Available online at http://scik.org
J. Math. Comput. Sci. 4 (2014), No. 2, 148-166
ISSN: 1927-5307

OPERATORIAL TAU METHOD FOR FRACTIONAL DIFFERENTIAL EQUATIONS

DAMIAN TRIF

Department of Mathematics, Babeş–Bolyai University, 400084, Cluj–Napoca, Romania

Abstract. The paper extends the applicability of our Matlab package *Chebpack* to find Chebyshev polynomial approximations of the solutions of some fractional differential equations. Numerical examples confirm the efficiency of this approach.

Keywords: fractional differential equations; Chebyshev polynomials; tau method.

2010 AMS Subject Classification: 26A33, 34A08, 34K28.

1. Introduction

The field of fractional calculus is almost as old as calculus itself. The non-integer derivatives are non-local operators which possess a memory effect explaining their significant uses in many applications such as the behavior of viscoelastic materials or polymers, for example.

Several methods to solve the fractional derivatives, fractional integrals and fractional differential equations have recently been proposed. Many interesting Matlab functions were already submitted to the *MathWorks, Inc. Matlab Central File Exchange*, where they are freely downloadable for sharing among the users.

Received February 6, 2014

We refer here to the well-documented programs of Podlubny [24], [25] for fractional differential equations and [26] for distributed order ODEs, based on the Grünwald–Letnikov derivative. Another useful Matlab program is `fde12.m` of Garrappa [15] for (non-linear) differential equations of fractional order, based on the predictor-corrector method of Adams-Bashforth-Moulton described in Diethelm's paper [11].

We must remark that any algorithm using a discretization of a non-integer derivative has to take into account its non-local structure which imply, in general, high storage requirements and a great overall complexity of the algorithm.

There is a comprehensive literature on the theory, applications and numerical methods of the fractional calculus and the number of such textbooks and papers has grown rapidly in the recent years. The main sources used in this paper were [17], [9], [19], [23], [3], [14], [33], [22] and the references therein.

The aim of the presented paper is to extend the capabilities of our Matlab package *Chebpack* [32] for fractional calculus problems. *Chebpack* is based on the operational form of the Chebyshev spectral tau method [20], [8], [7]. The main advantage of *Chebpack* is a unified approach for initial value problems, boundary value problems, eigenproblems, nonlocal problems for ordinary, fractional or distributed order differential equations.

The Chebyshev polynomials were previously used for approximating the fractional derivatives in the Matlab package *Chebfun* of Trefethen [30]. Chebyshev and other kind of orthogonal polynomials were used for fractional differential equations in the papers [18], [16], [12], [13], [28], [4], [5], [6].

The paper is structured as follows: in section 2 we describe the tau method, in section 3 we give the discretization of the main fractional operators and in section 4 we give numerical examples to illustrate the facilities of *Chebpack* for fractional calculus. Finally, the paper ends with conclusions and references.

All the necessary Matlab code for reproducing the examples are now part of an updated version of *Chebpack* [32], in the folder *Examples,* subfolder *Fractional differential equations*.

2. The tau method

If we have to solve a differential or an integral problem, one of the most effective methods is to use the Chebyshev spectral methods [7], i.e. to approximate the solution y by a finite sum of a Chebyshev series

$$y(x) \approx \frac{1}{2}c_0T_0(x) + c_1T_1(x) + \cdots + c_{n-1}T_{n-1}(x),\ x \in [-1,1].$$

Here $T_k(x) = \cos(k\cos^{-1}(x))$, $k = 0,1,2,...,n-1$ are the Chebyshev polynomials of the first kind and the coefficients $\mathbf{c} = \{c_k, k = 0,1,...,n-1\}$ represented as a column vector are the unknowns. If $\mathscr{L} : C^\infty(-1,1) \to C^\infty(-1,1)$ is a linear operator then let $\mathbf{u}$ be the corresponding coefficients of $\mathscr{L}(y)(x)$. The matrix $L : \mathbb{R}^n \to \mathbb{R}^n$ that maps $\mathbf{c}$ into $\mathbf{u} = L\mathbf{c}$ is the Chebyshev approximation of $\mathscr{L}$.

Our Matlab package *Chebpack* [31], [32] implements the Chebyshev spectral method as a *Lanczos' tau method*, where we work in the spectral space of the coefficients. The linear operators of the differential or integral problem, such as differentiation, integration, product with the independent variable, modified argument, are discretized to their corresponding approximating matrices. The final form of the linear problem $\mathscr{L}(y)(x) = f(x)$ becomes, after the discretization, a pure algebraic linear problem in an *operational form* $L \cdot \mathbf{c} = \mathbf{f}$ with the supplementary conditions of the continuous problem included.

Consequently, *Chebpack* assumes the representation of the unknown functions in truncated Chebyshev polynomials series and the corresponding matrices L are given to act directly on the Chebyshev coefficients. Generally, this kind of discretization is performed by using the recurrence formula and the fast schemes of computation with Chebyshev polynomials, see [31] for more technical details on this topic.

Precisely, a sufficiently well-behaved function $y(x)$ can be approximated by its physical representation $y(x_1)$, $y(x_2)$, ..., $y(x_n)$ of values of y at the given gridpoints $\mathbf{x}$, for example at the *Chebyshev points of the first or of the second kind*

$$x_k^{(1)} = -\cos\frac{(2k-1)\,\pi}{2n}, x_k^{(2)} = -\cos\frac{(k-1)\,\pi}{n-1},\ k = 1,...,n \tag{2.1}$$

or by its spectral representation $\mathbf{c} = \{c_0, c_1, ..., c_{n-1}\}$ where

$$y_{n-1}(x) = \frac{c_0}{2}T_0(x) + c_1T_1(x) + \cdots + c_{n-1}T_{n-1}(x) \tag{2.2}$$

is the unique polynomial obtained by interpolating $y(x)$ through the points x_1, ..., x_n.

Of course, the Chebyshev polynomials are defined on $dom = [-1,1]$ but any interval $dom = [a,b]$ can be shifted to $[-1,1]$ and we may use the shifted Chebyshev polynomials

$$T_k^*(x) = T_k(\alpha x+\beta),\ x\in[a,b],\ \alpha=\frac{2}{b-a},\ \beta=-\frac{b+a}{b-a},\ k=0,1,...,n-1. \tag{2.3}$$

The code `[x,w]=pd(n,dom,kind)` of *Chebpack* calculates the n Chebyshev points $\mathbf{x}$ of the kind "`kind`" on `dom` as a column vector and the n quadrature weights $\mathbf{w}$ as a row vector for the Clenshaw-Curtis quadrature formula

$$\int_a^b y(x)dx \simeq \sum_{j=1}^{n} w_j y\left(x_j\right) \equiv \mathbf{w}\cdot y\left(\mathbf{x}\right).$$

The *fast* conversion between the spectral representation $\mathbf{c}$ of a function y and its physical values $\mathbf{v} = y(\mathbf{x})$ is performed by the functions `v=t2x(c,kind)` and `c=x2t(v,kind)` based on the Fast Chebyshev Transform.

If a function y is given by its Chebyshev coefficients $\mathbf{c}$ and we need its values at some points $\mathbf{x}_c \in dom$, a conversion matrix T_c is obtained from the code `Tc=cpv(n,xc,dom)`, where

$$T_c = [T_0/2, T_1(\xi), ..., T_{n-1}(\xi)],\ \xi = \frac{2\mathbf{x}_c}{b-a} - \frac{b+a}{b-a} \in [-1,1] \tag{2.4}$$

and ξ is transposed as a column vector. The code is based on the recurrence relations

$$T_1(x) = xT_0,\ T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x),\ k = 1,2,... \tag{2.5}$$

and we have $\mathbf{v}_c = T_c\mathbf{c}$. If `T=cpv(n,x,dom)` where $\mathbf{x}$ are the Chebyshev points, then the matrix T performs the (*non fast*, but useful if $n \le 256$) conversion between the coefficients $\mathbf{c}$ of the function y and the values $\mathbf{v} = y(\mathbf{x})$ through the formulas $\mathbf{v} = T\mathbf{c}$ and $\mathbf{c} = T^{-1}\mathbf{v}$.

It is important to remark that linear operators are better represented in the spectral space of the coefficients while the nonlinear operators are easily handled in the physical space of the values, see *Example 5*.

Another useful code is `X=mult(n,dom)` also based on the recurrence relations (2.5). If $\mathbf{c}$ is the column of Chebyshev coefficients of a function $y(x)$, then $X\cdot\mathbf{c}$ is the column of the Chebyshev coefficients of the multiplication by the independent variable $xy(x)$ for $x \in dom$. If `a` is

the vector of the first m Chebyshev coefficients of a function $a(x)$, then the vector of the coefficients of $a(x)y(x)$ is $A\cdot\mathbf{c}$ where the multiplication matrix A is given by the code `A=fact(a,m)` based on the relations

$$2T_j(x)T_k(x)=T_{j+k}(x)+T_{|k-j|}(x),\ j,k=0,1,... \tag{2.6}$$

The differentiation matrix D is given by the code `D=deriv(n,dom)`. If $\mathbf{c}$ is the column of Chebyshev coefficients of a function $y(x)$, then $D\cdot\mathbf{c}$ is the column of the Chebyshev coefficients of the derivative $\frac{dy}{dx}$. The definition of D is based on the relations

$$T_0=T_1',\ T_1=\frac{T_2'}{4},\ T_k=\frac{T_{k+1}'}{2(k+1)}-\frac{T_{k-1}'}{2(k-1)},\ k=2,3,...,x\in[-1,1]. \tag{2.7}$$

Similarly, the code `[J,J0]=prim(n,dom)` calculates the integration matrix J such that the coefficients of a primitive of $y(x)$ are $J\cdot\mathbf{c}$. The coefficients of the particular primitive vanishing at $a=dom(1)$ are obtained by using $J_0\cdot\mathbf{c}$.

All the above codes take into account a general *dom*, see [31], [32] for more details.

3. The fractional operators

The operators of the fractional calculus can also be expressed by matrices. Let us start with *the Riemann–Liouville fractional integral operator of order q*

$$J^qy(x)=\frac{1}{\Gamma(q)}\int_0^x\frac{y(t)dt}{(x-t)^{1-q}}, \tag{3.1}$$

where $0<q<1$, $0\leq x\leq b$. If $y\in L_1[0,b]$ is a sufficiently well-behaved function, see [9], then it has the spectral approximation

$$y(x)\simeq{\sum_{k=0}^{n-1}}'c_kT_k(\alpha x-1),$$

where $\alpha=2/b$ and the prime sign denotes the summation whose first term is halved. Then

$$J^qy(x_j)\simeq\frac{1}{\Gamma(q)}{\sum_{k=0}^{n-1}}'c_k\int_0^{x_j}\frac{T_k(\alpha t-1)\,dt}{(x_j-t)^{1-q}},\ j=1,...,n$$

approximates the physical values $\mathbf{v}$ of $J^qy(\mathbf{x})$ at the Chebyshev points $\mathbf{x}=\left(x_j\right)_{j=1,...,n}$. Consequently, the spectral approximation of $J^qy(x)$ is given by the Chebyshev coefficients $T^{-1}\mathbf{v}\equiv I\cdot\mathbf{c}$, where T is given by (2.4), $\mathbf{c}$ is the column vector of the coefficients $c_0,...,c_{n-1}$ and I is the

basic integration matrix. If `dom=[0,b]` and **x** is given by `x=pd(n,dom,kind)`, then the code

```
I=fracbas(x,dom,q)
```

for $0<q<1$ calculates the above matrix I.

The calculation starts with the elements I_{jk}, $j=1,2,...,n$, $k=0,1,...,n-1$,

$$I_{jk}=\int_0^{x_j}\frac{T_k(\alpha t-1)\,dt}{(x_j-t)^{1-q}}. \tag{3.2}$$

We have

$$\int_0^x\frac{dt}{(x-t)^{1-q}}=\frac{x^q}{q},\ \int_0^x\frac{tdt}{(x-t)^{1-q}}=\frac{x^{q+1}}{q(q+1)},$$
$$\int_0^x\frac{t^2dt}{(x-t)^{1-q}}=\frac{2x^{q+2}}{q(q+1)(q+2)}.$$

Consequently, the columns $I_{\cdot,k}$ for $k=0,1,2$ are

$$I_{\cdot,0}=\frac{1}{2}\int_0^x\frac{dt}{(x-t)^{1-q}}=\frac{x^q}{2q},\ I_{\cdot,1}=\int_0^x\frac{(\alpha t-1)\,dt}{(x-t)^{1-q}}=\alpha\frac{x^{q+1}}{q(q+1)}-\frac{x^q}{q},$$
$$I_{\cdot,2}=\int_0^x\frac{\left[2(\alpha t-1)^2-1\right]dt}{(x-t)^{1-q}}=4\alpha^2\frac{x^{q+2}}{q(q+1)(q+2)}-4\alpha\frac{x^{q+1}}{q(q+1)}+\frac{x^q}{q}.$$

The columns for $k>2$ are calculated iteratively, by using the recurrence formulas

$$T_{k+1}(x)=2xT_k(x)-T_{k-1}(x),\ T_k(x)=\frac{T'_{k+1}(x)}{2(k+1)}-\frac{T'_{k-1}(x)}{2(k-1)},\ -1\le x\le 1.$$

shifted to $x\in[0,b]$. We have

$$I_{\cdot,k}=\int_0^x\frac{T_k(\alpha t-1)}{(x-t)^{1-q}}dt=\int_0^x\frac{2(\alpha t-1)T_{k-1}(\alpha t-1)-T_{k-2}(\alpha t-1)}{(x-t)^{1-q}}dt=$$
$$=2\alpha\int_0^x\frac{tT_{k-1}(\alpha t-1)}{(x-t)^{1-q}}dt-2I_{\cdot,k-1}-I_{\cdot,k-2}=$$
$$=-2\alpha\int_0^x\frac{T_{k-1}(\alpha t-1)}{(x-t)^{-q}}dt+2\alpha xI_{\cdot,k-1}-2I_{\cdot,k-1}-I_{\cdot,k-2}.$$

But

$$\int_0^x \frac{T_{k-1}(\alpha t-1)}{(x-t)^{-q}}dt=\frac{1}{\alpha}\int_0^x\left(\frac{\frac{d}{dt}T_k(\alpha t-1)}{2k}-\frac{\frac{d}{dt}T_{k-2}(\alpha t-1)}{2(k-2)}\right)\frac{dt}{(x-t)^{-q}}=$$
$$=\frac{1}{2\alpha}\left\{\left[\frac{T_k(\alpha t-1)}{k}-\frac{T_{k-2}(\alpha t-1)}{k-2}\right]\frac{1}{(x-t)^{-q}}\Bigg|_{t=0}^{t=x}\right\}+$$
$$+\frac{1}{2\alpha}\int_0^x\left[\frac{T_k(\alpha t-1)}{k}-\frac{T_{k-2}(\alpha t-1)}{k-2}\right]\frac{q}{(x-t)^{1-q}}dt=$$
$$=\frac{1}{2\alpha}\left\{-\left[\frac{(-1)^k}{k}-\frac{(-1)^{k-2}}{k-2}\right]x^q+\frac{q}{k}I_{\cdot,k}-\frac{q}{k-2}I_{\cdot,k-2}\right\}$$

and, consequently,

$$I_{\cdot,k}=-\left\{\frac{2(-1)^k x^q}{k(k-2)}+\frac{q}{k}I_{\cdot,k}-\frac{q}{k-2}I_{\cdot,k-2}\right\}+2\alpha x I_{\cdot,k-1}-2I_{\cdot,k-1}-I_{\cdot,k-2}, \tag{3.3}$$
$$I_{\cdot,k}\left(1+\frac{q}{k}\right)=2(\alpha x-1)I_{\cdot,k-1}+\left(\frac{q}{k-2}-1\right)I_{\cdot,k-2}-\frac{2(-1)^k x^q}{k(k-2)}.$$

Finally, the above calculated matrix I must be replaced by $\frac{1}{\Gamma(q)}\cdot T^{-1}\cdot I$ and, if the function y has the spectral representation given by its Chebyshev coefficients $\mathbf{c}=\left(c_j\right)_{j=0,\ldots,n-1}$, then the vector $I\cdot\mathbf{c}$ will contain the spectral coefficients of $J^q y(x)$, for $q\in(0,1)$.

The function `fracbas` can be tested on the well known example

$$J^{1/2}e^x\equiv\frac{1}{\sqrt{\pi}}\int_0^x\frac{e^t}{\sqrt{x-t}}dt=e^x\operatorname{erf}\left(\sqrt{x}\right),$$

for $x>0$, say for $x\in[0,2]$. The code `test1.m`

```
n=16;dom=[0,2];kind=2;x=pd(n,dom,kind);q=1/2;
I=fracbas(x,dom,q);c=x2t(exp(x),kind);reznum=t2x(I*c,kind);
```

gives the numerical approximation `reznum` which reproduces the exact result at the points `x` with an error of 2.65×10^{-15} for $n=16$.

Another example is the Abel integral equation of the second kind (compare with the results from [2], Example 7)

$$y(x)+\int_0^x\frac{y(t)dt}{\sqrt{x-t}}=2\sqrt{x},$$

with exact solution $y_{ex}(x) = 1 - e^{\pi x} erfc(\sqrt{\pi x})$. This equation of the form $\left(\frac{1}{\sqrt{\pi}} + J^{0.5}\right) u(x) = 2\sqrt{\frac{x}{\pi}}$ is discretized in spectral form as

$$\left(\frac{E}{\sqrt{\pi}} + I\right) \mathbf{c} = \mathbf{b}$$

where E is the unit matrix, I is the discretization of $J^{\frac{1}{2}}$ given by `fracbas` and $\mathbf{b}$ is the column vector of the coefficients of the function $2\sqrt{\frac{x}{\pi}}$ at the Chebyshev points $\mathbf{x}$. Finally, the coefficients $\mathbf{c}$ are converted to the numerical values $\mathbf{v}$ of the solution y at the points $\mathbf{x}$. The code `test2.m`

```
function [solnum,solex]=test2(n)
dom=[0,1];kind=2;q=0.5;x=pd(n,dom,kind);
I=fracbas(x,dom,q);
A=1/sqrt(pi)*eye(n)+I;b=x2t(2*sqrt(x/pi),kind);
c=A\b;% here the equation is solved
solnum=t2x(c,kind);solex=1-exp(pi*x).*erfc(sqrt(pi*x));
```

gives for $n = 64$ the numerical values of the solution with an error of 1.7543×10^{-4} and needs a computing time of 6×10^{-3} seconds. For $n = 512$ the error becomes 2.7658×10^{-6} for an elapsed time of 3.3 seconds.

Of course, this method is not suitable for functions with singularities in $[0,b]$, or singularities of lower-order derivatives, like $\sqrt{x}$ or $erfc(\sqrt{x})$; in these cases, n must be excessively large. In practical applications, however, it is required to approximate the fractional integrals or derivatives of such badly-behaved functions. A nice procedure is given in [18] for the fractional derivatives of functions f such that $f(x) = x^{\alpha} g(x)$, $\alpha > -1$, where g is assumed to be a well-behaved function. The idea is to use the above Chebyshev spectral approximation only for the function g. Recurrence relations for the Chebyshev coefficients of the fractional derivative are also given, but the algorithm is not put in an operational form. A future extension of *Chebpack* will take into account the approximation of such functions, too.

However, we may improve the above result by a small change in the Abel equation

$$y(x) - 2\sqrt{x} + \int_0^x \frac{y(t) - 2\sqrt{t}}{\sqrt{x-t}} dt = -\int_0^x \frac{2\sqrt{t}}{\sqrt{x-t}} dt.$$

The right-hand side is known to be the well-behaved function $-\pi x$ and for the new unknown function $v(x)=y(x)-2\sqrt{x}$ the equation becomes

$$v(x)+\int_0^x \frac{v(t)dt}{\sqrt{x-t}}=-\pi x.$$

With the corresponding change in the above code (`test2_improved.m`) we obtain the physical values of $v(x)$ and then $y(x)=v(x)+2\sqrt{x}$. Now, for $n=64$ the error becomes 6.4887×10^{-8} for almost the same computing time.

If $q=1$ the code `[~,J0]=prim(n,dom)` gives the integration matrix of order $q=1$ that vanishes at $x=0$. For a general $q\geq 0$ the Riemann-Liouville integration matrix Jrl of order q is:

- the unit matrix for $q=0$

- the matrix $I^{(q)}$ given by `fracbas` for $0<q<1$

- the matrix J_0 given by `prim` for $q=1$

- the matrix $I^{(q-[q])}\cdot J_0^{[q]}$ where $[q]$ is the largest integer not larger than q (`floor(q)` in Matlab), J_0 is given by `prim` and I is given by `fracbas` for $q-[q]$. The above formulas are implemented in an extended code `Jrl=primrl(x,dom,q)`.

Again, if y has the spectral representation given by its Chebyshev coefficients $\mathbf{c}=\left(c_j\right)_{j=0,\ldots,n-1}$, the vector $Jrl\cdot\mathbf{c}$ will contain the spectral coefficients of $J^q y(x)$, for $q\geq 0$.

The *Riemann-Liouville derivative of order q* is

$$ {}^{RL}D^q y(x)=\frac{1}{\Gamma(m-q)}\left(\frac{d}{dx}\right)^m\int_0^x\frac{y(t)dt}{(x-t)^{m-q-1}} \tag{3.4}$$

where $q\geq 0$, $m=\lceil q\rceil$ is the smallest integer not smaller than q (`ceil(q)` in Matlab). In an operatorial (discrete) form, ${}^{RL}D^q y=D^m\cdot J^{m-q}y$ where `D=deriv(n,dom)` is the usual first order differentiation matrix and J^{m-q} is calculated as above. The fractional Riemann-Liouville matrix ${}^{RL}D^q$ is given by the code `Drl=derivrl(x,dom,q)`.

The *Caputo derivative of order q* is

$$ {}^{C}D^q y(x)=\frac{1}{\Gamma(m-q)}\int_0^x\frac{y^{(m)}(t)dt}{(x-t)^{1+q-m}} \tag{3.5}$$

where q and m are as above. In the operatorial form, $D^q y=J^{m-q}\cdot D^m y$. The matrix ${}^{C}D^q$ is discretized by the code `Dc=derivc(x,dom,q)`.

We remark that for $0 < q < 1$, ${}^{RL}D^q y(x) = {}^{C}D^q y(x) + \frac{y(0)x^{-q}}{\Gamma(1-q)}$, see [9] for more details. All the above codes, `fracbas`, `primrl`, `derivrl`, `derivc` are now enclosed in an updated form of *Chebpack* in the folder `level0`. The same Chebyshev spectral method, *but not in operational form*, is used in Theorem 3.2 from [29] where the uniform approximation of the above fractional derivatives is proved.

Let us test the differentiation matrix for the function $y(x) = \sin x$, $x \in [0, 2\pi]$. The exact fractional derivative of order $q = 0.5$ is [9]

$$ {}^{RL}D^q \sin x = {}^{C}D^q \sin x = \frac{x^{-0.5}}{2i\Gamma(0.5)} \left[{}_1F_1(1, 0.5, ix) - {}_1F_1(1, 0.5, -ix) \right].$$

We approximate this derivative by using the above *Chebpack* matrix ${}^{C}D^{0.5}$, the *Chebfun* package [30] and the Podlubny's package [25]. The Matlab code is `test3.m` and the results are given in Table 1. For this example *Chebpack* has the most efficient running time and accuracy.

TABLE 1. Test of the fractional derivative $D^{0.5} \sin x$

Method	**Error**	**Time** (sec)	**Dimension**
Exact	0	4.45	-
Chebpack	1.09e-14	0.005	64
Chebfun	3.58e-13	0.59	20
Podlubny	0.05	0.02	629

Another test for a fractional differential problem is [24]

$$ {}^{C}D^q y(t) + y(t) = 1, t \in [0, 5]$$

$$ y(0) = 1, y'(0) = -1, q = 1.8$$

with the exact solution

$$ y(t) = E_{q,1}(-t^q) - tE_{q,2}(-t^q) + t^q E_{q,q+1}(-t^q).$$

The equation is discretized as a linear system $(D_c + E)\mathbf{c} = \mathbf{b}$, where $\mathbf{c}$ are the Chebyshev coefficients of the solution y and $\mathbf{b}$ are the Chebyshev coefficients of the constant function 1. The initial conditions are implemented in the last two rows of the above system by using the

matrix T given by `T=cpv(n,0,[0,1])`. Precisely, $T\cdot c = y(0) = 1$ and $T\cdot D\cdot c = y'(0) = -1$. The Matlab code is `test4.m`

```
[x,solnum]=test4(n,dom,q)
x=pd(n,dom,2);T=cpv(n,dom(1),dom);
Dc=derivc(x,dom,q);D=deriv(n,dom);
A=Dc+eye(n);b=x2t(ones(n,1),2);% the discrete form Ay=b
A(n-1,:)=T;b(n-1)=1;% y(0)=1
A(n,:)=T*D;b(n)=-1;% y'(0)=-1
sol=A\b;solnum=t2x(sol,2);
```

The error of the numerical solution `solnum` for $n = 64$ is 1.63×10^{-7} and the computing time is 0.012 seconds. By using the Podlubny's package [24], [25] with $h = 0.01$ the error is 0.0114 and the computing time is 0.47 seconds while `fde12` [15] with $h = 0.01$ gives an error of 9.24×10^{-6} with a computing time of 0.07 seconds.

4. Numerical examples

This section contains 5 examples for specific problems on fractional differential equations.

Example 1. Let us consider the initial-conditions problem for the multi-term Bagley-Torvik fractional differential equation [23]

$$y'' + {}^{C}D^{\frac{3}{2}}y + y = \begin{cases} 8, & x \le 1 \\ 0, & x > 1 \end{cases}, x \in [0,30]$$

$$y(0) = y'(0) = 0.$$

The "exact" solution of this problem is useless for comparison with the numerical solution. It is calculated in [23] and it uses the derivatives of the Mittag-Lefler function. The numerical solution on the grid x given by *Chebpack,* `example1.m` is compared with the numerical solution given by Podlubny's code and by `fde12` code for some stepsizes h. The results of this comparison (the maximal absolute differences between *Chebpack* solution and other solutions) are given in Table 2

TABLE 2. Bagley-Torvik equation, discontinuous rhs

Method	**Difference**	**Time** (sec)	**Dimension**
Chebpack	0	0.32	256
Podlubny h=0.075	0.48	0.27	400
Podlubny h=0.04	0.17	1.49	750
Podlubny h=0.02	0.07	11.16	1500
Podlubny h=0.01	0.04	86.38	3000
fde12 h=0.01	0.1202	0.49	3000
fde12 h=0.001	0.0957	5.27	30000

We remark that the code `fde12` can be used for the Bagley-Torvik equation as a fractional differential system of order $\frac{1}{2}$ [10]

$$y_1^{(\frac{1}{2})} = y_2, y_2^{(\frac{1}{2})} = y_3, y_3^{(\frac{1}{2})} = y_4, y_4^{(\frac{1}{2})} = F - y_1 - y_4,$$
$$y_1(0) = 0,\ y_2(0) = 0,\ y_3(0) = 0, y_4(0) = 0,$$

where F is the function given in the right-hand side of the equation.

For a more relevant comparison let us consider the Bagley-Torvik equation

$$y'' + y^{(\frac{3}{2})} + y = 2 + 4\sqrt{\frac{x}{\pi}} + x^2, x \in [0, 10]$$
$$y(0) = y'(0) = 0.$$

with the exact solution $y_{ex}(x) = x^2$. The results are given by `example1bis.m` in Table 3.

TABLE 3. Bagley-Torvik equation, continuous rhs

Method	**Dimension**	**Time** (sec)	**Error**
Chebpack	n=32	0.003	2.84e-14
Podlubny	h=0.01	3.55	0.01
fde12	h=0.001	5.23	1.08e-4

Example 2. Let us consider the test boundary value problem from [34]

$$y''(x)+0.5{}^{C}D^{q}y(x)+y(x)=3+x^{2}\left(\frac{x^{-q}}{\Gamma(3-q)}+1\right),x\in[0,1],q=0.5$$
$$y(0)=1,y(1)=2.$$

The exact solution is $y(x)=x^2+1$. The Matlab code in *Chebpack* is `example2.m` with $n=16$, $dom=[0,1]$, $q=0.5$. The maximal error is about 2×10^{-16}. For comparison, the error given in [34] for the same dimension of the problem (i.e. $h=\frac{1}{16}$) and by using a cubic spline approximation is 4.35×10^{-3}.

A more relevant example is the eigenvalue problem [1]

$${}^{C}D^{1.5}y(x)+\left(\frac{1}{x}+\lambda\right)y(x)=0,x\in(0,1),$$
$$y(0)=0,y'(1)=0.$$

This problem is discretized as $A\mathbf{c}=-\lambda B\mathbf{c}$ where $A=D_c+X^{-1}$, B is the unit matrix of order n and $\mathbf{c}$ is the column vector of the coefficients of the eigenfunctions. Now the boundary conditions are implemented such that the two last rows of B are the two last rows of A divided by $\lambda^*=1000$. Consequently, the boundary conditions are numerically imposed and two spurious eigenvalues $\lambda=-\lambda^*$ are introduced and can be easily eliminated while keeping the matrix B nonsingular. The approximating generalized eigenproblem $AY=-\lambda BY$ is then solved by using the function `eig` from Matlab.

The Matlab code is `example2bis.m`

```
x=pd(n,dom,2);Dc=derivc(x,dom,q);X=mult(n,dom);
T=cpv(n,x,dom);D=deriv(n,dom);
A=Dc+inv(X);B=eye(n);
A(n-1,:)=T(1,:);B(n-1,:)=A(n-1,:)/1000;
A(n,:)=T(n,:)*D;B(n,:)=A(n,:)/1000;
[V,L]=eig(full(A),full(B));[LL,ind]=sort(diag(L));VV=V(:,ind);
lam=-LL(1:3);
```

where $n=128$, $dom=[0,1]$, $q=1.5$. The numerical eigenvalues are compared with those from [1] in Table 4. The computing time was 0.13 seconds.

TABLE 4. Fractional eigenproblem

λ **(Chebpack)**	λ [1]
1.6610	1.6609
13.5509	13.5504
20.5147	20.5145

Example 3. Let us consider now a nonlocal boundary value problem adapted from [27]

$$
\begin{aligned}
&{}^{C}D^{\frac{3}{4}}y(x)+\frac{x^2e^x}{5}y(x)-e^x\int_0^x ty(t)dt=\frac{6x^{\frac{9}{4}}}{\Gamma\left(\frac{13}{4}\right)},\, x\in[0,1]\,,\\
&y(0)+y(1)-\frac{e+1}{e-2}y'(0)+\frac{1}{6}y''(1)-10\int_0^1 ty(t)dt=0.
\end{aligned}
$$

with the exact solution $y(x)=x^3$. Here the nonlocal condition is implemented by using the matrix T and the integration matrix J_0. Precisely, if `T=cpv(n,[0,1],[0,1])` and **c** are the Chebyshev coefficients of the solution y, then `J0*X*c` approximates the coefficients of the primitive $\int_0^x ty(t)dt$ and `(T(2,:)-T(1,:))*J0*X*c` implements the Newton-Leibniz formula to approximate $\int_0^1 ty(t)dt$. The Matlab code is `example3.m` and for $n=64$ the computing time is 0.01 sec. while the error is about 2.45×10^{-12}.

Example 4. Let us consider a distributed order differential problem, precisely the distributed order oscillator [19]

$$
\begin{aligned}
&y''(x)+D^{\varphi(\alpha)}y(x)+y(x)=\begin{cases}8, & 0\le x\le 1\\ 0, & x>1\end{cases},\, x\in[0,30]\,,\\
&y(0)=y'(0)=0,\,\varphi(\alpha)=6\alpha(1-\alpha)\,,\alpha\in[0,1]\,.
\end{aligned}
$$

Here the distributed order derivative is defined as

$$
D^{\varphi(\alpha)}y(x)=\int_0^1\varphi(\alpha)\frac{d^\alpha}{dx^\alpha}y(x)d\alpha
$$

where $\varphi(\alpha)$ is the weight function of distribution of order $\alpha\in[0,1]$.

The distributed order derivative is discretized as

$$
D^{\varphi(\alpha)}=\sum_{k=1}^{m}w_k\cdot\varphi(\alpha_k){}^{C}D^{\alpha_k}
$$

where the nodes α_k and the weights w_k are given by `[alpha,w]=pd(m,[0,1],1)` and it is coded in `doderiv.m`. This function is included in `level0` of *Chebpack*.

The code for the above problem is `example4.m`, the dimension of the problem is $n = 256$ for $dom = [0,30]$ and the discretization of the distributed order derivative is performed with $m = 16$. Chebpack needs 3.61 seconds computing time. Podlubny's code from [19] A.5 needs 1.28 seconds for $h = 0.075$ (dimension of the problem 400), 5.96 seconds for $h = 0.0375$ and 29.61 seconds for $h = 0.01875$. A magnified portion of the graphs of these numerical solutions for $x \in [5,15]$ is shown in Figure 1.

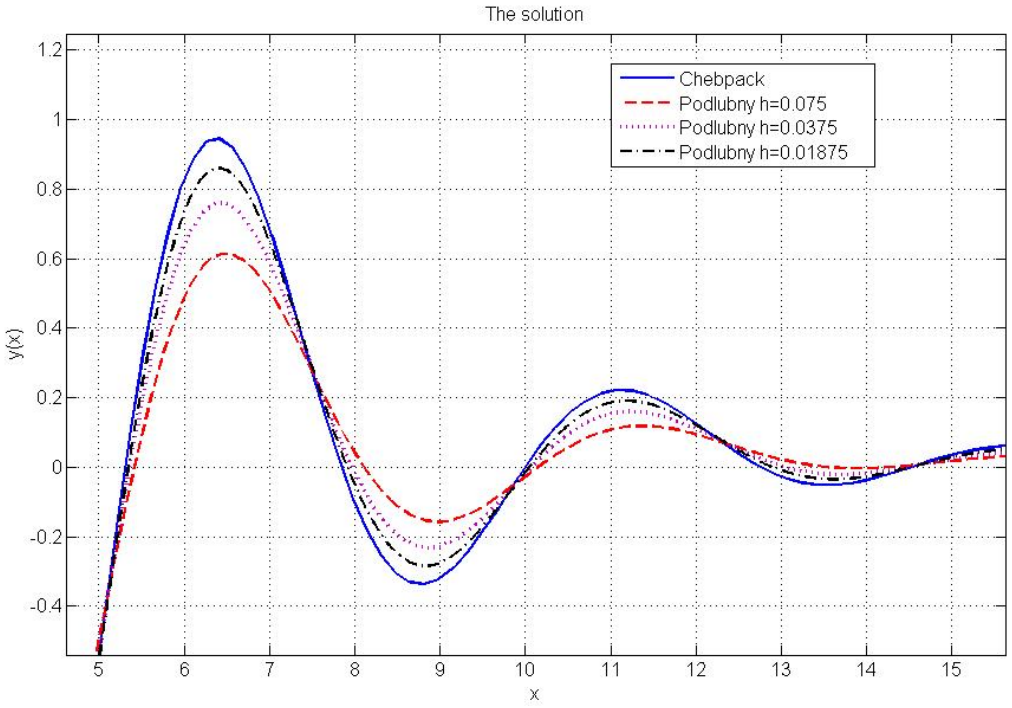

FIGURE 1. The numerical solutions for Bagley-Torvik distributed order problem

Example 5. Finally, let us consider a nonlinear problem [21]

$$D^{1.2}y(x) = y^4(x) + \frac{\Gamma(2.8)}{\Gamma(1.6)}x^{0.6} - \left(x^{1.8} - x + 1\right)^4, x \in [0,1]$$

$$y(0) = 1, y'(0) = -1.$$

This problem is of the form $Ly = N(x,y)$. After discretization, the discrete problem becomes $\mathscr{L}\mathbf{c} = \mathscr{N}(\mathbf{c})$ where $\mathbf{c}$ are the Chebyshev coefficients of y, $\mathscr{L}$ is the Caputo differentiation

matrix of order 1.2 and $\mathscr{N}$ is the nonlinear right-hand side function of the coefficients $\mathbf{c}$. Here the coefficients $\mathbf{C}$ of the nonlinear part y^4 are calculated by using the scheme

$$\mathbf{c} \xrightarrow{\texttt{t2x}} y(\mathbf{x}) \longrightarrow y^4(\mathbf{x}) \xrightarrow{\texttt{x2t}} \mathbf{C}.$$

Finally, after imposing the initial conditions in the last two rows of the matrix $\mathscr{L}$ and of the vector $\mathscr{N}(\mathbf{c})$, we obtain a nonlinear system $\mathbf{c} = \mathscr{L}^{-1}\mathscr{N}(\mathbf{c})$. This system can be solved in this case by successive iterations started by the Chebyshev coefficients of the constant function $yo(x) = 1$, for all $x \in [0,1]$. The code for this problem is `example5.m` and only 20 iterations are needed for a difference less than $1.e-13$ between two successive iterations. The exact solution is $y_{ex}(x) = x^{1.8} - x + 1$. The errors of the numerical solutions for different dimensions n are given in Table 5.

TABLE 5. Nonlinear problem

n	**Error**	**Time** (sec)
32	1.1e-5	0.013
64	1.1e-6	0.022
128	1.2e-7	0.08
256	1.3e-8	0.45
512	1.4e-9	3.86

We remark that the use of a grading exponent r in formula (3.1) from [21] makes the grid points more densely clustered near the left endpoint of the interval $[0,1]$. This distribution of the grid points is automatically performed in *Chebpack* by using the Chebyshev points (2.1) and consequently, the above results are consistent with the best results in [21].

5. Conclusion

The tau method implemented in our package *Chebpack* leads to very simple and efficient codes that can solve different kinds of problems for fractional differential equations in a unified approach. The Chebyshev grid points are automatically clustered near the left endpoint of the working interval but this is not enough for a good approximation if the solution has singularities at that point. This remaining problem to extend *Chebpack* to efficiently handle functions of algebraic singularity will be the aim of future work. All the necessary Matlab sources for reproducing the above tests and examples are now part of an updated version of *Chebpack* [32], in the folder *Examples,* subfolder *Fractional differential equations*.

Conflict of Interests

The author declares that there is no conflict of interests.

Acknowledgment

The author was supported by a grant of the Romanian National Authority for Scientific Research, CNCS – UEFISCDI, project number PN-II-ID-PCE-2011-3-0094.

REFERENCES

[1] S. Abbasbandy, A. Shirzadi, Homotopy analysis method for multiple solutions of the fractional Sturm-Liouville problems, Numer. Algorithms 54(2010), 521-532.

[2] Z. Avazzadeh, B. Shafiee, G. B. Loghmani, Fractional Calculus for Solving Abel's Integral Equations Using Chebyshev Polynomials, Appl. Math. Sci. 5(45)(2011), 2207 - 2216.

[3] P. Agrawal, P. Kumar, Comparison of Five Numerical Schemes for Fractional Differential Equations, in Advances in Fractional Calculus, J. Sabatier et al (eds), Springer, 2007, 43-60.

[4] A.H. Bhrawy, A.S. Alofi, S.S. Ezz-Eldien, A quadrature tau method for fractional differential equations with variable coefficients, Appl. Math. Lett. 24(2011), 2146-2152.

[5] A.H. Bhrawy, A.S. Alofi, The operational matrix of fractional integration for shifted Chebyshev polynomials, Appl. Math. Lett. 26(2013), 25-31.

[6] A.H. Bhrawy, M.M. Tharwat, A. Yildirim, A new formula for fractional integrals of Chebyshev polynomials: Application for solving multi-term fractional differential equations, Appl. Math. Model. 37(6)(2013), 4245-4252.

[7] J. P. Boyd, Chebyshev and Fourier Spectral Methods, Dover Publications, Inc., 2000.

[8] M. K. El-Daou, H. G. Khajah, Iterated solutions of linear operator equations with the tau method, Math. Comput. 66(217)(1997), 207-213.

[9] K. Diethelm, The Analysis of Fractional Differential Equations, An Application - Oriented Exposition Using Differential Operators of Caputo Type, Springer, 2010.

[10] K. Diethelm, N. J. Ford, Numerical Solution of the Bagley–Torvik Equation, BIT 42(3)(2002), 490-507.

[11] K. Diethelm, A.D. Freed, The Frac PECE subroutine for the numerical solution of differential equations of fractional order, in: S. Heinzel, T. Plesser (Eds.), Forschung und Wissenschaftliches Rechnen 1998, Gessellschaft fur Wissenschaftliche Datenverarbeitung, Gottingen, 1999, 57-71.

[12] E.H. Doha, A.H. Bhrawy, S.S. Ezz-Eldien, A Chebyshev spectral method based on operational matrix for initial and boundary value problems of fractional order, Comput. Math. Appl. 62(2011), 2364-2373.

[13] E.H. Doha, A.H. Bhrawy, S.S. Ezz-Eldien, Efficient Chebyshev spectral methods for solving multi-term fractional orders differential equations, Appl. Math. Model. 35(2011), 5662-5672.

[14] F. Farokhi, M. Haeri, M. S. Tavazoei, Comparing Numerical Methods for Solving Nonlinear Fractional Order Differential Equations, in New Trends in Nanotechnology and Fractional Calculus Applications, D. Baleanu, Z. B. Guvenc, J. A. T. Machado (eds), Springer 2010, 171-179.

[15] R. Garrappa, Predictor-corrector PECE method for fractional differential equations, 2012, available from http://www.mathworks.com/matlabcentral/fileexchange/32918

[16] F. Ghoreishi, S. Yazdani, An extension of the spectral Tau method for numerical solution of multi-order fractional differential equations with convergence analysis, Comput. Math. Appl. 61(2011), 30-43.

[17] R. Gorenflo, F. Mainardi, Fractional calculus, integral and differential equations of fractional order, in A. Carpinteri and F. Mainardi (Editors), Fractals and Fractional Calculus in Continuum Mechanics, Springer Verlag, Wien (1997), 223-276.

[18] T. Hasegawaa, H. Sugiura, An approximation method for high-orede fractional derivatives of algebraically singular functions, Comput. Math. Appl. 62 (2011), 930-937.

[19] Z. Jiao, Y. Chen, I. Podlubny, Distributed-Order Dynamic Systems: Stability, Simulation, Applications and Perspectives, Springer, 2012.

[20] E. L. Ortiz, H. Samara, An Operational Approach to the Tau Method for the Numerical Solution of Non-Linear Differential Equations, Computing, 27(1981), 15-25.

[21] A. Pedas, E. Tamme, Numerical solution of nonlinear fractional differential equations by spline collocation methods, J. Comput. Appl. Math. 255(2014), 216-230.

[22] I. Petrás, Fractional Derivatives, Fractional Integrals, and Fractional Differential Equations in Matlab, in Engineering Education and Research Using MATLAB, Dr. Ali Assi (Ed.), InTech 2011, Available from: http://www.intechopen.com/books/engineering-education-and-research-usingmatlab/fractional-derivatives-fractional-integrals-and-fractional-differential-

equations-in-matlab

[23] I. Podlubny, Fractional Differential Equations, Academic Press, 1999.

[24] I. Podlubny, Matrix Approach to Discrete Fractional Calculus, Fract. Calc. Appl. Anal. 3(4)(2000), 359-386.

[25] I. Podlubny, Matrix approach to discretization of ODEs and PDEs of arbitrary real order, 2009, available from http://www.mathworks.com/matlabcentral/fileexchange/22071

[26] I. Podlubny, Matrix approach to distributed-order ODEs and PDEs, 2012, available from http://www.mathworks.com/matlabcentral/fileexchange/36570

[27] E.A. Rawashdeh, Numerical solution of fractional integro-differential equations by collocation method, Appl. Math. Comput. 176(2006), 1-6.

[28] A. Saadatmandi, M. Dehghan, A new operational matrix for solving fractional-order differential equations, Comput. Math. Appl. 59(2010), 1326-1336.

[29] H. Sugiura, T. Hasegawaa, Quadrature rule foe Abel's equations: Uniformly approximating fractional derivatives, J. Comput. Appl. Math. 223(2009), 459-468.

[30] L. N. Trefethen and others, Chebfun Version 4.2, The Chebfun Development Team, 2011, available from http://www.chebfun.org/

[31] D. Trif, Matrix based operatorial approach to differential and integral problems, in MATLAB: A Ubiquitous Tool for the Practical Engineer, C.M. Ionescu (Ed.), InTech, Rijeka, 2011, available from http://www.intechopen.com/articles/show/title/matrix-based-operatorial-approach-to-differential-and-integral-problems.

[32] D. Trif, Chebpack, 2011, available from http://www.mathworks.com/matlabcentral/fileexchange/32227

[33] D. Valerio et al, Fractional calculus: A survey of useful formulas, Eur. Phys. J. Special Topics, 222(8)(2013), 1827-1846.

[34] W. K. Zahra, S. M. Elkholy, Cubic Spline Solution of Fractional Bagley-Torvik Equation, EJMAA, 1(2)(2013), 230-241, available from http://ejmaa.6te.net/