British Journal of Mathematics & Computer Science
3(4): 772-793, 2013

SCIENCEDOMAIN *international*
www.sciencedomain.org

SCIENCEDOMAIN

Operatorial Tau Method for Higher Order Differential Problems

Damian Trif[*]

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

Research Article

Received: 11 June 2013
Accepted: 09 August 2013
Published: 24 August 2013

Abstract

The paper extends the applicability of our freely accessible Matlab package *Chebpack* to calculate the eigenvalues and eigenfunctions of some higher order differential problems as well as to semidiscretize evolution problems, directly or by using the Lyapunov-Schmidt reduction method. The numerical examples illustrate the accuracy and the simplicity of the algorithms and prove the importance of this approach for practical applications.

Keywords: higher order problems; Chebyshev polynomials; tau method; Lyapunov-Schmidt method
2010 Mathematics Subject Classification: 35G16; 34B09; 65L106

1 Introduction

The mathematical models of many phenomena in the real world are expressed by higher order differential equations and they have attracted considerable attention of many authors. The problem of vibrating rods and plates, the problems in hydrodynamics and magnetohydrodynamics, problems in quantum mechanics or elasticity for example led to such higher-order boundary value problems.

The new *Matlab* package *Chebfun* [1] uses Chebyshev pseudospectral method. It is built on piecewise Chebyshev polynomial interpolants [2] and has a module for operatorial treating linear or nonlinear ordinary differential equations, initial value or boundary value problems. It has the necessary tools to be used for higher order differential problems. However, in *Chebfun* the discretizations are chosen automatically (not visible to the user) to achieve the maximum possible accuracy available from double precision arithmetic and this fact increases the computing time. A number of eigenvalues and eigenfunctions of a differential operator are approximated by applying `eigs` of Matlab to the matrix of the discretized problem. This approach does not offer a complete image of the spectral behavior of that approximating operator.

Our *Matlab* package *Chebpack* [3] [4] is based directly on the Chebyshev spectral *tau* method, also used in an operatorial form. It is aimed to approximate solutions of differential problems - initial value, boundary value or non-local problems, integro-differential equations, rootfinding, delay

*Corresponding author: E-mail: dtrif@math.ubbcluj.ro, dvtrif@gmail.com

differential equations [5] [6]. Here we extend the applicability of *Chebpack* to find Chebyshev polynomial approximations of the solutions for higher order differential problems. In our approach, the matrix of the discretized problem is directly obtained by following the structure of the given mathematical problem. The Chebyshev spectral *tau* method is more convenient for approximating linear operators. Their eigenvalues and eigenfunctions are numerically calculated by using `eig` of Matlab which offers the complete spectrum of the approximating operator. This fact is useful for the stability analysis of the algorithms used for evolution problems. Moreover, the spectral *tau* method also gives an evaluation of the residual, needed to estimate the numerical errors.

The paper is structured as follows. In section 2 we shortly describe *Chebpack* and the *tau method* in order to introduce the main discrete operator companions to the continuous ones. Section 3 presents three applications of *Chebpack* to higher order eigenvalue problems, Lyapunov-Schmidt reduction method and evolution equations. Section 4 contains numerical examples illustrating the accuracy and the simplicity of the algorithms of *Chebpack* for some typical higher order problems. Some comparisons with other *Matlab* packages and error evaluation are also given. All the necessary *Matlab* codes for reproducing these examples are now part of an updated version of *Chebpack,* in the folder *Examples,* subfolder *High order problems*.

2 The *tau* method

The Chebyshev spectral method [7] approximates the solution y of a differential problem by a finite sum of a Chebyshev series

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

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

Our *Matlab* package *Chebpack* implements the Chebyshev spectral method as a *tau* method, where we work in the spectral space of the coefficients. The *tau* method was invented by Lanczos (1938, 1956) and later developed in an operatorial approach by Ortiz and Samara [8] [9].

The term "*operatorial*" means that the linear operators that compose the differential or integral problem (such as differentiation, integration, product with the independent variable or with known functions) are discretized to their finite dimensional linear operator versions, expressed by appropriate matrices $M \equiv D,\ J,\ X$, and preserving the *operatorial* structure of the original problem. The discretized version, finally represented by a matrix acting on the Chebyshev coefficients vector of the unknown function includes the additional initial, boundary value or nonlocal linear conditions, again represented by certain vectors that replace certain rows of that matrix.

Chebpack assumes the representation of the unknown functions in truncated Chebyshev polynomial series and the corresponding matrices M are given to act directly on the Chebyshev coefficients. The spectral differentiation matrix behaves better than the corresponding physical one, that is full and sensitive to rounding errors, see [3] for a relevant example. Generally, this kind of discretization is performed by using the recurrence formula and the fast schemes of computation with Chebyshev polynomials, see again [3] for more technical details on this topic.

Precisely, a function $u(x)$ can be represented by its physical representation $u(x_1),\ u(x_2)$, ..., $u(x_n)$ of values of u at a given grid $\mathbf{x}$, for example at *Chebyshev points of the second kind*

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

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

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

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

Of course, the Chebyshev polynomials are defined on $dom = [-1, 1]$ but any interval $dom = [a, b]$ can be shifted to $[-1, 1]$ and we may use the shifted Chebyshev polynomials $T_k^*(x) = T_k(\alpha x + \beta)$, $x \in [a, b]$ with $\alpha = 2/(b-a)$ and $\beta = -(b+a)/(b-a)$. The code `[x,w]=pd(n,dom,kind)` of *Chebpack* calculates the n points x of the kind "`kind`" on dom as a column vector. The weights w are used for approximating the integrals

$$\int_a^b f(x)dx \approx \sum_{i=1}^n w_i f(x_i) \equiv \mathbf{w} \cdot f(\mathbf{x}).$$

For `kind=1`, x are the Chebyshev points of the first kind

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

for `kind=2` x are the Chebyshev points of the second kind (2.1) while for `kind=3` x are the Legendre roots, all these points shifted to dom. For each `kind`, w are the corresponding weights. Chebyshev points are used in the Clenshaw-Curtis quadrature while the Legendre roots are used in the Gauss quadrature [10].

An immediate application of the Legendre points is the weighted inner product of two functions

$$I = \int_a^b f(x)g(x)r(x)dx,$$

with the weight $r(x)$. Here we use the more accurate Gauss formula, i.e. x_{leg} are the Legendre points and w_{leg} are the corresponding quadrature weights given by `kind=3` and then

$$I \approx \sum_{k=1}^n f(x_{leg})g(x_{leg})r(x_{leg})w_{leg}.$$

This formula is implemented in `I = wip( f,g,r,wleg);`

The fast conversion between the spectral representation $\mathbf{c}$ of a function u and its physical values $\mathbf{v} = u(\mathbf{x})$ for `kind=1` or `kind=2` is performed by the commands `v=t2x(c,kind)` and `c=x2t(v,kind)` based on the Fast Chebyshev Transform. It is important to remark that linear operators $L(u(x))$ are better represented directly in the spectral space while the nonlinear operators $N(u(x))$ are easily handled in the physical space through the scheme

$$\texttt{c}\longrightarrow\texttt{v=t2x(c,kind)}\longrightarrow\texttt{vn=}N(v)\longrightarrow\texttt{cn=x2t(vn,kind)}.$$

Here `c` are the coefficients of u while `cn` are the coefficients of $N(u)$.

If we need a matrix form T of these conversions, it is obtained from the code `T=cpv(n,xc,dom)`, where $\mathbf{x}_c \in dom$ is an arbitrary column vector and

$$T = [T_0/2, T_1(\xi), ..., T_{n-1}(\xi)]\,,\ \xi = \frac{2\mathbf{x}_c}{b-a} - \frac{b+a}{b-a} \in [-1, 1] \tag{2.3}$$

is the matrix containing the values of the Chebyshev polynomials on the grid ξ. This code is based on the recurrence relations

$$T_1(x) = xT_0,\ T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x),\ k = 1, 2, ... \tag{2.4}$$

and we have $\mathbf{v} = T\mathbf{c}$ and $\mathbf{c} = T^{-1}\mathbf{v}$.

If we need the roots of a function $u : [a, b] \to \mathbb{R}$, we first approximate this function by a combination P of n Chebyshev polynomials by using `x2t`. The roots of P are the eigenvalues of the corresponding colleague matrix C [2] [11]. This zerofinder utility is implemented in Chebpack by

```
[z,uz,m]=chebroots(c,dom,loc,tol),
```

where the polynomial P is given by its Chebyshev coefficients `c` for the general computational domain `dom`. At output, `z` are the roots in the interval `loc`, $uz = p(z)$ (for test) while `tol` is a tolerance used for retaining only the first `m` coefficients from `c` and for selecting the roots $z > loc(1) + tol$.

Another useful code is `X=mult(n,dom)` also based on the recurrence relations (2.4). If $\mathbf{c}$ is the column of Chebyshev coefficients of a function $u(x)$, then $X \cdot \mathbf{c}$ is the column of the Chebyshev coefficients of $xu(x)$ for $x \in dom$. If `a` is the vector of the first m Chebyshev coefficients of a function $a(x)$, then $A \cdot \mathbf{c}$ is the vector of the n coefficients of $a(x)y(x)$. The multiplication matrix A is given by the code `A=fact(a,m)` based on the relations

$$2T_j(x)T_k(x) = T_{j+k}(x) + T_{|k-j|}(x), j, k = 0, 1, ... \tag{2.5}$$

Of course, if $a(x)$ is a short polynomial like $a(x) = x^2 - 1$ for example, then $A = X^2 - E$ where E is the unit matrix of size n. If $a(x)$ is an elementary function, for example $a(x) = e^x$, then $A \approx e^X \equiv expm(X)$, the corresponding matrix exponential function.

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

$$T_0 = T_1', \; T_1 = \frac{T_2'}{4}, \; T_k = \frac{T_{k+1}'}{2(k+1)} - \frac{T_{k-1}'}{2(k-1)}, \; k = 2, 3, ..., \tag{2.6}$$

see [3] [4] for details.

Similarly, the code `[J,J0]=prim(n,dom)` calculates the integration matrix J such that the coefficients of a primitive of a function u with the Chebyshev coefficients $\mathbf{c}$ are $J \cdot \mathbf{c}$. Here the first coefficient of the result $J \cdot \mathbf{c}$ may be changed in order to obtain the coefficients for a specific primitive of u. For example, the coefficients of the primitive which vanishes at $a = dom(1)$ are obtained by using $J_0 \cdot \mathbf{c}$. All the above codes take into account a general dom.

The basic results for the spectral approximations are given by the following theorems from [2]:

Theorem 8.1 (Chebyshev coefficients of analytic functions). *Let a function u analytic in $[-1, 1]$ be analytically continuable to the open Bernstein ellipse E_ρ, where it satisfies $|u(x)| \leq M$ for some M. Then its Chebyshev coefficients satisfy $|a_0| \leq M$ and $|a_k| \leq 2M\rho^{-k}$, $k \geq 1$.*

Theorem 8.2 (Convergence for analytic functions)**.** *If u has the properties of the above theorem, then for each $n \geq 0$ its Chebyshev projections u_n and its Chebyshev interpolants p_n satisfy respectively*

$$\|u - u_n\| \leq \frac{2M\rho^{-n}}{\rho - 1}, \; \|u - p_n\| \leq \frac{4M\rho^{-n}}{\rho - 1}.$$

Theorem 21.1 (Geometric convergence of derivatives). *Let a function u analytic in $[-1, 1]$ be analytically continuable to the closed Bernstein ellipse $\overline{E}_\rho$ for some $\rho > 1$. Then for each integer $\nu \geq 0$, the ν-th derivatives of the Chebyshev projections u_n and interpolants p_n* satisfy as $n \to \infty$

$$\left\| u^{(\nu)} - u_n^{(\nu)} \right\| = O(\rho^{-n}), \; \left\| u^{(\nu)} - p_n^{(\nu)} \right\| = O(\rho^{-n}).$$

The above theorem ensures that if u is an analytic solution to a differential problem $\mathcal{L}u = f$, then the Chebyshev interpolants of $\mathcal{L}u$ would converge geometrically to u as $n \to \infty$. In spectral computations we must approximate this exact solution by solving matrix problems $L\mathbf{u} = \mathbf{f}$.

For linear differential equations with polynomial coefficients we may also follow the ideas from [9] [12]. The main quality of the *tau* method is that *the numerical solution of the exact problem is also an exact solution of a perturbed problem*. For some fixed $n' \geq n$, we associate with $\mathcal{L}u = f$ the following problem

$$\mathcal{L}\mathbf{u} = f + \sum_{k=0}^{k'} \tau_{n',k} T^*_{n'-k}, \tag{2.7}$$

where $\mathbf{u}$ is the above numerical solution represented by its coefficients filled with zeros to the dimension n'. Here $\mathbf{u}$ is called the n*-th tau method approximation* to u, $e_n = u - \mathbf{u}$ is the n*-th tau error* and

$$\sum_{k=0}^{k'} \tau_{n',k} T^*_{n'-k} = \rho_{n'}$$

is called the *tau perturbation term* (or the residual). The *tau* perturbation term $\rho_{n'}$ can be calculated by reconstructing the matrix L to the dimension n' obtaining the matrix $\widehat{L}$ and using the remark that in this case $\mathcal{L}\mathbf{u}$ is exactly represented by $\widehat{L}\mathbf{u}$ (of course, limited by the computer rounding errors). Subtracting the perturbed equation from the exact one we obtain the error equation $\mathcal{L}e_n = -\rho_{n'}$. We can now *estimate* the n-th Tau error e_n by using the extended discretization matrix $\widehat{L}$. Of course, for nonlinear equations the problem is more complicated.

3 Applications

3.1 Eigenvalues and eigenvectors

A general form for a Sturm-Liouville problem is

$$\begin{gathered} -\big(p(x)u'(x)\big)' + q(x)u(x) = \lambda r(x)u(x),\ x \in [a,b]\,, \\ c_1u(a) + p(a)u'(a) = 0,\ c_2u(b) - p(b)u'(b) = 0 \end{gathered} \tag{3.1}$$

where $p \in C^1[a,b]$, $q, r \in C[a,b]$, $p > 0$, $r \geq 0$ on $[a,b]$. There is an infinite number of eigenvalues λ_n that can be ordered such that $\lambda_0 < \lambda_1 < ... \to \infty$. The corresponding eigenfunctions ϕ_n form an orthogonal basis on the Hilbert space $L^2_r(a,b)$. Sometimes one wishes to compute a large number of eigenvalues and then a higher order accuracy of the methods is needed [13].

The are similar problems for the fourth-order Sturm–Liouville problem

$$\big(p(x)u''(x)\big)'' - (s(x)u'(x))' + q(x)u(x) = \lambda r(x)u(x),\ x \in [a,b]\,, \tag{3.2}$$

with four boundary conditions on u, u', $p(x)u''$ and/or $(p(x)u'')' - s(x)u'$ specified at boundary $\{a,b\}$. Under suitable conditions, the problem has an infinite sequence of eigenvalues λ_k, $k = 0, 1, ...$ which are bounded from below and can be ordered as an increasing sequence, $\lambda_0 \leq \lambda_1 \leq ... \to \infty$. In this case, each eigenvalue has the multiplicity of at most 2 [14]. These results are extended to higher order Sturm-Liouville problems [15].

The Chebyshev *tau* method can be applied and we get the following equation for (3.1)

$$[-D * p(X) * D + q(X)] * c = \lambda * r(X) * c,$$

where D is the differentiation matrix and X is the multiplication by x matrix. Here $p(X)$, $q(X)$ and $r(X)$ are exponential versions of the functions p, q and r. In matrix form, this is a generalized eigenvalue problem $A * c = \lambda * R * c$. If $r \equiv 1$ then $R = E$, the unit matrix of size n.

The boundary conditions could be implemented by changing the last two rows of A by

$$\begin{gathered} A(n-1,:) = c_1T(1,:) + p(a) * T(1,:) * D \\ A(n,:) = c_2T(2,:) - p(b) * T(2,:) * D \\ T = cpv(n, dom, dom),\ dom = [a,b] \end{gathered}$$

and the last two rows of R by zeros, obtaining new matrices $\widetilde{A}$, $\widetilde{R}$. Here, we take into account that $T(1,:) * c = u(a)$, $T(2,:) * c = u(b)$, $T(1,:) * D * c = u'(a)$ and $T(2,:) * D * c = u'(b)$. Now, the generalized eigenvalue problem $\widetilde{A} * c = \lambda * \widetilde{R} * c$ has two infinity eigenvalues (generated by the two zero rows of $\widetilde{R}$) and, depending on n, a third to a quarter of numerical eigenvalues approximate well the exact ones.

Let us describe this *tau* method on a very simple example from [13], with boundary conditions containing the parameter λ.

Consider the Sturm-Liouville problem

$$-y'' + e^x y = \lambda y,\ x \in [0,1]\,,$$
$$y(0) = 0,\ \sqrt{\lambda}\sin(\sqrt{\lambda})y(1) - \cos(\sqrt{\lambda})y'(1) = 0.$$

The algorithm is the following: for each given λ

- consider the initial conditions $y(0) = 0,\ y'(0) = 1$
- calculate the solution $y(x;\lambda)$ and $y'(x,\lambda)$ at $x = 1$. This means solving

$$\begin{pmatrix} [-D^2 + e^X - \lambda E]_{(1\ldots n-2)\times(1\ldots n)} \\ T(1,:) \\ T(1,:)*D \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} [\mathbf{0}]_{(1\ldots n-2)\times 1} \\ 0 \\ 1 \end{pmatrix}$$

from where we obtain the Chebyshev coefficients $\mathbf{c} = (c_0, ..., c_{n-1})^T$ of the solution $y(x,\lambda)$. Now, $y(1,\lambda) = T(2,:)*\mathbf{c}$ and $y'(1,\lambda) = T(2,:)*D*\mathbf{c}$,

- evaluate $F(\lambda) = \sqrt{\lambda}\sin(\sqrt{\lambda})y(1,\lambda) - \cos(\sqrt{\lambda})y'(1,\lambda)$
- solve the equation $F(\lambda) = 0$ by using `chebroots.m` for intervals of interest for λ; for this, we take m values of $\lambda_k,\ k = 1, ..., m$ at each interval $interval$, given by $\lambda = pd(m, interval, 2)$ and the values of $F(\lambda_k)$ give through `x2t` the Chebyshev coefficients of the function F on the $interval$. Now apply `chebroots` and obtain the roots - the approximate eigenvalues of the original problem - at that $interval$.

These calculations are coded in `ElDaou.m`. The command

```
lambda=ElDaou(96,0:10:240,64);
```

i.e. for $n = 96$, on the intervals $[0,10]$, $[10,20]$,..., $[230,240]$ for λ, with 64 grid points for λ on each interval, gives the 10 exact eigenvalues from Table 2 from [13].

The same method could be applied for the problem (3.2). We illustrate the method on the following example from [16],

$$u^{(4)} = \lambda u'',\ -1 < x < 1, \qquad (3.3)$$
$$u(\pm 1) = u'(\pm 1) = 0.$$

The analytical eigenvectors are: even modes $u(x) = 1 - \cos(n\pi x)/\cos(n\pi)$ with $\lambda = -n^2\pi^2$ and odd modes $u(x) = x - \sin(q_n x)/\sin(q_n)$ with $\lambda = -q_n^2$ where $q_n = \tan q_n,\ n\pi < q_n < 2n+1)\pi/2$ for all integer $n > 0$. All the exact eigenvalues are real, negative and distinct.

The direct method, implemented in `eig42D.m` defines the matrices $A = D^4$ and $B = D^2$. The problem becomes $A\mathbf{c} = \lambda B\mathbf{c}$, where $\mathbf{c}$ is the column of the Chebyshev coefficients of the eigenvector corresponding to λ. In order to implement the boundary conditions, we calculate $T = cpv(n, [-1,1], [-1,1])$ and consequently, we must replace the last four rows of A by

```
[T(1,:);T(2,:);T(1,:)*D;T(2,:)*D]
```

and the last four rows of B by zeros, obtaining the matrices $\widetilde{A}$ and $\widetilde{B}$.

This Chebyshev *tau* method provides spectrally accurate approximations to the lower magnitude eigenvalues but it also yields four infinity eigenvalues generated by the zero rows of $\widetilde{B}$ *and two large positive eigenvalues*, that are *spurious*.

A simple change from Chebyshev polynomials $T_j(x)$ to $(1-x^2)^2T_j(x)$ eliminates all those spurious eigenvalues, as it is proven in [17] and coded here in `eig42F.m`. If we use the Galerkin method with these basic functions verifying the boundary conditions,

$$u(x)=\sum_{j=0}^{n-5}c_j(1-x^2)^2T_j(x)=(1-x^2)^2\sum_{j=0}^{n-5}c_jT_j(x)$$

the coefficients $\mathbf{c}$ are expressed in the new basis $(1-x^2)^2T_j(x)$ by $(E-X^2)^2\mathbf{c}$ where $c_{n-4}=...=c_{n-1}=0$. The problem becomes

$$D^4(E-X^2)^2\mathbf{c}=\lambda D^2(E-X^2)^2\mathbf{c}$$

In order to pass the results again to Galerkin basis $(1-x^2)^2T_j(x)$, we must transform the above problem to

$$(E-X^2)^2D^4(E-X^2)^2\mathbf{c}=\lambda(E-X^2)^2D^2(E-X^2)^2\mathbf{c}.$$

This means that the discrete form of the eigenproblem (3.3) is $\overline{A}\overline{\mathbf{c}}=\lambda\overline{B}\overline{\mathbf{c}}$ where the matrices

$$A=(E-X^2)^2D^4(E-X^2)^2,\ B=(E-X^2)^2D^2(E-X^2)^2$$

must be truncated by eliminating the last four rows and the last four columns to obtain $\overline{A}$, $\overline{B}$ and $\overline{\mathbf{c}}=[c_0,...,c_{n-5}]^T$. This problem gives the eigenvalues (without any spurious eigenvalue) while the eigenfunctions in the original $T_j(x)$ basis are given by $(E-X^2)^2\mathbf{c}$ where again $c_{n-4}=...=c_{n-1}=0$.

The computation time is the same as for using DMS (`eig42DMS.m`)[18], i.e. 0.016 sec. with a difference of about $1.e-7$ between the first 26 eigenvalues. *Chebfun* (`eig42CHEBFUN.m`) [1] needs 3.16 sec. by using an adaptive algorithm.

In order to estimate the errors we apply the ideas from [19]. Let us consider again the problem $\widetilde{A}c=\lambda\widetilde{B}c$ with the boundary conditions enclosed in the last four rows of $\widetilde{A}$ and $\widetilde{B}$. These matrices will be partitioned as

$$\begin{pmatrix} A_{11} & A_{12}\\ A_{21} & A_{22}\end{pmatrix}\begin{pmatrix} c'\\ c''\end{pmatrix}=\lambda\begin{pmatrix} B_{11} & B_{12}\\ B_{21} & B_{22}\end{pmatrix}\begin{pmatrix} c'\\ c''\end{pmatrix}$$

where the size of the blocks is $(A_{11})_{(n-4)\times(n-4)}$, $(A_{12})_{(n-4)\times 4}$, $(A_{21})_{4\times(n-4)}$, $(A_{22})_{4)\times 4}$ and the same for $\widetilde{B}$ while the size of the vectors is $(c')_{(n-4)\times 1}$ and $(c'')_{4\times 1}$.

We calculate the eigenvalues and the eigenvectors from

$$\begin{aligned}&A_{11}c'+A_{12}c''=\lambda\left(B_{11}c'+B_{12}c''\right),\\ &A_{21}c'+A_{22}c''=0.\end{aligned}$$

From the second equation we have $c''=-A_{22}^{-1}A_{21}c'$ and then the problem becomes

$$(A_{11}-A_{12}A_{22}^{-1}A_{21})c'=\lambda(B_{11}-B_{12}A_{22}^{-1}A_{21})c'.$$

The numerical solutions λ and $c=[c';-A_{22}^{-1}A_{21}c']$ verifies *exactly* the equation (2.7)

$$Ac-\lambda Bc=\begin{pmatrix}\mathbf{0}\\ \tau\end{pmatrix}$$

where $(\tau)_{4\times 1}$. The last four rows give the four coefficients τ and this is a measure of the accuracy for each eigenvalue λ and corresponding eigenvector c.

The program `eig42E.m` gives the corresponding norms of τ in Table 1 for the first 3 eigenvalues. We remark that this elimination method does not discard the spurious eigenvalues.

There is another problem with the order of growth of the entries of D^4 which is $O(n^7)$ and can lead to serious round off error problems. We use instead FD^4F where $F=(E-X^2)^2$. A numerical

Table 1: The norm of the τ coefficients for the first three eigenvalues

#	λ	$\|\tau\|$
1	-9.869584957916786	0.000000072237616
2	-20.190724958589310	0.000000560578509
3	-39.478422984210063	0.000000134315862

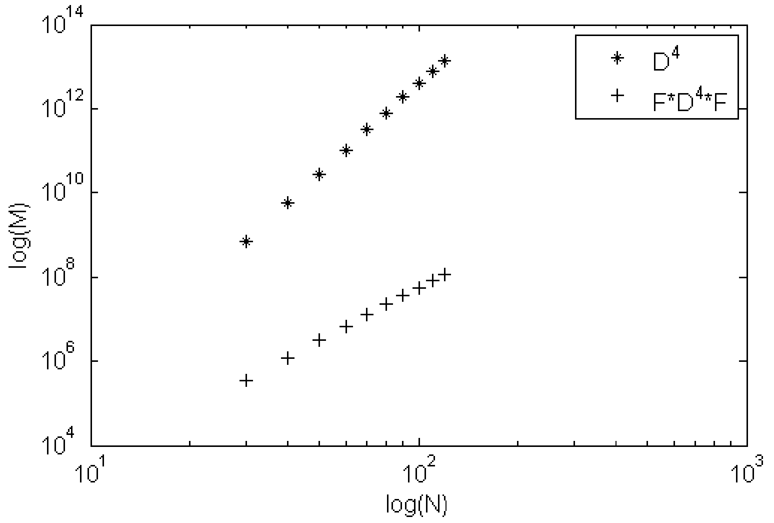

Figure 1: Growth order for D^4 and FD^4F

experiment with `magnitude.m`, for `n=20:10:120` shows in Figure 1 the maximum absolute value of the terms $M = \max(\max(abs(D^4)))$ versus the dimension $n.$ It is $O(n^{7.13})$ for D^4 and only $O(n^{4.2})$ for FD^4F (between $O(n^5)$ for D^3 and $O(n^3)$ for D^2). Moreover, the structure of the matrix FD^4F is improved compared to that of the matrix D^4.

The case of a second order Surm-Liouville problem is solved in an older Matlab package *LiScEig* [20]. The above case of higher dimension will be included in an extended version of *Chebpack*.

3.2 Lyapunov-Schmidt reduction method

Let us suppose that we have to solve the nonlinear *steady* differential problem $Lu = Nu$ together with boundary conditions, where L is a linear operator and $N = N(x, u(x))$ is a nonlinear operator

$$Lu = -(p(x)u')' + q(x)u, \in [a, b],$$
$$\alpha_{11}u(a) + \alpha_{12}u'(a) = 0, \ \alpha_{21}u(b) + \alpha_{22}u'(b) = 0.$$

If the operator L admits the eigenfunctions $\phi_k, k = 1, 2, ...$ and the eigenfunctions form an orthonormal basis in the domain of L, then we can search the solution u by expansion in eigenfunctions of L

$$u = \sum_{k=1}^{n} c_k \phi_k.$$

Following Orszag [21] we may remark that the infinite-order accuracy of Chebyshev polynomial approximations to infinitely differentiable functions should be contrasted with the situation when expansions are made in terms of orthogonal eigenfunctions where only finite-order rates of convergence are obtained. However, the discretized form of L only involves sparse matrices. The pseudospectral differentiation matrices are full and even though we work in spectral space of the coefficients, some matrices like e^X are also full. Finally, we must solve a large nonlinear system of equations for the coefficients perturbed by imposing the boundary value conditions. The eigenfunctions of L verify the boundary conditions and the linear part of the discretized system becomes diagonal.

Indeed,

$$Lu \equiv L\left(\sum_{k=1}^{n} c_k\phi_k\right) = \sum_{k=1}^{n} c_k\lambda_k\phi_k = N\left(\sum_{k=1}^{n} c_k\phi_k\right) = \sum_{k=1}^{n} C_k\phi_k$$

where

$$C_i = \int_a^b N\left(\sum_{k=1}^{n} c_k\phi_k\right)\phi_i,\ i = 1, ..., n.$$

Projecting on the eigenfunctions we obtain the nonlinear system

$$c_k\lambda_k = C_k,\ k = 1, ..., n.$$

The Lyapunov-Schmidt reduction method is the following, see [22] for more details. If we choose a small index m and project the equation $Lu = Nu$ only on $sp\{\phi_{m+1}, ..., \phi_n\}$ we obtain

$$c_k = \frac{C_i(c_1, ..., c_n)}{\lambda_k},\ k = m+1, ..., n. \tag{3.4}$$

For a sufficiently great m and for fixed $c_1, ..., c_m$, the above operator becomes a contraction and we can iterate until a fixed point is reached

$$c^* = (c_1, ..., c_m, c^*_{m+1}, ..., c^*_n),$$

which is a solution of the equation (3.4), *the associated function* to $(c_1, ..., c_m)$. Given $(c_1, ..., c_m)$ the associated function coefficients are computed by `as.m`. Of course, c^*_i, $i = m+1, ..., n$ depend on c_i, $i = 1, ..., m$.

Now, let us project the equation $Lu = Nu$ on $sp\{\phi_1, ..., \phi_m\}$,

$$c_k\lambda_k = C_k(c_1, ..., c_m, c^*_{m+1}, ..., c^*_n),\ k = 1, ..., m$$

and obtain a nonlinear small system for $c_1, ..., c_m$. Given $c_1, ..., c_m$, each evaluation of

$$c^* = (c_1, ..., c_m, c^*_{m+1}, ..., c^*_n)$$

requires fixed point iterations (3.4). We solve this system by Newton method (the Jacobian of C is obtained again by fixed point iterations) and finally we obtain the coefficients of the solution

$$c^* = (c^*_1, ..., c^*_m, c^*_{m+1}, ..., c^*_n)$$

(i.e. the solution $u = \phi \cdot \mathbf{c}^*$) of the problem $Lu = Nu$. These calculations are implemented in `lisc.m`. We remark that $m = 0$ means pure fixed point iterations while $m = n$ means pure Galerkin's method.

There is a natural extension for equations of the form $Lu = N(x, u(x), u'(x))$, that is

$$L\left(\sum_{k=1}^{n} c_k\phi_k\right) = N\left(x, \sum_{k=1}^{n} c_k\phi_k, \sum_{k=1}^{n} c_k\phi'_k\right)$$

or for higher order operators L.

The case of a second order Surm-Liouville operator L is solved in an older Matlab package *LiScNLS* [23]. The extension for higher dimension will be included into an extended version of *Chebpack*.

3.3 Evolution problems

If we have to solve numerically a nonlinear *evolution* problem,

$$u_t + \mathbf{L}u = \mathbf{N}u$$

like KdV problem (Example 6),

$$\begin{aligned} &u_t + u_{xxx} + 6uu_x = 0,\ x \in (-L, L),\ t \in [0, t_{final}] \\ &u(-L,t) = u(L,t) = u_x(L,t) = 0,\ u(x,0) = u_0(x) \end{aligned} \tag{3.5}$$

we first perform a spatial semidiscretization using the *tau* spectral method and we obtain a differential system for the expansion coefficients.

The solution is represented as a truncated series

$$u_N(x,t) = \sum_{k=0}^{N-1} c_k(t) T_k^*(x) \tag{3.6}$$

and we denote by $\mathbf{c}(t)$ the column vector of the coefficients. From (3.5) we obtain the differential system

$$\mathbf{c}'(t) + D^3\mathbf{c}(t) = N(\mathbf{c}(t)),\ \mathbf{c}(0) = \mathbf{c}_0$$

where

```
N(c(t))=-6*x2t(t2x(c,kind)*t2x(D*c,kind),kind), c0=x2t(u0(x),kind).
```

Here x are the Chebyshev points of the second kind and `dom=[-L,L]`. This system has the general form

$$\frac{\partial \mathbf{c}}{\partial t} + \mathcal{L}\mathbf{c} = \mathcal{N}[\mathbf{c}], \tag{3.7}$$

where $\mathcal{L}$ is the linear part and $\mathcal{N}$ is the nonlinear part and it is a stiff system, with a *sparse* matrix $\mathcal{L}$. As above, we obtain a new system with boundary condition enclosed. This system is partitioned as

$$\begin{pmatrix} c^{(1)} \\ c^{(2)} \end{pmatrix}' + \begin{pmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} c^{(1)} \\ c^{(2)} \end{pmatrix} = \begin{pmatrix} N^{(1)} \\ N^{(2)} \end{pmatrix}$$

where

$$\begin{gathered} c^{(1)} = \mathbf{c}(1:n-3),\ c^{(2)} = \mathbf{c}(n-2:n), \\ N^{(1)} = \mathcal{N}[\mathbf{c}](1:n-3),\ N^{(2)} = \mathcal{N}[\mathbf{c}](n-2:n), \\ L_{11} = \mathcal{L}(1:n-3, 1:n-3),\ L_{12} = \mathcal{L}(1:n-3, n-2:n), \\ L_{21} = \mathcal{L}(n-2:n, 1:n-3),\ L_{22} = \mathcal{L}(n-2:n, n-2:n). \end{gathered}$$

Consequently, (3.7) becomes

$$\begin{gathered} c^{(1)\prime} + L_{11}c^{(1)} + L_{12}c^{(2)} = N^{(1)}, \\ 0 + L_{21}c^{(1)} + L_{22}c^{(2)} = 0. \end{gathered}$$

From the second equation which represents the boundary conditions, we obtain

$$c^{(2)} = -L_{22}^{-1} L_{21} c^{(1)} \tag{3.8}$$

and then the first equation becomes

$$c^{(1)\prime} + \left(L_{11} - L_{12}L_{22}^{-1}L_{21}\right)c^{(1)} = N^{(1)}. \tag{3.9}$$

The system (3.9) is integrated with the initial condition $c^{(1)}(0) = c_0^{(1)}$ and finally $c^{(2)}$ is obtained from (3.8).

In order to avoid the small time step imposed by explicit integrators, we will use an exponential time differencing integrator for (3.9). Let $A = L_{11} - L_{12}L_{22}^{-1}L_{21}$.

If we multiply (3.7) by the exponential matrix e^{At} as an integrating factor, we obtain

$$\frac{d}{dt}\left(e^{At}c^{(1)}(t)\right) = e^{At}N^{(1)}\left[\mathbf{c}(t)\right].$$

By integrating between time levels t_n and t_{n+1} and denoting $c_n^{(1)} = c^{(1)}(t_n)$, we are led to

$$e^{At_{n+1}}c_{n+1}^{(1)} - e^{At_n}c_n^{(1)} = \int_{t_n}^{t_{n+1}} e^{A\tau}N^{(1)}\left[\mathbf{c}(\tau)\right]d\tau,$$

and finally we obtain

$$c_{n+1}^{(1)} = e^{-Ah}c_n^{(1)} + \int_0^h e^{-A(h-\tau)}N^{(1)}\left[\mathbf{c}(t_n+\tau)\right]d\tau,$$

where $h = t_{n+1} - t_n$ is the time step.

Now we approximate $N^{(1)}\left[\mathbf{c}(t_n+\tau)\right]$ by a polynomial in τ and calculate the integral exactly. Assuming $N^{(1)}\left[\mathbf{c}(t_n+\tau)\right] = constant = N^{(1)}\left[\mathbf{c}(t_n+h)\right] = N_{n+1}^1$ we obtain the *implicit exponential time differencing Euler* method,

$$c_{n+1}^{(1)} = e^{-Ah}c_n^{(1)} + h\Phi_1(-Ah)N^{(1)}\left[\mathbf{c}_{n+1}\right],\ c_{n+1}^{(2)} = -L_{22}^{-1}L_{21}c_{n+1}^{(1)}$$

where $\Phi_1(M) = M^{-1}(e^M - I)$ and I is the unit matrix. These methods can be combined into a symmetric exponential integrator

$$\begin{gathered} c^{(1)}{=}e^{-A\frac{h}{2}}c_n^{(1)} + \frac{h}{2}\Phi_1(-A\frac{h}{2})N^1\left[\mathbf{c}\right],\ c^{(2)} = -L_{22}^{-1}L_{21}c^{(1)} \\ c_{n+1}^1 = e^{-A\frac{h}{2}}c^{(1)} + \frac{h}{2}\Phi_1(-A\frac{h}{2})N^{(1)}\left[\mathbf{c}\right],\ \ c_{n+1}^{(2)} = -L_{22}^{-1}L_{21}c_{n+1}^{(1)}, \\ n = 0,1,...,\ \ \mathbf{c}_0 = \mathbf{c}(0) \end{gathered} \tag{3.10}$$

and we refer to [24] for the properties of the symmetric exponential integrators.

The first (implicit) relations are solved with respect to $\mathbf{c} = \left(c^{(1)}, c^{(2)}\right)^T$ by fixed point iterations

$$c_{(k+1)}^{(1)}{=}e^{-A\frac{h}{2}}c_n^{(1)} + \frac{h}{2}\Phi_1(-A\frac{h}{2})N^1\left[\mathbf{c}_{(k)}\right],\ c_{(k+1)}^{(2)} = -L_{22}^{-1}L_{21}c_{(k+1)}^{(1)},\ \mathbf{c}_{(0)} = \mathbf{c}_n$$

that, for a sufficiently small time step h, converge toward $\mathbf{c}$. The last relations give now the approximate solution $\mathbf{c}_{n+1}$ at the next time level t_{n+1}.

The case of a second order Surm-Liouville operator $\mathbf{L}$ is solved in an older Matlab package *LiScNLE* [25] which solves (3.7) by Crank-Nicolson or backward-Euler methods. *Chebpack* solves (3.9) with exponential time differencing method. The above algorithms for higher dimension will be included in an extended version of *Chebpack*.

4 Numerical examples

Example 1. Let us calculate the eigenvalues of the third order problem [26]

$$\begin{gathered} v''' = \lambda v,\ x \in [-1,1], \\ v(-1) = v'(-1) = v(1) = 0. \end{gathered}$$

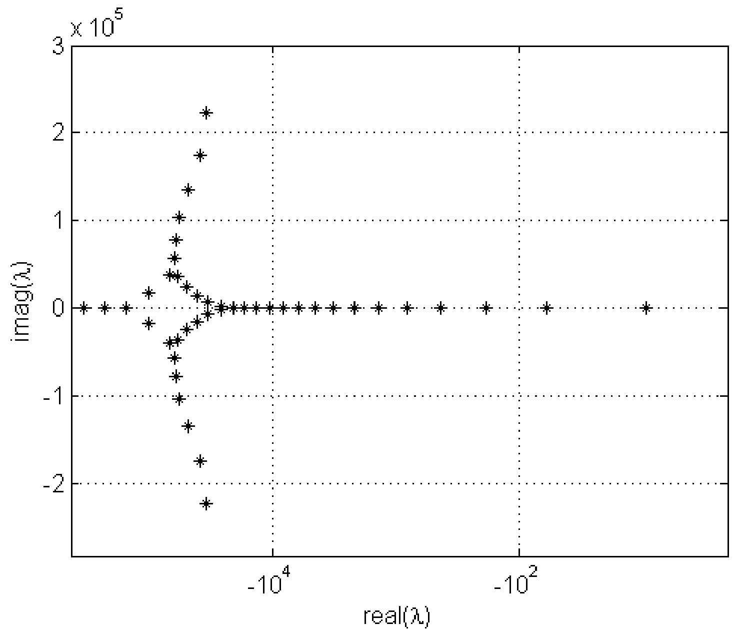

Figure 2: The eigenvalues for Example 1, $n = 64$

The code `KdV_eig.m` uses the factor $F = (E - X)(E + X)^2$ to include the boundary condition in the problem. The exact eigenvalues are real and negative, defined by the zeroes of the function

$$f(\lambda) = e^{3\lambda^{1/3}} - 2\sin\left(\sqrt{3}\lambda^{1/3} + \frac{\pi}{6}\right).$$

For $n = 64$, the first 15 numerical eigenvalues are real and negative but the other are complex, see Figure 2. Table 2 contains the values of λ_k and $f(\lambda_k)$ for $k = 1, 2, 3$. The eigenfunctions are not orthogonal. *Chebpack* needs 0.13 sec. to obtain 15 eigenvalues while *Chebfun* (`KdV_eig_CHEBFUN.m`) needs 1 sec. with much larger values of $f(\lambda_k)$.

Table 2: First three eigenvalues for example 1

#	$\lambda * \mathbf{1.e-4}$	$\mathbf{f}(\lambda)$
1	-0.000948240693555	-0.00000000001
2	-0.006069365841144	-0.00000000003
3	-0.018948497984446	-0.00000000021

It is well known (see [27] for a recent analysis of the problem and [28] for historical remarks) that the eigenfunctions of the third order differential operator $S = \pm i\left(-i\frac{\partial}{\partial x}\right)^3$ form or do not form a basis in the domain of S. The corresponding initial-boundary value problem for the partial differential equation $q_t + aSq = 0$ (with $a = \pm i$) is well-posed or ill-conditioned. Consequently, the initial-boundary value problem admits or does not admit a unique solution representable by a discrete series expansion in the eigenfunctions of S. These facts depend on the sign of a and on the value of β ($\beta = 0$, $\beta \neq 0$ or $\beta = \infty$) in the boundary conditions

$$u(0) = u(1) = u'(0) + \beta u'(1) = 0.$$

There exists a biorthogonality between the eigenfunctions ϕ_n and ψ_m of the adjoint problems

$$\begin{array}{ll} \phi''' = -\lambda\phi,\ x \in [0,1], & \psi''' = \lambda\psi,\ x \in [0,1], \\ \phi(0) = \phi'(0) = \phi(1) = 0, & \psi(0) = \psi(1) = \psi'(1) = 0, \end{array}$$

i.e.

$$\int_{-1}^{1} \phi_n \psi_m dx = 0\ for\ m \neq n.$$

The Fourier series of a function $f(x)$ has the form

$$F(x) = \sum_{n=1}^{\infty} \frac{\int_0^1 f(x)\psi_n(x)dx}{\int_0^1 \phi_n(x)\psi_n(x)dx}\phi_n,$$

The code `Lutzen.m` verifies this biorthogonality but with poor accuracy compared to the self-adjoint problems.

Example 2. The *Orr-Sommerfeld equation* [21] governs the linear stability of a two-dimensional shear flow. It can be put in the form

$$\frac{y'''' - 2y'' + y}{R} - 2iy - i(1-x^2)(y'' - y) = c(y'' - y)$$
$$y\,(\pm 1) = y'\,(\pm 1) = 0.$$

This is an eigenvalue problem and an important example is for $R = 5772$. Here we implement the boundary conditions again with the factor $F = (1-x^2)^2$ in the program `OrrSom.m`. The program `OrrSomDMS.m` uses *DMS* [18] and the program `OrrSomCHEBFUN.m` uses *Chebfun* [1]. The results are shown in Figure 3. The computing time is 0.77 sec. for *Chebpack*, 0.22 sec. for *DMS* and 4.5 sec. for *Chebfun* with an adaptive algorithm. For $R = 10000$ the result with $n = 128$ reproduces Table 5 from [21].

Example 3. *A Legendre-like equation* from [14],

$$\big(P(x)y''\big)'' - \big(S(x)y'\big)' = \lambda y,\ x \in (-1,1),\ P(x) = (1-x^2)^2,\ S(x) = 12 - 4x^2,$$
$$-8y'(-1) = \lambda y(-1),\ -8y'(-1) = \lambda y(-1),$$
$$-(Py'')' + Sy' = \frac{\lambda}{x}y - \frac{\lambda}{8x}Py'',\ x = \pm 1.$$

This is an example that contains the eigenparameter in boundary conditions. The exact eigenvalues are

$$\lambda_k = 8k + 16k(k-1) + 8k(k-1)(k-2) + k(k-1)(k-2)(k-3),\ k = 0,1,2,...$$

The implementation is direct in `Greenberg6.m`, i.e. $Ac = \lambda Bc$, where $A = D^2(PD^2) - D(SD)$ and $B = E$ where $P = (E - X^2)^2$, $S = 12E - 4X^2$, E is the unit matrix, X is the multiplication by x matrix and D is the differentiation matrix. By implementing the boundary conditions, the problem becomes

$$\begin{pmatrix} A \\ -8T_1D \\ 8T_2D \\ T_1(-D(PD^2) + SD) \\ T_2(-D(P*D^2) + SD) \end{pmatrix} \cdot \mathbf{c} = \lambda \begin{pmatrix} A \\ T_1 \\ T_2 \\ T_1(-E + PD^2/8) \\ T_2(E - PD^2/8); \end{pmatrix} \cdot \mathbf{c}$$

and the error for the first 10 eigenvalues is less than $1.e-6$ for $n = 96$.

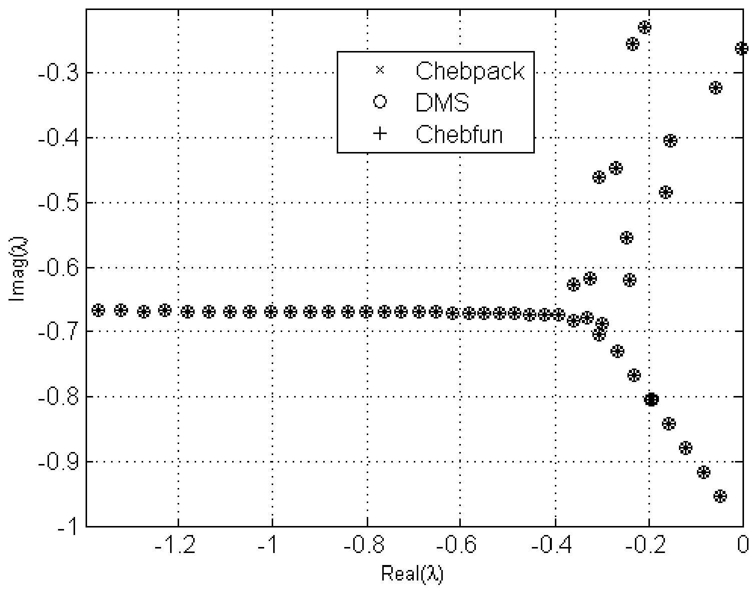

Figure 3: Eigenvalues of the Orr-Sommerfeld problem, $R = 5772$

Example 4. The Kuramoto-Sivashinski equation dates to the mid-1970s and can be written as

$$u_t + \nu\, u_{xxxx} + u_{xx} = -u\, u_x.$$

Periodic boundary value conditions was used in [29] for $\nu = 0.127$ and implemented as an example for Lyapunov-Schmidt method in [25]. We consider here the nonperiodic boundary value conditions used in [30],

$$u(\pm 16\pi, t) = u_x(\pm 16\pi, t) = 0$$

with an initial condition of the form

$$u(x,0) = \varphi(x) \equiv \begin{cases} 0.1 \sin^2\left(\frac{x^2}{2}\right), & -12\pi < x < -10\pi \\ 0 & \text{otherwise} \end{cases}$$

over the domain $[-16\pi, 16\pi]$ and for $\nu = 1$.

We search the solution as an eigenfunction expansion

$$u(x,t) = \sum_{k=1}^{100} c_k(t)\phi_k(x)$$

where $\phi_k(x)$ are the orthonormal system of eigenfunctions of the problem

$$\phi_{xxxx} + \phi_{xx} = \lambda\phi,\ x \in [-16\pi, 16\pi],$$
$$\phi(\pm 16\pi) = \phi_x(\pm 16\pi) = 0.$$

The spatial interval $[-16\pi, 16\pi]$ is discretized with $n = 256$ Legendre points given by

```
[xleg,wleg]=pd(256,[-16*pi,16*pi],3).
```

We calculate the eigenfunctions ϕ_k and the eigenvalues λ_k like in *Example 1*, by using the factor $F = (16\pi + x)^2(16\pi - x)^2$ to implement the boundary conditions. The semidiscretized problem becomes

$$c_k'(t) = -\lambda_k c_k(t) + N(c_1, ..., c_{100})_k,\ k = 1, ..., 100$$

$$c_k(0) = \int_{-16\pi}^{16\pi} \varphi(x)\phi_k(x)dx,\ N(c_1, ..., c_{100})_k = -\int_{-16\pi}^{16\pi} u(x)u_x(x)\phi_k(x)dx.$$

The integrals are calculated by using the function `wip.m`. This initial value problem is solved by using the stiff integrator `ode15s` of Matlab in the program `KS_ode.m`. The result is presented in Figure 4 and it is in a good concordance with the corresponding figure from [30]. The code needs 1.9 sec. for calculating 100 eigenfunction and 2.2 sec. for time evolution for $t \in [0, 15]$. The solution is also

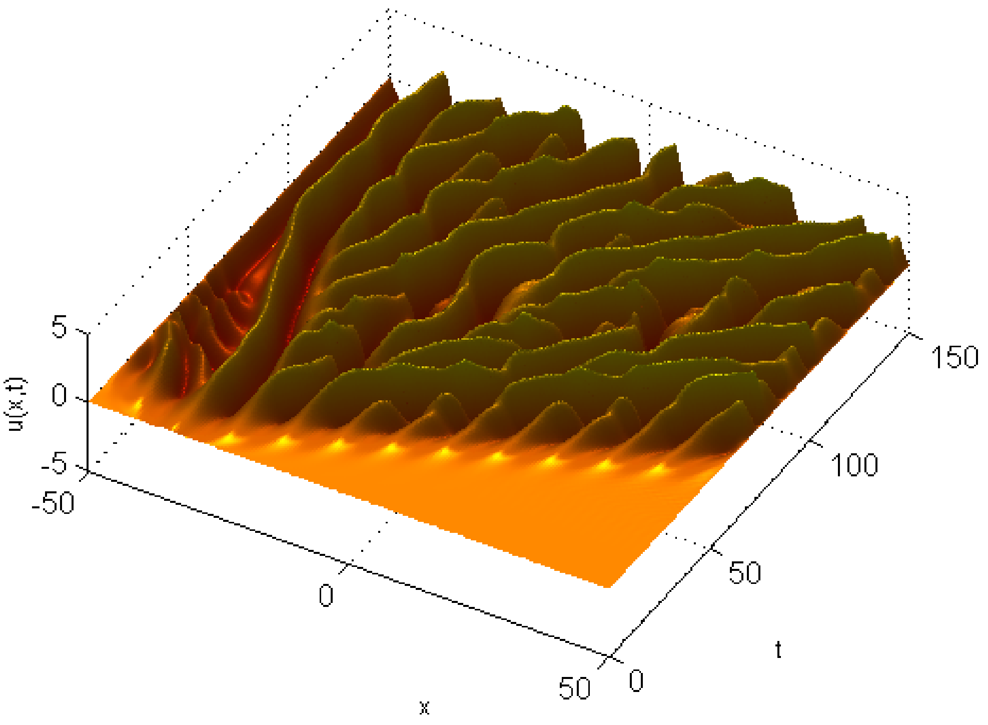

Figure 4: Solution of the Kuramoto-Sivashinski problem

obtained by using the Lyapunov-Schmidt method for the same data in the program `KS_evol.m` which integrates in time the semidiscretized problem by using Crank-Nicolson method with 1500 time steps of $dt = 0.1$. The nonlinear system is solved at each time step by at most 7 fixed point iterations, the time for calculating the eigenfunctions was 1.9 sec. and the time for time evolution was 6.3 sec. The two solutions coincide for $t \in [0, 60]$ while for $t > 60$ the differences increase due to the fact that the solution is chaotic and the two methods have different accuracies.

Example 5. The Boussinesq equation is one of the partial differential equations which models nonlinear dispersive waves and has applications in many areas. It possesses special solutions in the form of *solitons*. The solitons maintain their shape localized within a region while traveling at constant speed. They can interact with other solitons, and emerge unchanged from the collision, except for a phase shift. Nowadays the solitons are used to describe the complex dynamical behavior of wave systems in hydrodynamics, optical fibers, plasmas, shock waves, tornados, etc.

The example of the *"good"* Boussinesq equation has the following form

$$\begin{gathered} u_{tt} - u_{xx} - \left(u^2\right)_{xx} + u_{xxxx} = 0,\ x \in [-50, 50],\ t \in [0, 20]\,, \\ u(x,0) = \varphi(x),\ u_t(x,0) = \psi(x), \\ u(-50,t) = u(50,t) = u_x(-50,t) = u_x(50,t) = 0. \end{gathered}$$

with an exact solution

$$u(x,t) = -\left\{ Asech^2\left(\sqrt{\frac{A}{6}}(x - ct - x_0)\right) + b + \frac{1}{2}\right\}$$

where $A = 0.369$, $b = -0.5$, $x_0 = 0$ and $c = \sqrt{-2\,(b + A/3)}$. The initial conditions are calculated from this exact solution. We refer to [31] for a review on the numerical methods for this problem.

We search here for the solution as an eigenfunction expansion

$$u(x,t) = \sum_{k=1}^{100} c_k(t)\phi_k(x) \tag{4.2}$$

where $\phi_k(x)$ are the orthonormal system of eigenfunctions of the problem

$$\begin{gathered} \phi_{xxxx} - \phi_{xx} = \lambda\phi,\ x \in [-50, 50]\,, \\ \phi(\pm 50) = \phi_x(\pm 50) = 0. \end{gathered}$$

The spatial interval $[-50, 50]$ is discretized with $n = 256$ Legendre points given by

```
[xleg,wleg]=pd(256,[-50,50],3);
```

and we calculate the eigenfunctions ϕ_k and the eigenvalues λ_k like in Example 1, by using the factor $F = (-50 - x)^2(50 - x)^2$ to implement the boundary conditions. The semidiscretized problem (4.2) becomes

$$\begin{gathered} c_k''(t) = -\lambda_k c_k(t) + N(c_1, ..., c_{100})_k,\ k = 1, ..., 100 \\ c_k(0) = \int_{-50}^{50} \varphi(x)\phi_k(x)dx,\ c_k'(0) = \int_{-50}^{50} \psi(x)\phi_k(x)dx, \\ N(c_1, ..., c_{100})_k = -\int_{-50}^{50} \left[u(x)^2\right]'' \phi_k(x)dx. \end{gathered}$$

The integrals are calculated by using the function `wip.m`. This initial value problem is solved again by using the stiff integrator `ode15s` of Matlab. Once the solution is calculated, it is compared with the exact solution. We also test the law of mass conservation

$$\mathcal{M} = \mathcal{M}(t) = \int_{-\infty}^{\infty} u(x,t)dx \approx \sum_{k=1}^{n} wleg_k \cdot u(xleg_k, t)$$

by plotting $\mathcal{M} - \mathcal{M}(t = 0)$. The calculations performed by `Boussinesq.m` need 3.9 seconds to approximate the eigenfunctions and 1.05 seconds to obtain the numerical solution of the whole problem. The test results are:

$$-2.9759032 \leq \mathcal{M}(t) \leq -2.9759024$$
$$err(t) < 1.e-7 \text{ for } t \leq 10$$
$$err(t) < 1.e-6 \text{ for } t \leq 20.$$

Example 6. *The Korteweg – de Vries (KdV) equation* [32] is a time dependent non-linear partial differential equation of third order and among its solutions are again solitary waves (solitons).

The KdV equation is well suited as a test object in applying numerical methods to non-linear PDEs with unbounded spatial domain, since we have analytical solutions and we can appreciate the quality of the numerical approximation. Moreover, the KdV equation admits an infinite number of conservation laws and the corresponding invariants of motion can be used as verification tools for the conservation properties of the numerical scheme.

We test *Chebpack* on the problem

$$u_t + u_{xxx} + 6uu_x = 0,\ x \in (-\infty, \infty),\ t \in [-20, 20]$$
$$u(\pm\infty, t) = 0,\ u(x, 0) = u_0(x).$$

Most works on spectral methods applied to above KdV equation use a periodicity condition on a bounded interval $x \in [-L, L]$ to simulate the behavior at infinity – exponential decay. Here we will use the same truncation of the unbounded spatial domain but we impose artificial homogeneous boundary conditions

$$u(-L, t) = u(L, t) = u_x(L, t) = 0,\ L = 50.$$

We must stop the calculations if the nonzero part of the solution overtakes that boundary, in order to avoid the reflections. In fact, both methods solve a modified problem, not the given problem but they are appropriate since we are not interested in the effect of boundary conditions.

We choose an initial condition

$$u(x, t) = \sum_{i=1}^{3} \beta_i \text{sech}^2\left(\sqrt{\frac{\beta_i}{2}}\,(x - 2\beta_i t)\right),\ t = -20$$
$$\beta_1 = 0.4,\ \beta_2 = 0.7,\ \beta_3 = 1$$

which generates a solution as a nonlinear superposition of three solitons.

We integrate the problem by using the scheme (3.10) for $t \in [-20, 20]$. The exact solution, given in [32] is too complicated to be used for comparison and, of course, the numerical solution is affected by the artificial boundary conditions. We test instead the time invariants of motion

$$I_1 = \int_{-50}^{50} u(x, t)dx,\ I_2 = \int_{-50}^{50} u(x, t)^2 dx,\ I_3 = \int_{-50}^{50} \left[2u(x, t)^3 - u_x(x, t)^2\right] dx.$$

Figure 5 contains the solution given by `KdV.m` with $n = 256$, $dt = 0.05$, 800 time steps and for the steps from 10 to 700 we have

$$I_1 \in [6.9828, 6.9850]\,,\ I_2 \in [3.4659, 3.4675]\,,\ I_1 \in [-3.4201, -3.4162]\,.$$

The code needs 10 iterations per time step and the computation time was about 0.6 sec. per time step.

Example 7. *An eighth order eigenvalue problem with hinged boundary conditions*. This problem comes from electrohydrodynamics and it has been extensively studied numerically in [33]:

$$\left(D^2 - a^2\right)^4 F - La^4 F + Ra^2(D^2 - a^2)F = 0,\ z \in (-0.5, 0.5)$$
$$F = D^2F = D^4F = D^6F = 0\ at\ z = \pm 0.5.$$

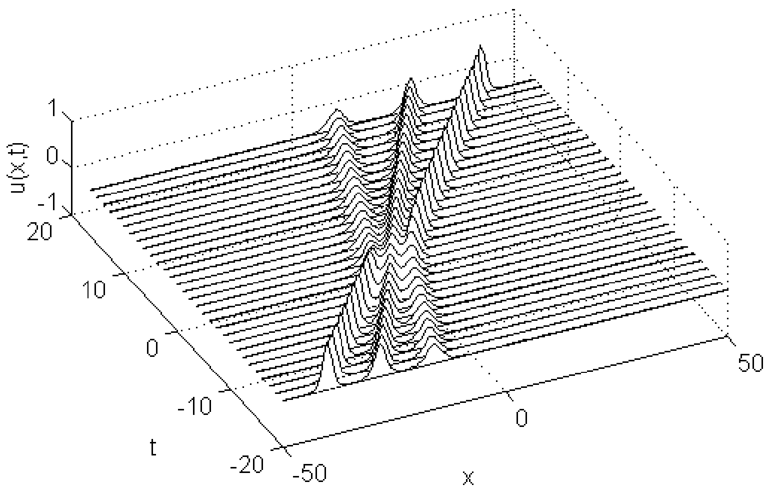

Figure 5: Solution of the KdV problem

Here L and a are physical parameters and R stands for the Rayleigh number whose critical value should be determined.

Following [33], the above eighth order eigenvalue problem is transformed into a second order system of differential equations supplied only with Dirichlet boundary conditions,

$$\begin{gathered} U_1 - (D^2 - a^2)U_4 = 0, \\ U_2 - (D^2 - a^2)U_1 = 0, \\ U_3 - (D^2 - a^2)U_2 = 0, \\ (D^2 - a^2)U_3 - La^4 U_4 = -R\,a^2 U_1, \\ U_1 = U_2 = U_3 = U_4 = 0 \; at \; z = \pm 0.5 \end{gathered}$$

where $U_4 = F, U_1 = (D^2 - a^2)U_4$, $U_2 = (D^2 - a^2)U_1$ and $U_3 = (D^2 - a^2)U_2$.

In matrix form, this problem becomes a generalized eigenvalue problem $\widetilde{A}U = R\,\widetilde{B}U$, where

$$A = \begin{pmatrix} E & O & O & -D^2 + a^2E \\ -D^2 + a^2E & E & O & O \\ O & -D^2 + a^2E & E & O \\ O & O & D^2 - a^2E & -La^4E \end{pmatrix},$$

$$U = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix}, \; B = \begin{pmatrix} O & O & O & O \\ O & O & O & O \\ O & O & O & O \\ -a^2E & O & O & O \end{pmatrix},$$

E being the unit matrix of order n and O being the zero matrix of the same order. The matrices $\widetilde{A}$ and $\widetilde{B}$ are obtained by implementing the Dirichlet boundary conditions in the last two rows of each row of blocks.

The Matlab code is

```
function lam=Dragomirescu_system(n,a,L)
%call lam=Dragomirescu_system(32,sqrt(4.92),1);
dom=[-0.5,0.5];D=deriv(n,dom);T=cpv(n,dom,dom);E=speye(n);Z=zeros(n);
A11=E;A11(n-1:n,:)=T;A22=A11;A33=A11;A44=-L*a^4*E;A44(n-1:n,:)=T;
A14=-D^2+a^2*E;A14(n-1:n,:)=zeros(2,n);A21=A14;A32=A14;A43=-A14;
B41=-a^2*E;B41(n-1:n,:)=zeros(2,n);
A=[A11,Z,Z,A14;A21,A22,Z,Z;Z,A32,A33,Z;Z,Z,A43,A44];
B=[Z,Z,Z,Z;Z,Z,Z,Z;Z,Z,Z,Z;B41,Z,Z,Z];
lam=sort(eig(full(A),full(B)));
```

The above program generates many infinite or spurious eigenvalues due to the singularity of B and to the nature of Chebyshev spectral method but the smallest positive eigenvalue is

$$lam(3) = 657.1806756...$$

We remark that the analytical value for $a = \sqrt{4.92}$ and $L = 1$ is $R_{analytical} = 657.1806...$ The interested reader is encouraged to experiment other values of a and L.

Example 8. The Viola eigenvalue problem. This is a singularly perturbed eigenvalue problem, see [34] for more details:

$$D^2\left[(1-\theta x)^3 D^2 u\right] = \lambda (1-\theta x) u,\ x \in (0,1)$$
$$u = D^2 u = 0\ at\ x = 0,1,$$

where the parameter θ satisfies $0 \leq \theta < 1$.

For $\theta << 1$ the problem is discretized directly as $AU = \lambda BU$, where

$$A = D^2\left[(E-\theta X)^3 D^2\right],\ B = E - \theta X.$$

The boundary value conditions are implemented in the last four rows of A and B.

The Matlab code is

```
function lam=Viola(n,theta)
%call lam=Viola(24,0.5);
dom=[0,1];X=mult(n,dom);D=deriv(n,dom);T=cpv(n,dom,dom);E=speye(n);
A=D^2*((E-theta*X)^3*D^2);B=E-theta*X;
B(n-3:n,:)=zeros(4,n);A(n-3,:)=T(1,:);A(n-2,:)=T(1,:)*D^2;
A(n-1,:)=T(2,:);A(n,:)=T(2,:)*D^2;
L=eig(full(A),full(B));lam=sort(diag(L));
```

The above program generates some infinite or spurious eigenvalues but the smallest positive eigenvalue is

$$\lambda_1 = 50.716230632...$$

We remark that this eigenvalue is confirmed by *Chebfun* [1],

$$\lambda_1 = 50.716216...$$

and it is in the range given by Fichera (Fichera, G., *Numerical and quantitative analysis*, Pitman Press, London, 1978, p. 43),

$$50.71623063 \leq \lambda_1 \leq 50.71623066.$$

For $\theta \to 1^-$ the problem becomes singular. Following [34], the problem is transformed into a second order system of differential equations, supplied only with Dirichlet boundary conditions,

$$(1-\theta x)^3 D^2 u - v = 0,$$
$$D^2 v = \lambda (1-\theta x) u,$$
$$u = v = 0\ at\ x = 0,1.$$

In matrix form, we have a generalized eigenvalue problem $AU = \lambda BU$,

$$\begin{pmatrix} (E-\theta X)^3 D^2 & -E \\ O & D^2 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \lambda \begin{pmatrix} O & O \\ E-\theta X & O \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}.$$

The Matlab program is

```
function lam=Viola_system(n,theta)
%call lam=Viola_system(256,0.9999);
dom=[0,1];X=mult(n,dom);D=deriv(n,dom);T=cpv(n,dom,dom);E=speye(n);
A11=(E-theta*X)^3*D^2;A12=-E;A21=zeros(n);A22=D^2;A=[A11,A12;A21,A22];
A(n-1,:)=[T(1,:),zeros(1,n)];A(n,:)=[T(2,:),zeros(1,n)];
A(2*n-1,:)=[zeros(1,n),T(1,:)];A(2*n,:)=[zeros(1,n),T(2,:)];
B=zeros(2*n);B(n+1:2*n,1:n)=E-theta*X;B(2*n-1:2*n,1:n)=zeros(2,n);
L=eig(full(A),full(B));lam=sort(L);
```

This program generates in 12 sec. many infinite or spurious eigenvalues but the smallest positive eig envalue for $\theta = 0.9999$ is $\lambda_2 = 1.7744...$ We remark that the corresponding eigenfunction has a very large derivative near $x = 1$ and this induces difficulties for spectral methods. *Chebfun* [1] gives almost the same eigenvalue in 254 sec. For $n = 256$ and $\theta = 0.999999$ the above program gives $\lambda_1 = 1.2525...$in 12 sec. while *Chebfun* failed to converge.

5 Conclusions

The Chebyshev spectral tau method implemented in our package *Chebpack* in an operatorial form leads to very simple and efficient codes that can solve higher order linear problems and evolution problems including Lyapunov-Schmidt reduction method with controllable accuracy. The advantage of discretization of linear operators by their finite dimensional versions, expressed by appropriate matrices, is manifested by efficiency of programming and reduction of differential or integral problems to algebraic ones (linear or non-linear algebraic systems, eigenvalue problems for matrices). These finite dimensional problems can then be solved by specific methods. Moreover, the use of spectral Galerkin method for eliminating the appearance of spurious eigenvalues (specific to the tau spectral method for higher order problems) can be simplified by this operatorial treatment.

The algorithms are accompanied by many numerical examples. All the necessary Matlab sources for reproducing the examples are now part of a freely accessible updated version of *Chebpack* [4], in the folder *Examples*, subfolder *High order problems*.

Acknowledgment

The research was supported by a grant of the Romanian National Authority for Scientific Research, CNCS – UEFISCDI, project number PN-II-ID-PCE-2011-3-0094. The author is grateful to both anonymous referees for their careful reading of the paper and suggestions for improvement.

Competing Interests

The author declares that no competing interests exist.

References

[1] Trefethen LN et al. Chebfun Version 4.2. The Chebfun Development Team 2011. Accessed 6 June 2013.
Available: http://www.maths.ox.ac.uk/chebfun/

[2] Trefethen LN. Approximation Theory and Approximation Practice. SIAM; 2013.

[3] Trif D. Matrix based operatorial approach to differential and integral problems. In: Ionescu CM, editor. MATLAB: A Ubiquitous Tool for the Practical Engineer. Rijeka: InTech; 2011. Accessed 6 June 2013.
Available: http://www.intechopen.com/articles/show/title/matrix-based-operatorial-approach-to-differential-and-integral-problems

[4] Trif D. Chebpack 2011. Accessed 6 June 2013.
Available: http://www.mathworks.com/matlabcentral/fileexchange/32227-chebpack

[5] Trif D. Direct operatorial tau method for pantograph type equations. Appl. Math. Comput. 2012;219:2194-2203.

[6] Trif D. Operatorial tau method for some delay equations. Annals of the Tiberiu Popoviciu Seminar. 2012;10:117-137.

[7] Boyd JP. Chebyshev and Fourier Spectral Methods. Dover Publications Inc; 2000.

[8] Ortiz EL, Samara H. An Operational Approach to the Tau Method for the Numerical Solution of Non-Linear Differential Equations. Computing. 1981;27:15-25.

[9] Liu KM, Pan CK. The Automatic Solution to Systems of Ordinary Differential Equations by the Tau Method. Comput. Math. Appl. 1999;38:197-210.

[10] Trefethen LN. Spectral methods in Matlab. SIAM; 2000.

[11] Boyd JP, Gally DH. Numerical experiments on the accuracy of the Chebyshev–Frobenius companion matrix method for finding the zeros of a truncated series of Chebyshev polynomials. J. Comput. Appl. Math. 2007;205:281-295.

[12] El-Daou MK, Khajah HG, Iterated solutions of linear operator equations with the tau method. Math. Comput. 1997;66(217):207-213.

[13] El-Daou MK, Al-Matar NR. An improved Tau method for a class of Sturm–Liouville problems. Appl. Math. Comput. 2010;216:1923-1937.

[14] Greenberg L, Marletta M. Algorithm 775: the code SLEUTH for solving fourth-order Sturm-Liouville problems. ACM T. Math. Software. 1997;23(4):453-493.

[15] Greenberg L, Marletta M. Numerical methods for higher order Sturm–Liouville problems. J. Comput. Appl. Math. 2000;125(1–2):367–383.

[16] Gottlieb D, Orszag SA. Numerical Analysis of Spectral Methods: Theory and Applications. SIAM; 1977.

[17] Charalambides M, Waleffe F. Gegenbauer Tau Methods With and Without Spurious Eigenvalues. SIAM J. Numer. Anal. 2008;47:48-68.

[18] Weideman JAC, Reddy SC. A Matlab differentiation matrix suite. ACM T. Math. Software. 2000;26:465-519.

[19] Gardner DR, Trogdon SA, Douglass RW. A Modified Tau Spectral Method That Eliminates Spurious Eigenvalues. J. Comput. Phys. 1989;80:137-167.

[20] Trif D. LiScEig 1.0 2005. Accessed 6 June 2013.
Available: http://www.mathworks.com/matlabcentral/fileexchange/8689-lisceig-1-0

[21] Orszag SA. Accurate solution of the Orr-Sommerfeld stability equation. J . Fluid Mech. 1971;50(4):659-703.

[22] Trif D. The Lyapunov-Schmidt problem for two-point boundary value problems. Fixed Point Theory. 2005;6(1):119-132.

[23] Trif D. LiScNLS 1.0 2005. Accessed 6 June 2013.
Available: http://www.mathworks.com/matlabcentral/fileexchange/8715-liscnls-1-0

[24] Celledoni E, Cohen D, Owren B. Symmetric Exponential Integrators with an Application to the Cubic Schrödinger Equation. Found. Comput. Math. 2008;8:303-317.

[25] Trif D. LiScNLE 1.0 2006. Accessed 6 June 2013.
Available: http://www.mathworks.com/matlabcentral/fileexchange/11620-liscnle-1-0

[26] Heinrichs W. Spectral Approximation of Third-Order Problems. J. Sci. Comput. 1999;14(3):275-289.

[27] Pelloni B, Smith DA. Spectral theory of some non-selfadjoint linear differential operators. Proc. R. Soc. A. 2013; 469 20130019; doi:10.1098/rspa.2013.0019.

[28] Lutzen J. Sturm and Liouville's Work on Ordinary Linear Differential Equations. The Emergence of Sturm-Liouville Theory. Accessed 6 June 2013.
Available: http://www.maths.ed.ac.uk/ aar/papers/lutzen.pdf

[29] Zgliczynski P. Rigorous numerics for dissipative Partial Differential Equations II. Periodic orbit for the Kuramoto-Sivashinsky PDE - a computer assisted proof. Foundations of Comput. Math. 2004;4:157–185.

[30] Fornberg B. A pseudospectral fictitious point method for high order initial-boundary value problems. SIAM J. Sci. Comput. 2006;28(5):1716-1729.

[31] Mohebbi A, Asgari Z. Efficient numerical algorithms for the solution of "good" Boussinesq equation in water wave propagation. Comput. Phys. Commun. 2011;182:2464-2470.

[32] Brauer K. The Korteweg-de Vries Equation: History, exact Solutions, and graphycal Representation. Accessed 6 June 2013.
Available: http://www.usf.uni-osnabrueck.de/ kbrauer/solitons/KdV.pdf

[33] Dragomirescu FI, Gheorghiu CI. Analytical and numerical solutions to an electrohydrodynamic stability problem. Appl. Math. Comput. 2010;216:3718-3727.

[34] Gheorghiu CI, Dragomirescu FI. Spectral methods in linear stability. Applications to thermal convection with variable gravity field. Appl. Numer. Math. 2009;59:1290-1302.

©2013 Trif; This is an Open Access article distributed under the terms of the Creative Commons Attribution License http://creativecommons.org/licenses/by/3.0, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
www.sciencedomain.org/review-history.php?iid=240&id=6&aid=1919