SPSMAT Documentation

Sobhan Latifi [1] and Yunpeng Zhang [2*]

Abstract
SPSMAT is an add-on for Octave/MATLAB, that helps you solve non-fractional/fractional ordinary/partial differential/integral equations. This documentation file is a guidance that wants to walk you trough using SPSMAT package. This file is a supplementary for its relevant journal paper. In the main paper, i.e the journal paper, all the notions are discussed completely.
Keywords: Spectral methods , PseudoSpectral methods , Matrix computation.

Keywords
Spectral methods — PseudoSpectral methods — Matrix computation

[1] *Shahid Beheshti university of Tehran*, Sobhanlatifi89@gmail.com
[*2] *University o Houston*, yzhang119@uh.edu

Contents

1. Introduction

Over the recent years working with spectral and pseudospectral methods, when it comes to solving differential/integral equations, we have found out that there is a dearth of a suitable package in terms of covering spectral and pseudospectral method helping us solve differential/integral equations. For that, it has been attempted to provide a consistent and user-friendly high-level functions that allow experimental scientists work properly and beneficially on their equations.
Developers can easily extend the functionality and implement new algorithms due to the modular design functionalities of this package namely SPSMAT.
The open source nature of SPSMAT help us claim that this package is portable and open source. The accompanying source code and the documentation of the toolbox has been enriched with numerous comments, examples, and detailed instructions for extensions.
Some of the major benefit of using this package are:

- This package is highly modular.
- you can combine the approximation techniques.
- SPSMAT simplifies the maintenance of the code due to its modular design.
- It is free and open source.
- It is such a multi-platform toolbox.

In addition to this, in this version, also for the simplicity's sake, we cover Jacobi polynomials which are a wide-ranging polynomials encompassing Gegenbauer (ultrasperical), Chebyshev and Legendre polynomials. But, for those who are not into these polynomials, we also offered the derived subcategories including Gegenbauer, Chebyshev (all four kinds) and Legendre polynomials.

```
>> help jacobi_zeros
'jacobi_zeros' is a function from the file /SPSMAT
 /jacobi_zeros.m


Overview
jacobi_zeros( N, alpha_, beta_ ) is a function returing N
roots of the N-th sentence of  Standard orthogonal Jacobi
polynomials.


inputs:
-----------------------------------------------------
| N     : integer          : Jacobi function paramter|
| alpha_: double           : Jacobi parameter        |
| beta_ : double           : Jacobi parameter        |
-----------------------------------------------------

Output:
-----------------------------------------------------
| out   : [Nx1] double : Jacobi zeros                |
-----------------------------------------------------
```

Figure 1. Help command result.

SPSMAT, lo and behold, was created to address some of the issues that other works lacked: simplicity, and covering spectral and pseudospectral methods.

Undergraduate students taking a course in Scientific computing and any relevant course can find this package as a test machine to check different things with orthogonal polynomials and develop their work with this package: this is your lightweight companion. In nutshell, whoever you are, we aim to increase your appreciation of this fundamental subject.

2. module Details

MATLAB/Octave modules come in the form of what are called M-files, files with the generic name *module()*.m or *module*.m, where module is the module name. SPSMAT consists of the following modules:

help

For any help you need, you can type the name of the function (without parentheses in front): for instance if you need help for function *jacobi_zeros*() you must receive a result like in Fig. 1:

Jacobi Polynomials

Some features of Jacobi polynomials are defined on $[-1,1]$ and are of high interest in different research studies [8, 9, 11]. Their recurrence relation is

$$J_{k+1}^{\alpha,\beta}(x) = (a_k^{\alpha,\beta}x - b_k^{\alpha,\beta})J_k^{\alpha,\beta} - c_k^{\alpha,\beta}J_{k-1}^{\alpha,\beta}(x),\ k \geq 1$$

$$J_0^{\alpha,\beta}(x) = 1,\ J_1^{\alpha,\beta}(x) = \frac{1}{2}(\alpha+\beta+2)x + \frac{1}{2}(\alpha-\beta),$$

where

$$a_k^{\alpha,\beta} = \frac{(2k+\alpha+\beta+1)(2k+\alpha+\beta+2)}{2(k+1)(k+\alpha+\beta+1)},$$

$$b_k^{\alpha,\beta} = \frac{(\beta^2-\alpha^2)(2k+\alpha+\beta+1)}{2(k+1)(k+\alpha+\beta+1)(2k+\alpha+\beta)},$$

$$c_k^{\alpha,\beta} = \frac{(k+\beta)(k+\alpha)(2k+\alpha+\beta+2)}{(k+1)(k+\alpha+\beta+1)(2k+\alpha+\beta)},$$

The Jacobi polynomials are satisfying the following identities:

$$J_n^{\alpha,\beta}(-x) = (-1)^n J_n^{\beta,\alpha}(x),$$
$$J_n^{\alpha,\beta}(-1) = \frac{(-1)^n\Gamma(n+\beta+1)}{n!\Gamma(\beta+1)},$$
$$J_n^{\alpha,\beta}(1) = \frac{\Gamma(n+\alpha+1)}{n!\Gamma(\alpha+1)},$$
$$\left(J_n^{\alpha,\beta}(x)\right)^{(m)} = 2^{-m}\frac{\Gamma(m+n+\alpha+\beta+1)}{\Gamma(n+\alpha+\beta+1)}J_{n-m}^{\alpha+m,\beta+m}(x).$$

and its relevant weight function is defined as $w^{\alpha,\beta}(x) = (1-x)^\alpha(1+x)^\beta$.
The orthogonal property of these polynomials are shown

$$\int_{-1}^{1} J_n^{\alpha,\beta}(x)J_m^{\alpha,\beta}(x)w^{\alpha,\beta}(x) = \delta_{m,n}\gamma_n^{\alpha,\beta},$$

$$\gamma_n^{\alpha,\beta} = \frac{2^{\alpha+\beta+1}\Gamma(n+\alpha+1)\Gamma(n+\beta+1)}{(2n+\alpha+\beta+1)\Gamma(n+1)\Gamma(n+\alpha+\beta+1)},$$

Another form of showing these polynomilas in $[-1,1]$ is

$$J_i^{\alpha,\beta}(x) = \sum_{k=0}^{i}\frac{(-1)^{(i-k)}\Gamma(i+\beta+1)\Gamma(i+k+\alpha+\beta+1)}{\Gamma(k+\beta+1)\Gamma(i+\alpha+\beta+1)(i-k)!k!}\frac{(x+1)^k}{2^k}, \quad (1)$$

and the shifted ones, by shifting $z = 2\frac{x-a}{b-a} - 1$, over $[a,b]$ are

$$J_i^{\alpha,\beta}(z) = \sum_{k=0}^{i}\frac{(-1)^{(i-k)}\Gamma(i+\beta+1)\Gamma(i+k+\alpha+\beta+1)}{\Gamma(k+\beta+1)\Gamma(i+\alpha+\beta+1)(i-k)!k!}\frac{(x-a)^k}{(b-a)^k}, \quad (2)$$

in other words

$$J_i^{\alpha,\beta}(z) = \sum_{k=0}^{i}\sum_{j=0}^{k}\binom{k}{j}\frac{(-1)^{(i-j)}\Gamma(i+\beta+1)\Gamma(i+k+\alpha+\beta+1)}{\Gamma(k+\beta+1)\Gamma(i+\alpha+\beta+1)(i-k)!k!}\frac{a^{k-j}x^j}{(b-a)^k}, \quad (3)$$

in short it is rewritten as

$$J_i^{\alpha,\beta}(z) = \sum_{k=0}^{i}\sum_{j=0}^{k}R(i,j,k,a,b)x^j. \quad (4)$$

jacobi_zeros()

jacobi_zeros$(N, alpha\_, beta\_)$ is a function returning N roots of the N-th sentence of standard orthogonal Jacobi polynomials. alpha_ and beta_ are the two parameters of Jacobi polynomials. As Jacobi polynomials are orthogonal, their roots are found by achieving the eigenvalues of the three-diagonal a matrix like bellow (theorem 3.6.3 in [12]):

$$K_n := \begin{bmatrix} \delta_1 & \gamma_2 & \cdots & \cdots & & & \\ \gamma_2 & \delta_2 & \gamma_3 & & & & \\ & \gamma_3 & \delta_3 & \gamma_4 & & & \\ \vdots & & & & & & \\ \vdots & & & & & & \\ & & & & \gamma_{n-1} & \delta_{n-1} & \gamma_n \\ & & & & & \gamma_n & \delta_n \end{bmatrix},$$

where

$$\delta_{i+1} := \frac{\langle xJ_i^{\alpha,\beta}, J_i^{\alpha,\beta}\rangle}{\langle J_i^{\alpha,\beta}, J_i^{\alpha,\beta}\rangle},\ i \geq 0,$$

$$\gamma_{i+1}^2 := \begin{cases} 0, & i = 0, \\ \frac{\langle J_i^{\alpha,\beta}, J_i^{\alpha,\beta}\rangle}{\langle J_{i-1}^{\alpha,\beta}, J_{i-1}^{\alpha,\beta}\rangle}, & i \geq 1. \end{cases}$$

Declaration
jacobi_zeros(N, alpha_, beta_):
Input:
N: integer
alpha_, beta_ : double
Output:
class: double
Dimension:[Nx1]

Code execution
>>> jacobi_zeros(4, 0, 1.5)
ans=

-6.543493e-01
-9.659864e-02
4.874293e-01
8.950976e-01

As might be expected these zeros are located in $[-1,1]$. If you are to shift these zeros into $[0,1]$ the code execution must be:

Code execution
>>> (1+jacobi_zeros(4, 0, 1.5))./2
ans=

0.1728253
0.4517007
0.7437146
0.9475488

jacobi_w()

This subsection intended for Jacobi-Gauss-type quadratures. We recall that in the Jacobi case, the quadrature formula reads [14]

$$\int_{-1}^{1} p(x)w^{\alpha,\beta}(x) = \sum_{k=0}^{N} p(x_k)w_k + E_n[p],$$

where x_k are the roots of Jacobi standard polynomials, w_k are the corresponding weights and $E_N[p]$ is the error term discussed in [14]; this formula is proposed for interval $[-1,1]$.

In the same fashion, in case of interval $[a,b]$, that is, $u(x) = 2\frac{x-a}{b-a} - 1$ we need to give

$$\int_a^b p(x)w^{\alpha,\beta}(u(x)) = \sum_{k=0}^{N} p(\frac{b-a}{2}x_k + \frac{b+a}{2})(\frac{b-a}{2})w_{k+\hat{E}_n[p]}.$$

This formula is exact, in terms of Jacobi gauss quadratures, for $p \in P_{2N+1}$. It goes without saying that the nodes and weights are computed with this algorithm:

$x_k,\ k = 0...N$ are the zeros of $J_{N+1}^{\alpha,\beta}(x)$ for $x \in [-1,1]$, and

$$w_k = \frac{G_N^{\alpha,\beta}}{(1-x_k^2)\left(\partial_x J_{N+1}^{\alpha,\beta}(x_k)\right)^2},$$

where

$$G_N^{\alpha,\beta} = \frac{2^{\alpha+\beta+1}\Gamma(N+\alpha+2)\Gamma(N+\beta+2)}{(N+1)!\Gamma(N+\alpha+\beta+2)}.$$

Also for Jacobi-Gauss-Radau quadratures, the formula is exact for $p \in P_{2N}$ with nodes and weights as:

$x_0 = -1,\ x_k,\ k = 1...N$ are the zeros of $J_N^{\alpha,\beta+1}(x)$ for $x \in [-1,1]$,

$$w_0 = \frac{2^{\alpha+\beta+1}\Gamma(N+\alpha+1)(\beta+1)\Gamma^2(\beta+1)N!}{\Gamma(N+\beta+2)\Gamma(N+\alpha+\beta+2)},$$

$$w_k = \frac{G_{N-1}^{\alpha,\beta+1}}{(1-x_k)(1+x_k)^2\left(\partial_x J_N^{\alpha,\beta+1}(x_k)\right)^2},\ k = 1...N.$$

In addition to these, for Jacobi-Gauss-Lobatto quadratures, the formula is exact for $p \in P_{2N-1}$ with the following nodes and weights:

$x_0 = -1$, $x_{N+1} = 1$ and $x_k,\ k = 1...N-1$ are the zeros of $\partial_x J_N^{\alpha,\beta}(x)$ for $x \in [-1,1]$, and also

$$w_0 = \frac{2^{\alpha+\beta+1}\Gamma(N+\alpha+1)(\beta+1)\Gamma^2(\beta+1)\Gamma(N)}{\Gamma(N+\beta+1)\Gamma(N+\alpha+\beta+2)},$$

$$w_N = \frac{2^{\alpha+\beta+1}\Gamma(N+\beta+1)(\alpha+1)\Gamma^2(\alpha+1)\Gamma(N)}{\Gamma(N+\alpha+1)\Gamma(N+\alpha+\beta+2)},$$

$$w_k = \frac{G_{N-2}^{\alpha+1,\beta+1}}{(1-x_k{}^2)^2\left(\partial_x J_{N-1}^{\alpha+1,\beta+1}(x_k)\right)^2},\ k = 1...N-1.$$

Declaration
jacobi_w(N, alpha_, beta_, type):
Input:
N: integer
alpha_, beta_ : double
type: can be each *'gauss'*, *'gauss_rdu'*, or *'gauss_lbt'*
Output:
class: double
Dimension:[1x(N+1)]

```
function Exp0

alpha_=1;
beta_=3;
N=4;
a=0;
b=1;

X=[-1,jacobi_zeros(N-1,alpha_+1,beta_+1)',1];
r=f(((b-a)/2).*X+(b+a)/2);
w=((b-a)/2)*jacobi_w(N,alpha_,beta_,'gauss_lbt');
r*w'
end


function out=f(X)
  for i=1:length(X)
    out(i)=3*X(i)**3+X(i)**2
  end%for
end
```

Figure 2. Running Code for Example 0

Code execution

$>>> jacobi\_w(4,1,3,'gauss')$

ans =

2.456646e-02 2.225593e-01 5.765700e-01 5.871497e-01 1.891545e-01

$>>> jacobi\_w(4,1,3,'gauss\_rdu')$

ans =

2.834467e-05 1.134519e-01 5.258315e-01 7.011261e-01 2.586834e-01

$>>> jacobi\_w(4,1,3,'gauss\_lbt')$

ans =

1.814059e-03 1.948332e-01 7.272936e-01 6.316148e-01 4.444444e-02

Example 0: For instance, for calculation of $\int_0^1 (3x^3+x^2)W(u(x))$, where $u(x)=2x-1$, we run the code in Fig. 2: In this piece of code, we used Jacobi-Gauss-Lobatto quadratures that results in 1.238095; this value is somehow exact.

jacobi_()

If $J_N^{\alpha,\beta}(x)$ be the N-th Jacobi polynomial of order N, for having the shifted form of this polynomial from $[-1,1]$ into the interval $[a,b]$ we need a shifting parameter like u. If we define $u(x)=\frac{2x-(a+b)}{b-a}$, $J_N^{\alpha,\beta}\left(u(x)\right)$ is promising for our goal. When z is a scalar, the function $jacobi\_(N,\alpha,\beta,k,z,u)$ is giving:

$$\left[\frac{d^k J_0^{\alpha,\beta}(u(z))}{dx^k} \quad \frac{d^k J_1^{\alpha,\beta}(u(z))}{dx^k} \quad \frac{d^k J_2^{\alpha,\beta}(u(z))}{dx^k} \quad \dots \quad \frac{d^k J_N^{\alpha,\beta}(u(z))}{dx^k}\right].$$

In today coding world and conventions, "vectorization" is of high interest. This simply means that if you write a function operating on one input, it would be much more handy if it can handle a bunch of inputs at once. This happens where you send a vector of inputs to a function. For our case, if z be a vector our function must be written and developed in order to handle a vector of inputs as z. So, for z as a vector $z=[z_0,z_1,z_2,...,z_m]$, $jacobi\_(N,a,b,k,z,u)$ will return

$$\begin{bmatrix} \frac{d^k J_0^{\alpha,\beta}(u(z_0))}{dx^k} & \frac{d^k J_1^{\alpha,\beta}(u(z_0))}{dx^k} & \frac{d^k J_2^{\alpha,\beta}(u(z_0))}{dx^k} & \dots & \frac{d^k J_N^{\alpha,\beta}(u(z_0))}{dx^k} \\ \frac{d^k J_0^{\alpha,\beta}(u(z_1))}{dx^k} & \frac{d^k J_1^{\alpha,\beta}(u(z_1))}{dx^k} & \frac{d^k J_2^{\alpha,\beta}(u(z_1))}{dx^k} & \dots & \frac{d^k J_N^{\alpha,\beta}(u(z_1))}{dx^k} \\ \vdots & & & & \\ \frac{d^k J_0^{\alpha,\beta}(u(z_m))}{dx^k} & \frac{d^k J_1^{\alpha,\beta}(u(z_m))}{dx^k} & \frac{d^k J_2^{\alpha,\beta}(u(z_m))}{dx^k} & \dots & \frac{d^k J_N^{\alpha,\beta}(u(z_m))}{dx^k} \end{bmatrix}.$$

The important fact here is that this matrix does not use derivative at all. It utilizes the benefit of the following relation for computing derivative operation:

$$R_{n+2}^{\alpha,\beta}(u)=(\hat{a}_n^{\alpha,\beta}u-\hat{b}_n^{\alpha,\beta})R_{n+1}^{\alpha,\beta}(u)-\hat{c}_n^{\alpha,\beta}R_n^{\alpha,\beta}(u),\ n\geq 1$$

$$R_0^{\alpha,\beta}(u)=1,\ R_1^{\alpha,\beta}(u)=\frac{1}{2}(\alpha+\beta+2+2k)u+\frac{1}{2}(\alpha-\beta),$$

where

$$\hat{a}_n^{\alpha,\beta}=\frac{(2n+\alpha+\beta+1+2k)(2n+\alpha+\beta+2+2k)}{2(n+1)(n+\alpha+\beta+1+2k)},$$

$$\hat{b}_n^{\alpha,\beta}=\frac{\left((\beta+k)^2-(\alpha+k)^2\right)(2n+\alpha+\beta+1+2k)}{2(n+1)(n+\alpha+\beta+1+2k)(2n+\alpha+\beta)},$$

$$\hat{c}_n^{\alpha,\beta}=\frac{(n+\beta+k)(n+\alpha+k)(2n+\alpha+\beta+2+2k)}{(n+1)(n+\alpha+\beta+1+2k)(2n+\alpha+\beta+2k)},$$

then

$$\frac{\partial^k J_n^{\alpha,\beta}(u)}{\partial x^k}=(u')^k\frac{\Gamma(\alpha+\beta+n+k)}{2^k\Gamma(\alpha+\beta+n)}R_{n-k}^{\alpha,\beta}(u). \qquad (5)$$

Declaration

$jacobi\_(N,alpha\_,beta\_,k,z,u)$:

Input:

N: integer

alpha_, beta_ : double

k (derivative order): integer

z: [double 1xM]

u: (symbolic shifting parameter)

Output:

class: double

Dimension:[Mx(N+1)]

The exact fact that should be noted here is that with only one line of the code one can obtain whatever he needs from this function. : For instance, executing

$$jacobi\_(N,alpha\_,beta\_,0,z,u),$$

will give the Jacobi polynomials matrix for inputs of z, while

$$jacobi\_(N, alpha\_, beta\_, 1, z, u),$$

will give the first order derivative matrix of Jacobi polynomials for inputs z.

Code execution

$>>> jacobi\_(3,0,1,1,[-0.2,0,0.25],@(x)x)$
ans =

0.00000	1.50000	-2.00000	-0.60000
0.00000	1.50000	-1.00000	-1.87500
0.00000	1.50000	0.25000	-1.99218

$>>> jacobi\_(3,0,1,0,[-0.2,0,0.25],@(x)x)$
ans =

1.00000	-0.80000	-0.20000	0.64000
1.00000	-0.50000	-0.50000	0.37500
1.00000	-0.12500	-0.59375	-0.14258

$>>> jacobi\_(3,0,1,1,0.7,@(x)2*x-1)$
ans =

0.00000	3.00000	2.00000	-2.55000

jacobi_frac()

Similar to the previous section, for fractional Jacobi functions, we need to have its matrix for fractional derivatives. Using the shifting variable $u(x)$ in fractional derivatives leads to errors due to the lack of a correct chain rule for fractional derivatives. To avoid this, we need to consider a fixed boundary like $[a,b]$ with a shifting parameter $u(x) = 2\frac{x-a}{b-a} - 1$. This helps us to have a fixed form and present a particular formula for that. When derivate order is integer this function produces a output similar to the function $jacobi\_()$. Be aware, however, as we suggest here, for the mater of the speed of computation, when we aim at an integer derivative, we use $jacobi\_()$. The reason beyond this fact is that for calculation of jacobi_frac() we used the following formula that is not as simple as the function discussed earlier for jacobi_().

$$J_i^{\alpha,\beta}(x) = \sum_{k=0}^{i} \frac{(-1)^{i-k}\Gamma(i+\beta+1)\Gamma(i+\alpha+k+\beta+1)}{\Gamma(k+\beta+1)\Gamma(i+\alpha+\beta+1)(i-k)!k!} \cdot \frac{\sum_{j=0} k\binom{k}{j}(-1)^{k-j}a^{k-j}x^j}{(b-a)^k}, \quad (6)$$

Something that has more products and computations. Obviously, when this combines with other calculations of fractional derivatives, it is not as fast as $jacobi\_()$ function.
Anyway, for this fractional function we use the following formula:

$$\frac{d^\vartheta J_i^{\alpha,\beta}(x)}{dx^\vartheta} = \sum_{k=0}^{i} \frac{(-1)^{i-k}\Gamma(i+\beta+1)\Gamma(i+\alpha+k+\beta+1)}{\Gamma(k+\beta+1)\Gamma(i+\alpha+\beta+1)(i-k)!k!} \cdot \frac{\sum_{j=0} k\binom{k}{j}(-1)^{k-j}a^{k-j}\frac{\Gamma(j+1)}{\Gamma(j+1-\vartheta)}x^{j-\vartheta}}{(b-a)^k}, \quad (7)$$

and recall that:

$$\frac{d^\vartheta x^j}{dx^\vartheta} = \begin{cases} \frac{\Gamma(j+1)}{\Gamma(j+1-\vartheta)}x^{j-\vartheta}, & \vartheta \le j, \\ 0, & O.W. \end{cases}$$

Declaration
$jacobi\_frac(N, alpha\_, beta\_, nu, z, u)$:
Input:
N: integer
alpha_, beta_ : double
nu (derivative order): double
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

Example 1: Consider

$$D^{(\frac{3}{2})}y(x) + D^{(2)}y(x) + y(x) = 1 + x,\ 0 \le x \le 1. \quad (8)$$

with the exact solution $y(x) = 1+x$. As the interval is over $[0,1]$ we have $u(x) = 2x-1$. If we assume

$$y = \phi(x)C^T,\ \ y^{(i)} = \phi^i(x)C^T,$$

$$f(x) = 1 + x,$$

where $\phi_j = J_j^{\alpha,\beta}(u(x))$, $\phi^i(x) = [\phi_0^i(x), ..., \phi_N^i(x)]$, and $C = [c_0, ..., c_N]^T$, and now by collocating N+1 zeros of Jacobi function $J_{N+1}^{\alpha,\beta}(u)$ in

$$\psi^i = \begin{bmatrix} \phi_0^i(x_0) & \phi_1^i(x_0) & \dots & \phi_N^i(x_0) \\ \phi_0^i(x_1) & \phi_1^i(x_1) & \dots & \phi_N^i(x_1) \\ \vdots & \dots & \ddots & \vdots \\ \phi_0^i(x_N) & \phi_1^i(x_N) & \dots & \phi_N^i(x_N) \end{bmatrix},$$

we will obtain $[\psi^{1.5} + \psi^2 + \psi^0]C^T = H^T$,

$$[f(x_0), ..., f(x_N)] = H.$$

The code in Fig. 3 can determine the unknown C^T. The error of this approximation is depicted in Fig. 4.

```
N=10;
alpha_=0;
beta_=1;
u=@(x)2*x-1;

X=(1+jacobi_zeros(N+1,alpha_,beta_))/2;


H=F(X);

D_0  =jacobi_(N,alpha_,beta_,0  ,X,u);
D_3_2=jacobi_frac(N,alpha_,beta_,1.5,X,u);
D_2  =jacobi_(N,alpha_,beta_,2  ,X,u);


A=D_3_2+D_2+D_0;

% Boundary condition y(0)=1, y'(0)=1
A(1,:)=jacobi_(N,alpha_,beta_,0,0,u);
H(1,1)=1;
A(2,:)=jacobi_(N,alpha_,beta_,1,0,u);
H(1,2)=1;

% Ac=H'-->H'=?
c=A\H';
```

Figure 3. Running Code for Example 1

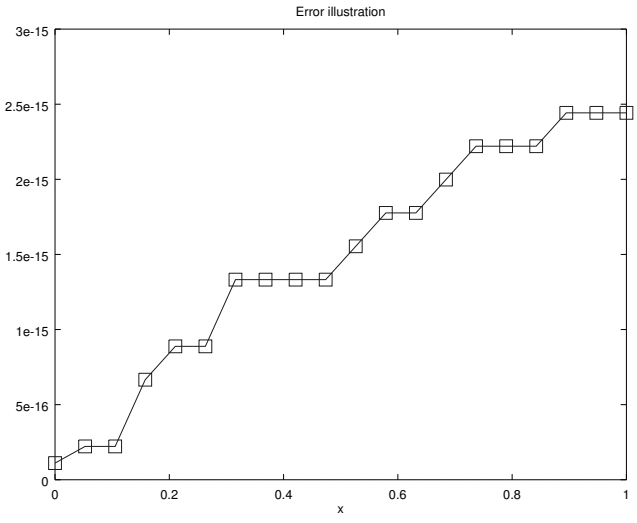

Figure 4. Error illustration of Example1.

D_jacobi()

In case of working with derivative operational matrices we offer a function providing operational matrix. As this operational matrix philosophy goes [15]: $y^{(i)}(x) = \phi^T D^{(i)} C$, where $C = [c_0, ..., c_N]^T, \phi(x) = [J_0^{\alpha,\beta}, ..., J_N^{\alpha,\beta}]$,

$$D = \begin{bmatrix} 0 & E^{-1} \\ 0 & 0 \end{bmatrix}_{(N+1)\times(N+1)},$$

$$E_{pq} = \begin{cases} \frac{-2(p+\alpha+1)(p+\beta+1)}{(\alpha+\beta+p+1)(\alpha+\beta+2p+2)(\alpha+\beta+2p+3)}, & p = q-2, \\ \frac{2(\alpha+\beta+p)}{(\alpha+\beta+2p-1)(\alpha+\beta+2p)}, & p = q, \\ \frac{2(\alpha-\beta)}{(\alpha+\beta+2p)(\alpha+\beta+2p)}, & p = q-1, \\ 0 & o.w. \end{cases}$$

$$D^{(1)} = D,\ D^{(i)} = (D^{(1)})^i$$

The fact is that this formula is set for $u(x) = 2x-1$ i.e. for interval $[0,1]$; so, for $[a,b]$ this matrix must be updated as $D = \frac{2}{b-a} \times D$.

Declaration
$D\_jacobi(N, alpha\_, beta\_, u\_)$:
Input:
N: integer
alpha_, beta_ : double
u_: (symbolic parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

Example 2 consider $2y''(x) - y(x) = 4 - x^2$ with the exact solution $y(x) = x^2$.
By setting $y(x) = \phi(x)^T C$ while $\phi(x) = [J_0^{\alpha,\beta}(u), ..., J_N^{\alpha,\beta}(u)]^T$, $C = [c_0, ..., c_N]^T$. Then we have

$$2\phi^T(x) D^{(2)} C - \phi^T(x) C = \phi^T(x) H,$$

$$[2D^{(2)} - I]C = H,$$

in which H^T is reachable as

$$\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}^T \cdot \begin{bmatrix} \phi_0(x_0) & \phi_0(x_1) & ... & \phi_0(x_N) \\ \phi_1(x_0) & \phi_1(x_1) & ... & \phi_1(x_N) \\ \vdots & & & \\ \phi_N(x_0) & \phi_N(x_1) & ... & \phi_N(x_N) \end{bmatrix}^{-1} = H^T.$$

and $f(x) = 4 - x^2$. The $\{x_i\}_{i=0}^N$ are the roots of $J_{N+1}^{\alpha,\beta}(u)$. As this problem is defined in $[0,1]$, $u(x) = 2x-1$. The piece of code if Fig. 5 shows how the solution can be reached by the mentioned method. Executing this code leads to the error depicted in Fig. 6.

```
N=2;
alpha_=1;
beta_=1;

%collocation points
X=(1+jacobi_zeros(N+1,alpha_,beta_))/2;


H=F(X)*inv(jacobi_(N,alpha_,beta_,0,X,@(x)2*x-1)');

D_=D_jacobi(N,alpha_,beta_,@(x)2*x-1);
A=2*(D_*D_)-eye(N+1,N+1);

% Boundary condition y(0)=0
A(1,:)=jacobi_(N,alpha_,beta_,0,0,@(x)2*x-1);
H(1,1)=0;

% Ac=H'-->c=A\H'
c=A\H';
```

Figure 5. Running Code for Example2

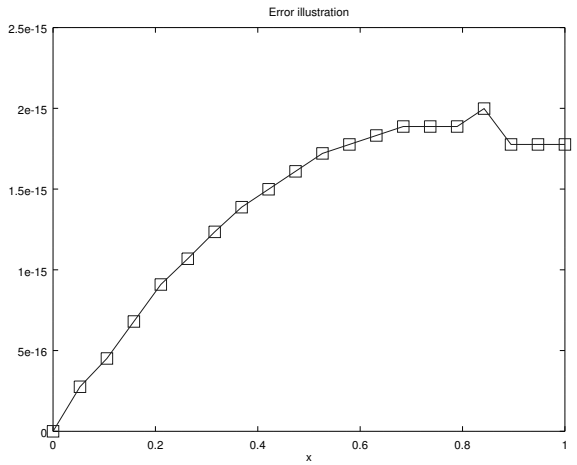

Figure 6. Error illustration of Example2.

G_jacobi()

In addition to the previous subsection, to have a matrix form for for $x^i y^{(j)}(x) = CD^{(j)}G^iJ$ we need to define [15]:

$$G_{pq} = \begin{cases} \frac{2(p+\alpha)(p+\beta)}{(\alpha+\beta+2p)(\alpha+\beta+2p+1)}, & p = q-1, \\ \frac{(-\alpha^2+\beta^2)}{(\alpha+\beta+2p+1)(\alpha+\beta+2p+2)} & p = q, \\ \frac{2p(\alpha+\beta+p)}{(\alpha+\beta+2p)(\alpha+\beta+2p-1)} & p = q+1, \\ 0 & o.w. \end{cases}$$

Declaration
$G\_jacobi(N, alpha\_, beta\_)$:
Input:
N: integer
alpha_, beta_ : double
Output:
class: double
Dimension:[(N+1)x(N+1)]

D_Jacobi_frac()

Here, the fractional derivative operational matrix over $[a,b]$ with $u(x) = 2\frac{x-a}{b-a} - 1$ is of interest.

If $\tau^{(\nu)}$ be the fractional derivative operator, we write

$$\tau^{(\nu)}\phi^T(x) = D^\nu \phi^T(x)$$

where $\phi(x) = [J_0^{\alpha,\beta}, ..., J_N^{\alpha,\beta}]$ and
$D^\nu_{(N+1)\times(N+1)}$=

$$\begin{bmatrix} 0 & 0 & \dots & 0 \\ \vdots 0 & 0 & \dots & 0 \\ D^\nu(\lceil\nu\rceil,0) & D^\nu(\lceil\nu\rceil,1) & \dots & D^\nu(\lceil\nu\rceil,N) \\ D^\nu(\lceil\nu\rceil+1,0) & D^\nu(\lceil\nu\rceil+1,0) & \dots & D^\nu(\lceil\nu\rceil+1,N) \\ \vdots & \vdots & & \vdots \\ D^\nu(N,0) & D^\nu(N,1) & \dots & D^\nu(N,N) \end{bmatrix}$$

The elements of this matrix are [13]

$$D^\nu(i,j) = \sum_{k=\lceil\nu\rceil}^{i} \frac{(u')^{\alpha+\beta+1}(-1)^{i-k}(b-a)^{\alpha+\beta+1-\nu}\Gamma(\beta+1+j)}{\Gamma(j+1+\alpha+\beta)} \times \frac{\Gamma(i+\beta+1)\Gamma(k+i+\alpha+\beta+1)}{\gamma_j^{\alpha,\beta}(i-k)!(j+\alpha+\beta)!\Gamma(k+\beta+1)\Gamma(k+1-\nu)} \times \sum_{l=0}^{j} \frac{(-1)^{j-l}\Gamma(j+l+\alpha+\beta+1)\alpha!\Gamma(k+l+\beta-\nu+1)}{l!(j-l)!\Gamma(l+\beta+1)\Gamma(k+l+\alpha+\beta+2-\nu)}.$$

It is clear that when the derivative order is 1 (an integer number) $D\_Jacobi\_frac(N, alpha\_, beta\_, r, u)$ function is providing the same result with $D\_Jacobi(N, alpha\_, beta\_, u)$.

Declaration
$D\_Jacobi\_frac(N, alpha\_, beta\_, r, u)$:
Input:
N: integer
alpha_, beta_ : double
r :(the derivative order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

Example 3: Consider

$$D^{(\frac{3}{2})}y(x) + D^{(2)}y(x) + y(x) = 1 + x,\ 0 \le x \le 1. \qquad (9)$$

with the exact solution $y(x) = 1 + x$. If we assume

$$y = \phi^T(x)C,\ \ y^{(i)} = \phi^T(x)D^{(i)}C,$$

$$f(x) = 1 + x,$$

$$f(x) = H^T\phi(x),$$

Hence we will arrive at $[D^{(1.5)} + D^{(2)} + I] = H^T$, where $\phi(x) = [\phi_0(x), ..., \phi_N(x)]$. By collocating N+1 zeros of Jacobi function $J^{\alpha,\beta}_{N+1}(u)$, H^T is reachable as

$$\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix}^T \cdot \begin{bmatrix} \phi_0(x_0) & \phi_0(x_1) & \dots & \phi_0(x_N) \\ \phi_1(x_0) & \phi_1(x_1) & \dots & \phi_1(x_N) \\ \vdots & & & \\ \phi_N(x_0) & \phi_N(x_1) & \dots & \phi_N(x_N) \end{bmatrix}^{-1} = H^T$$

Then we can rewrite the equation as

$$C^T[I^{(1-\vartheta)} + I^{(1)}] = H^T$$

The code in Fig. 7 can find the unknown C^T.

```
X=(1+jacobi_zeros(N+1,alpha_,beta_))/2;


H=F(X)*inv(jacobi_(N,alpha_,beta_,0,X,@(x)2*x-1)');

D_3_2=D_jacobi_frac(N,alpha_,beta_,1.5,@(x)2*x-1);
D_2=  D_jacobi_frac(N,alpha_,beta_,2,  @(x)2*x-1);

A=D_3_2+D_2+eye(N+1,N+1);

% Boundary condition y(0)=1, y'(0)=1
A(1,:)=jacobi_(N,alpha_,beta_,0,0,@(x)2*x-1);
H(1,1)=1;

A(2,:)=jacobi_(N,alpha_,beta_,1,0,@(x)2*x-1);
H(1,2)=1;

% Ac=H'-->H'=?
c=A\H';
```

Figure 7. Running Code for Example 3.

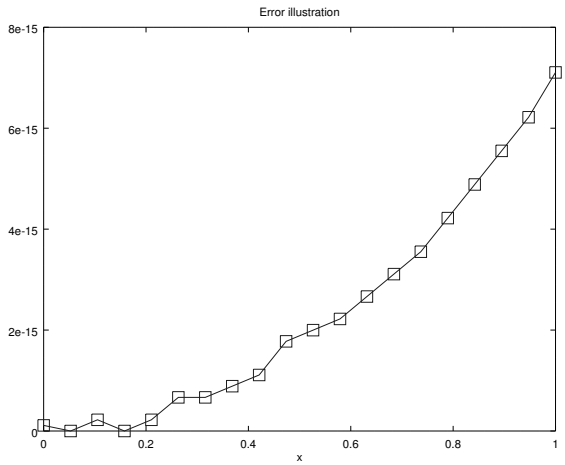

Figure 8. Error illustration of Example 3.

I_Jacobi()
To introduce the integral operational matrix, we start off by speaking of some useful relations as

$$\int_0^1 x^{j+1+s}(1-u(x))^{\alpha}(1+u(x))^{\beta}dx = \frac{2^{(\alpha+\beta)}\Gamma(\alpha+1)\Gamma(s+j+2+\beta)}{\Gamma(3+\alpha+\beta+j+s)}. \qquad (10)$$

We intend to calculate the operational matrix over [0,1] ($u(x)=2x-1$).

$$\int_0^x J_i^{\alpha,\beta}(u(t))dt=\sum_{l=0}^{N}\Big(\sum_{k=0}^{i}\frac{(-1)^{i-k}\Gamma(i+\beta+1)\Gamma(i+k+\alpha+\beta+1)}{\Gamma(k+\beta+1)\Gamma(i+\alpha+\beta+1)(k+1)!(i-k)!}$$
$$\sum_{r=0}^{l}\frac{2^{\alpha+\beta+1}(-1)^{l-r}\Gamma(l+r+\alpha+\beta+1)\Gamma(l+\beta+1)\Gamma(r+k+\alpha+\beta+2)\Gamma(\alpha+1)}{\gamma_l^{\alpha,\beta}\Gamma(r+\beta+1)\Gamma(l+\alpha+\beta+1)\Gamma(r+k+\alpha+\beta+3)(l-r)!r!}\Big)$$
$$\times J_l^{\alpha,\beta}(u(t)) \tag{11}$$

$$\int_0^x J_i^{\alpha,\beta}(u(t))dt=\sum_{l=0}^{N}I(i,l).J_l^{\alpha,\beta}(u(t))$$
$$=[I(i,0),I(i,1),...,I(i,N)]\phi(x), \tag{12}$$

where $\phi(x)=[J_0^{\alpha,\beta}(u(x)),J_1^{\alpha,\beta}(u(x)),...,J_N^{\alpha,\beta}(u(x))]^T$. So the operational matrix would be [14]

$$\int_0^x \phi(t)dt=I\phi(x),$$

in which

$$I_{(N+1)\times(N+1)}=\begin{bmatrix}I(0,0) & I(0,1) & \dots & I(0,N)\\ I(1,0) & I(1,1) & \dots & I(1,N)\\ \vdots & \ddots & & \\ I(N,0) & I(N,1) & \dots & I(N,N)\end{bmatrix},$$

$$I(i,l)=\sum_{k=0}^{i}w_{ilk},\ 0\le i,l\le N, \tag{13}$$

$$w_{ilk}=\frac{(-1)^{i-k}\Gamma(i+\beta+1)\Gamma(i+k+\alpha+\beta+1)}{\Gamma(k+\beta+1)\Gamma(i+\alpha+\beta+1)(k+1)!(i-k)!}$$
$$\sum_{r=0}^{l}\frac{2^{\alpha+\beta+1}(-1)^{l-r}\Gamma(l+r+\alpha+\beta+1)\Gamma(l+\beta+1)\Gamma(r+k+\alpha+\beta+2)\Gamma(\alpha+1)}{\gamma_l^{\alpha,\beta}\Gamma(r+\beta+1)\Gamma(l+\alpha+\beta+1)\Gamma(r+k+\alpha+\beta+3)(l-r)!r!}$$

It has been said that for an integral defined over [0,b], we need to multiply Eq. 13 by b: $I=bI$.

Declaration
$I\_Jacobi(N,alpha\_,beta\_,u)$:
Input:
N: integer
alpha_, beta_ : double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

The following execution will show the $I_{(3)\times(3)}$ when $u(x)=2x-1$.

Code execution
>>> $I\_jacobi(2,1,1,@(x)2*x-1)$
ans =

0.50000	0.25000	0.00000
-0.40000	0.00000	0.13333
0.25000	-0.05357	0.00000

I_Jacobi_frac()
Like the previous section, we aim at the fractional integral operational matrix over $[0,1]$ with $u(x)=2x-1$. The Riemann–Liouville fractional integral operator of order $\vartheta>0$ of the vector $\phi(x)$ can be expressed by:

$$\kappa^{\vartheta}\phi(x)\simeq I^{\vartheta}\phi(x),$$

where I^{ϑ} is the $(N+1)$ by $(N+1)$ operational matrix of Riemann–Liouville fractional integral of order ϑ.

$$I^{(\vartheta)}_{(N+1)\times(N+1)}=\begin{bmatrix}I(0,0,\vartheta) & I(0,1,\vartheta) & \dots & I(0,N,\vartheta)\\ I(1,0,\vartheta) & I(1,1,\vartheta) & \dots & I(1,N,\vartheta)\\ \vdots & \ddots & & \\ I(N,0,\vartheta) & I(N,1,\vartheta) & \dots & I(N,N,\vartheta)\end{bmatrix}.$$

The elements of this matrix are

$$I(i,j,\vartheta)=\sum_{k=0}^{i}\sum_{l=0}^{j}p_k^{(i)}p_l^{(j)}\times$$
$$\frac{2^{(\alpha+\beta+1)}\Gamma(k+1)B(k+l+\vartheta+\beta+1,\alpha+1)}{\Gamma(k+\vartheta+1)\gamma_j^{\alpha,\beta}}, \tag{14}$$

where

$$B(s,t)=\frac{\Gamma(s)\Gamma(t)}{\Gamma(s+t)},\ p_i^{(n)}=(-1)^{(n-i)}\binom{n+\alpha+\beta+i}{i}\binom{n+\alpha}{n-i}.$$

To see how this formula is reached see [16].

Declaration
$I\_Jacobi\_frac(N,alpha\_,beta\_,r,u)$:
Input:
N: integer
alpha_, beta_ : double
r :(the integral order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

Obviously, when $\vartheta=1$ this matrix is equal to the operational matrix of integral that is explained in the previous subsection.

Code execution

$>>> I\_jacobi\_frac(2,1,1,0.8,@(x)2*x-1)$
ans =

0.605454	0.252272	-0.016238
-0.426060	0.104389	0.166203
0.250533	-0.108823	0.060581

$>>> I\_jacobi\_frac(2,1,1,1,@(x)2*x-1)$
ans =

5.0000e-01	2.5000e-01	8.8818e-16
-4.0000e-01	4.4409e-16	1.3333e-01
2.5000e-01	-5.3571e-02	-3.5527e-15

Example 4: Consider an inhomogeneous linear equation [16]

$$D^{\vartheta}y(x)+y(x)=\frac{2x^{2-\vartheta}}{\Gamma(3-\vartheta)}-\frac{x^{1-\vartheta}}{\Gamma(2-\vartheta)}+x^2-x,\ {}_{y(0)=0,\ 0<\vartheta\leq 1.} \tag{15}$$

with the exact solution $y(x)=x^2-x$. If we assume

$$D^1y=C^T\phi(x),\quad y=C^TI^{(1)}\phi(x),\quad D^{\vartheta}y=C^TI^{(1-\vartheta)}\phi(x)$$

$$f(x)=\frac{2x^{2-\vartheta}}{\Gamma(3-\vartheta)}-\frac{x^{1-\vartheta}}{\Gamma(2-\vartheta)}+x^2-x$$

$$f(x)=H^T\phi(x),$$

where $\phi(x)=[\phi_0(x),...,\phi_N(x)]$, $\phi_k(x)=J_k^{\alpha,\beta}(u(x))$, and $u(x)=2x-1$. By collocating $N+1$ zeros of Jacobi function $J_{N+1}^{\alpha,\beta}(u)$, H^T is reachable as

$$\begin{bmatrix} f(x_0)\\ f(x_1)\\ \vdots\\ f(x_N)\end{bmatrix}^T\cdot\begin{bmatrix}\phi_0(x_0) & \phi_0(x_1) & \dots & \phi_0(x_N)\\ \phi_1(x_0) & \phi_1(x_1) & \dots & \phi_1(x_N)\\ \vdots & & & \\ \phi_N(x_0) & \phi_N(x_1) & \dots & \phi_N(x_N)\end{bmatrix}^{-1}=H^T.$$

Then we can rewrite the equation as

$$C^T[I^{(1-\vartheta)}+I^{(1)}]=H^T.$$

The code in Fig. 9 can determine the unknown C^T.

In Fig. 10 we showed the difference between $Dy=C^T\phi(x)$ and $(2x)-1$ as the exact solution.

```
N=1;
alpha_=1;
beta_=1;
r=0.2;

u_=@(x)2*x-1;
X=(1+jacobi_zeros(N+1,alpha_,beta_))/2;


H=f(X,r)*inv(jacobi_(N,alpha_,beta_,0,X,u_)');


A=I_jacobi_frac(N,alpha_,beta_,1  ,)+...
  I_jacobi_frac(N,alpha_,beta_,1-r,u_);

%Ac=H-->c=?
c=H*inv(A);
```

Figure 9. Running Code for Example 4.

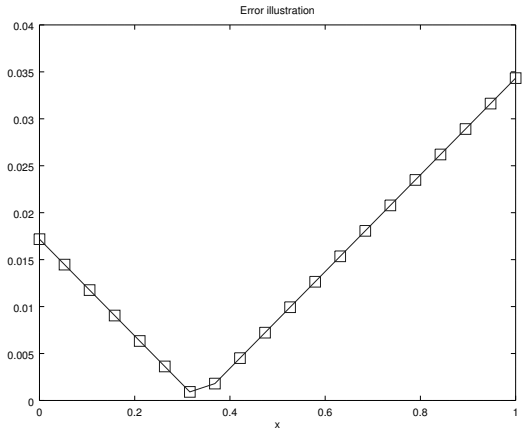

Figure 10. Error illustration of Example 4.

lgr_Jacobi_()

We happened to cope with a situation when we need to know the value of Lagrange polynomials in certain points. Function $lgr\_Jacobi\_()$ gives the this value by the following declaration:

Declaration
$lgr\_Jacobi\_(N, z, points, u)$:
Input:
N: integer
z: [double 1xm]
points: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[MxN]

which represent the following implementations.

Note: *points* are the points that Lagrange polynomials are made out of.
z are the values we want to put in the Lagrange polynomials to give us the relevant values.

With $w(x) = \prod_{i=0}^{N} \big(h(x) - h(x_i)\big)$ the generalized Lagrange (GL) functions formula is formulated as ($j = 0, ..., N$) [17]:

$$L_j^u(x) = \frac{w(x)}{(u-u_j)\partial_x w(x_j)} = \frac{u_j' w(x)}{(u-u_j)\partial_h w(x_j)} = \kappa_j \frac{w(x)}{(u-u_j)}, \tag{16}$$

where $\kappa_j = \frac{u_j'}{\partial_u w(x_j)}$, $\partial_u w(x) = \frac{1}{u'}\partial_x w(x)$, and $u(x)$ is the shifting function parameter. For understanding how it is reached see [17]. While with the vector input $z = [z_0, .., z_m]$, this function must be written as it gives a vector for each of the value of z_i:

$$L_j^u(z_i) = \kappa_j \frac{w(z_i)}{(u(z_i) - u(z_j))}, \ j = 0, ..., N, \ i = 0, .., m.$$

So, aforementioned function declaration code is giving this desired result. The subsequent code execution is showing some instant examples for this manner.

Code execution
$>>> r = (1 + jacobi\_zeros(3, 0, 1))/2;$
$>>> lgr\_Jacobi\_(3, r, r, @(x)2*x-1)$
ans =

1	0	-0
-0	1	0
0	-0	1

$>>> lgr\_Jacobi\_(3, r(2), r, @(x)2*x-1)$
ans =

-0	1	0

$>>> lgr\_Jacobi\_(3, [r(2), -0.7, 5], r, @(x)2*x-1)$
ans =

-0.000000e+00	1.000000e+00	0.000000e+00
7.865768e+00	-1.211461e+01	5.248840e+00
6.819065e+01	-1.613030e+02	9.411234e+01

As pointed earlier, for reaching the derivative matrix of these functions we require a formula providing us with a derivative-free method; the first order derivative of these functions based on Gauss/Gauss-Radau/Gauss-Lobatto Jacobi points are represented in the next subsection.

D_lgr_jacobi()

This function returns the first-order derivative matrix of generalized Lagrange Jacobi [17, 18, 19]. For instance, one can have derivative matrix of generalized Lagrange with Gauss, Gauss-Radau(both left- and right- fixed side), and Gauss-Lobatto by calling:

$D\_lgr\_jacobi(N, \alpha, \beta, points, u, 'gauss')$
$D\_lgr\_jacobi(N, \alpha, \beta, points, u, 'gauss\_rdu\_l')$
$D\_lgr\_jacobi(N, \alpha, \beta, points, u, 'gauss\_rdu\_f')$
$D\_lgr\_jacobi(N, \alpha, \beta, points, u, 'gauss\_lbt')$
The relevant function call and syntax for obtaining these values is as follows:

Declaration
$D\_lgr\_jacobi(N, alpha\_, beta\_, points, u, type)$:
Input:
N: integer
alpha_, beta_ : double
points: [double 1xM]
u: (symbolic shifting parameter)
type: can be each $'gauss', 'gauss\_rdu\_l', 'gauss\_rdu\_f'$), or $'gauss\_lbt'$
Output:
class: double
Dimension:[MxN]

Note: The input *points* are declaring the points the Lagrange polynomials are made out of (x_i in Eq. 16).
If we show this matrix by $\hat{D} = [d_{kj}]_{0 \le k,j \le N}$, then its Gauss

form is filled as [18]:

$$d_{kj}=\begin{cases}\dfrac{u'_k J_n^{\alpha+1,\beta+1}(u_k)}{J_n^{\alpha+1,\beta+1}(u_j)(u_k-u_j)}, & j\neq k,\\[2ex] \dfrac{u'_j(\alpha-\beta+\{2+\alpha+\beta\}u_j)}{-2(u_j^2-1)}, & j=k.\end{cases}$$

The following example shows how this formula can be reached by calling its relevant function $D\_lgr\_jacobi()$.

Code execution

$>>>$ $D\_lgr\_jacobi(4,0,1,(jacobi\_zeros(4,0,1)+1)/2,@(x)\ 2*x-1,'gauss')$

ans =

-6.57390	10.48709	-5.65025	1.73705
-1.24590	-1.54472	3.70395	-0.91333
0.52000	-2.86928	0.42325	1.92603
-0.89250	3.95001	-10.75286	7.69536

The prominent formula for reaching the first order derivative matrix formula of Gauss-Radau left- and right- fixed side are as follow respectively [14]:

$d_{kj}=$

$$\begin{cases}\frac{u'_k J_{n-1}^{\alpha+1,\beta+2}(u_k)(n!)(\alpha+\beta+n+2)\Gamma(\alpha+1)}{2\Gamma(\alpha+1+n)}, j=n, 0\leq k\leq n-1,\\ \frac{2u'_k\Gamma(\alpha+n+1)}{\Gamma(\alpha+1)(n!)(\alpha+\beta+n+2)}\times\\ \frac{1}{J_{n-1}^{\alpha+1,\beta+2}(u_j)(u_k-u_j)(u_j-u_k)}, k=n, 0\leq j\leq n-1,\\ \frac{u'_k(u_k-u_n)}{(u_j-u_n)(u_k-u_j)}\frac{J_{n-1}^{\alpha+1,\beta+2}(u_k)}{J_{n-1}^{\alpha+1,\beta+2}(u_j)}\\ , 0\leq j\neq k\leq n-1,\\ \frac{u'_j}{u_j-u_n}+u'_j(\alpha+\beta+n+3)\frac{J_{n-2}^{\alpha+2,\beta+3}(u_j)}{4J_{n-1}^{\alpha+1,\beta+2}(u_j)}\\ , 0\leq k=j\leq n-1,\\ \frac{u'_k(n-1)(\alpha+\beta+n+2)}{2(\alpha+1)}, \quad k=j=n,\end{cases}$$

$d_{kj}=$

$$\begin{cases}\frac{u'_k J_{n-1}^{\alpha+1,\beta+2}(u_k)(n!)(\alpha+\beta+n+2)\Gamma(\beta+2)}{2\Gamma(\beta+2+n)(-1)^n}\\ , j=0, 1\leq k\leq n,\\ \frac{2u'_k\Gamma(\beta+n+2)(-1)^n}{\Gamma(\beta+2)(n!)(\alpha+\beta+n+2)}\times\\ \frac{1}{J_{n-1}^{\alpha+1,\beta+2}(u_j)(u_k-u_j)(u_j-u_k)}, k=0, 1\leq j\leq n,\\ \frac{u'_k(u_k-u_0)}{(u_j-u_0)(u_k-u_j)}\frac{J_{n-1}^{\alpha+1,\beta+2}(u_k)}{J_{n-1}^{\alpha+1,\beta+2}(u_j)}, 1\leq j\neq k\leq n,\\ \frac{u'_j}{u_j-u_0}+u'_j(\alpha+\beta+n+3)\frac{J_{n-2}^{\alpha+2,\beta+3}(u_j)}{4J_{n-1}^{\alpha+1,\beta+2}(u_j)}\\ , 1\leq k=j\leq n,\\ \frac{-u'_k(n)(\alpha+\beta+n+2)}{2(\beta+2)}, \quad k=j=0,\end{cases}$$

The relevant calling function for this is brought bellow:

Code execution

$>>>$ $D\_lgr\_jacobi(4,0,1,[(jacobi\_zeros(3,0,2)+1)/2;1],@(x)2*x-1,'gauss\_rdu\_l')$

ans =

-5.79400	13.0897	-19.96621	-12.67045
-0.59608	-3.73801	9.82742	5.49332
0.12539	-1.35528	-8.46799	-9.69788
-0.15879	1.51181	-19.35302	-18.0000

$>>> D\_lgr\_jacobi(4,0,1,[-1;jacobi\_zeros(3,0,2)],@(x)x,'gauss\_rdu\_f')$

ans =

-3.00000	4.53462	-2.13459	0.59997
-0.63352	-0.49286	1.45530	-0.32893
0.27467	-1.34037	0.33760	0.72810
-0.48489	1.90281	-4.57317	3.15525

The first order derivative matrix of Generalized Lagrange Jacobi functions for Gauss-Lobatto is also with [19]:

$$d_{kj} =$$

$$\begin{cases}
\frac{u'_k J_{n-2}^{\alpha+2,\beta+2}(u_k)(u_k-u_n)(n-1)!(\alpha+\beta+n+2)\Gamma(\beta+2)}{2(u_0-u_n)(-1)^{n-1}\Gamma(\beta+1+n)} \\
, j=0, 1\le k\le n-1, \\
\\
\frac{u'_n\Gamma(\alpha+n+1)\Gamma(\beta+2)}{(u_0-u_n)(-1)^{n-1}\Gamma(\beta+n+1)\Gamma(\alpha+2)} \\
, j=0, k=n, \\
\\
\frac{2(-1)^n u'_0(u_0-u_n)\Gamma(\beta+n+1)}{\Gamma(\beta+2)(n-1)!(\alpha+\beta+n+2)J_{n-2}^{\alpha+2,\beta+2}(u_j)(u_0-u_j)^2(u_j-u_n)} \\
, k=0, 1\le j\le n-1, \\
\\
\frac{u'_0\Gamma(\beta+n+1)\Gamma(\alpha+2)(-1)^{n-1}}{(u_n-u_0)\Gamma(\alpha+n+1)\Gamma(\beta+2)} \\
, k=0, j=n, \\
\\
\frac{u'_k J_{n-2}^{\alpha+2,\beta+2}(u_k)(u_k-u_0)(n-1)!(\alpha+\beta+n+2)\Gamma(\alpha+2)}{2(u_n-u_0)\Gamma(\alpha+1+n)} \\
, j=n, 1\le k\le n-1, \\
\\
\frac{(u_k-u_0)(u_k-u_n)u'_k}{(u_j-u_0)(u_j-u_n)(u_k-u_j)}\frac{J_{n-2}^{\alpha+2,\beta+2}(u_k)}{J_{n-2}^{\alpha+2,\beta+2}(u_j)} \\
, 1\le j\ne k\le n-1, \\
\\
\frac{2u'_n(u_n-u_0)\Gamma(\alpha+n+1)}{-(u_j-u_0)(u_j-u_n)^2 J_{n-2}^{\alpha+2,\beta+2}(u_j)(n+\alpha+\beta+2)(n-1)!\Gamma(\alpha+2)} \\
, 1\le j\le n-1, k=n \\
\\
\frac{u'_j}{u_j-u_0}+\frac{u'_j}{u_j-u_n}+u'_j(\alpha+\beta+n+3)\frac{J_{n-3}^{\alpha+3,\beta+3}(u_j)}{4J_{n-2}^{\alpha+2,\beta+2}(u_j)} \\
, 1\le k=j\le n-1, \\
\\
\frac{u''_j}{2u_j}-\frac{u'_j}{u_n-u_0}+\frac{u'_j(n+\alpha+\beta+2)J_{n-2}^{\alpha+2,\beta+2}(u_j)}{2J_{n-1}^{\alpha+1,\beta+1}(u_j)} \\
, k=j=0, \\
\frac{u''_j}{2u_j}+\frac{u'_j}{u_n-u_0}+\frac{u'_j(n+\alpha+\beta+2)J_{n-2}^{\alpha+2,\beta+2}(u_j)}{2J_{n-1}^{\alpha+1,\beta+1}(u_j)} \\
, k=j=n,
\end{cases}$$

and, similarly coding this function is as follows:

```
u_=@(x)x/2;
X=[-2,2*jacobi_zeros(N-1,alpha_+1,beta_+1)',2];

H=F(X);

D_1=D_lgr_jacobi(N+1,alpha_,beta_,X,u_,'gauss_lbt');
V_1=diag(ones(N+1,1)*(1/2.0))
V_minu_1=diag(ones(N+1,1)*(1/2.0))^(-1);
D_2=   (V_1*D_1)*V_minu_1*D_1;%

A=2*(D_2)-eye(N+1,N+1);

% Boundary condition y(-2)=4
A(1,:)=[1,zeros(1,N)];
H(1,1)=4;
% Boundary condition y(2)=4
A(N+1,:)=[zeros(1,N),1];
H(1,N+1)=4;

% Ac=H'-->c=A\H'
c=A\H';
```

Figure 11. Running Code for Example 5.

Code execution

$>>> D\_lgr\_jacobi(5,0,1,[-1; jacobi\_zeros(3,1,2); 1], @(x)x,'gauss\_lbt')$

ans =

...

$>>> D\_lgr\_jacobi(6,0,1,[0;(1+jacobi\_zeros(4,1,2))/2;1], @(x)2*x-1,'gauss\_lbt')$

ans =

...

If anyone eager to understand the proof of these formulas, see [18, 19].

What we have said were all about the first order derivatives. For derivatives of higher orders we need to calculate a mathematical relation as:

$$D^{(m)} = \left(\sum_{k=0}^{m-1}\binom{k}{m-1}V^{(k)}D^{(m-1-k)}\right)V^{-1}D. \qquad (17)$$

where $V = Diag(u'_0, u'_1, .., u'_N)$, $V^{(1)} = Diag(u''_0, u''_1, .., u''_N)$, and $V^{-1} = Diag(\frac{1}{u'_0}, \frac{1}{u'_1}, .., \frac{1}{u'_N})$.

Example 5: The relevant piece of code in Fig. 11 solves Example 2 with Generalized Lagrangian Jacobi collocation method. In this example we assume that the domain is defined over $[-2,2]$. The error between the exact solution and approximate solutions is displayed in Fig. 12.

Example 6: Imagine a nonlinear PDE like the one in [10] called Fokker Planck:

$$\frac{\partial y}{\partial t} = [\frac{-\partial}{\partial x}A(x,t,y) + \frac{\partial^2}{\partial x^2}B(x,t,y)]y,$$

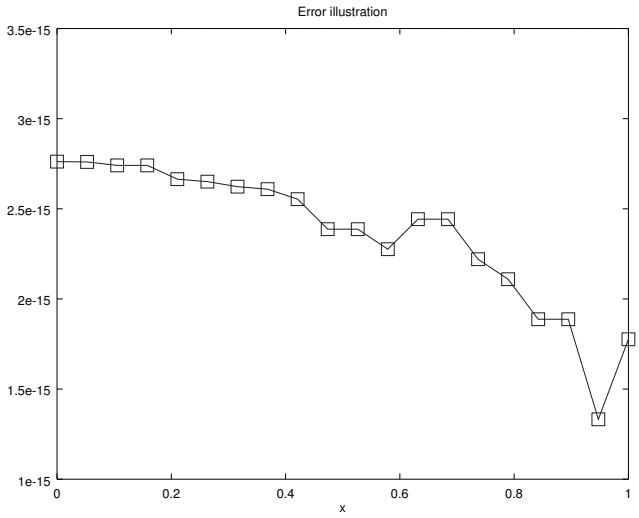

Figure 12. Error illustration of Example 5.

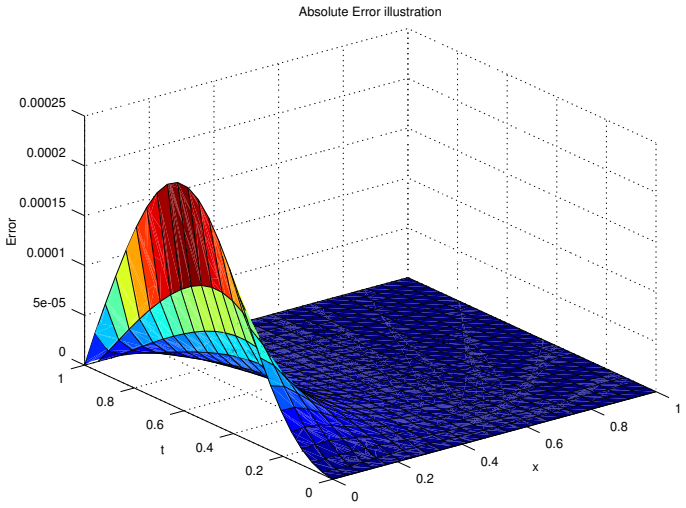

Figure 13. Error illustration of Example 6.$\alpha = \beta = 1, N = 7$.

where $A(x,t,y) = \frac{4y}{x} - \frac{x}{3}$ and $B(x,y,t) = y$ with the initial and boundary conditions $y(x,0) = x^2$, $y(0,t) = 0$ and $y(1,t) = e^t$. The exact solution is $y(x,t) = x^2 e^t$ and $x \in [0,1]$, $t \in [0,1]$.

For solving this problem, with help of Crank-Nicolson method, we discritized the variable t and solved the equation in different iterations [19]. Notice that $t_n = n \times \triangle t$ and $\triangle t = t_{n+1} - t_n$.

Using $\frac{\partial y}{\partial t} = \frac{y^{n+1} - y^n}{\triangle t}$ where $y^n = y(x,t_n)$ and applying Crank-Nicolson method we have got

$$y_t = (\frac{4y}{x^2} - \frac{8y_x}{x} + \frac{1}{3} + 2y_{xx})y + (\frac{x}{3} + 2y_x)y_x,$$

$$\frac{y^{n+1} - y^n}{\triangle t} = \theta(\frac{4y^n}{x^2} - \frac{8y_x^n}{x} + \frac{1}{3} + 2y_{xx}^n)y^{n+1} + (\frac{x}{3} + 2y_x^n)y_x^{n+1} + \triangle t(1-\theta)(\frac{4y^n}{x^2} - \frac{8y_x^n}{x} + \frac{1}{3} + 2y_{xx}^n)y^n + (\frac{x}{3} + 2y_x^n)y_x^n, \quad (18)$$

in which y^{n+1} is unknown and is supposed to be determined. Simplifying the last equation we have

$$\left[1 - \theta \triangle t\left((\frac{4y^n}{x^2} - \frac{8y_x^n}{x} + \frac{1}{3} + 2y_{xx}^n)\right)\right]y^{n+1} + \theta \triangle t(\frac{x}{3} + 2y_x^n)y_x^{n+1} = y^n + (1-\theta)\triangle t\left((\frac{4y^n}{x^2} - \frac{8y_x^n}{x} + \frac{1}{3} + 2y_{xx}^n)\right)y^n + (1-\theta)\triangle t(\frac{x}{3} + 2y_x^n)y_x^n, \quad (19)$$

Now for solving this equation, we use Generalized Lagrange polynomials: $y^{n+1} = \sum_{i=0}^{N} L_i^u(x) C_i^{n+1}$, where $u = 2x - 1$. By defining

$$tem1 := x \to \frac{4y^n}{x^2} - \frac{8y_x^n}{x} + \frac{1}{3} + 2y_{xx}^n,$$

$$tem2 := x \to \frac{x}{3} + 2y_x^n,$$

With Jacobi Gauss Lobatto points $x_0 = 0$, $x_N = 1$ and roots of $P_{N-1}^{\alpha+1,\beta+1}(u)$

$$x_i = [0, roots(P_{N-1}^{\alpha+1,\beta+1}(u)), 1], i = 0..N,$$

we will get a matrix form as

$$\Big[\big(I - \triangle t Diag(tem1)\big)I - \big(\triangle t Diag(tem2)\big)D\Big]C^{n+1} = \Big[\big(I + \triangle t Diag(tem1)\big)I + \big(\triangle t Diag(tem2)\big)D\Big]C^n, \quad (20)$$

where

$$Diag(tem1) = \begin{bmatrix} tem1(x_0) & 0 & \dots & 0 \\ 0 & tem1(x_1) & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & tem1(x_N) \end{bmatrix},$$

$$Diag(tem2) = \begin{bmatrix} tem2(x_0) & 0 & \dots & 0 \\ 0 & tem2(x_1) & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & tem2(x_N) \end{bmatrix},$$

and D is the Generalized Lagrange first order derivative matrix that is defined earlier. Also applying initial condition $y^0 = x^2$ leads to $C^0 = [x_0^2, x_1^2 ..., x_N^2]$ .The result of solving the system in Eq. (20) with SPSMAT is depicted in Fig. 13

3. Appendix

Gegenbauer Polynomials

Before we start, we recall that Gegenbauer polynomials are driven from Jacobi polynomials with formula [20]:

$$G_i^{\theta}(u(x)) = \frac{(i)!\Gamma(\theta+\frac{1}{2})}{\Gamma(i+\theta+\frac{1}{2})} J_i^{\theta-\frac{1}{2},\theta-\frac{1}{2}}(u(x)) \tag{21}$$

$u(x)$ is the shifting function parameter.

Gnbr_zeros()

The N zeros of N-th sentence of Gegenbauer polynomials with parameter tetha_ are achieved by $Gnbr\_zeros(N, tetha\_)$ that is equal to $jacobi\_zeros(N, tetha\_ - 0.5, tetha\_ - 0.5)$. The following function syntax can produce the Gegenbauer zeros over $[-1,1]$.

Declaration
$Gnbr\_zeros(N, tetha\_)$:
Input:
N: integer
theta_ : double
Output:
class: double
Dimension:[Nx1]

Gnbr_()

In regards with the relation between Jacobi polynomials and Gegenbauer polynomials, to have Gegnbauer polynomials matrix like what given in section 2.2.3, we need the following Hadamard product($\nu = \theta - \frac{1}{2}$).

$$\begin{bmatrix} \frac{0!\Gamma(\theta+0.5)}{\Gamma(\theta+0.5)} & \frac{1!\Gamma(\theta+0.5)}{\Gamma(1+\theta+0.5)} & \cdots & \frac{(N)!\Gamma(\theta+0.5)}{\Gamma(N+\theta+0.5)} \end{bmatrix} \circ \begin{bmatrix} \frac{d^k J_0^{\nu,\nu}(u(z_0))}{dx^k} & \frac{d^k J_1^{\nu,\nu}(u(z_0))}{dx^k} & \frac{d^k J_2^{\nu,\nu}(u(z_0))}{dx^k} & \cdots & \frac{d^k J_N^{\nu,\nu}(u(z_0))}{dx^k} \\ \frac{d^k J_0^{\nu,\nu}(u(z_1))}{dx^k} & \frac{d^k J_1^{\nu,\nu}(u(z_1))}{dx^k} & \frac{d^k J_2^{\nu,\nu}(u(z_1))}{dx^k} & \cdots & \frac{d^k J_N^{\nu,\nu}(u(z_1))}{dx^k} \\ \vdots & & & & \\ \frac{d^k J_0^{\nu,\nu}(u(z_m))}{dx^k} & \frac{d^k J_1^{\nu,\nu}(u(z_m))}{dx^k} & \frac{d^k J_2^{\nu,\nu}(u(z_m))}{dx^k} & \cdots & \frac{d^k J_N^{\nu,\nu}(u(z_m))}{dx^k} \end{bmatrix}$$

That can be shown with

$$P_N^{\theta} = \begin{bmatrix} \frac{0!\Gamma(\theta+0.5)}{\Gamma(\theta+0.5)} & \frac{1!\Gamma(\theta+0.5)}{\Gamma(1+\theta+0.5)} & \cdots & \frac{(N)!\Gamma(\theta+0.5)}{\Gamma(N+\theta+0.5)} \end{bmatrix}, \tag{22}$$

$$Gnbr\_(N,\theta,k,z,u) = P_N^{\theta} \circ jacobi\_(N,\nu,\nu,k,z,u), \tag{23}$$

where $\nu = \theta - \frac{1}{2}$.

Declaration
$Gnbr\_(N, theta\_, k, z, u)$:
Input:
N: integer
theta_ : double
k (derivative order): integer
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

Gnbr_frac()

The fractioanl derivatives of Gegenbauer functions are obtained as:

$$Gnbr\_frac(N,\theta,\vartheta,z,u) = P_N^{\vartheta} \circ jacobi\_frac(N,\nu,\nu,\vartheta,z,u).$$

where $\nu = \theta - \frac{1}{2}$ and ϑ is the fractional derivative order.

Declaration
$Gnbr\_frac(N, theta\_, nu, z, u)$:
Input:
N: integer
theta_ : double
nu (derivative order): double
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

D_Gnbr()

As said in subsection 2.2.5 for obtaining the operational matrix of derivative, we write $\phi^T D^{(i)} = \varphi^T$, where $\phi(x) = [J_0^{\alpha,\beta},...,J_N^{\alpha,\beta}]$, $\varphi(x) = [\frac{dJ_0^{\alpha,\beta}(u)}{dx},...,\frac{dJ_N^{\alpha,\beta}(u)}{dx}]$. Hence,

$$P_N^T \circ \phi^T \times D\_Gnbr(N,\theta,u) = P_N^T \circ \varphi^T,$$

in other words,

$$D\_Gnbr(N,\theta,u) = Diag(1./P_N) \times D\_jacobi(N,\theta-0.5,\theta-0.5,u) \times Diag(P_N).$$

where

$$Diag(P_N) = \begin{bmatrix} \frac{0!\Gamma(\theta+0.5)}{\Gamma(\theta+0.5)} & 0 & \cdots & & \\ & \frac{1!\Gamma(\theta+0.5)}{\Gamma(1+\theta+0.5)} & & & \\ & & & & \\ & & \cdots & 0 & \frac{N!\Gamma(\theta+0.5)}{\Gamma(N+\theta+0.5)} \end{bmatrix},$$

$$Diag(1./P_N) = \begin{bmatrix} \frac{\Gamma(\theta+0.5)}{0!\Gamma(\theta+0.5)} & 0 & \cdots & & \\ & \frac{\Gamma(1+\theta+0.5)}{1!\Gamma(\theta+0.5)} & & & \\ & & & & \\ & & \cdots & 0 & \frac{\Gamma(N+\theta+0.5)}{N!\Gamma(\theta+0.5)} \end{bmatrix}.$$

Declaration
$D\_Gnbr(N, theta\_, u\_)$:
Input:
N: integer
theta_ : double
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

D_Gnbr_frac()

In order to give the fractional operational matrix of derivative for Gegnbauer functions, we need to use the assumptions: $\phi^T D^{(i)} = \varphi^T$, where $\phi(x) = [G_0^\theta(u), ..., G_N^\theta(u)]$, $\varphi(x) = [\frac{dG_0^\theta(u)}{dx}, ..., \frac{dG_N^\theta(u)}{dx}]$. Hence,

$$P_N^T \circ \phi^T \times D\_Gnbr\_frac(N, \theta, \nu, u) = P_N^T \circ \varphi^T,$$

in other words,

$$D\_Gnbr(N, \theta, u) = Diag(1./P_N)$$
$$D\_jacobi\_frac(N, \theta - 0.5, \theta - 0.5, \nu, u) Diag(P_N)$$

Declaration
$D\_Gnbr\_frac(N, theta\_, r, u\_)$:
Input:
N: integer
theta_ : double
r :(the derivative order) double
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Gnbr()

In short, or obtaining the integral operational matrix of Gegenbauer functions, like preceding subsections, we need to:

$$I\_Gnbr(N, \theta, u) =$$
$$Diag(1./P_N) I\_jacobi(N, \theta - 0.5, \theta - 0.5, u) Diag(P_N).$$

Declaration
$I\_Gnbr(N, theta\_, u)$:
Input:
N: integer
theta_ : double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Gnbr_frac()

Similarly, for the fractional integral operational matrix of Gegenbauer functions, we write:

$$I\_Gnbr\_frac(N, \theta, \vartheta, u) =$$
$$Diag(1./P_N) I\_jacobi\_frac(N, \theta - 0.5, \theta - 0.5, \vartheta, u)$$
$$(P_N).$$

Declaration
$I\_Gnbr\_frac(N, theta\_, r, u)$:
Input:
N: integer
theta_ : double
r :(the integral order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

Igr_Gnbr()

Similar to Generalized Lagrange Jacobi in subsection 2.2.10, for having Generalized Lagrange Gegenbauer functions we need to send the points (Gauss/Gauss-Radau/Gauss-Lobatto) to the Lagrange function and build Generalized Lagrange Gegenbauer functions for any of Gauss/Gauss-Radau/Gauss-Lobatto points. The only and key difference between these functions and those of subsection 2.2.10 is that here we use Gegenbauer points (Gauss/Gauss-Radau/Gauss-Lobatto) and all the processes for assumptions and methodologies are literally the same. So, the code syntax for reaching these functions is:

Declaration
$lgr\_Gnbr\_(N, z, points, u)$:
Input:
N: integer
z: [double 1xm]
points: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[MxN]

D_lgr_Gnbr()

By setting $\alpha = \beta = \theta - 0.5$ in formula subsection 2.2.11, the first order derivative of Generalized Gegenbauer functions is obtained. For higher order, one must act as like as what shown in Eq. (17).

Declaration
$D\_lgr\_Gnbr(N, theta\_, points, u, type)$:
Input:
N: integer
theta_ : double
points: [double 1xM]
u: (symbolic shifting parameter)
type: can be each $'gauss', 'gauss\_rdu\_l', 'gauss\_rdu\_f'$), or $'gauss\_lbt'$
Output:
class: double
Dimension:[MxN]

Chebyshev Polynomials(1st kind)

Chebyshev polynomials are a special form of Jacobi polynomials:

$$T_i(u(x)) = \frac{(i)!\Gamma(\frac{1}{2})}{\Gamma(i+\frac{1}{2})} J_i^{-\frac{1}{2},-\frac{1}{2}}(u(x)) \tag{24}$$

Cby_zeros_1()

The N zeros of N-th sentence of Chebyshev of the first kind polynomials are achieved by $Cby\_zeros\_1(N)$. This gives the same result when we call $jacobi\_zeros(N, -0.5, \_\ 0.5)$.

Declaration
$Cby\_zeros\_1(N)$:
Input:
N: integer
Output:
class: double
Dimension:[Nx1]

Cby_1_ ()

The formula

$$Cby\_1\_(N,k,z,u) = R_N \circ jacobi\_(N,-0.5,-0.5,k,z,u). \tag{25}$$

leads to achieving Chebyshev functions of the first kind, where

$$R_N = \left[\frac{\Gamma(0.5)}{\Gamma(0+0.5)} \quad \frac{1!\Gamma(0.5)}{\Gamma(1+0.5)} \quad \cdots \quad \frac{N!\Gamma(0.5)}{\Gamma(N+0.5)}\right].$$

Declaration
$Cby\_1(N,k,z,u)$:
Input:
N: integer
k (derivative order): integer
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

Cby_1_frac()

The fractional derivatives of Chebyshev functions of the first kind are obtained as:

$$Cby\_frac\_1(N,\vartheta,z,u) = R_N \circ jacobi\_frac(N,\nu,\nu,\vartheta,z,u).$$

where $\nu = -\frac{1}{2}$ and ϑ is the fractional derivative order.

Declaration
$Cby\_1\_frac(N, nu, z, u)$:
Input:
N: integer
nu (derivative order): double
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

D_Cby_1()

As we said earlier, when we use the formula in Eq. 2.2.5, we can not use $\alpha = -0.5$, $\beta = -0.5$. The reason beyond this fact is that the matrix of Eq. 2.2.5 is singular for these values and the inverse matrix of E is not computable. We, however, used the formula in [9] for obtaining $D\_Cby\_1()$. So, if we show the elements of Chebyshev derivative operational matrix of the first kind with d_{ji}, then it is filled with:

$$d(j,i) = (\frac{2}{b-a}) \begin{cases} 0, & j \geq i, \\ i-1, & j = 1 \text{ and } i \text{ is odd}, \\ 2(i-1), & j!=1 \text{ and } (i \text{ is odd}) \text{ and } (j \text{ is even}), \\ & j!=1 \text{ and } (j \text{ is odd}) \text{ and } (i \text{ is even}), \\ 0, & O.W. \end{cases}$$

$0 \leq i, j \leq N$. The function $D\_Cby\_1()$ is returning the values of d_{ji}. Notice that $[a,b]$ is the interval of our interest.

Declaration
$D\_Cby\_1(N,u\_)$:
Input:
N: integer
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

D_Cby_1_frac()

For presenting the fractional derivative operational matrix of the first kind of Chebyshev functions, we have used the idea in [9].
If $\tau^{(\nu)}$ be the fractional derivative operator, we write

$$\tau^{(\nu)}\phi^T(x) = D^\nu \phi^T(x),$$

where $\phi(x) = [T_0(u), ..., T_N(u)]$, and
$D^\nu_{(N+1)\times(N+1)}$=

$$\begin{bmatrix} 0 & 0 & \dots & 0 \\ \vdots 0 & 0 & \dots & 0 \\ D^\nu(\lceil\nu\rceil,0) & D^\nu(\lceil\nu\rceil,1) & \dots & D^\nu(\lceil\nu\rceil,N) \\ D^\nu(\lceil\nu\rceil+1,0) & D^\nu(\lceil\nu\rceil+1,0) & \dots & D^\nu(\lceil\nu\rceil+1,N) \\ \vdots & \vdots & & \vdots \\ D^\nu(N,0) & D^\nu(N,1) & \dots & D^\nu(N,N) \end{bmatrix},$$

The elements of this matrix are filled as

$$D^\nu(i,j) = \frac{1}{\varepsilon_j (b-a)^\nu} \times \sum_{k=\lceil\nu\rceil}^{i} \frac{(-1)^{i-k} 2i(i+k-1)!\Gamma(k-\nu+\frac{1}{2})}{\Gamma(k+\frac{1}{2})(i-k)!\Gamma(k-\nu-j+1)\Gamma(k+j-\nu+1)},$$

where $0 \le \nu \le 1$, and

$$\varepsilon_j = \begin{cases} 2, & j = 0, \\ 1, & j \ge 1. \end{cases}$$

Declaration
$D\_Cby\_frac\_(N,r,u)$:
Input:
N: integer
r :(the derivative order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Cby_1()

For the integral operational matrix for Chebyshev functions of the 1st kind we assume

$$\int_0^x T_i(u(t))dt = \sum_{l=0}^{N} I(i,l).T_l(u(t)) = [I(i,0),I(i,1),...,I(i,N)]\phi(x),$$

where $\phi(x) = [T_0(u(x)), T_1(u(x)), ..., T_N(u(x))]^T$. So the operational matrix would be [21]

$$\int_0^x \phi(t)dt = I\phi(x),$$

in which

$$I_{(N+1)\times(N+1)} = \begin{bmatrix} I(0,0) & I(0,1) & \dots & I(0,N) \\ I(1,0) & I(1,1) & \dots & I(1,N) \\ \vdots & \ddots & & \\ I(N,0) & I(N,1) & \dots & I(N,N) \end{bmatrix}$$

$$\begin{cases} I(j+1,1) = \frac{1}{\Gamma(2)} c_{0j}, & j \ge 0, \\ I(j+1,i+1) = i\sum_{k=0}^{i} (-1)^{i-k} \frac{(i+k-1)!2^{2k}\Gamma(k+1)}{(i-k)!(2k)!(b)^k\Gamma(k+2)} c_{kj}, & i \ge 1,\ j \ge 0, \end{cases}$$

$$c_{kj} = \begin{cases} \frac{b^{k+1}\Gamma(k+\frac{3}{2})}{\sqrt{\pi}\Gamma(k+2)}, & j = 0, \\ \frac{jb^{k+1}}{\sqrt{\pi}} \sum_{r=0}^{j} (-1)^{j-r} \frac{(j+r-1)!2^{2r+1}\Gamma(r+k+1+\frac{1}{2})}{(j-r)!(2r)!\Gamma(r+k+2)} & j \ge 1. \end{cases}$$

This formula is for integrals defined over [0,b].

Declaration
$I\_Cby\_1(N,u)$:
Input:
N: integer
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Cby_1_frac()

The fractional integral operational matrix can be obtained with

$$\kappa^\vartheta \phi(x) \simeq I^\vartheta \phi(x),$$

where I^ϑ is the $(N+1)$ by $(N+1)$ Chebyshev (1st kind) operational matrix of Riemann–Liouville fractional integral of order ϑ [21]. in which

$$I^\vartheta_{(N+1)\times(N+1)} = \begin{bmatrix} I^\vartheta(0,0) & I^\vartheta(0,1) & \dots & I^\vartheta(0,N) \\ I^\vartheta(1,0) & I^\vartheta(1,1) & \dots & I^\vartheta(1,N) \\ \vdots & \ddots & & \\ I^\vartheta(N,0) & I^\vartheta(N,1) & \dots & I^\vartheta(N,N) \end{bmatrix},$$

where its elements are filled as

$$\begin{cases} I^{\vartheta}(j+1,1) = \frac{1}{\Gamma(1+\vartheta)} c_{0j}, \\ j \geq 0, \\ I^{\vartheta}(j+1,i+1) = i\sum_{k=0}^{i} (-1)^{i-k} \frac{(i+k-1)!2^{2k}\Gamma(k+1)}{(i-k)!(2k)!(b)^{k}\Gamma(k+\vartheta+1)} c_{kj}, \\ i \geq 1,\ j \geq 0, \end{cases}$$

$$c_{kj} = \begin{cases} \frac{b^{k+1}\Gamma(k+\vartheta+\frac{1}{2})}{\sqrt{\pi}\Gamma(k+\vartheta+1)}, & j = 0, \\ \frac{jb^{k+\vartheta}}{\sqrt{\pi}} \sum_{r=0}^{j} (-1)^{j-r} \frac{(j+r-1)!2^{2r+1}\Gamma(r+k+\vartheta+\frac{1}{2})}{(j-r)!(2r)!\Gamma(r+k+\vartheta+1)}, & j \geq 1. \end{cases}$$

This formula is for integrals defined over $[0,b]$.

Declaration
$I\_Cby\_frac\_1(N,r,u)$:
Input:
N: integer
r :(the integral order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

lgr_Cby_1()

Like Gegenbauer polynomials, only with Chebyshev polynomial points, Generalized Lagrange Chebyshev of first kind can be achieved with the following code function:

Declaration
$lgr\_Cby\_1\_(N,z,points,u)$:
Input:
N: integer
z: [double 1xm]
points: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[MxN]

D_lgr_Cby_1()

By setting $\alpha = \beta = -0.5$ in the formula subsection 2.2.11, the first order derivative of Generalized Chebyshev of 1st kind functions are achieved. For higher order, one must act as like as formula in Eq. (17).

Declaration
$D\_lgr\_Cby\_1(N,points,u,type)$:
Input:
N: integer
points: [double 1xM]
u: (symbolic shifting parameter)
type: can be each $'gauss'$,$'gauss\_rdu\_l'$,$'gauss\_rdu\_f'$), or $'gauss\_lbt'$
Output:
class: double
Dimension:[MxN]

Chebyshev Polynomials(2nd kind)

The Chebyshev Polynomials of 2nd kind are considered as one special case of Gegenbauer polynomials with input parameter $\theta = 1$:

$$U_i(u(x)) = G_i^{(1)}(u(x)), \tag{26}$$

Cby_zeros_2()

For having N zeros of N-th sentence of Chebyshev of the second kind polynomials, we type $Cby\_zeros\_2(N)$. This function is returning the same values when one types $jacobi\_zeros(N,0.5,0.5)$.

Declaration
$Cby\_zeros\_2(N)$:
Input:
N: integer
Output:
class: double
Dimension:[Nx1]

Cby_2_ ()

Setting $\theta = 1$ in Gegenbauer polynomials, one can define a matrix filled with the Chebyshev functions of the 2nd kind:

$$Cby\_2\_(N,k,z,u) = Gnbr\_(N,1,k,z,u). \tag{27}$$

This can be coded as

Declaration
$Cby\_2(N,k,z,u)$:
Input:
N: integer
k (derivative order): integer
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

Cby_2_frac()

The fractional derivatives of Chebyshev functions of the Second kind are obtained as in regards with its relation with Gegenbauer functions as follows:

$$Cby\_frac\_2(N,\vartheta,z,u) = Gnbr\_frac(N,1,\vartheta,z,u). \quad (28)$$

where ϑ is the fractional derivative order.

Declaration
$Cby\_2\_frac(N,nu,z,u)$:
Input:
N: integer
nu (derivative order): double
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

D_Cby_2()

As we mentioned the relation between Gegenbauer and Chebyshev (2nd kind) functions in 26, for derivative operational matrix of Chebyshev (2nd kind), we can write

$$D\_Cby\_2(N,u) = D\_Gnbr(N,1,u)$$

Declaration
$D\_Cby\_2(N,u\_)$:
Input:
N: integer
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

D_Cby_2_frac()

Due to the relation between Gegenbauer and Chebyshev (2nd kind) functions in 26, for fractional derivative operational matrix of Chebyshev (2nd kind) we have:

$$D\_Cby\_frac\_2(N,\nu,u) = D\_Gnbr\_frac(N,1,\nu,u)$$

where ν is fractional derivative order.

Declaration
$D\_Cby\_frac\_2(N,r,u\_)$:
Input:
N: integer
r :(the derivative order) double
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Cby_2()

The integral operational matrix Chebyshev functions of the 2nd kind are defined as follows:

$$I\_Cby\_2(N,u) = I\_Gnbr(N,1,u)$$

Declaration
$I\_Cby\_2(N,u)$:
Input:
N: integer
theta_ : double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Cby_2_frac()

Like before, for the fractional integral operational matrix of 2nd kind Chebyshev functions, we write:

$$I\_Cby\_frac\_2(N,\vartheta,u) = I\_Gnbr\_frac(N,1,\vartheta,u)$$

Declaration
$I\_Cby\_frac\_2(N,r,u)$:
Input:
N: integer
r :(the integral order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

Igr_Cby_2()

The Generalized Lagrange Chebyshev of the 2nd kind is obtained with the following function:

Declaration
$lgr\_Cby\_2\_(N,z,points,u)$:
Input:
N: integer
z: [double 1xm]
points: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[MxN]

D_lgr_Cby_2()

With $\alpha=\beta=0.5$ in formula subsection 2.2.11, the first order derivative of Generalized Chebyshev of 2nd kind functions are obtained.

Declaration
$D\_lgr\_Cby\_2(N,points,u,type)$:
Input:
N: integer
points: [double 1xM]
u: (symbolic shifting parameter)
type: can be each $'gauss',' gauss\_rdu\_l',' gauss\_rdu\_f')$, or $'gauss\_lbt'$
Output:
class: double
Dimension:[MxN]

Chebyshev Polynomials(3rd kind)

The Chebyshev polynomials of the 3rd kind can be obtained with considering $\alpha=-0.5$ and $\beta=0.5$ in Jacobi polynomials.

$$V_i(u(x))=\frac{2^{2i}}{\binom{2i}{i}}J_i^{-\frac{1}{2},\frac{1}{2}}(u(x)), \tag{29}$$

Cby_zeros_3()

The function $Cby\_zeros\_3(N)$ gives the same result when we call $jacobi\_zeros(N,-0.5,_0.5)$. These functions with these input values are representing the N zeros of N-th sentence of Chebyshev of the third kind polynomials.

Declaration
$Cby\_zeros\_3(N)$:
Input:
N: integer
Output:
class: double
Dimension:[Nx1]

Cby_3_ ()

By seting $\alpha=-0.5$ and $\beta=0.5$ in Jacobi polynomials, Chebyshev function matrix of the 3rd kind filled with filled with the help of the following relation:

$$Cby\_3\_(N,k,z,u)=\left[\frac{2^{2*0}}{\binom{2*0}{0}} \quad \frac{2^{2*1}}{\binom{2*1}{1}} \quad \cdots \quad \frac{2^{2*N}}{\binom{2N}{N}}\right]\circ jacobi\_(N,-0.5,0.5,k,z,u). \tag{30}$$

Declaration
$Cby\_3(N,k,z,u)$:
Input:
N: integer
k (derivative order): integer
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

Cby_3_frac()

The fractional derivatives of Chebyshev functions of the third kind are obtained as:

$$Cby\_frac\_3(N,\vartheta,z,u)=\left[\frac{2^{2*0}}{\binom{2*0}{0}} \quad \frac{2^{2*1}}{\binom{2*1}{1}} \quad \cdots \quad \frac{2^{2*N}}{\binom{2N}{N}}\right]\circ jacobi\_(N,-0.5,0.5,\vartheta,z,u), \tag{31}$$

where ϑ is the fractional derivative order.

Declaration
$Cby\_3\_frac(N,nu,z,u)$:
Input:
N: integer
nu (derivative order): double
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

D_Cby_3()

Similar to subsection 2.2.5 for obtaining the operational matrix of derivative, we write $\phi^T D^{(i)}=\varphi^T$, where $\phi(x)=[J_0^{\alpha,\beta},...,J_N^{\alpha,\beta}]$, $\varphi(x)=[\frac{dJ_0^{\alpha,\beta}(u)}{dx},...,\frac{dJ_N^{\alpha,\beta}(u)}{dx}]$. Therefore,

$$S_N^T\circ\phi^T\times D\_Cby\_3(N,u)=S_N^T\circ\varphi^T,$$

in other words,

$$D\_Cby\_3(N,u)=Diag(1./S_N)D\_jacobi(N,-0.5,0.5,u)Diag(S_N),$$

where

$$S_N = \left[\frac{2^{2*0}}{\binom{2*0}{0}} \quad \frac{2^{2*1}}{\binom{2*1}{1}} \quad \cdots \quad \frac{2^{2*N}}{\binom{2N}{N}} \right].$$

Declaration
$D\_Cby\_3(N, u\_)$:
Input:
N: integer
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

D_Cby_3_frac()

The fractional derivative operational matrix of Chebyshev functions of the 3rd kind, we imagine that $\phi^T D^{(i)} = \varphi^T$, where $\phi(x) = [V_0(u(x)), ..., V_N(u(x))]$, $\varphi(x) = [\frac{dV_0(u(x))}{dx}, ..., \frac{dV_N(u(x))}{dx}]$. Therefore,

$$S_N^T \circ \phi^T \times D\_Cby\_frac\_3(N, \nu, u) = S_N^T \circ \varphi^T,$$

in other words,

$$D\_Cby\_frac\_3(N, \nu, u) =$$
$$Diag(1./S_N) D\_jacobi\_frac(N, -0.5, 0.5, \nu, u) Diag(S_N),$$

where ν is fractional derivative order.

Declaration
$D\_Cby\_frac\_3(N, r, u\_)$:
Input:
N: integer
r :(the derivative order) double
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Cby_3()

In order to achieve the integral operational matrix of Chebyshev functions of the 3rd kind, like preceding subsections, we need to:

$$I\_Cby\_3(N, u) =$$
$$Diag(1./S_N) I\_jacobi(N, -0.5, 0.5, u) Diag(S_N).$$

Declaration
$I\_Cby\_3(N, , u)$:
Input:
N: integer
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Cby_frac_3()

The fractional integral operational matrix of 3rd kind Chebyshev functions, we write:

$$I\_Cby\_frac\_3(N, \vartheta, u) =$$
$$Diag(1./S_N) I\_jacobi\_frac(N, -0.5, 0.5, \vartheta, u) Diag(S_N).$$

Declaration
$I\_Cby\_frac\_3(N, r, u)$:
Input:
N: integer
r :(the integral order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

Igr_Cby_3()

Like other Generalized Lagrange functions, the Generalized Lagrange Chebyshev of the 3rd kind is obtained with the following function:

Declaration
$lgr\_Cby\_3\_(N, z, points, u)$:
Input:
N: integer
z: [double 1xm]
points: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[MxN]

D_lgr_Cby_3()

Having $\alpha = -0.5$, $\beta = 0.5$, the first order derivative of Generalized Chebyshev of 3rd kind functions are coded as follows:

Declaration
$D\_lgr\_Cby\_3(N, points, u, type)$:
Input:
N: integer
points: [double 1xM]
u: (symbolic shifting parameter)
type: can be each $'gauss', 'gauss\_rdu\_l', 'gauss\_rdu\_f'$), or $'gauss\_lbt'$
Output:
class: double
Dimension:[MxN]

Chebyshev Polynomials(4th kind)

The Chebyshev polynomials of the 4th kind can be obtained with considering $\alpha = 0.5$ and $\beta = -0.5$ in Jacobi polynomials.

$$W_i(u(x)) = \frac{2^{2i}}{\binom{2i}{i}} J_i^{\frac{1}{2},-\frac{1}{2}}(u(x)), \tag{32}$$

Cby_zeros_4()

The N zeros of N-th sentence of Chebyshev of the forth kind polynomials are being obtained with $Cby\_zeros\_4(N)$. Its result is exactly the same as $jacobi\_zeros(N, 0.5, -0.5)$.

Declaration
$Cby\_zeros\_4(N)$:
Input:
N: integer
Output:
class: double
Dimension:[Nx1]

Cby_4_ ()

By setting $\alpha = 0.5$ and $\beta = -0.5$ in Jacobi polynomials, Chebyshev function matrix of the 4th kind can be filled with

$$Cby\_4\_(N,k,z,u) = \left[\frac{2^{2*0}}{\binom{2*0}{0}} \quad \frac{2^{2*1}}{\binom{2*1}{1}} \quad \cdots \quad \frac{2^{2*N}}{\binom{2N}{N}}\right] \circ jacobi\_(N, 0.5, -0.5, k, z, u). \tag{33}$$

Declaration
$Cby\_4(N,k,z,u)$:
Input:
N: integer
k (derivative order): integer
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

Cby_4_frac()

The fractional derivatives of Chebyshev functions of the forth kind are obtained as:

$$Cby\_frac\_4(N,\vartheta,z,u) = \left[\frac{2^{2*0}}{\binom{2*0}{0}} \quad \frac{2^{2*1}}{\binom{2*1}{1}} \quad \cdots \quad \frac{2^{2*N}}{\binom{2N}{N}}\right] \circ jacobi\_(N, 0.5, -0.5, \vartheta, z, u). \tag{34}$$

where ϑ is the fractional derivative order.

Declaration
$Cby\_4\_frac(N,nu,z,u)$:
Input:
N: integer
nu (derivative order): double
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

D_Cby_4()

Similar to 2.2.5 for obtaining the operational matrix of derivative, we write $\phi^T D^{(i)} = \varphi^T$, where $\phi(x) = [J_0^{\alpha,\beta}, ..., J_N^{\alpha,\beta}]$, $\varphi(x) = [\frac{dJ_0^{\alpha,\beta}(u)}{dx}, ..., \frac{dJ_N^{\alpha,\beta}(u)}{dx}]$. Therefore,

$$D\_Cby\_4(N,u) = Diag(1./S_N) D\_jacobi(N, 0.5, -0.5, u) Diag(S_N),$$

where S_N is defined in subsection 3.4.4.

Declaration
$D\_Cby\_4(N,u\_)$:
Input:
N: integer
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

D_Cby_4_frac()

The fractional derivative operational matrix of Chebysheve functions of the 4th kind, are computed by the following assumptions.
$\phi^T D^{(i)} = \varphi^T$, where $\phi(x) = [U_0(u(x)), ..., U_N(u(x))]$, $\varphi(x) = [\frac{dU_0(u(x))}{dx}, ..., \frac{dU_N(u(x))}{dx}]$, therefore,

$$S_N^T \circ \phi^T \times D\_Cby\_frac\_4(N,\nu,u) = S_N^T \circ \varphi^T,$$

in other words,

$$D\_Cby\_frac\_4(N,\nu,u) = Diag(1./S_N) D\_jacobi\_frac(N, 0.5, -0.5, \nu, u) Diag(S_N),$$

where ν is fractional derivative order.

Declaration
$D\_Cby\_frac\_4(N,r,u\_)$:
Input:
N: integer
r :(the derivative order) double
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Cby_4()

In order to achieve the integral operational matrix of Chebysheve functions of the 4th kind, like preceding subsections, we need to:

$$I\_Cby\_4(N,u) = Diag(1./S_N)I\_jacobi(N,0.5,-0.5,u)Diag(S_N).$$

Declaration
$I\_Cby\_4(N,u)$:
Input:
N: integer
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Cby_frac_4()

The fractional integral operational matrix of 4th kind Chebyshev functions, we write:

$$I\_Cby\_frac\_4(N,\vartheta,u) = Diag(1./S_N)I\_jacobi\_frac(N,0.5,-0.5,\vartheta,u)Diag(S_N).$$

Declaration
$I\_Cby\_frac\_4(N,r,u)$:
Input:
N: integer
r :(the integral order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

lgr_Cby_4()

Generalized Lagrange Chebyshev functions of the 4th kind is obtained with the following function code:

Declaration
$lgr\_Cby\_4\_(N,z,points,u)$:
Input:
N: integer
z: [double 1xm]
points: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[MxN]

D_lgr_Cby_4()

Having $\alpha = 0.5$, $\beta = -0.5$, the first order derivative of Generalized Chebyshev of 4th kind functions are being coded as follows:

Declaration
$D\_lgr\_Cby\_4(N,points,u,type)$:
Input:
N: integer
points: [double 1xM]
u: (symbolic shifting parameter)
type: can be each $'gauss',' gauss\_rdu\_l',' gauss\_rdu\_f'$), or $'gauss\_lbt'$
Output:
class: double
Dimension:[MxN]

Legendre Polynomials

The Legendre functions are equal to Gegenbauer functions with $\theta = 0.5$:

$$P_i(u(x)) = G_i^{(\frac{1}{2})}(u(x)). \tag{35}$$

Lgdr_zeros()

The Legendre polynomial of order N always produces the N symmetric zeros in addition to 0. These zeros are reached with calling $Lgdr\_zeros(N)$. Its result is exactly the same as $jacobi\_zeros(N,0,0)$ or $Gnbr\_zeros(N,0.5)$.

Declaration
$Lgdr\_zeros(N)$:
Input:
N: integer
Output:
class: double
Dimension:[Nx1]

Lgdr_ ()

As said the Legendre function matrix is also determined with the aid of Gegenbauer functions with $\theta = 0.5$:

$$Lgdr\_(N,k,z,u) = Gnbr\_(N,0.5,k,z,u). \tag{36}$$

Declaration
$Lgdr\_(N,k,z,u)$:
Input:
N: integer
k (derivative order): integer
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

Lgdr_frac()

The fractional derivatives of Chebyshev functions of the forth kind are obtained as:

$$Lgdr\_(N,\vartheta,z,u) = Gnbr\_(N,0.5,\vartheta,z,u). \tag{37}$$

where ϑ is the fractional derivative order.

Declaration
$Lgdr\_frac(N,nu,z,u)$:
Input:
N: integer
nu (derivative order): double
z: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[Mx(N+1)]

D_Lgdr()

As we mentioned the relation between Gegenbauer and Legendre functions, for derivative operational matrix of Legendre, we write

$$D\_Lgdr(N,u) = D\_Gnbr(N,0.5,u).$$

Declaration
$D\_Lgdr(N,u\_)$:
Input:
N: integer
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

D_Lgdr_frac()

For fractional derivative operational matrix of Legendre, we write

$$D\_Lgdr\_frac(N,\nu,u) = D\_Gnbr\_frac(N,0.5,\nu,u).$$

Declaration
$D\_Lgdr\_frac(N,r,u\_)$:
Input:
N: integer
r :(the derivative order) double
u_: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Lgdr()

The integral operational matrix of Legendre functions, we have:

$$I\_Lgdr(N,u) = I\_Gnbr(N,0.5,u),$$

Declaration
$I\_Lgdr(N,u)$:
Input:
N: integer
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

I_Lgdr_frac()

The fractional integral operational matrix of Legendre functions, we have:

$$I\_Lgdr\_frac(N,\vartheta,u) = I\_Gnbr\_frac(N,0.5,\vartheta,u),$$

where ϑ is the fractional integral order.

Declaration
$I\_Lgdr\_frac(N,r,u)$:
Input:
N: integer
r :(the integral order) double
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[(N+1)x(N+1)]

lgr_Lgdr()
Generalized Lagrange Legendre functions are being coded as:

Declaration
$lgr\_Lgdr\_(N, z, points, u)$:
Input:
N: integer
z: [double 1xm]
points: [double 1xM]
u: (symbolic shifting parameter)
Output:
class: double
Dimension:[MxN]

D_lgr_Lgdr()
If $\theta = 0.5$ in subsection 3.1.9, then the first order derivative of Generalized Legendre functions are as follows:

Declaration
$D\_lgr\_Lgdr(N, points, u, type)$:
Input:
N: integer
points: [double 1xM]
u: (symbolic shifting parameter)
type: can be each $'gauss', 'gauss\_rdu\_l', 'gauss\_rdu\_f'$), or $'gauss\_lbt'$
Output:
class: double
Dimension:[MxN]

4. Conclusion and Future

Over the recent years many researchers have been interested in studying the properties of different equations and providing robust and accurate analytical and numerical methods for solving these equations and recorded their work in their published papers: what we have seen at this moment is to see and gather their works in one single work. Moreover, some of the high-qualified state-of-the-art methods and landmark works are done and analyzed perfectly that let us code them in the best shape that is readable and understandable, and specifically, beneficial.
Due to shortcomings of current used packages expressed earlier SPSMAT, lo and behold, was created to address some of these issues. For the strength and liability of using this package, we offered such easy but important examples with their relevant results.

At the and of this paper, we profoundly invite others to develop their works and add their relevant functions and thoughts into this open source package. Don't hesitate to request for a *commit* operation in Github for contribution.

Acknowledgments

In the end we owe some researchers a thank who were at our disposal when we were in dire need. We thank profusely these scientists named Dr. Mehdi Delkhosh, Dr. Saeed Kazem and esteemed Phd student Mohammad Hemami who helped us to collect information about this work.

So long and thanks for all the beloved scientists who assist us write this guidance. I couldn't have done it without them.

References

[1] Trefethen LN, Birkisson A and Driscoll TA. Exploring ODEs. SIAM. 2017 (157)

[2] Doha EH, Bhrawy AH and Ezz-Eldien SS. "A new Jacobi operational matrix: an application for solving fractional differential equations." Applied Mathematical Modeling 36.10 (2012): 4931-4943.

[3] Doha EH and Bhrawy AH. "Efficient spectral-Galerkin algorithms for direct solution of fourth-order differential equations using Jacobi polynomials." Applied Numerical Mathematics 58.8 (2008): 1224-1244.

[4] Parand K, Latifi S, Delkhosh M, and Moayeri MM. "Generalized Lagrangian Jacobi Gauss collocation method for solving unsteady isothermal gas through a micro-nano porous medium." The European Physical Journal Plus 133.1 (2018): 28.

[5] Parand K, Latifi S, Moayeri MM and Delkhosh M. "Generalized Lagrange Jacobi Gauss-Lobatto (GLJGL) Collocation Method for Solving Linear and Nonlinear Fokker-Planck Equations." Communications in Theoretical Physics 69.5 (2018): 519.

[6] Delkhosh M, and Parand K. "Generalized Pseudospectral Method: Theory and Application", Submitted. 2018.

[7] Garg D, Patterson M, Hager WW, Rao AV, Benson DA and Huntington GT. "A unified framework for the numerical solution of optimal control problems using pseudospectral methods." Automatica 46.11 (2010): 1843-1851.

[8] Doha EH and Bhrawy AH. Efficient spectral-Galerkin algorithms for direct solution of fourth-order differential equations using Jacobi polynomials. Applied Numerical Mathematics. 2008;58(8):1224-44.

[9] Doha EH, Bhrawy AH and Ezz-Eldien SS. A Chebyshev spectral method based on operational matrix for initial and boundary value problems of fractional order. Computers and Mathematics with Applications. 2011 ;62(5):2364-73.

[10] Lakestani, M, Dehghan and M. Numerical solution of Fokker-Planck equation using the cubic B-Spline scaling functions. Numer. Meth. Part. D. E. 2009; 25(2): 418–429

[11] Abdelkawy MA, Amin AZ, Bhrawy AH, Machado JA and Lopes AM. Jacobi collocation approximation for solving multi-dimensional volterra integral equations. Inter-

national Journal of Nonlinear Sciences and Numerical Simulation. 2017;18(5):411-25.

[12] Stoer J and Bulirsch R. Introduction to numerical analysis. Springer Science and Business Media; 2013.

[13] Doha EH, Bhrawy AH and Ezz-Eldien SS. A new Jacobi operational matrix: an application for solving fractional differential equations. Applied Mathematical Modelling. 2012 Oct 1;36(10):4931-43.

[14] Shen J, Tang T and Wang LL. Spectral Methods Algorithms, Analysis and Applications. 2011.

[15] Eslahchia MR, Dehghan M and Ahmadi S. The general Jacobi matrix method for solving some nonlinear ordinary differential equations 36(8) 2012.

[16] Kazem S. An integral operational matrix based on Jacobi polynomials for solving fractional-order differential equations Applied Mathematical Modelling.2013. 37 (3), 1126-1136.

[17] Delkhosh M and Parand K. Generalized Pseudospectral Method: Theory and Application, Submitted.

[18] Parand K, Latifi S, Delkhosh M and Moayeri MM. Generalized Lagrangian Jacobi Gauss collocation method for solving unsteady isothermal gas through a micro-nano porous medium The European Physical Journal Plus. 2018; 133 (1), 28.

[19] Parand K, Latifi S, Moayeri MM and Delkhosh M. Generalized Lagrange Jacobi Gauss-Lobatto (GLJGL) Collocation Method for Solving Linear and Nonlinear Fokker-Planck Equations Communications in Theoretical Physics 69 (5), 519. 2018.

[20] Robert SM Algebraic Generating Functions for Gegenbauer Polynomials. 2016.

[21] Fathizadeh E, Ezzati R and Maleknejad K. The Construction of Operational Matrix of Fractional Integration Using the Fractional Chebyshev Polynomials. International Journal of Applied and Computational Mathematics. 2017 ;3(1):387-409.

[22] Lakhani KR, Von Hippel E. How open source software works:free user-to-user assistance. InProduktentwicklung mit virtuellen Communities 2004; 303-339.