Program for precise solution of Schrödinger equation by the QLM method

R. Krivec[a,*]

[a] J. Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

Abstract

This paper presents a code for solving the Schrödinger equation for non singular as well as very singular potentials using a revised and generalized quasilinearization method (QLM). Due to availability of alternative methods quantum mechanics is a suitable test bed for improving the numerical aspects of OLM. QLM is applied by rewriting the Schrödinger equation as a Riccati differential equation for the inverse of the logarithmic derivative of the radial wave function. QLM is also an extremely efficient resummation of the WKB series, and the first QLM iteration is an exact solution of some problems. The mathematical conditions on the convergence of QLM in the functional space in physics have been generalized beyond the original Bellman-Kalaba proposal, in particular regarding the absence of a guarantee of quadratic convergence in that work. In fact, the WKB initial approximation has been shown by the author to be a sufficient condition for immediate onset of QLM quadratic convergence. This, together with the fact that QLM does not require a small parameter and converges even where the perturbation theory or the $1/N$ expansion fail, enabled the extension of QLM to supersingular potentials.

Keywords: quasilinearization; QLM; quadratic convergence; Schrödinger equation; nonlinear differential equations; supersingular potentials

PROGRAM SUMMARY
Program Title: SEM4
Licensing provisions(please choose one): GPLv3.0
Programming language: Fortran
Nature of problem: QLM is a quasilinearization method for problems in physics, generalized [1] to work for regular and singular potentials alike, and guarantee [2] immediate quadratic, possibly monotonic, convergence, for both energies and wave functions, and does not require expansion in a small parameter. QLM is applied here to the Schrödinger equation rewritten as a Riccati equation for the inverse of the logarithmic derivative of the radial wave function.
Solution method: QLM is implemented as an iterative sequence of ordinary linear differential equations. The differential equations are solved as initial value problems by integration from origin to a matching point and from a large abscissa to the matching point. In the first step the integration parameters are optimized by repeating integration and iteration without performing the matching, then matching is carried out with optimized parameters. All iterations are integrated at the same time [3].
Additional comments including Restrictions and Unusual features: Included are optional filter scripts for selective source conversion to quadruple precision; requires a double-precision NAG Library installation.

References

[1] V. B. Mandelzweig, J. Math. Phys. **40**, 6266 (1999).
[2] R. Krivec and V. B. Mandelzweig, Computer Physics Comm. **174**, 119 (2006).
[3] R. E. Bellman and R. E. Kalaba, *Quasilinearization and Nonlinear Boundary-Value Problems*, Elsevier Publishing Company, New York, 1965.

1. Introduction

Many calculations in physics rely on approximation techniques that lend themselves to efficient numerical implementation. Typically they involve perturbation expansions in small parameters within a region of the parameter space. The fact that many equations of physics are nonlinear or could be rewritten as such brings forward additional, potentially very efficient iterative approximations in functional space.

One such method is quasilinearization, whereby a nonlinear problem is converted to an iterative sequence of linear initial value problems, for which efficient algorithms exist. Quasilinearization does not rely on the smallness of some parameter for convergence. Moreover, the quasilinearization iteration may provide quadratic convergence, i.e., add the same number of significant digits to the result in every iteration. The quasilinearization method was initially developed in connection with the theory of linear programming by Bellman and Kalaba [1, 2] as a generalization of the Newton-Raphson method [3, 4] for systems of nonlinear ordinary and partial differential equations, and subsequently applied to different fields [5]. However, in the early works little attention was paid to the systematic development of convergence criteria. Convergence was proven under restricted conditions of small intervals and bounded, non singular forces. Consequently the method was seldom used in physics [18, 19, 20, 21, 22].

*Corresponding author.
E-mail address: rajmund.krivec@ijs.si

The generalizations required to make quasilinearization useful in a wide range of physical problems have been carried out in the work [6] and extended in Refs. [7, 8]. A different proof of convergence was provided which extended the applicability of the method to realistic forces defined on infinite intervals with singularities at some points. The iterates were shown to converge quadratically past the iteration which first exhibits quadratic convergence, and to often converge monotonically. The proof did not predict at which iteration the convergence would start to be be quadratic. The new method was subsequently implemented computationally in the papers [9, 10, 11, 12, 13, 14, 15, 16, 17]. In particular, in Ref. [11] it was shown that the n-th QLM iteration is a sum of 2^n WKB terms, and that the first QLM iteration provides exact solutions for a variety of quantum mechanical potentials. Finally, it was shown in Ref. [12] that if the Langer WKB solution is used as the initial approximation (zeroth iterate) for the QLM iteration, the quadratic convergence sets in immediately. The guarantee of immediate onset of quadratic convergence opened up more possibilities, in particular in the works [16, 17] on supersingular potentials that exhibit the Klauder phenomenon [23, 24, 25, 26, 27, 28], where a numerical QLM implementation was devised that was able to solve the problem even though a perturbation series does not even exist. This was achieved by employing adaptive integration point distributions, based on the first term of the nonanalytical expansion of the solution near the singularity, as well as the logarithmic representation of large values of potentials, with the result that the same computational scheme could be used for all problems, without resort to variable substitution.

2. Quasilinearization of the Schrödinger equation

The generalization to partial differential equations being straightforward [8], we limit ourselves to quantum mechanical calculations with a central potential $V(r)$ and the associated radial Schrödinger equation

$$-\frac{\hbar^2}{2m}\chi''(r) + \left[V(r) + \frac{l(l+1)\hbar^2}{2mr^2}\right]\chi(r) = E\chi(r) \tag{1}$$

where $\chi(r) = rR(r)$ and R is the radial wave function. With the notation $\hbar = 1$, $\kappa^2 = 2m|E|$, and $U(r) = 2mV(r) + l(l+1)/r^2$, we obtain the differential equations for negative and positive energies

$$\chi''(r) - (\kappa^2 + U(r))\chi(r) = 0, \; E < 0 \tag{2}$$

and

$$\chi''(r) + (\kappa^2 - U(r))\chi(r) = 0, \; E > 0. \tag{3}$$

Limiting ourselves to regular or sufficiently weakly singular potentials for the moment, the boundary condition at the origin is

$$\chi(r) \underset{r\to 0}{\sim} r^{l+1} \tag{4}$$

and the boundary condition at infinity for potentials falling off at large r is

$$\chi(r) \underset{r\to\infty}{\sim} e^{-\kappa r}, \; E < 0, \tag{5}$$

$$\chi(r) \underset{r\to\infty}{\sim} \sin(\kappa r - \frac{\pi l}{2} + \delta_l), \; E > 0. \tag{6}$$

For potentials behaving at large r as $\lambda^2 \ln r/R$ or $\lambda^2 r^p$ with positive R, p and λ the boundary conditions at infinity are

$$\chi(r) \underset{r\to\infty}{\sim} e^{-\lambda \int^r \sqrt{\ln r/R}dr} \tag{7}$$

or

$$\chi(r) \underset{r\to\infty}{\sim} e^{-\frac{2\lambda}{p+2} r^{p/2+1}}. \tag{8}$$

Eq. (8) with r changed to $|r|$ applies at $r = \pm\infty$ in the one dimensional problem for the double-well potential $(r^2 - 16)^2/128$ in Ref. [29] where we look for both the ground and the first excited state solutions.

The inverse of the logarithmic derivative of χ,

$$\phi(x) = \kappa\frac{\chi(r)}{\chi'(r)}, \tag{9}$$

satisfies the Riccati equations

$$\phi'(x) = 1 - (1 + W(x))\phi^2(x), \; E < 0 \tag{10}$$

and

$$\phi'(x) = 1 + (1 - W(x))\phi^2(x), \; E > 0, \tag{11}$$

where $x = \kappa r$ and $W(x) = U(x/\kappa)/\kappa^2$. As $\phi(x)$ has poles at the bound state energies, it is convenient to define [6, 7] a new function $u(x)$

$$\phi(x) = -\tan u(x). \tag{12}$$

satisfying

$$u'(x) = -1 + (2 + W(x))\sin^2 u(x), \; E < 0 \tag{13}$$

and

$$u'(x) = -1 + W(x)\sin^2 u(x), \; E > 0. \tag{14}$$

Similar equations were derived earlier [30, 31, 32, 33, 34, 35, 36, 18, 37]. The boundary conditions for the function $u(x)$ are then

$$u(x) \underset{x\to 0}{\sim} -\frac{x}{l+1}, \tag{15}$$

$$u(x) \underset{x\to\infty}{\sim} \frac{\pi}{4} - n\pi, \; E < 0 \tag{16}$$

and

$$u(x) + x \underset{x\to\infty}{\sim} \frac{\pi l}{2} - \delta_l, E > 0. \tag{17}$$

The boundary conditions corresponding to Eqs. (7) and (8) are derived via those for $\phi(x)$

$$\phi(x) \underset{x\to\infty}{\sim} -\frac{\kappa}{\lambda\sqrt{\ln\frac{x}{\kappa R}}} \to 0, \tag{18}$$

and

$$\phi(x) \underset{x\to\infty}{\sim} -\frac{x^{-p/2}}{\lambda} \to 0, \tag{19}$$

and have the form

$$u(x) \underset{x\to\infty}{\sim} -n\pi. \tag{20}$$

Finally, for the spiked harmonic oscillator

$$V(r) = r^2 + \frac{\lambda}{r^\alpha} \tag{21}$$

as a representative of the supersingular potentials, the boundary condition at the origin is derived in Ref. [16]:

$$u(x) \underset{x\to 0}{\sim} -\frac{\kappa}{\sqrt{\lambda}}\left(\frac{x}{\kappa}\right)^{\alpha/2}. \tag{22}$$

Here and in Eq. (16) n is the number of the excited state with $n = 1$ corresponding to the ground state. The minus sign reflects the fact that in the regions of r where $V(r) < E$ in view of Eqs. (13) and (14) the derivative $u'(x)$ is negative and $u(x)$ is decreasing; since its value at the origin is zero $u(x)$ stays negative.

For scattering length calculation [9], we define a function $a(r)$ with the dimension of length by the relation $\phi(x) = \kappa(r + a(r))$ and substitute it into Eqs. (10) and (11) to obtain

$$a'(r) = -(\kappa^2 + U(r))(r + a(r))^2,\ E < 0 \tag{23}$$

and

$$a'(r) = (\kappa^2 - U(r))(r + a(r))^2,\ E > 0, \tag{24}$$

which are very similar to the Calogero equation

$$a'(r) = -2mV(r)\,(r + a(r))^2, \tag{25}$$

where $a(r)$ has the meaning of the variable s-wave scattering length [18]. These equations are obviously a generalization of the Calogero Eq. (25) for arbitrary values of l and κ and reduce to it when l and κ are equal to zero.

The $(k+1)$-th QLM iteration $u_{k+1}(x)$ [6, 7, 8, 1, 2] is obtained via the ordinary differential equation (ODE) in one variable

$$u'(x) = u'(x) = f(u(x), x),\ u(0) = 0 \tag{26}$$

as

$$\begin{aligned} u'_{k+1}(x) &= f(u_k, x) + (u_{k+1}(x) - u_k(x)) f_u(u_k, x), \\ u_{k+1}(0) &= 0, \end{aligned} \tag{27}$$

where $f_u(u, x) = \partial f(u, x)/\partial u$ is a functional derivative of $f(u(x), x)$, with the solution

$$u_{k+1}(x) = \int_0^x ds(f(u_k(s), s) - f_u(u_k(s), s)u_k(s)) \times \exp\int_s^x dt\, f_u(u_k(t), t). \tag{28}$$

Introducing the functionals $F(u(x), x) \equiv f_u(u(x), x)$ and $G(u(x), x) \equiv f(u(x), x) - u(x) f_u(u(x), x)$, Eqs. (27) and (28) become

$$u'_{k+1}(x) - u_{k+1}(x)F(u_k(x), x) = G(u_k(x), x), \tag{29}$$

and

$$u_{k+1}(x) = \int_0^x ds G(u_k(s), s) \exp\int_s^x dt F(u_k(t), t). \tag{30}$$

$u_k(x),\ k = 0, 1, 2, ...$ converge uniformly and quadratically to $u(x)$ of Eq. (26) once an iteration is a sufficiently close approximation of $u(x)$. It was shown in Ref. [12] that using the Langer WKB solution as the initial QLM guess fulfills this condition, making the QLM numerically predictable. In addition, monotonic convergence, or the difference $u_{k+1}(x) - u_k(x)$ being strictly positive (negative) is obtained for strictly convex (concave) functionals $f(u(x), x)$. The precise conditions for the convergence and monotonicity for realistic physical conditions on infinite intervals with possible singularities are described in Ref. [6]. Also, the QLM energy satisfies the Rayleigh-Ritz variational principle ensuring its quadratic convergence [18].

For $E < 0$ the functionals $f(u(x), x)$, $F(u(x), x)$ and $G(u(x), x)$ follow from Eq. (13):

$$f(u(x), x) = -1 + (2 + W(x))\sin^2 u(x), \tag{31}$$

$$F(u(x), x) = (2 + W(x))\sin 2u(x), \tag{32}$$

$$\begin{aligned} G(u(x), x) &= -1 + (2 + W(x))\ \sin u(x) \\ &\quad \times [\sin u(x) - 2u(x)\cos u(x)]. \end{aligned} \tag{33}$$

For $E > 0$ $W(x)$ is replaced by $2 + W(x)$ as per Eq. (14).

3. Numerical method

We use the differential formulation of QLM of Eqs. (27, 29) rather than the integral formulation of Eqs. (28, 30), as it was realized early in the QLM development [9] that it is faster. The numerical approach is described in most detail in Refs. [14, 16]. Each iteration involves two integrations in stable directions: one from $x = x_i$ to $x = x_m$, where $x_m > x_i$ is the matching point, and one from $x = x_u$, where $x_u > x_m$ is the finite upper limit approximation to infinity, to $x = x_m$.

A complete QLM iteration is typically performed in several passes where each pass is defined by a set of quadrature parameters, e.g. the maximum QLM iteration number k_m, x_u, x_m and N, the number of subdivisions of $[x_i, x_m]$ and $[x_m, x_u]$ (in some cases N differs between the two intervals). For each pass, the parameter that is estimated to have caused the largest error in the latest few results is adjusted in the direction error reduction, thus creating an approximate optimum path in parameter space. Convergence of the solution is observed along this path and not simply based on differences $u_{k_m}(x) - u_{k_m-1}(x)$ for a single k_m [14]. After the norm of the subsequent solution differences is deemed small enough, the energy is calculated by a matching process at the last parameter space point. The quadrature point density is uniform for regular potentials, but in the case of supersingular potentials the program can generate a highly nonlinear point distribution which is very dense in the vicinity of the singularity [16, 17].

The present program uses most of the subprograms of the original program used in the works [9, 10, 11, 12, 13, 14, 15, 16, 17] and referred to as `qlmmr*()`, but differs in the implementation of the QLM iteration, i.e., in the way stored function values of $u_k(x)$ are reused in the quadrature for $u_{k+1}(x)$. The original program used interpolation to obtain intermediate values of the previous iteration for the predictor-corrector integrator. It was found [14] that the minimum total number of quadrature points required for the given accuracy was reached (in 128-bit precision arithmetic mode) when the fourth-order Runge-Kutta method was used as it required only one intermediate point. The latter was provided by a five-point Lagrange interpolation [38] whose estimated error is one order in step h smaller [39] than the estimated solver error. In addition, for supersingular potentials, it was observed [16] that a nonuniform point density was more effective than low-order implicit methods.

Sufficiently accurate interpolation did not seem to appreciably impede the speed of the QLM iteration when using the Runge-Kutta fourth-order order method. Partly this is the consequence of the intrinsic numerical stability against numerical noise, or the self-correcting property of the QLM. The present program was originally written as an independent verification program, avoiding the interpolation step by implementing an old idea [2] of solving all iterations simultaneously in the form of a system of linear ODE. Removing the need for interpolation also makes it possible to use any ODE solver, including faster high order ones (included is an implementation of the Adams predictor-corrector method, `ode`.) Let us note that the approach of setting up a specialized point density is also helpful to the ODE solver. The drawback of the method is that it has to be decided in advance how many iterations are necessary; however the guaranteed quadratic convergence which provides several significant digits in each iteration makes this a sensible approach. In fact, it has been shown [16, 17] that even for supersingular potentials typically only of the order of 10 iterations are required for very precise results (over 20 significant digits). Due to the described method of testing for convergence, one can also let the algorithm increase k_m as required, starting with a small value. Conversely, k_m can also be fixed by setting `iterl` to $-k_m$.

Working in quadruple precision (128-bit) requires compiler support. Some subprograms make use of the NAG library [40], in particular some special function evaluators used in the Langer WKB solution calculation, but only optionally for the ODE integration. The former may be replaced by other procedures, but the program behavior may change unless suitable wrappers are written that mimic the definition intervals and the error detecting and reporting algorithms of the NAG procedures. Such replacement has not yet been successfully completed by the author. However, in the case the code should be run in and is converted to quadruple precision by included scripts, a double-precision NAG library is sufficient, as it only pertains to the calculation of the Langer WKB function. The scripts leave the corresponding code isolated in double-precision and also provide proper variable conversion.

4. Program description

The program package can be unpacked in a directory (installation directory) and should create the following initially partly populated or empty subdirectories: `bin` (run and compilation helper scripts and executables), `inp` (input files and run scripts), `pd` (double-precision sources and their automatically preprocessed (filtered) versions), `pq` (optional automatically generated higher precision sources), `out` (output files), `gnup` (plots and automatically generated plot scripts), `lib, libq` (third-party source packages), and `examples` (top of directory tree containing run, input and output files and plots for examples). Third-party source packages were selected for their ability to be automatically converted to quadruple precision; otherwise they could be replaced by other routines or libraries. The packages are: `ode` (`www.netlib.org/ode`, by Shampine and Gordon), `odepack` (`computation.llnl.gov/casc/odepack`, in public domain) and `spline` (`www.netlib.no/netlib/sfmm`, from Ref. [41]) for double and quadruple precision, respectively. Some of these programs are used optionally.

The program is written in a simple but strict style with indentation and annotations, for readability and easy automatic transformation by included filters using standard Unix shell tools to quadruple precision. There are also pre-filters that insert two main array dimension parameters to the source code, specified in the run script. None of the filters overwrites the original sources. Instead, they write to source files which have a capital-letter string (`NNN` and/or `DP` or `QP`) inserted between the original file root name and extension. The filtered files are input to the compiler in place of their original files. In addition, the quadruple-precision filtered sources are put in a separate directory (`pq`). All alterations should be done exclusively in the original source files (with unmodified names) in the directory `pd`, otherwise they get overwritten. The list of these files is also provided by the run script `bin/sem4_C1.csh` if its second parameter is `list`.

The program has worked with diverse compilers and to the best of our knowledge it should work on most. Helper scripts optionally automatically generate plotting commands for the `gnuplot` program.

The input files contain names of variables to the right of their values for readability. Variable names and other text are ignored by the program, except the first line which is read into a character variable.

The package contains filter scripts that optionally automatically convert source syntax for use with quadruple-precision (128-bit) arithmetic if available. Again, all programming changes in the code are to be performed in the original double precision sources only (directory `pd`), and the quadruple precision modified sources are filtered and placed in a separate directory (`pq`), where any manual changes are overwritten. The scripts may require some adherence to clean code writing and the output sources should be verified against the originals using the `diff` program, or equivalent.

In the following program variables are used interchangeably with the corresponding mathematical symbols, e.g. `nx` and N. Program unit names are distinguished from variable names by appending "`()`".

The main program input and related variables are listed below approximately in the order they appear in the input file. Most of the names are the same across the sources to facilitate searching, and documented where they are used.

`xi, xm, xf` Interval boundaries x_i, x_m, x_u.

`nx` Number N of points on interval $[x_i, x_m]$; if `ixdist` $\neq 0$, the input value is used initially; if `ixdist` $= 0$, `nx` is initially calculated from input value of step `h`. This is used for E-dependent initial approximations which are later interpolated, and does not prescribe the number of ODE steps.

`h` Point separation on $[x_m, x_u]$ and on $[x_i, x_m]$ when `ixdist` $= 0$. If `ixdist` $\neq 0$, `h` can be kept of the order of the last $[x_i, x_m]$ point spacing (near $x = x_m$) independently of `nx`.

`itolr, itola, itolna, irelab` Negative base 10 logarithms of the relative and absolute tolerance pair for ODE solution for the Netlib solver, and the tolerance and relative/absolute error method of NAG solvers. The error method is translated for NAG calls as follows: 1 to 'A', 2 to 'B', any other number to 'M'.

`itolr0, itola0, ixerrp` Negative base 10 logarithms of the relative and absolute tolerances and the upper x limit of interval for x-dependent error control in `seem0()`: relative tolerances increase from $10^{-\texttt{itolr0}}$ to $10^{-\texttt{itolr}}$ on $(0, 10^{-\texttt{ixerrp}}]$ if `ixerrp` $\neq 0$, and similarly for the absolute tolerances.

`iodes` ODE solver type: 1 - normal, 2 - stiff, 3 - Runge-Kutta.

`jt` Jacobian type indicator for Netlib ODE solver routine `dlsoda()`.

`lnag, lnagpd` Use the NAG ODE solver; use the NAG routine that requires one to supply the Jacobian explicitly.

`ixdist` Point distribution type and `nx`, `h` precedence:

- 0 : linear point distribution on LHS, RHS, `h` takes precedence, `nx` (input) is ignored.
- > 0 : nonlinear point distribution on LHS, determined by the value of `ixdist` (which is translated to `xns1` for `qlmmr*()` routines); linear point distribution on RHS; `ixdist` also determines the type of regularization: this behavior is like `xns1` which also encodes both point distribution and regularization.

`ixdist` is modified in special cases to make sure the ODE output routine uses points calculated in advance instead of propagating using `h`. (Propagation algorithm may not be the same as used by the WKB wrapper-called routines.)

`ix` Point counter.

`iter` Maximum QLM iteration k_m.

`iterl` Lowest `iter` in the convergence (parameter space path) loops. Negative value indicates `iter` is fixed.

`inxl` Minimum number of (N, x_u) convergence passes. Need not account for initial passes. A positive value used in cases when Aitken method does not work initially.

`inxu` Maximum number of (N, x_u) convergence passes. Need not account for initial passes.

`fninc, fxinc, iaep` Error optimization in parameter space: N increase (`h` reduction) factor (> 1); x increase value defined like `fxconv` in `qlmmr*()`; negative base 10 logarithm of error tolerance `e0, sem4()` in parameter space.

`iis` Type of initial solution. Cf. `sei0()` (values may be calculated by `sei0()` at separate points as called from the ODE callback routines. Cf. `iinit` (`qlmmr*()`). `iis` = 220: calculate and use the Langer WKB solution as zeroth iteration.

`vpar0` - `vpar6` Potential $V(x)$ parameters. For interpretation see the corresponding subroutine returning values and derivatives of $V(x)$. In particular:

`vpar0` type of potential. Translated to `isel` of the `qlmmr*()` package (normally for WKB only). Selects the subroutine hard coded to evaluate a particular $V(x)$ and derivatives; some $V(x)$ physical parameters may be input variables.

`vpar2` initial value of κ for parameter optimization.

`ngw` The number of iterations to output for plotting. Solutions above the maximum iteration are repeated in order to be able to plot using a general file. `ngw` enables to keep the number of solutions written below nd. In fact it can be 0 as then the actual number of iterations is output.

`ixreg1` Negative base 10 logarithm of `xgreg1`, exponential regularization if no other regularization is present. If 0, this regularization is ignored.

`ixeps` Base 10 logarithm of smallest x separation in nonuniform point distributions (`ixdist` $\neq 0$, `qlmxs*()`).

`z1, z2` Search limits in κ for the QLM E after the QLM iteration.

`iabsxd, iabsfd` One term of the negative base 10 logarithms of zero search abscissa and function tolerances; the other term is `iaep`.

`z1w, z2w` Search limits in κ for the Langer WKB solution E.

`iwkber` Negative base 10 logarithm of Langer WKB solution relative accuracy.

`iwabsx, iwabsf` Negative base 10 logarithms of one set of zero search abscissa and function tolerances for Langer WKB energy zero search; the smaller of `iwabsx` and `iabsxd` + `iaep` is taken, and similarly for `iwabsf`.

`xbcinf` Use interpolated value between initial approximation value and exact boundary condition at infinity as boundary condition at x_u for higher iterations. This is a plausible option because the QLM iterations do not approach the exact solution very fast like they do in `qlmmr∗()`.

`lignrh` Ignore RHS solution (even if calculated) and use the solution value at $x = x_m$ as convergence criterion. `lignrh` is set by $x_m < 0$.

`lsplin` Use cubic spline interpolation of initial solution. `lumidp` overrides `lsplin` if `iodes` = 3.

`lumidp` Use precalculated midpoint initial solution values (4th order Runge-Kutta only). `lumidp` overrides `lsplin` if `iodes` = 3.

`lzero` E search after parameter optimization.

`ipoptv` Select convergence variable to use in parameter optimization:

- **0, 1** : highest iteration solution difference at matching point;
- **2** : average square solution difference at matching point;
- **3** : value of LHS solution at matching point at highest iteration;
- **4** : value of RHS solution at matching point at highest iteration.

`lpopta` Use absolute value of convergence function selected by `ipoptv`.

`lnoopt` Skip parameter optimization (cf. `lzero`).

`laitkn` Use Aitken scheme instead of simple error evaluation (difference of last two values similar to `qlmmr∗()`) for parameter optimization.

`literx` Bail out if on `iter` increase errors increase.

`fmzcc` $F_m > 1$, coefficient of minorizing function at small x for supersingular potentials [17].

`ixi` State number (usually principal quantum number).

`ips` Step for point output, typically 100.

`ixoff` Base 10 logarithm of the multiplier of the machine precision for some error tolerances for Langer WKB solution in `wkbch9()`; typically 2.

`zprecp` Tolerance in radians for detecting which state the solution belongs to (cf. `wkbch9()`).

`xewkb` E for Langer WKB function if it is not requested to be calculated by zero search, i.e., if `lewkb = .false.`.

`lewkb` Calculate E for Langer WKB function by zero search.

`it999` Print additional output, according to a nonzero value.

During the initialization stage, for supersingular potentials for which an E-independent initial approximation is employed [16], the main program calculates the integration point distribution by calling the subroutine `qlmxs5()`. If the Langer WKB solution is employed, this task is performed in the next stage by `qlmxs7()` which is called from the Langer WKB subroutine.

In the next stage, the parameter optimization loop starts for an approximate value of E given by the input parameter `vpar2`. In each loop pass, the subroutine `semm4()` performs a complete QLM iteration for up to iteration `iter`, calling `wkb9d0()` to calculate the Langer WKB solution and `seqlm4()` twice for each iteration to perform the LHS and RHS integrations towards the matching point (no matching is performed at this stage). `seqlm4()` contains the selector of integration methods corresponding the the input value of `iodes`, in each case except the Runge-Kutta case offering the choice between a NAG subroutine or an alternative (parameter `lnag`), the latter being invoked via wrappers `nlode1()` and `odep1()`.

In the third, optional pass (flag `lzero`), the QLM E value is determined by the zero search for the LHS and RHS final QLM iteration mismatch, calling the zero-search subroutine `zli3a()` with the subroutine `semm4()` as function evaluator. The parameters optimized in the previous stage for the approximate input E value are used.

Selected auxiliary subroutines are as follows.

`sede4()` calculates the right hand side of Eqs. (13) and (14). It also performs regularization for supersingular potentials as described in Refs. [16, 17].

Values of $V(x)$ and optionally its derivatives for a class of potentials are provided by writing simple dedicated subroutines that accept $V(x)$ coefficients (currently up to five, `vpar1` and `vpar3` - `vpar6`, and optionally return results in the logarithmic computer floating point representation. Several such subroutines are included: the $V(r)$ parameters are set up by `qlmx0c()`, based on `isel` translated from input parameter `vpar0`, $V(r)$ and $V'(r)$ are calculated by `qlmvaa()` and the $V''(r)$ by `qlmv2a()`. These three subroutines need to be updated to include new potentials $V(r)$.

5. Examples

The example inputs in the beginning of the file `inp/sem4_C1.i` calculate low-precision solutions for the harmonic oscillator and the Woods-Saxon potentials of Ref. [12, 17]. The file contains sample inputs for other systems which are activated by moving them to the beginning of the file.

The directory `examples` has the same structure as the installation directory and contains ASCII text output files as well as plots obtained by the author for the test examples, for direct comparison with test runs.

To run the examples, the run script `bin/sem4_C1.csh` in the installation directory should be modified to reflect the compiler and library locations, as well as the main dimension parameters for automatic source filters. The input file `inp/sem4_C1.i`

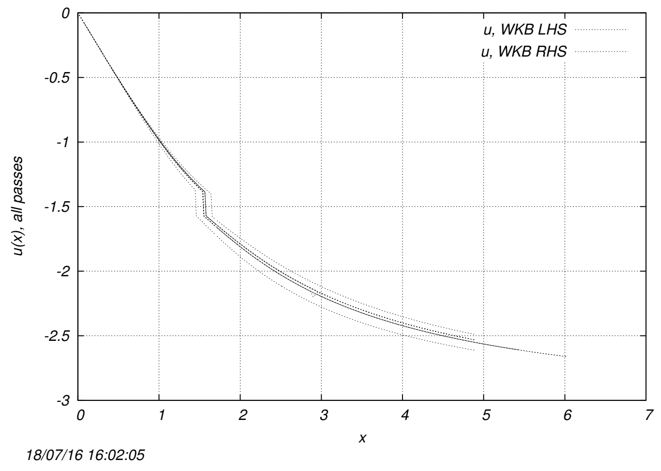

Figure 1: The parameter space optimization passes as illustrated by the variation of the initial Langer WKB approximation in the harmonic oscillator test run.

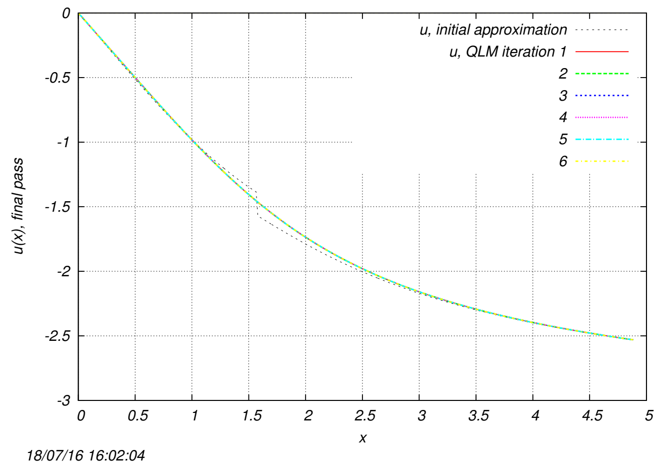

Figure 2: The initial Langer WKB approximation and the QLM iterations 1 – 6 for the inverse of logarithmic derivative of the radial wave function of the harmonic oscillator test run.

should be edited by moving the desired input block to the beginning of the file. Then the run script can be run from any directory, for example as `../bin/sem4_C1.csh D 2 TEST 6`, as it recognizes its own location. This command will leave the sources in double precision, compile the program with optimization level 2 (`-O2`), add the string `TEST` to executable name, create a unique output file root name including a unique number at end if `xunum` is set to 1, and run the program. To avoid recompilation, the program can be rerun as `../bin/sem4_C1.csh D -2 TEST 0` to use the previously compiled executable. The script will also display plots up to the number of iterations specified by the 4th argument (6), and show most interesting lines from the output file in the terminal. (If `xunum` = `1` and the run script breaks, and if it fails to remove the numbering lock file `bin/sem4_C1_lck`, that file can be removed from another terminal before or during a repeat run, while the script is waiting.) The original input and output files are saved in the `examples` directory.

Figs. 1 and 2 are a subset of figures in `examples/gnup`. Fig. 1 shows the initial parameter optimization by the program; visible is the upper integration limit adjustment to minimize the highest iteration LHS/RHS solution difference (`ipoptv` = 0). In Fig. 2 the initial approximation is discontinuous due to the discontinuity of the derivative of the Langer WKB function; see the discussion in Ref. [12]. However, already the first QLM iteration is continuous and indistinguishable from higher iterations.

Acknowledgment. I am grateful to Victor B. Mandelzweig of the Hebrew University, Jerusalem, Israel for the fruitful long-term collaboration during which the code described in this paper was written.

References

[1] R. Kalaba, J. Math. Mech. **8**, 519 (1959).

[2] R. E. Bellman and R. E. Kalaba, *Quasilinearization and Nonlinear Boundary-Value Problems*, Elsevier Publishing Company, New York, 1965.

[3] S. D. Conte and C. de Boor, *Elementary numerical analysis*, McGraw Hill International Editions, 1981.

[4] A. Ralston and P. Rabinowicz, *A first course in numerical analysis*, McGraw Hill International Editions, 1988.

[5] V. Lakshmikantham and A. S. Vatsala, *Generalized Quasilinearization for Nonlinear Problems*, MATHEMATICS AND ITS APPLICATIONS, Volume 440, Kluwer Academic Publishers, Dordrecht, 1998.

[6] V. B. Mandelzweig, J. Math. Phys. **40**, 6266 (1999).

[7] R. Krivec and V. B. Mandelzweig, Computer Physics Comm. **138**, 69 (2001).

[8] V. B. Mandelzweig and F. Tabakin, Computer Physics Comm. **141**, 268 (2001).

[9] R. Krivec and V. B. Mandelzweig, Computer Physics Comm. **152**, 165 (2003).

[10] R. Krivec, V. B. Mandelzweig and F. Tabakin, Few-Body Systems **34**, 57 (2004).

[11] R. Krivec and V. B. Mandelzweig, Phys. Lett. **A337**, 354 (2005).

[12] R. Krivec and V. B. Mandelzweig, Computer Physics Comm. **174**, 119 (2006).

[13] E. Z. Liverts, R. Krivec and V. B. Mandelzweig, Phys. Scripta **77**, 045004 (2008).

[14] R. Krivec and V. B. Mandelzweig, Computer Physics Comm. **179**, 865 (2008).

[15] E. Z. Liverts, E. G. Drukarev, R. Krivec and V. B. Mandelzweig, Few-Body Systems **44**, 367 (2008).

[16] R. Krivec, Computer Physics Comm. **183**, 2601-2607 (2012).
[17] R. Krivec, Applied Mathematics and Computation **246**, 440-446 (2014).
[18] F. Calogero, *Variable Phase Approach to Potential Scattering*, Academic Press, New York, 1965.
[19] A. A. Adrianov, M. I. Ioffe and F. Cannata, Modern Phys. Lett. **11**, 1417 (1996).
[20] M. Jameel, J. Physics A: Math. Gen. **21**, 1719 (1988).
[21] K. Raghunathan and R. Vasudevan, J. Physics A: Math. Gen. **20**, 839 (1987).
[22] M. A. Hooshyar and M. Razavy, Nuovo Cimento B75, **65** (1983).
[23] L.C. Detwiler and J.R. Klauder, Phys. Rev. **D 11** (1975) 1436.
[24] J.R. Klauder, Acta Phys. Austriaca, Suppl. **11** (1973) 341.
[25] J.R. Klauder, Phys. Lett. **B 47** (1973) 523.
[26] J.R. Klauder, Science **199** (1978) 735.
[27] H. Ezawa, J.R. Klauder, and L.A. Shepp, J. Math. Phys. **16** (1975) (1975) 783.
[28] B. Simon, J. Funct. Anal. 14 (1973) 295.
[29] N. E. Jannik Bjerrum-Bohr, J. Math. Phys. **41**, 2515 (2000)
[30] J. F. Drukarev, JETP **18**, 247 (1949).
[31] O. Bergmann, Acta Physica Austriaca **4**, 62 (1950).
[32] P. O. Olsson, Ark. Fys. **4**, 217 (1952).
[33] I. Kynch, Proc. Phys. Soc. **A65**, 83, 94 (1952).
[34] S. Franchetti, Nuovo Cimento **6**, 601 (1957).
[35] L. Spruch, in "Lectures in Theoretical Physics", Vol. 4, eds. W. E. Brittin, B. Downs and I. Downs, Intern. Publ. Inc., N.Y. (1962).
[36] R. F. Dashen, Nuovo Cimento 28, 229 (1963); J. Math. Phys. 4, 338 (1963).
[37] V. V. Babikov, Sov. Phys. Uspekhi **10**, 271 (1967).
[38] M. Abramovitz, I. Stegun, Handbook of Mathematical Functions, Dover, New York, 1972, p. 879.
[39] E. Isaacson, H.B. Keller, Analysis of Numerical Methods, J. Wiley & Sons, New York, 1966.
[40] NAG Program Library Manual Mark 17, NAG Ltd., Oxford 1995.
[41] George Elmer Forsythe, Michael A. Malcolm and Cleve B. Moler, Computer Methods for Mathematical Computations, Prentice Hall, Upper Saddle River, 1977.