MATSLISE 2.0: A Matlab Toolbox for Sturm-Liouville Computations

VEERLE LEDOUX and MARNIX VAN DAELE, Ghent University

The MATSLISE 2.0 software package is a thorough revision of the successful Matlab package MATSLISE of 2005. The package can be used to compute the eigenvalues and eigenfunctions of regular and some important classes of singular self-adjoint Sturm-Liouville boundary value problems. The code uses new or improved algorithms, offers some new features, and has an updated graphical user interface.

Categories and Subject Descriptors: G.4 [**Mathematical Software**]: Algorithm Design and Analysis; User Interfaces; G.1.7 [**Ordinary Differential Equations**]: Boundary Value Problems; J.2 [**Physical Sciences and Engineering**]: Mathematics and Statistics; Physics

General Terms: Algorithms, Design

Additional Key Words and Phrases: Sturm-Liouville problems, Schrödinger equations, eigenvalues, shooting

ACM Reference Format:
Veerle Ledoux and Marnix Van Daele. 2016. Matslise 2.0: A matlab toolbox for Sturm-Liouville computations. ACM Trans. Math. Softw. 42, 4, Article 29 (June 2016), 18 pages.
DOI: http://dx.doi.org/10.1145/2839299

1. INTRODUCTION

MATSLISE [Ledoux et al. 2005] is a graphical Matlab software package for the interactive numerical study of self-adjoint Sturm-Liouville problems (SLPs). It allows the fast and accurate computation of the eigenvalues and the visualization of the corresponding eigenfunctions by making use of the power of high-order piecewise constant perturbation (CP) methods. Since its release in 2005, the package has been downloaded hundreds of times. Many researchers, in particular the ones from applied fields, prefer to use the user-friendly problem-solving environment MATSLISE over Fortran subroutines, like SLEDGE [Pruess and Fulton 1993] and SLEIGN2 [Bailey et al. 2001], although these latter packages can deal with a wider range of problems.

The successor code MATSLISE 2.0 now includes the recent extension of the CP algorithm from problems in Liouville normal form to the general Sturm-Liouville form (see Ledoux and Van Daele [2010]). This extension and some specially adapted algorithms that are applied in a narrow interval around the singularity make MATSLISE 2.0 suitable for a broader class of singular problems. Other new additions to MATSLISE 2.0 include the enabling of piecewise continuous coefficient functions and the option to evaluate the eigenfunctions in a set of user-specified points, which facilitates further manipulation. The calculation of the eigenfunctions has also been improved in terms of numerical stability by rescaling the wave function variables in each CP step. The code has an updated user-friendly interface and a broad set of test problems, including the ones

Authors' addresses: Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Krijgslaan 281 - S9, B-9000 Ghent, Belgium; emails: {Ledoux.Veerle, Marnix.VanDaele}@gmail.com. V. Ledoux is a Postdoctoral Fellow of the Fund for Scientific Research–Flanders (Belgium) (F.W.O.-Vlaanderen).

DOI: http://dx.doi.org/10.1145/2839299

from Pryce [1999], has been predefined. The package no longer needs any additional Matlab toolboxes: first- and second-order derivatives of the coefficient functions, which are needed in the implementation of the Liouville's transformation, are now computed by automatic differentiation techniques, avoiding the need of the symbolic toolbox. The class structure of the package has been revised. There is now only one class storing all relevant data that determines an SLP and defining the operations on the SLP-object, that is, calculation of the eigenvalues and eigenfunctions. The additional classes for specific types of SLPs have been removed, as well as the classes implementing CP/LP algorithms of different order. MATSLISE 2.0, in contrast to its predecessor, has been implemented in the new class-definition syntax introduced with Matlab software version 7.6.

MATSLISE 2.0 is a Matlab code for the computation of the eigenvalues and eigenfunctions of regular and some important types of singular self-adjoint SLPs. SLPs are concerned with solutions of the linear, homogeneous, ordinary differential equation

$$-(py')' + qy = Ewy \quad x \in (a, b), \tag{1}$$

where (a, b) is an open interval of the real line; p, q, w are real-valued coefficient functions defined on (a, b); and E is a spectral parameter. To specify an SLP, two linearly independent, homogeneous boundary conditions are added to this differential equation (Equation (1)). The required number and form of these conditions depends on the interval (a, b) and the properties of the coefficient functions at the endpoints a and b. Additional information, including the classification of the endpoints a and b as regular, limit-point (LP), or limit-circle (LC), is given in subsequent sections and are discussed in more detail in Pryce [1993], Weidmann [1987], and Zettl [2005]. In the so-called regular case, MATSLISE 2.0 can handle boundary conditions that are separated,

$$A_0 y(a) + B_0 (py')(a) = 0, \tag{2}$$

$$A_1 y(b) + B_1 (py')(b) = 0, \tag{3}$$

where $A_0, A_1, B_0, B_1 \in \mathbb{R}$, and $A_0^2 + B_0^2 > 0$ and $A_1^2 + B_1^2 > 0$. The spectrum of such problems is then known to consist of only real values of the spectral parameter E; an eigen-solution to the problem is a pair (E, y) where E is the eigenvalue for which the differential equation has a non-null solution y, the eigenfunction, which satisfies the boundary conditions. For regular problems, the eigenvalue spectrum of the problem is discrete. In the more general case of singular problems, the spectrum may be more complicated and contains not only discrete eigenvalues but also an essential spectrum consisting of bands of continuous spectrum; see Pryce [1993] and Zettl [2005]. It is well known (see Atkinson [1964] and Pryce [1993]) that under the condition that p, q, w are defined on $[a, b]$, continuous except for finitely many jumps, with p, w strictly positive and the assumption that each endpoint is regular or limit-circle (see further), there exists an infinite number of eigenvalues. These are real, countable, and isolated, and each eigenfunction is unique up to constant multiples. If, in addition, the endpoints are in the nonoscillatory case (which automatically holds for regular endpoints), then the eigenvalues are bounded below. They can be indexed such that

$$-\infty < E_0 < E_1 < E_2 < \cdots < +\infty.$$

Furthermore, if y_k denotes an eigenfunction corresponding to E_k, then y_k has exactly $k \in \mathbb{N}_0$ zeros in the open interval (a, b), see Weidmann [1987]. This fact is important in the numerical determination of the eigenvalues in MATSLISE 2.0. In particular, it is used to enable the computation of a particular eigenvalue with a user-specified index. If one or both endpoints are oscillatory, then the eigenvalues are not bounded below, and each eigenfunction has infinitely many zeros in (a, b), see Weidmann [1987].

MATSLISE 2.0 can be used to compute the discrete spectrum of a wide range of singular self-adjoint Sturm-Liouville problems with positive p and w and separated boundary conditions. MATSLISE 2.0 can, however, not handle problems that have a discrete spectrum that is unbounded below, that is, the oscillatory-for-all E problems. MATSLISE 2.0 can only deal with separated boundary conditions, in contrast to the Fortran Sturm-Liouville solver SLEIGN2 [Bailey et al. 2001], which also allows coupled boundary conditions. Also in contrast to SLEIGN2, MATSLISE 2.0 does not offer the possibility of imposing arbitrary boundary conditions at a singular endpoint. In the case of a limit-circle non-oscillatory endpoint, the code selects a boundary condition, which is usually the Friedrichs (principal) condition [Bailey et al. 2001; Pryce 1993]. For problems with a continuous spectrum, MATSLISE 2.0 provides the starting point of the continuous spectrum, the number and numerical value of the eigenvalues below the continuous spectrum, but no information on the spectral density function or spectral bands and gaps is computed.

The MATSLISE 2.0 package can be downloaded from `http://sourceforge.net/projects/matslise/`. The package contains a number of routines that the user can run from the Matlab command line or that can be invoked from user-written scripts or functions. Given the interest of researchers from various fields, a user-friendly graphical user interface with built-in helpfiles is also provided, allowing us to enter the input in a straightforward manner to have easy access to the results and to present them graphically.

2. COMPUTATIONAL METHODS

2.1. The CP Propagation Algorithm

As the initial MATSLISE package [Ledoux et al. 2005], MATSLISE 2.0 uses CP algorithms to propagate the solution forwardly or backwardly over each mesh interval $[x_i, x_{i+1}]$. These so-called (reference potential) CP methods were originally devised for explicitly integrating (y, y') over the x-range of a Schrödinger problem, that is, an SLP in the Liouville normal form $(p = w = 1)$

$$y'' = (q(x) - E)y. \tag{4}$$

They use a perturbation technique to construct some correction terms that are added to the known solution of the approximating problem with a piecewise-constant potential q. In this way, methods up to order 16 were constructed (see Ixaru [2000] and Ledoux et al. [2004]) that can efficiently compute the eigenvalues of regular Schrödinger problems. The CP methods were extended to the more general SLP form (1) using the Liouville transformation [Ixaru et al. 1999; Ledoux et al. 2005]. This Liouville transformation converts an SLP into a Schrödinger problem (4) with the same eigenvalues. However, this transformation may be rather expensive near a singular endpoint due to the quadrature that is needed for the conversion between old and new variables. Moreover, the transformation can only be realized for sufficiently well-behaved (and non-singular) p, q, and w functions (see Pryce [1993]): q must be continuous and p and w should have a continuous second-order derivative. As a consequence, the original software package MATSLISE has a smaller range of applicability in comparison with, for example, the Fortran solver SLEDGE [Pruess and Fulton 1993], which applies a second-order coefficient approximation method directly to an SLP.

In Ledoux and Van Daele [2010], the extension of the CP algorithm to the general form of an SLP (1) was described. A sixth-order CP method was presented that can be applied directly to the SLP without the need for a Liouville transformation. This algorithm is now incorporated into MATSLISE 2.0.

For problems in Liouville normal form, it has been proven that the CP error improves as E increases, and this occurs despite the high oscillations, see Ixaru et al. [1997]. For problems not in Liouville normal form, however, the (absolute) error of the CP method behaves asymptotically (i.e., for large index) like $O(E)$, see Ledoux and Van Daele [2010]. Moreover, higher-order CP methods are available for Schrödinger problems: We choose to use the CP method of order 16 from Ledoux et al. [2004] in MATSLISE 2.0. For orders larger than 6, the complexity of the CP formulae for general SLP quickly increases. For these reasons, MATSLISE 2.0 still converts well-behaved regular SLPs to the Liouville normal form before a CP method is applied. This means that remarkably large stepsizes can then be taken, and one and the same E-independent mesh can be used to compute all eigenvalues required. The sixth-order CP algorithm from Ledoux and Van Daele [2010] is only used to apply a CP method directly to the SLP when a Liouville transformation may be difficult or expensive to realize, such as, for example, in the presence of jumps in the coefficient functions or a singular endpoint.

In the original MATSLISE package, the user had the option to use a so-called LP method instead of a CP method. Where a CP method adds perturbation correction terms to the known solution of an approximating problem with piecewise constant coefficient functions, the LP algorithms use a perturbation technique to construct correction terms for the solution of a problem with piecewise linear coefficient functions. In Ledoux and Van Daele [2011], however, it was proven that the LP algorithms are equivalent to their CP counterparts and that the LP approach does not induce any advantage over the CP technique. Since the inclusion of LP algorithms does not form a real asset, we only used CP algorithms in MATSLISE 2.0. Discharging the user from the choice between different methods has improved the user-friendliness of the package and reduced the complexity of the source code.

2.2. The Regular Eigenvalue Problem

Suppose the boundary value problems (1)–(3) are given. The eigenvalue problem consists of finding the value E_k of the parameter E and the corresponding solution y of Equation (1), such that both boundary conditions are satisfied. Basically, this requirement is accomplished by integrating Equation (1), with initial condition $y(a) = -B_0$, $p(a)y'(a) = A_0$, from the endpoint a to some interior point $c \in [a, b]$, thereby obtaining a solution y_L, py'_L, and then integrating Equation (1), with initial condition $y(b) = -B_1$, $p(b)y'(b) = A_1$, from endpoint b to c, thereby obtaining a solution y_R, py'_R. Both these integrations are made using an estimated value of E and using the CP algorithm for propagating the solution y, py' over the mesh. Theoretically, which interior point is chosen for c is immaterial, but, in practice, it is best to avoid particularly poor choices for c, see Ledoux [2007] and Pryce [1993]. The so-called miss-distance function

$$\phi(E) = py'_L(c, E)y_R(c, E) - py'_R(c, E)y_L(c, E)$$

is zero if and only if the used value E of the spectral parameter equals an eigenvalue. Thus, the procedure for finding the numerical value of the eigenvalue E_k consists in evaluating the function ϕ numerically and then, through a finite series of (Newton) iterations, finding the value of E such that $\phi(E) = 0$ to the required degree of approximation.

Since there may be an infinite number of eigenvalues, the code must be able to automatically select the one(s) the user requests. Because the core of the eigenvalue calculation is a zero-finding process, the code must be able to generate an interval that contains a prescribed eigenvalue and only this eigenvalue. For this purpose, we use a Prüfer-based procedure for calculating the number of zeros in the solution corresponding to a trial E value. As mentioned earlier, this number is related to the index of an eigenvalue. The same Prüfer procedure as in MATSLISE is used that was described in Ixaru et al. [1997, 1999] and now extended to work also for problems not in Liouville normal form.

It is critical to the shooting algorithm discussed above that approximate solution values are computed accurately. It is well known that when dealing with Schrödinger equations, care must be taken when integrating into classically forbidden regions, that is, regions where the potential q is larger than the energy E [Pryce 1993]. The same problem of numerical instability appears for general SLPs. Suppose p, q, w are approximated by the constants $\bar{p}, \bar{q}, \bar{w}$ in the interval $[x_i, x_{i+1}]$ and define $Z_i = (\bar{q} - E\bar{w})/\bar{p}$ and $\omega_i = \sqrt{|Z_i|}$. The transfer matrix T_i of the CP algorithm

$$\begin{pmatrix} y(x_{i+1}) \\ py'(x_{i+1}) \end{pmatrix} = T_i \begin{pmatrix} y(x_i) \\ py'(x_i) \end{pmatrix}$$

has then the following form (see Ledoux and Van Daele [2011]):

$$T_i = \bar{T}_i + \text{CP correction terms}, \quad h = x_{i+1} - x_i, \tag{5}$$

where

$$\bar{T}_i = \begin{pmatrix} \cos(\omega_i h) & \sin(\omega_i h)/(\bar{p}\omega_i) \\ -\bar{p}\omega_i \sin(\omega_i h) & \cos(\omega_i h) \end{pmatrix}, \quad h = x_{i+1} - x_i, \tag{6}$$

when $Z_i < 0$ and there is a similar expression involving cosh and sinh when $Z_i > 0$. The spectral radius of $\bar{T}_i$ in Equation (6) is $\sigma(\bar{T}_i) = 1$ in the case $Z_i \leq 0$ and $\sigma(\bar{T}_i) = \exp(\sqrt{Z_i}|h|)$ when $Z_i > 0$. The recursion can be unstable when many $Z_i > 0$ because of the exponential growth. For this reason, we choose in MATSLISE the matching point c to be in a stable region where $Z < 0$, that is, as the right endpoint of the interval where Z_i has its minimum value. For problems in Schrödinger form, this means that the matching point is near the minimum of the potential. The plots in Figure 1 show the value of the wave function y for the half-range harmonic oscillator problem over [0,20], that is, a Schrödinger problem with $q(x) = x^2$, for the eigenvalue $E = 11$. These plots illustrate that integrating from the $Z > 0$ region where the solution has an exponential behaviour into the oscillatory $Z < 0$ region is more stable than integrating from the $Z < 0$ region into the $Z > 0$ one. The left top plot was generated by using the CP algorithm to propagate the solution from $x = 0$ with initial values $y(0) = 0, y'(0) = 1$ to $x = 20$. When propagating too far into the classically forbidden region, the solution starts to blow up and we do not obtain the correct value in $x = 20$. When CP propagating the solution from $x = 20$ with $y(20) = 0, y'(20) = 1$ to $x = 0$, on the other hand, as in the right top plot, the boundary condition $y(0) = 0$ is met, as one expects for an eigenvalue.

In MATSLISE 2.0, the procedure is further improved by dividing T_i by $\sigma_i = \sigma(\bar{T}_i)$ at each step. This means that we work with the scaled variables $z_i = y_i/(\sigma_0 ... \sigma_{i-1}), z_i' = py_i'/(\sigma_0 ... \sigma_{i-1})$ instead of the unscaled y_i, py_i'. This prevents overflow due to exponential growth and makes the approach less sensitive to the choice of the matching point. The bottom plots in Figure 1 show the forward and backward propagation from one endpoint to the other, using the scaled algorithm. In both cases, the boundary conditions are satisfied.

2.3. Mesh Construction and Error Estimation

MATSLISE 2.0 uses a fixed mesh throughout the computation of each eigenvalue approximation that is consistent with a user input tolerance *tol*. The meshing algorithm works by controlling the local error in the CP transfer matrix. This places a higher concentration of mesh points in regions with a large variation in the coefficient functions and

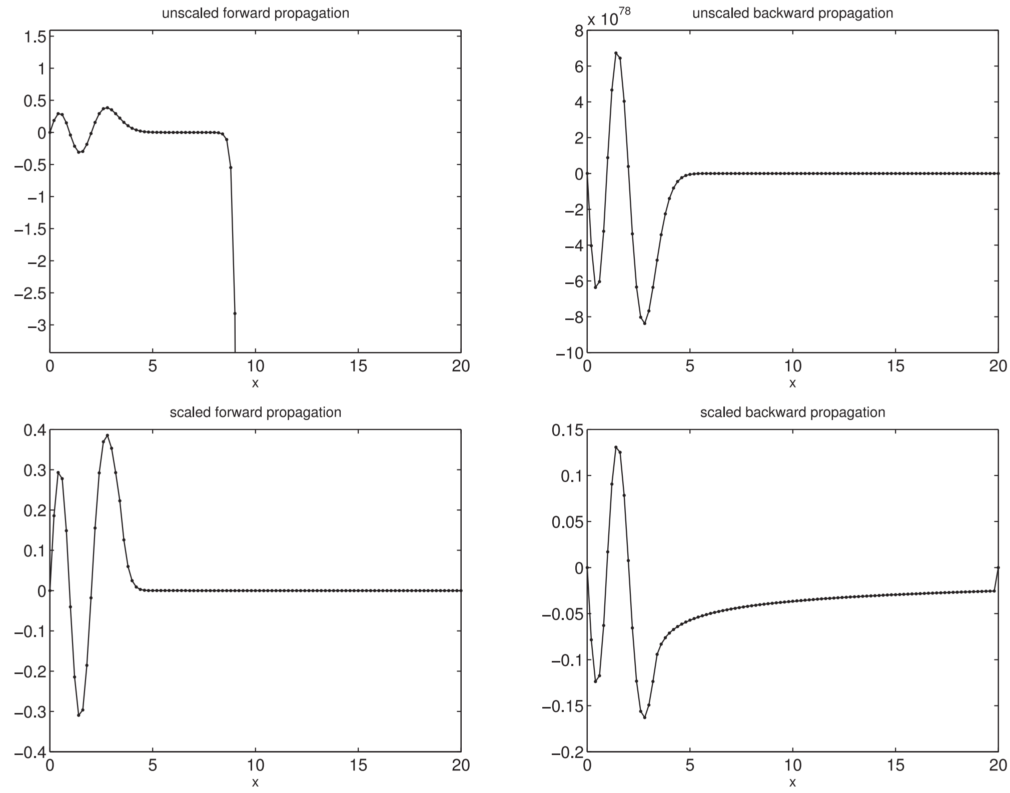


Fig. 1. CP propagation of the wave function corresponding to the eigenvalue $E = 11$ for the half-range harmonic oscillator problem.

selects larger steps elsewhere. This requires an estimation for the local error. For the CP method CPM{16,14} of order 16, which is used in the case of problems in Liouville normal form (this form may have been obtained after a Liouville's transformation), the same meshing algorithm is used as in Matslise, also described in Ledoux [2007]. All disregarded contributions that appear in the CPM{18,16} algorithm but not in the CPM{16,14} formulae are used to estimate the error. The Matslise 2.0 package also delivers an estimation of the error in the eigenvalue approximations. In the case of problems in Liouville normal form, Matslise 2.0 measures this error as the difference between the calculated eigenvalue (the so-called basic eigenvalue) computed with the CPM{16,14} method and a reference eigenvalue computed on the same mesh but with the higher-order method CPM{18,16}. The Newton iteration process in the shooting procedure for the reference eigenvalue always starts with the basic eigenvalue. Since the difference of the two eigenvalues is usually small, only a small number of iterations is necessary, and thus the calculation of the reference eigenvalue requires only a small extra effort. Moreover, meshing is done once and for all eigenvalue computations, which means that all evaluations of the SLP coefficient functions are performed during the construction of the mesh and are saved for later reuse, which makes the integration during shooting fast.

For the sixth-order CP method for problems in full Sturm-Liouville form that are not transformed into the Schrödinger form, an embedded method of order four is used for the purpose of error estimation and adaptive stepsize selection.

2.4. Computing Eigenfunctions

First, we address the question how to accumulate the integral for the norm, that is, $\int_{x_i}^{x_{i+1}} y^2(x)w(x)\mathrm{d}x$ step by step. As described for the Schrödinger equation in Ixaru et al. [1997], we use the derivative with respect to E to approximate this integral. The derivative of the Sturm-Liouville equation with respect to E is

$$-(py')'_E + qy_E = wy + Ewy_E. \tag{7}$$

Subtracting the multiplication of the Sturm-Liouville equation by y_E,

$$-(py')'y_E + qyy_E = Ewyy_E,$$

from the multiplication of Equation (7) by y leads to

$$-(py')'_E y + (py')'y_E = wy^2.$$

Integrating over the step $[x_i, x_{i+1}]$ gives us

$$\int_{x_i}^{x_{i+1}} y^2 w dx = [-(py')_E y + (py')y_E]_{x_i}^{x_{i+1}}. \tag{8}$$

Note that the CP algorithm can easily give us the values of the derivatives of y, py' with respect to E in the endpoints of each mesh interval, see Ixaru et al. [1997] and Ledoux [2007].

While computing the eigenfunction, however, we again avoid overflow by scaling at each step by the reciprocal of the spectral radius of the transfer matrix, that is, we work with scaled variables z and their derivatives instead of the unscaled y. The eigenfunction computation in MATSLISE 2.0 has two stages. In the first stage, the left scaled solution z^L, z'^L and z^L_E, z'^L_E are advanced on each mesh interval between the endpoint $a = x_0$ and the matching point $c = x_m$, and, similarly, the right solution z^R, z'^R and z^R_E, z'^R_E is propagated from $b = x_n$ to $c = x_m$. These recursions are started with the initial values

$$z^L_0 = y^L_0 = -B_0/s_1,\ z'^L_0 = (py')^L_0 = A_0/s_1, \quad s_1 = \sqrt{A_0^2 + B_0^2}, \tag{9}$$

$$z^R_n = y^R_n = -B_1/s_2,\ z'^R_n = (py')^R_n = A_1/s_2, \quad s_2 = \sqrt{A_1^2 + B_1^2}, \tag{10}$$

$$\left(z^L_E\right)_0 = \left(z^R_E\right)_n = \left(z'^L_E\right)_0 = \left(z'^R_E\right)_n = 0. \tag{11}$$

In c, we have then $z^L_m = y^L_m/\sigma^L$, $z'^L_m = (py')^L_m/\sigma^L$, $(z^L_E)_m = (y^L_E)_m/\sigma^L$, $(z'^L_E)_m = ((py')^L_E)_m/\sigma^L$ with $\sigma^L = \sigma_0 \dots \sigma_{m-1}$ and $z^R_m = y^R_m/$, $z'^R_m = (py')^R_m/\sigma^R$, $(z^R_E)_m = (y^R_E)_m/\sigma^R$, $(z'^R_E)_m = ((py')^R_E)_m/\sigma^R$ with $\sigma^R = \sigma_m \dots \sigma_{n-1}$. Based on formula (8), we use these values to compute

$$Q_L = -\left(z'^L_E\right)_m z^L_m + \left(z^L_E\right)_m z'^L_m, \tag{12}$$

$$Q_R = \left(z'^R_E\right)_m z^R_m - \left(z^R_E\right)_m z'^R_m. \tag{13}$$

With

$$F = Q_L + Q_R\left(z^L_m/z^R_m\right)^2$$

we start the propagation of the left and right solutions in the second stage with the initial conditions

$$\hat{z}^L_0 = z^L_0/\sqrt{F},\ \hat{z}'^L_0 = z'^L_0/\sqrt{F}, \tag{14}$$

$$\hat{z}^R_n = z^R_n\left(z^L_m/z^R_m\right)/\sqrt{F},\ \hat{z}'^R_n = z'^R_n\left(z^L_m/z^R_m\right)/\sqrt{F}, \tag{15}$$

which ensure smooth solutions in the matching point c when $\hat{z}^L, \hat{z}'^L$ is advanced from $a = x_0$ to $c = x_m$ and $\hat{z}^R, \hat{z}'^R$ from $b = x_n$ to $c = x_m$. The evaluation of the normalized solution $\hat{y}_i$ (eigenfunction) in each meshpoint x_i is then finally given by

$$\hat{y}_i = \begin{cases} \hat{z}_i^L/(\sigma_i \ldots \sigma_{m-1}) & i < m \\ \hat{z}_m^L & i = m\,, \\ \hat{z}_i^R/(\sigma_m \ldots \sigma_{i-1}) & i > m \end{cases} \tag{16}$$

where, during the propagation of $\hat{z}$, the transfer matrix was again divided at each step by the reciprocal of the spectral radius. Similar expressions exist for the approximate derivative $(p\hat{y}')_i$.

This algorithm gives us the normalized eigenfunction in the (relatively small number of) mesh points of the mesh automatically constructed by MATSLISE 2.0. However, there are many possible uses for eigenfunctions of SLPs and having the eigenfunction only available in some specific points might not be sufficient for use in subsequent calculations. For this reason, the user can now specify his/her own vector of x_i-values in which the eigenfunction needs to be evaluated. Internally, MATSLISE 2.0 generates a new mesh by adding the original mesh points to the user-supplied vector of x-values. Adding the original mesh points assures sufficient accuracy when the distance between the values in the x-vector is too large. The eigenfunction over the new mesh is then computed as described above. Only the evaluations of $\hat{y}$, $\hat{p}y'$ in the user-requested x-values are returned to the user.

2.5. Half-Range Reduction

When an SLP is symmetric, it is useful to apply half-range reduction. Half-range reduction is now automatically applied in MATSLISE 2.0. Solving the problem on the half-range not only is cheaper and faster but also makes the problem more tractable. For symmetric double-well problems, for instance, this reduction may make the difference between a highly ill-conditioned problem and a straightforward one. A problem is symmetric when the endpoints of the integration interval satisfy $a = -b$, where b may be ∞, the coefficient functions are even, and the boundary conditions are symmetric, which, in the regular case, means that $A_0 = A_1$, $B_0 = -B_1$.

The eigenvalues are obtained by solving the given equation, but on the interval $[0, b]$, with the given boundary condition at b and with $y'(0) = 0$ to get the eigenvalues with even index, and $y(0) = 0$ to get the eigenvalues with odd index. The normalized eigenfunctions of the full-range problem are reconstructed from those of the half-range problem by extending in the appropriate way and dividing by $\sqrt{2}$.

2.6. Piecewise Continuous Coefficient Functions

MATSLISE 2.0 no longer requires that the coefficient functions p, q, w are continuous over the integration interval. A finite number of finite jump discontinuities is handled automatically and efficiently by including the location of the jumps as mesh points. The user should therefore pass a vector with the jump locations to the MATSLISE 2.0 function constructing the mesh (see further). Some example M-files included into the `examples/discontinousProblems` directory solve test problems with discontinuities.

3. SINGULAR PROBLEMS

MATSLISE 2.0 can handle some important types of singular SLPs. A singular problem occurs when at least one of the coefficients $1/p$, w, q is not integrable up to the endpoint or if one or both of these endpoints is infinite. These present some specific difficulties both in the determination of well-posed problems and in the numerical solution. We give a brief description of the technical issues in this section. For the broad theory on singular SLPs, we refer to Pryce [1993], Dunford and Schwartz [1988], and Zettl [2005]. In these references, more information can also be found about some important properties of a singular endpoint, such as the so-called Weyl-Kodaira limit-point/limit-circle classification, which is independent of E, and whether it is oscillatory or nonoscillatory, which may depend on E.

Both problems defined on an infinite integration interval and problems with singular endpoints require a special numerical treatment. In these cases, an interval truncation procedure is adopted. This approach used by MATSLISE 2.0 works for singularities of limit-point and of limit-circle Friedrichs BC type [Pryce 1993]. In the limit-point case, MATSLISE 2.0 can compute all real E outside the continuous spectrum. The algorithm on which MATSLISE 2.0 is based depends on knowing that, when the eigenvalues are enumerated appropriately, the kth eigenfunction has precisely k zeros in the interval (a, b). This is the reason why MATSLISE 2.0 currently does not deal with oscillatory endpoints: In this case every eigenfunction has an infinite number of zeros in any neighbourhood of the endpoint.

The automatic classification algorithm by Pruess et al. [1999] is incorporated into MATSLISE 2.0. This algorithm, which was also implemented in the SLEDGE package [Pruess and Fulton 1993], automatically classifies the nature of the problem, regular or singular, limit-circle singularity or limit-point singularity and so on. This classification information is important to determine whether there is a continuous spectrum, when there are discrete eigenvalues and how many, and what boundary condition should be imposed at a singular endpoint. By integrating this algorithm into MATSLISE 2.0, MATSLISE 2.0 does not require the user to distinguish between regular and singular eindpoints or between limit point or limit circle singularities and automatically informs the user about the form of the spectrum. Also, an appropriate message is returned to the user when a problem was entered that cannot be solved by MATSLISE 2.0.

3.1. General Treatment of a Singular Endpoint

The CP methods preserve the interesting advantage of the second-order SLEDGE method [Pryce 1993; Pruess and Fulton 1993] in that they allow a very simple interval truncation algorithm for singular problems. A CP method evaluates the coefficients only in the Legendre points of each mesh interval (see Ixaru et al. [1997] and Ledoux et al. [2004]), which effectively regularizes the problem and can be regarded as truncating the integration interval at the first and last Legendre node of the initial and final intervals, respectively. Every time a mesh interval is refined near the endpoint, these implicit truncation points move closer to the singular endpoint.

This idea was used to introduce in MATSLISE 2.0 a general method that can be applied to a wide variety of singular endpoint behaviour. The MATSLISE 2.0 algorithm selects an initial point $b^* < b$ (in the assumption that b is the singular finite endpoint, an analog procedure is followed when a is singular). The mesh selection algorithm is then applied to construct a mesh over the interval $[a, b^*]$. To this automatically constructed mesh, one mesh interval is appended: $[b^*, b]$. The CP method evaluates the coefficients only in the Legendre points of each mesh interval (and, consequently, not in the endpoint b). A first eigenvalue approximation is computed over the mesh. The final mesh interval $[b^*, b]$ is then refined by defining a new b^* value: $b^* + (b - b^*)/2$, and a new eigenvalue approximation is computed. This process is repeated until two successive eigenvalue approximations agree within the requested accuracy. In each iteration, the shooting algorithm for the next eigenvalue approximation is started using the previous approximation, so the process gets faster as b^* comes closer to b.

In the case of a problem defined on an infinite integration interval, suppose $b = \infty$, the following procedure is applied. First, a cutoff value $\hat{b}$ is selected. The selection procedure for $\hat{b}$ is based on the Wentzel-Kramers-Brillouin approximation and was described for

Schrödinger problems in Ledoux et al. [2006] and Ledoux [2007]. The value returned for $\hat{b}$ depends on the requested eigenvalue: The value of $\hat{b}$ grows with the eigenvalue. The procedure was slightly adopted for the small group of SLPs not in Schrödinger form and not suitable for the Liouville's transformation. A mesh is constructed on $[a, \hat{b}]$ and a first eigenvalue approximation is computed. Next, the value of $\hat{b}$ is enlarged and the mesh extended. A further eigenvalue approximation is obtained. Also here, the process is repeated until two successive eigenvalue approximations agree sufficiently.

3.2. Solution Near the Origin for Radial Schrödinger Equations

An important class of Schrödinger equations covering a wide variety of physical problems has a potential function of the form

$$q(x) = \frac{l(l+1)}{x^2} + V(x). \tag{17}$$

For such radial Schrödinger problems defined over the interval $(0, +\infty)$, we apply an adapted perturbative procedure in a region $(0, \epsilon]$ around the origin that takes into account the specific singular nature of the potential. The algorithm gives us a good approximation for the value of the solution in $\epsilon \neq 0$. The solution and its derivatives in ϵ then form the starting values for the integration (using CP) on the interval $[\epsilon, +\infty)$. By using a specific perturbative procedure over $(0, \epsilon]$, we avoid the repeated refinement of the mesh near the origin and the computation of an eigenvalue approximation over each of these refined meshes in the approach discussed in the previous section. In this way, we managed to deal with the singularity in a very efficient and robust way for many problems for which there is a real need for efficient numerical solution techniques in physical applications.

The perturbative procedure discussed in this section is applied by MATSLISE 2.0 on problems in the Schrödinger form defined on $(0, +\infty)$ and with a singularity in the origin and for which $x^2q(x)$ is not singular in the origin. For problems not in the Schrödinger (Liouville normal) form, or with a different type of singularity, the procedure discussed in Section 3.1 is applied.

Whereas a CP method replaces the potential function $q(x)$ over each mesh interval by a polynomial, we now approximate $x^2q(x)$ by a polynomial. We interpolate $x^2q(x)$ through the five Legendre nodes in the interval $(0, \epsilon]$ to obtain a function of the form $\bar{q}(x) = V_{-2}/x^2 + V_{-1}/x + V_0 + V_1x + V_2x^2$ as approximation for $q(x)$ over $(0, \epsilon]$. The $V_{-2}, V_{-1}, V_0, V_1, V_2$ variables are E-independent constant values. Note that ϵ is small enough such that the $1/x^2$ term is numerically dominating with respect to the other terms of the potential and the fitting of q by $\bar{q}$ is sufficiently accurate. With the operator $L = \frac{d^2}{dx^2} - \frac{V_{-2}}{x^2}$ and $\Delta V(x) = V_{-1}/x + V_0 + V_1x + V_2x^2 - E$, we can write the Schrödinger equation as

$$Ly(x) = \Delta V(x)y(x) \tag{18}$$

with the dominant term in x^{-2} in the left-hand side. We introduce a parameter $\lambda \in [0, 1]$ and consider the equation

$$L\bar{y}(x; \lambda) = \lambda\Delta V(x)\bar{y}(x; \lambda). \tag{19}$$

When expanding $\bar{y}(x; \lambda)$ in powers of λ,

$$\bar{y}(x; \lambda) = y_0(x) + \lambda y_1(x) + \lambda^2 y_2(x) + \dots, \tag{20}$$

the coefficients $y_r(x)$ are found to satisfy the recurrence relations

$$Ly_0(x) = 0, \quad Ly_{r+1}(x) = \Delta V(x)y_r(x), \quad r = 0, 1, \dots. \tag{21}$$

For $\lambda = 1$, the expansion (20) gives us the solution of Equation (18). For the zeroth-order solution, we take

$$y_0(x) = x^{l+1}, \tag{22}$$

with

$$l = \big(-1 + \sqrt{1 + 4V_{-2}}\big)/2, \tag{23}$$

since x^{l+1} is the regular solution of the equation $Ly_0(x) = 0$. Note that we suppose that V_{-2} is larger than $-1/4$; if this is not the case, then the general procedure for a singular endpoint is applied. The first perturbation $y_1(x)$ is obtained using the recurrence relation in Equation (21),

$$Ly_1(x) = \Delta V(x) x^{l+1}, \tag{24}$$

which can also be written as

$$Ly_1(x) = \sum_{p=1}^{4} A_1^p x^{l+p-1}, \tag{25}$$

where $A_1^1 = V_{-1}$, $A_1^2 = \bar{V}_0 = V_0 - E$, $A_1^3 = V_1$, and $A_1^4 = V_2$. $y_1(x)$ is then of the form $y_1(x) = \sum_{p=1}^{4} z_p(x)$, where $z_p(x)$ is the solution of

$$Lz_p(x) = A_1^p x^{l+p-1}. \tag{26}$$

Suppose z_p is of the following form:

$$z_p(x) = B_1^p x^k, \tag{27}$$

and then Equation (26) gives

$$[k(k-1) - l(l+1)] B_1^p x^{k-2} = A_1^p x^{l+p-1}. \tag{28}$$

By identification, it results that $k = l + p + 1$ and

$$B_1^p = \frac{A_1^p}{p(1 + p + 2l)}. \tag{29}$$

The first-order solution y_1 is then introduced in the right-hand side of Equation (21) to compute y_2 and so on. Each rth perturbation is obtained from an equation of the form

$$Ly_r(x) = \sum_{p=r}^{4r} A_r^p x^{l+p-1} \tag{30}$$

with

$$A_r^p = V_{-1}\bar{B}_{r-1}^{p-1} + \bar{V}_0 \bar{B}_{r-1}^{p-2} + V_1 \bar{B}_{r-1}^{p-3} + V_2 \bar{B}_{r-1}^{p-4}, \quad p = r, \ldots, 4r$$

where

$$\bar{B}_{r-1}^{j} = \begin{cases} B_{r-1}^{j} & \text{if } r-1 \le j \le 4(r-1) \\ 0 & \text{otherwise.} \end{cases}$$

We can write the perturbation as $y_r(x) = \sum_{p=r}^{4r} z_p(x)$, where each $z_p(x)$ is of the form

$$z_p(x) = B_r^p x^{l+p+1}, \tag{31}$$

with

$$B_r^p = \frac{A_r^p}{p(1+p+2l)}. \tag{32}$$

The calculation of the perturbations leads to values that decrease in magnitude with r. We add (iteratively) as many perturbation corrections y_r as necessary to reach a certain preset accuracy.

4. USING MATSLISE 2.0

MATSLISE 2.0 has received a much clearer and updated class structure, which makes the package more straightforward to use. The user no longer has to make the distinction between regular Schrödinger-type problems, Coulomb-type Schrödinger problems, or problems in full SLP form. The software automatically makes the decision on when to apply a Liouville transformation, half-range reduction, an interval truncation algorithm, and so on. The user does not have to specify which CP algorithm to use. In this section, we describe how one can enter an SLP into MATSLISE 2.0 and how the package can be used to compute eigenvalues and eigenfunctions.

4.1. The MATSLISE 2.0 Functions

The MATSLISE 2.0 package contains a class called `slp`, storing all relevant data that define a Sturm-Liouville problem (e.g., coefficient functions, x-domain) and also defines the operations the user can perform on the slp-object (computation of eigenvalues or eigenfunctions). An instance `slpObject` of the `slp` class is created by a call to the constructor

```
slpObject = slp(@p,@q,@w,a,b);
```

When boundary conditions of the form (3) need to be specified, use the constructor

```
slpObject = slp(@p,@q,@w,a,b,A0,B0,A1,B1);
```

MATSLISE 2.0 can also deal with coefficient functions that are continuous except for finitely many jumps. The location of the jumps has to be passed as a vector to the constructor:

```
slpObject = slp(@p,@q,@w,a,b,A0,B0,A1,B1,jumps);
```

The first arguments to the constructor, that is, `@p,@q,@w` are function handles to MATLAB functions implementing the coefficient functions p, q, and w.

The class `slp` contains two methods that a user can apply on an object of the class. The first method computes the eigenvalues of the SLP by calling the method `computeEigenvalues`:

```
[E,mesh_data] = slpObject.computeEigenvalues(pmin,pmax,tol,indices)
```

The values `pmin` and `pmax` and the Boolean `indices` determine which eigenvalues need to be approximated. When `indices` is true, `pmin` and `pmax` represent the lower index and the upper index of the range of eigenvalues searched for. When `indices` is false, `pmin` and `pmax` represent the lower and upper limit of the energy range to be scanned for eigenvalues. `tol` is a positive constant representing the accuracy requested in the results. The method returns a structure `E` containing the eigenvalue results. `E.eigenvalues` is a vector with the computed eigenvalues in the ascending order. The vector `E.indices` contains the indices of the computed eigenvalues, while the vector `E.error` collects the estimations of the errors in the computed eigenvalues. Of course, the MATSLISE 2.0 algorithm, as any other numerical method, cannot be expected to work successfully on all problems. Some problems are hard for all available SLP software.

We have incorporated a number of checks that are performed during computation and a set of status values that are returned that indicate whether the problem has been solved satisfactorily. `E.success` is a Boolean value that is false when the CP-method was not able to obtain the data due to an error. `E.msg` contains the error message. `E.status` is a vector of status flags. `E.status(i)=0` indicates that no difficulties were detected during the calculation of `E.eigenvalues(i)`. If `E.status(i)>0`, then difficulties were detected and the result in `E.eigenvalues(i)` may be inaccurate or wrong. A call to `computeEigenvalues` automatically starts the construction of a mesh consistent with `tol`. Information (stepsizes, truncated singular endpoints, etc.) on this mesh can be derived from the optional output argument `mesh_data`; we refer to the MATSLISE 2.0 helpfiles for details.

This `mesh_data` structure can be passed to the class method `computeEigenfunction`, which computes the normalized eigenfunction corresponding to the eigenvalue `e`:

```
[x,y,yp] = slpObject.computeEigenfunction(e,mesh_data)
```

`e` is an eigenvalue approximation that was computed over the mesh in `mesh_data`. The output vector `x` contains the mesh points. The vectors `y` and `yp` contain the evaluations of the normalized eigenfunction $y(x)$ and the derivative $p(x)y'(x)$ in these mesh points.

```
[x,y,yp] = slpObject.computeEigenfunction(e,mesh_data,evalPoints)
```

evaluates the (normalized) eigenfunction corresponding to the eigenvalue `e` in the vector of `x` values in `evalPoints`. This can, for instance, be used to produce a more smooth graphical representation of the eigenfunction or it allows us to deliver the eigenfunction to the user so he can use it for subsequent calculations.

4.2. The MATSLISE 2.0 GUI

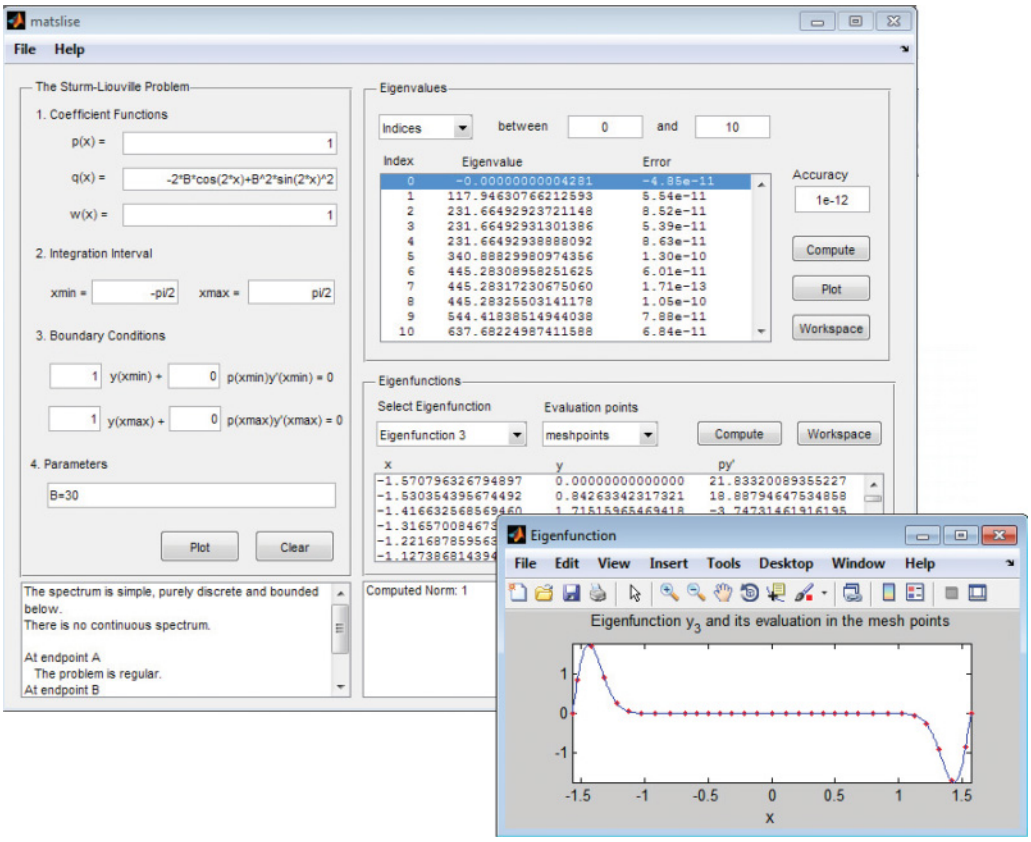
A user of the MATSLISE 2.0 package can call the `slp` class constructor and the class methods from the Matlab command line or from his own Matlab script or function files. As in the original MATSLISE package, another option is to use the Graphical User Interface (GUI), which is built on top of the package and which offers some extra visualization features. The GUI has been updated and has received a new, more intuitive look with the intention to further improve the user-friendliness. The GUI is shown in Figure 2. The GUI now consists of only one figure window in which the problem is defined and the eigenvalues, as well as the corresponding eigenfunctions, can be computed and visualized. The use of the GUI is straightforward and we refer to the Help-menu of the MATSLISE 2.0 GUI for details on the different GUI components. The GUI is opened by typing the command `matslise` at the Matlab command line.

5. SOME TEST EXAMPLES AND RESULTS

The package contains a directory `examples` with a wide range of predefined problems. The files with extension `.m` are Matlab M-files that illustrate the use of the different MATSLISE 2.0 functions discussed in Section 4.1 while the `.mat` files can be loaded into the MATSLISE 2.0 GUI. The set of test problems is based on the list of test problems provided by John D. Pryce [Pryce 1993] and the catalogue of Sturm-Liouville problems composed by Everitt [2005]. The numerical results for these test problems illustrate the capability of MATSLISE 2.0 to handle a variety of problems.

As a first example of the use of MATSLISE 2.0, we choose a difficult example from Pryce [1993], the Coffey-Evans equation. The triple well of the Coffey-Evans potential produces triplets of eigenvalues that can be made arbitrarily close by deepening the well. Here

$$-y'' + (-2\beta \cos 2x + \beta^2 \sin^2 2x)y = Ey \qquad (33)$$



The Sturm-Liouville Problem

1. Coefficient Functions

p(x) = 1

q(x) = -2*B*cos(2*x)+B^2*sin(2*x)^2

w(x) = 1

2. Integration Interval

xmin = -pi/2 xmax = pi/2

3. Boundary Conditions

1 y(xmin) + 0 p(xmin)y'(xmin) = 0

1 y(xmax) + 0 p(xmax)y'(xmax) = 0

4. Parameters

B=30

```
The spectrum is simple, purely discrete and bounded
below.
There is no continuous spectrum.

At endpoint A
  The problem is regular.
At endpoint B
```

Eigenvalues

Indices between 0 and 10

Accuracy 1e-12

Index	Eigenvalue	Error
0	-0.00000000004281	-4.85e-11
1	117.94630766212593	5.54e-11
2	231.66492923721148	8.52e-11
3	231.66492931301386	5.39e-11
4	231.66492938888092	8.63e-11
5	340.88829980974356	1.30e-10
6	445.28308958251625	6.01e-11
7	445.28317230675060	1.71e-13
8	445.28325503141178	1.05e-10
9	544.41838514944038	7.88e-11
10	637.68224987411588	6.84e-11

Eigenfunctions

Select Eigenfunction: Eigenfunction 3

Evaluation points: meshpoints

x	y	py'
-1.570796326794897	0.00000000000000	21.83320089355227
-1.530354395674492	0.84263342317321	18.88794647534858
-1.416632568569460	1.71515965469418	-3.74731461916195
-1.316570084673		
-1.221687859563		
-1.127386814394		

Computed Norm: 1

Fig. 2. Matslise 2.0 GUI.

with

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

The parameter β controls the depth of the well, and we have used a value of $\beta = 30$, which may cause difficulties to SLP software. The MATSLISE 2.0 package generates the first 11 eigenvalues in 1s of time on an ordinary computer by executing the following commands:

```
slpObject = slp(@p,@q,@w,-pi/2,pi/2,1,0,1,0);
[E, meshData] = computeEigenvalues(slpObject,0,10,1e-12,true);

function r=p(x)
  r=1;
end

function r=q(x)
  r = −2 ∗ 30 ∗ cos(2 ∗ x) + 30ˆ2 ∗ sin(2*x)ˆ2;
end

function r=w(x)
  r=1;
end
```

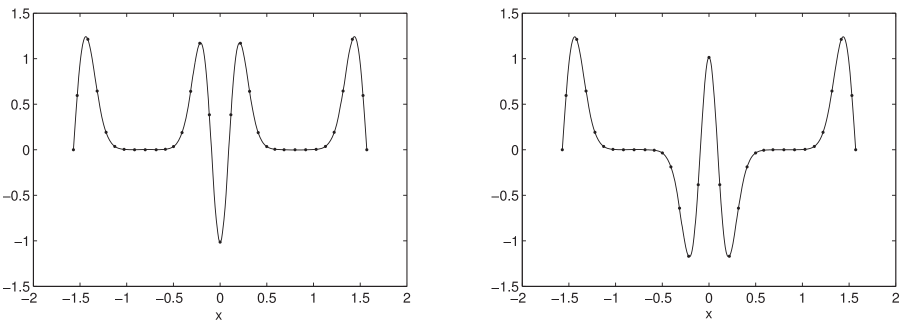


Fig. 3. Eigenfunctions y_2 and y_4 of the Coffey-Evans problem returned by MATSLISE 2.0; the dots represent the evaluation of the eigenfunction in the meshpoints.

The structure E then contains information on the index, value, and error estimation of each estimated eigenvalue. The generated eigenvalue approximations are equal to the ones visible in Figure 2.

The following lines of code illustrate the use of the MATSLISE 2.0 function `computeEigenfunction` to generate the eigenfunction plots from Figure 3:

```
e=E.eigenvalues(3);
finerGrid=linspace(-pi/2,pi/2,2^9);
figure
[x1,y1,yp] = o.computeEigenfunction(e,meshData);
[x2,y2,yp] = o.computeEigenfunction(e,meshData,finerGrid);
plot(x1,y1,'.',x2,y2)
e=E.eigenvalues(5);
figure
[x1,y1,yp] = o.computeEigenfunction(e,meshData);
[x2,y2,yp] = o.computeEigenfunction(e,meshData,finerGrid);
plot(x1,y1,'.',x2,y2)
```

Next, we take the six example problems that were used in the "comparative examples" section of Bailey et al. [2001] to compare results from SLEIGN2 with SLEDGE. For each test problem, we show in Tables I-VI results obtained in MATSLISE and in MATSLISE 2.0, as well as the results for SLEDGE and SLEIGN2, see Bailey et al. [2001]. Note that Bailey et al. [2001] also contains results for three additional examples that are not included here. These problems with limit-circle non-oscillatory endpoints and coupled boundary conditions can only be processed by the code SLEIGN2.

(1) The hydrogen atom equation:

$$-y'' + \left(\frac{2}{x^2} - \frac{1}{x}\right) y = Ey, \quad \text{for all } x \in (0, +\infty) \tag{34}$$

with eigenvalues $E_k = \frac{-1}{4(k+2)^2}$, $k = 0, 1, \ldots$. A suitable value for the right-hand truncation point $\hat{b}$ strongly depends on k. This differential equation is limit-point at 0 and $+\infty$.

(2) A weakly regular differential equation:

$$-(x^{1/2}y'(x))' = Ex^{-1/2}y(x) \ \text{ for all } x \in (0, 1]. \tag{35}$$

With boundary conditions $y(0) = 0 = y(1)$, the exact eigenvalues are $E_k = ((k + 1)\pi)^2/4$, $k = 0, 1, \ldots$ and with boundary conditions $py'(0) = 0 = y(1)$, the exact

Table I. Hydrogen Atom Equation

k	SLEIGN2	SLEDGE	MATSLISE/MATSLISE 2.0
0	−0.062499996	−0.062500000000	−0.062500000000
1	−0.027777772	−0.027777777778	−0.027777777778
2	−0.015624997	−0.015625000000	−0.015625000000
10^2	FAIL	−0.00002402921953	−0.00002402921953
10^3		−0.0000024900299	−0.0000024900299

Table II. A Weakly Regular Differential Equation

k	SLEIGN2	SLEDGE	MATSLISE	MATSLISE 2.0
0	2.467424	2.467401	FAIL	2.467401
1	9.69696	9.869605		9.869605
2	22.206815	22.206610		22.206610
0	0.616856	0.616850	FAIL	0.616850
1	5.551704	5.551653		5.551653
2	15.421399	15.421257		15.421257

Table III. The Legendre Equation

k	SLEIGN2	SLEDGE	MATSLISE	MATSLISE 2.0
0	0.250000	0.250000000000	FAIL	0.250000000000
1	2.250000	2.249999999984		2.250000000000
2	6.250000	6.249999999766		6.250000000000

eigenvalues are $E_k = ((2k+1)\pi/4)^2$. For these boundary value problems, with $p(0) = 0$ and $w(0) = +\infty$, the codes SLEIGN [Bailey et al. 1978] and MATSLISE cannot find the eigenvalues. The implementation of the Liouville's transformation in MATSLISE, see Ledoux [2007], requires the functions $1/p$ and w to have finite values at the endpoints.

(3) The Legendre equation:

$$-((1-x^2)y'(x))' + \frac{1}{4}y(x) = Ey(x), \quad \text{for all } x \in (-1, 1) \tag{36}$$

with eigenvalues $E_k = k(k+1) + 1/4, k = 0, 1, \ldots$. Again, the algorithm for the Liouville's transformation fails in MATSLISE due to the singular endpoints. This problem is limit-circle non-oscillatory at -1 and $+1$. MATSLISE 2.0 automatically chooses the Friedrichs boundary condition at both ± 1, that is, $y(\pm 1) = 0$.

(4) The Morse equation:

$$-y'' + (9e^{-2x} - 18e^{-x})y = Ey, \quad \text{for all } x \in (-\infty, \infty) \tag{37}$$

This differential equation is limit-point at both endpoints. The spectrum has exactly three negative, simple eigenvalues and a continuous spectrum on $[0, \infty)$; the eigenvalues are given explicitly by $E_k = -(k-2.5)^2$ for $k = 0, 1, 2$.

(5) The Lohner equation:

$$-y''(x) - 1000xy(x) = Ey(x), \quad \text{for all } x \in (0, 1). \tag{38}$$

Lohner computed the Dirichlet eigenvalues using interval arithmetic and obtained the rigorous bounds -766.18925895^{39}_{41}, 508.108007^{5}_{3}, 24174.85^{6}_{4}.

(6) The Marletta equation:

$$-y''(x) + \frac{3(x-31)}{4(x+1)(x+4)^2}y(x) = Ey(x), \text{ for all } x \in [0, +\infty). \tag{39}$$

The spectrum is discrete below zero, with at most one negative eigenvalue, and continuous above zero. For the boundary condition $5y(0) + 8py'(0) = 0$ there is the

Table IV. The Morse Equation

k	SLEIGN2	SLEDGE	MATSLISE/MATSLISE 2.0
0	−6.2500004	−6.250003	−6.2500000000
1	−2.2499986	−9.000005	−2.2500000000
2	−0.2500010	−0.250000	−0.2500000000

Table V. The Lohner Equation

k	SLEIGN2	SLEDGE	MATSLISE/MATSLISE 2.0
0	−766.189209	−766.1892589540	−766.1892589540
9	508.10800738	508.108007483843	508.1080073843
49	24174.8549	24174.854861270	24174.854861272

Table VI. The Marletta Equation

k	SLEIGN2	SLEDGE	MATSLISE	MATSLISE 2.0
0	−1.1852133	−1.185214105	−1.185214105	−1.185214105

negative eigenvalue E_0 near -1.185. The equation with $E = 0$ has a solution,

$$y(x) = \frac{1 - x^2}{(1 + x/4)^{5/2}}, \text{ for all } x \in [0, +\infty),$$

which also satisfies the boundary condition $5y(0) + 8py'(0) = 0$; this solution deceives SLEDGE into reporting $E = 0$ as a second eigenvalue; SLEIGN2 and MATSLISE 2.0 correctly report that E_0 is the only eigenvalue, and that the continuous spectrum starts at 0.

REFERENCES

Frederick V. Atkinson. 1964. *Discrete and Continuous Boundary Problems*. Academic Press, New York, NY.

Paul B. Bailey, William N. Everitt, and Anton Zettl. 2001. The SLEIGN2 Sturm-Liouville code. *ACM Trans. Math. Softw.* 21 (2001), 143–192.

Paul B. Bailey, Marilyn K. Gordon, and Lawrence F. Shampine. 1978. Automatic solution of the Sturm-Liouville problem. *ACM Trans. Math. Softw.* 4 (1978), 139–207.

Nelson Dunford and Jacob T. Schwartz. 1988. *Linear Operators. Part I: Operators; Part II: Spectral Theory*. John Wiley Interscience, New York, NY.

William N. Everitt. 2005. A catalogue of Sturm-Liouville differential equations. *Sturm-Liouville Theory: Past and Present* (2005), 271–331.

Liviu Gr. Ixaru. 2000. CP methods for the Schrödinger equation. *J. Comput. Appl. Math.* 125 (2000), 347–357.

Liviu Gr. Ixaru, Hans De Meyer, and Guido Vanden Berghe. 1997. CP methods for the Schrödinger equation, revisited. *J. Comput. Appl. Math.* 88 (1997), 289–314.

Liviu Gr. Ixaru, Hans De Meyer, and Guido Vanden Berghe. 1999. SLCPM12 - A program for solving regular Sturm-Liouville problems. *Comp. Phys. Commun.* 118 (1999), 259–277.

Veerle Ledoux. 2007. *Study of Special Algorithms for solving Sturm-Liouville and Schrödinger Equations*. Ph.D. Dissertation. Ghent University.

Veerle Ledoux, Marnix Van Daele, and Guido Vanden Berghe. 2004. CP methods of higher order for Sturm-Liouville and Schrödinger equations. *Comp. Phys. Commun.* 162 (2004), 151–165.

Veerle Ledoux, Marnix Van Daele, and Guido Vanden Berghe. 2005. Matslise: A matlab package for the numerical solution of Sturm-Liouville and Schrodinger equations. *ACM Trans. Math. Software* 31 (2005), 532–554.

Veerle Ledoux, Liviu Gr. Ixaru, Margarit Rizea, Marnix Van Daele, and Guido Vanden Berghe. 2006. Solution of the Schrödinger equation over an infinite integration interval by perturbation methods, revisited. *Comp. Phys. Commun.* 175 (2006), 612–619.

Veerle Ledoux and Marnix Van Daele. 2010. Solving Sturm-Liouville problems by piecewise perturbation methods, revisited. *Comput. Phys. Commun.* 181 (2010), 1335–1345.

Veerle Ledoux and Marnix Van Daele. 2011. On CP, LP and other piecewise perturbation methods for the numerical solution of the Schrödinger equation. *Z. Angew. Math. Phys.* 62 (2011), 993–1011.

Steven Pruess and Charles T. Fulton. 1993. Mathematical software for Sturm-Liouville problems. *ACM Trans. Math. Softw.* 19 (1993), 360–376.

Steven Pruess, Charles T. Fulton, and Yuantao Xie. 1999. *The Automatic Classification of Sturm-Liouville Problems*. FIT Tech. Report.

John D. Pryce. 1993. *Numerical Solution of Sturm-Liouville Problems*. Oxford University Press, Oxford.

John D. Pryce. 1999. A test package for Sturm-Liouville solvers. *ACM Trans. Math. Softw.* 25 (1999), 21–57.

Joachim Weidmann. 1987. *Spectral Theory of Ordinary Differential Operators*. Springer-Verlag, Heidelberg.

Anton Zettl. 2005. *Sturm-Liouville Theory*. American Mathematical Society.

Received May 2014; revised May 2015; accepted October 2015