywords Fast convolution · The (truncated) Laguerre–Gauss quadrature · Short memory principle · Fractional differential equations · Fractional Lorenz system

This work was supported by ARC Discovery Project DP150103675.
1 Introduction

The convolution of the form

\[(k * u)(t) = \int_0^t k(t - s)u(s) \, ds\]  

(1)

arises in many physical models, such as integral equations, integrodifferential equations, fractional differential equations, and integer-order differential equations such as wave propagation with nonreflecting boundary conditions, see for example, [3, 8, 22, 30, 31, 37].

The aim of this paper is to present a stable and fast memory-saving time-stepping algorithm for the convolution (1) with a kernel \(k(t) = t^{\alpha - 1}/\Gamma(\alpha)\). This kind of kernel has found wide applications in science and engineering [8, 30, 31]. When \(\alpha \geq 0\), Eq. (1) gives a fractional integral of order \(\alpha\). If \(\alpha < 0\), then Eq. (1) can be interpreted as the Hadamard finite part integral, which is equivalent to the Riemann–Liouville (RL) fractional derivative of order \(-\alpha\), see [32, p. 112].

The direct discretization of (1) takes the following form

\[\sum_{k=0}^{n} \omega_{n,k} u(t_k), \quad n = 1, 2, \ldots, n_T,\]  

(2)

where \(\omega_{n,k}\) are the convolution quadrature weights. The direct computation of (2) requires \(O(n_T)\) active memory and \(O(n_T^2)\) operations, which is expensive for long time computations. The computational difficulty in both memory requirement and computational cost will increase greatly when the direct approximation (2) is applied to resolve high-dimensional time evolution equations and/or a large system of time-fractional partial differential equations (PDEs) involving (1), see e.g., [2, 10, 22, 43, 45]. However, we believe this computational difficulty for fractional operators has not been fully addressed in literature. The short memory principle (see [6, 31]) seems promising to resolve this difficulty, but it has not been widely applied in fractional calculus due to its inaccuracy.

Up to now, some progress has been made in reducing storage requirements and computational cost for resolving fractional models. The basic idea is to seek a suitable sum-of-exponentials to approximate the kernel function \(k_\alpha(t) = t^{\alpha - 1}/\Gamma(\alpha)\), i.e.,

\[k_\alpha(t) = t^{\alpha - 1}/\Gamma(\alpha) = \sum_{j=1}^{O} w_j e^{\lambda_j t} + O(\epsilon), \quad t \in [\delta, T]\]  

(3)

where \(\delta, T > 0\) and \(\epsilon > 0\) is a given precision. The key is to determine \(w_j\) and \(\lambda_j\) in (3) in order that the desired accuracy up to \(O(\epsilon)\) can be achieved.

In order to derive (3), Lubich and Schädle [22] expressed \(k_\alpha(t)\) in terms of its inverse Laplace transform \(k_\alpha(t) = \frac{1}{2\pi i} \int_C \mathcal{L}[k_\alpha] e^{\lambda t} \, d\lambda\), where \(\mathcal{L}[k_\alpha](\lambda)\) denotes the Laplace transform of \(k_\alpha(t)\) and \(C\) is a suitable contour. Then a suitable quadrature was applied to approximate \(\int_C \mathcal{L}[k_\alpha] e^{\lambda t} \, d\lambda\), which leads to (3). This approach can be applied to construct fast methods for a wide class of nonlocal models. The method in [22] was then extended to calculate the discrete convolution (2) in [3, 33], where the coefficients \(\omega_{n,j}\) are generated from the generating functions. A graded mesh version of [22] was developed in [20] and an application of [22] in the simulation of fractional-order viscoelasticity in complicated arterial geometries was proposed in [43]. The storage and computational cost of the fast methods in [3, 20, 22, 33] are \(O(\log n_T)\) and \(O(n_T \log n_T)\), respectively, which are much less than the direct methods with \(O(n_T)\) memory and \(O(n_T^2)\) operations. Recently, Baffet and Hesthaven
[1, 2] have proposed to approximate $\mathcal{L}[k_{\alpha}]$ using the multipole approximation, which yields (3).

Another approach to derive (3) is based on the following integral expression

$$k_{\alpha}(t) = \frac{\sin(\alpha \pi)}{\pi} \int_0^\infty \lambda^{-\alpha} e^{-t \lambda} d\lambda, \quad \alpha < 1.$$  \hfill (4)

Equation (4) can be derived by inserting $\mathcal{L}[k_{\alpha}] = \lambda^{-\alpha}$ into Henrici’s formula (see [5]), i.e.,

$$k_{\alpha}(t) = \frac{1}{2\pi i} \int_0^\infty \left[ \mathcal{L}[k_{\alpha}](\lambda e^{-i\pi}) - \mathcal{L}[k_{\alpha}](\lambda e^{i\pi}) \right] e^{-t \lambda} d\lambda.$$  \hfill (4)

Li [18] transformed the above integral into its equivalent form, then multi-domain Legendre–Gauss quadrature was applied to approximate the transformed integral to obtain (3). Jiang et al. [14] combined Jacobi–Gauss quadrature and multi-domain Legendre–Gauss quadrature to discretize (4) for $-1 < \alpha < 1$, then a global after-processing optimization technique was applied to further reduce the number of quadrature points, see also [42]. The exponential sum approximation for $t^{-\beta}$ ($\beta > 0$) has been studied in the literature, which can be used to design fast algorithm to approximate the fractional operators. The interested readers can refer to [4, 28]. In the references [1–3, 20, 22, 33], $w_j$ and $\lambda_j$ are complex, however, they are real in [4, 14, 18, 28].

Apart from the above mentioned fast methods for fractional operators, McLean [27] proposed to use a degenerate kernel for evaluating $k_{\alpha} * u(t)$.

In this work, we derive (3) by approximating (4) using a truncated Laguerre–Gauss (LG) quadrature. We list the main contributions of this work as follows.

1. We follow and generalize the framework in [22] to resolve the short memory principle with a lag-term, see Sect. 3. By choosing suitable parameters, the current method can be simplified as that in [1, 2, 14, 18], where the time domain is not divided into exponentially increasing subintervals. This approach simplifies the implementation of the algorithm.

2. The present fast method unifies the calculation of the discrete convolutions to the approximation of both fractional integral and derivative operators with arbitrary accuracy (see Tables 2, 10), for example, the trapezoidal rule for the fractional integral operator (see [8]) and the L1 method for the fractional derivative operator (see [7, 36]); see Fig. 1.

3. Given any basis $B$ ($B > 1$ is an integer), any stepsize $\tau$, any memory length $\Delta T \geq \tau$, and any precisions $\epsilon, \epsilon_0 > 0$, the truncation number $q_{\alpha}(N_\ell)$ [see (45) and (36)] of the truncated LG quadrature is determined in order that the overall error of the present fast method from the truncated LG quadrature is $O(\epsilon + \epsilon_0)$, see (49). The truncated LG quadrature and/or a relatively smaller basis $B$ saves memory and computational cost, see numerical results in Tables 1, 2 and Fig. 3.

We would like to emphasize that the truncated LG quadrature reduces the memory and computational cost significantly, see Table 1. The memory and computational cost in [20, 22] can be halved due to the symmetry of the trapezoidal rule, but operations with complex numbers are involved. In addition, the Gauss–Jacobi and Gauss–Legendre quadrature used in [14, 18] may not be truncated. Furthermore, the discretization error caused from the LG quadrature is independent of the stepsize and the regularity of the solution to the considered fractional differential equation (FDE), and does not appear to be sensitive to the fractional order $\alpha \in (-2, 1)$ as exhibited in Fig. 3, which is competitive with the mutlipole approximation [2] in both accuracy and memory requirement, see Fig. 4.

In real applications, analytical solutions to FDEs are unknown and often non-smooth, and may have strong singularity at $t = 0$, see, for example, [16, 23, 29, 35]. In order to resolve the singularity of the solution to the considered FDE, a graded mesh approach was adopted by some researchers [16, 29, 35]. In [35], an optimal graded mesh was obtained to achieve the global convergence of order $2 - \alpha, \alpha \in (0, 1)$, which is very effective when the fractional
Fig. 1 Comparison between the trapezoidal rule based on the Talbot contour (black curve) and the LG quadrature (red curve), $\tau = 1, B = 5, N = 100$. The optimal contour $\zeta(\theta, t) = t (-0.4814 + 0.6443(\theta \cot(\theta) + i0.5653))$ obtained in [39] is applied here, i.e., $\hat{\lambda}^{(\ell)}_j$ in (13) is given by $\hat{\lambda}^{(\ell)}_j = \zeta(\theta_j, N/(2T_\ell - t))$ with the corresponding weight $\hat{\omega}^{(\ell)}_j = \partial_\theta \zeta(\theta_j, N/(2T_\ell - t))$, where $\theta_j = (2j + 1)\pi/(2N)$, $j = -N, \ldots, N - 1$ and $T_\ell = B\tau$, $B = 5, N = 32$. a $\alpha = -0.5$, b $\alpha = 0.5$ (Color figure online)

Table 1 The maximum relative error $\|e\|_\infty$ under the precision $\epsilon = 10^{-10}, \alpha = -0.5, \tau = 0.1, \Delta T = 1$, and $T = 10^4$

| B  | $\sum N_\ell$ | $\sum q_\alpha(N_\ell)$ | $\|e\|_\infty$ | B  | $\sum N_\ell$ | $\sum q_\alpha(N_\ell)$ | $\|e\|_\infty$ |
|----|---------------|--------------------------|----------------|----|---------------|--------------------------|----------------|
| 2  | 583           | 389                      | 6.8651e−13     | 30 | 3333          | 545                      | 7.2134e−13     |
| 3  | 605           | 323                      | 6.6718e−13     | 40 | 3583          | 514                      | 7.3190e−13     |
| 4  | 687           | 318                      | 7.4246e−13     | 50 | 4521          | 576                      | 7.3391e−13     |
| 5  | 783           | 320                      | 7.4754e−13     | 60 | 5463          | 637                      | 7.4632e−13     |
| 8  | 1151          | 370                      | 7.4377e−13     | 70 | 6406          | 689                      | 7.3451e−13     |
| 10 | 1246          | 359                      | 7.1388e−13     | 80 | 7345          | 737                      | 7.4678e−13     |
| 15 | 1598          | 376                      | 6.9954e−13     | 90 | 8288          | 785                      | 7.3892e−13     |
| 20 | 2173          | 438                      | 7.0761e−13     | 100| 9231          | 829                      | 7.4941e−13     |

order is relatively large, but is less effective when the fractional order tends to zero. In this paper, we follow Lubich’s approach [21] to deal with the singularity by introducing correction terms.

As we mentioned above, a rational approximation was made in [2] to approximate the Laplace transform of the fractional kernel, while the method in [18] used the Legendre–Gauss quadrature to approximate the transformed integral of (4). These approaches were initially designed for the fractional integral operator. However, we have found that they can be applied to the RL fractional derivative operator directly as done in the present work.

This paper is organized as follows. We follow the approach in [22] to present our fast method in Sect. 2. The short memory principle with lag terms is resolved in Sect. 3, it unifies the calculation of the discrete convolutions to the approximation of the fractional integral and derivative operators. The error analysis of the fast method is presented in Sect. 4, where all the parameters needed in numerical simulations are explicitly given. Numerical simulations are presented to verify the effectiveness of the fast method in Sect. 5 before the conclusion in the last section.
Table 2 The maximum relative error \( \|e\|_\infty \) under different precision \( \epsilon \), \( B = 5 \), \( \alpha = -0.5 \), \( \Delta T = 1 \)

| \( \epsilon \) | \( \tau = 0.01, T = 10^4 \) | \( \tau = 0.1, T = 10^5 \) |
|---|---|---|
| \( 10^{-12} \) | \( \sum N_\ell \) | \( \sum q_\ell \)(\( N_\ell \)) | \( \|e\|_\infty \) | \( \sum N_\ell \) | \( \sum q_\ell \)(\( N_\ell \)) | \( \|e\|_\infty \) |
| \( 10^{-10} \) | 1018 | 384 | 1.1937e−12 | 1203 | 442 | 6.8208e−12 |
| \( 10^{-8} \) | 848 | 349 | 4.7044e−12 | 1001 | 402 | 6.8191e−12 |
| \( 10^{-6} \) | 680 | 313 | 4.7044e−12 | 799 | 361 | 6.5369e−12 |
| \( 10^{-5} \) | 512 | 271 | 3.1858e−09 | 604 | 312 | 1.2450e−10 |
| \( 10^{-4} \) | 427 | 243 | 2.1577e−06 | 503 | 281 | 1.8889e−08 |
| \( 10^{-3} \) | 340 | 215 | 2.1577e−06 | 404 | 252 | 2.3317e−07 |

2 A Stable Fast Convolution

In this section, we follow the approach given in [22] to develop our fast convolution.

The goal is to discretize the right-hand side of (4) using a highly accurate numerical method. It is natural to use LG quadrature to approximate (4), i.e.,

\[
 k_\alpha(t) = \frac{\sin(\alpha \pi)}{\pi} \int_{-\infty}^{\infty} \lambda^{-\alpha} e^{-T \lambda} e^{-(t-T)\lambda} d\lambda \approx \frac{\sin(\alpha \pi)}{\pi} \sum_{j=0}^{N} \omega_j e^{-(t-T)\lambda_j}, \tag{5}
\]

where \( \{\omega_j\} \) and \( \{\lambda_j\} \) are the LG quadrature weights and points that correspond to the weight function \( \lambda^{-\alpha} e^{-T \lambda} \). The quadrature (5) is exponentially convergent for any \( t \geq T \) if \( N \) is sufficiently large, which is discussed in Sect. 4.

Denote \( t_n = n \tau \) \((n = 0, 1, \ldots, n_T) \) as the grid point, where \( \tau \) is the stepsize. We first restrict ourselves to \( \alpha \in [0, 1) \). Using (5) and following the idea in [22], we present our stable fast convolution for approximating \( k_\alpha \ast u(t) \) as follows:

- Step (1) Decompose the convolution \( k_\alpha \ast u(t) \) as

\[
 k_\alpha \ast u(t) = \int_{t-\tau}^{t} k_\alpha(t-s)u(s) \, ds + \int_{0}^{t-\tau} k_\alpha(t-s)u(s) \, ds \equiv L^\alpha(u, t) + H^\alpha(u, t), \tag{6}
\]

where we call \( L^\alpha(u, t) \) and \( H^\alpha(u, t) \) the local and history parts, respectively. Let \( I^{(1)}_\tau \) \( u(t) \) be the linear interpolation of \( u(t) \). Then the local and history parts can be approximated by

\[
 L^\alpha(u, t_n) \approx L^\alpha(I^{(1)}_\tau \, u(t_n)), \quad H^\alpha(u, t_n) \approx H^\alpha(I^{(1)}_\tau \, u(t_n)) = H^\alpha(I^{(1)}_\tau \, u(t_n)) = H^\alpha(I^{(1)}_\tau \, u(t_n)). \tag{7}
\]

- Step (2) For every \( t = t_n \), let \( L \) be the smallest integer satisfying \( t_n < 2B^{L} \tau \), where \( B > 1 \) is a positive integer. For \( \ell = 1, 2, \ldots, L - 1 \), determine the integer \( q_\ell \) such that

\[
 s_\ell = q_\ell B^{\ell} \tau \quad \text{satisfies} \quad t_n - s_\ell \in [B^{\ell} \tau, (2B^{\ell} - 1)\tau]. \tag{7}
\]

Set \( s_0 = t_n - \tau \) and \( s_L = 0 \). Then \( t_n - \tau = s_0 > s_1 > \cdots > s_{L-1} > s_L = 0 \).

- Step (3) Using (5), we approximate the history part \( H^\alpha(I^{(1)}_\tau \, u(t_n)) \) by

\[
 H^\alpha(I^{(1)}_\tau \, u(t_n)) \approx \frac{\sin(\alpha \pi)}{\pi} \sum_{\ell=1}^{L} \sum_{j=0}^{N} \omega_j e^{-(t_n-s_{\ell-1}-T_{\ell-1})\lambda_j} y(s_{\ell-1}, s_\ell, \lambda_j) = f H^\alpha(I^{(1)}_\tau \, u(t_n)) \tag{8}
\]
with \( y(s_{\ell-1}, s_\ell, \lambda) \) given by
\[
y(s_{\ell-1}, s_\ell, \lambda) = \int_{s_{\ell-1}}^{s_\ell} e^{(s-s_{\ell-1})\lambda} I_{s_\ell}^{(1)} u(s) \, ds,
\]
(9)

where \( \{\omega_j^{(l)}\} \) and \( \{\lambda_j^{(l)}\} \) are the LG quadrature weights and points that correspond to the weight function \( \lambda^{-\alpha} e^{-T_{\ell-1}\lambda} \) (see (7.27) in [34]), and \( T_{\ell-1} = B^{\ell-1} \tau \) satisfying \( t_n - s - T_{\ell-1} \geq 0 \) for all \( t_n - s \in [B^{\ell-1}\tau, (2B^{\ell-1} - 1)\tau] \), \( s \in [s_{\ell-1}, s_\ell] \). Here \( y(s) = y(s, s_\ell, \lambda_j^{(l)}) \) used in (8) that is defined by (9) satisfies the following ODE
\[
y'(s) = -\lambda_j^{(l)} y(s) + I_{s_\ell}^{(1)} u(s), \quad y(s_\ell) = 0,
\]
(10)

which can be exactly solved by the following recursive relation
\[
y(t_{m+1}) = e^{-\lambda_j^{(l)} \tau} y(t_m) + \int_{t_m}^{t_{m+1}} e^{-\lambda_j^{(l)} (s-t_m)} I_{s_\ell}^{(1)} u(s) \, ds.
\]
(11)

Step (4) Calculate the local part \( L_{\tau}^{(\alpha,n)} u = \int_{t_{n-\tau}}^{t_n} k_\alpha (t_n - s) I_{s_\ell}^{(1)} u(s) \, ds \)
\[
L_{\tau}^{(\alpha,n)} u = L(I_{s_\ell}^{(1)} u, t_n) = \frac{\tau^\alpha}{\Gamma(2+\alpha)} (u_n - u_{n-1}).
\]
(12)

Combining Steps (1)–(4), we obtain our fast convolution for approximating \( k_\alpha * u(t) \). The above fast convolution has the same storage and computational cost as that in [22], the main differences are listed below:

(i) The LG quadrature is applied instead of the trapezoidal rule to approximate the history part \( H_\alpha (I_{\tau} u, t) = \int_0^{t_\alpha} k_\alpha (t - s) I_{s_\ell} u(s) \, ds \), that is, the history part \( H_\alpha (I_{\tau} u, t) \) in [22] was approximated by
\[
f \hat{H}_\tau^{(\alpha,n)} u = \text{Im} \left\{ \sum_{\ell=1}^{L} \sum_{j=-N}^{N-1} \omega_j^{(l)} \mathcal{L}[k_\alpha] (\hat{\lambda}_j^{(l)}) e^{(t_n-s_{\ell-1})\hat{\lambda}_j^{(l)}} \hat{y}(s_{\ell-1}, s_\ell, \hat{\lambda}_j^{(l)}) \right\},
\]
(13)

where \( \{\omega_j^{(l)}\} \) and \( \{\hat{\lambda}_j^{(l)}\} \) are the weights and quadrature points for the Talbot contour \( \Gamma_\ell \), and \( \hat{y}(s) = \hat{y}(s, s_\ell, \hat{\lambda}_j^{(l)}) = \int_{s_\ell}^{s} e^{-(s-s_{\ell-1})\hat{\lambda}_j^{(l)}} I_{s_\ell}^{(1)} u(s) \, ds \) satisfies the following ODE
\[
\hat{y}'(s) = \hat{\lambda}_j^{(l)} \hat{y}(s) + I_{s_\ell}^{(1)} u(s), \quad \hat{y}(s_\ell) = 0.
\]
(14)

(ii) We solve a stable ODE (10) instead of a possibly unstable ODE (14) that may affect the stability and accuracy of (13). Indeed, for the Talbot contour used in [22] (see also the parabolic contour or hyperbolic contour discussed in [40]), there exist \( \hat{\lambda}_j^{(l)} \)'s, whose real parts are positive. Numerical tests show that (13) still works well since one may not solve (14) for a long time, which reduces the iteration error from solving (14) even though the real part of some \( \hat{\lambda}_j^{(l)} \) is positive.

The above fast convolution Step (1)–Step (4) holds for \( \alpha < 1 \), that is, the RL fractional derivative operator of order \( -\alpha \) is thus discretized if \( \alpha < 0 \).

Example 1 Let \( u(s) = 1 + s \) in (8) and define the relative error
\[
e_n = \frac{|H_\alpha (u(t_n)) - f \hat{H}_\tau^{(\alpha,n)} u|}{|H_\alpha (u(t_n))|}, \quad \hat{e}_n = \frac{|H_\alpha (u(t_n)) - f \hat{H}_\tau^{(\alpha,n)} u|}{|H_\alpha (u(t_n))|}
\]
(15)
We choose \( u(s) = 1 + s \) in order that the errors \( e_n \) and \( \hat{e}_n \) mainly come from the quadrature used in the discretization of the kernel \( k_\alpha(t) \). We show the errors \( e_n \) and \( \hat{e}_n \) for different \( \alpha \) in Fig. 1. We can see that the LG quadrature shows better accuracy than the trapezoidal rule based on the Talbot contour in this example.

### 3 Short Memory Principle with Lag Terms

In this section, we generalize the fast convolution in the previous section to resolve the short memory principle (see [6,31]). The error analysis of the method is given in the next section.

Let \( \Delta T \geq 0 \) be a memory length. Divide the convolution \( k_\alpha * u(t) \) into the local part \( L^\alpha_{\Delta T}(u, t) \) and the history part \( H^\alpha_{\Delta T}(u, t) \) as shown below

\[
L^\alpha_{\Delta T}(u, t) \equiv \int_{\max[0, t-\Delta T]}^{t} k_\alpha(t-s)u(s) \, ds,
\]

\[
H^\alpha_{\Delta T}(u, t) = \int_{0}^{\max[0, t-\Delta T]} k_\alpha(t-s)u(s) \, ds.
\]

If we drop the history part in (16), then the remaining part is the famous short memory principle rule (see [31]). However, the short memory principle has not been widely applied, since \( L^\alpha_{\Delta T}(u, t) \) is not a good approximation of \( k_\alpha * u(t) \). Our goal is to develop a good numerical approximation of \( H^\alpha_{\Delta T}(u, t) \), such that the storage and computational cost are reduced significantly compared with the direct approximation. The local part \( L^\alpha_{\Delta T}(u, t) \) is just the fractional operator defined on \( [\max[0, t-\Delta T], t] \), which can be discretized directly by the known methods, see [11,21,24,36]. Next, we introduce how to discretize (16) efficiently and accurately.

#### 3.1 Interpolations

In this work, the discretization of (17) and (18) is based on the interpolation of \( u \). Specifically, \( L^\alpha_{\Delta T}(u, t) \) and \( H^\alpha_{\Delta T}(u, t) \) are approximated by \( L^\alpha_{\Delta T}(I^L_T u, t) \) and \( H^\alpha_{\Delta T}(I^H_T u, t) \), respectively, where \( I^L_T \) and \( I^H_T \) are two suitable piecewise interpolation operators, which will be discussed in the following.

Linear interpolation is simple and has been applied widely in the discretization of the fractional operators, see [8,15,17,36]. Several quadratic interpolations have been used to discretize the Caputo fractional operators, see, for example, [19,24,42]. We adopt the quadratic interpolation in [24] to illustrate the implementation of the present fast algorithm. Define the local interpolation operator \( \Pi^L_t \) as

\[
\Pi^L_t u(t) = \sum_{k=1}^{3} u_{j+k-1} F_k^{(j)}(t), \quad t \in [t_j, t_{j+1}], \ j \geq 0,
\]

where \( F_1^{(j)} = \frac{(t-t_{j+1})(t-t_{j+2})}{(t_j-t_{j+1})(t_j-t_{j+2})} \), \( F_2^{(j)} = \frac{(t-t_j)(t-t_{j+2})}{(t_{j+1}-t_{j})(t_{j+1}-t_{j+2})} \), and \( F_3^{(j)} = \frac{(t-t_{j+1})(t-t_{j+2})}{(t_j-t_{j+2})(t_{j+2}-t_{j+1})} \).

Let \( \Pi^L_T^{-1} u(t) = \Pi^L_T^0 u(t) \). Then, for each \( n \geq 1 \), the quadratic interpolation \( I^L_T^{(2,n)} \) is defined by

\[
I^L_T^{(2,n)} u(t) = \begin{cases} \Pi^L_T u(t), & t \in [t_j, t_{j+1}], 0 \leq j \leq n-2, \\ \Pi^L_T^{n-2} u(t), & t \in [t_{n-1}, t_n]. \end{cases}
\]
For each $n \geq 1$, we define $I^L_t$ and $I^H_t$ as follows

$$I^L_t u(t) = I^{(2,n)}_t u(t), \quad t \in [\max\{t_n - \Delta T, 0\}, t_n],$$

$$I^H_t u(t) = I^{(2,n)}_t u(t), \quad t \in [0, \max\{t_n - \Delta T, 0\}].$$

Let $j_n = \max\{n - n_0, 0\}$. Then $L^{(\alpha,n)}_{\Delta T,t} u = L^\alpha_{\Delta T} (I^L_t u, t_n)$ and $H^{(\alpha,n)}_{\Delta T,t} u = H^\alpha_{\Delta T} (I^H_t u, t_n)$ are given by

$$L^{(\alpha,n)}_{\Delta T,t} u = \sum_{j=j_n}^{n-2} \left( b^{(1)}_{n-1-j} u_j + b^{(2)}_{n-1-j} u_{j+1} + b^{(3)}_{n-1-j} u_{j+2} \right) + \sum_{j=0}^{2} d_j u_{n+j-2},$$

$$H^{(\alpha,n)}_{\Delta T,t} u = \sum_{j=0}^{j_n-1} \left( b^{(1)}_{n-1-j} u_j + b^{(2)}_{n-1-j} u_{j+1} + b^{(3)}_{n-1-j} u_{j+2} \right),$$

where $b^{(k)}_{n-j-1} = \int_{t_j}^{t_{j+1}} k_{\alpha}(t_n - s) F_k^{(j)}(s) \, ds$ and $d_j = \int_{t_{n-1}}^{t_n} k_{\alpha}(t_n - s) F_j^{(n-2)}(s) \, ds$, i.e.,

$$d_0 = -\frac{\alpha \tau^\alpha}{2 \Gamma(\alpha + 3)}, \quad d_1 = \frac{\alpha(3 + \alpha) \tau^\alpha}{2 \Gamma(\alpha + 3)}, \quad d_2 = \frac{(4 + \alpha) \tau^\alpha}{2 \Gamma(\alpha + 3)},$$

$$b^{(1)}_j = \frac{\tau^\alpha}{2 \Gamma(\alpha)} \left[ \ell_j^{(\alpha+2)} - (2j-1) \ell_j^{(\alpha+1)} + j(j-1) \ell_j^{(\alpha)} \right],$$

$$b^{(2)}_j = -\frac{\tau^\alpha}{\Gamma(\alpha)} \left[ \ell_j^{(\alpha+2)} - 2j \ell_j^{(\alpha+1)} + (j+1)(j-1) \ell_j^{(\alpha)} \right],$$

$$b^{(3)}_j = \frac{\tau^\alpha}{2 \Gamma(\alpha)} \left[ \ell_j^{(\alpha+2)} - (2j+1) \ell_j^{(\alpha+1)} + (j+1)(j+1) \ell_j^{(\alpha)} \right],$$

$$\ell_j^{(\alpha)} = \frac{1}{\alpha} \left[ (j+1)^\alpha - j^\alpha \right].$$

Some researchers used other interpolations in the discretization of the fractional integral and derivative operators, we refer readers to [7, 8, 10, 17, 19, 44].

### 3.2 Corrections

From (23)–(24) and for any $\Delta T \geq 0$, we always have

$$D^{(\alpha,n)}_{\Delta T,t} u = L^{(\alpha,n)}_{\Delta T,t} u + H^{(\alpha,n)}_{\Delta T,t} u = \sum_{j=0}^{n} w_{n,j} u_j,$$

where $w_{n,j}$ can be derived from (25)–(28), which is not explicitly given here. Clearly, $D^{(\alpha,n)}_{\Delta T,t} u$ is just the second-order trapezoidal rule (or the $(2+\alpha)$-order L1 method) for the fractional integral of order $\alpha > 0$ (or the RL fractional derivative of order $0 < -\alpha < 1$) if the linear interpolation is used and $u(t)$ is sufficiently smooth, see [7, 8]. For quadratic interpolation (20), $D^{(\alpha,n)}_{\Delta T,t} u$ achieves the $(3+\alpha)$-order accuracy for the RL derivative of order $0 < -\alpha < 1$ when $u(t)$ is smooth. However, the approximation $D^{(\alpha,n)}_{\Delta T,t} u$ defined by (30) is not a good approximation of $k_{\alpha} u(t_n)$ when $u(t)$ has strong singularities.

In this work, we follow Lubich’s idea (see [21]) to use correction terms to capture the singularity of the solution $u(t)$ to the considered FODE. The correction method is based on
the assumption that the solution $u(t)$ has the following form

$$u(t) - u(0) = \sum_{j=1}^{m} c_j t^{\sigma_j} + t^{\sigma_{m+1}} \tilde{u}(t), \quad 0 < \sigma_j < \sigma_{j+1},$$

(31)

where $\tilde{u}(t)$ is uniformly bounded for $t \in [0, T]$. Readers can refer to [8,13,23,31] for more detailed results of the regularity of FODEs.

Combining (30) and (31) gives the following correction method

$$D^{(\alpha,n,m)}_{\Delta T, \tau} u = D^{(\alpha,n)}_{\Delta T, \tau} u + \tau^\alpha \sum_{j=1}^{m} W_{n,j} (u_j - u_0),$$

(32)

where $W_{n,j}$ are the starting weights that are chosen such that

$$D^{(\alpha,n)}_{\Delta T, \tau} u + \tau^\alpha \sum_{j=1}^{m} W_{n,j} (u_j - u_0) = k_\alpha \ast u(t_n) = \frac{\tau^\alpha n^\alpha + \alpha}{\Gamma(\sigma_k + 1 + \alpha)}, \quad 1 \leq k \leq m$$

(33)

for $u(t) = t^{\sigma_k}, 0 < \sigma_k < \sigma_{k+1}$. For each $n > 0$, one can resolve $W_{n,j} (1 \leq j \leq m)$ from the above linear system and $W_{n,j}$ are independent of $\tau$. The error analysis of the direct method (32) is discussed in Sect. 4. Readers can refer to [9,21,46] for more discussions of the correction method.

### 3.3 The Fast Implementation

In this subsection, we generalize the fast algorithm in Sect. 2 to approximate $D^{(\alpha,n)}_{\Delta T, \tau}$ defined by (30), which is given as follows.

- Step (A) Decompose $D^{(\alpha,n)}_{\Delta T, \tau} u$ into two parts as $D^{(\alpha,n)}_{\Delta T, \tau} u = L^{(\alpha,n)}_{\Delta T, \tau} u + H^{(\alpha,n)}_{\Delta T, \tau} u.$
- Step (B) Assume that $\Delta T = n_0 \tau = t_{n_0}$. For every $t = t_n, n \geq n_0$, let $L$ be the smallest integer satisfying $t_{n-n_0+1} < 2B^L \tau$. For $\ell = 1, 2, \ldots, L-1$, determine the integer $q_{\ell}$ such that

$$s_\ell = q_{\ell} B^\ell \tau \quad \text{satisfies} \quad t_{n-n_0+1} - s_\ell \in [B^\ell \tau, (2B^\ell - 1) \tau].$$

(34)

Set $s_0 = t_{n-n_0+1} - \tau$ and $s_L = 0$.
- Step (C) Let $\hat{t}_n = t_n - \Delta T + \tau = t_{n-n_0+1} \geq \tau$. Then the history part $H^{(\alpha,n)}_{\Delta T, \tau} u = H^{\alpha}_{\Delta T} (I^{\alpha}_t u, t_n)$ is approximated by

$$H^{(\alpha,n)}_{\Delta T, \tau} u = \frac{\sin(\alpha \pi)}{\pi} \sum_{\ell=1}^{L} \int_0^\infty \lambda^{-\alpha} e^{-(t_{\ell-1} + \Delta T - \tau) \lambda} e^{-\hat{t}_n s_{\ell-1} - \hat{t}_{\ell-1} s_{\ell-1} - \hat{t}_{\ell-1} \lambda} y(s_{\ell-1}, s_{\ell}, \lambda) d\lambda$$

$$\approx \frac{\sin(\alpha \pi)}{\pi} \sum_{\ell=0}^{L-1} \sum_{j=0}^{q_{\ell}(N_{\ell})} \omega_j^{(\ell)} e^{-\hat{t}_n s_{\ell-1} - \hat{t}_{\ell-1} s_{\ell-1} - \hat{t}_{\ell-1} \lambda_j^{(\ell)}} y(s_{\ell-1}, s_{\ell}, \lambda_j^{(\ell)}) := f H^{(\alpha,n)}_{\Delta T, \tau} u,$$

(35)

where $\{\omega_j^{(\ell)}\}$ and $\{\lambda_j^{(\ell)}\}$ are the LG quadrature weights and points that correspond to the weight function $\lambda^{-\alpha} e^{-(t_{\ell-1} + \Delta T - \tau) \lambda}$, $q_{\alpha}(N_{\ell})$ is defined by (45), $N_{\ell}$ is given by

$$N_{\ell} = \left\lceil \frac{\log \epsilon}{2 \log(\frac{T_{\ell}}{T_{\ell+1}})} \right\rceil, \quad T_{\ell} = \frac{2B - 1 - B^1 - \ell}{1 + B^1 - \ell (\Delta T / \tau - 1)}, \quad \epsilon > 0,$$

(36)
and \( y(s_{\ell -1}, s_{\ell}, \lambda_j^{(\ell)}) \) can be obtained exactly by solving the following linear ODE

\[
y'(s) = -\lambda_j^{(\ell)} y(s) + I_t^H u(s), \quad y(s_l) = 0,
\]

see also (10) and (11).

- Step (D) Calculate the local part \( L_{\Delta T, \tau}^{(\alpha, n)} u = L_{\Delta T}^{\alpha} (I_L^u, t_n) \) directly.

The fast algorithm for the discretization of \( k_\alpha * u(t) \) is now given by

\[
f D_{\Delta T, \tau}^{(\alpha, n, m)} u = L_{\Delta T, \tau}^{(\alpha, n)} u + f H_{\Delta T, \tau}^{(\alpha, n)} u + \tau^\alpha \sum_{j=1}^m W_{n,j}(u_j - u_0),
\]

where \( L_{\Delta T, \tau}^{(\alpha, n)} \) is given by (23), \( f H_{\Delta T, \tau}^{(\alpha, n)} u \) is given by (35), and the starting weights \( W_{n,j} \) are determined by the linear system (33).

Next, we analyze the complexity of the present fast method (38). For the local part \( L_{\Delta T, \tau}^{(\alpha, n)} u \), the memory requirement is \( O(n_0) \) with the computational cost of \( O(n_0 (n_T - n_0)) \) for all \( n_0 < n \leq n_T \). For the history part \( f H_{\Delta T, \tau}^{(\alpha, n)} u \), we have \( O(\sum_{\ell} q_\alpha(N_\ell)) \) active memory and \( O((n_T - n_0) \sum_{\ell} q_\alpha(N_\ell)) \) operations, where \( L = \log_B (n_T - n_0) \). An additional cost is required to obtain the starting weights \( W_{n,j} \) in (38), which can be performed by use of fast Fourier transform with \( O(n_T \log(n_T)) \) arithmetic operations. Hence, the overall active memory and computational cost are \( O(n_0 + \sum_{\ell} q_\alpha(N_\ell)) \) and \( O(n_0 n_T + (n_T - n_0) \sum_{\ell} q_\alpha(N_\ell)) \), respectively.

### 4 Error Analysis

In this section, we analyse the overall discretization error of the fast method (38) in Sect. 3. Firstly, we present the exponential convergence rate of the LG quadrature used in (8) and (35). Then we show how to choose \( N_\ell \) and \( q_\alpha(N_\ell) \) used in (35), such that the desired accuracy is maintained with the use of the minimum number of the quadrature points.

For simplicity, we denote

\[
J^\alpha[T, f] = \int_0^\infty \lambda^\alpha e^{-T\lambda} f(\lambda) \, d\lambda \quad \text{and} \quad J^\alpha[f] = \int_0^\infty \lambda^\alpha e^{-\lambda} f(\lambda) \, d\lambda.
\]

The LG quadrature for \( J^\alpha[T, f] \) and \( J^\alpha[f] \) are given by (see [34])

\[
Q^\alpha_N[T, f] = T^{-\alpha - 1} \sum_{j=0}^N w_j^{(\alpha)} f(\lambda_j / T), \quad Q^\alpha_N[f] = \sum_{j=0}^N w_j^{(\alpha)} f(\lambda_j),
\]

where \( \lambda_j \) are the roots of the Laguerre polynomial \( L_{N+1}^{(\alpha)}(\lambda) \), and \( w_j^{(\alpha)} \) are the corresponding weights given by

\[
w_j^{(\alpha)} = \frac{\Gamma(N + \alpha + 1) \lambda_j}{(N + \alpha + 1)(N + 1)!} \left( L_N^{(\alpha)}(\lambda_j) \right)^{-2}.
\]

We show the convergence of the quadrature \( Q^\alpha_N[T, e^{-i\lambda}] \) and the property of the quadrature weight \( w_j^{(\alpha)} \) [see (41)] in the following two theorems. The proofs are given in “Appendix A.”

\( \text{Springer} \)
Theorem 1 Let \( t \geq 0, T > 0, \alpha > -1, \) and \( N \) be sufficiently large. Then

\[
\left| J^\alpha[T, e^{-t\lambda}] - Q^\alpha_N[T, e^{-t\lambda}] \right| \leq \begin{cases} 
CT^{-\alpha-1} \left( \frac{t/T}{1+t/T} \right)^{2N}, & -1 < \alpha \leq 0, \\
CT^{-\alpha-1}N^\alpha \left( \frac{t/T}{1+t/T} \right)^{2N}, & \alpha > 0,
\end{cases}
\] (42)

where \( C \) is a constant independent of \( N \).

Theorem 2 Let \( \alpha > -1 \) and \( w^{(\alpha)}_j \) be defined by (41). If \( N \) and \( j \) are sufficiently large, then there exists a positive constant \( C \) independent of \( N \) such that

\[
w^{(\alpha)}_j \leq C(N+1)^\alpha e^{-0.25\pi^2(j+1)^2/(N+1)}.
\] (43)

In real applications, the LG quadrature can be truncated such that high accuracy can be maintained and the computational cost can be reduced. Given a precision \( \epsilon_0 > 0 \), the truncated LG quadrature is given by

\[
Q^\alpha_{N,\epsilon_0}[T, f] = T^{-\alpha-1} \sum_{j=0}^{q_0(N)} w^{(\alpha)}_j f(\lambda_j/T),
\] (44)

where \( q_0(N) \) is a positive number given by

\[
q_0(N) = \min \left\{ N, \left\lceil 2\pi^{-1} \sqrt{(N+1) \log((N+1)^\alpha \epsilon_0^{-1})} \right\rceil - 1 \right\}.
\] (45)

From (43), we can find the smallest integer \( j \) satisfying \((N+1)^\alpha e^{-0.25\pi^2(j+1)^2/(N+1)} \leq \epsilon_0\), which yields (45). We choose \( \epsilon_0 = 10^{-16} \) in this paper if \( \epsilon_0 \) is not set to be other values.

The following theorem states an error bound of \( f H^{(\alpha,n)}_{\Delta T,\tau} \), the proof of which is given in “Appendix A”.

Theorem 3 Let \( \Delta T = n_0\tau, n_0 \in \mathbb{N}, \alpha > -1. \) Let \( f H^{(\alpha,n)}_{\Delta T,\tau} \) and \( H^{(\alpha,n)}_{\Delta T,\tau} \) be defined by (35) and (24), respectively. For any precisions \( \epsilon, \epsilon_0 > 0 \) and \( t_n > \Delta T \), there exists a positive constant \( C \) independent of \( n \) and \( \tau \), such that

\[
\left| H^{(\alpha,n)}_{\Delta T,\tau} u - f H^{(\alpha,n)}_{\Delta T,\tau} u \right| \leq C \max\{0, t_{n+1} - \Delta T\} ||u||_\infty (\epsilon + \epsilon_0),
\] (46)

where \( ||u||_\infty = \max_{n=0}^{T/\tau} |u_n| \).

Figure 2a shows the exponential decay of \( w^{(-\alpha)}_j \) when \( j \geq q_{\alpha}(128) \) for different fractional orders \( \alpha = -1.8, -1.2, -0.8, -0.2, 0.2, 0.8 \). Figure 2b displays similar behaviors as shown in Fig. 2a. We can see that Eq. (45) works well, i.e., \( w^{(-\alpha)}_j < C \epsilon_0 \) when \( j \geq q_0(N) \), and \( q_0(N) \) is not very sensitive to the fractional order \( \alpha \in (-2, 1) \). For example, \( q_0(128) = (48, 47, 46, 44, 43, 41) \) (or \( q_0(256) = (69, 67, 65, 62, 61, 58) \)) for \( \alpha = (-1.8, -1.2, -0.8, -0.2, 0.2, 0.8) \), and \( q_0(N) \leq N \) when \( N \) is sufficiently large. For \( \alpha \leq -2 \), similar results are obtained, which is not shown here.

Figure 2 also verifies that the truncated LG quadrature (44) can preserve accuracy up to \( \epsilon_0 \), since \( f(\lambda) \) in (44) takes the form \( e^{-\hat{H}^n_{\ell}} \lambda \) in this paper, where \( \hat{H}^n_{\ell} \geq 0 \) is given in the proof of Theorem 3 in “Appendix A”. Hence, we have

\[
Q^\alpha_N[T, f] - Q^\alpha_{N,\epsilon_0}[T, f] = T^{-\alpha-1} \sum_{j=q_0(N)+1}^{N} w^{(\alpha)}_j f(\lambda_j/T) \leq T^{-\alpha-1} \sum_{j=q_0(N)+1}^{N} w^{(\alpha)}_j.
\] (47)
The exponential decay of \( w_j(\alpha) \) implies that the right-hand-side of the above equation is very small, i.e., \( \sum_{j=q(\alpha)}^N w_j(\alpha) = O(w_j(\alpha)) = O(\epsilon_0) \). In [25], \( q(\alpha) \) is chosen as \( q(\alpha) = \max\{ j : \lambda_j < 4\theta N, 0 < \theta < 1 \} \), where \( \theta \) was chosen as 1/4, 1/8, 1/16 and 1/32 in the numerical simulations and good results were obtained. In this work, we find that \( q(\alpha) \) given by (45) works very well for the truncated LG quadrature (44) and is equivalent to the practical option given in Remark 2.1 of [26]. We investigate the effectiveness of the truncated LG quadrature further throughout this section.

Next, we present the overall error bound of (49). Denote by
\[
R^{(n)} = (k_\alpha * u)(t_n) - D^{(\alpha,n,m)}_{\Delta T,T} u. \tag{48}
\]
Note that the above discretization error \( R^{(n)} \) depends on the smoothness of \( u(t) \) and the discretization method \( D^{(\alpha,n,m)}_{\Delta T,T} \). If \( u(t) \) is sufficiently smooth, no correction terms are needed to achieve \( (2 + \alpha) \)-order (or \( (3 + \alpha) \)-order) accuracy if linear (or quadratic) interpolation is applied for \( \alpha < 0 \). If \( u(t) \) satisfies (31), then the global \( (2 + \alpha) \)-order (or \( (3 + \alpha) \)-order) accuracy can be achieved for \( \sigma_{m+1} \geq 2 \) (or \( \sigma_{m+1} \geq 3 \)). In numerical simulations, the condition \( \sigma_{m+1} \geq 2 \) (or \( \sigma_{m+1} \geq 3 \)) does not need to be satisfied, \( (2 + \alpha) \)-order (or \( (3 + \alpha) \)-order) accurate numerical solutions are observed far from \( t = 0 \). In the numerical simulations, only a small number of correction terms are sufficient to achieve very accurate numerical solutions; see numerical simulations in the following section and see also related results in [46].

Combining (46) and (48) yields
\[
|k_\alpha * u(t_n) - f D^{(\alpha,n,m)}_{\Delta T,T} u| = |H^{\alpha}_{\Delta T}( I^{H}_{\tau} u, t_n) - f H^{(\alpha,n)}_{\Delta T,T} u + R^{(n)} | \\
\leq C \max\{0, t_{n+1} - \Delta T \} \| u \|_{\infty} (\epsilon + \epsilon_0) + |R^{(n)}|, \tag{49}
\]
where the error in (49) originates from two parts: the LG quadrature for discretizing \( J^{-\alpha} [\hat{T}_\tau, e^{-\hat{T}^{(s)}_\tau}] \) [see (61)] and the discretization error defined by (48).

Next, we numerically study the error (49) caused by the LG quadrature.

**Example 2** Let \( m = 0 \) and \( u(t) = 1 + t \). Then \( R^{(n)} \) in (49) is zero. Denote the relative pointwise error
\[
e_n = (k_\alpha * u(t_n))^{-1} |k_\alpha * u(t_n) - f D^{(\alpha,n,0)}_{\Delta T,T} u|, \quad 1 \leq n \leq n_T = T/\tau.
\]

Fig. 2 The exponential decay of the quadrature weights \( w_j(-\alpha) \) defined by (41). a \( N = 128 \), b \( N = 256 \).
where $k_{\alpha} * u(t_n) = t_n^\alpha / \Gamma(1 + \alpha) + t_n^{\alpha+1} / \Gamma(2 + \alpha)$ for $u(t) = 1 + t$.

Given a precision $\epsilon = 10^{-10}$, the maximum relative error $\|e\|_\infty = \max_{0 \leq n \leq n_T} |e_n|$, the total number of the quadrature points $\sum N_\ell$, and the total number of the truncated quadrature points $\sum q_\alpha(N_\ell)$ are shown in Table 1 for different basis $B$, $\alpha = -0.5$, $\tau = 0.1$, $\Delta T = 1$, and $T = 10^4$. We can see that the truncated LG quadrature saves memory. A relatively smaller basis $B$ needs less quadrature points and thus saves memory, which can be explained from (59) and (60). Equation (60) implies a faster convergence as $B$ decreases, due to $T_\ell \approx 2B - 1$, $\ell$ is sufficiently large. Hence, a relatively smaller $B$ means that smaller $N_\ell$ are needed to achieve high accuracy that leads to the use of less LG quadrature points.

We change the precision $\epsilon$ and show the corresponding relative maximum errors $\|e\|_\infty$ in Table 2. We can see that $\|e\|_\infty$ increases as $\epsilon$ increases and the total number of the quadrature points are reduced. It also shows that much better results are obtained than the theoretical prediction, see also the related results in Table 10.

Figure 3a shows the relative maximum error $\|e\|_\infty$ against the basis $B$ for different fractional orders, $\tau = 0.01$, $T = 10^4$, $\epsilon = 10^{-10}$. We can see that better results are obtained than the predicted precision $\epsilon = 10^{-10}$, even for $\alpha = -1.8$ (the fractional derivative of order 1.8). Figure 3b shows the total number of the truncated LG quadrature points against the basis $B$. For a fixed $B$, the number of truncated quadrature points increases as $\alpha$ decreases, which agrees with the theoretical prediction (45). For a fixed fractional order $\alpha$, the number of the truncated quadrature points increases as $B$ increases, which agrees with (36), due to $T_\ell \approx 2B - 1$ when $\ell$ is sufficiently large.

It is reasonable to choose a relatively smaller precision $\epsilon$ and smaller basis $B$ in numerical simulations, since a smaller $\epsilon$ ensures a relatively large $N_\ell$ that guarantees the exponential convergence of the LG quadrature, and a small $B$ ensures a smaller radius of convergence of the quadrature error (60).

Table 3 shows the relative maximum error $\|e\|_\infty$ for different $\Delta T$ under the precision $\epsilon = \epsilon_0 = 10^{-10}$. We see that the numerical results exhibit a slightly better performance than the theoretical prediction and are not sensitive to the stepsize $\tau$ and $\Delta T$. Table 4 shows the relative maximum error $\|e\|_\infty$ for different precision $\epsilon = \epsilon_0$. Obviously, for a fixed fractional order $\alpha$, the maximum error $\|e\|_\infty$ increases as the precision decreases. For a fixed precision $\epsilon = \epsilon_0$, numerical results are not very sensitive to the fractional order, which may be explained
The relative pointwise errors we compare the accuracy of the sum-of-exponentials from the LG quadrature and the multipole approximation and the truncated LG quadrature, see Fig. 4a. Figure 4b shows the $\epsilon$ achieve similar accuracy with, respectively, $\epsilon = \epsilon_0 = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$ from the use of the LG quadrature where the singularity is treated exactly. Table 5 shows the maximum relative error $\|e\|_\infty$ using different number of truncation quadrature points, $\Delta T = 0.5, \tau = 0.01, T = 1000, \alpha = -0.5, B = 5, \epsilon = 10^{-10}$.
The multi-pole approximation: $P = 832$

The truncated LG quadrature: $Q = 777$

The multi-pole approximation: $P = 832$

The truncated LG quadrature: $Q = 818$

(a) $\alpha = 0.5$, $\tau = 0.0001$

(b) $\alpha = -0.5$, $\tau = 0.0001$

**Fig. 4** Comparison of the truncated LG quadrature and the multipole approximation [2]. a $\alpha = 0.5$, $\tau = 0.0001$, b $\alpha = -0.5$, $\tau = 0.0001$

For other fractional orders $\alpha \in (-1, 1)$, the truncated LG quadrature is competitive with the multipole approximation both in accuracy and the computational cost. These results are not shown here.

## 5 Numerical Examples

In this section, two examples are presented to verify the effectiveness of the present fast method when it is applied to solve FDEs. All the algorithms are implemented using MATLAB 2016a, which were run in a 3.40 GHz PC having 16GB RAM and Windows 7 operating system.

**Example 3** Consider the following scalar FDE

$$
C_{D_0,t}^\alpha u(t) = -Au(t) + F(u, t), \quad u(0) = u_0, \quad t \in (0, T],
$$

(50)

where $0 < \alpha \leq 1$ and $A \geq 0$, and $C_{D_0,t}^\alpha$ is the Caputo fractional operator, which satisfies $C_{D_0,t}^\alpha u(t) = k_\alpha * u(t) - u(0)t^{-\alpha}/\Gamma(1 - \alpha)$, see [31].

We present our fast numerical method for (50) as follows: For a given length $\Delta T = n_0 \tau$, find $U_n$ for $n > n_0$ such that

$$
F_D^{(-\alpha,n,m)} U - u_0 t_n^{-\alpha}/\Gamma(1 - \alpha) = -AU_n + F(U_n, t_n),
$$

(51)

where $F_D^{(-\alpha,n,m)}$ is defined by (38). The application of the direct convolution method means here that $F_D^{(-\alpha,n,m)} U$ in (51) is replaced by $D_{\Delta T, \tau}^{(-\alpha,n,m)} U$. The Newton method is used to solve the nonlinear system (51). The direct method is applied to obtain $U_k (1 \leq k \leq n_0)$, and the starting values $U_k (1 \leq k \leq m)$ are obtained using the direct method with one correction term and a smaller stepsize $\tau^{-2}$.

The following two cases are considered in this example.

Case I: For the linear case of $F = 0$, the exact solution of (50) is

$$
u(t) = E_\alpha(-At^\alpha),$$

where $E_\alpha(t)$ is the Mittag–Leffler function defined by $E_\alpha(t) = \sum_{k=0}^{\infty} \frac{t^k}{\Gamma(k\alpha+1)}$. 

Case II: Let \( F = u(1 - u^2) \) and the initial condition is taken as \( u_0 = 1 \).

The maximum error is defined by

\[
\|e\|_\infty = \max_{0 \leq n \leq T/\tau} |e_n|, \quad e_n = u(t_n) - U_n, \quad T = 40.
\]

In this example, we always choose the memory length \( \Delta T = 0.5 \), the basis \( B = 5 \), the truncation number \( q_\alpha(N_\ell) \) in (35) is calculated by (45) and \( N_\ell \) is determined by (36) under the precision \( \epsilon = 10^{-10} \), and \( A = 1 \). We will reset these parameters if needed. The fast method based on quadratic interpolation (20) is applied in this example if there is no further illustration.

We give a simple illustration of how to choose the correction indices \( \sigma_k \).

(i) If we know the explicit form of the solution, then \( \sigma_k \) can be chosen according to the regularity of the solution. For Case I, we have \( u(t) = E_\alpha(-At^\alpha) = 1 + \sum_{k=1}^{\infty} c_k t^{k\alpha}, \) so that \( \sigma_k = k\alpha \); see results shown in Tables 6, 7, 8 and 9.

(ii) For Case II, we do not know the solution. However, we do know the regularity of the solution (see Corollary 6.37 in [8]), and we have \( \sigma_k \in \{i + j\alpha, i = 0, 1, \ldots; j = 0, 1, \ldots; ij \neq 0\}; \) see results shown in Fig. 6a.

(iii) If we do not know anything about the regularity of the solution, we can set \( \sigma_k = 0.1k \) in the computation. This option will improve the accuracy of the numerical solutions even if the solution has a strong singularity and would not harm the accuracy if the solution has high regularity; see Fig. 5.

The purpose of Case I is to check the effectiveness of the present fast method for non-smooth solutions. We demonstrate that adding correction terms improves the accuracy of the numerical solutions significantly. We first let \( \alpha = 0.8 \), the maximum error and the error at \( t = 40 \) are shown in Tables 6 and 7, respectively. We can see that the expected convergence rate \( 3 - \alpha \) is almost achieved by adding two or three correction terms, and the numerical solution at the final time is much more accurate than that near the origin. This phenomenon can be observed from the existing time-stepping methods for time-fractional differential equations.

As the fractional order decreases, the singularity of the analytical solution becomes stronger. Tables 8 and 9, respectively, show the maximum errors and the errors at \( t = 40 \) for \( \alpha = 0.1 \). We observe that the maximum error, which occurs near the origin, almost does not improve, though the step size decreases. We find that as the number of correction terms increases, the accuracy of the numerical solutions increases significantly. We need almost 30 correction terms to derive the expected global convergence rate \( 3 - \alpha \) for \( \alpha = 0.1 \), which is impossible to achieve by double precision, since one needs to solve an ill-conditioned linear system (33) to derive the starting weights. However, a small number of correction terms

| \( \tau \) | \( m = 0 \) | \( \text{Order} \) | \( m = 1 \) | \( \text{Order} \) | \( m = 2 \) | \( \text{Order} \) | \( m = 3 \) | \( \text{Order} \) |
|---|---|---|---|---|---|---|---|---|
| \( 2^{-5} \) | 2.7966e−3 | 1.5674e−3 | 3.0233e−5 | 3.4818e−5 | 0.6726 | 5.4427e−4 | 2.3568 | 7.7701e−6 | 2.1638 |
| \( 2^{-6} \) | 1.7545e−3 | 0.6726 | 5.4427e−4 | 1.5260 | 5.9024e−6 | 2.3568 | 7.7701e−6 | 2.1638 |
| \( 2^{-7} \) | 1.0604e−3 | 0.7264 | 1.8498e−4 | 1.5569 | 1.6683e−6 | 1.8229 | 1.7199e−6 | 2.1756 |
| \( 2^{-8} \) | 6.2689e−4 | 0.7584 | 6.2086e−5 | 1.5751 | 4.2485e−7 | 1.9733 | 3.7854e−7 | 2.1838 |
| \( 2^{-9} \) | 3.6575e−4 | 0.7773 | 2.0686e−5 | 1.5856 | 1.0205e−7 | 2.0576 | 8.3007e−8 | 2.1892 |
is enough to yield satisfactory numerical solutions, which makes the present method more practical. We refer readers to [46] for more explanations and numerical results associated with this phenomenon.

In Fig. 5, we show the pointwise errors for Case I, where the correction indices $\sigma_k = k/10$ do not completely match the singularity indices $k\alpha$ of the solution when $\alpha = 0.8$ and 0.15. We observe that the analytical solution has relatively good regularity for $\alpha = 0.8$ and the numerical solutions even become better with $m \geq 2$ correction terms, see Fig. 5a. We can see that the option of $\sigma_k = k/10$ works effectively when the solution has relatively low regularity, see Fig. 5b. This phenomenon has been investigated in [46] and a theoretical explanation was also provided. We refer the interested reader to [46] for more details.

Table 10 shows the difference $\eta = \max_{0 \leq n \leq T/\tau} |U_D^n - U_F^n|$, where $U_D^n$ are numerical solutions derived from the fast method under the precision $\epsilon$, and $U_F^n$ are numerical solutions from the direct method. We can see that the difference $\eta$ is independent of the stepsize $\tau$, which confirms (62). Table 10 also shows that the accuracy of the present fast calculation outperforms the predicted accuracy $\epsilon$, see (62).
Fig. 5 Pointwise errors for Case II, where the correction indices $\sigma_k$ do not match the singularity of the solution, $\tau = 1/256$, $\sigma_k = k/10$. a $\alpha = 0.8$, b $\alpha = 0.15$

Table 10 The difference of the numerical solutions between the fast method and the direct method, Case I, $m = 0$, $\alpha = 0.1$, $T = 40$, $B = 5$

| $\epsilon$ | $\tau = 2^{-5}$ | $\tau = 2^{-6}$ | $\tau = 2^{-7}$ | $\tau = 2^{-8}$ | $\tau = 2^{-9}$ |
|------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| $10^{-12}$  | 2.8255e-13      | 2.7850e-13      | 2.7167e-13      | 1.1102e-15      | 1.4988e-15      |
| $10^{-10}$  | 2.8239e-13      | 2.7839e-13      | 2.7162e-13      | 5.6066e-15      | 2.1094e-15      |
| $10^{-8}$   | 2.8116e-13      | 3.8219e-13      | 6.6397e-13      | 2.2093e-14      | 8.2682e-13      |
| $10^{-6}$   | 7.5677e-11      | 7.2366e-11      | 8.0116e-10      | 2.6198e-11      | 8.0352e-12      |
| $10^{-5}$   | 3.2916e-11      | 1.9709e-10      | 1.2477e-10      | 2.6461e-10      | 2.2769e-10      |
| $10^{-4}$   | 3.0868e-09      | 9.4674e-09      | 2.4896e-09      | 1.5458e-08      | 4.3178e-08      |

In Table 11, we compare the present fast method (51) based on the linear interpolation with the graded mesh method in [35], in which the nonuniform grid points are given by $t_j = (j \tau)^\gamma$, $0 \leq j \leq 1/\tau$. We can see that the fast method with correction terms is competitive with the graded mesh method for $\alpha = 0.5$. In [35], the authors obtained the optimal grid mesh $t_j = (j \tau)^{(2-\alpha)/\alpha}$ to achieve the global $(2-\alpha)$-order accuracy, which works well when $\alpha$ is relatively large and the expected convergence rate is achieved. For the correction method, too many correction terms may harm the accuracy, but a few number of correction terms can achieve highly accurate numerical solutions, which is not investigated here, readers can refer to [9, 21, 46] for more discussion. Note that for values of $\alpha$ close to zero, the correction method is even more effective than the optimal graded mesh approach in [35].

Next, we implement the present fast method for a longer time computation and compare it with the direct convolution method. We show in Fig. 6a numerical solutions for $t \in [0, T]$, $T = 10,000$, where we choose $\alpha = 0.1, 0.5, 0.9$, and the time stepsize $\tau = 0.01$. In Fig. 6b, we plot the computational time of the fast method and the direct method with two correction terms applied. It shows that the computational cost of the fast convolution almost achieves linear complexity, which is much less than that of the direct convolution with $O(n_T^2)$ operations when $n_T$ is sufficiently large, where $n_0 = \Delta T/\tau$ and $n_T = T/\tau$.

Next, we consider a system of FODEs each having a different fractional index.
Table 11 Comparison of the present method with the graded mesh method in [35] based on linear interpolation, Case I, $\sigma_k = k\alpha$, $\alpha = 0.5$, $B = 5$. The errors $\|e\|_\infty$ are displayed.

| $\tau$ | $r = 1$ | $r = 3/2$ | $r = 3$ | $r = 6$ |
|--------|---------|---------|---------|---------|
| $2^{-5}$ | $3.6491e-2$ | $1.6839e-2$ | $2.6568e-3$ | $2.5431e-3$ |
| $2^{-6}$ | $2.7048e-2$ | $0.4320$ | $0.7108$ | $1.3986$ | $9.2945e-4$ | $1.4521$ |
| $2^{-7}$ | $1.9772e-2$ | $6.2162e-3$ | $3.7277e-4$ | $1.4347$ | $3.3644e-4$ | $1.4660$ |
| $2^{-8}$ | $1.4312e-2$ | $3.7315e-3$ | $1.3582e-4$ | $1.4566$ | $1.2096e-4$ | $1.4759$ |
| $2^{-9}$ | $1.0288e-2$ | $2.2313e-3$ | $4.9041e-5$ | $1.4696$ | $3.3644e-4$ | $1.4827$ |

The fast method based on linear interpolation with correction terms

| $\tau$ | $m = 1$ | $m = 2$ | $m = 3$ | $m = 4$ |
|--------|---------|---------|---------|---------|
| $2^{-5}$ | $5.6366e-3$ | $2.1893e-4$ | $1.4976e-4$ | $7.2351e-5$ |
| $2^{-6}$ | $3.1659e-3$ | $0.8322$ | $1.0737e-4$ | $1.1156$ | $4.4890e-5$ | $0.6886$ |
| $2^{-7}$ | $1.7231e-3$ | $0.8777$ | $4.9198e-5$ | $1.1260$ | $2.9276e-5$ | $0.9898$ |
| $2^{-8}$ | $9.1600e-4$ | $0.9116$ | $2.1427e-5$ | $1.1991$ | $1.0105e-5$ | $1.1614$ |
| $2^{-9}$ | $4.7862e-4$ | $0.9364$ | $8.9763e-6$ | $1.2552$ | $4.5167e-6$ | $1.2696$ |

Fig. 6 Numerical solutions and the computational time of the two different convolutions for Case II, $\tau = 0.01$, $B = 5$, $m = 2$, $\sigma_k = k\alpha$. a Numerical solutions, b computational time

Example 4 Consider the following system of fractional differential equations each with different fractional index

\[
\begin{align*}
 cD_{0,t}^{\alpha_1}u(t) &= w + (v - c_1)u, \\
 cD_{0,t}^{\alpha_2}v(t) &= 1 - c_2v - u^2, \\
 cD_{0,t}^{\alpha_3}w(t) &= -u - c_3w,
\end{align*}
\]

(52)

where $0 < \alpha_k \leq 1$ ($k = 1, 2, 3$), $c_1$, $c_2$ and $c_3$ are positive constants, $c_2 > 1/2$.

Let $U_n$, $V_n$ and $W_n$ be the approximate solutions of $u(t_n)$, $v(t_n)$ and $w(t_n)$, respectively. Then the fully discrete scheme for the system (52) is given by:
Numerical solutions for Example 4, Fig. 7. We can easily find that \( \alpha \) need to be known, which can be derived using the known methods with smaller stepsize. Here we use the L1 method with one correction term and smaller stepsize to derive these values. We take \( \Delta T = \tau \) and \( B = 5 \) in numerical simulations of this example.

If \( \alpha_1 = \alpha_2 = \alpha_3 = \alpha \), then (52) is just the fractional Lorenz system [38]. It has been proved that the fractional Lorenz system (52) is dissipative [38] and has an absorbing set defined by a ball \( B(0, \sqrt{a/b} + \delta) \), where \( a = 1/2 \) and \( b = \min[c_1, c_2 - 1/2, c_3] \). We take \( c_1 = 1/4, c_2 = 1, c_3 = 1/4 \) and the initial conditions as those in [38], i.e., \( U_0 = u(0) = 2, V_0 = v(0) = 0.9, W_0 = w(0) = 0.2 \), the time stepsize is taken as \( \tau = 0.01 \). We compute numerical solutions for \( t \in [0, 1000] \), which is much larger than that in [38]. Numerical solutions for \( \alpha = 0.9 \) are shown in Fig. 7. We can easily find that \( U_n^2 + V_n^2 + W_n^2 < 2 \), which means \( (U_n, V_n, W_n) \in B(0, \sqrt{2}) \). For other fractional orders \( \alpha_k = \alpha \), we have similar results, see also [38]. Next, we choose different \( \alpha_1, \alpha_2, \) and \( \alpha_3 \), and exhibit the numerical solutions in Fig. 8 for \( (\alpha_1, \alpha_2, \alpha_3) = (0.9, 0.8, 0.7) \) and \( (\alpha_1, \alpha_2, \alpha_3) = (0.7, 0.8, 0.9) \). We
can see that the numerical solutions \((U_n, V_n, W_n)\) are also in a ball. For other choices of the fractional orders, we have similar results, which are not provided here.

6 Conclusion and Discussion

We propose a unified fast memory-saving time-stepping method for both fractional integral and derivative operators, which is applied to solve FDEs. We generalize the fast convolution in [22] for fractional operators, while we use the (truncated) LG quadrature to discretize the kernel in the fractional operators instead of the trapezoidal rule in [22]. We also introduced correction terms in the fast method such that the non-smooth solutions of the considered FDEs can be resolved accurately. The present fast method has \(O(n_0 + \sum L q(\alpha(N_\ell)))\) active memory and \(O(n_0 n_T + (n_T - n_0) \sum L q(\alpha(N_\ell)))\) operations. If a suitable memory length \(\Delta T\) and a very large basis \(B\) are applied, then the present fast method is similar to that in [14,18], that is, the kernel \(k_\alpha(t)\) is approximated via the truncated LG quadrature for all \(t \in [\Delta T, T]\).

We present the error analysis of the present fast method. For a given precision, a criteria on how to choose the parameters used in the fast method is given explicitly, which works very well in numerical simulations.

We considered the fast method for the fractional operator of order \(\alpha < 1\). In fact, our method can be extended to the fractional integral of order greater than one. For example, for \(\alpha \in (1, 2)\), Eq. (4) still holds in the sense of the finite part integral, that is, Eq. (4) is equivalent to \(k_\alpha(t) = \frac{1}{\Gamma(\alpha)\Gamma(2-\alpha)} \int_0^\infty \lambda^{1-\alpha} e^{-t\lambda} d\lambda\). In such a case, a coupled ODEs are needed to be resolved instead of one for \(\alpha < 1\).

In the future, we will apply the fast method to solve a large system of time-fractional PDEs, perhaps in three spatial dimensions. We will also explore how to efficiently calculate the discrete convolution \(\sum_{k=0}^n \omega_{n-k} u(t_k)\), where the quadrature weights \(\{\omega_k\}\) are not from the interpolation as done in the present work, but from the generating functions, see, [21]. The code for numerical simulations in this paper can be found at https://github.com/fanhaizeng/Fast-method-for-fractional-operators.

Acknowledgements The authors are very grateful to the anonymous referees for the careful reading of a preliminary version of the manuscript and their valuable suggestions and comments, which greatly improve the quality of this paper.
A Proofs

Proof of Theorem 1.

Proof We follow the proof of Theorem 2.2 in [41]. We first expand \( g(\lambda) = e^{-t\lambda} \) in terms of the Laguerre polynomials, i.e.,
\[
g(\lambda) = \sum_{n=0}^{\infty} a_n L_n^{(\alpha)}(\lambda),
\]
where
\[
a_n = \frac{\Gamma(n+1)}{\Gamma(n+1+\alpha)} \int_0^{\infty} \lambda^n e^{-\lambda} g(\lambda) L_n^{(\alpha)}(\lambda) \, d\lambda = \frac{t^n}{(t+1)^{n+1+\alpha}}.
\]

The following property will be used, see [41],
\[
|Q_N^\alpha[L_n^{(\alpha)}]| \leq \begin{cases} 
2\Gamma(1+\alpha), & -1 < \alpha \leq 0, \\
2^{1+\alpha} \frac{\Gamma(n+1+\alpha)}{\Gamma(n+1)}, & \alpha > 0.
\end{cases}
\]

With the above two equations and \( J^{\alpha}[e^{-t\lambda}] - Q_N^\alpha[e^{-t\lambda}] = \sum_{n=2N}^{\infty} a_n Q_N^\alpha[L_n^{(\alpha)}] \) gives
\[
|J^{\alpha}[e^{-t\lambda}] - Q_N^\alpha[e^{-t\lambda}]| \leq \begin{cases} 
2^{1+\alpha} \frac{\Gamma(n+1+\alpha)}{\Gamma(n+1)} \sum_{n=2N}^{\infty} a_n, & -1 < \alpha \leq 0, \\
c_2^{1+\alpha} \sum_{n=2N}^{\infty} n^\alpha a_n, & \alpha > 0.
\end{cases}
\]

(54)

Let \( q = t/(1+t) \). Then we have \( \sum_{n=2N}^{\infty} a_n = (1+t)^{-\alpha} q^{2N^2} \) for \(-1 < \alpha \leq 0\). For \( \alpha > 0 \), one has
\[
\sum_{n=2N}^{\infty} n^\alpha a_n = (1+t)^{-\alpha} q^{2N^2} (2N)^\alpha \sum_{n=0}^{\infty} q^n \left( 1 + \frac{n}{2N} \right)^\alpha \leq C_{\alpha,t} 2^\alpha (1+t)^{1-\alpha} q^{2N^2} N^{\alpha},
\]
where \( C_{\alpha,t} = \sum_{n=0}^{\infty} (n+2)^\alpha q^{2N^2} \) is used. With the above equation and (54) yields (42) for \( T = 1 \). Using the following relation
\[
|J^{\alpha}[T, e^{-t\lambda}] - Q_N^\alpha[T, e^{-t\lambda}]| = T^{-\alpha-1} |J^{\alpha}[e^{-(t/T)\lambda}] - Q_N^\alpha[e^{-(t/T)\lambda}]|
\]
leads to (42) for any \( T > 0 \). The proof is complete.

Proof of Theorem 2.

Proof The following results can be found in [12],
\[
\lambda_j < \frac{2j+\alpha+3}{2N+\alpha+3} \left( 2j+\alpha+3 + \sqrt{(2j+\alpha+3)^2 + 0.25 - \alpha^2} \right), \quad (55)
\]
\[
\lambda_j > \frac{2(J_{\alpha,j}/2)^2}{2N+\alpha+3}, \quad J_{\alpha,j} = \pi (j+3/4+\alpha/2) + O(j^{-1}), \quad (56)
\]
where (56) holds when \( j \) is sufficiently large. For a sufficiently large \( N \), the Laguerre polynomial satisfies (see (7.14) in [34])
\[
|L_N^{(\alpha)}(x)| \approx \pi^{-1/2}(Nx)^{-1/4} e^{x/2}, \quad \forall x \geq 0.
\]

(57)

With (55)–(57) and (41), we have the following estimate
\[
w_j^{(\alpha)} = C_\alpha (N+1)^\alpha \left( \frac{j+1}{N+1} \right)^3 e^{-\lambda_j} \leq C (N+1)^\alpha e^{-\lambda_j},
\]

(58)
for sufficiently large \( j \), where \( \lambda_j = \theta_j (j + 1)^2 / (N + 1) \) and \( \theta_j \) is bounded and approximately between 0.25 and 4. The proof is completed. \( \square \)

**Proof of Theorem 3.**

**Proof** For notational simplicity, we denote
\[
\hat{T}_\ell = T_{\ell - 1} + \Delta T - \tau, \quad \hat{H}_\ell^0 (s) = \hat{t}_n - T_{\ell - 1} - s.
\]

Next, we investigate how to estimate \( N_\ell \) in (35), such that the LG quadrature \( Q^{-\alpha}_{N_\ell} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] \) to the integral \( J^{-\alpha} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] \) preserves the accuracy up to \( O(\epsilon) \) for all \( s \in [s_\ell, s_{\ell - 1}] \).

From Theorem 1, we know that the error of \( Q^{-\alpha}_{N_\ell} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] \) mainly depends on the following term
\[
(\hat{H}_\ell^0 (s)/\hat{T}_\ell)^{2N} = \left( \frac{\hat{t}_n - s - T_{\ell - 1}}{T_{\ell - 1} + \Delta T - \tau} \right)^{2N}, \quad s \in [s_\ell, s_{\ell - 1}].
\]

Using (34) and \( T_{\ell - 1} = B^{\ell - 1} \tau \) gives
\[
0 \leq \frac{\hat{t}_n - s - T_{\ell - 1}}{T_{\ell - 1} + \Delta T - \tau} \leq \frac{2B - 1 - B^{1-\ell}}{1 + B^{1-\ell} (\Delta T / \tau - 1)} = T_\ell \leq 2B - 1, \quad \forall \ell \geq 1. \tag{59}
\]

Using the above inequality and Eq. (42) yields
\[
|J^{-\alpha} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] - Q^{-\alpha}_{N_\ell} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}]| \leq C a \hat{T}_\ell^{\alpha - 1} N_\ell^{\alpha} \left( \frac{\hat{H}_\ell^0 (s)/\hat{T}_\ell}{1 + \hat{H}_\ell^0 (s)/\hat{T}_\ell} \right)^{2N_\ell} \leq C a \hat{T}_\ell^{\alpha - 1} N_\ell^{\alpha} \left( \frac{T_\ell}{1 + T_\ell} \right)^{2N_\ell}. \tag{60}
\]

Since the relative error of (60) is independent of \( \hat{T}_\ell \), we can let \( (T_\ell / (T_\ell + 1))^{2N_\ell} \leq \epsilon \), which yields the minimum \( N_\ell \) given by (36).

From (44) and (60), we derive that the pointwise error of the truncated quadrature \( Q^{-\alpha}_{N_\ell, \epsilon_0} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] \) for all \( s \in [s_\ell, s_{\ell - 1}] \) is given by
\[
E_{\epsilon, \epsilon_0}^{-\alpha, \ell} [e^{-\hat{H}_\ell^0 (s) \lambda}] = J^{-\alpha} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] - Q^{-\alpha}_{N_\ell, \epsilon_0} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}]
\]
\[
= \left( J^{-\alpha} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] - Q^{-\alpha}_{N_\ell} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] \right) + \left( Q^{-\alpha}_{N_\ell} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] - Q^{-\alpha}_{N_\ell, \epsilon_0} [\hat{T}_\ell, e^{-\hat{H}_\ell^0 (s) \lambda}] \right)
\]
\[
= O(\epsilon) + O(\epsilon_0),
\]

where \( Q^{-\alpha}_{N_\ell, \epsilon_0} \) is defined by (44) and \( N_\ell \) is given by (36).
From (35) and (61), we have

\[
|H_{\Delta T}^n(I_T^H u, t_n) - F_{\Delta T}(\alpha, n) u| = \left| \frac{\sin(\alpha \pi)}{\pi} \sum_{\ell=1}^{L} E^{-\alpha, \ell} \left[ e^{-\alpha(\ell-1)} y(s_{\ell-1}, s_\ell, \lambda) \right] \right|
\]

\[
= \left| \frac{\sin(\alpha \pi)}{\pi} \sum_{\ell=1}^{L} \int_{s_{\ell-1}}^{s_\ell} E^{-\alpha, \ell} \left[ e^{-\alpha(H(s)\lambda)} \right] I_T^H u(s) \, ds \right|
\]

\[
\leq C(\epsilon + \epsilon_0) \int_{0}^{t_n} |I_T^H u(s)| \, ds
\]

\[
\leq C \max\{0, t_{n+1} - \Delta T\} \|u\|_{\infty}(\epsilon + \epsilon_0).
\]

The proof is complete. \qed

References

1. Baffet, D., Hesthaven, J.S.: High-order accurate adaptive kernel compression time-stepping schemes for fractional differential equations. J. Sci. Comput. 72(3), 1169–1195 (2017)
2. Baffet, D., Hesthaven, J.S.: A kernel compression scheme for fractional differential equations. SIAM J. Numer. Anal. 55(2), 496–520 (2017)
3. Banjai, L., López-Fernández, M., Schädle, A.: Fast and oblivious algorithms for dissipative and two-dimensional wave equations. SIAM J. Numer. Anal. 55(2), 621–639 (2017)
4. Beylkin, G., Monzón, L.: Approximation by exponential sums revisited. Appl. Comput. Harmon. Anal. 28(2), 131–149 (2010)
5. D’Amore, L., Murli, A., Rizzardi, M.: An extension of the Henrici formula for Laplace transform inversion. Inverse Probl. 16(5), 1441–1456 (2000)
6. Deng, W.: Short memory principle and a predictor–corrector approach for fractional differential equations. J. Comput. Appl. Math. 206(1), 174–188 (2007)
7. Diethelm, K.: Generalized compound quadrature formulae for finite-part integrals. IMA J. Numer. Anal. 17(3), 479–493 (1997)
8. Diethelm, K.: The Analysis of Fractional Differential Equations. Springer, Berlin (2010)
9. Diethelm, K., Ford, J.M., Ford, N.J., Weilbeer, M.: Pitfalls in fast numerical solvers for fractional differential equations. J. Comput. Appl. Math. 186(2), 482–503 (2006)
10. Ford, N.J., Simpson, A.C.: The numerical solution of fractional differential equations: speed versus accuracy. Numer. Algorithms 26(4), 333–346 (2001)
11. Galeone, L., Garrappa, R.: Fractional Adams–Moulton methods. Math. Comput. Simul. 79(4), 1358–1367 (2008)
12. Gatteschi, L.: Asymptotics and bounds for the zeros of laguerre polynomials: a survey. J. Comput. Appl. Math. 144(1), 7–27 (2002)
13. Jiang, H., Liu, F., Turner, I., Burrage, K.: Analytical solutions for the multi-term time-fractional diffusion-wave/diffusion equations in a finite domain. Comput. Math. Appl. 64(10), 3377–3388 (2012)
14. Jiang, S., Zhang, J., Zhang, Q., Zhang, Z.: Fast evaluation of the Caputo fractional derivative and its applications to fractional diffusion equations. Commun. Comput. Phys. 21(3), 650–678 (2017)
15. Jin, B., Lazarov, R., Zhou, Z.: An analysis of the L1 scheme for the subdiffusion equation with nonsmooth data. IMA J. Numer. Anal. 36(1), 197–221 (2016)
16. Li, C., Yi, Q., Chen, A.: Finite difference methods with non-uniform meshes for nonlinear fractional differential equations. J. Comput. Phys. 316, 614–631 (2016)
17. Li, C., Zeng, F.: Numerical Methods for Fractional Calculus. Chapman & Hall/CRC Numerical Analysis and Scientific Computing. CRC Press, Boca Raton (2015)
18. Li, J.R.: A fast time stepping method for evaluating fractional integrals. SIAM J. Sci. Comput. 31(6), 4696–4714 (2010)
19. Li, Z., Liang, Z., Yan, Y.: High-order numerical methods for solving time fractional partial differential equations. J. Sci. Comput. 71(2), 785–803 (2017)
20. López-Fernández, M., Lubich, C., Schädle, A.: Adaptive, fast, and oblivious convolution in evolution equations with memory. SIAM J. Sci. Comput. 30(2), 1015–1037 (2008)
21. Lubich, C.: Discretized fractional calculus. SIAM J. Math. Anal. 17(3), 704–719 (1986)
22. Lubich, C., Schädle, A.: Fast convolution for nonreflecting boundary conditions. SIAM J. Sci. Comput. 24(1), 161–182 (2002)
23. Luchko, Y.: Initial-boundary problems for the generalized multi-term time-fractional diffusion equation. J. Math. Anal. Appl. 374(2), 538–548 (2011)
24. Lv, C., Xu, C.: Error analysis of a high order method for time-fractional diffusion equations. SIAM J. Sci. Comput. 38(5), A2699–A2724 (2016)
25. Mastroianni, G., Monegato, G.: Truncated quadrature rules over (0, ∞) and Nyström-type methods. SIAM J. Numer. Anal. 41, 1870–1892 (2006)
26. Mastroianni, G., Monegato, G.: Some new applications of truncated Gauss–Laguerre quadrature formulas. Numer. Algorithms 49, 283–297 (2008)
27. McLean, W.: Fast summation by interval clustering for an evolution equation with memory. SIAM J. Sci. Comput. 34(6), A3039–A3056 (2012)
28. McLean, W.: Exponential sum approximations for $t^{-\beta}$. arXiv:1606.00123 (2016)
29. McLean, W., Mustapha, K.: A second-order accurate numerical method for a fractional wave equation. Numer. Math. 105(3), 481–510 (2007)
30. Metzler, R., Klafter, J.: The random walk’s guide to anomalous diffusion: a fractional dynamics approach. Phys. Rep. 339(1), 77 (2000)
31. Podlubny, I.: Fractional differential equations. Academic Press Inc, San Diego (1999)
32. Samko, S.G., Kilbas, A.A., Marichev, O.I.: Fractional Integrals and Derivatives: Theory and Applications. Gordon and Breach Science Publishers, Yverdon (1993)
33. Schädle, A., López-Fernández, M., Lubich, C.: Fast and oblivious convolution quadrature. SIAM J. Sci. Comput. 28(2), 421–438 (2006)
34. Shen, J., Tang, T., Wang, L.L.: Spectral Methods, Springer Series in Computational Mathematics, vol. 41. Springer, Heidelberg (2011)
35. Stynes, M., O’Riordan, E., Gracia, J.L.: Error analysis of a finite difference method on graded meshes for a time-fractional diffusion equation. SIAM J. Numer. Anal. 55(2), 1057–1079 (2017)
36. Sun, Zz, Wu, X.: A fully discrete difference scheme for a diffusion-wave system. Appl. Numer. Math 56(2), 193–209 (2006)
37. Wang, C.L., Wang, Z.Q., Jia, H.L.: An hp-version spectral collocation method for nonlinear Volterra integro-differential equation with weakly singular kernels. J. Sci. Comput. 72, 647–678 (2017)
38. Wang, D., Xiao, A.: Dissipativity and contractivity for fractional-order systems. Nonlinear Dyn. 80(1), 287–294 (2015)
39. Weideman, J.A.C.: Optimizing talbots contours for the inversion of the laplace transform. SIAM J. Numer. Anal. 44(6), 2342–2362 (2006)
40. Weideman, J.A.C., Trefethen, L.N.: Parabolic and hyperbolic contours for computing the Bromwich integral. Math. Comput. 76(259), 1341–1356 (2007)
41. Xiang, S.: Asymptotics on Laguerre or Hermite polynomial expansions and their applications in Gauss quadrature. J. Math. Anal. Appl. 393(2), 434–444 (2012)
42. Yan, Y., Sun, Z.Z., Zhang, J.: Fast evaluation of the caputo fractional derivative and its applications to fractional diffusion equations: a second-order scheme. Commun. Comput. Phys. 22(4), 1028–1048 (2017)
43. Yu, Y., Perdikaris, P., Karniadakis, G.E.: Fractional modeling of viscoelasticity in 3D cerebral arteries and aneurysms. J. Comput. Phys. 323, 219–242 (2016)
44. Zayernouri, M., Matzavinos, A.: Fractional Adams-Bashforth/Moulton methods: an application to the fractional Keller–Segel chemotaxis system. J. Comput. Phys. 317, 1–14 (2016)
45. Zeng, F., Zhang, Z., Karniadakis, G.E.: Fast difference schemes for solving high-dimensional time-fractional subdiffusion equations. J. Comput. Phys. 307, 15–33 (2016)
46. Zeng, F., Zhang, Z., Karniadakis, G.E.: Second-order numerical methods for multi-term fractional differential equations: smooth and non-smooth solutions. Comput. Methods Appl. Mech. Eng. 327, 478–502 (2017)