MATSLISE: A MATLAB Package for the Numerical Solution of Sturm-Liouville and Schrödinger Equations

V. LEDOUX, M. VAN DAELE, and G. VANDEN BERGHE
Ghent University

MATSLISE is a graphical MATLAB software package for the interactive numerical study of regular Sturm-Liouville problems, one-dimensional Schrödinger equations, and radial Schrödinger equations with a distorted Coulomb potential. It allows the fast and accurate computation of the eigenvalues and the visualization of the corresponding eigenfunctions. This is realized by making use of the power of high-order piecewise constant perturbation methods, a technique described by Ixaru. For a well-outlined class of problems, the implemented algorithms are more efficient than the well-established SL-solvers SL02F, SLEDGE, SLEIGN, and SLEIGN2, which are included by Pryce in the SLDRIVER code that has been built on top of SLTSTPAK.

Categories and Subject Descriptors: G.4 [**Mathematical Software**]: *Algorithm design and analysis user interfaces*; G.1.7 [**Numerical Analysis**]: Ordinary Differential Equations; J.2 [**Physical Sciences and Engineering**]: *Mathematics and statistics*

General Terms: Design

Additional Key Words and Phrases: Sturm-Liouville problems, Schrödinger equations, software package

1. INTRODUCTION

Many physical phenomena, both in classical mechanics and in quantum mechanics, are described mathematically by Sturm-Liouville (SL) problems (or Schrödinger problems, which are special cases of Sturm-Liouville problems). There has been much interest in the numerical solution of SL problems and several computer codes have been developed, among them the well-established Fortran codes SLEDGE [Pruess and Fulton 1993], SLEIGN [Bailey et al. 1978], SLEIGN2 [Bailey et al. 1991], and SL02F [Pryce and Marletta 1992]. These four

V. Ledoux is Research Assistant of the Fund for Scientific Research-Flanders (Belgium) (F.W.O.-Vlaanderen).
Authors' Addresses: Department of Applied Mathematics and Computer Science, Ghent University, Krijgslaan 281-S9, B-9000 Gent, Belgium; email: {Veerle.Ledoux,Marnix.VanDaele,Guido.VandenBerghe}@Ugent.be.

SL solvers are all included in the SLTSTPAK package [Pryce 1999]. Recently it has been shown that for regular problems a very valuable alternative exists for the well-known codes, namely, the CP methods. The CPM family (abbreviation for Constant reference potential Perturbation Method), was described in detail in Ixaru [1984], revisited in Ixaru et al. [1997, 2000], and extended for applications involving regular Sturm-Liouville problems in Ixaru et al. [1999], radial Schrödinger equations with Coulomb-like potentials in Ixaru et al. [2000], and systems of coupled Schrödinger equations in Ixaru [2002]. Since the so-called CPM{12, 10} was very successful, higher-order versions of the CP method have been constructed in Ledoux et al. [2004]: CPM{14, 12}, CPM{16, 14}, and CPM{18, 16}. The CP methods exhibit the very important advantage that with them the energy dependence of the error is bounded. This is in contrast to some other numerical methods where the error increases with the energy so that separate techniques have to be introduced in order to partially remove such behavior (see [Andrew and Paine 1985; Paine et al. 1981] for such a technique). As a direct consequence of this energy independence of the error, the step widths are unusually big and the computation is fast. Although we do not ignore the fact that at present SLEDGE, SLEIGN, SLEIGN2, and SL02F have a larger range of applicability when it comes to nonregular problems, for a well-outlined class of problems the CP algorithm is more efficient.

The currently available codes were all written in Fortran. But given the interest of researchers in various fields (quantum chemistry, quantum physics, ...) in this type of software, more user-friendly and effective software is required. Therefore a graphical user interface (GUI) has been developed. This GUI allows one to enter the input in a straightforward manner, to control certain parameters interactively, and to present the results graphically. At present the GUI collects the CPM codes for regular Schrödinger equations, SL problems, and Schrödinger equations with distorted Coulomb potentials.

2. THE CPM ALGORITHM

2.1 The Regular Sturm-Liouville Problem

The regular Sturm-Liouville problems read

$$-\frac{d}{dr}\left(p(r)\frac{dz}{dr}\right)+q(r)z=Ew(r)z, \qquad r_{\min}<r<r_{\max}, \tag{1}$$

where $r_{\min}$ and $r_{\max}$ are finite, functions p, q, and w are defined on the closed interval $[r_{\min}, r_{\max}]$ with p and w strictly positive (it is tacitly assumed that q is continuous and that p and w can be differentiated twice on that interval), and the boundary conditions read:

$$a_0 z(r_{\min})+b_0 p(r_{\min})z'(r_{\min})=0, \quad a_1 z(r_{\max})+b_1 p(r_{\max})z'(r_{\max})=0, \tag{2}$$

where the constants a_0 and b_0 are not both zero and similarly for a_1 and b_1. A value of the parameter E for which there is a nontrivial solution subject to the boundary conditions, is called an *eigenvalue*, and the solution z is the corresponding eigenfunction. A fundamental theorem on regular SL problems is

THEOREM 2.1. *For a regular SL problem the following conditions hold:*

(1) *The eigenvalues E_k are simple (i.e., there do not exist two linearly independent eigenfunctions with the same eigenvalue).*
(2) *The E_k can be ordered as an increasing sequence tending to infinity,*

$$E_0 < E_1 < E_2 < \cdots$$

and with this labeling the eigenfunction z_k corresponding to E_k has exactly k zeros on the open interval $(r_{\min}, r_{\max})$. The integer k is called the index of the eigenvalue E_k.
(3) *The z_k form a complete orthogonal set of functions over $(r_{\min}, r_{\max})$ with respect to the inner product*

$$\langle z_1, z_2 \rangle = \int_{r_{\min}}^{r_{\max}} z_1(r) z_2(r) w(r) dr.$$

Our aim is to solve the Sturm-Liouville problems, that is, to calculate the set of eigenvalues and eigenfunctions, using the CP methods. The CP methods are constructed for equations of the Schrödinger form, not for equations of the SL form. For this reason these methods can be applied to the SL problems only if the SL equation can be converted to the Schrödinger form. The conversion is possible and is achieved via the so-called Liouville's transformation. In fact with Liouville's transformation

$$x = \int_{r_{\min}}^{r} \sqrt{w(r')/p(r')}\, dr', \tag{3}$$

and $m = (pw)^{-1/4}$, Equation (1) becomes of the Schrödinger form in the x variable, namely:

$$y'' = (V(x) - E)y, \qquad 0 = x_{\min} < x < x_{\max}, \tag{4}$$

where the potential function $V(x)$ is (see Section 2.5 in Pryce [1993])

$$V(x) = \frac{q}{w} + m \frac{d^2}{dx^2}\left(\frac{1}{m}\right). \tag{5}$$

With x and r connected as in Equation (3) the original $z(r)$ and the new $y(x)$ are related as follows:

$$z(r) = m(r)y(x), \quad \frac{dz}{dr} = m'(r)y(x) + y'(x)/(m(r)p(r)), \tag{6}$$

and the original boundary conditions (2) now read

$$A_0 y(x_{\min}) + B_0 y'(x_{\min}) = 0, \quad A_1 y(x_{\max}) + B_1 y'(x_{\max}) = 0, \tag{7}$$

with $B_0 = b_0$, $B_1 = b_1$ and

$$\begin{aligned} A_0 &= a_0 m^2(r_{\min}) + b_0 p(r_{\min}) m'(r_{\min}) m(r_{\min}), \\ A_1 &= a_1 m^2(r_{\max}) + b_1 p(r_{\max}) m'(r_{\max}) m(r_{\max}). \end{aligned} \tag{8}$$

In short, the original regular SL problem, Equations (1) and (2), is equivalent to the Schrödinger boundary value problem, Equations (4) and (7), and the latter is actually the one which is solved numerically. The old r and the new

x are related by a quadrature, Equation (3); in Ixaru et al. [1999] an efficient procedure was suggested for calculating x for a given r and vice versa. First a set of points $r_0^G = r_{\min} < r_1^G < r_2^G < \cdots < r_{K+1}^G = r_{\max}$ is identified, such that the integral

$$q_k = \int_{r_k^G}^{r_{k+1}^G} \sqrt{w(r')/p(r')}dr', \quad k = 0, 1, \ldots K, \tag{9}$$

evaluated by a Gauss formula with 12 points (denoted as Q_k) is correct in all digits available in the double-precision arithmetic. More exactly, the interval $[r_k^G, r_{k+1}^G]$ is taken small enough such that $|Q_k^{(1)} + Q_k^{(2)} - Q_k| < \epsilon$, with ϵ a precision threshold representing the double-precision arithmetic and $Q_k^{(1)}$ and $Q_k^{(2)}$ the numerical values of the integrals

$$q_k^{(1)} = \int_{r_k^G}^{\left(r_{k+1}^G + r_k^G\right)/2} \sqrt{w(r')/p(r')}dr', \quad q_k^{(2)} = \int_{\left(r_{k+1}^G + r_k^G\right)/2}^{r_{k+1}^G} \sqrt{w(r')/p(r')}dr', \tag{10}$$

evaluated by the same 12-points Gauss method. The value of x_k^G associated with r_k^G is $x_k^G = q_0 + q_1 + \cdots + q_{k-1}$ and we store the x^G and the r^G in some vectors. When, during the computations, the value of x corresponding to some given r is required, then the interval $[r_k^G, r_{k+1}^G]$ which contains the input r is first identified and only the integral from r_k^G up to r is evaluated. Conversely, when x is given and r is required, one should first identify the interval $[x_k^G, x_{k+1}^G]$ and then r is evaluated as the root of

$$g(r) = \int_{r_k^G}^{r} \sqrt{w(r')/p(r')}dr' - x + x_k^G, \tag{11}$$

by a Newton iteration procedure.

2.2 Solving Schrödinger Equations: the CP Method

We first focus on the initial value problem for the Schrödinger equation,

$$y'' = (V(x) - E)y, \quad x \in [a, b], \quad y(a) = \alpha, \quad y'(a) = \beta, \tag{12}$$

where $V(x)$, the potential, is supposed to be a well-behaved function and E, the energy, is a constant. The current interval I_k of the partition is denoted generically by $I = [X, X + h]$. On I the solution is advanced by the so-called propagation matrix algorithm:

$$\begin{bmatrix} y(X+h) \\ y'(X+h) \end{bmatrix} = \begin{bmatrix} u(h) & v(h) \\ u'(h) & v'(h) \end{bmatrix} \begin{bmatrix} y(X) \\ y'(X) \end{bmatrix}. \tag{13}$$

The functions $u(\delta)$ and $v(\delta)$, where $\delta = x - X$, called *propagators*, are the solutions of the local problem

$$y''(\delta) = (V(X + \delta) - E)y(\delta), \quad \delta \in [0, h], \tag{14}$$

with the initial values $y(0) = 1$, $y'(0) = 0$, for u and $y(0) = 0$, $y'(0) = 1$ for v. The one-step propagation matrix is

$$\mathbf{P}(\delta) = \begin{bmatrix} u(\delta) & v(\delta) \\ u'(\delta) & v'(\delta) \end{bmatrix} \tag{15}$$

and its inverse reads

$$\mathbf{P}^{-1}(\delta) = \begin{bmatrix} v'(\delta) & -v(\delta) \\ -u'(\delta) & u(\delta) \end{bmatrix}, \tag{16}$$

because $\mathrm{Det}[\mathbf{P}(\delta)] = 1$. It follows that the knowledge of $u(h), v(h), u'(h)$, and $v'(h)$ is sufficient to advance the solutions in both directions.

To construct the propagators $v(\delta)$ and $u(\delta)$, the perturbation approach is used. Let

$$\bar{V} = \frac{1}{h}\int_0^h V(X+\delta)\,d\delta\,, \quad \Delta V(\delta) = V(X+\delta) - \bar{V}. \tag{17}$$

The original potential then reads $V(X+\delta) = \bar{V} + \Delta V(\delta)$, where $\bar{V}$ is a constant. The procedure consists in taking $\bar{V}$ as the reference potential and $\Delta V(\delta)$ as a perturbation.

As explained in Ixaru [1984], each of $u(\delta)$ and $v(\delta)$, denoted generically as $p(\delta)$, is written as a perturbation series:

$$p(\delta) = p_0(\delta) + p_1(\delta) + p_2(\delta) + p_3(\delta) + \ldots, \tag{18}$$

where the zeroth-order term $p_0(\delta)$ is the solution of $p_0'' = (\bar{V} - E)p_0$ with $p_0(0) = 1$, $p_0'(0) = 0$ for u_0 and $p_0(0) = 0$, $p_0'(0) = 1$ for v_0. The correction $p_q, q = 1, 2, \ldots$ obeys the equation

$$p_q'' = (\bar{V} - E)p_q + \Delta V(\delta)p_{q-1}\,, \quad p_q(0) = p_q'(0) = 0. \tag{19}$$

With $Z(\delta) = (\bar{V} - E)\delta^2$ and functions $\xi(Z)$, $\eta_0(Z)$, $\eta_1(Z)$, ..., defined in the Appendix, the zeroth-order propagators are

$$u_0(\delta) = \xi(Z(\delta))\,, \quad v_0(\delta) = \delta\eta_0(Z(\delta)), \tag{20}$$

and the following iteration procedure exists to construct the corrections.

Correction p_{q-1} is assumed to be known and of such a form that the product $\Delta V(\delta)p_{q-1}(\delta)$ reads

$$\Delta V(\delta)p_{q-1}(\delta) = Q(\delta)\xi(Z(\delta)) + \sum_{m=0}^{+\infty} R_m(\delta)\delta^{2m+1}\eta_m(Z(\delta)). \tag{21}$$

Then $p_q(\delta)$ and $p_q'(\delta)$ are of the form

$$p_q(\delta) \;=\; \sum_{m=0}^{+\infty} C_m(\delta)\delta^{2m+1}\eta_m(Z(\delta)), \tag{22}$$

$$p_q'(\delta) \;=\; C_0(\delta)\xi(Z(\delta)) + \sum_{m=0}^{+\infty}(C_m'(\delta) + \delta C_{m+1}(\delta))\delta^{2m+1}\eta_m(Z(\delta)), \tag{23}$$

where $C_0(\delta)$, $C_1(\delta)$, ... are given by quadrature (again see Ixaru [1984]):

$$C_0(\delta) \;=\; \frac{1}{2}\int_0^\delta Q(\delta_1)\,d\delta_1, \tag{24}$$

$$C_m(\delta) \;=\; \frac{1}{2}\delta^{-m}\int_0^\delta \delta_1^{m-1}[R_{m-1}(\delta_1) - C_{m-1}''(\delta_1)]\,d\delta_1\,, \quad m = 1, 2, \ldots\,. \tag{25}$$

To calculate successive corrections for u, the starting functions in $\Delta V(\delta)p_0(\delta)$ are $Q(\delta) = \Delta V(\delta)$, $R_0(\delta) = R_1(\delta) = \cdots = 0$, while for v they are $Q(\delta) = 0$, $R_0(\delta) = \Delta V(\delta)$, $R_1(\delta) = R_2(\delta) = \cdots = 0$.

The practical inconvenience is that successive quadratures starting from an arbitrary $\Delta V(\delta)$ are difficult to manipulate. For this reason, there is an intermediary stage in the procedure in which $V(X+\delta)$ is approximated by a polynomial in δ. More exactly, it is assumed that $V(X+\delta)$ can be written as a series over shifted Legendre polynomials $P_n^*(\delta/h)$ in the following way:

$$V(X+\delta) = \sum_{n=0}^{+\infty} V_n h^n P_n^*(\delta/h). \tag{26}$$

The original $V(X+\delta)$ is then approximated by the truncated series

$$V^{(N)}(X+\delta) = \sum_{n=0}^{N} V_n h^n P_n^*(\delta/h), \tag{27}$$

and therefore the equation

$$y^{(N)\prime\prime} = \left(V^{(N)}(X+\delta) - E\right) y^{(N)}, \quad \delta \in [0,h], \tag{28}$$

is the one whose propagators are actually constructed via CPM. With

$$\bar{V} = V_0, \quad \Delta V(\delta) = \Delta V^{(N)}(\delta) = \sum_{n=1}^{N} V_n h^n P_n^*(\delta/h), \tag{29}$$

the integrals (24)–(25) can be solved analytically. Each $C_m(\delta)$ is a polynomial and the series (22) and (23) are finite. The values $\bar{V}_i = V_i h^{i+2}$, $i = 0, 1, \ldots, N$, are actually important. They are calculated by quadrature

$$\bar{V}_i = (2i+1)h \int_0^h V(X+\delta) P_i^*(\delta/h)\, d\delta. \tag{30}$$

Each value for N and for the maximal number of perturbation corrections Q would result in a version identified as CPM[N,Q].

The versions described in Ixaru [1984] take either $N = Q = 0$ or $N = 2$ as a default value and $Q = 1, 2$. The existence of powerful packages for symbolic computations, however, enabled us to obtain expressions of $u(h)$, $hu'(h)$, $v(h)/h$, and $v'(h)$ with more terms. This allows an alternative way of formulating and identifying a CPM version. We can just ask to retain in the algorithm only the terms consistent with some input values for the order: n_0 and n_{as}. This version is denoted as CPM$\{n_0, n_{as}\}$, meaning that the order of the algorithm is n_0 when $Z(h) = (V_0 - E)h^2 \to 0$ and of order n_{as} for $-Z(h) \to +\infty$. This will lead to a unique N but to a sum over incomplete perturbations. In Ixaru et al. [1997] the code CPM{12, 10} was introduced and in Ledoux et al. [2004] versions of higher-order CPM{14, 12}, CPM{16, 14}, and CPM{18, 16} were proposed. The expressions of $u(h)$, $hu'(h)$, $v(h)/h$, and $v'(h)$ of the CPM$\{n_0, n_{as}\}$ algorithms have the following form:

$$u(h) = \xi(Z) + \sum_{m=1}^{\infty} C_m^{(u)} \eta_m(Z), \tag{31}$$

$$hu'(h) = Z\eta_0(Z) + \sum_{m=0}^{\infty} C_m^{(u')}\eta_m(Z), \tag{32}$$

$$v(h)/h = \eta_0(Z) + \sum_{m=2}^{\infty} C_m^{(v)}\eta_m(Z), \tag{33}$$

$$v'(h) = \xi(Z) + \sum_{m=1}^{\infty} C_m^{(v')}\eta_m(Z). \tag{34}$$

where the coefficients $C_m^{(u)}$, $C_m^{(u')}$, $C_m^{(v)}$, and $C_m^{(v')}$ are expressed in terms of the $\bar{V}_i$ variables from Equation (30).

The possibility of introducing an asymptotic order is very important. For the CP methods it means that the partition of a finite integration interval can be formulated from the very beginning of the run and never altered again, no matter how small or how big the energy is. The E-independent coefficients $C_m^{(u)}$, $C_m^{(u')}$, $C_m^{(v)}$, and $C_m^{(v')}$ are also computed once on each step and stored. When the solution for a given E is advanced on successive steps, only the E-dependent ξ and η_m remain to be calculated. This possibility of separating the relatively time-consuming task of generating the partition (of a finite integration interval) and of calculating the information to be used later on, from the repeatedly asked but fast executable task of integrating the equation at various values for E, explains why the run with the CP algorithms is so efficient. The CP methods are uniformly fast over the whole energy range. For a problem with an infinite integration interval, however, this valuable feature is lost. Such an infinite problem is regularized by truncating the integration interval but the values of the truncated endpoints depend on the value of the eigenenergy E. This means that the size of the integration interval (and thus also the partition) can be different for various values of E (see further discussion in Section 3.1).

To locate the eigenvalues of the boundary-value problem for the Schrödinger equation, a shooting procedure is used. For each (test)-value of E, the solution is advanced in two directions, forward from a up to the matching point x_{match} and backward from b down to x_{match}. The numerical method used to advance the solution is one of the CP methods introduced for the initial value problem. This enables obtaining not only y and $y\prime$ at each meshpoint x_i of the partition but also their derivatives with respect to E: $y_E(x_i)$ and $y'_E(x_i)$. The latter data is important both for the location of the eigenvalue by Newton iteration and for an accurate normalization of the eigenfunction. At the same time, the values of $y(x_i)$ and $y'(x_i)$ are used to construct the scaled Prüfer form of the solution. This scaled Prüfer form makes the entire eigenvalue finding process more robust. It enables a correct estimation of the eigenvalue index and therefore it prevents any accidental jump over some eigenvalue during the search. In Ixaru et al. [1997], more technical details on the procedure were given.

2.3 Radial Schrödinger Equations with Distorted Coulomb Potential

We consider the radial Schrödinger equation of the following form:

$$y'' = \left(\frac{l(l+1)}{r^2} + V(r) - E\right) y, \quad r > 0, \tag{35}$$

and follow the theory developed in Ixaru et al. [2000]. We are concerned with potentials of the form

$$V(r) = \frac{S(r)}{r} + R(r), \tag{36}$$

where $S(r)$ and $R(r)$ are well behaved functions such that

$$\lim_{r\to 0} S(r) = S_0, \quad \lim_{r\to\infty} S(r) = S_{\text{as}},$$

and

$$\lim_{r\to 0} R(r) = R_0, \quad \lim_{r\to\infty} R(r) = R_{\text{as}},$$

where S_0, S_{as}, R_0, and R_{as} are constants. Specifically, it is assumed that around the origin $S(r)$ and $R(r)$ can be written in polynomial form

$$S(r) = \sum_{m=0}^{M} S_m r^m, \quad R(r) = \sum_{m=0}^{M-1} R_m r^m, \quad 0 \le r \le r_0, \tag{37}$$

and that some r_{as} exists such that

$$V_{\text{as}}(r) = \frac{S_{\text{as}}}{r} + R_{\text{as}} \tag{38}$$

is a good approximation of $V(r)$ for all $r > r_{as}$. The eigenenergies are also located by shooting, that is, the half-axis $r > 0$ is cut at some large $r_{\max} > r_{\text{as}}$ and a matching point r_{match} is fixed somewhere on $I = (0, r_{\max}]$. For the radial Schrödinger equations considered, a specific problem occurs: the equation is singular at the origin and therefore the numerical integration around the origin should be carried out by a procedure which explicitly accounts for this fact. Since, on the other hand, the effect of the singularity progressively dies out when r is increased, the integration interval I has to be split into two subintervals, a narrow subinterval around the origin, $I_1 = (0, r_0]$, and the remaining $I_2 = (r_0, r_{\max}]$. The algorithm to be used on I_1 should be consistent with the singular nature of the equation. An accurate solution for potentials of the form (36) on I_1 is given in Ixaru et al. [2000]. On I_2, one of the CP algorithms described in Section 2.2 can be chosen.

In the procedure described in Ixaru et al. [2000], the parameter l is assumed to be a positive integer (because l represents the orbital quantum number from the atomic physics field). However, some Schrödinger problems with a singular endpoint in $a = 0$ can also be written down as in (35), but with a negative or noninteger value l. We reflect now that the procedure is easily extended to negative or noninteger values of l. When $l \le -1$, then l can be replaced by the positive value

$$l' = [-1 + \sqrt{1 + 4l(l+1)}]/2 \tag{39}$$

such that $l'(l'+1) = l(l+1)$. The only problem, caused by noninteger values of $l > -1$, is then that η_{l+s} $(s = 0, 1, \ldots)$ is not defined. However, the η functions can be expressed in terms of Bessel functions, thus removing the need for l to be an integer (see Equation (A.8)).

3. STRUCTURE OF MATSLISE

The package MATSLISE is available for download from online at

`http://allserv.ugent.be/~vledoux/MATSLISE/`

It uses the MATLAB symbolic toolbox, for example, to calculate the derivatives of the coefficient functions of a SL equation, and has been developed with Windows version 7 of MATLAB. By changing to the MATSLISE directory or by adding the MATSLISE directory (and subdirectories) to the path, all methods mentioned in this section can be called.

The MATSLISE package allows at present the numerical solution of regular Sturm-Liouville and Schrödinger problems and a limited class of singular problems, for instance, problems with a distorted Coulomb potential. Let's consider again the general form of the Sturm-Liouville problem (of which the Schrödinger problem is a special case)

$$-\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right)+q(x)y=Ew(x)y,\qquad -\infty\le a<x<b\le\infty, \tag{40}$$

where we assume p, q, and w are continuous on the open interval (a,b) and $p>0$, $w>0$ on (a,b). An endpoint e ($e=a$ or $e=b$) of the SL problem (40) is called a *regular endpoint* if and only if (i) e is finite and (ii) $1/p(x)$, $q(x)$, and $w(x)$ are absolutely integrable near $x=e$. Otherwise it is called a *singular endpoint*. There are two fundamental disjoint types into which the SL equation is classified at each endpoint (see [Pryce 1993]): (i) limit-point (LP) or limit-circle (LC), and (ii) nonoscillatory (N) or oscillatory (O) for real values of E. For a given real E, Equation (40) is *oscillatory* at $x=e$ if and only if every solution has infinitely many zeros clustering at e. Otherwise it is called *nonoscillatory* at $x=e$. This N/O classification can change with E. The MATSLISE package presented here can only calculate the eigenvalues in a discrete spectrum which is bounded below. This means that MATSLISE cannot handle problems with an endpoint which is oscillatory for all real E values. Also problems with discontinuous coefficients are refused in the current MATSLISE version. Infinite endpoints, however, are allowed: a problem with an infinite endpoint is approximated by a regular problem on a truncated interval. The choice of this truncated interval will be discussed in Section 3.1.

The numerical solution of the SL or Schrödinger problems by MATSLISE is subdivided into three stages. Into the first stage, the partition is constructed, and this partition is passed into the second stage where the eigenvalues are calculated. In the third stage, it is possible to calculate the eigenfunctions of some of the eigenvalues. Each of these stages has its own method(s), and information is passed from one stage to the other by the input arguments of these method(s).

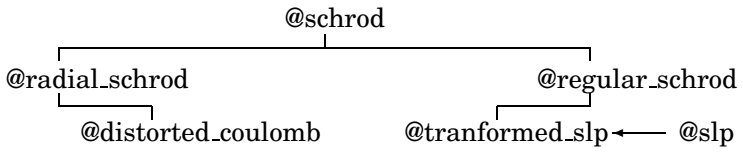
In order to make the package easy to maintain and extend, an object-oriented way of programming has been used. The basic principles of object-oriented programming are supported by MATLAB: function overloading, inheritance, encapsulations, and aggregation.

3.1 Construction of the Partition/Liouville Transformation (Stage 1)

3.1.1 *Specification of the Problem.* A regular Schrödinger object is created by:

```
sch = regular_schrod(V,a,b,a0,b0,a1,b1)
sch = regular_schrod(V,a,b,a0,b0,a1,b1,var)
```

where `V` is a string representing the potential function V. The double-precision constants `a, b, a0, b0, a1`, and `b1` specify the integration interval and the boundary conditions. It is allowed to enter `-inf` or `inf` for the two input parameters `a` and `b`. Good truncated endpoints are then determined automatically for every eigenvalue (see later). If there are fewer then eight input arguments, then the independent variable is supposed to be x; otherwise the independent variable is the character or string in `var` and `V` is then V(`var`). The method `regular_schrod(\ldots)` is the constructor of the class `regular_schrod`. An overview of the classes used in the representation of the equations is displayed in the following scheme:



Notice the inheritance used: all problems will be represented in the MATSLISE code by subclasses of the parent class `schrod` (even the Sturm-Liouville problems after the Liouville transformation).

In a very analogous manner a Sturm-Liouville equation is specified:

```
sl = slp(p,q,w,a,b,a0,b0,a1,b1)
sl = slp(p,q,w,a,b,a0,b0,a1,b1,var)
```

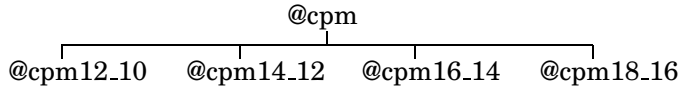
where `p, q`, and `w` are strings representing the coefficient functions $p(x)$, $q(x)$, and $w(x)$ (or p(`var`), q(`var`), and w(`var`)) and the double-precision numbers `a`, `b`, `a0`, `b0`, `a1`, `b1` are the endpoints of the integration interval and the coefficients of the boundary conditions. Again `inf`-values are allowed for `a` and `b`.

An object representing an equation with a distorted Coulomb potential is produced by

```
d = distorted_coulomb(l,S,R)
d = distorted_coulomb(l,S,R,var)
```

where `S` and `R` are two strings (representing $S(x)$ and $R(x)$) and the orbital quantum number `l` is a double.

3.1.2 *Initialization of the CP Method.* In addition to the classes for the equations themselves, a number of classes which implement the actual CP methods, was developed:



These classes collect the methods, which construct the partition and calculate the eigenvalues and eigenfunctions, and all these methods take a `schrod`-object as input. The parent class `cpm` contains all methods and properties which are shared among the various CP algorithms. The child classes `cpm12_10`, `cpm14_12`, `cpm16_14`, and `cpm18_16` only contain the information which is specific for a certain order. In this way, a new method of a given order can be added easily.

The user calls a constructor of a child class, for example:

```
cp = cpm16_14(slp/regular_schrod,tol)
cp = cpm16_14(distorted_coulomb,tol,rmax)
```

with `tol` a positive constant representing the accuracy requested in the results. This constructor of the child class then calls the constructor of `cpm`. The `cpm` constructor is never called by the user directly. The constructors start the calculation of the partition but in two cases some additional work is done first:

—When the first input argument is a `slp` object, then the SL equation is converted first to the Schrödinger form: that is, the `slp` object is transformed into a `transformed_slp`-object. A `transformed_slp`-object is a `regular_schrod`-object but contains some more information which is necessary, for example, to obtain the eigenfunctions of the original SL equation.

—When the first input argument is a `distorted_coulomb`-object, r_0 is set by an empirical formula and the polynomial form 37 of $S(x)$ and $R(x)$ is calculated.

Then the partition of $[a, b]$ (or $(r_0, r_{\max})$ for distorted Coulomb problems) is constructed in terms of the tolerance `tol` and some potential dependent expressions ($C_m^{(u)}$, $C_m^{(u')}$, $C_m^{(v)}$, $C_m^{(v')}$ from Equation (34) and V_0 from Equation (29)), which will be used repeatedly in the second and third stages, are calculated in each step of the partition and stored. Additionally the execution of this stage furnishes the value of the matching point x_{match}. It is important to point out that the partition is dictated only by the behavior of $V(x)$; the value of E is not involved. So the construction of the partition (stage 1) happens completely in advance of and separate from the calculation of the eigenvalues and eigenfunctions (stages 2 and 3). At least this is true for a problem with a finite integration interval. For an infinite integration interval, however, the E-independence of the partition is lost: a higher eigenvalue needs a larger (truncated) integration interval than a lower eigenvalue. This means that in the calculation of the eigenvalues (stage 2) a lengthening of the partition interval may be necessary, which makes the eigenvalue search for an infinite problem somewhat slower. We will discuss here briefly how MATSLISE regularizes a problem with an infinite integration interval by finding appropriate truncation points. As in other methods (e.g., SLEIGN, SL02F, SLEDGE), a two-pass process is done. The next part is described for the case where a is regular and b is singular ($b = \infty$), the modifications for the other possible combinations being clear. An initial point $b^* < b$ is selected (by examing the potential function), and the interval $[a, b^*]$ is covered by a very coarse mesh. By considering the potential function $V(x)$, the highest "possible" energy level E^* in the truncated interval $[a, b^*]$ can be estimated; that is, E^* is the highest energy level for which $V(b^* - \epsilon) > E^*$, with ϵ a certain parameter set by experimentation. By calculating the Prüfer variable θ associated with

E^*, we have an idea about the index of this highest "possible" eigenvalue, that is, the highest eigenvalue (of the infinite problem) which can be calculated correctly over the truncated problem. When this index is smaller than the index of the eigenvalue we are searching for, some additional intervals are added to the mesh in order to obtain a larger integration interval $[a, b^*]$. The number of intervals added is problem-dependent, that is, it depends on the slope of the potential function in b^*. The process is repeated until eventually a truncated integration interval $[a, b^*]$ is found where the index of the highest "possible" eigenvalue is larger than the index of the eigenvalue searched for. In the second phase, the partition of the truncated integration interval $[a, b^*]$ is constructed (as for a regular problem) and the main eigenvalue computation is started on this (finer) mesh. To improve the relialibity of the procedure, the integration interval is slightly lengthened and a further eigenvalue approximation is calculated. This process is repeated until two successive eigenvalue approximations agree within the tolerance specified by the user. Only one or two iterations are typically necessary in this second phase. Moreover, the shooting algorithm for the next eigenvalue approximation is always started using the approximation last obtained, which makes the process more efficient.

3.2 Calculation of the Eigenvalues (Stage 2)

In this stage the eigenvalues, in a range fixed by the user, are calculated. The user starts the calculation by calling a method from the `cpm` class:

```
E = get_eigenvalues(cp_child,pmin,pmax)
E = get_eigenvalues(cp_child,pmin,pmax,indices)
```

where `cp_child` is an instance of one of the child classes of the `cpm` class. If `indices` is `true`, the eigenvalues E_k $(k = 0, 1, \ldots)$ with indices k between `pmin` and `pmax` are calculated; if `indices` is `false` or omitted, the eigenvalues in the range [`pmin`,`pmax`] are calculated. The method returns a structure `E`, in which all information related to the calculated eigenvalue(s) is stored. The fields `E.eigenvalues`, `E.indices`, and `E.errors` are three vectors. `E.eigenvalues` contains the calculated eigenvalues in ascending order. The associated indices are collected in `E.indices`, and `E.errors` holds an estimation of the error for each eigenvalue. The field `E.success` is false when the CP method was not able to obtain the data due to an error, for example, when there is no eigenvalue in the interval [`pmin`,`pmax`]. In some cases, a warning is generated in the method `get_eigenvalues`:, for example, when two eigenvalues are so close that double-precision accuracy is not sufficient to differentiate between them adequately. The estimation of the error in `E.errors` is the difference between the calculated eigenvalue (the so-called basic eigenvalue) and a reference eigenvalue computed on the same partition but with a higher-order method. To make things clear, let us assume that the CPM{16, 14} is used. First this CPM{16, 14} is used to calculate the basic eigenvalue, and then the reference eigenvalue is calculated on the same partition but using the higher-order CPM{18, 16}. The difference between the basic eigenvalue and the (more accurate) reference eigenvalue forms the estimation of the error in the basic eigenvalue. The Newton iteration process in the shooting procedure for the reference eigenvalue starts with the

Table I. Ratio $\frac{|\text{Actual Error}|}{|\text{Error Estimate}|}$ for the Harmonic Oscillator and Hydrogen Equation

	Harmonic Oscillator			Hydrogen		
k	10^{-8}	10^{-10}	10^{-12}	10^{-8}	10^{-10}	10^{-12}
0	1.016	1.006	1.034	0.987	0.063	0.063
1	0.971	0.929	0.600	0.978	0.703	0.016
2	0.990	0.981	0.945	0.981	0.678	0.008
3	0.980	0.984	0.871	0.974	0.648	0.008
4	0.991	0.990	1.054	0.975	0.613	0.004
5	1.000	0.996	0.909	0.974	0.566	0.004
6	1.020	0.999	0.995	0.975	0.516	0
7	0.989	0.993	0.995	0.978	0.463	0.006
8	0.988	0.995	0.884	0.495	0.404	0.008
9	1.006	0.998	1.054	0.507	0.346	0.004
10	1.001	0.997	0.995	0.975	0.274	0.001

Table II. Time (s) to Compute Some Eigenvalues

	Mathieu			Hydrogen		
k	10^{-8}	10^{-10}	10^{-12}	10^{-8}	10^{-10}	10^{-12}
0	0.11	0.10	0.16	2.61	2.89	4.03
10	0.11	0.11	0.16	4.42	5.09	6.33
100	0.13	0.14	0.20	12.7	15.2	19.0
1000	0.16	0.19	0.27	47.5	59.7	77.7
10000	0.22	0.27	0.34	221	294	376

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 an extra effort which is almost negligible. Analogously, the error in a CPM{12, 10} (or CPM{14, 12}) eigenvalue is estimated using the CPM{14, 12} (or CPM{16, 14} respectively). Table I shows the ratio of the true error to the estimated error (using the CPM{16, 14}–CPM{18, 16} combination) for the harmonic oscillator

$$y'' = (x^2 - E)y, \quad x \in (-\infty, \infty), \tag{41}$$

for which the correct eigenvalues are known: $E_k = 2k + 1,\ k = 0, 1, \ldots,$ and for the hydrogen atom equation

$$y'' = \left(-\frac{1}{x} + \frac{2}{x^2} - E\right) y, \quad x > 0, \tag{42}$$

with known eigenvalues $E_k = -1/(2k+4)^2,\ k = 0, 1, \ldots$. This "goodness" ratio always has values smaller or very close to 1, which illustrates the adequacy of our error estimation. When the first eigenvalues of the hydrogen atom equation are calculated with tolerance $\leq 10^{-12}$, the obtained eigenvalues are (nearly) as accurate as the machine precision (10^{-16}). This means that the "actual error" will be very close to zero (or even zero), which explains the very small values in the last column of Table I.

Table II displays the times needed by CPM{16, 14} (on a 2.4-GHz PC) to calculate some eigenvalues of the Mathieu equation

$$y'' = \big(2\cos(2x) - E\big)\, y, \quad 0 < x < \pi, \tag{43}$$

and of the hydrogen atom equation (42). The Mathieu problem is a regular problem with a finite integration interval. This means that the shooting procedure for each eigenvalue is performed on one and the same partition which was already constructed in the first stage. This explains why the time increases only very slowly with the eigenvalue index. For the hydrogen atom equation, however, the partition is modified (lengthened) during the eigenvalue calculation. For such an infinite problem, the truncated integration interval grows with the eigenvalue index and as a consequence the calculation of a higher eigenvalue requires a larger amount of time.

3.3 Calculation of the Eigenfunctions (Stage 3)

Another method in the `cpm` class that is visible to the user is the method which allows the calculation of the eigenfunction associated with a certain eigenvalue `e`:

```
V = get_eigenfunction(cp_child,e)
V = get_eigenfunction(cp_child,e,n)
V = get_eigenfunction(cp_child,e,ap,bp,n)
```

where again `cp_child` is an instance of one of the child classes of `cpm`. To obtain a good approximation for the eigenfunction, the eigenvalue `e` must be sufficiently accurate (e.g., by choosing a small value for `tol` in stage 1). If `n` is omitted, then the eigenfunction is evaluated only in the meshpoints of the partition. In most cases the number of meshpoints in the partition is too small to have a good idea of the shape of the eigenfunction. Therefore the eigenfunction can be evaluated in more points by choosing a sufficiently high value for `n`: the interval `[ap,bp]` is then taken and partitioned in `n` intervals of equal size. On these intervals, the additional potential-dependent expressions are calculated and the propagation matrix algorithm (13) is applied to produce the eigenfunction. Only the part of the eigenfunction corresponding with the `n` + 1 points in `[ap,bp]` is returned. When the input arguments `ap` and `bp` are omitted, then the whole interval $[a, b]$ is considered. The structure `V` has three fields: the three vectors `V.x`, `V.y`, and `V.yprime` of which the meaning is clear.

4. THE GUI

The methods mentioned in Section 3 can all be called from the MATLAB command line. But in order to increase the ease of use and to hide the technical issues from the user, a graphical user interface (GUI) has been built on top of the classes in MATSLISE. It slows the computations down somewhat; but on the other hand, it facilitates giving input, it gives the user more control, and graphical features are built in. The root directory of MATSLISE contains two files: `matslise.m` and `matslise_help.m`. By entering `matslise` at the command line, the GUI is opened. `matslise_help` opens the corresponding Help files. The directory `predefined_problems` contains several example problems (saved as *.mat files), many of which are available in SLTSTPAK [Pryce 1999], and is organized as shown in Figure 1.

The GUI version of MATSLISE uses the CPM{16, 14} method to calculate the (basic) eigenvalue, and the higher-order CPM{18, 16} is used on the same

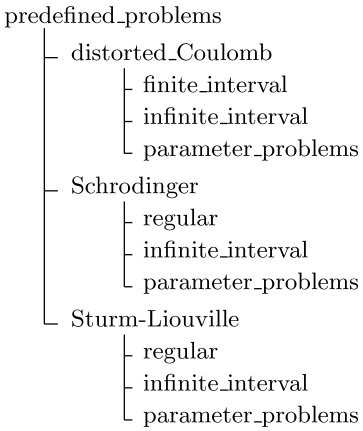


Fig. 1. The predefined_problems directory in MATSLISE.

CPM{16, 14} partition to obtain a reference eigenvalue. As seen in Section 3.2, this reference eigenvalue enables an accurate error estimation. The (slower) CPM{12, 10} and CPM{14, 12} are not used in the GUI but they can still be invoked from the command line using the constructors `cpm12_10` and `cpm14_12`, which were discussed in Section 3.1.

In Figure 2 the input window is shown for the harmonic oscillator equation (41). Similar windows can be opened for SL equations or for problems with distorted Coulomb potentials. The harmonic oscillator problem is one of the problems which is included within MATSLISE in the directory `predefined_problems/schrodinger/infinite_interval`.

After the input has been entered, the construct button starts the calculations of stage 1: that is, the constructors of the classes `regular_schrod` and `cpm16_14` are called. The partition for the harmonic oscillator on the begin interval $[-10, 10]$ and obtained by the CPM{16, 14} algorithm with tol $= 10^{-10}$ is displayed in Figure 3; the full line represents the potential function, and the x values of the thicker points are the meshpoints of the partition. The number of meshpoints in the partition is remarkably small compared to the number of oscillations an eigenfunction may have.

The partitions (and the information associated to them) are passed to a second window: the eigenvalues window (Figure 4). In this window the user specifies which eigenvalues he/she wants to calculate. Several batches of eigenvalues can be calculated one after the other without revisiting the input window. Besides plotting the eigenvalues, the associated eigenfunctions can be computed. Figure 5 shows the first 12 eigenfunctions for the harmonic oscillator obtained by MATSLISE. Some more plots of the eigenfunctions can be made in the GUI: for example, the eigenfunctions together with the potential function as shown in Figure 6. The correctness of the eigenfunctions can be tested via an orthogonality check. This check applies the trapezoidal rule on each interval between two points where the eigenfunction was evaluated, in order to compute a crude approximation of $\int y_k(x)y_l(x)dx$ (or $\int z_k(x)z_l(x)w(x)dx$ for an SL problem). When two eigenfunctions seem to be not orthogonal, the eigenfunctions cannot be

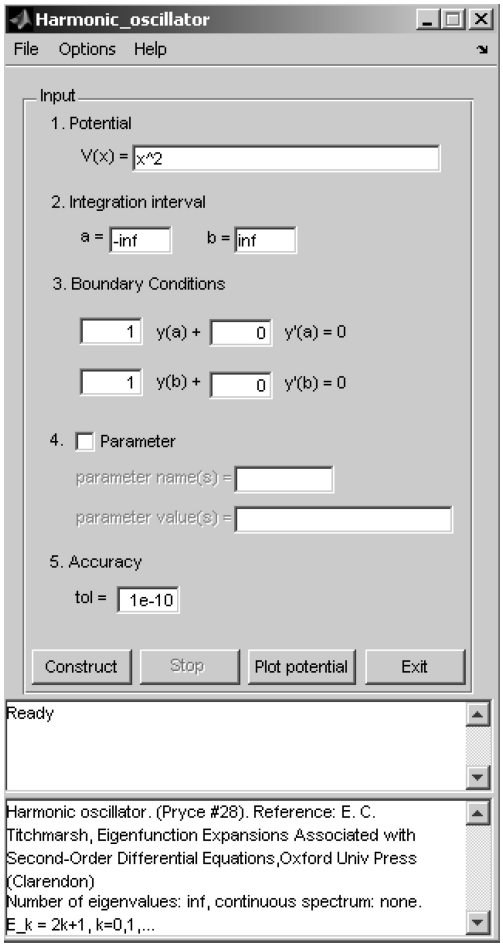


Fig. 2. Example of the input window of MATSLISE.

correct. This inadequacy can be caused in two ways:

—The eigenvalue is not accurate enough to compute the associated eigenfunction correctly. Decreasing the input tolerance might help in this case.

—Very close eigenvalues occur and as mentioned in Pryce [1993] clustering causes the eigenfunctions to be very ill-conditioned. Therefore the user will be warned and asked to be cautious when close eigenvalues are detected. For symmetric double well problems, half-range reduction may make the problem more tractable: check the option "Half-range reduction" in the Options menu of the input window.

A Schrödinger problem (or SL problem) is symmetric when the problem is posed on the interval $-b$ to b, where b may be ∞, the potential function is even and the boundary conditions are similarly symmetric. In this case the eigenfunctions belonging to eigenvalue E_k, $(k = 0, 1, \ldots)$ are even or odd functions according

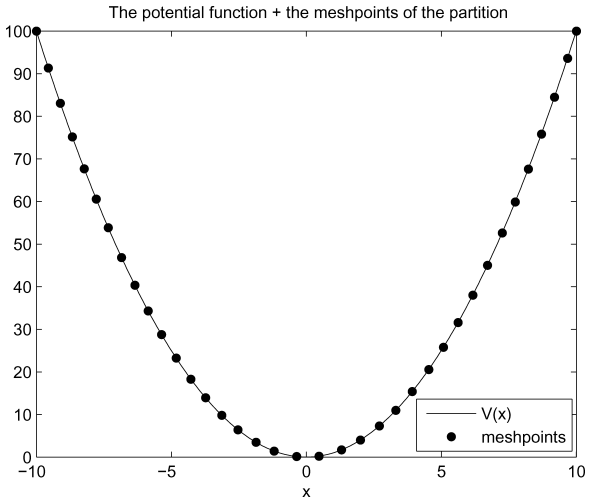


Fig. 3. The partition of the harmonic oscillator on $[-10, 10]$ and tol $= 10^{-10}$.

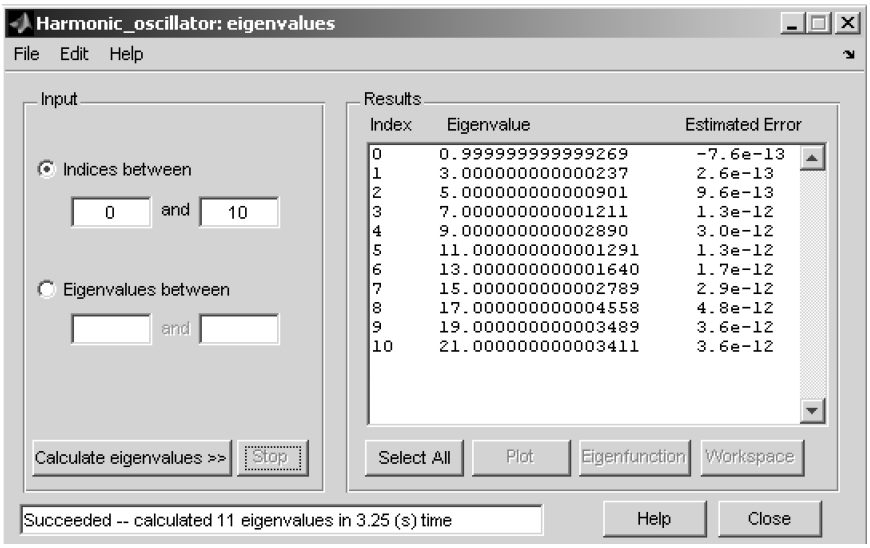


Fig. 4. Example of the eigenvalues window of MATSLISE.

as k is even or odd. Hence, the eigenvalues can be obtained by solving the given equation, but on the interval $[0, b]$, with the given boundary condition at b and with

$$\begin{cases} y'(0) = 0 & \text{to get the even eigenvalues,} \\ y(0) = 0 & \text{to get the odd eigenvalues.} \end{cases}$$

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 $\sqrt{2}$. For symmetric double well problems, this reduction may

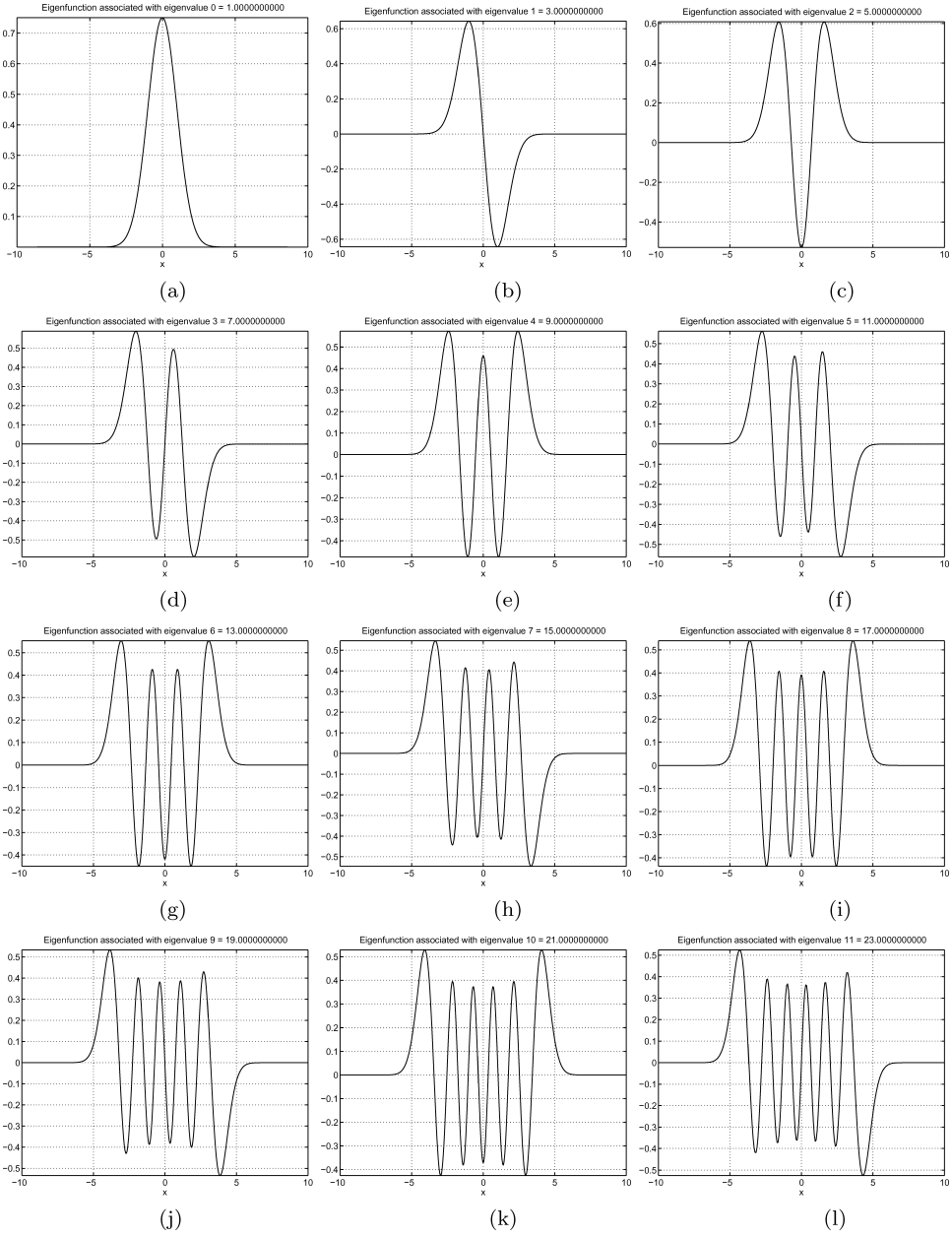


Fig. 5. The eigenfunctions of the harmonic oscillator obtained in MATSLISE.

make the difference between a highly ill-conditioned problem and a perfectly straightforward one. The problem in `Close_eigenvalues.mat` in the `predefined_problems` directory, where

$$V(x) = x^4 - 25x^2, \quad x \in (-\infty, \infty), \quad y(-\infty) = y(\infty) = 0, \tag{44}$$

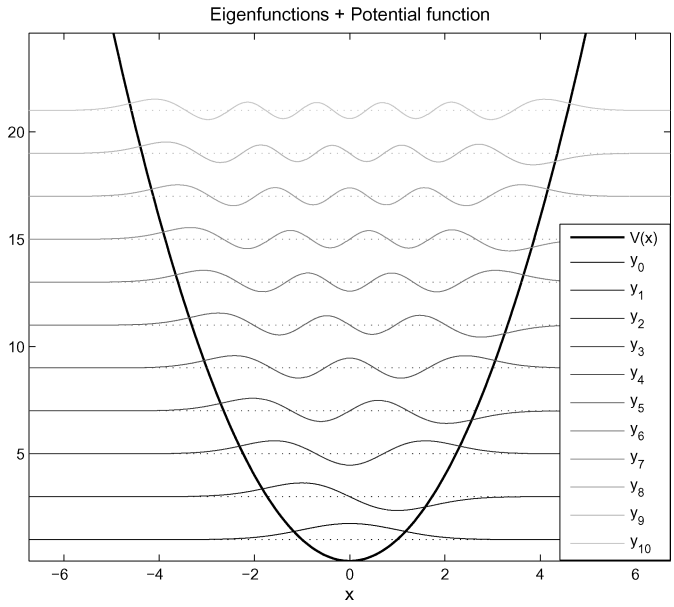


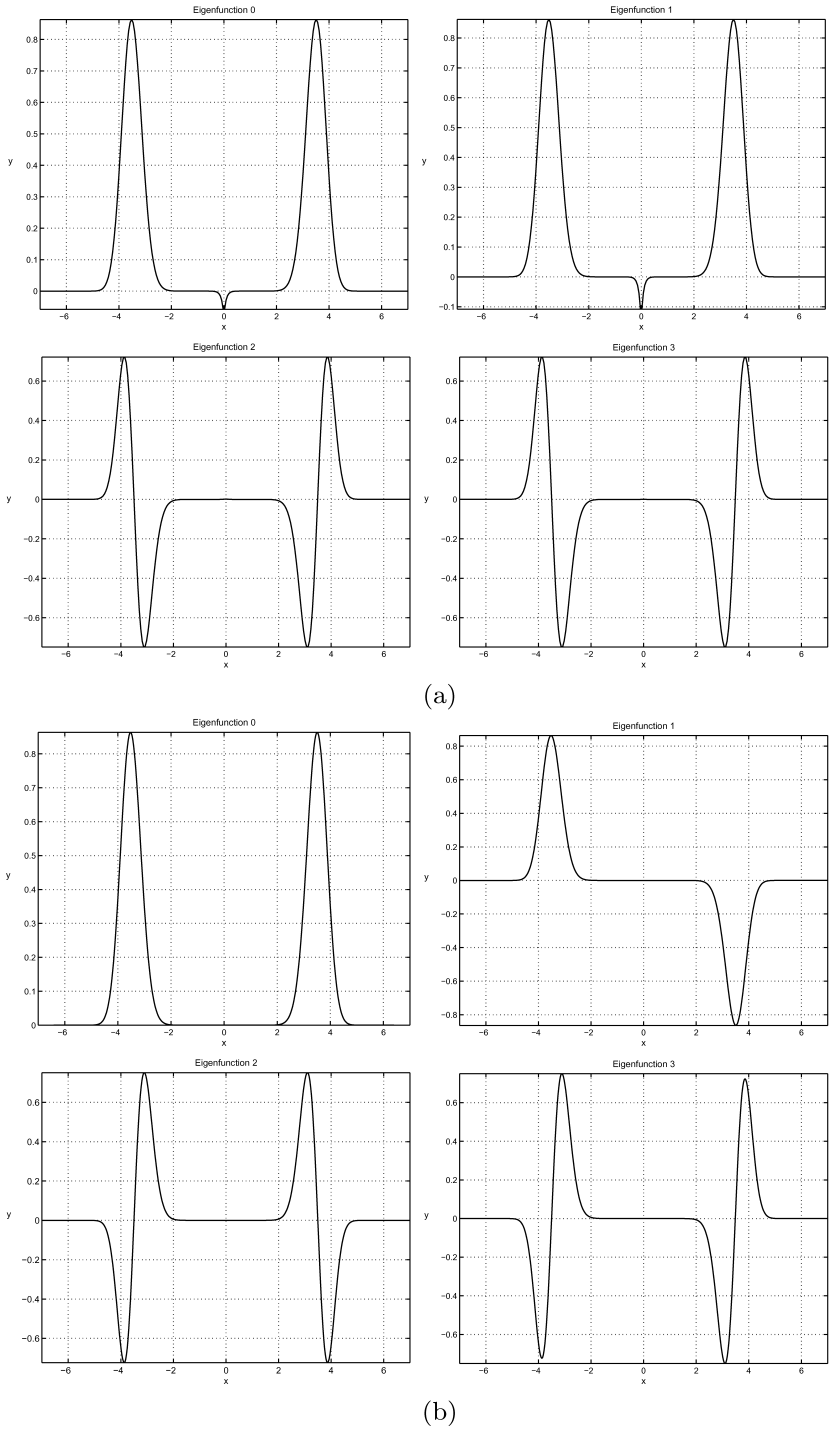
Fig. 6. Energy levels and eigenfunctions of the harmonic oscillator.

is an example of such a symmetric problem. The lowest eigenvalues occur in very close pairs. Figure 7(a) shows the first four eigenfunctions of the close eigenvalues problem, calculated in the GUI version of MATSLISE with tol $= 10^{-12}$ but without half-range reduction. Figure 7(b) shows the same eigenfunctions but now calculated with half-range reduction. It is clear that these last eigenfunctions are the correct ones.

It is possible to use a parameter (or parameters) in the specification of the problem. The parameter-name(s) and -value(s) are defined by checking the "Parameter" box and filling in the two corresponding fields in the input window (Figure 2). The parameter can then be used in the other input-fields. This is used to facilitate the input process or to replace rather lengthy subexpressions in the potential function by a parameter, but also to study the behaviour of the eigenvalue(s) or solution when the parameter changes. The directory `predefined_problems` contains some examples of the use of parameters in the problem specification. One such an example is the problem in `parameter_example3.mat`. It is a regular Schrödinger equation: the Coffey-Evans equation with

$$V(x) = -2B\cos(2x) + B^2\sin(2x)^2, \quad y(-\pi/2) = y(\pi/2) = 0. \tag{45}$$

The parameter name is B and as parameter values we take `10:5:30`, that is, the values in $[10, 15, 20, 25, 30]$. The potential $V(x)$ changes with the parameter values as shown in Figure 8. Note that the problem is symmetric and half-range reduction can be applied. We take tol $= 10^{-10}$ and solve the problem using the GUI. MATSLISE automatically generates a `cpm16_14`-object for each parameter value and all these `cpm16_14`-objects are used when the



(a)

(b)

Fig. 7. The eigenfunctions of Equation (44), (a) calculated on the full range, (b) on the half-range.

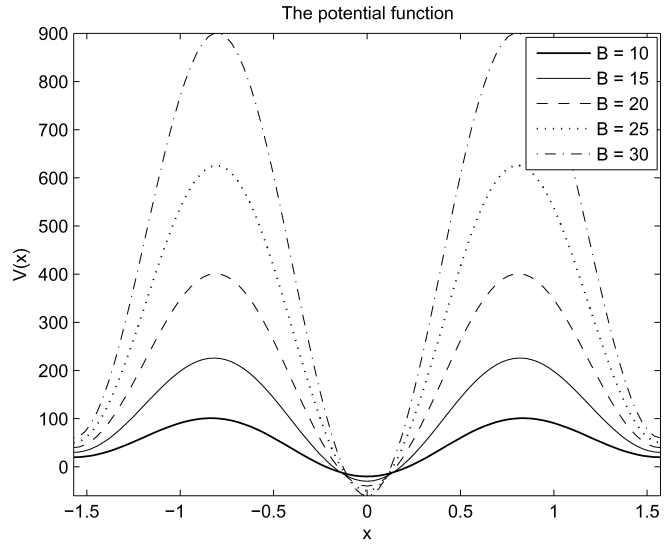


Fig. 8. Solving Equation (45) in MATSLISE: the potential function.

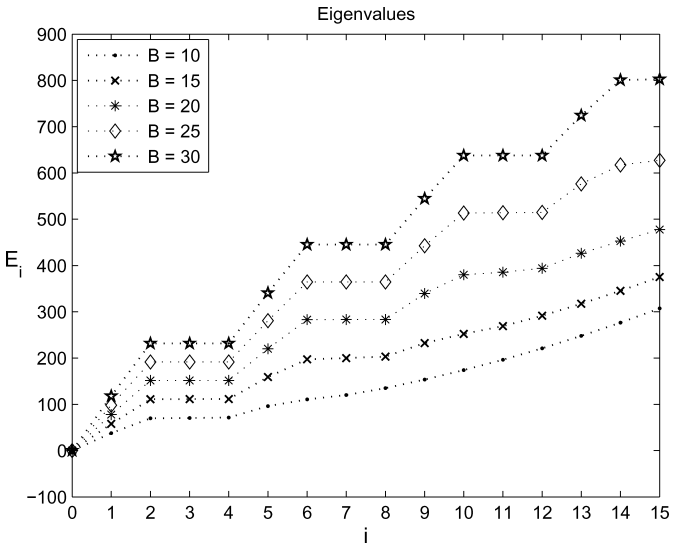


Fig. 9. Solving Equation (45) in MATSLISE: a plot of the eigenvalues.

eigenvalues are calculated. The eigenvalues of the different problems are easily compared by plotting them: Figure 9 contains the plot of the first sixteen eigenvalues of Equation (45). The lower eigenvalues are clustered in groups of three with an isolated eigenvalue between clusters. Increasing B makes more clusters appear and makes each one tighter. After calculating the eigenvalues, the eigenfunction corresponding to a certain eigenvalue index can be computed for each parameter value. In Figure 10 the result is shown for eigenfunction 2 of Equation (45).

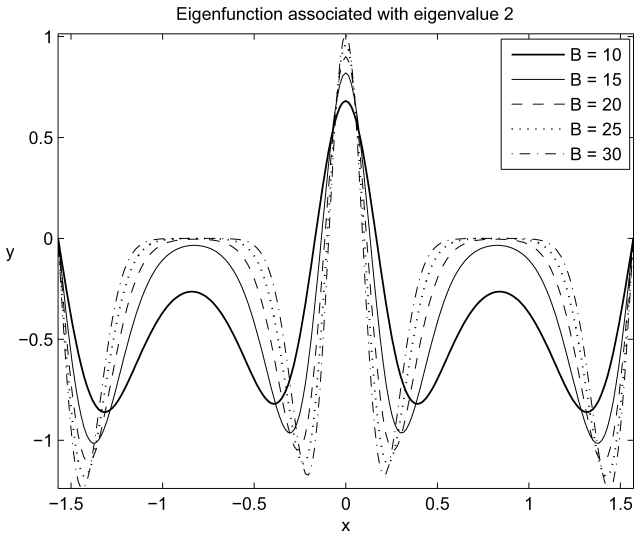


Fig. 10. Solving Equation (45) in MATSLISE: eigenfunction 2.

More information on the features of MATSLISE and on the use of the GUI in general is available in the Help files.

APPENDIX

Functions $\xi(Z)$, $\eta_0(Z)$, $\eta_1(Z)$, ..., originally introduced in Ixaru [1984] (they are denoted there as $\bar{\xi}(Z)$, $\bar{\eta}_0(Z)$, $\bar{\eta}_1(Z)$, ...), are defined as follows:

$$\xi(Z) = \begin{cases} \cos(|Z|^{1/2}), & \text{if } Z \leq 0\,, \\ \cosh(Z^{1/2}), & \text{if } Z > 0\,, \end{cases} \tag{A.1}$$

$$\eta_0(Z) = \begin{cases} \sin(|Z|^{1/2})/|Z|^{1/2}, & \text{if } Z < 0\,, \\ 1, & \text{if } Z = 0\,, \\ \sinh(Z^{1/2})/Z^{1/2}, & \text{if } Z > 0. \end{cases} \tag{A.2}$$

$\eta_1(Z)$, $\eta_2(Z)$, ..., are constructed by recurrence:

$$\eta_1(Z) \;=\; [\xi(Z) - \eta_0(Z)]/Z\,, \tag{A.3}$$

$$\eta_s \;=\; [\eta_{m-2}(Z) - (2s-1)\eta_{s-1}(Z)]/Z\,, \quad s = 2, 3, \ldots. \tag{A.4}$$

These functions obey the following properties:
(i) Power series:

$$\eta_s = 2^s \sum_{q=0}^{\infty} g_{sq} Z^q/(2q + 2s + 1)! \tag{A.5}$$

with

$$g_{sq} = \begin{cases} 1, & \text{if } s = 0, \\ (q+1)(q+2)\ldots(q+s), & \text{if } s > 0. \end{cases} \tag{A.6}$$

(ii) Differentiation with respect to Z:

$$\xi'(Z) = \frac{1}{2}\eta_0(Z), \quad \eta_s'(Z) = \frac{1}{2}\eta_{s+1}(Z), \ s = 0, 1, 2, \ldots. \tag{A.7}$$

One can write [Ixaru et al. 2000]

$$\eta_s(-z^2) = (\pi/2)^{1/2} z^{-(q+1/2)} J_{s+1/2}(z), \quad s = 0, 1, 2, \ldots, \tag{A.8}$$

where $J_{s+1/2}$ is a spherical Bessel function. In contrast to the η functions, the spherical Bessel functions are defined for all real values of s Abramowitz and Stegun [1965].

ACKNOWLEDGMENTS

The authors would like to thank Professor Liviu Ixaru (Romania) for helpful comments and suggestions and for making available some unpublished Fortran code. They would also like to express their gratitude to two anonymous referees whose detailed suggestions led to substantial improvements over previous versions of this article.

REFERENCES

Abramowitz, M. and Stegun, I. A. 1965. *Handbook of Mathematical Functions*. Dover, New York, NY.

Andrew, A. L., and Paine, J. W. 1985. Correction of Numerov's eigenvalue estimates. *Numer. Math. 47*, 289–300.

Bailey, P. B., Garbow, B. S., Kaper, H. G., and Zettl, A. 1991. Eigenvalue and eigenfunction computations for Sturm-Liouville problems. *ACM Trans. Math. Softw. 17*, 491–499.

Bailey, P. B., Gordon, M. K., and Shampine, L. F. 1978. Automatic solution of the Sturm-Liouville problem. *ACM Trans. Math. Softw. 4*, 193–207.

Ixaru, L. Gr. 1984. *Numerical Methods for Differential Equations and Applications*. Reidel, Dordrecht, The Netherlands/Boston, Massachusetts/Laucaster, U.K.

Ixaru, L. Gr., De Meyer, H., and Vanden Berghe, G. 1997. CP methods for the Schrödinger equation, revisited. *J. Comput. Appl. Math. 88*, 289–314.

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

Ixaru, L. Gr., De Meyer, H., and Vanden Berghe, G. 1999. SLCPM12—a program for solving regular Sturm-Liouville problems. *Comp. Phys. Comm. 118*, 259–277.

Ixaru, L. Gr., De Meyer, H., and Vanden Berghe, G. 2000. Highly accurate eigenvalues for the distorted Coulomb potential. *Phys. Rev. E 61*, 3151–3159.

Ixaru, L. Gr. 2002. LILIX—a package for the solution of the coupled channel Schrödinger equation. *Comput. Phys. Commun. 147*, 834–852.

Ledoux, V., Van Daele, M., and Vanden Berghe, G. 2004. CP methods of higher order for Sturm-Liouville and Schrödinger equations. *Comput. Phys. Commun. 162*, 151–165.

Paine, J. W., de Hoog, F. R., and Anderssen, R. S. 1981. On the correction of finite difference eigenvalue approximations for Sturm-Liouville problems. *Comput. 26*, 123–139.

Pryce, J. D. 1993. *Numerical Solution of Sturm-Liouville Problems*. Clarendon Press, Oxford, U.K.

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

Pryce, J. D. and Marletta M. 1992. Automatic solution of Sturm-Liouville problems using the Pruess method. *J. Comput. Appl. Math. 39*, 57–78.

Pruess, S. and Fulton, C. T. 1993. Mathematical software for Sturm-Liouville problems. *ACM Trans. Math. Softw. 19*, 360–376.

Received June 2004; revised January 2005, April 2005; accepted April 2005