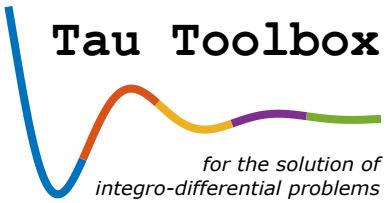
Tau Toolbox Technical Report **TR-1**
available at www.fc.up.pt/tautoolbox



Tau method: an introduction

J. M. A. Matos
M. Trindade
P. B. Vasconcelos

Last update: July 2016 (Tau Toolbox 1.0)
Previous updates: –

Tau method: an introduction

J. M. A. Matos
CMUP and Polytechnic of Porto - School of Engineering
M. Trindade
CMUP and University of Porto - Faculty of Sciences
P. B. Vasconcelos
CMUP and University of Porto - Faculty of Economics

Abstract

This report introduces the Lanczos' tau method to compute a polynomial approximate solution of differential problems. Furthermore, it also explains the first steps on how to use the `Tau Toolbox` to solve the above mentioned problem.

1 Introduction

The tau method, introduced by Cornelius Lanczos (Lanczos, 1938), is a spectral method originally developed to compute a polynomial that approximates the solution of a linear ordinary differential problem with polynomial coefficients. The method has been used since then and extended to problems with non polynomial coefficients, to functional and to nonlinear differential equations, among others. This widespread was only possible from the pioneering work of Ortiz and Samara (1981) with the introduction of an algebraic formulation of the method.

The `Tau Toolbox` is a project to aggregate all these contributions, to enhance the use of the method by developing more stable algorithms and to offer efficient implementations of its algebraic formulation. It is able to solve various integro-differential problems, linear and nonlinear, with initial and/or boundary or others conditions, working with the most common polynomial orthogonal basis.

Non expert can now profit from this spectral method and its solutions properties. On the other hand, experts can easily analyze new problems by exploring the large set of building blocks functions provided.

2 Overview of the tau method

In what follows, we assume that the solution of the differential problem to be solved can be expressed by an orthogonal polynomial series development. Furthermore, the tau method is introduced considering linear differential equations, with polynomial coefficients.

In the tau method sense, we get an $(n-1)^{th}$ degree polynomial approximation y_n to the differential problem's solution y by imposing that y_n solves exactly the differential problem with a polynomial perturbation term τ_n in the differential equation, or system of differential equations. To achieve good minimization properties for the error, τ_n is projected onto an orthogonal polynomial basis.

The `Tau Toolbox` is based on the so called operational version of the tau method, introduced by Ortiz and Samara (1981). The differential equation together with initial and/or boundary conditions, expressed in an orthogonal polynomial basis, are casted onto an infinite algebraic system of equations, relating the series coefficients. The approximate solution y_n, can be obtained solving a truncated algebraic linear system of order n. This truncation gives rise to an approximate solution in the tau sense, that is, the residual polynomial τ_n in the differential equation has the maximum approximation order for an $(n-1)^{th}$ degree polynomial.

If the differential equation is non linear, an usual procedure is to build a succession of linear equations in a linearization process (we refer the reader to Matos et al. (2016)). Other generalizations are tackled on other `Tau Toolbox` Technical Reports.

Several studies applying the tau method have been performed to approximate the solution of differential linear and non-linear equations (Ortiz, 1978; Crisci and Russo, 1983; Liu and Pan, 1999), partial differential equations (Namasivayam and Ortiz, 1981; Ortiz and Dinh, 1987; Matos et al., 2004) and integro-differential equations (Pour-Mahmoud et al., 2005), among others. Nevertheless, in all these works the tau method is tuned for the approximation of specific problems.

If we are interested in solving an ordinary (or partial) differential equation to high accuracy on a simple domain and if the data problem are smooth, then spectral methods are usually the best tool. With respect to the more usual finite differences or finite elements, spectral methods allow to achieve high order accuracy (Trefethen, 2000). Their important spectral properties is a motivation to improve the tau method with new techniques, and new software applications.

2.1 Preliminaries

Let us consider matrices M and N satisfying

$$\mathcal{X}\mathsf{M}\mathsf{a} = xy, \quad \mathcal{X}\mathsf{N}\mathsf{a} = \frac{d}{dx}y,$$

where $\mathcal{X} = [x^0, x^1, x^2, ...]$ is the powers basis for the polynomials space $\mathbb{P}$ of any non-negative integer degree, and $y = \sum_{i\geq 0} a_i x^i = \mathcal{X}\mathsf{a}$ is a formal series with coefficients $\mathsf{a} = [a_0, a_1, \ldots]^T$. Thus we find, respectively,

$$\mathsf{M} = \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 1 & 0 & & \\ & & 1 & 0 & \\ & & & \ddots & \ddots \end{bmatrix}, \quad \mathsf{N} = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 2 & & \\ & & 0 & 3 & \\ & & & 0 & 4 \\ & & & & \ddots & \ddots \end{bmatrix}.$$

Proposition 1 shows the basic procedure on how to change from the power basis $\mathcal{X}$ to another orthogonal polynomial basis $\mathcal{P}$.

Proposition 1. *Let $\mathcal{P} = [P_0, P_1, P_2, ...]$ be an orthogonal basis, $\mathsf{V}_\mathcal{P}$ the triangular matrix such that $\mathcal{P} = \mathcal{X}\mathsf{V}_\mathcal{P}$ and $\mathsf{a}_\mathcal{P} = \mathsf{V}_\mathcal{P}^{-1}\mathsf{a}$. Then $y = \mathcal{P}\mathsf{a}_\mathcal{P}$, $xy = \mathcal{P}\mathsf{M}_\mathcal{P}\mathsf{a}_\mathcal{P}$ and $\frac{d}{dx}y = \mathcal{P}\mathsf{N}_\mathcal{P}\mathsf{a}_\mathcal{P}$, where*

$$\mathsf{M}_\mathcal{P} = \mathsf{V}_\mathcal{P}^{-1}\mathsf{M}\mathsf{V}_\mathcal{P} \quad \textit{and} \quad \mathsf{N}_\mathcal{P} = \mathsf{V}_\mathcal{P}^{-1}\mathsf{N}\mathsf{V}_\mathcal{P}. \tag{1}$$

Proof. See Ortiz and Samara (1981) □

For theoretical purposes the similarity transformations (1) are established with no restrictions, since, for any orthogonal polynomial basis $\mathcal{P}$, each polynomial P_n has exact degree n, and so $\mathsf{V}_\mathcal{P}$ is regular since it is triangular with non null entries in the main diagonal. For implementation purposes, however, this matrix can be highly ill-conditioned and the computation of (1) must be avoided. Proposition 2 presents an equivalent procedure but numerically more stable to compute $\mathsf{M}_\mathcal{P}$ and $\mathsf{N}_\mathcal{P}$, based on recurrence relations.

Proposition 2. *Let $\mathcal{P} = [P_0, P_1, P_2, \ldots]$ be an orthogonal basis satisfying the recurrence relation $xP_j = \alpha_j P_{j+1} + \beta_j P_j + \gamma_j P_{j-1}$, $j \geq 0$, $P_0 = 1$, $P_{-1} = 0$. The coefficients of $\mathsf{M}_\mathcal{P} = [\mu_{ij}]$ can be obtained by*

$$\begin{cases} \mu_{j+1,j} = \alpha_{j-1}, \quad \mu_{j,j} = \beta_{j-1}, \quad \mu_{j,j+1} = \gamma_{j-1} \\ \mu_{i,j} = 0, \quad |i-j| > 1 \end{cases}, \quad j = 1, 2, \ldots \tag{2}$$

and those of $\mathsf{N}_\mathcal{P} = [\eta_{ij}]$ by

$$\begin{cases} \eta_{i,j+1} = \frac{1}{\alpha_j}\left[\alpha_{i-1}\eta_{j,i-1} + (\beta_i - \beta_j)\eta_{j,i} + \gamma_{i+1}\eta_{j,i+1} - \gamma_j\eta_{j-1,i}\right], \quad i = 0, \ldots, j-1 \\ \eta_{j,j+1} = \frac{1}{\alpha_j}(\alpha_{j-1}\eta_{j,j-1} + 1) \end{cases} \tag{3}$$

for $j = 1, 2, \ldots$.

Proof. See Matos et al. □

2.2 Tau method for linear differential problems

Let $\mathcal{D} = \sum_{k=0}^{\nu} p_k \frac{d}{dx^k}$ represent an order ν linear differential operator acting on $\mathbb{P}$, where $p_k = \sum_{i=0}^{n_k} p_{ki} x^i$ are polynomial coefficients, $n_k \in \mathbb{N}_0$, $p_{k,i} \in \mathbb{R}$ and let $f \in \mathbb{P}$ with finite degree λ. An approximate polynomial solution y_n for the differential problem

$$\begin{cases} \mathcal{D}y = f \\ c_i(y) = s_i,\ i = 1, \dots, \nu \end{cases}, \tag{4}$$

is obtained in the tau sense by solving a perturbed system

$$\begin{cases} \mathcal{D}y_n = f + \tau_n \\ c_i(y_n) = s_i,\ i = 1, \dots, \nu \end{cases}, \tag{5}$$

Problem (4) has a matrix representation given by

$$\begin{cases} \mathsf{C}_{\mathcal{P}} \mathsf{a}_{\mathcal{P}} = \mathsf{s} \\ \mathsf{D}_{\mathcal{P}} \mathsf{a}_{\mathcal{P}} = \mathsf{f}_{\mathcal{P}} \end{cases},$$

where

$$\mathsf{C}_{\mathcal{P}} = [c_{ij}]_{\nu \times \infty} = c_i(P_{j-1}),\ i = 1, \dots, \nu,\ j = 1, \dots,$$

$$\mathsf{D}_{\mathcal{P}} = \sum_{k=0}^{\nu} p_k(\mathsf{M}_{\mathcal{P}}) \mathsf{N}_{\mathcal{P}}^k, \quad p_k(\mathsf{M}_{\mathcal{P}}) = \sum_{i=0}^{n_k} p_{ki} \mathsf{M}_{\mathcal{P}}^i,$$

$\mathsf{a}_{\mathcal{P}} = [a_0, a_1, \dots]^T$ contains the coefficients of the series representation of y in $\mathcal{P}$, $\mathsf{s} = [s_1, \dots, s_\nu]^T$ the right hand side vector of the boundary conditions and $\mathsf{f}_{\mathcal{P}} = [f_0, \dots, f_\lambda, 0, 0, \dots]^T$ the coefficients of the right hand side differential equations on the basis $\mathcal{P}$.

The coefficients $a_{\mathcal{P}}$ satisfies

$$\mathsf{T}_{\mathcal{P}} \mathsf{a}_{\mathcal{P}} = \mathsf{b}_{\mathcal{P}}, \text{ with } \mathsf{T}_{\mathcal{P}} = \begin{bmatrix} \mathsf{C}_{\mathcal{P}} \\ \mathsf{D}_{\mathcal{P}} \end{bmatrix} \text{ and } \mathsf{b}_{\mathcal{P}} = \begin{bmatrix} \mathsf{s} \\ \mathsf{f}_{\mathcal{P}} \end{bmatrix}. \tag{6}$$

Matrix $\mathsf{T}_{\mathcal{P}}$ has an upper trapezoidal structure, with $h = \sup\{|\mathsf{D}P_k - k|,\ k = 0, 1, \dots\}$ non-null sub-diagonals.

Choosing an integer $n \geq \nu + \lambda$, an $(n-1)^{th}$ degree polynomial approximate solution $\mathsf{y}_n = \mathcal{P}_n \mathsf{a}_n$, expressed by the coefficients $\mathsf{a}_n = [a_{n,0}, a_{n,1}, \dots, a_{n,n-1}]$ on $\mathcal{P}_n = [P_0, P_1, \dots, P_{n-1}]^T$, is obtained by truncating system (6) to its first n columns. The resulting system has $n + h$ equations. Restricting this system to its first n equations, a linear system

$$\mathsf{T}_n \mathsf{a}_n = \mathsf{b}_n \tag{7}$$

is obtained with T_n of dimension $n \times n$, which is equivalent to introduce a polynomial residual

$$\tau_n = \mathcal{D}y - \mathcal{D}y_n = \sum_{i=1}^{\nu+h} r_{n,i} \mathcal{P}_{n-\nu+i} \tag{8}$$

in the right hand side. Defining $\hat{\mathcal{P}}_n = [P_{n-\nu+1}, P_{n-\nu+2}, \dots, P_{n+h}]$ and $\mathsf{r}_n = [r_{n,1}, r_{n,2}, \dots, r_{n,\nu+h}]$, then equation (8) can be written as

$$\tau_n = \hat{\mathcal{P}}_n r_n,\ r_n = \mathsf{R}_n \mathsf{a}_n, \tag{9}$$

where R_n is the triangular block of matrix T, restricted to its first n columns and rows $n+1, \dots, n+\nu+h$ (see Figure 1).

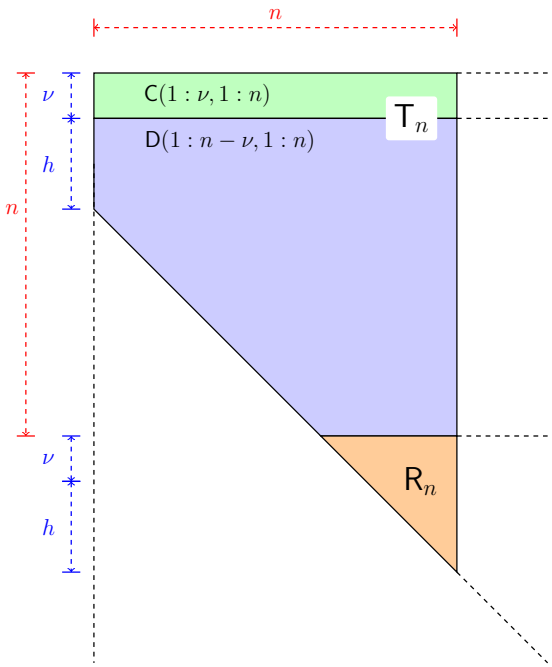


Figure 1: Schematic view for the coefficient matrix on the truncated system (7) and for the residual τ_n

3 Tau method on `Tau Toolbox`

In this section we will briefly explain how to work with `Tau Toolbox` to compute approximate solutions of differential systems of equations using the tau method.

The `Tau Toolbox` is a MATLAB toolbox that uses object-oriented techniques in the programming design.

The function `[x, y] = tau(basis, domain, n)` creates two objects: `x` an independent tau variable and `y` an dependent tau variable. All necessary operations are created from the provided parameters: the `basis` (e.g. Chebyshev of the first and second kind, Legendre, Hermite, Laguerre), the required `domain` and the order `n` of the approximate polynomial solution.

The following example creates the objects to work with a polynomial approximation of degree 4, on the $[-1, 1]$ domain, using the Chebyshev of first kind basis:

```
>> [x, y] = tau('ChebyshevT', [-1 1], 5)

    x =                              y =

    itau with properties:            dtau with properties:

    basis: 'ChebyshevT'              basis: 'ChebyshevT'
    domain: [-1 1]                   domain: [-1 1]
    n: 5                             n: 5
    mat: []                          mat: []
```

By default the same result is obtained just by calling `[x, y] = tau`.

Now if we write $x^2 - 4x^5$, since x is an `itau` object, this is interpreted as the $n \times n$ matrix $\mathsf{M}_{\mathcal{T}}^2 - 4\mathsf{M}_{\mathcal{T}}^5$, where $\mathcal{T}$ is the Chebyshev basis of the first kind and $\mathsf{M}_{\mathcal{T}}$ is the matrix defined by (2) for that basis:

```
>> x^2-4*x^5
ans =

    0.5000   -1.2500    0.2500   -0.6250         0
   -2.5000    0.7500   -1.8750    0.2500   -0.6250
    0.5000   -1.8750    0.5000   -1.2500    0.2500
   -1.2500    0.2500   -1.2500    0.5000   -0.6250
         0   -0.6250    0.2500   -0.6250    0.2500
```

Moreover, $(x^2 - 4x^5)\frac{d^3y}{dx^3}$ is interpreted as $(\mathsf{M}_{\mathcal{T}}^2 - 4\mathsf{M}_{\mathcal{T}}^5)\mathsf{N}_{\mathcal{T}}^3$:

```
>> (x^2-4*x^5)*diff(y,3)
ans =

     0     0     0    12  -240
     0     0     0   -60   144
     0     0     0    12  -360
     0     0     0   -30    48
     0     0     0     0  -144
```

With these objects is thus very easy to tackle any differential problem by the tau method. The following examples illustrate how to use the `Tau Toolbox` in the solution of linear differential problems.

Example 1: a linear second order initial value problem

Using `Tau Toolbox` to approximate the solution of the following differential problem

$$\begin{cases} y''(x) + y(x) = 0, \quad x \in]0,\ 5[\\ y(0) = 0, \quad y'(0) = 1 \end{cases}, \tag{10}$$

can be done using just two functions

`Tau Toolbox` code 1: basic level

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [0 5], 20);
% Call the tausolver function.
a = tausolver(x, y, 'diff(y,2)+y=0', {'y(0)=0';'y''(0)=1'});
```

The sought approximation is a 19^{th} degree polynomial projected on ChebyshevT basis (shifted to $[0,\ 5]$). The `a` vector contains the 20 coefficients of the tau approximation (Figure 2).

Selecting the polynomial basis where to project the solution is simple. For instance, for the Legendre case just perform `[x, y] = tau('Legendre', [0 5], 20);`. Figure 3 shows the distance $|y_{20}(x) - y_{19}(x)|$ for Legendre and ChebyshevT basis.

Remark 1. Function `diff` is a `Tau Toolbox` function that overwrites the one from MATLAB.

Remark 2. The first derivative of y are written as `'y''(0)=1'`, since the initial conditions is introduced as a string.

Remark 3. Note that the output coefficients in vector `a` are ordered from the lowest to the highest degree term.

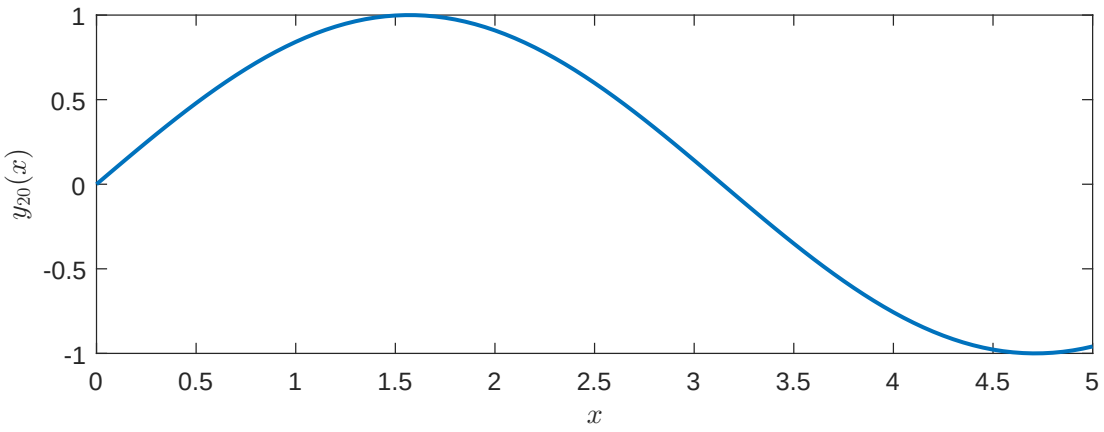


Figure 2: Approximate solution $y_n = \sum_{i=0}^{n-1} a_{n,i} T_i(x) = \mathcal{T}_n \mathsf{a}_n$, $n = 20$.

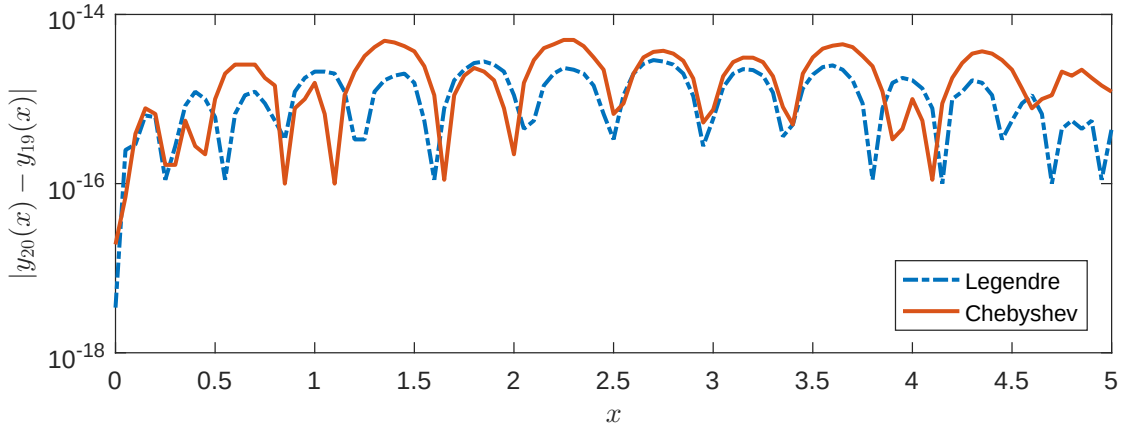


Figure 3: $|y_{20}(x) - y_{19}(x)|$

At an intermediate level, the user can parameterize the `tausolver` function specifying parameters. For instances, if the exact solution is known (in this case $y(x) = \sin(x)$), this can be specified by the parameter `'exact_solution'`, and the solver retrieves a graphic of the absolute error $|y(x) - y_{20}(x)|$ (Figure 4a).

`Tau Toolbox` code 2: intermediate level

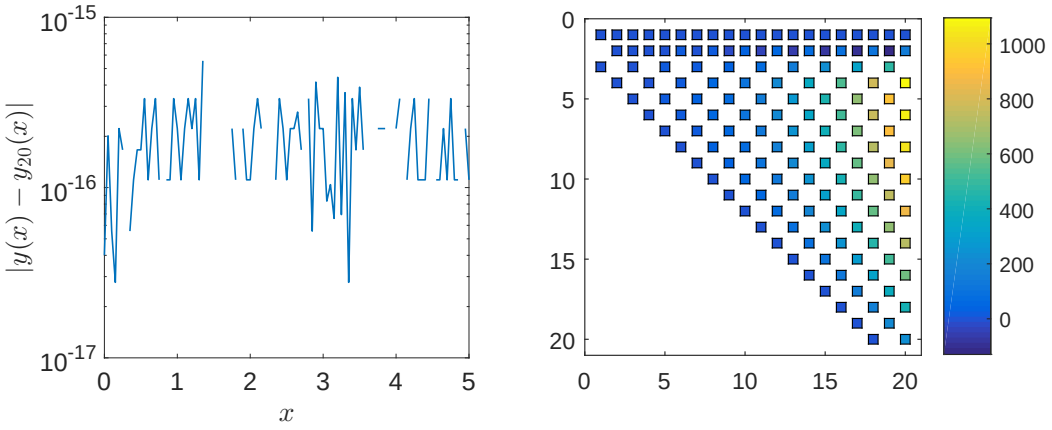
```
% Create tau objects.
[x, y] = tau('ChebyshevT', [0 5], 20);
% Call the tausolver function.
a = tausolver(x, y, 'diff(y,2)+y=0', ...
              {'y(0)=0';'y''(0)=1'}, 'exact_solution', 'sin(x)', 'spy', ...
                  1);
```

The differential problem was translated to the linear system $\mathsf{T}_n \mathsf{a}_n = \mathsf{b}_n$, where

$$\mathsf{T}_n = \begin{bmatrix} \mathsf{C}_\mathcal{T}(1:\nu, 1:n) \\ \mathsf{D}_\mathcal{T}(1:n-\nu, 1:n) \end{bmatrix}, \quad \mathsf{b}_n = [0,\, 1,\, \underbrace{0, \ldots, 0}_{n-\nu}]^T,$$

with $\mathsf{D}_\mathcal{T} = \mathsf{N}_\mathcal{T}^2 + \mathsf{I}$, $\mathsf{C}_\mathcal{T} = [c_{i,j}]$, $c_{1,j} = T_j(0)$, $c_{2,j} = T_j'(0)$, $j = 0, 1, \ldots, n-1$, $n = 20$ and $\nu = 2$.

The `tausolver` function provides a graphical representation of T_n: this is accomplished by setting the binary parameter `'spy'` to 1 (Figure 4b).



(a) error $|\sin(x) - y_{20}(x)|$ (b) spy of T_{20} matrix

Figure 4: Error and spy

More advanced Tau Toolbox users can detail their codes making use of more elaborate functions and explore new approaches. Matrix T_n and vector b_n can be easily accessed using Tau Toolbox function tausystem, and the linear system $\mathsf{T}_n\mathsf{a}_n = \mathsf{b}_n$ can be solved by a specific solver. In the next illustration the Gaussian elimination provided by MATLAB is used.

Tau Toolbox code 3: advanced level

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [0 5], 6);
% Obtain T matrix and b vector.
[T, b] = tausystem(x, y, 'diff(y,2)+y=0',{'y(0)=0';'y''(0)=1'});
% Solve the linear system.
a = T\b;
```

Among the set of functions available at Tau Toolbox (see Tau Toolbox functions) we illustrate that idea with the function orthoval, returning pointwise values of a polynomial from their coefficients in an orthonormal polynomial basis. In Figure 5 we compare orthoval function with the well known polyval MATLAB function. Our evaluation, by avoiding change of basis, is more stable and thus provides a better approximation.

Tau Toolbox code 4: advanced level

```
% Create tau objets.
[x, y] = tau('ChebyshevT', [0 20], 100);
% Call the tausolver function.
a = tausolver(x, y, 'diff(y,2)+y=0', {'y(0)=0';'y''(0)=1'});
% Points for evaluation.
points = linspace(x.domain(1), x.domain(2));
% Change of basis.
a_bar = orth2powmatrix(y)*a;
% Polynomial evaluation (power basis).
yn_bar = polyval(a_bar(end:-1:1), points);
% Polynomial evaluation (orthogonal basis).
yn = orthoval(x, points, 'coef', a);
% Plot the error (with polyval)
semilogy(points, abs(yn_bar-sin(points))); hold on;
% Plot the error (with orthoval)
semilogy(points, abs(yn-sin(points))); hold off;
```

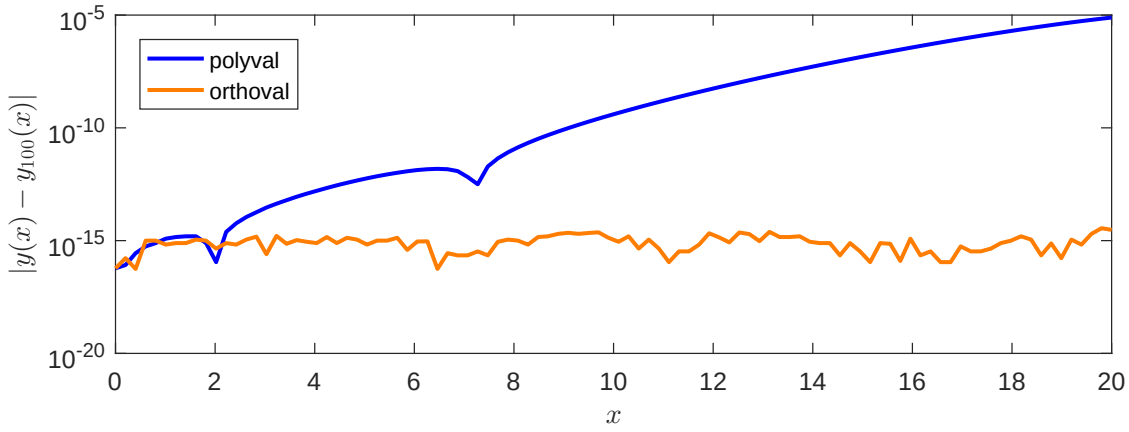


Figure 5: Error using `orthoval` and `polyval` functions

Example 2: a third order equation with multipoint conditions

Let us now solve the differential problem

$$\begin{cases} (x^2+1)y'''(x) - (x^2+3x)y''(x) + 5xy'(x) - 5y = 60x^2 - 10 \quad x \in]-1,\ 1[\\ y(-1) = 4, \quad y'(1) = 2, \quad y''(0) = 0 \end{cases} \tag{11}$$

The problem can be solved by `Tau Toolbox` as:

`Tau Toolbox` code 5: basic level

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [-1 1], 5);
% Call the tausolver function.
a = tausolver(x, y, ...
          ['(x^2+1)*diff(y,3)-(x^2+3*x)*diff(y,2)+5*x*diff(y)-5*y', ...
          ' = 60*x^2-10'], ...
          {'y(-1)=4';'y''(1)=2';'y''''(0)=0'}, ...
          'spy', 1, 'exact_solution', 'x^5-3*x+2');
```

The vector solution a obtained, with $n = 6$, was $a = [2,\ -2.375,\ 0,\ 0.3125,\ 0,\ 0.0625]$ which corresponds to the Chebyshev form of $y(x) = 2T_0(x) - \frac{19}{8}T_1(x) + \frac{5}{16}T_3(x) + \frac{1}{16}T_5(x) = 2 - 3x + x^5$ the exact solution. This illustrate the tau method's property that, with n large enough, we recover the exact solution whenever it is a polynomial.

With `tausolver` function we recover the exact solution even when working with n values considerably greater then the degree of exact solution. Figure 6 shows the absolute errors $|a - a_n|$ observed in the coefficients of the `tausolver` solution a_n with sample values for n.

Example 3: a system of differential equations

In order to illustrate an application to a system of differential equations, the previous example is translated into a system of first order differential equations, as

$$\begin{cases} y_2(x) - y_1'(x) = 0 \\ y_3(x) - y_2'(x) = 0 \\ (x^2+1)y_3'(x) - (x^2+3x)y_2'(x) + 5xy_2(x) - 5y_1 = 60x^2 - 10 \\ y_1(-1) = 4, \quad y_2(1) = 2, \quad y_3(0) = 0 \end{cases} . \tag{12}$$

The `Tau Toolbox` `tausolver` function allows a fast and basic approach to solve problem (12):

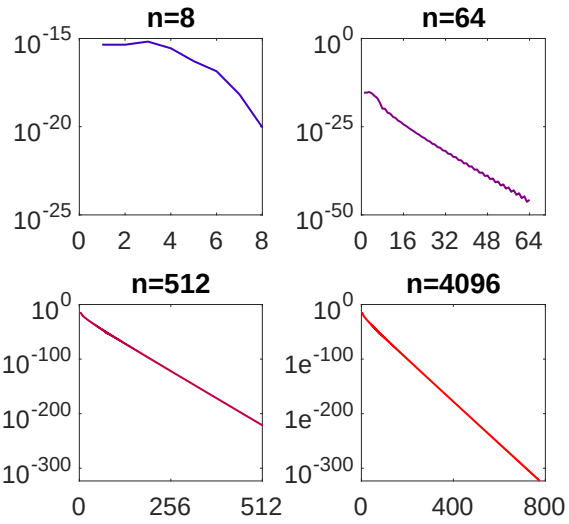
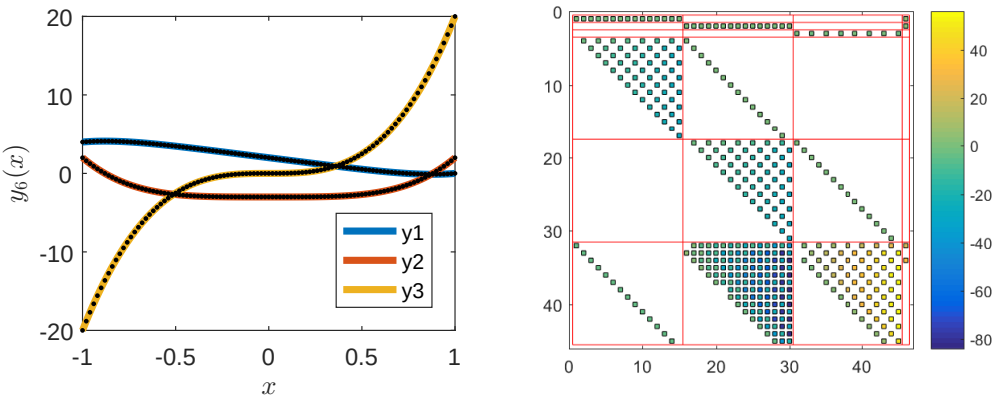


Figure 6: errors $|a - a_n|$, $n = 8,\ 64,\ 512,\ 4096$

`Tau Toolbox` code 6: basic level

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [-1 1], 15);
% Call the tausolver function.
a = tausolver(x, y, {'y2-1*diff(y1)=0'; ...
              'y3-1*diff(y2)=0'; ...
              ['(x^2+1)*diff(y3)-(x^2+3*x)*diff(y2)+5*x*y2-5*y1=', ...
              '60*x^2-10']}, ...
              {'y1(-1)=4';'y2(1)=2';'y3(0)=0'}, ...
              'exact_solution', {'x^5-3*x+2';'5*x^4-3';'20*x^3'}, 'spy', ...
                 1);
```

The shape of $\mathsf{T}_{\mathcal{T}}$ is presented at Figure 7b and the solution plots shown at Figure 7a.



(a) Solution plots (b) Spy of the coefficent matrix

Figure 7: Solution and spy plots for problem (12)

Once the problem (12) is translated into an algebraic linear system $[\mathsf{C}_{\mathcal{P}};\ \mathsf{D}_{\mathcal{P}}]\mathsf{a}_{\mathcal{P}} = \mathsf{b}_{\mathcal{P}}$, where

$$\mathsf{D}_{\mathcal{P}} = \begin{bmatrix} -\mathsf{N}_{\mathcal{P}} & \mathsf{I} & \mathbf{0} \\ \mathbf{0} & -\mathsf{N}_{\mathcal{P}} & \mathsf{I} \\ -5\mathsf{I} & 5\mathsf{M}_{\mathcal{P}}\mathsf{I} - (\mathsf{M}_{\mathcal{P}}^2 + 3\mathsf{M}_{\mathcal{P}})\mathsf{N}_{\mathcal{P}} & (\mathsf{M}_{\mathcal{P}}^2 + \mathsf{I})\mathsf{N}_{\mathcal{P}} \end{bmatrix}$$

translates the differential operator,

$$\mathsf{C}_{\mathcal{P}} = \begin{cases} c_{1,i} = P_i(-1) \\ c_{2,n+i} = P_i(1) \\ c_{3,2n+i} = P_i(0) \end{cases}, \quad i = 0, 1, 2, ..., n-1$$

translates the boundary conditions and

$$\mathsf{b}_{\mathcal{P}} = [\underbrace{4,\ 2,\ 0}_{\nu=3},\ \underbrace{0,\ \ldots,\ 0}_{2(n-1)},\ \mathsf{f}_{\mathcal{P}}]^T.$$

simultaneously the left-hand-side of boundary conditions and of differential equations.

The solution of the system provides

$$y \approx \sum_{i=0}^{n-1} a_{\mathcal{P}i} P_i, \quad y' \approx \sum_{i=0}^{n-1} a_{\mathcal{P}i+n} P_i \quad \text{and} \quad y'' \approx \sum_{i=0}^{n-1} a_{\mathcal{P}i+2n} P_i.$$

More advanced users can explore even more the `Tau Toolbox` to detail their solution approach. The following sequence of codes present, in detail, all the steps to tackle problem (12):

- Define the tau objects and the block matrix:

```
% Create tau objects.
[x, y] = tau('ChebyshevT', [-1 1], 6);

% Building the blocks for compose D.
A = -diff(y);   B = 1*y;                                C = zeros(x.n);
D = zeros(x.n); E = -diff(y);                           F = 1*y;
G = -5*y;       H = 5*x*eye(x.n)-(x^2+3*x)*diff(y);     I = (x^2+1)*diff(y);
```

- Define the boundary conditions and store them in b and T:

```
% Allocating the conditions.
x1 = -1; x2 = 1; x3 = 0;
nu = 3; b = zeros(3*x.n, 1); b(1) = x1^5-3*x1+2;
                             b(2) = 5*x2^4-3;
                             b(3) = 20*x3^3;
```

- Define and store the matrix C

```
T(1,1:y.n) = orthoval(x, x1, 'difforder', 0);
T(2,y.n+1:2*y.n) = orthoval(x, x2, 'difforder', 0);
T(3,2*y.n+1:3*y.n) = orthoval(x, x3, 'difforder', 0);
```

- Define and store the operator at D:

```
% Allocating the truncated blocks.
T(nu+1:y.n-1+nu,1:y.n) = A.mat(1:end-1,:);
T(nu+1:y.n-1+nu,y.n+1:2*y.n) = B.mat(1:end-1,:);
T(nu+1:y.n-1+nu,2*y.n+1:3*y.n) = C(1:end-1,:);

T(y.n+nu:2*(y.n-1)+nu,1:y.n) = D(1:end-1,:);
T(y.n+nu:2*(y.n-1)+nu,y.n+1:2*y.n) = E.mat(1:end-1,:);
T(y.n+nu:2*(y.n-1)+nu,2*y.n+1:3*y.n) = F.mat(1:end-1,:);

T(2*y.n-1+nu:3*(y.n-1)+nu,1:y.n)= G.mat(1:end-1,:);
T(2*y.n-1+nu:3*(y.n-1)+nu,y.n+1:2*y.n)= H.mat(1:end-1,:);
T(2*y.n-1+nu:3*(y.n-1)+nu,2*y.n+1:3*y.n)= I.mat(1:end-1,:);
```

- Right-hand-side

```
% Transforming and allocating the right-hand-side.
p = pow2orth(x, '60*x^2-10');
b(2*y.n-1+nu:3*(y.n-1)+nu) = p(1:end-1);
```

- Compute the tau solution:

```
% Solving the system.
a = T\b; a = reshape(a, y.n, 3);
```

- Results are shown by:

```
% Showing the results.
points = linspace(x.domain(1), x.domain(2), 100);
y1 = orthoval(x, points, 'coef', a(:,1));
y2 = orthoval(x, points, 'coef', a(:,2));
y3 = orthoval(x, points, 'coef', a(:,3));
semilogy(points, abs(y1-(points.^5-3*points+2))); hold on
semilogy(points, abs(y2-(5*points.^4-3)));
semilogy(points, abs(y3-(20*points.^3)));
l=legend('$|y(x)-y_{6}(x)|$','$|y''(x)-y''_{6}(x)|$','$|y''''(x)-y''''_...
    {6}(x)|$');
set(l,'Interpreter','Latex'); xlabel('$x$','Interpreter','Latex')
figure; spy(T)
```

4 Highlights

In this Technical Report:

- the tau method is introduced;
- first insights on the `Tau Toolbox` are provided
- the three level functions of `Tau Toolbox` are presented: basic, intermediate and advanced.

References

M. R. Crisci and E. Russo. An extension of Ortiz recursive formulation of the Tau method to certain linear systems of ordinary differential equations. *Mathematics of computation*, pages 27–42, 1983.

C. Lanczos. Trigonometric interpolation of empirical and analytical functions. *Journal of Mathematics and Physics*, 17(1):123–199, 1938.

K. Liu and C. Pan. The automatic solution to systems of ordinary differential equations by the Tau method. *Computers & Mathematics with Applications*, 38(9):197–210, 1999.

J. Matos, M. J. Rodrigues, and J. C. de Matos. Avoiding similarity transformations in the operational Tau method. *submitted*.

J. Matos, M. J. Rodrigues, and P. B. Vasconcelos. New implementation of the Tau method for pdes. *Journal of computational and applied mathematics*, 164:555–567, 2004.

J. Matos, P. B. Vasconcelos, and M. Trindade. Tau Toolbox for nonlinear ordinary differential systems. Technical Report TR4, CMUP and University of Porto, 2016.

S. Namasivayam and E. L. Ortiz. Best approximation and the numerical solution of partial differential equations with the Tau method. *Portugaliae mathematica*, 40(1):97–119, 1981.

E. Ortiz. On the numerical solution of nonlinear and functional differential equations with the Tau method. In *Numerical treatment of differential equations in applications*, pages 127–139. Springer, 1978.

E. Ortiz and A. P. N. Dinh. Linear recursive schemes associated with some nonlinear partial differential equations in one dimension and the Tau method. *SIAM Journal on Mathematical Analysis*, 18(2): 452–464, 1987.

E. L. Ortiz and H. Samara. An operational approach to the Tau method for the numerical solution of non-linear differential equations. *Computing*, 27(1):15–25, 1981.

J. Pour-Mahmoud, M. Rahimi-Ardabili, and S. Shahmorad. Numerical solution of the system of fredholm integro-differential equations by the Tau method. *Applied Mathematics and Computation*, 168(1):465–478, 2005.

L. N. Trefethen. *Spectral methods in MATLAB*, volume 10. SIAM, 2000.