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.

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 \texttt{Chebpack} [32] for fractional calculus problems. \texttt{Chebpack} is based on the operational form of the Chebyshev spectral tau method [20], [8], [7]. The main advantage of \texttt{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 \texttt{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 \texttt{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 \texttt{Chebpack} [32], in the folder \texttt{Examples}, subfolder \texttt{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

$$y(x) \approx \frac{1}{2} c_0 T_0(x) + c_1 T_1(x) + \cdots + c_{n-1} T_{n-1}(x), \quad x \in [-1, 1].$$

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

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 $\mathcal{L}(y)(x) = f(x)$ becomes, after the discretization, a pure algebraic linear problem in an operational form $L \cdot c = f$ 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 $y(x_1), y(x_2), \ldots, y(x_n)$ of values of $y$ at the given gridpoints $x$, for example at the Chebyshev points of the first or of the second kind

$$x_k^{(1)} = -\cos\left(\frac{(2k - 1)\pi}{2n}\right), \quad x_k^{(2)} = -\cos\left(\frac{(k - 1)\pi}{n - 1}\right), \quad k = 1, \ldots, n$$

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

$$y_{n-1}(x) = \frac{c_0}{2} T_0(x) + c_1 T_1(x) + \cdots + c_{n-1} T_{n-1}(x)$$
is the unique polynomial obtained by interpolating \( y(x) \) through the points \( x_1, \ldots, x_n \).

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

\[
T^*_k(x) = T_k(\alpha x + \beta), \quad x \in [a, b], \quad \alpha = \frac{2}{b-a}, \quad \beta = \frac{-b+a}{b-a}, \quad k = 0, \ldots, n-1.
\]

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

\[
\int_a^b y(x) dx \approx \sum_{j=1}^n w_j y(x_j) \equiv w \cdot y(x).
\]

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

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

\[
T_c = [T_0/2, T_1(\xi), \ldots, T_{n-1}(\xi)], \quad \xi = \frac{2x_c}{b-a} - \frac{b+a}{b-a} \in [-1, 1]
\]

and \( \xi \) is transposed as a column vector. The code is based on the recurrence relations

\[
T_1(x) = xT_0, \quad T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad k = 1, 2, \ldots
\]

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

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 \( X = \text{mult}(n, \text{dom}) \) also based on the recurrence relations (2.5). If \( c \) is the column of Chebyshev coefficients of a function \( y(x) \), then \( X \cdot c \) is the column of the Chebyshev coefficients of the multiplication by the independent variable \( xy(x) \) for \( x \in \text{dom} \). 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 \( A \cdot c \) where the multiplication matrix \( A \) is given by the code \( A=\text{fact}(a,m) \) based on the relations

\[
2T_j(x)T_k(x) = T_{j+k}(x) + T_{|k-j|}(x), \quad j, k = 0, 1, \ldots
\]

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

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

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

All the above codes take into account a general \( \text{dom} \), 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 \( q \)

\[
J^q y(x) = \frac{1}{\Gamma(q)} \int_0^x \frac{y(t)dt}{(x-t)^{1-q}}
\]

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

\[
y(x) \simeq \sum_{k=0}^{n-1} c_k T_k(\alpha x - 1),
\]

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

\[
J^q y(x_j) \simeq \frac{1}{\Gamma(q)} \sum_{k=0}^{n-1} c_k \int_0^{x_j} \frac{T_k(\alpha t - 1) dt}{(x_j-t)^{1-q}}, \quad j = 1, \ldots, n
\]

approximates the physical values \( v \) of \( J^q y(x) \) at the Chebyshev points \( x = (x_j)_{j=1,\ldots,n} \). Consequently, the spectral approximation of \( J^q y(x) \) is given by the Chebyshev coefficients \( T^{-1}v \equiv I \cdot c \), where \( T \) is given by (2.4), \( c \) is the column vector of the coefficients \( c_0, \ldots, c_{n-1} \) and \( I \) is the
**basic integration matrix.** If \( \text{dom} = [0, b] \) and \( x \) is given by \( x = \text{pd}(n, \text{dom}, \text{kind}) \), then the code

\[
I = \text{fracbas}(x, \text{dom}, q)
\]

for \( 0 < q < 1 \) calculates the above matrix \( I \).

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

\[
I_{jk} = \int_0^{x_j} \frac{T_k(\alpha t - 1) dt}{(x_j - t)^{1-q}}.
\]

We have

\[
\int_0^x \frac{dt}{(x-t)^{1-q}} = \frac{x^q}{q}, \quad \int_0^x \frac{tdt}{(x-t)^{1-q}} = \frac{x^{q+1}}{q(q+1)},
\]

\[
\int_0^x \frac{t^2 dt}{(x-t)^{1-q}} = \frac{2x^{q+2}}{q(q+1)(q+2)}.
\]

Consequently, the columns \( I_{.,k} \) for \( k = 0, 1, 2 \) are

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

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

The columns for \( k > 2 \) are calculated iteratively, by using the recurrence formulas

\[
T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad T_k(x) = \frac{T_{k+1}(x)}{2(k+1)} - \frac{T_{k-1}(x)}{2(k-1)}, \quad -1 \leq x \leq 1.
\]

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

\[
I_{.,k} = \int_0^x \frac{T_k(\alpha t - 1) dt}{(x-t)^{1-q}} = \int_0^x \frac{2(\alpha t - 1) T_{k-1}(\alpha t - 1) - T_{k-2}(\alpha t - 1)}{(x-t)^{1-q}} dt =
\]

\[
= 2\alpha \int_0^x \frac{t T_{k-1}(\alpha t - 1) dt}{(x-t)^{1-q}} - 2I_{.,k-1} - I_{.,k-2} =
\]

\[
= -2\alpha \int_0^x \frac{T_{k-1}(\alpha t - 1) dt}{(x-t)^{1-q}} + 2\alpha x I_{.,k-1} - 2I_{.,k-1} - I_{.,k-2}.
\]
But

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

and, consequently,

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

\[ I_{k-1} \left( 1 + \frac{q}{k} \right) = 2(\alpha x - 1) I_{k-1} + \left( \frac{q}{k-2} - 1 \right) I_{k-2} - \frac{2(-1)^k x^q}{k(k-2)}. \]

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 \( c = (c_j)_{j=0,...,n-1} \), then the vector \( I \cdot c \) will contain the spectral coefficients of \( J^q y(x) \), for \( q \in (0, 1) \).

The function \texttt{fracbas} can be tested on the well known example

\[
J^{1/2} e^{x} \equiv \frac{1}{\sqrt{\pi}} \int_0^x \frac{e^t}{\sqrt{x-t}} dt = e^{x} \text{erf}(\sqrt{x}),
\]

for \( x > 0 \), say for \( x \in [0,2] \). The code \texttt{test1.m}

```matlab
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 \( \text{reznum} \) which reproduces the exact result at the points \( x \) with an error of \( 2.65 \times 10^{-15} \) 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 \( y_{\text{ex}}(x) = 1 - e^{\pi x} \text{erfc}(\sqrt{\pi} x) \). 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

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

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

```matlab
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 \( 1.7543 \times 10^{-4} \) and needs a computing time of \( 6 \times 10^{-3} \) seconds. For \( n = 512 \) the error becomes \( 2.7658 \times 10^{-6} \) 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 \( \text{erfc}(\sqrt{x}) \); 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 $6.4887 \times 10^{-8}$ for almost the same computing time.

If $q = 1$ the code $[-, J0] = \text{prim}(n, \text{dom})$ 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 $\text{fracbas}$ for $0 < q < 1$
- the matrix $J_0$ given by $\text{prim}$ for $q = 1$
- the matrix $I^{[q-[q]]} \cdot J_0^{[q]}$ where $[q]$ is the largest integer not larger than $q$ ($\text{floor}(q)$ in Matlab), $J_0$ is given by $\text{prim}$ and $I$ is given by $\text{fracbas}$ for $q - [q]$. The above formulas are implemented in an extended code $Jrl = \text{primrl}(x, \text{dom}, q)$.

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

The Riemann-Liouville derivative of order $q$ is

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

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

The Caputo derivative of order $q$ is

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

where $q$ and $m$ are as above. In the operatorial form, $D^q y = J^{m-q} \cdot D^m y$. The matrix $CD^q$ is discretized by the code $Dc = \text{derivc}(x, \text{dom}, q)$.
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 $D^{0.5} \sin x$**

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
<th>Time (sec)</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact</td>
<td>0</td>
<td>4.45</td>
<td>-</td>
</tr>
<tr>
<td>Chebpack</td>
<td>1.09e-14</td>
<td>0.005</td>
<td>64</td>
</tr>
<tr>
<td>Chebfun</td>
<td>3.58e-13</td>
<td>0.59</td>
<td>20</td>
</tr>
<tr>
<td>Podlubny</td>
<td>0.05</td>
<td>0.02</td>
<td>629</td>
</tr>
</tbody>
</table>

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

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

The equation is discretized as a linear system $(D_c + E)c = b$, where $c$ are the Chebyshev coefficients of the solution $y$ and $b$ 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 $T = \text{cpv}(n, 0, [0, 1])$. Precisely, $T \cdot c = y(0) = 1$ and $T \cdot D \cdot c = y'(0) = -1$. The Matlab code is test4.m

```matlab
[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 $\text{solnum}$ for $n = 64$ is $1.63 \times 10^{-7}$ 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 $9.24 \times 10^{-6}$ 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]

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

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

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

\begin{align*}
y_1^{\left(\frac{1}{2}\right)} &= y_2, \\
y_2^{\left(\frac{1}{2}\right)} &= y_3, \\
y_3^{\left(\frac{1}{2}\right)} &= y_4, \\
y_4^{\left(\frac{1}{2}\right)} &= F - y_1 - y_4,
\end{align*}

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

\begin{align*}
y'' + y^{\left(\frac{1}{2}\right)} + y = 2 + 4 \sqrt{\frac{x}{\pi}} + x^2, \\
x \in [0, 10]
\end{align*}

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

with the exact solution $y_{ex}(x) = x^2$. The results are given by \texttt{example1bis.m} in Table 3.
Example 2. Let us consider the test boundary value problem from [34]

\[ y''(x) + 0.5 D_{1.5} y(x) + y(x) = 3 + x^2 \left( \frac{x^{-q}}{\Gamma(3-q)} + 1 \right), \quad x \in [0,1], \quad q = 0.5 \]

\[ y(0) = 1, \quad y(1) = 2. \]

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 \( 2 \times 10^{-16} \). 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 \( 4.35 \times 10^{-3} \).

A more relevant example is the eigenvalue problem [1]

\[ C D_{1.5} y(x) + \left( \frac{1}{x} + \lambda \right) y(x) = 0, \quad x \in (0,1), \]

\[ y(0) = 0, \quad y'(1) = 0. \]

This problem is discretized as \( A c = -\lambda B c \) where \( A = D_c + X^{-1} \), \( B \) is the unit matrix of order \( n \) and \( c \) 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 \( \text{eig} \) from Matlab.

The Matlab code is example2bis.m

\begin{verbatim}
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);
\end{verbatim}

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

<table>
<thead>
<tr>
<th>λ (Chebpack)</th>
<th>λ [1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6610</td>
<td>1.6609</td>
</tr>
<tr>
<td>13.5509</td>
<td>13.5504</td>
</tr>
<tr>
<td>20.5147</td>
<td>20.5145</td>
</tr>
</tbody>
</table>

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

\[ cD^{\frac{3}{4}}y(x) + \frac{x^2 e^x}{5} y(x) - e^x \int_0^x ty(t)dt = \frac{6x^2}{\Gamma\left(\frac{13}{4}\right)}, \quad x \in [0, 1], \]

\[ y(0) + y(1) - \frac{e + 1}{e - 2} y'(0) + \frac{1}{6} y''(1) - 10 \int_0^1 ty(t)dt = 0. \]

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 \( T = \text{cpv}(n, [0, 1], [0, 1]) \) and \( e \) are the Chebyshev coefficients of the solution \( y \), then \( J_0 * X * c \) approximates the coefficients of the primitive \( \int_0^x ty(t)dt \) and \( (T(2,:) - T(1,:)) * J_0 * X * c \) implements the Newton-Leibniz formula to approximate \( \int_0^1 ty(t)dt \). The Matlab code is \texttt{example3.m} and for \( n = 64 \) the computing time is 0.01 sec. while the error is about \( 2.45 \times 10^{-12} \).

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

\[ y''(x) + D^{\varphi(\alpha)}y(x) + y(x) = \begin{cases} 8, & 0 \leq x \leq 1 \\ 0, & x > 1 \end{cases}, \quad x \in [0, 30], \]

\[ y(0) = y'(0) = 0, \quad \varphi(\alpha) = 6\alpha(1 - \alpha), \quad \alpha \in [0, 1]. \]

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

\[ D^{\varphi(\alpha)} = \sum_{k=1}^m w_k \cdot \varphi(\alpha_k) cD^{\alpha_k} \]
where the nodes $\alpha_k$ and the weights $w_k$ are given by $[\text{alpha}, w] = \text{pd}(m, [0, 1], 1)$ 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 $\text{dom} = [0, 30]$ 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](image)

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

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

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

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

This problem is of the form $Ly = N(x, y)$. After discretization, the discrete problem becomes $\mathcal{L}c = \mathcal{N}(c)$ where $c$ are the Chebyshev coefficients of $y$, $\mathcal{L}$ is the Caputo differentiation
matrix of order $1.2$ and $\mathcal{N}$ is the nonlinear right-hand side function of the coefficients $c$. Here the coefficients $C$ of the nonlinear part $y^4$ are calculated by using the scheme

$$c \xrightarrow{t^2x} y(x) \xrightarrow{} y^4(x) \xrightarrow{x^2t} C.$$  

Finally, after imposing the initial conditions in the last two rows of the matrix $\mathcal{L}$ and of the vector $\mathcal{N}(c)$, we obtain a nonlinear system $c = \mathcal{L}^{-1}\mathcal{N}(c)$. This system can be solved in this case by successive iterations started by the Chebyshev coefficients of the constant function $y_0(x) = 1$, 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>
<thead>
<tr>
<th>$n$</th>
<th>Error</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1.1e-5</td>
<td>0.013</td>
</tr>
<tr>
<td>64</td>
<td>1.1e-6</td>
<td>0.022</td>
</tr>
<tr>
<td>128</td>
<td>1.2e-7</td>
<td>0.08</td>
</tr>
<tr>
<td>256</td>
<td>1.3e-8</td>
<td>0.45</td>
</tr>
<tr>
<td>512</td>
<td>1.4e-9</td>
<td>3.86</td>
</tr>
</tbody>
</table>

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


