JOURNAL OF COMPUTATIONAL PHYSICS 112, 234–246 (1994)

Efficient, Reliable Computation of Resonances of the One-Dimensional Schrödinger Equation*

JOHN D. PRYCE

Software Engineering Group, Royal Military College of Science, Cranfield, Shrivenham, Swindon SN6 8LA, United Kingdom

Received January 15, 1992; revised November 16, 1992

We present a numerical method, implemented in a Fortran code RESON, for computing resonances of the radial one-dimensional Schrödinger equation, for an underlying potential that decays sufficiently fast at infinity. The basic approach is to maximize the time-delay function $\tau(\lambda)$ as in the LeRoy program TDELAY. We present some theory that allows a preliminary bracketing of the resonance and various ways of reducing the total work. Together with automatic meshsize selection this leads to a method that has proved efficient, robust, and extremely trouble-free in numerical tests. The code makes use of Marletta's Sturm–Liouville solver, SL02F, due to go into the NAG library.

1. INTRODUCTION

Quantum-mechanical resonances, or quasi-bound states, are a phenomenon displayed by certain types of potential function. In the context of electromagnetic radiation from excited atoms they are observable spectroscopically as generating bright (emission) or dark (absorption) bands which are broader than the sharp lines shown by bound-state energy transitions. There is a gradation from narrow, distinct resonances to broad resonances that are hard to distinguish from the background. J. R. Taylor's book "scattering Theory" [1] gives an introduction to the physical effects of resonances; see also the bibliography by Leroy *et al.* [2, 3] for a survey of their effects and importance in chemical reaction kinetics. They are also significant in some technological areas such as wave-guide design.

This paper describes a new algorithm for computing resonances of the one-dimensional Schrödinger equation—modelling, for instance, the interaction of an electron with a spherically symmetric nucleus. The method is based on the *time-delay* definition of a resonance, as is R. J. LeRoy's well-known TDELAY program [4], but it incorporates methods which make it more robust and easier to use than the latter on typical problems. A good description of the physical content of the time-delay definition is given in Taylor [1, Chap. 13]. For a mathematically rigorous treatment of scattering theory see Reed and Simon's "Methods of Mathematical Physics" [5, Vol. III]. Formula (19) for the time delay seems to be due to F. T. Smith [6].

The material is organized as follows. Section 2 defines the notation and describes informally what a resonance is. Section 3 gives the necessary theory of the time-delay definition. Section 4 describes the theory underlying the parts of the method that appear to be new. Section 5 discusses algorithm design, outlines the overall algorithm that results, and the user interface of the code. Section 6 gives numerical results and comparisons. Section 7 summarizes the results and gives pointers for further work.

2. NOTATION AND INFORMAL DESCRIPTION

We treat the *radial Schrödinger equation*,

$$-u'' + q(x)u = \lambda u \qquad \text{on} \quad 0 < x < \infty, \tag{1}$$

where x is the distance from a spherically symmetric nucleus, i.e., the radius r in spherical polar coordinates; $q(x)$ is actually of the form

$$q(x) = q_0(x) + l(l+1)/x^2 \tag{2}$$

and l is a constant arising out of the method of separation of variables applied to the three-dimensional Schrödinger equation. Physically, l is called an orbital rotational quantum number, and the term $l(l+1)/x^2$ is the "centrifugal component" of the *effective potential* $q(x)$. The *underlying potential* $q_0(x)$ tends to a limit, the *dissociation energy*, at infinity, which without loss of generality we can take as zero.

Depending on $q(x)$ the system may be *limit-point* or *limit-circle* at $x = 0$ in the standard Weyl classification: in fact, barring exceptional $q_0(x)$, it is limit-point if $l \geqslant \frac{1}{2}$, the commonest case. Like most codes that handle singular points automatically, ours always seeks the solution that is "small"

* This work is supported by NAG Ltd. of Oxford and by the SERC.

0021-9991/94 $6.00

at $x=0$, since this is usually the physically meaningful one. In the limit-circle (or regular) case this corresponds to the so-called Friedrichs boundary condition.

Potentials possessing *bound states* necessarily have a *potential well*—a region of the x-axis where q is below the dissociation energy, i.e., $q<0$. Potentials possessing *resonances* necessarily have a *potential barrier*—a local maximum q_{barr} at $x=x_{\text{barr}}$ of $q(x)$ which lies above the dissociation energy. Usually there is also a well to the left of the barrier, but this is not necessary. Resonances occur for energies between the dissociation energy and the barrier maximum, that is, for $0<\lambda\leqslant q_{\text{barr}}$. Weaker resonances can also occur for λ slightly above the barrier.

Most resonating potentials that have been studied numerically have one well and one barrier. The interactions between multiple barriers can be exceedingly complex and make reliable software hard to construct. Therefore we concentrate on the one-barrier case, where it is assumed that $q(x)$ has a strict local maximum $(x_{\text{barr}}, q_{\text{barr}})$ with $x_{\text{barr}}>0$ and $q_{\text{barr}}>0$, called the barrier top; that any local maxima of q to the left of x_{barr} are less than zero; and that q decays monotonely to zero on $x_{\text{barr}}<x<\infty$. That said, our code has features to make it possible to handle problems with more than one barrier, as described later.

Informally, resonances are like eigenvalues in the following way. Solutions of (1) are, broadly speaking, of oscillating or of exponential form according as $q(x)-\lambda$ is less than or greater than zero. For any λ there is a unique solution, up to a scalar factor, of (1) subject to the BC at zero—call it $u=u(x;\lambda)$. An eigenvalue is a $\lambda<0$ for which $u(x;\lambda)$ just hits exponentially decaying rather than growing behaviour as $x\to\infty$. A resonance band is a usually small range of $\lambda>0$ for which $u(x;\lambda)$ decays rather than grows within the barrier so that when it reaches its ultimate oscillating behaviour for large x, the ratio of its "size near zero" to its "size at infinity" is large, whereas for most values of $\lambda>0$ this ratio is small. The "centre" of the resonance is the λ which maximizes some measure of this ratio.

The *time-delay* definition is based on the "drift" of phase of the oscillation at infinity as λ varies. Asymptotically, for $\lambda>0$ and large x, the solution $u(x;\lambda)$ is approximated by

$$u=r\sin(x\sqrt{\lambda}+\delta), \tag{3}$$

where r is the limiting amplitude (arbitrary) and $\delta=\delta(\lambda)$ is the limiting phase, which is determined uniquely by the differential equation and the BC at zero. A resonance is defined to be a local maximum of the derivative

$$\tau(\lambda)=\delta'(\lambda), \tag{4}$$

that is, a point of inflection of $\delta(\lambda)$. To see informally why this is related to amplitudes, note that when λ is at a resonance, so that the solution is decaying exponentially through a barrier, then the solution at the end of the barrier is extremely sensitive to small changes in the value of u or u' at the beginning of the barrier. In fact, a small increase in λ can make the solution go from exponentially increasing positive to exponentially increasing negative at the end of the barrier. This "flip" is associated with a rapid increase of δ by approximately π, superposed on a generally slowly varying trend, and hence with a "spike" in $\tau(\lambda)$.

Smith has shown by physical arguments [6] that $\tau(\lambda)$ is the expected delay of an electron coming in from a large distance to the neighbourhood of the nucleus and then escaping again, compared with the time this would take in the absence of the nucleus.

There are other definitions used in numerical work. The *internal amplitude* definition of a resonance, briefly covered by [2, p. 5116] and going back to Rice in 1929–1930 [7], comes from maximizing a straightforward interpretation of the "size near zero: size at ∞" ratio. Since the early days, quantum physicists have made the physically attractive interpretation that while a bound state corresponds to a real eigenvalue, a resonance corresponds to a complex eigenvalue with small imaginary part v and is therefore a metastable or "nearly bound" state, the size of v giving the rate of decay out of this state and being inversely related to the resonance sharpness. Taylor [1] explains the relation of resonances to poles of the *scattering operator* S as an analytic function of the complex momentum p or equivalently the energy $E=p^2/2m$, which puts the above interpretation on a firm footing.

The definition used by Hehenberger, Brändas *et al.* [8–10], based on properties of the Weyl–Titchmarsh $m(\lambda)$ function, is mathematically very interesting and closely related to the S operator interpretation. This is pursued in more detail by Brändas *et al.* in [11–14]. In particular [13] gives methods of proving the existence of a resonance within a certain region of the complex plane by a version of the Gerschgorin circle theorem, see, e.g., [15]. The different definitions of a resonance are not quite equivalent, although for "sharp" resonances they give very nearly the same value for the centre of the resonance.

3. THEORY OF THE TIME-DELAY

As above, let $u(x;\lambda)$ be for each λ a solution of (1) satisfying the BC at zero, plus some normalizing condition that ensures u depends smoothly on λ. This can always be done in various ways. To define $\delta(\lambda)$ and $\tau(\lambda)$ precisely, introduce *Prüfer variables* $r(x;\lambda)$ and $\theta(x;\lambda)$ (as in Hehenberger [16]) by

$$u'=\lambda^{1/4}r\cos\theta, \qquad u=\lambda^{-1/4}r\sin\theta. \tag{5}$$

This converts (1) to the two first-order equations

$$\theta' = \sqrt{\lambda} - \frac{q(x)}{\sqrt{\lambda}} \sin^2 \theta \tag{6}$$

$$(\log r)' = \frac{q(x)}{\sqrt{\lambda}} \sin \theta \cos \theta. \tag{7}$$

Then define

$$\delta(x; \lambda) = \theta(x; \lambda) - x\sqrt{\lambda}, \tag{8}$$

$$\tau(x; \lambda) = \frac{\partial \delta}{\partial \lambda}(x; \lambda) \tag{9}$$

and

$$\begin{aligned} r_\infty(\lambda) &= \lim_{x \to \infty} r(x; \lambda), \\ \delta(\lambda) &= \lim_{x \to \infty} \delta(x; \lambda), \\ \tau(\lambda) &= \lim_{x \to \infty} \tau(x; \lambda). \end{aligned} \tag{10}$$

The following is proved in [17] and covers many typical applications. Note that there are simple examples, such as the *Coulomb potential* $q(x) = 1/x$, where $\delta(\lambda)$ does not exist.

THEOREM 1. *If $\int^\infty q\,dx$ converges and q is eventually monotone then the limits in* (10) *do exist; moreover,*

$$\tau(\lambda) = \delta'(\lambda). \tag{11}$$

Some basic facts are brought together in the next lemma.

LEMMA 2. (i) *If u and v denote the solution $u(x; \lambda)$ for $\lambda = \mu$, ν, respectively, then for any b*

$$(\mu - \nu) \int_0^b uv\,dx = (uv' - vu')(b). \tag{12}$$

(ii) Differential Green's identity. *For $u = u(x; \lambda)$ and any b*

$$\int_0^b u^2\,dx = \left(u' \frac{\partial u}{\partial \lambda} - u \frac{\partial u'}{\partial \lambda}\right)(b). \tag{13}$$

(iii) *In terms of the Prüfer variables,*

$$u' \frac{\partial u}{\partial \lambda} - u \frac{\partial u'}{\partial \lambda} = r^2 \frac{\partial \theta}{\partial \lambda} - \frac{1}{2\lambda} u'u. \tag{14}$$

Proof. Part (i) follows from Green's Identity

$$\int_c^d (-u'' + qu) \cdot v\,dx - \int_c^d u \cdot (-v'' + qv)\,dx = [uv' - u'v]_c^d, \tag{15}$$

Note that the contribution from $x = 0$ on the right-hand side of (12) vanishes because u and v satisfy the same BC there; this is true whether zero is regular, limit-circle, or limit-point. Part (ii) follows from (12) by considering neighbouring solutions $u(x; \lambda)$ and $u(x; \lambda + \delta\lambda)$ and taking the limit, and (iii) comes from simple manipulation.

We then have

PROPOSITION 3. (i) *For any c, d in the interval of definition,*

$$\left[r^2 \frac{\partial \theta}{\partial \lambda}\right]_{x=c}^d = \int_c^d u^2\,dx + \left[\frac{1}{2\lambda} u'u\right]_{x=c}^d. \tag{16}$$

(ii) *In particular (recalling $u(x; \lambda)$ satisfies a λ-independent boundary condition at $x = 0$), for any b,*

$$\left(r^2 \frac{\partial \theta}{\partial \lambda}\right)_{x=b} = \int_0^b u^2\,dx + \left(\frac{1}{2\lambda} u'u\right)_{x=b}. \tag{17}$$

From the above immediately follows Smith's integral formula for $\tau(\lambda)$.

THEOREM 4. (F. T. Smith). *With the above notation and under the assumptions of Theorem* 1 *we have for each X,*

$$(r^2\tau)(X; \lambda) = \int_0^X u^2\,dx + \frac{uu'}{2\lambda}(X; \lambda) - \frac{X}{2\sqrt{\lambda}} \tag{18}$$

and

$$r_\infty^2 \tau(\lambda) = \lim_{X \to \infty} \left(\int_0^X u^2\,dx + \frac{uu'}{2\lambda}(X; \lambda) - \frac{X}{2\sqrt{\lambda}}\right). \tag{19}$$

Proof. The first part comes from the proposition, and the second follows from the first and Theorem 1.

4. THEORY OF THE METHODS USED

4.1. *Bracketing the Resonance*

Given λ with $0 < \lambda < q_{\text{barr}}$, define the *barrier* to be the interval I_B round x_{barr} in which $q(x) \geqslant \lambda$. Say a resonance is *normal* if the corresponding solution u is decaying within the barrier, i.e., $u'/u \leqslant 0$ in I_B. (Non-normal resonances seem rare.) Equivalently, the point $P(x; \lambda) = (u'(x; \lambda), u(x; \lambda))$ lies in the second or fourth quadrant in the (u', u) plane for x in I_B: by changing the sign of u we may assume the second quadrant. For a normal resonance there are no zeros of $u(x; \lambda)$ in I_B. Define the *index* k of the resonance to be the number of zeros of u to the left of the barrier (excluding $x = 0$).

Let λ_{res} be a normal resonance of index k. Let us (temporarily) write $\lambda_-(b)$ for the kth eigenvalue of the problem

defined by (1) with the given BC at zero, together with the BC $u'=0$ at the point b. Also write $\lambda_+(b)$ for the corresponding eigenvalue, but with the BC $u=0$ at the point b.

The differential Green's identity (13) implies the well-known fact that for each fixed b, the point $P(b;\lambda)$ rotates in the strictly positive sense around zero as λ increases. Thus, as λ increases from λ_{res}, the point $P(b;\lambda)$ moves anticlockwise in the second quadrant with no extra zeros of u appearing on the interval $(0, b)$, until it meets the $u=0$ axis when $\lambda=\lambda_+(b)$. Similarly, as λ decreases from λ_{res}, point $P(b;\lambda)$ moves clockwise to meet the $u'=0$ axis when $\lambda=\lambda_-(b)$. This proves that for a normal resonance,

$$\lambda_-(b)\leqslant\lambda_{\text{res}}\leqslant\lambda_+(b) \tag{20}$$

for all b in the barrier (with, obviously, the two outer terms being unequal). Further, from (12), we have for any b,

$$(\lambda_+(b)-\lambda_-(b))\int_0^b u_-u_+\,dx=-u_-(b;b)\,u'_+(b;b), \tag{21}$$

writing $u_-(x;b)$, $u_+(x;b)$ for the corresponding eigenfunctions. In practice these (normalized) functions are almost equal, deviating markedly only near b, so that the integral on the left is close to 1, and we have

$$\lambda_+(b)-\lambda_-(b)\approx -u_-(b;b)\,u'_+(b;b). \tag{22}$$

4.2. *Iteration to Find Bracketing Interval*

As b moves to the right within the barrier, the interval $[\lambda_-,\lambda_+]$ decreases. Thus as a preliminary to locating a resonance, a good strategy is to find a b precisely at the end of the barrier, i.e., a root of

$$q(b)=\lambda_-(b), \tag{23}$$

together with $\lambda_-(b)$ and $\lambda_+(b)$. The behaviour of λ_- and λ_+ helps to clarify the different cases that can arise and how software can recognize and handle them. In terms of the Prüfer variable θ, $\lambda_-(x)$ and $\lambda_+(x)$ are the solutions λ of

$$\theta(x;\lambda)=(k+\tfrac{1}{2})\pi, \tag{24}$$

$$\theta(x;\lambda)=(k+1)\pi, \tag{25}$$

respectively.

By an argument similar to the derivation of (13), for an eigenvalue of (1) subject to the given BC at zero and a fixed, regular BC at the *variable* endpoint b, the dependence of λ on b is continuous and

$$\frac{\partial\lambda}{\partial b}=-(u'^2+(\lambda-q)\,u^2)|_{x=b}, \tag{26}$$

where u denotes the corresponding eigenfunction (which depends on b) normalized so that $\int_0^b u^2\,dx=1$. In the present notation this gives

$$\lambda'_-(x)=(q(x)-\lambda(x))u_-(x;x)^2, \tag{27}$$

$$\lambda'_+(x)=-u'_+(x;x)^2. \tag{28}$$

Since an eigenfunction and its derivative cannot simultaneously vanish, the quantities $u_-(x;x)^2$ and $u'_+(x;x)^2$ are strictly positive. (Note that if q has isolated jumps, then (27) shows $\lambda_-(x)$ has "corners" at the jumps; this does not affect the theory.)

LEMMA 5. *With the above notation,*

(i) $\lambda_+(x)>\lambda_-(x)$ *for all* x.

(ii) $\lambda_+(x)$ *is always strictly decreasing.*

(iii) $\lambda_-(x)$ *is strictly increasing in any interval where* $q>\lambda_-$; *strictly decreasing in any interval where* $q<\lambda_-$; *and* $\lambda'(x)=0$ *at a point* x *where* $q(x)$ *is continuous and* $q(x)=\lambda_-(x)$.

(iv) *Let* q *be strictly decreasing on some interval* I. *If* $q(x_0)\leqslant\lambda_-(x_0)$ *at some points* x_0 *of* I *then* $q(x)<\lambda_-(x)$ *for all larger* x *in* I. *Similarly if* q *is strictly increasing on* I *and* $q(x_0)\geqslant\lambda_-(x_0)$ *at some* x_0 *in* I *then* $q(x)>\lambda_-(x)$ *for all larger* x *in* I.

Proof. Part (i) is in (20); parts (ii) and (iii) come from (27), (28).

(iv) Writing $k(x)=u(x;x)^2$ and using the integrating factor $e^{\int k\,dx}$, we obtain from (27) the explicit integral formula

$$[q(x)-\lambda_-(x)]_{x_0}^{x_1}=\int_{x_0}^{x_1}\exp\left(\int_t^{x_1}k(s)\,ds\right)q'(t)\,dt. \tag{29}$$

If q is strictly decreasing (increasing) then the right-hand side of (29) is strictly <0 (resp. >0) for any $x_1>x_0$ and the result follows. ∎

Note. If q has jumps, $q'(t)\,dt$ is to be taken in the Stieltjes sense $dq(t)$, and the result is still valid.

Specializing to the single-barrier case of most interest to us, let us suppose as before that q decreases to zero on the interval $I=[x_{\text{barr}},\infty)$, where $q(x_{\text{barr}})=q_{\text{barr}}$ and, further make the

Assumption. q' is continuous on I, and $q'<0$ for all $x>x_{\text{barr}}$.

THEOREM 6 (One-barrier cases). *There are three mutually exclusive cases:*

(A) $\lambda_-(x)\leqslant 0$ *for all* x *in* I, *in which case no* b *satisfying* (23) *exists.*

(B) $q_{\text{barr}} < \lambda_-(x_{\text{barr}})$ *in which case again no* b *exists.*

(C) *There is a root* b *of* (23) *in* I. *In this case* b *is unique, and is the unique global maximum of* $\lambda_-(x)$ *on* I.

Proof. Since $q > 0$ on I it is clear that (A) excludes (B) and (C). Also if (B) holds then by Lemma 5, $q(x) < \lambda_-(x)$ for all x in I so (C) cannot hold. Thus (A), (B), (C) are mutually exclusive. To show they are exhaustive we suppose that (A), (B), and (C) all fail and derive a contradiction. That is, $\lambda_-(x_0) > 0$ for some x_0 in I, and $\lambda_-(x_{\text{barr}}) \leqslant q_{\text{barr}} = q(x_{\text{barr}})$ and there is no root of $q(x) = \lambda_-(x)$ in I. Then $q - \lambda_-$ must be of one sign, namely > 0, on I, so by Lemma 5(iii), λ_- is increasing on I. This implies that $q(x) > \lambda_-(x) \geqslant \lambda_-(x_0) > 0$ for all $x \geqslant x_0$, contrary to the fact that $q \to 0$. This shows that (A), (B), and (C) exhaust the possibilities.

If a root exists in I then there is a smallest such root b and Lemma 5 shows that $q(x) < \lambda_-(x)$ for all $x > b$ so that b is unique. Then λ_- is increasing for $x < b$ and decreasing for $x > b$ by Lemma 5, so b is the unique global maximum of λ_- and the proof is complete. ∎

The last theorem leads to a simple iteration process for finding b and $\lambda_-(b)$:

ALGORITHM 1 (Iteration for $\lambda_-(b)$).

Input: $q_0(x)$ and l, defining $q(x)$.
x_{barr}, the position of the barrier top, and q_{barr}.
k = no. of oscillations "before the barrier."

1. Evaluate $\lambda_-(x_{\text{barr}})$. If this is $> q_{\text{barr}}$, fail.
   If it is $= q_{\text{barr}}$, exit with $b = q_{\text{barr}}$ and $\lambda_-(b) = q_{\text{barr}}$.
2. Find an x_0 (which may be x_{barr}) for which $\lambda_0 = \lambda_-(x_0) > 0$.
   If none can be found, fail.
3. (At this point necessarily $0 < \lambda_0 < q_{\text{barr}}$.)
   For $m = 1, 2, \ldots$ until converged, do steps 4, 5.
4. Set $x_m =$ (root of $q(x_m) = \lambda_{m-1}$).
5. Set $\lambda_m = \lambda_-(x_m)$.
6. Exit with the final x_m and λ_m as b and $\lambda_-(b)$.

THEOREM 7. *Subject to the assumptions above, the* x_m, λ_m *in steps* 3–5 *of this algorithm converge certainly and superlinearly to* b *and* $\lambda_-(b)$.

Proof. The method is simply fixed-point iteration

$$\lambda_m = \phi(\lambda_{m-1})$$

on the function $\phi(\lambda) = \lambda_-(q^{-1}(\lambda))$, where q^{-1} is the inverse function of q regarding the latter as a mapping from I to $(0, q_{\text{barr}}]$. Since $\lambda_-'(b) = 0$ and $q'(b)$ is assumed nonzero, it follows that $\phi'(\lambda_-(b)) = 0$ so, sufficiently close to $\lambda_-(b)$, convergence is superlinear. To see that the method always converges, note that $\lambda_m \leqslant \lambda_-(b) = q(b)$ for $m \geqslant 0$ because b is the global maximum of $\lambda_-(x)$; hence $x_m \geqslant b$ for $m \geqslant 1$ because q is decreasing. Thus, $q(x_{m-1}) \leqslant \lambda_-(x_m)$ for $m \geqslant 1$; that is, $\lambda_{m-1} \leqslant \lambda_m$. Thus the iterates λ_m form a bounded increasing sequence whose limit λ must satisfy $\lambda_-(x) = \lambda$ for some x with $q(x) = \lambda$; that is, x is a root of (23). By uniqueness, x must be b and λ must be $\lambda_-(b)$. ∎

When q is sufficiently smooth as in typical applications, $\lambda_-(x)$ has a smooth maximum at b (i.e., $\lambda_-(b+t) = \lambda_-(b) + O(t^2)$ for small t) and convergence is quadratic. In experiments, it was found that usually only three or four iterations were required for convergence to tolerances of around 10^{-6}.

4.3. *Asymptotic correction of* $\tau(\lambda)$

A disadvantage of existing algorithms based on the time-delay is that, with typical potentials $q(x)$, one needs to integrate out to very large distances in order that $\delta(x; \lambda)$ and $\tau(x; \lambda)$ approximate sufficiently well to their limiting values. It is useful to make use of the fact that the underlying potential $q_0(x)$ usually decays faster than $O(1/x^2)$ at infinity. We now describe how this can be done.

With b as in the last subsection, choose $X \geqslant b$ so that q can, for $x > X$, be taken to be equal to $l(l+1)/x^2$. We can write

$$r_\infty^2(\lambda)\,\tau(\lambda) = (r^2\tau)(X; \lambda) + [(r^2\tau)(X; \lambda)]_X^\infty. \tag{30}$$

For $x \geqslant X$ the differential equation is to be taken as being

$$u'' + \left(\lambda - \frac{\nu^2 - 1/4}{x^2}\right) u = 0, \tag{31}$$

where $\nu = l + \frac{1}{2}$ (so $l(l+1) = \nu^2 - \frac{1}{4}$), solutions of which are scaled Bessel functions:

$$u = A\lambda^{-1/4} j(x\sqrt{\lambda}) + B\lambda^{-1/4} y(x\sqrt{\lambda}), \tag{32}$$

where

$$j(t) = \sqrt{\pi t/2}\, J_\nu(t), \qquad y(t) = \sqrt{\pi t/2}\, Y_\nu(t) \tag{33}$$

the factor $\sqrt{\pi/2}$ being put in so that j and y each oscillate with amplitude one at infinity. Here A and B depend on λ in an unknown way, but note that the RHS of (16) in Proposition 3 contains no derivatives with respect to λ, so if we replace the actual family $u(x; \lambda)$ on $[X, \infty)$ by any other family of solutions to (31) that coincides with ours when $\lambda = \lambda_0$, then we shall obtain the same value of $[r^2\tau]_c^d$ (for $\lambda = \lambda_0$). A convenient choice is

$$u = \lambda^{-1/4} f(x\sqrt{\lambda}), \tag{34}$$

where

$$f(t) = Aj(t) + By(t) \tag{35}$$

and A, B are frozen at their values in (32) when $\lambda = \lambda_0$. Clearly then for $\lambda = \lambda_0$, (32) and (34) coincide.

The functions j, y and hence f are solutions of

$$\frac{d^2f}{dt^2} + \left(1 - \frac{\nu^2 - 1/4}{t^2}\right) f = 0. \tag{36}$$

We may now evaluate $\partial\theta/\partial\lambda$ by direct differentiation. From (14) and the definition of $\tau(x;\lambda)$ we have

$$r^2\tau = u' \frac{\partial u}{\partial \lambda} - u \frac{\partial u'}{\partial \lambda} + \frac{uu'}{2\lambda} - \frac{x}{2\sqrt{\lambda}}. \tag{37}$$

By (34), writing $t = x\sqrt{\lambda}$, we have

$$\begin{aligned} u' &= \lambda^{1/4} f'(t), \\ \frac{\partial u}{\partial \lambda} &= \frac{x}{2} \lambda^{-3/2} f'(t), \\ \frac{\partial u'}{\partial \lambda} &= \frac{1}{4} \lambda^{-3/4} f'(t) + \frac{x}{2} \lambda^{-1/4} f''(t), \end{aligned} \tag{38}$$

and

$$r^2 = (f^2 + f'^2)(t). \tag{39}$$

Substituting into (37) and using (36) gives

$$r^2\tau = -\frac{(\nu^2 - 1/4)u^2}{2x\lambda}. \tag{40}$$

As said above, this $r^2\tau$ is not the one for our actual solution, but the *change* in it over any subinterval of $[X, \infty)$ is the same as for our actual solution, when $\lambda = \lambda_0$. Now since all solutions of (36) are bounded at ∞ we have $\lim_{x\to\infty} r^2\tau = 0$ in (40), whence

$$\begin{aligned} [r^2\tau]_X^\infty &= -(r^2\tau)(X;\lambda) \\ &= \frac{(\nu^2 - 1/4)\, u(X;\lambda)^2}{2X\lambda}. \end{aligned}$$

We derived this for a particular $\lambda = \lambda_0$, but on the way, explicit reference to the A, B that belong to this λ_0 has dropped out of the RHS. We have thus proved

THEOREM 8 (Bessel correction). *Given that* $q(x) = (\nu^2 - \frac{1}{4})/x^2$ *for* $x \geqslant X$, *we have for any* $\lambda > 0$,

$$r_\infty^2(\lambda)\, \tau(\lambda) = r(X;\lambda)^2\, \tau(X;\lambda) + \frac{(\nu^2 - 1/4)\, u(X;\lambda)^2}{2X\lambda}. \tag{41}$$

Clearly, since the correction term is $O(1/X)$, the alternative of integrating out to a point where it is negligible can be quite expensive.

4.4. *Computation of* r_∞

It is necessary to compute $r_\infty^2(\lambda)$ in order to compute $\tau(\lambda)$ in (41) and for this we need to compute A, B in (35). By [18, p. 361, (9.2.29)],

$$\begin{aligned} j(t) &= m(t)\cos(\theta^*(t)), \\ y(t) &= m(t)\sin(\theta^*(t)) \qquad (t \to \infty), \end{aligned} \tag{42}$$

where $\theta^* = t - (\nu/2 + 1/4)\pi + O(1/t)$ and $m(t) = 1 + O(1/t^2)$, and it easily follows from this and (39) that $r^2 \to A^2 + B^2$ as $x \to \infty$, that is,

$$r_\infty^2(\lambda) = A^2 + B^2. \tag{43}$$

Now writing $t = x\sqrt{\lambda}$ and $T = X\sqrt{\lambda}$, we have by (34), (38)

$$\begin{aligned} u(X) &= \lambda^{-1/4} f(T) = \lambda^{-1/4}(Aj(T) + By(T)), \\ u'(X) &= \lambda^{1/4} f'(T) = \lambda^{1/4}(Aj'(T) + By'(T)). \end{aligned} \tag{44}$$

Since j, y are solutions of (36) the Wronskian $jy' - yj'$ is constant, and equal to one in view of (42), so these equations can be solved to give

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} y'(T) & -y(T) \\ -j'(T) & j(T) \end{pmatrix} \begin{pmatrix} \lambda^{1/4} u(X) \\ \lambda^{-1/4} u'(X) \end{pmatrix}, \tag{45}$$

and then (43) gives the result.

Note that by the recurrence relations [18, p. 361 (9.1.27)] we have

$$j'(t) = \sqrt{\pi t/2} \left(\frac{\nu + 1/2}{t} J_\nu(t) - J_{\nu+1}(t) \right) \tag{46}$$

and similarly for $y(t)$. A convenient way to compute the required Bessel functions is Amos' method [19] as implemented, for instance, in NAG routine S17DLF, which can deliver $H_\nu^{(1)} = J_\nu + iY_\nu$, $H_{\nu+1}^{(1)} = J_{\nu+1} + iY_{\nu+1}$ in a single call.

4.5. *Interpolation between* λ_- *and* λ_+

There is a further bonus of the bracketing process, as follows. Let b be the barrier end defined by (23). Revert to the notation $u(x;\lambda)$ of Section 3 and assume u normalized over $[0, b]$:

$$\int_0^b |u(x;\lambda)|^2\, dx = 1. \tag{47}$$

Consider the path of the point $P(b;\lambda)$ introduced in Section 4.1 as λ varies from $\lambda_-(b)$ to $\lambda_+(b)$. We leave b fixed, so we write λ_-, λ_+ for $\lambda_-(b)$, $\lambda_+(b)$ in this section.

$P(b;\lambda_-)$ has the form $(0, u_-)$, where u_- $(u_- > 0)$ equals $u_-(b;b)$ in the notation of Section 4.1 or $u(b;\lambda_-)$ in the present notation. Similarly $P(b;\lambda_+)$ has the form $(u'_+, 0)$, where u'_+ $(u'_+ < 0)$ equals $u'_+(b;b)$ in Section 4.1 or $u'(b;\lambda_+)$ here.

It turns out that the path of P between these limits is very nearly a *linear function of* λ; that is,

$$P(b;\lambda) \approx (tu'_+, (1-t)\,u_-) \tag{48}$$

very nearly, where $\lambda = (1-t)\,\lambda_- + t\lambda_+$, $0 \leqslant t \leqslant 1$, very nearly. (This is an experimental result. We have at present no theory that quantifies the closeness to linearity although it is clear that the sharper the exponential decay in the barrier, i.e., the sharper the resonance, the more nearly it holds.) This gives a cheap estimate for initial conditions at b from which we may integrate further up to an X where $q(x)$ approximates to $l(l+1)/x^2$ sufficiently that the Bessel correction may be applied.

Information that is available for free once λ_-, λ_+ have been found gives a higher order interpolant. Namely the differential Green's identity (13) gives

$$u'\frac{\partial u}{\partial \lambda} - u\frac{\partial u'}{\partial \lambda} = 1 \tag{49}$$

at b for the normalized solution. Applied for $\lambda = \lambda_-, \lambda_+$ this gives one value each of $\partial u/\partial\lambda$ and $\partial u'/\partial\lambda$. Matching a quadratic to these gives

$$P(b;\lambda) \approx (u'_+\{t - Ct(1-t)\},\, u_-\{(1-t) - Ct(1-t)\}). \tag{50}$$

This proved to give erratic results and the code uses an interpolation based on solving the eigenproblem one or two more times to find further "exact" $P(b;\lambda)$ values for suitably chosen $\lambda \in [\lambda_-, \lambda_+]$. This allows error estimation and has proved more reliable.

5. IMPLEMENTATION

5.1. *Design Criteria*

In outline, the algorithm has these two ingredients:

- The function $\tau(\lambda)$ is estimated for any given λ by integrating the differential equation from zero sufficiently far to estimate the limits on the left and right sides of (19).
- The resulting function is maximized over a suitable λ range.

The overall cost of the computation is due, first, to the cost of each individual integration and, second, to the number of integrations, i.e., $\tau(\lambda)$ evaluations, needed overall. In concept our method is very similar to that of LeRoy's TDELAY. However our aim was to make the code considerably more reliable than TDELAY while not losing efficiency. The various algorithmic features we have incorporated reduce cost on both the fronts just mentioned and appear to have achieved the desired combination of efficiency and robustness.

A second aim was to make a code that was easier to use than TDELAY. The latter is a complete program rather than a subroutine and has no clear separation between input which defines the potential function for the problem to be solved and input of initial guesses, stepsizes, etc. for the algorithm. We feel this also has been achieved.

TDELAY integrates (1) in untransformed form by the well-known Numerov method with a fixed stepsize h and without built-in error estimation. Thus for accuracy it is necessary to choose h to fit the worst-behaved part of the solution. The integration is also taken out to potentially very large x to estimate the limiting behaviour satisfactorily. The result is less expensive than one might expect because the code stops integrating when the estimates of τ at three successive steps agree within a tolerance: the smaller the stepsize the easier it is for this to happen. TDELAY searches for all resonances in a given λ range, without a means of individually identifying them. This is a good feature in some ways, but makes the code liable to miss very sharp resonances and, often, to fail if a good enough initial guess is not given.

Our code RESON makes use of the standard Sturm–Liouville eigenvalue solver SL02F (Marletta and Pryce [20], Marletta [21]) based on a Pruess coefficient approximation method [22, 23]. The integration from b to X is done by an adaptation of the same method. Both parts of the process have built-in error estimation and stepsize control. We have as yet no well worked out theory of the relation between this "internal" error control and the error in the computed resonance position and width, but it produces quite good correspondence between the error tolerance given to the code and the resulting error as far as we are able to tell.

We took the view that for a robust code it is desirable to "index" resonances in the same way as one can do for eigenvalues and to look for the presence or absence of a particular resonance. This has been done using the index defined in Subsection 4.1.

The code uses the methods described in Section 4, namely the preliminary bracketing "fix" $[\lambda_-, \lambda_+]$, the Bessel function asymptotics and the interpolation over $[\lambda_-, \lambda_+]$. The bracket has always proved to bracket a unique resonance in our experiments, and the sharper the resonance, the smaller the interval. This helps both robustness and efficiency.

The position x_{barr} of the barrier which produces the resonance is an important input to Algorithm 1. We ask the user to supply an interval expected to contain x_{barr}, and the

code locates it more accurately. This makes it possible (with circumspection) to handle problems with several barriers, provided they do not interact too badly.

An apparently minor but significant algorithmic point should be mentioned here. The resonance is found by maximizing $\tau(\lambda)$ without having derivatives of the latter available (we use NAG routine E04ACF). Clearly, at a typical (quadratic) maximum, errors of $O(\varepsilon)$ in τ can induce errors of $O(\sqrt{\varepsilon})$ in the computed resonance. Thus to obtain consistent results it helps to reduce any errors that are "uncorrelated" between different evaluations of τ. We experimented with several ways of doing the integration from b to X. Methods with variable stepsize control done independently at each evaluation seemed to cause too much "uncorrelated error." The method chosen uses a fixed initial mesh with repeated mesh bisection and Richardson extrapolation done as many times as required for accuracy. Thus in practice, all evaluations (once sufficiently near the maximum) use the same meshpoints. This appears to give satisfactory smoothness to $\tau(\lambda)$.

5.2. *Outline of Algorithm*

The resulting algorithm is as follows.

ALGORITHM 2 (Time delay method).

Input: $q_0(x)$ and l, defining $q(x)$.
x_{barr}, the position of the barrier top.
k = no. of oscillations "before the barrier."

1. Find λ_-, b, λ_+ by Algorithm 1.
   Record $u(b)$, $u'(b)$, $\int_0^b u^2$ for solutions found.
2. Find X where q_0 becomes negligible (this is independent of step 1.)
3. For $\lambda_- \leqslant \lambda \leqslant \lambda_+$ define a function $\tau(\lambda)$ by
   — Form $u(b)$, $u'(b)$ by interpolation.
   — Continue integration numerically from these initial values to X and compute $\phi(X)$.
   — Evaluate $[\phi]_X^\infty$ and r_∞ as above.
   — Hence evaluate $\tau(\lambda)$.
4. Maximize $\tau(\lambda)$ over $[\lambda_-, \lambda_+]$.

The above algorithm has been implemented in a Fortran 77 subroutine RESON, whose parameter list is summarized below.

Input

coeffn	real function defining $q_0(x)$.
aj	real, the parameter J, where $q(x) = q_0(x) + J(J+1)/x^2$.
k	integer, the index of the resonance.
xlo, xhi	real, defining an interval supposed to contain the position of the barrier top x_{barr}.
tol	real, an absolute accuracy tolerance.

Output

elam	real, the computed resonance λ_{res}.
delam	real, an estimate of the error in λ_{res}.

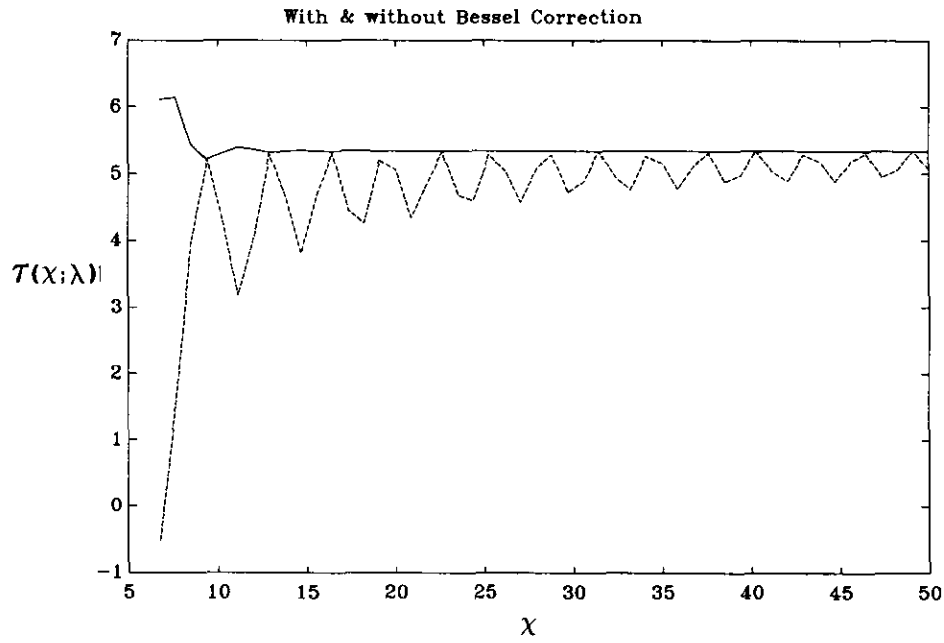

FIG. 1. LJ(12, 6) potential with $R_e = 3.56$, $l = 7$, $D_e = 40$. Dashed curve: $\tau(x; \lambda)$ against x. Solid curve: $\tau_{\text{bes}}(x; \lambda)$, the result of applying Bessel correction to $\tau(x; \lambda)$ at the point x.

tau	real, the value of $\tau(\lambda_{\text{res}})$. The larger this is, the sharper and narrower is the resonance.
info	real array, containing ancillary information.
ifail	integer, the usual NAG-style error indicator.

The code and user instructions are available from the author and will be covered more fully in a later paper.

6. NUMERICAL RESULTS AND COMPARISONS

Tests of RESON were carried out on two problems for which numerical results are available in Marletta [24, 3]. The computing system was the VAX 8820 at RMCS, using VMS Fortran 77 in double precision, a precision of about 16 decimal digits. The first problem is a Lennard–Jones LJ(12, 6) potential defined by

$$q(x)=\frac{1.92}{16.858056}D_e\left(\left(\frac{R_e}{x}\right)^{12}-2\left(\frac{R_e}{x}\right)^{6}\right)+\frac{l(l+1)}{x^2} \tag{51}$$

(the strange factor is due to conversion between different physical units). The second is a Morse potential defined by

$$q(x)=D_e((e^{-\alpha(x/R_e-1)})^2-2(e^{-\alpha(x/R_e-1)}))+\frac{l(l+1)}{x^2}. \tag{52}$$

We first give some results which illustrate the internal workings of the code, as follows. Figure 1 shows the improvement given by the Bessel correction for one typical case. One curve shows $\tau(X;\lambda)$ as a function of X. The other shows the approximation to $\tau(\lambda)$ obtained by applying (41) at the point X, as a function of X.

To illustrate the shape of the τ "spike," Fig. 2 shows the graph of $\tau(\lambda)$ between λ_- and λ_+ for three typical resonances—broad, medium, and narrow—of the LJ potential. For each one we graph τ as computed by the "linear" and "quadratic" methods of interpolating between λ_- and λ_+ and by a more expensive method which does not use interpolation. These were obtained using the subroutine for computing τ which forms part of the package, after locating λ_-,

TABLE I

Test of Three Codes on a Lennard–Jones Potential

$$q(x)=\frac{1.92}{16.858056}D_e\left(\left(\frac{R_e}{x}\right)^{12}-2\left(\frac{R_e}{x}\right)^{6}\right)+\frac{l(l+1)}{x^2}$$

	RESON					RESONE		TDELAY	
D_e	λ_{res}	λ_-	λ_+	τ	CPU	λ_{res}	CPU	λ_{res}	CPU
62	0.089966	0.089966	0.089966	2.9e8	18.4	—	—	0.99*	1.56
60	0.219685	0.219682	0.219690	4.0e5	10.9	0.219685	36	0.23765	1.66
55	0.525025	0.523375	0.527001	8.9e2	9.4	0.524179	43	0.52494	1.01
50	0.795038	0.776613	0.821643	6.9e1	10.3	0.795142	44	0.79492	0.47
45	1.034113	0.982052	1.154640	1.8e1	9.5	1.035426	137	1.03302	0.62
40	1.252641	1.182421	1.702625	7.2e0	8.9	1.248539	120	1.24970	0.67

Note. TOL $= 10^{-6}$, $R_e = 3.56$, $l = 7$, resonance index $k = 0$.

TABLE II

Test of Three Codes on a Morse Potential

$$q(x)=D_e((e^{-\alpha(x/R_e-1)})^2-2(e^{-\alpha(x/R_e-1)}))+\frac{l(l+1)}{x^2}$$

	RESON					RESONE		TDELAY	
D_e	λ_{res}	λ_-	λ_+	τ	CPU	λ_{res}	CPU	λ_{res}	CPU
55	0.044005	0.044005	0.044005	1.9e8	37.4	0.044005	59	0.0440	0.63
54	0.154967	0.154906	0.155041	2.4e4	16.6	0.154966	78	0.1550	1.08
53	0.252199	0.250714	0.254077	9.5e2	15.9	0.252205	99	0.2520	0.21
52	0.334729	0.327338	0.345509	1.7e2	14.9	0.335008	76	0.3347	0.27
51	0.404204	0.386700	0.437284	5.8e1	14.5	0.404043	99	0.4040	0.19
50	0.463002	0.435262	0.540247	2.8e1	14.4	0.462438	93	0.4623	0.26

Note. TOL $= 10^{-6}$, $R_e = 3.56$, $\alpha = 4$, $l = 7$, resonance index $k = 5$.

TABLE III

Some Results for Higher-Order LJ(12, 6) Resonances;
$R_e = 10$, $l = 20$, $k = 5, 6, 7$, TOL $= 10^{-6}$

D_e	λ_{res}	λ_-	λ_+	τ_{res}	CPU
Resonance index $k = 5$					
134		λ above barrier			
135	0.61186	0.5953	0.6723	5.1e1	14.3
140	0.55855	0.5498	0.5710	1.5e2	13.3
160	0.207128	Same	Same	1.7e10	14.3
165	0.090651	Same	Same	1.1e17	18.2
167	0.041595	Same	Same	1.4e12	21.2
168		SL02F fails			
169		λ below zero			
Resonance index $k = 6$					
165		λ above barrier			
166	0.55173	0.536	0.607	5.5e1	12.4
180	0.407217	0.406910	0.407532	5.4e3	15.4
190	0.250914	Same	Same	5.9e7	14.6
200	0.060958	Same	Same	4.0e19	18.2
202	0.019518	Same	Same	1.7e6?	24.2
202.5		SL02F fails			
203		λ below zero			
Resonance index $k = 7$					
199.5		λ above barrier			
200	0.50308	0.4894	0.5567	6.0e1	14.0
210	0.427436	0.4245	0.4306	5.5e2	14.3
220	0.316401	0.31638	0.31641	1.2e5	14.2
230	0.173339	Same	Same	1.8e10	15.6
238	0.040645	Same	Same	2.2e13	17.1
240		SL02F fails			
241		λ below zero			

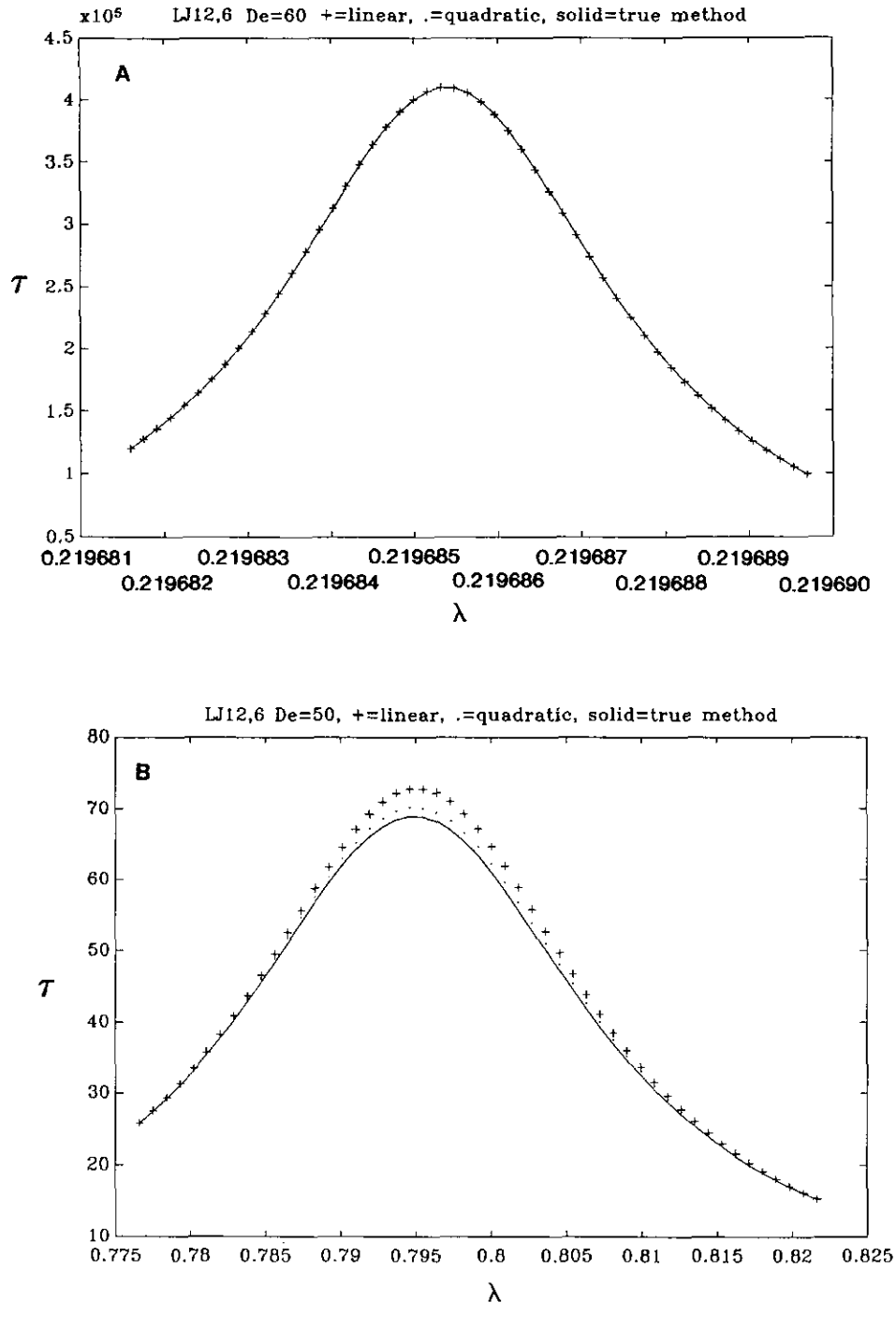

FIG. 2. Graphs of τ against t where $\lambda = \lambda_- + t(\lambda_+ - \lambda_-)$, for LJ(12, 6) potential, $R_e = 3.56$, $l = 7$, $D_e = 40, 50, 60$, resonance index $k = 0$.

λ_+ by the method used in RESON. They thus call on all the algorithms in RESON except the final maximizing process.

Second, we give results of some experiments with RESON itself, summarized in Tables I, II, with comparable results from Marletta's experimental code RESONE [24] and from TDELAY. RESONE uses similar ideas to RESON but has neither the bracketing λ_-, λ_+ process nor the Bessel correction. In these, the depth D_e of the well was varied keeping other parameters unchanged, to make the resonance λ come in at the top of the barrier (a broad resonance) and move down toward zero, at which point the resonance becomes infinitely sharp and λ becomes an eigenvalue. We give the value of RESON's λ_-, λ_+ ("same" means that they equal λ_{res} to the number of figures given). The

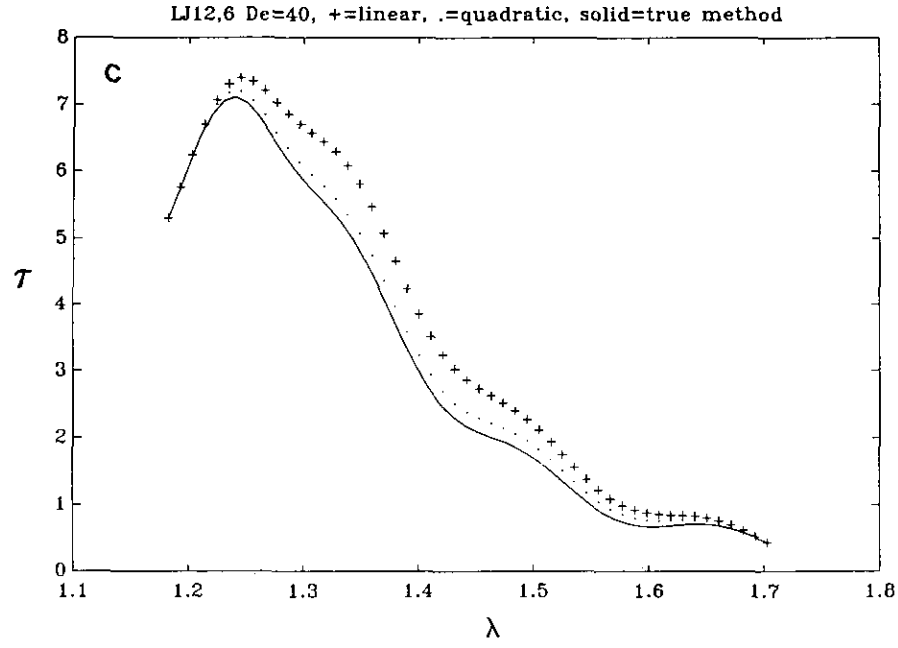

FIG. 2—*Continued*

value of τ at the resonance is given to indicate the sharpness of the resonance. "CPU" values are time in seconds on the VAX 8820 at RMCS.

TDELAY is far from automatic, requiring integration limits, (constant) stepsizes, and other method parameters to be set by the user; consequently comparisons of runtimes are not very meaningful. For the LJ(12, 6) results, integration was started at $x = 0.2$ with stepsize 0.03 and continued out to around $x = 300$. The author easily persuaded it to find an index zero resonance of the LJ(12, 6) problem for $40 \leqslant D_e \leqslant 56$ and (with difficulty) for $57 \leqslant D_e \leqslant 60.5$. On the latter range it found an index one resonance (marked * for $D_e = 62$) which RESON refused to acknowledge. For the Morse results integration was started at 0.2 with stepsize

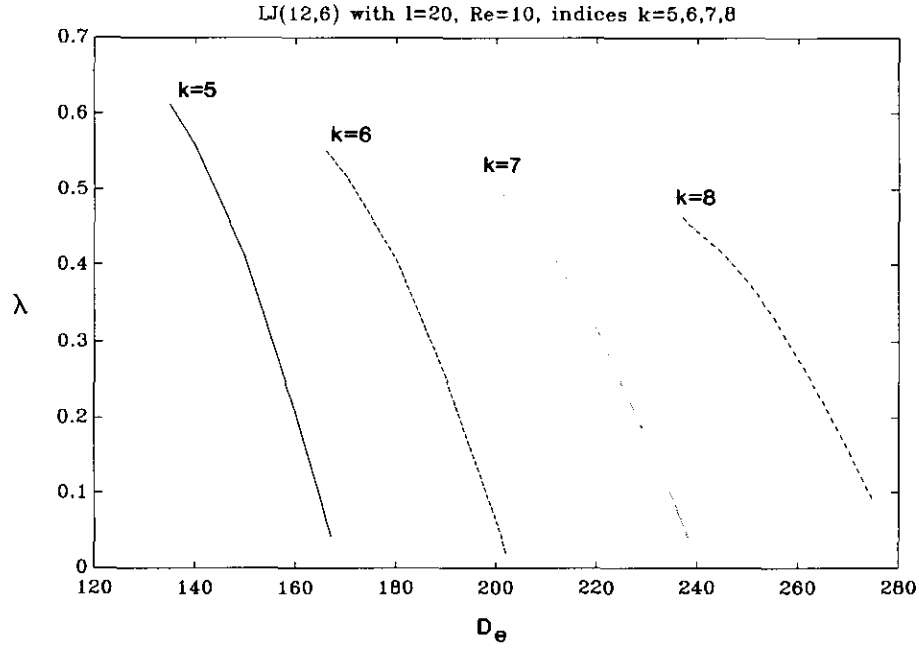

FIG. 3. LJ(12, 6) potential with $R_e = 10$, $l = 20$. Resonances of various indices k plotted against D_e.

0.02 or 0.01. TDELAY needed even more help with initial trial values of λ (which were chosen about 0.01 away from the value found by RESON) and the search-step in the λ direction (around 0.002 or 0.001), else it failed to converge or, sometimes, gave results which seemed to indicate success but were wrong on closer examination.

Table III shows a family of resonances for the LJ(12, 6) potential for $l = 20$. This was chosen as giving a barrier large enough to accommodate more than one resonance for certain values of D_e (e.g., $D_e = 200$ admits both a $k = 6$ and $k = 7$ resonance). R_e (which is essentially a scaling factor for the equation) has been taken as 10 to give modest values for D_e. The index $k = 5, 6, 7$ resonances are tabulated for various D_e over their range. This is an abbreviated version of the results shown graphically in Fig. 3. In these computations the only hiccups in RESON were due to the eigensolver SL02F failing. This happened at the sharpest resonances, for which the barrier end b is quite large (since λ is small positive). The reason seems to be that SL02F then has difficulty in mesh selection for a problem that has most of the action concentrated at one singular end ($x = 0$) with the other end being regular. This is not due to any inherent difficulty in the problem at hand.

7. SUMMARY AND CONCLUSIONS

The tests show that generally RESON is much cheaper to use than RESONE. Meaningful comparison with TDELAY as regards efficiency is difficult as the latter depends so much on user-supplied stepsizes and other tuning parameters. These affect not only the efficiency but whether any results at all are obtained. The strongest feature of RESON appears to be its robustness. It needs to be given the basic mathematical definition of the problem, namely the differential equation and a resonance index. Then, apart from an accuracy tolerance, the only extra information it needs to become started is some guide as to where the barrier is. If a resonance of the given index does not exist it diagnoses k as too large or too small; this seems to be reliably done. Thus RESON is a reasonable approximation to the goal of an "automatic" code.

RESON proved able to cope with resonances of sharpness from $\tau = O(1)$ to $\tau = O(10^{15})$ or higher with similar ease as shown by the CPU figures. However, at very sharp resonances there is some erratic behaviour in the computed τ values (see the tables).

Various convenience features could be added to RESON without undue effort. In particular if a resonance of the requested index does not exist it could be directed to return the "lowest," or the "highest," resonance associated with the barrier.

TDELAY is the most prone to obtain inaccurate results, especially for sharp resonances. RESON's preliminary location method makes it possible to find resonances which RESONE cannot and which require considerable experimentation to find with TDELAY.

The most doubt attaches to the accuracy of the results produced by *any* of the codes for the case of broad resonances. For the author's code, various different methods of doing the integration from b to X in computing $\tau(X; \lambda)$ were tried, and discrepancies of order 10^{-3}, in the computed λ_{res}, were common with tolerance around 10^{-6}. This is consistent with there being errors of $O(10^{-6})$ in values of a function which is being maximized by an algorithm that does not use derivatives. For example, one trial did integrations with a uniform mesh, using the Pruess-type method with extrapolation used in the eigenvalue-solver SL02F [20]; it was observed that merely changing the number of steps could cause significant movements in the position of the maximum. It seems that the position of broad resonances may be inherently ill-defined in the sense of being highly sensitive to small perturbations in $q(x)$. A rigorous sensitivity analysis of this problem is required and does not seem to have been carried out.

In future work we aim to report on the performance of RESON on problems with more than one barrier.

The author thanks Dr. Marco Marletta for his work in running some of the tests and for many constructive comments on earlier drafts of the paper and NAG and SERC for their financial support.

RESON is available from the author at: Software Engineering Group, Shrivenham, Swindon SN6 8LA, UK.

REFERENCES

1. J. R. Taylor, *Scattering Theory* (Wiley, New York, 1972).
2. R. J. LeRoy and R. B. Bernstein, *J. Chem. Phys.* **54**, 5114 (1971).
3. R. J. LeRoy and W.-K. Liu, *J. Chem. Phys.* **69**, 3622 (1978).
4. R. J. LeRoy, *Program No. KQ03 (TDELAY)*, Nat. Resource Comput. Chem., 1980, Solftware Catalog Vol. I.
5. M. Reed and B. Simon, *Methods of Mathematical Physics* (Academic Press, New York, 1979).
6. F. T. Smith, *Phys. Rev.* **118**, 349 (1960).
7. O. R. Rice, *Phys. Rev.* **33**, 748 (1929).
8. M. Hehenberger, B. Laskowski, and E. Brändas, *J. Chem. Phys.* **65**, 4559 (1976).
9. M. Hehenberger, P. Froelich, and E. Brändas, *J. Chem. Phys.* **65**, 4571 (1976).
10. M. Hehenberger, *J. Chem. Phys.* **67**, 1710 (1977).
11. E. Brändas, M. Rittby, and N. Elander, *J. Math. Phys.* **26**, 2648 (1985).
12. E. Engdahl and E. Brändas, *J. Math. Phys.* **27**, 2629 (1986).
13. E. Engdahl, E. Brändas, M. Rittby, and N. Elander, *Phys. Rev. A* **37**, 3777 (1988).
14. E. Engdahl and E. Brändas, *Phys. Rev. A* **37**, 4145 (1988).
15. G. Dahlquist and A. Bjørck, *Numerical Methods* (Prentice-Hall, Englewood Cliffs, NJ, 1974).

16. M. Hehenberger, "Numerical Aspects of Weyl's theory," *Quantum Science Methods and Structure*, edited by Calais, Goscinski, Linderberg, and Ohm (Plenum, New York, 1976).
17. J. D. Pryce, Technical report, Royal Military College of Science (Cranfield), 1990 (unpublished).
18. A. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, (Dover, New York, 1968).
19. D. E. Amos, *ACM Trans. Math. Software* **12**, 265 (1986).
20. M. Marletta and J. D. Pryce, *Comput. Phys. Commun.* **62**, 42 (1991).
21. M. Marletta, Ph.D. thesis, Royal Military College of Science (Cranfield), 1991 (unpublished).
22. S. Pruess, *SIAM J. Numer. Anal.* **10**, 55 (1973).
23. S. Pruess, *Numer. Math.* **24**, 241 (1975).
24. M. Marletta, Technical Report ACM 90-14, Royal Military College of Science (Cranfield), 1990 (unpublished).