Solving differential eigenproblems via the spectral Tau method

P. B. Vasconcelos* J. E. Roman† J. M. A. Matos‡

February 7, 2023

Abstract

The spectral Tau method to compute eigenpairs of ordinary differential equations is implemented as part of the `Tau Toolbox` - a numerical library for the solution of integro-differential problems. This mathematical software enables a symbolic syntax to be applied to objects to manipulate and solve differential problems with ease and accuracy. The library is explained in detail and its application to various problems is illustrated: numerical approximations for linear, quadratic, and nonlinear differential eigenvalue problems.

1 Introduction

The spectral Tau method produces, from a truncated series expansion in a complete set of orthogonal functions, a polynomial approximate solution for ordinary differential problems. Contrary to the most common Galerkin method, where the expansion functions must satisfy the boundary conditions, in the Tau method the coefficients of the series are computed by forcing the differential problem to be exact at some selected points and additional conditions are set such that the initial/boundary conditions are exactly satisfied. This approach, introduced by Lanczos, has also been applied to partial differential equations, integral equations, and integro-differential problems, but usually on a specific problem or a class of problems.

Applications of the Tau method to solve eigenproblems can be found, at least, since the 1960 decade. In [7] a recursive version of the Tau method, in terms of canonical polynomials, was used to calculate approximations of the eigenpairs in linear differential problems. In [27] Orszag solved the Orr-Sommerfeld problem using the Tau method together with the QR matrix eigenvalue algorithm. In [29, 4] the operational approach of the Tau method was applied in the solution of differential eigenproblems with nonlinear parameter. Following these pioneering works, the focus turned to eliminating spurious eigenvalues in the Tau method [40, 12, 24, 8, 6] and improving the method's accuracy [9], in particular for certain classes of problems.

Our work follows a different direction as we present a tool to calculate eigenpairs to generic problems, of any order, with any orthogonal polynomial basis. While solving effectively certain classes of differential eigenvalue problems is a relevant goal, the main contribution of this paper is providing a general framework for the solution of such problems and a stable numerical implementation of the Tau method, ensuring accurate and fast approximate solutions.

The complexity of such a task in delivering this tool appears to lie far beyond the tasks to which the literature has been addressing, which is why the Tau method has not been explored

*Faculdade Economia da Universidade do Porto, Rua Dr. Roberto Frias, 4299-464 Porto (Portugal) (`pjv@fep.up.pt`).

†DSIC, Universitat Politècnica de València, València, D. Sistemes Informàtics i Computació, Camí de Vera, s/n, E-46022 València (Spain) (`jroman@dsic.upv.es`).

‡Instituto Superior de Engenharia do Porto, Rua Dr. António Bernardino de Almeida, 431, 4249-015 Porto (Portugal) (`jma@isep.ipp.pt`).

as it should. The implementation relies on mathematical results from approximation theory to efficiently build the algorithms and on object-oriented programming design to deal with numerical objects as similarly as possible to symbolic manipulation.

The `Tau Toolbox`, based on the Tau method and implemented in `MATLAB`/ `Octave`, computes all the eigenvalues (or a subset) and eigenfunctions, if required, efficiently (the need for finding many eigenvalues has been reported, e.g. in [5, 34]). The evidence for the effectiveness of the tool as well as the simplicity of its use is highlighted in the Listing 1, where the approximate solution for the problem $y^{(4)}(x) - \lambda y(x) = 0$, $x \in [0, 1]$, with boundary conditions $y(0) = y(1) = 0$, $y'(0) = 0$, $y''(1) = 0$, is achieved with just a solver function, `tau.solve`. The user only needs to specify the `problem`, the `domain` and the `conditions`. By default the polynomial approximation is of degree 32 relying on the Chebyshev basis of the first kind. These defaults can be easily changed through options.

```
% set the problem
equation = "diff(y,4)=lambda*y";
domain = [0, 1];
conditions = {"y(0)=0"; "y(1)=0"; "y'(0)=0"; "y''(1)=0"};
problem = tau.problem(equation, domain, conditions);

% solve the problem
lambda = tau.solve(problem);
```

Listing 1: Code to compute the solution of a differential eigenvalue problem using `Tau Toolbox` (see Example 1).

This software tool has been greatly inspired by the performant Chebfun [11]. This `MATLAB` library provides an easy use (for the user perspective) in solving various differential problems using a Chebyshev spectral collocation approach. It can compute the whole set of eigenvectors approximating eigenfunctions while providing automatic estimation and control of the eigenvalue error. This is a relevant feature that will be considered in a new release of `Tau Toolbox`. For now, the order of approximation is fixed at the beginning.

The `Tau Toolbox` implements the spectral Tau method in the spirit originally proposed by Lanczos [20], whereas Chebfun uses a pseudospectral approach. The main conceptual difference between Tau and rectangular collocation (used in Chebfun) is that, in the former, differentiation and integration are simple and sparse, whereas multiplication is messier and dense; in the latter case, the situation is exactly the opposite [10]. The `Tau Toolbox` implementation tackles nonlinear differential problems by linearizing first and discretizing afterwards likewise Chebfun, which avoids the loss of richness in doing the opposite. Stability in `Tau Toolbox` is ensured by profusely using explicit relations for computing the operational matrices and working only within the orthogonal basis selected (no similarity transformations are performed nor evaluations on another basis).

Other software packages have been developed over time to solve differential eigenproblems, namely SLEIGN2 [2], SLEDGE [30], SL02F [32], and MATSLISE [21].

This paper is organized as follows. We first offer an overview of the spectral Tau method, provide some insights on the numerical implementation of `Tau Toolbox` and present the tackled differential eigenproblems. In section 3, we construct the methodology to build the algebraic formulation and shed light on the numerical techniques used in the solution process. In the fourth section, we illustrate both the use of the numerical library and its accuracy by solving a set of various differential problems that require the solution of linear, polynomial and nonlinear algebraic eigenvalue problems.

2 Preliminaries

2.1 The spectral Tau method

Spectral methods look for an approximate solution in a space of trial functions imposing the vanishing of the orthogonal projection of the residual on a finite dimensional subspace spanned by test functions. While for Galerkin, the most widely used approach, the two sets of functions are the same, for the Tau method test functions differ from trial functions (Petrov-Galerkin approch). It is more difficult for the former to cope with complicated boundary conditions since the trial functions must satisfy the boundary conditions. The latter works by adding more terms into the expansion.

The Tau method was originally proposed by C. Lanczos [20], initially to solve linear ordinary differential equations, and afterwards extended to a broad set of problem categories. It became more well-known from the work of Ortiz and Samara in the eighties [28, 29].

Let $D : \mathbb{E} \mapsto \mathbb{F}$ be an order $\nu \geq 0$ differential operator, $\mathbb{E}$ and $\mathbb{F}$ appropriate function spaces, $f \in \mathbb{F}$ and let $c_i : \mathbb{E} \mapsto \mathbb{R}, i = 1, \ldots, \nu$ be a set of functionals and $\sigma_i \in \mathbb{R}, i = 1, \ldots, \nu$ such that the differential problem

$$\begin{cases} Dy(x) = f(x) \\ c_i(y) = \sigma_i, \ i = 1, \ldots, \nu \end{cases} \tag{1}$$

is well-posed.

Moreover, let us consider that $\mathbb{P}$, the space of algebraic polynomials, is dense in $\mathbb{E}$ and let $\mathcal{P} = [P_0, P_1, \ldots]^T$ be the orthogonal polynomial basis of $\mathbb{P}$ associated to the inner product $\langle P_i, P_j \rangle = \|P_i\|^2 \delta_{i,j}$, where $\delta_{i,j}$ is the Kronecker delta function.

For a given natural $n > \nu$, the Tau method looks for an n degree polynomial $y_n = \mathcal{P}\mathsf{a}_n$, $\mathsf{a}_n = [a_0, a_1, \ldots, a_n, 0, \ldots]^T$ such that

$$\begin{cases} c_i(y_n) = \sigma_i, \ i = 1, \ldots, \nu \\ \langle Dy_n - f, P_i \rangle = 0, \ i = 0, \ldots, n - \nu \end{cases} \tag{2}$$

If D is, or can be approximated by, an operator of the form

$$D = \sum_{i=0}^{\nu} p_i \frac{d^i}{dx^i} \tag{3}$$

where p_i are polynomials, or polynomial approximations, then (2) implies that $Dy_n = f + \tau$, where τ is an order $n + 1 - \nu$ polynomial residual. This consideration gave rise to the method's name.

An operational formulation, introduced by Ortiz and Samara [29], translates (2) into an algebraic problem of finding the coefficients of y_n. Let M and N be the operational matrices, depending on $\mathcal{P}$, such that

$$x\mathcal{P} = \mathcal{P}\mathsf{M} \text{ and } \frac{d}{dx}\mathcal{P} = \mathcal{P}\mathsf{N}, \tag{4}$$

then the differential operator is cast as

$$Dy_n = \mathcal{P}\mathsf{D}\mathsf{a}_n, \text{ where } \mathsf{D} = \sum_{i=0}^{\nu} p_i(\mathsf{M})\mathsf{N}^i. \tag{5}$$

Let $\mathsf{D}_{(n-\nu+1)\times(n+1)}$ be the leading $(n-\nu+1)\times(n+1)$ block of D and $\mathsf{C}_{\nu\times(n+1)}$ the $\nu\times(n+1)$ matrix such that $\mathsf{C}_{i,j+1} = c_i(P_j), i = 1, \ldots, \nu, \ j = 0, \ldots, n$. Assuming that all functionals c_i are

linear, then the differential problem (2) can be written as

$$\left[\begin{array}{c}\mathsf{C}_{\nu\times(n+1)}\\ \hline \mathsf{D}_{(n-\nu+1)\times(n+1)}\end{array}\right]\mathsf{a}_n=\left[\begin{array}{c}\sigma\\ \hline \mathsf{f}\end{array}\right] \tag{6}$$

for $\sigma=[\sigma_1,\ldots,\sigma_\nu]^T$ and $\mathsf{f}=[f_0,\ldots,f_{n-\nu}]^T$, $f_i=\langle f,P_i\rangle/\|P_i\|^2$. The coefficients of the Tau residual can be obtained by $\tau=\mathsf{D}_{(n-\nu+2:n+1)\times(n+1)}\mathsf{a}_n$, where $\mathsf{D}_{(n-\nu+2:n+1)\times(n+1)}$, the remaining block of matrix D, contains the rows corresponding to the coefficients that were not cancelled [38].

Since the initial/boundary conditions must be always satisfied they are imposed in the first rows in `Tau Toolbox` (contrary to the literature where they are placed in the last rows). As usual in the operational implementation of the Tau method, particularly in `Tau Toolbox`, the same set of orthogonal polynomials for both the trial functions and the test functions have been used. In fact, this option is only justified by a matter of simplicity, since the heart of the method is in the choice of the set of orthogonal polynomials for test functions. Some authors have pointed out numerical advantages in other choices, notably in distinct choices for trial and test functions basis [26], or in the use of bases recombination other than orthogonal polynomials [15]. Nevertheless, building the operational matrices M and N on the bases that one intends to consider, the original Tau approach can be easily extended to those Tau method reformulations. In the `Tau Toolbox` project, we use explicit formulas to calculate the operational matrices, which has allowed us to obtain high precision results.

2.2 Numerical implementation

When working with the Tau method, the accuracy achieved evaluating matrix D in (5) is often neglected. The pioneering literature in this topic [28], proposed the evaluation of the operational matrices M and N in the polynomial canonical basis followed by a similarity transformation to get those matrices in the selected orthogonal basis. This procedure has the major disadvantage of introducing instability for increasing values of n (due to the poor conditioning of the matrices involved).

Profiting from the three-term recurrence relation characteristic of orthogonal polynomials

$$xP_j=\alpha_jP_{j+1}+\beta_jP_j+\gamma_jP_{j-1},\ j\geq 0,\ P_0=1,\ P_{-1}=0, \tag{7}$$

similarity transformations are avoided [23]. From (4) we have $\mathsf{M}=[\mu_{i,j}]_{i,j\geq 0}$ and $\mathsf{N}=[\eta_{i,j}]_{i,j\geq 0}$, with $\mu_{i,j}=\langle xP_j,P_i\rangle/\|P_i\|^2$ and $\eta_{i,j}=\left\langle P_j',P_i\right\rangle/\|P_i\|^2$, and so

$$xP_j=\sum_{i=0}^{j+1}\mu_{i,j}P_i,\ \text{and}\ P_j'=\sum_{i=0}^{j-1}\eta_{i,j}P_i,\ j=0,1,\ldots$$

Using (7) the elements of matrix M can be explicitly obtained from

$$\begin{cases}\mu_{0,0}=\beta_0,\ \mu_{1,0}=\alpha_0,\\ \mu_{j-1,j}=\gamma_j,\ \mu_{j,j}=\beta_j,\ \mu_{j+1,j}=\alpha_j,\quad j=1,2,\ldots\\ \mu_{i,j}=0,\quad |i-j|>1\end{cases} \tag{8}$$

and those of matrix N by recursive formulas

$$\begin{cases}\eta_{i,j+1}=\dfrac{\alpha_{i-1}\eta_{i-1,j}+(\beta_i-\beta_j)\eta_{i,j}+\gamma_{i+1}\eta_{i+1,j}-\gamma_j\eta_{i,j-1}}{\alpha_j}\\ \eta_{j,j+1}=\dfrac{1+\alpha_{j-1}\eta_{j-1,j}}{\alpha_j}\\ \eta_{i,j}=0,\ i\geq j,\quad \eta_{0,1}=\dfrac{1}{\alpha_0}\end{cases}. \tag{9}$$

Furthermore, iterating (9), explicit formulas for the first diagonals of N can be developed

$$\begin{aligned}
\eta_{j,j+1} &= \frac{j+1}{\alpha_j}, \\
\eta_{j,j+2} &= \frac{1}{\alpha_j \alpha_{j+1}} \sum_{i=0}^{j} (\beta_i - \beta_{j+1}), \\
\eta_{j,j+3} &= \frac{1}{\alpha_j \alpha_{j+1} \alpha_{j+2}} \sum_{i=0}^{j} \left[(\beta_i - \beta_{j+2})(\beta_i - \beta_{j+1}) + 2\alpha_i \gamma_{i+1} - \alpha_{j+1}\gamma_{j+2} \right],
\end{aligned}$$

for $j \geq 0$.

Explicit formulas for $\eta_{i,j}$, for the most commonly used orthogonal polynomials can be deduced [23] and are exploited in the implementation of `Tau Toolbox`.

Table 1 presents, for a set of classical orthogonal polynomials, the coefficients for the recurrence relation (7) and the explicit formulas for the coefficients $\eta_{i,j}$. Since matrix N is upper triangular with null diagonal, only possibly non-null $\eta_{i,j}$ elements, $i < j$, are presented.

Table 1: Values $\alpha_j,\ \beta_j,\ j = 0, 1, \ldots$ and $\gamma_j,\ j = 1, 2, \ldots$ in (8), and values $\eta_{i,j},\ i = 0, 1, \ldots, j-1,\ j = 1, 2, \ldots$ in (9). $\delta_{i,j}$ is the Kronecker symbol, $1_{i,j} = (-1)^{i+j}$ and $2_{i,j} = 1 - 1_{i,j}$.

P_j	orthogonal polynomials	α_j	β_j	γ_j	$\eta_{i,j}$ $i = 0, 1, \ldots, j-1$
$C_j^{(\lambda)}$	Gegenbauer	$\frac{j+1}{2(j+\lambda)}$	0	$\frac{j+2\lambda-1}{2(j+\lambda)}$	$2_{i,j}(i+\lambda)$
T_j	Chebyshev 1^{st} kind	$2^{\delta_{0,j}-1}$	0	2^{-1}	$2_{i,j}j/2^{\delta_{i,0}}$
U_j	Chebyshev 2^{nd} kind	2^{-1}	0	2^{-1}	$2_{i,j}(i+1)$
V_j	Chebyshev 3^{th} kind	2^{-1}	$1 - 2^{-\delta_{0,j}}$	2^{-1}	$2_{i,j}(i+\frac{1}{2}) + j - i$
W_j	Chebyshev 4^{th} kind	2^{-1}	$2^{-\delta_{0,j}} - 1$	2^{-1}	$(2_{i,j}(i+\frac{1}{2}) + j - i)1_{i,j+1}$
P_j	Legendre	$\frac{j+1}{2j+1}$	0	$\frac{j}{2j+1}$	$2_{i,j}(i+\frac{1}{2})$
L_j	Laguerre	$-j-1$	$2j+1$	$-j$	-1
H_j	Hermite	2^{-1}	0	$2j$	$2j\delta_{j-1-i,0}$
y_j	Bessel	$\frac{1}{2j+1}$	$-\delta_{0,j}$	$-\frac{1}{2j+1}$	$(i-j)(i+j+1)(i+\frac{1}{2})1_{i,j}$

Matrix D, specifying the differential operator, is built from M and N and the Tau method can be elegantly applied to differential eigenvalue problems as we will discuss next.

2.3 Differential eigenvalue problems

Let us assume that the differential operator D in (1) depends polynomially on a parameter λ, so that we can write

$$D(\lambda) = \sum_{k=0}^{m} \lambda^k D_k,$$

where each D_k is a differential operator of the form (3)

$$D_k = \sum_{i=0}^{\nu_k} p_{k,i} \frac{d^i}{dx^i}.$$

Then, with $\nu = \max_{k=0,\ldots,m} \nu_k$, the second set of equations in (2) becomes

$$\sum_{k=0}^{m} \lambda^k \left\langle D_k y_n - f, P_i \right\rangle = 0,\ i = 0, \ldots, n - \nu.$$

Like in (5), this can be translated into linear algebraic operations on coefficient vectors represented by matrices

$$\mathsf{D}(\lambda) = \sum_{k=0}^{m} \lambda^k \mathsf{D}_k, \text{ where } \mathsf{D}_k = \sum_{i=0}^{\nu_k} p_{k,i}(\mathsf{M})\mathsf{N}^i.$$

This is illustrated, in this paper, with Examples 1-4 in §4.

In a similar way, when in (1) at least one of the linear functionals c_i depends on λ, we have to consider a block $\mathsf{C}(\lambda)$ representing initial or boundary conditions in (6). This is illustrated with Example 5 in §4.

2.4 Solving the algebraic eigenproblem

As will be explained in more detail in §3, we propose a systematic methodology that consists in forming a matrix-valued function T that depends on the eigenvalue parameter λ, and then solve for all (or part) of the eigenvalues and eigenvectors of $T(\lambda)$. This computation should not be approached via a root-finder for the scalar equation $\det T(\lambda) = 0$, unless the problem size is tiny. Instead, algebraic eigensolvers must be employed that rely on numerically stable linear algebra algorithms. We next discuss briefly the methods available for both linear and nonlinear eigenproblems.

The first thing to take into account is that $T(\lambda)$ is, by construction, a non-symmetric matrix. Hence, a general algorithm is required and it is not possible to have recourse to specialized methods that exploit symmetry.

We say the problem is linear if T depends only linearly on the parameter λ, that is, T can be written as $A - \lambda B$ for two constant matrices A and B. Then the problem is a generalized matrix eigenvalue problem for the matrix pencil (A, B). The case where B is the identity matrix is referred to as the standard eigenproblem, but we will see that in the Tau method the simplest cases do not result in $B = I$ but in B being an identity padded with zeros on the top part. It is relevant to consider the case where B has zero rows or, more generally, is singular. In that case, the algebraic eigenproblem has infinite eigenvalues. We can let the eigensolver deal with such infinite eigenvalues, or handle them separately.

If all eigenvalues (and eigenvectors) must be computed the method of choice is the QZ algorithm [25]. This algorithm is the one executed if `MATLAB`'s command `eig` is issued on a matrix pencil. On the other hand, if only a subset of eigenvalues is needed, then one of the many available sparse matrix algorithms [1] must be employed. One possibility is to apply a Krylov method on the transformed problem

$$(A - \sigma B)^{-1} B x = \theta x,$$

whose eigenvectors x are the same as the original problem $Ax = \lambda Bx$. With this shift-and-invert approach, the Krylov method will approximate the eigenvalues $\lambda = \sigma + \theta^{-1}$ that are closest to the shift σ. Examples of restarted Krylov methods are Krylov-Schur and implicitly restarted Arnoldi. This type of computation is implemented in `MATLAB`'s command `eigs`.

In the nonlinear eigenvalue problem, T is either a matrix polynomial or a general nonlinear matrix-valued function. In the polynomial case, the most commonly used solution strategy is linearization, in which a linear eigenvalue problem of size $d \times n$, where d is the degree of the polynomial and n is the dimension of T, is built such that the eigenpairs of T can be directly extracted from the eigenpairs of this linear eigenproblem. Still, additional numerical ingredients such as scaling are usually necessary to get a trustworthy computed solution. The `MATLAB` code QUADEIG [19] implements such a robust numerical scheme for quadratic polynomials ($d = 2$). For higher degree polynomials, `MATLAB`'s command `polyeig` provides similar functionality. Note

that it is also possible to apply a Krylov method to the linearization in order to compute only a subset of eigenvalues.

For the general nonlinear eigenvalue problem, many different methods have been proposed that compute a few eigenvalues located inside a given region of the complex plane or closest to a given target value. In this paper, we will use the NLEIGS method [18], which essentially consists in two phases: in a first stage, $T(\lambda)$ is replaced by a carefully chosen rational approximation $Q_N(\lambda)$ that is built in such a way that it interpolates $T(\lambda)$ at N points in the region of interest. Then, the second step is to solve the resulting rational eigenvalue problem via linearization, in a similar way as discussed for matrix polynomials.

3 Proposed methodology

Let Ω be the domain containing the eigenvalues and consider the matrix-valued functions of the Tau method, $C : \Omega \to \mathbb{R}^{\nu\times n}$, representing the (boundary or initial) conditions, and $D : \Omega \to \mathbb{R}^{n\times n}$, obtained from the general equation including the differential operator. The general solution scheme proceeds by stacking these, obtaining a rectangular matrix of size $(\nu + n) \times n$ that depends on the parameter λ. For practical purposes, we repartition this matrix

$$\begin{matrix}\nu\\ n\end{matrix}\left[\begin{array}{c}C(\lambda)\\ \hline D(\lambda)\end{array}\right] = \left[\begin{array}{c}T(\lambda)\\ \hline R(\lambda)\end{array}\right]\begin{matrix}n\\ \nu\end{matrix} \tag{10}$$

by writing $D(\lambda) = \begin{bmatrix}E(\lambda)\\ R(\lambda)\end{bmatrix}$, where $R(\lambda)$ involves the last ν rows, and then defining

$$T(\lambda) := \begin{bmatrix}C(\lambda)\\ E(\lambda)\end{bmatrix}. \tag{11}$$

The $n \times n$ eigenvalue problem to be solved is

$$T(\lambda)x = 0, \quad x \neq 0, \tag{12}$$

that can be linear or nonlinear depending on how C and D are defined. We will use the following notation for the different cases:

$$\text{Linear Eigenvalue Problem (LEP)}: \qquad T(\lambda) = T_0 + T_1\lambda, \tag{13}$$

$$\text{Quadratic Eigenvalue Problem (QEP)}: \qquad T(\lambda) = T_0 + T_1\lambda + T_2\lambda^2, \tag{14}$$

$$\text{Polynomial Eigenvalue Problem (PEP)}: \qquad T(\lambda) = \sum_{i=0}^{d} T_i\lambda^i, \tag{15}$$

$$\text{Nonlinear Eigenvalue Problem (NEP)}: \qquad T(\lambda) = \sum_{i=0}^{d} T_i f_i(\lambda). \tag{16}$$

In the above expressions, λ is the eigenvalue, x is the eigenvector, d is the degree of the matrix polynomial or the number of terms in the NEP case, and f_i are scalar functions analytic in Ω.

In the solution of the linear eigenvalue problem (13) it is relevant to specifically consider the common case when the boundary conditions do not depend on the eigenvalue. In this case, $C(\lambda)$ is constant and hence the upper block of T_1 is zero. This implies that T_1 is singular, with a null space of dimension ν, or, equivalently, problem (13) has ν infinite eigenvalues. This is consistent with the fact that the characteristic equation $\det T(\lambda) = 0$ has $n-\nu$ roots only. In this scenario,

MATLAB's command `eig` on the matrix pencil $(T_0, -T_1)$ may give slightly inaccurate results, since it is using the QZ method. To avoid this, we propose two alternative solution schemes.

The first alternative is to solve the equivalent standard eigenvalue problem

$$-T_0^{-1}T_1 x = \theta x, \tag{17}$$

and then recover the original eigenvalues as $\lambda = \theta^{-1}$. This can be implemented with MATLAB's command `eig(-T0\T1)`, which involves factorizing matrix T_0. From the computed eigenvalues λ, we must remove the ν largest ones, corresponding to the infinite eigenvalues in exact arithmetic (or, equivalently, discard ν nearly zero values of θ). If not all eigenvalues are required, then `eigs(T0,-T1,k,0)` can be used instead to compute `k` eigenvalues closest to 0.

The second alternative is to exploit the structure of the matrices, since the upper block of T_1 is zero,

$$\begin{bmatrix} C_{01} & C_{02} \\ E_{01} & E_{02} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \lambda \begin{bmatrix} 0 & 0 \\ E_{11} & E_{12} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0. \tag{18}$$

The location of zero blocks enables the solution of the problem via a Schur complement approach,

$$(E_{02} - E_{01}C_{01}^{-1}C_{02})x_2 + \lambda(E_{12} - E_{11}C_{01}^{-1}C_{02})x_2 = 0, \tag{19}$$

for non singular $\nu \times \nu$ block C_{01}. After computing x_2 from (19), the upper part of the eigenvector can be obtained as $x_1 = -C_{01}^{-1}C_{02}x_2$. The advantage of this approach is that only $n - \nu$ eigenvalues are computed, so no eigenvalues must be discarded. Another benefit is that this scheme can also be extended to the PEP (15) and NEP (16) cases whenever $C(\lambda)$ does not depend on λ.

The Schur complement approach (19) avoids numerical difficulties arising from the fact that T_1 is singular. Still, the second matrix of the eigenproblem (18) could be singular if the E_{12} block is singular, which can happen in some singular Sturm-Liouville equations. In this case, the problem has $\lambda = \infty$ eigenvalues, which in the case of some iterative eigensolvers might result in contamination of the computed eigenvectors with components lying in the nullspace of the B-matrix. To avoid these problems, one should solve the eigenproblem with a more elaborate scheme that includes purification or related techniques [35, 13].

4 Numerical examples

All tests were performed on a laptop with Processor AMD Ryzen 7 4800H, 2900 Mhz, 8 Core(s), 16 Logical Processor(s) and MATLAB Version 9.8, Release R2020a.

Example 1 We address the 4th order Sturm-Liouville equation

$$\frac{d^4y}{dx^4}(x) = \lambda y(x)$$

for $x \in [0, 1]$ with boundary conditions

$$y(0) = 0, \quad y(1) = 0, \quad \frac{dy}{dx}(0) = 0, \quad \frac{d^2y}{dx^2}(1) = 0,$$

whose exact solution satisfies equation $\tanh(\lambda^{1/4}) = \tan(\lambda^{1/4})$ [22].

The code for solving this problem is depicted in Listing 2, where the outputs are the eigenvalues vector, `lambda`, the set of associated eigenfunctions, `V` (a Tau object), and the Tau associated residual, `R` (Tau object).

Table 2: Six smallest eigenvalues computed via Chebyshev-Tau with degree 24 and 48, norm of the residual and error related with verification equation.

k	λ_k	$\|\frac{T(\lambda_k)\cdot x}{\|\lambda_k\|}\|$	$\|\tanh(\lambda_k^{1/4}) - \tan(\lambda_k^{1/4})\|$
$n = 24$			
1	2.377210675311166e+02	7.4e-15	4.4e-16
2	2.496487437856829e+03	3.5e-15	3.3e-15
3	1.086758221697888e+04	5.9e-15	5.3e-15
4	3.178009645407464e+04	1.2e-14	1.4e-12
5	7.400084934957961e+04	4.2e-14	4.7e-11
6	1.486344773662659e+05	3.5e-14	5.3e-09
$n = 48$			
1	2.377210675311169e+02	7.5e-15	2.2e-15
2	2.496487437856829e+03	6.1e-15	3.3e-15
3	1.086758221697889e+04	4.5e-15	1.8e-15
4	3.178009645408105e+04	1.6e-14	5.6e-15
5	7.400084934915529e+04	1.2e-14	2.6e-14
6	1.486344772857699e+05	3.4e-14	2.3e-14

```
% set the problem
equation = "diff(y,4)=lambda*y";
domain = [0, 1];
conditions = {"y(0)=0"; "y(1)=0"; "y'(0)=0"; "y''(1)=0"};
options = tau.settings('degree', 24, 'basis', 'ChebyshevT');
problem = tau.problem(equation, domain, conditions, options);

% solve the problem
[lambda, V, R] = tau.eig.LEP(problem);
```

Listing 2: MATLAB code to compute the solution of Example 1 using Tau Toolbox.

Results for the Chebyshev-Tau polynomial approximation solution with degree 24 and 48 are given in Table 2, and show good accuracy even for the smallest degree.

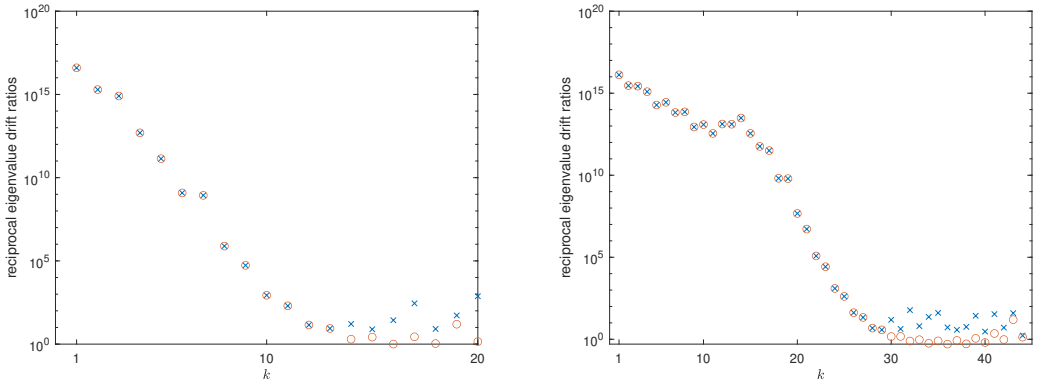
The computation of the LEP can be done by the MATLAB `eig` and `eigs` functions, respectively, for all the spectrum (dense data) and a subset of the spectrum (sparse data). For the Tau methodology, matrices are blocked to order n. Internally, `tau.eig.LEP`, builds the Tau matrix T and solves the algebraic eigenvalue problem in the form (18) using (19).

To get any matrix, a call to `tau.eig.matrix(op, problem)`, where `op` stands for matrix operator T, C, D (or for building block matrices M and N) is enough. In turn, and going in-depth within the tools available, matrix T can be explicitly built (see (11) and (13)) by executing the Listing 3.

```
n = options.n;
nu = problem.nconditions;

% create matrices C, D and build T
C = tau.eig.matrix('C', problem);
D = tau.eig.matrix('D', problem);
T{1} = [C; D{1}];
T{2} = [zeros(nu, size(D{2}, 2)); D{2}];
```

Listing 3: MATLAB excerpt to build tau matrix T using Tau Toolbox.



(a) $n_1 = 24,\ n_2 = 36$ (b) $n_1 = 48,\ n_2 = 60$

Figure 1: The reciprocal eigenvalue drift ratios $1/\delta_{k,\text{nearest}}$ (x-marks) and $1/\delta_{k,\text{ordinal}}$ (circles) on a logarithmic scale against eigenvalue order k.

The drift concept, heuristically introduced in [3], to distinguish "good" from "bad" computed eigenvalues is now applied to graphically visualize the differences between two degree approximations n_1 and n_2, scaled by some measure of the size of the eigenvalues. Figure 1 shows semilogarithmic plots with the reciprocal of the ordinal difference $\delta_{k,\text{ordinal}} \equiv |\lambda_k^{n_1} - \lambda_k^{n_2}|/\sigma_k$ and with the reciprocal of the nearest difference (to acommodate clustered eigenvalues)

$$\delta_{k,\text{nearest}} \equiv \min_{k\in[1,n_2]} |\lambda_k^{n_1} - \lambda_k^{n_2}|/\sigma_k$$

where $\sigma_1 = |\lambda_1 - \lambda_2|$ and $\sigma_k = \frac{1}{2}(|\lambda_k - \lambda_{k-1}| + |\lambda_{k+1} - \lambda_k|)$, $j > 1$. The well computed eigenvalues are those with larger reciprocal drift ratios, so problems only occur for the largest eigenvalues in magnitude and where the two criteria notably deliver different results (the nearest ratios are consistently a little larger than the ordinal).

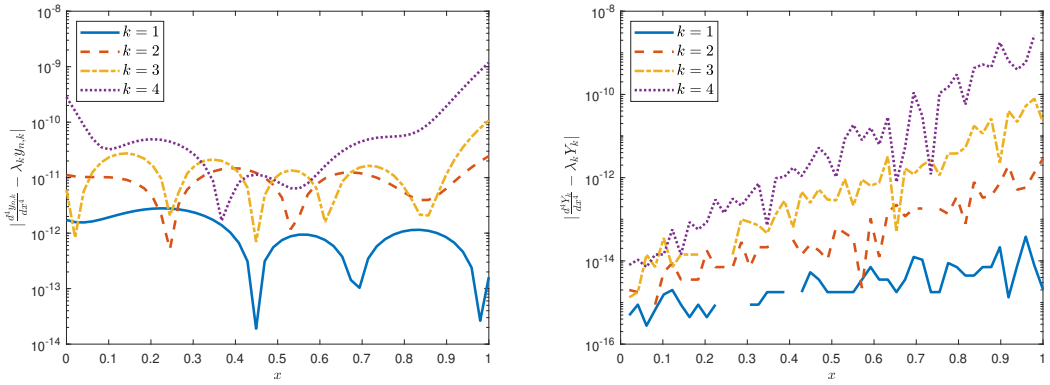
This problem, associated to the free vibration of a thin beam clamped at $x = 0$ and supported at $x = 1$, has exact eigenfunction associated to the eigenvalue λ_k given by [22]

$$Y_k = \frac{\cosh\left(\lambda_k^{1/4}x\right) - \cos\left(\lambda_k^{1/4}x\right) - \frac{\cosh\left(\lambda_k^{1/4}\right)+\cos\left(\lambda_k^{1/4}\right)}{\sinh\left(\lambda_k^{1/4}\right)+\sin\left(\lambda_k^{1/4}\right)}\left(\sinh\left(\lambda_k^{1/4}x\right) - \sin\left(\lambda_k^{1/4}x\right)\right)}{2\lambda_k^{1/2}}.$$

Figure 2 plots the residual vectors for the first four eigenvalues using the computed eigenvectors $|\frac{d^4 y_{n,k}}{dx^4} - \lambda_k y_{n,k}|$, $y_{n,k} = \sum_{i=0}^{n} \mathtt{a}_{i,k} P_i$, where P_i stands for the shifted Chebyshev polynomials of the first kind and $\mathtt{a}_{i,k}$ the coefficients of the object component `V(k)` (Figure 2a) and the exact eigenfunctions Y_k, $|\frac{d^4 Y_k}{dx^4} - \lambda_k Y_k|$, $k = 1, \ldots, 4$, (Figure 2b). Both residuals suggest a good approximation for the eigenpairs.

The Tau residual associated with each computed eigenpair can be plotted by the `Tau Toolbox` function `plot` (see Figure 3). The class `tau.polynomial` redefines the `plot MATLAB` function, enabling it to handle different types of input arguments. If the dominant argument is a Tau polynomial, `MATLAB` calls the method defined in the object's class.

The accuracy in computing the eigenfunctions can be emphasized by estimating the loss of orthogonality, using a large ($n = 240$) polynomial degree approximation. Figure 4 shows a log-linear plot with the absolute values of the scalar products of `V(1)` and `V(j)`, `j` $= 2, \ldots, 120$. For the first modes the departure is close to machine precision and, overall, all within a good accuracy.



(a) $|\frac{d^4 y_{n,k}}{dx^4} - \lambda_k y_{n,k}|$ (b) $|\frac{d^4 Y_k}{dx^4} - \lambda_k Y_k|$

Figure 2: Absolute value of the residual vectors: (a) eigenmodes $y_{n,k}$ computed with `Tau Toolbox` and (b) using the analytical eigenfunctions Y_k, $k = 1, \ldots, 4$.

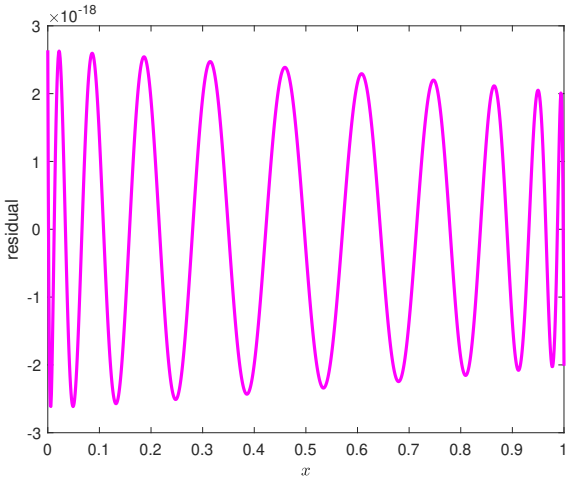


Figure 3: Tau residual for the eigenfunction corresponding to the smallest eigenvalue in magnitude, for $n = 24$.

Example 2 Let us consider the solution of the Coffey-Evans equation

$$-\frac{d^2 y}{dx^2}(x) + ((\beta \sin 2x)^2 - 2\beta \cos 2x) y(x) = \lambda y(x), \quad -\pi/2 < x < \pi/2$$

with boundary conditions $y(-\pi/2) = y(\pi/2) = 0$.

This is a second order Sturm-Liouville problem with non-polynomial coefficients, a particular case of a (regular) Schrödinger problem with a periodic potential.

For separated boundary conditions it is guaranteed that there are no multiple eigenvalues [39], but triplets of increasingly tight together eigenvalues are produced for increasing values of β, which controls the depth of the well potential. This is a challenging and interesting problem to solve [31]. For $\beta = 30$ the difference between the second and fourth eigenvalues, $\lambda_2 < \lambda_4$, is under 10^{-8} (see [16]).

The Tau method expects polynomial coefficients and thus `Tau Toolbox` provides all necessary approximations. Although the process can be fully automatically done via a call to

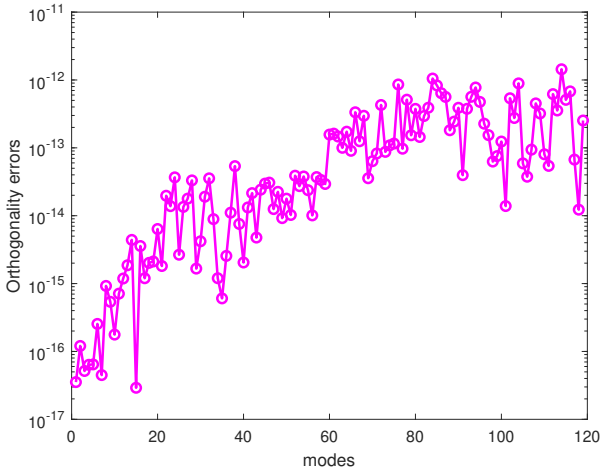


Figure 4: Orthogonality errors (deficiency) of the first eigenvector with respect to the subsequent.

`tau.solve` (`tau.eig.solve` or `tau.eig.LEP`), in the Listing 4 we detail one possible procedure to approximate the non-polynomial terms. For the Tau polynomial object `x`, the interpolatory polynomial approximation `pn` of `beta^2*sin(2*x).^2-2*beta*cos(2*x)` is computed. For the Tau operator object `y`, `diff(y, 2)` is the second derivative, which is similar to apply twice the derivative matrix operator `N` (see (5)).

```
n = problem.options.n; nu = problem.nconditions;

% build building blocks
C = tau.matrix('cond', problem);

beta = 30;
x = tau.polynomial(options);
pn = @(x) beta^2*sin(2.*x).*sin(2.*x)-2*beta*cos(2.*x);
y = tau.operator(options); lhs = -diff(y, 2) + pn(x)*y;
D = lhs.mat;

% build terms
T{1} = [C; D(1:n-nu, 1:n)];            % independent term
T{2} = [zeros(nu, n); -eye(n-nu, n)];  % term on lambda

% use QZ to solve
[V, D] = eig(T{1}, -T{2}); lambda = diag(D);
[~,p] = sort(abs(lambda));
lambda = lambda(p); V = V(:, p);
```

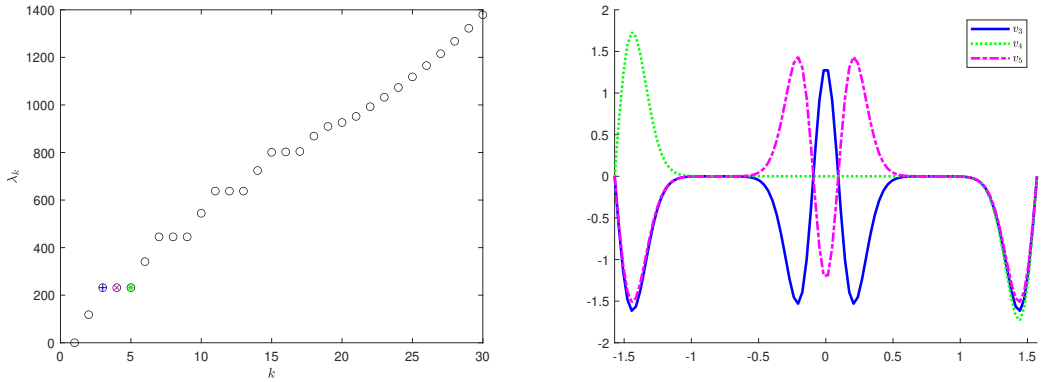
Listing 4: Excerpt of `MATLAB` code to explicitly build the differential problem in Example 2 with `Tau Toolbox`.

In the code, we just used QZ algorithm to tackle the problem. Yet, `Tau Toolbox` provides several techniques. Table 3 shows the first 10 eigenvalues, for a Chebyshev-Tau approach with a polynomial of degree 106, using (19). It is worth mentioning that techniques such as `split` (perfect shuffle permutation) and `deflation` are available.

As stated, this problem is challenging since the numerical method must distinguish eigenvalues within the triple clusters that arise for increasing values of β. Figure 5a depicts the first 30 eigenvalues and Fig. 5b the three eigenmodes associated with the first triplet (`plot(V(3:5))`.

Table 3: Ten smallest eigenvalues computed via Chebyshev-Tau with degree 106.

k	λ_k	k	λ_k
$n = 106$			
1	0.000000000000068e+02	6	3.408882998096128e+02
2	1.179463076620689e+02	7	**4.45283**0895823253e+02
3	**2.31664929**2370212e+02	8	**4.45283**1723066395e+02
4	**2.31664929**3129615e+02	9	**4.45283**2550313369e+02
5	**2.31664929**3887466e+02	10	5.444183851493636e+02



(a) Smallest eigenvalues

(b) Eigenfunctions corresponding to the cluster of smallest eigenvalues

Figure 5: Coffey-Evans problem for $\beta = 30$ using Chebyshev-Tau.

Numerical results presented in [14] reports that the first of those triplets have 2.31664929 as its common numeric part and for the eigenvalues in the second triplet it is found the common numeric part 4.45283, the same exact decimal places we present in Table 3, obtained with `Tau Toolbox`.

The scaled differences (or "drift-with-n") are also used to assess the quality of the computed eigenvalues for the Coffey Evans problem (see Figure 6).

Table 4: High index eigenvalue computation by different methods, for Coffey-Evans eigenproblem (Example 2).

	λ_{200}
Chebfun [11]	**40851.6376**459609
Chebyshev collocation [14]	**40851.6376**460506
`Tau Toolbox`	**40851.6376**460506

We can also report that `Tau Toolbox` is appropriate to accurately evaluate high order eigenvalues. In Table 4, distinct approximations for λ_{200} are provided, taking $n = 400$. The approximation provided by `Tau Toolbox` is in line with the results with Chebfun [11] and using Chebyshev collocation [14]. Computation time is negligible: 0.12 seconds are reported to interpret, build and solve the differential eigenvalue problem with the `Tau Toolbox` using implicitly

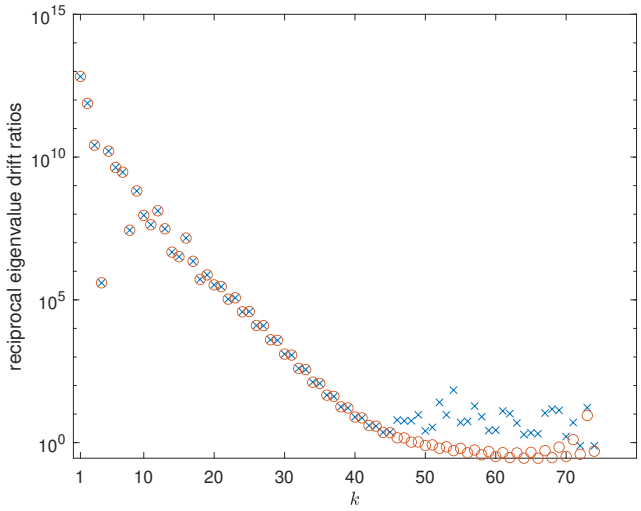


Figure 6: The reciprocal eigenvalue drift ratios $1/\delta_{k,\text{nearest}}$ (x-marks) and $1/\delta_{k,\text{ordinal}}$ (circles) on a logarithmic scale against eigenvalue order k ($n_1 = 76, n_2 = 86$).

restarted Arnoldi method and 0.22 seconds to obtain all eigenvalues (QZ).

Example 3 We present now a quadratic eigenvalue problem $-y''(x) = \lambda y(x) + \lambda^2 x^2 y(x)$, for $x \in [-1, 1]$ with boundary conditions $y(-1) = y(1) = 0$ (see [29]).

Table 5: Ten smallest eigenvalues in magnitude computed via Chebyshev-Tau with degree 10, using `roots` (eigenvalues of the companion matrix) and relative errors with QEP and PEP, for Example 3.

k	λ_k	$\frac{\|\lambda_k^{\text{QEP}} - \lambda_k\|}{\|\lambda_k\|}$	$\frac{\|\lambda_k^{\text{PEP}} - \lambda_k\|}{\|\lambda_k\|}$
1	1.951702296762780	1.14e-15	1.44e-14
2	4.286091310533131	1.45e-15	8.29e-16
3	-6.597087872105731	1.43e-14	9.15e-15
4	-7.036504991248593	2.01e-14	2.57e-14
5	7.544667375997218	1.26e-14	1.53e-14
6	10.199214296244536	2.53e-14	5.42e-14
7	-13.281221649396613	1.93e-13	1.96e-13
8	13.630851375970161	1.93e-13	1.94e-13
9	-13.757341018400584	4.38e-14	3.73e-14
10	16.367795989738894	2.26e-14	7.55e-14

Proceeding in the usual way as in the previous examples, the user only has to specify the `problem`. The `Tau Toolbox` detects the type of problem and solves it, given a basis and a polynomial degree approximation. For this example, `Tau Toolbox` can either take `quadeig` [19] or `MATLAB`'s `polyeig` as solvers, which are implemented, respectively, in `tau.eig.QEP` and `tau.eig.PEP`. Table 5 shows a set of the eigenvalues for this problem using a polynomial approximation of degree 10. Using symbolic computations, we provide the Tau approximation

for the roots of the characteristic polynomial and the relative errors for the approximation obtained with QEP and PEP. The accuracy of the approximations are very good, particularly for QEP.

Example 4 The Orr-Sommerfeld type equation governs the stability of shear and related flows and was first solved by **(author?)** [27] using the spectral Tau method.

The Orr–Sommerfeld equation

$$\frac{d^4y}{dx^4}(x) - 2\alpha^2\frac{d^2y}{dx^2}(x) + \alpha^4 y(x) = i\alpha Re\left[\Big(U(x)-\omega\Big)\Big(\frac{d^2y}{dx^2}(x) - \alpha^2 y(x)\Big) - \frac{d^2U}{dx^2}(x)y(x)\right]$$

results from the linearization of the incompressible Navier–Stokes equations, where y is the potential (or stream function), α the wavenumber, Re the Reynolds number and ω the radian frequency (growth rate).

The stability problem of interest is that of a plane Poiseuille flow in a channel with stream velocity in the x direction given by $U(x) = 1 - x^2$ and side walls located at $y = \pm 1$. The boundary conditions are set as $y(\pm 1) = \frac{dy}{dx}(\pm 1) = 0$. Another usual problem is that of Couette flow which is driven by the upper boundary being sheared relative to the lower one, $U = x$.

The computation of the eigenvalues for this problem has already been tackled, among others, by [4, 33, 9, 37].

Spatial stability calculations, where disturbances are periodic in time (real ω), lead to a quartic eigenvalue problem (eigenpairs (α, y)) and temporal stability calculations, where α is fixed and real, lead to a fourth-order generalized eigenvalue problem (eigenpairs (ω, y)).

We consider the plane Poiseuille flow, first with $\alpha = 1$ and $Re = 10000$, and compute the eigenpairs (ω, y). Whenever the imaginary part of ω is positive the base flow is unstable [27]. The Chebyshev approximation to the most unstable mode for several values of the polynomial approximation is

n	ω
40	$0.237527301412992 + 0.0037418759134 93i$
60	$0.237526488828910 + 0.0037396706248 33i$
80	$0.237526488820470 + 0.0037396706229 79i$
100	$0.237526488820470 + 0.0037396706229 80i$

Now we consider $\alpha = 1.02056$ and a Reynolds number $Re = 5772.22$, close to the critical value for eigenvalue instability [27]. The spectrum of the Orr–Sommerfeld equation at these critical values is drawn in Fig. 7, for $\lambda = -i\alpha\omega$.

The rightmost eigenvalue is the most unstable one, and its real part is expected to be zero. The supremum among the real part of the elements in the spectrum — the spectral abscissa — is $\mathrm{Re}(\lambda_1) = 3 \cdot 10^{-9}$, using Chebyshev-Tau approximation with $n = 100$. It has been discussed that Chebyshev polynomials are better suited for the solution of hydrodynamic stability problems compared to other sets of orthogonal functions expansions [27], and require less computer time and storage to achieve good accurate flow simulations. The `Tau Toolbox` is implemented with stability concerns: all operations are performed directly on the specific orthogonal basis. For the case at hand, we registered similar magnitude of the real part of the eigenvalue closest to zero for several polynomial orthogonal bases (Chebyshev second, third and fourth kind, and Legendre and Gegenbauer).

The elapsed time to interpret, build and solve the differential eigenvalue problem for a polynomial approximation of degree 100 is just 0.02 seconds.

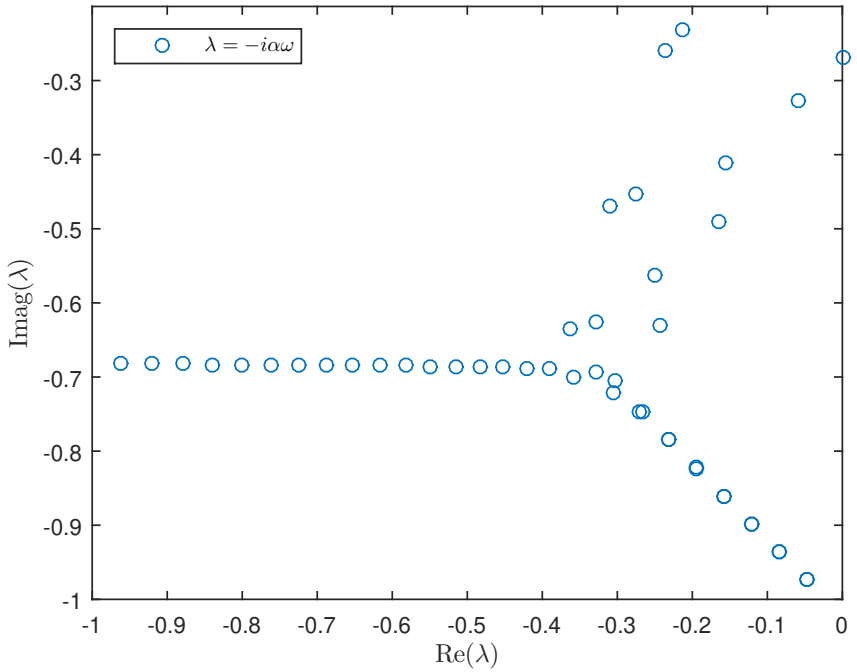


Figure 7: Eigenvalues ($\lambda = -i\alpha\omega$) for the Orr–Sommerfeld problem for $Re = 5772.22$ and $\alpha = 1.02056$ using the Chebyshev basis.

Example 5 The loaded string problem [36] is a boundary value problem related to the vibration of a string with a load of mass m attached by an elastic spring of stiffness k. We seek $\lambda \in \mathcal{I}$ and nontrivial functions $y(x)$, $x \in [0, 1]$, such that

$$-\frac{d^2y}{dx^2}(x) = \lambda y(x), \tag{20}$$

with boundary conditions $y(0) = 0$ and $-y'(1) = \varphi(\lambda)y(1)$, where $\mathcal{I} = (\sigma, +\infty)$, $\sigma = k/m$ and $\varphi(\lambda) = \sigma m\lambda/(\lambda - \sigma)$. We consider the particular case with $m = 1$, $k = 1$, and hence

$$\varphi(\lambda) = \frac{\lambda}{\lambda - 1}. \tag{21}$$

The fact that the boundary condition on the right end of the domain depends on the eigenvalue via a rational function turns it into a rational eigenvalue problem. This problem can be solved via linearization, but here we consider a more general approach that can be used for the case of general nonlinear eigenvalue problems.

The `Tau Toolbox` includes `tau.eig.NEP` as a solver, based on a `MATLAB` implementation[1] of NLEIGS – a class of fully rational Krylov methods for nonlinear eigenvalue problems [18]. This method is based on rational interpolation followed by linearization, and is appropriate for the case of nonlinear eigenproblems when the associated nonlinear function has singularities. The problem is cast in the form $F(\lambda) = T_0 + \lambda T_1 + \lambda/(\lambda - 1)T_2$ (16).

The exact eigenvalues are given by

$$\lambda = \psi(\sigma) = \frac{2a_2\cos(\sigma h) + a_1}{2b_2\cos(\sigma h) + b_1},$$

[1] http://twr.cs.kuleuven.be/research/software/nleps/nleigs.php

where $a_1 = 2/h$, $a_2 = -1/h$, $b_1 = 4h/6$, $b_2 = h/6$, and the σ values are the solutions of (see [36])

$$\frac{\tan(\sigma)}{\sin(\sigma h)} = \frac{a_2 - \psi(\sigma) b_2}{\varphi(\psi(\sigma))}.$$

Table 6 reports the six smallest eigenvalues, considering $h = 10^{-4}$ and a 15th polynomial degree approximation with Chebyshev-Tau, along with the relative error, $\frac{\|\lambda_k - \psi(\sigma)\|}{\|\lambda_k\|}$, and the residual, $\frac{\|F(\lambda_k)v_k\|_2}{\|F(\lambda_k)v_k\|_F\|v_k\|_2}$ [17].

Table 6: First smallest eigenvalues in magnitude computed via Chebyshev-Tau with degree 15, for Example 5.

k	λ_k	relative error	residual
1	0.45731832396312	2.87e-08	2.81e-17
2	4.48202429555980	1.60e-09	5.90e-17
3	24.2187013911745	2.12e-08	2.46e-17
4	63.6900266349135	5.39e-08	4.01e-17
5	122.905311160680	4.10e-08	2.02e-17
6	201.861537085238	1.91e-06	5.09e-17

5 Conclusions

Computing approximate solutions of integro-differential problems can be difficult and time consuming, requiring large expertise. The `Tau Toolbox` is a numerical library that produces approximate polynomial solutions of integro-differential equations via the Lanczos' spectral Tau method. This paper describes the use of the library to solve eigenproblems for linear, quadratic and nonlinear differential operators, exploring operational matrix formulations obtained directly on the chosen orthogonal basis.

The library allows the use of classical orthogonal bases, not being limited to the most current Chebyshev basis of the first kind. Results are obtained with high accuracy and the user interface is simple and intuitive. For the moment, it is up to the user to specify the degree of the polynomial approximation. In future versions, it is planned to adaptively consider the necessary number of terms in the series development to get a solution within a prescribed accuracy. Furthermore, the use of different bases for the trial and test functions is to be explored.

The software `Tau Toolbox` is available from Netlib Repository (`http://www.netlib.org/numeralgo/`) as the na55 package. The development repository is hosted at `https://bitbucket.org/tautoolbox/tautoolbox/`.

Acknowledgments This work was partially supported by the Spanish Agencia Estatal de Investigación under grant PID2019-107379RB-I00 / AEI / 10.13039/501100011033, and by CMUP, which is financed by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the project with reference UIDB/00144/2020.
The authors thank the anonymous referees for their careful reading and valuable suggestions.

Data availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Conflict of interest The authors declare that they have no conflict of interest.

References

[1] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: a practical guide*. SIAM, 2000.

[2] P. B. Bailey, W. N. Everitt, and A. Zettl. Algorithm 810: The sleign2 Sturm-Liouville code. *ACM Trans. Math. Softw.*, 27(2):143–192, jun 2001.

[3] JP Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications Inc, 2000.

[4] T. J. Bridges and P. J. Morris. Differential eigenvalue problems in which the parameter appears nonlinearly. *J. Comput. Phys.*, 55(3):437–460, 1984.

[5] Kathryn M. Butler and Brian F. Farrell. Three-dimensional optimal perturbations in viscous shear flow. *Phys. Fluids A-Fluid*, 4(8):1637–1650, 1992.

[6] Marios Charalambides and Fabian Waleffe. Gegenbauer Tau methods with and without spurious eigenvalues. *SIAM J. Numer. Anal.*, 47(1):48–68, 2009.

[7] T. Chaves and E. L. Ortiz. On the numerical solution of two-point boundary value problems for linear differential equations. *Z. Angew. Math. Mech.*, 48(6):415–418, 1968.

[8] Paul T. Dawkins, Steven R. Dunbar, and Rod W. Douglass. The origin and nature of spurious eigenvalues in the spectral Tau method. *J. Comput. Phys.*, 147(2):441–462, 1998.

[9] J.J. Dongarra, B. Straughan, and D.W. Walker. Chebyshev Tau-QZ algorithm methods for calculating spectra of hydrodynamic stability problems. *Appl. Numer. Math.*, 22(4):399–434, 1996.

[10] Tobin A Driscoll and Nicholas Hale. Rectangular spectral collocation. *IMA Journal of Numerical Analysis*, 36(1):108–132, 2016.

[11] Tobin A Driscoll, Nicholas Hale, and Lloyd N Trefethen. Chebfun Guide, 2014.

[12] David R. Gardner, Steven A. Trogdon, and Rod W. Douglass. A modified Tau spectral method that eliminates spurious eigenvalues. *J. Comput. Phys.*, 80(1):137–167, 1989.

[13] C. I. Gheorghiu and J. Rommes. Application of the Jacobi-Davidson method to accurate analysis of singular linear hydrodynamic stability problems. *International Journal for Numerical Methods in Fluids*, 71(3):358–369, 2012.

[14] Călin-Ioan Gheorghiu. Accurate spectral collocation computation of high order eigenvalues for singular Schrödinger equations. *Computation*, 9(2), 2021.

[15] Călin-Ioan Gheorghiu and I. S. Pop. A modified Chebyshev-Tau method for a hydrodynamic stability problem. In Dimitrie D. Stancu, editor, *Approximation and Optimization*, pages 119–126, Cluj-Napoca, 1996. Transilvania Press.

[16] Leon Greenberg and Marco Marletta. Algorithm 775: the code SLEUTH for solving fourth-order Sturm-Liouville problems. *ACM T. Math. Software*, 23(4):453–493, 1997.

[17] Stefan Güttel and Françoise Tisseur. The nonlinear eigenvalue problem. *Acta Numerica*, 26:1–94, 2017.

[18] Stefan Güttel, Roel van Beeumen, Karl Meerbergen, and Wim Michiels. NLEIGS: A class of fully rational Krylov methods for nonlinear eigenvalue problems. *SIAM J. Sci. Comput.*, 36(6):A2842–A2864, 2014.

[19] S. Hammarling, C. J. Munro, and F. Tisseur. An algorithm for the complete solution of quadratic eigenvalue problems. *ACM T. Math. Software*, 39(3):1–19, 2013.

[20] Cornelius Lanczos. Trigonometric interpolation of empirical and analytical functions. *J. Math. Phys.*, 17(1-4):123–199, 1938.

[21] Veerle Ledoux, M Van Daele, and G Vanden Berghe. Matslise: A matlab package for the numerical solution of Sturm-Liouville and Schrödinger equations. *ACM Transactions on Mathematical Software (TOMS)*, 31(4):532–554, 2005.

[22] Moinuddin Malik and Hao Huy Dang. Vibration analysis of continuous systems by differential transformation. *Applied Mathematics and Computation*, 96(1):17–26, 1998.

[23] J. M. A. Matos, M. J. Rodrigues, and J. C. Matos. Explicit formulae for integro-differential operational matrices. *Math. Comput. Sci.*, 15:45–61, 2021.

[24] G. B. McFadden, B. T. Murray, and R. F. Boisvert. Elimination of spurious eigenvalues in the Chebyshev Tau spectral method. *J. Comput. Phys.*, 91(1):228–239, 1990.

[25] C. B. Moler and G. W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10(2):241–256, 1973.

[26] Sheehan Olver and Alex Townsend. A fast and well-conditioned spectral method. *SIAM Review*, 55(3):462–489, 2013.

[27] S. A. Orszag. Accurate solution of the Orr–Sommerfeld stability equation. *J. Fluid Mech.*, 50(04):689, 1971.

[28] E. L. Ortiz and H. Samara. An operational approach to the Tau method for the numerical solution of non-linear differential equations. *Computing*, 27:15–26, 1981.

[29] E. L. Ortiz and H. Samara. Numerical solution of differential eigenvalue problems with an operational approach to the Tau method. *Computing*, 31(2):95–103, 1983.

[30] Steven Pruess and Charles T Fulton. Mathematical software for Sturm-Liouville problems. *ACM Transactions on Mathematical Software (TOMS)*, 19(3):360–376, 1993.

[31] J. D. Pryce. Error control of phase-function shooting methods for Sturm-Liouville problems. *IMA J. Numer. Anal.*, 6(1):103–123, 1986.

[32] John D Pryce and Marco Marletta. A new multi-purpose software package for Schrödinger and Sturm-Liouville computations. *Computer Physics Communications*, 62(1):42–52, 1991.

[33] S. C. Reddy, P. J. Schmid, and D. S. Henningson. Pseudospectra of the Orr–Sommerfeld operator. *SIAM J. Appl. Math.*, 53(1):15–47, 1993.

[34] Satish C. Reddy and Dan S. Henningson. Energy growth in viscous channel flows. *J. Fluid Mech.*, 252:209–238, 1993.

[35] J. Rommes. Arnoldi and Jacobi-Davidson methods for generalized eigenvalue problems $Ax = \lambda Bx$ with singular B. *Mathematics of Computation*, 77(262):995–1016, 2007.

[36] S. I. Solov'ëv. Preconditioned iterative methods for a class of nonlinear eigenvalue problems. *Linear Algebra Appl.*, 415(1):210–229, 2006.

[37] F. Tisseur and N. J. Higham. Structured pseudospectra for polynomial eigenvalue problems, with applications. *SIAM J. Matrix Anal. Appl.*, 23(1):187–208, 2001.

[38] M. Trindade, José Matos, and Paulo B. Vasconcelos. Towards a Lanczos' τ-method toolkit for differential problems. *Math. Computer Science*, 10(3):313–329, 2016.

[39] S. Yuan, K. Ye, C. Xiao, D. Kennedy, and F.W. Williams. Solution of regular second-and fourth-order Sturm-Liouville problems by exact dynamic stiffness method analogy. *J. Eng. Math.*, 86(1):157–173, 2014.

[40] Abdelfattah Zebib. Removal of spurious modes encountered in solving stability problems by spectral methods. *J. Comput. Phys.*, 70(2):521 – 525, 1987.