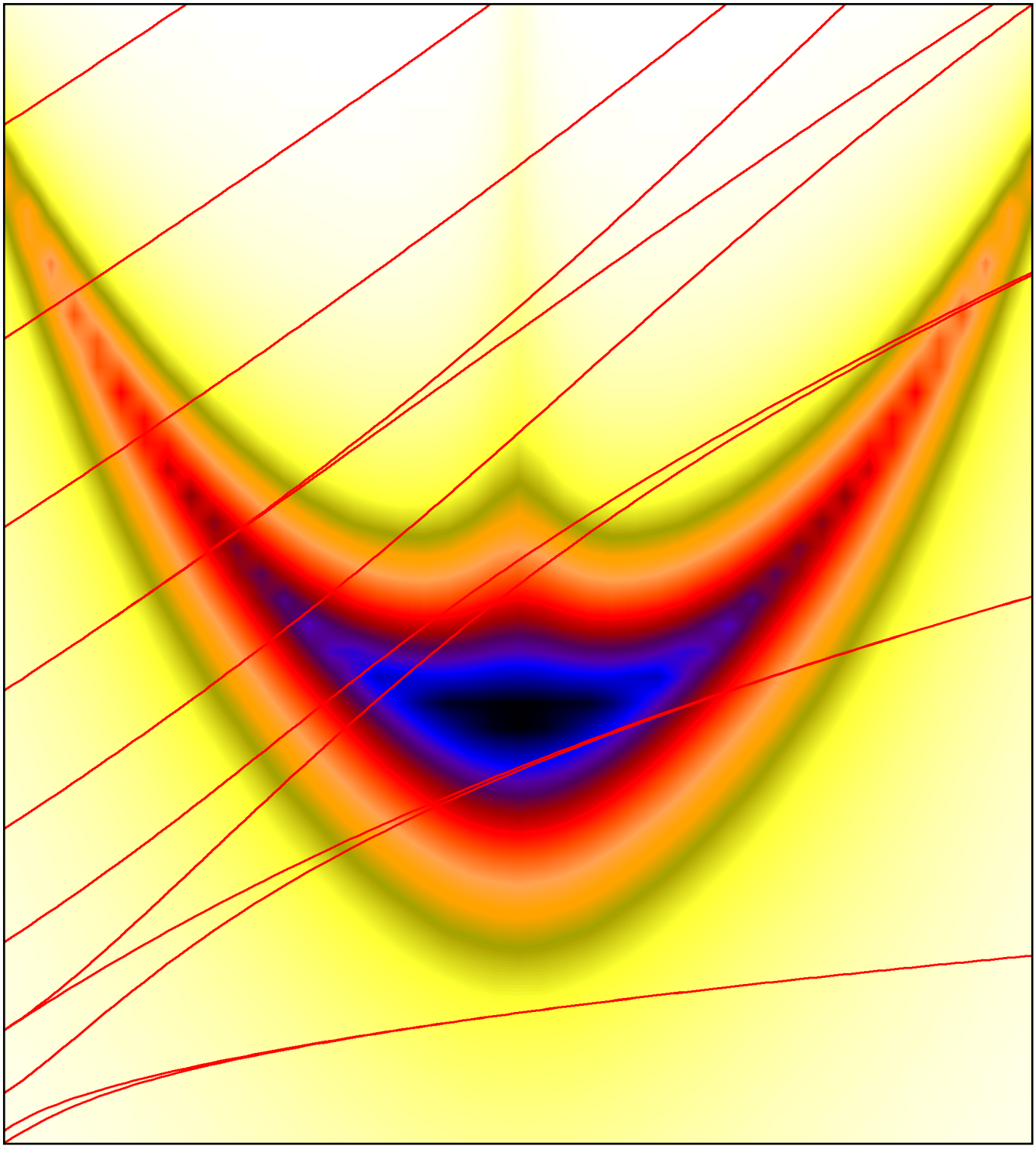
SIMPLE QUANTUM VISUALIZATIONS USING IDL



Rob Dimeo
NIST Center for Neutron Research

SIMPLE QUANTUM VISUALIZATIONS USING IDL

Robert M. Dimeo

NIST Center for Neutron Research
National Institute of Standards and Technology
Gaithersburg, Maryland

DRAFT COPY

November 26, 2008

Disclaimer
Opinions expressed herein are those of the author and should not be interpreted as opinions of NIST or of the NIST Center for Neutron Research. NIST assumes no responsibility whatsoever for use of this software or documentation, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic. The use of certain trade names or commercial products does not imply any endorsement of a particular product, nor does it imply that the named product is necessarily the best product for the stated purpose.

Contents

Contents i

Preface iii

Acknowledgments v

1 Introduction 1

1.1 Topics Covered in the Book . . . 2

1.2 Selected Results from Quantum Mechanics . . . 2

1.3 IDL Utilities . . . 5

2 Stationary States 13

2.1 The Discrete Variable Approximation in One Dimension . . . 13

2.2 The Discrete Variable Approximation in Two Dimensions . . . 24

2.2.1 The Stadium Potential . . . 27

2.2.2 The Cardioid Potential . . . 32

3 Quantum Rotations 37

3.1 Hindered Methyl Rotations . . . 37

3.2 Determining the Hamiltonian Matrix . . . 38

3.2.1 Solving the Hamiltonian . . . 39

3.2.2 Transitions . . . 43

3.3 Hindered Dumbell Rotations . . . 46

4 Quantum Dynamics in One Dimension 53

4.1 The Goldberg-Schey-Schwartz Algorithm . . . 53

4.1.1 Development of the GSS Algorithm . . . 54

4.1.2 Implementation . . . 56

4.1.3 Stability and Selection of Parameters 57
4.1.4 Getting the Units Right with the GSS Algorithm 58
4.2 Scattering from a High Step Potential 59
4.3 Scattering from a Square Barrier 62
4.4 Scattering from a Square Well 65
4.5 Scattering from a Periodic Potential: Bragg's Law 70
4.6 Scattering from a Well with a Lip: Metastable/Virtual States 73

5 Quantum Dynamics in Two Dimensions 79
5.1 The Algorithm of Askar and Cakmak 79
5.2 Two-Dimensional Scattering 81
5.2.1 Self-Interference: Transmission through Double-Slits 83
5.2.2 Dynamics of Quantum Billiards 85
5.2.3 Dynamics of a Wavepacket in Archimedes' Spiral 88
5.3 Colliding Particles 92
5.3.1 Colliding Billiard Balls 95
5.3.2 Scattering from an Harmonic Oscillator 98

6 Quantum Visualizations 103

A The Wigner 3-j Symbol 115

B The Simple Harmonic Oscillator Eigenfunctions 117

Answers to Selected Exercises 119

Index 122

Preface

Man is most nearly himself when he achieves the seriousness of a child at play.

—Heraclitus

One of the biggest frustrations I had as a graduate student studying physics was my inability to visualize one of the phenomena that I spent years studying. At the time in the mid 90's, there were only a handful of us investigating the collective excitations of superfluid helium in confined geometries—and it's probably a safe bet that none of us had any idea (beyond some very simple-minded views) what it was we were *seeing* with the tools of neutron scattering. And if pressed, our response might involve mumbling something about "density fluctuations." I recall spending a great deal of time trying to understand what a *collective excitation* looked like. Though we all learned in our undergraduate solid-state physics courses what a phonon looked like in terms of lots of sloshing about of many masses connected with many springs, it was Feynman himself who provided the first qualitative glimpse of what a collective excitation in superfluid helium looked like. Specifically he described the helium excitation known as a *roton* as a really tiny smoke ring—a microscopic quantized vortex ring. Though quantum vortices *were* eventually discovered in superfluid helium, these were *not* the rotons. Nevertheless Feynman used his intuition and a very deep knowledge of quantum mechanics to construct an early mental picture of a roton. And the point is that the visualization process is individual and relies on the creativity of the individual.

It was not long after getting my Ph.D. that I began research on a class of systems that were quite accessible to calculation and visualization. I built my early research program investigating the quantum dynamics of small molecules in solids that can rotate about a single axis yet are hindered from rotating freely. I learned that these *hindered rotors* could be modeled simply as one-dimensional single particles and the dynamics could be obtained by solving the Schrödinger equation for a very simple potential. With the frustration of my graduate school days still fresh, I became intensely interested in trying to develop a mental picture—and a real picture—of the quantum rotational dynamics for the systems that I was studying using neutron scattering. Thus began my entry into computational methods and visualization.

I have been involved in software development at the NIST Center for Neutron Research since 1999, working with staff to develop data reduction, visualization and analysis programs for neutron scattering experiments. During this period I developed programs that have been used for analyzing neutron scattering data both for my own basic research purposes and even for education and outreach activities. The main programs presented in this book are based on programs that were developed for one or both of those purposes.

I had a number of goals in writing this book. My main goal was to provide a useful resource for staff at the NCNR (and elsewhere) in developing software for analyzing their neutron scattering

data, or perhaps even developing end-user applications for the growing NCNR user program. Oftentimes the analysis of neutron scattering data requires one to perform some form of quantum mechanical calculation and then use the results to interpret the phenomena being investigated. However as I proceeded to write down the main ideas it became clear that the material is more general than just neutron scattering.

This book is *not* intended to be used a textbook in IDL [1] programming or quantum mechanics. Rather it can be used as supplement to a quantum mechanics course or even as a vehicle for a researcher to wade gently into IDL programming in a topic with which he/she is already familiar before embarking on a specific research application involving computing. Nevertheless some background in both areas is assumed. In particular this book is intended for use by individuals who have had at least an upper-division quantum mechanics course [2, 3, 4] and can program in IDL at the beginner-to-intermediate level.[6, 7]. There are numerous excellent texts on quantum mechanics and many manuals are available for learning IDL. IDL programming topics that readers are expected to have some working knowledge of include pointers, structures, procedures, functions, and simple plots using IDL's direct graphics system (i.e. it is assumed that the reader has programmed using these constructs). No widget or object programming is required.

There are excellent texts that contain many visualizations.[3] This book is geared toward the individual that wishes to create his/her own visualizations, explore the effects of different potentials, experiment with different manners of viewing the data, and is not afraid to modify existing code or write their own programs for their own purposes. This book is meant to be used in a hands-on style—the reader is expected to run the code presented within as well as try the exercises that are scattered throughout the chapters. These exercises are meant to extend the ideas that have already been presented and entice the reader to apply the code to different situations. It is hoped that the programs presented herein are a starting point for the reader to perform his/her own calculations. All of the programs are available for download from the author's website[8].

The topics covered in this book are intended as a starting point for the reader to extend in his/her own investigations into quantum phenomena. More than anything else—and especially in the spirit of Heraclitus' sentiment—the material in this book is meant to arouse the reader's curiosity and creativity and allow him/her to take the next step by creating his/her own meaningful computations and visualizations.

Rob Dimeo

Acknowledgments

The software in this book is the result of time spent learning IDL programming from many outstanding resources. The individuals who have helped me at one point or another include David Fanning, Beau Legeer, Mark Piper, Mike Galloy, Ronn Kling, Craig Markwardt, John Copley, Richard Azuah, Larry Kneller, Alan Munter, and Yiming Qiu. My understanding and intense interest and enthusiasm for quantum mechanics is attributed to a number of Physics faculty as well as current colleagues. These individuals include Rick Robinett, Murat Gunaydin, Roy Willis, Paul Sokol, Dan Neumann, Craig Brown, Jack Rush, and the late Sam Trevino.

This work was done in support of the DAVE software development project at the NIST Center for Neutron Research and it is based upon activities supported by the National Science Foundation under Agreement No. DMR-0454672.

Chapter 1

Introduction

The more I think of the physical part of the Schrödinger theory, the more detestable I find it. What Schrödinger writes about visualization makes scarcely any sense, in other words I think it is shit. The greatest result of his theory is the calculation of matrix elements.

—Werner Heisenberg in a letter to Wolfgang Pauli (June 8, 1926)

Heisenberg's skepticism of Schrödinger's theory notwithstanding, I think it is safe to assume that we've come a long way in over 80 years. While imagination and a very deep intuition of mathematics lay at the heart of early attempts to assign physical meaning (i.e. visualize) quantum mechanics, the latter half of the 20th century saw ever-increasing computing power brought to bear on quantum phenomena. Arguably one of the most interesting applications of scientific computing is visualizing phenomena that were once thought to be beyond the limits of human perception. Such esoteric phenomena as quantum tunneling are now commonplace in industrially-relevant and standard technologies. The development of today's technologies required a deep understanding of atomic-scale phenomena in order to design at these scales (i.e. nanotechnology). No doubt the trend towards smaller scale designs will continue where quantum mechanics manifests itself and therefore it is more important than ever that scientists and engineers cultivate a *common sense* and intuition regarding quantum phenomena. Indeed modern texts on quantum mechanics emphasize the role of visualization in developing such intuition.[2, 3]

Computer visualization methods have grown in popularity as a means to gain an intuition about quantum phenomena. The substantial increase in computational power of desktop computers has enabled researchers to perform very sophisticated numerical computations in the comfort of their own offices or homes. High-level programming languages that emphasize matrix manipulations such as IDL and MATLAB have lowered the barrier to novice users (e.g. students) and permit computation and visualization of phenomena that can be performed relatively quickly both from a programming and software development perspective as well as in terms of processor time.[1, 5] The first point is important, particularly for students. High-level programming languages now feature fulsome command syntax that is very close to mathematical syntax, making it even easier for a student to transition from the equations in a textbook to computation. The second point, that improvements in processors have reduced the time to perform many complex calculations, has made investigations into complex scientific phenomena more realizable on conventional desktop computers. The combination of these two factors has resulted in making education and research involving computation of modern quantum phenomena more accessible to a wider range of individuals. In

this book we will show how one can use one of these high-level graphics and numerical manipulation packages, IDL, to solve standard (and not-so-standard) problems from quantum mechanics and visualize the results. Though the development of the algorithms is motivated strongly through the use of quantum mechanical and mathematical arguments, use of the algorithms requires far less sophistication. It is hoped that the simplicity of implementation will foster exploration.

1.1 Topics Covered in the Book

The topics from quantum mechanics covered in this book are organized to some extent in order of increasing complexity. The material begins with a presentation of the discrete-variable approximation to solve the time-independent Schrödinger equation for a one-dimensional potential. Dynamics are introduced early and motivated through the time-development of a state composed of two eigenstates. The phenomenon of quantum tunneling is presented as an illustration of the time-development of a quantum state initially localized in one side of a parabolic well separated by a barrier. The discrete-variable method is extended to two dimensions and applied to a class of closed potentials and the stationary-states are computed for systems known as *quantum billiards*. The next chapter continues the discussion of solutions to the time-independent Schrödinger equation but covers the phenomenon of quantum rotations. The new wrinkle in the discussion is that the potentials are periodic. The single-particle motion of a three-fold rotor is discussed first and serves as an excellent example of a one-dimensional system with periodic boundary conditions. It illustrates the phenomena of quantum tunneling and quantum *librations* (i.e. torsional oscillations) which are observed in real physical systems. The chapter concludes with a presentation of quantum rotations of a dumbell rotor, both free and hindered. The results of those computations are applicable to the dynamics of H_2 adsorbed onto surfaces. Next the method of Goldberg, Schey, and Schwartz is described to solve the time-dependent Schrödinger equation in one dimension.[9] A remarkably short function is presented that performs the computation of the time-evolution of the wavefunction subject to a user-defined potential. This enables one to construct striking animations of the space-time and momentum-time perspectives of scattering phenomena. In chapter 5, a method to solve the time-dependent Schrödinger equation in two dimensions is presented and applied to two situations. First, we examine a single particle in two dimensions (represented by a Gaussian wavepacket) evolving under the influence of a two-dimensional potential. Remarkable phenomena become visually accessible with this capability such as a single particle passing through two slits, hence *interfering with itself*. The second application of this two-dimensional algorithm is to compute the consequences of two one-dimensional particles (Gaussian wavepackets) interacting with each other. Specific examples include observing the effects of a Gaussian wavepacket scattering from a harmonic oscillator through a repulsive interaction potential. In the final chapter we present a series of color images of quantum scattering phenomena, many of which are alternative representations of examples presented in previous chapters.

1.2 Selected Results from Quantum Mechanics

This section is not a primer on quantum mechanics nor is it a complete summary of all of the results from quantum mechanics. Rather we present a few of the main equations and quantities that will be used in this book. The reader is assumed to have a working knowledge of the equations in this section. See, for instance, the excellent description of this material in the text by Robinett.[2]

The equation of motion for matter on the atomic scale, specifically for a material particle of

mass m, is the time-dependent Schrödinger equation which is given (in Dirac's bra-ket notation) by

$$H|\psi\rangle = i\hbar\frac{\partial}{\partial t}|\psi\rangle, \tag{1.1}$$

where $|\psi\rangle$ is the probability amplitude that describes the quantum mechanical motion of the system. In position space, the Hamiltonian operator H is given by

$$H = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x), \tag{1.2}$$

where m is the mass and the Hamiltonian operator acts on the complex wavefunction $\langle x|\psi\rangle$, also sometimes written as $\psi(x,t)$. For the remainder of this book we will consider time-independent potentials only $V(x,t) = V(x)$. Moreover we will use $\psi(x,t)$ and $\langle x|\psi\rangle$ interchangeably.

An observable quantity that we can obtain once we know the wavefunction is the probability density

$$P(x,t) = \left|\langle x|\psi\rangle(t)\right|^2. \tag{1.3}$$

which is the probability of finding the particle at time t within a range dx of x.

For a time-independent potential, there are special solutions, the stationary states, of the Schrödinger equation that satisfy an eigenvalue equation

$$H|E\rangle = E|E\rangle, \tag{1.4}$$

where $|E\rangle$ is the eigenfunction corresponding to the eigenvalue E and H is given by equation 1.2 in one dimension. That the eigenstates are stationary means that if the state of the quantum system is prepared in one of these eigenstates, it will stay there forever unless perturbed. If a state is composed of a number of eigenstates then the state is time-dependent due to transitions among the eigenstates present in the initial state. As an example, consider a state composed of an equal mixture of the two lowest energy eigenstates

$$\langle x|\psi\rangle(0) = \frac{1}{\sqrt{2}}\left(\langle x|E_0\rangle + \langle x|E_1\rangle\right). \tag{1.5}$$

Applying the time-development operator on this state, $U(t,0) = exp(-iHt/\hbar)$, yields a state that evolves with the following time dependence

$$\langle x|\psi\rangle(t) = \frac{1}{\sqrt{2}}\left(e^{-iE_0t/\hbar}\langle x|E_0\rangle + e^{-iE_1t/\hbar}\langle x|E_1\rangle\right). \tag{1.6}$$

In chapter 4 we will be investigating extensively wavepackets that scatter from various potentials. In our visualizations of the scattering process it will be useful to represent the energy of the wavepacket in some manner that allows us to compare it to the potential from which it scatters. We will find it useful therefore to calculate the kinetic energy of the wavepacket and offset the wavepacket by that amount. The kinetic energy is found by calculating the integral

$$\langle T\rangle = \frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\mathrm{d}x\left|\frac{\partial\langle x|\psi\rangle}{\partial x}\right|^2. \tag{1.7}$$

In the spatial domain the probability current density is defined by

$$j(x,t) \equiv \frac{\hbar}{2mi}\left(\langle\psi|x\rangle\left(\frac{\partial}{\partial x}\langle x|\psi\rangle\right) - \left(\frac{\partial}{\partial x}\langle\psi|x\rangle\right)\langle x|\psi\rangle\right). \tag{1.8}$$

This quantity is real and observable. As semiconductor features shrink (and micro-electronics tends to nano-electronics), it becomes increasingly important to consider the quantum mechanical transport properties of electrons. It can be quite useful then to calculate the probability current density for models of such nano-electronic structures.

It is possible to view the quantum state of a system in a representation other than the position-space representation. Often it can be more convenient to obtain detailed information on the momentum content of a wavefunction by examining the momentum space wavefunction. In fact there are occasions when the interpretation of certain physical phenomena is simplified when examined in the momentum domain. The momentum-space wavefunction is obtained through Fourier transformation of the position-space wavefunction

$$\langle p|\phi\rangle = \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} dx\ e^{-ipx/\hbar} \langle x|\psi\rangle\,(t)\,. \tag{1.9}$$

Note that $\phi(p,t)$ and $\langle p|\phi\rangle$ are used interchangeably. The momentum distribution

$$P\,(p,t) = \left|\langle p|\phi\rangle\,(t)\right|^2 \tag{1.10}$$

is an obervable, so we will often be interested in calculating it. It is the probability that we will find the particle at time t with momentum within a range dp of p. Note that we will often use $\langle k|\phi\rangle$ and $\langle p|\phi\rangle$ interchangeably since we know from the de Broglie relations that $p = \hbar k$ where the wavevector $k \equiv 2\pi/\lambda$.

In two dimensions the Schrödinger equation in cartesian coordinates is given by

$$H = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) + V\,(x,y) \tag{1.11}$$

which acts on the wavefunction $\langle x,y|\psi\rangle$. Note that this is also related closely to the Hamiltonian for two interacting particles in one dimension. If we label the position coordinate for particle 1 by x_1 and that of particle 2 by x_2 and define the interaction between the two particles as $V\,(x_1,x_2)$ then the two-particle Hamitonian is given by

$$H = -\hbar^2\left(\frac{1}{2m_1}\frac{\partial^2}{\partial x_1^2} + \frac{1}{2m_2}\frac{\partial^2}{\partial x_2^2}\right) + V\,(x_1,x_2)\,. \tag{1.12}$$

Similarly this Hamiltonian acts on the two-particle wavefunction $\langle x_1,x_2|\psi\rangle$. Strictly speaking, the two-particle wavefunction "entangles" the two particles in such a way that does not permit one *in general* to disentangle the behavior of a single particle upon solving for the two-particle wavefunction. If there were no correlation between the two particles—which is not the case for interactions between two particles—then the two-particle wavefunction could be factored due to their independence (note that this would be the case if the potential could be separated into two terms, i.e. $V(x_1,x_2) = V_1(x_1)+V_2(x_2)$). This is not the case in general. Nevertheless we can project out the single-particle densities $\rho_1\,(x_1)$ and $\rho_2\,(x_2)$ in an effort to view the single-particle motion. This approach is exact for separable potentials. However it is also a reasonable approximation for times long before and long after an interaction. The expressions for the single-particle densities are

$$\rho_1\,(x_1) = \int_{-\infty}^{\infty} \mathrm{d}x_2 \left|\langle x_1,x_2|\psi\rangle\right|^2 \tag{1.13}$$

and

$$\rho_2\,(x_2) = \int_{-\infty}^{\infty} \mathrm{d}x_1 \left|\langle x_1,x_2|\psi\rangle\right|^2. \tag{1.14}$$

1.3 IDL Utilities

[This section should be considered optional and can be skipped, especially on reading through for the first time.]

Throughout this book we will require a number of IDL routines (functions and procedures) to perform basic tasks. There are some 3rd party IDL routines that are used in the programs in this book and readers should download these from the corresponding websites. `PLOTIMAGE` from Craig Markwardt's IDL library is among those.[10] Also some general-purpose utilities available from David Fanning's Coyote Library, an excellent web-based resource[7], are used. His image display function `TVIMAGE` is used in many plots presented here, his color function `FSC_COLOR` is used in some places (also requiring his programs `PICKCOLORNAME` and `ERROR_MESSAGE`), as is `INSIDE` which determines if a point is located within a polygon. This will be particularly useful in the implementation of two dimensional potentials. These routines are not discussed in any detail in this book or in the program listings. Those details can be obtained from the websites listed above. There are a number of utilities that will be used extensively in programs presented in subsequent chapters of this book and these *are* listed in this section. Where appropriate, details of the implementation are presented.

In this section we will present a number of functions and procedures that were written especially for the purpose of visualizing data for this book. These will simply presented without much explanation except in a few cases. It is assumed (1) that the reader knows enough IDL to be able to understand these routines, and (2) the reader has put these routines (and all programs used in this book) in his/her IDL path so that the programs are able to compile the routines at runtime.

The first utility is `LINSPACE`. This function creates an equally-spaced vector of N_x points between `xlo` and `xhi`. The function listing is shown below.

```
function linspace,xlo,xhi,nx
compile_opt idl2,hidden
dx = (xhi-xlo)/(nx-1.0)
return,xlo+dx*findgen(nx)
end
```

Another function that we will find useful, particularly when calculating matrix elements of a Hamiltonian, is the Kronecker delta function. This function is defined as follows:

$$\delta_{i,j} = \begin{cases} 0 & \text{for } i \neq j \\ 1 & \text{for } i = j. \end{cases} \tag{1.15}$$

Implementation of vector and matrix calculations in IDL is relatively straightforward. In fact, it is one of the advantages of using a language such as IDL—matrix and vector calculations can be written in a compact form. For example, the Kronecker delta function defined in (1.15) can be written in essentially one line: `i eq j` where i and j are vectors or matrices. Let's consider first a vector of indices of length 4, `ivec=indgen(4)`. Converting (or inflating) to a form of a 4×4 matrix is done easily using the `REBIN` function: `i=REBIN(ivec,4,4,/sample)` which results in the

matrix

$$i = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{pmatrix}. \tag{1.16}$$

Next let `j=TRANSPOSE(i)` such that,

$$j = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{pmatrix}. \tag{1.17}$$

By inspection we can see that the diagonal elements of these two matrices are equal so that

$$\delta_{i,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1.18}$$

Our IDL implementation of the Kronecker delta function is listed below:

```
function ds_delta,i,j
compile_opt idl2,hidden
return,i eq j
end
```

We can execute the following commands to illustrate how this Kronecker delta function works.

```
IDL> ivec = indgen(4)
IDL> i = rebin(ivec,4,4,/sample)
IDL> j = transpose(i)
IDL> print,ds_delta(i,j)
   1   0   0   0
   0   1   0   0
   0   0   1   0
   0   0   0   1
```

A visualization utility that displays an animated sequence based on a user-specified transition between eigenstates is given in the IDL function named `ANIMATE_TRANSITION_PROB`. The input parameters are the position vector `x`, the potential vector `v` defined at `x`, the complex wavefunctions `wf` (a two dimensional complex array in which the first dimension is the spatial dimension and the second dimension specifies which eigenvalue in ascending order), the eigenvalue array `eigvals` given in ascending order, and the identification of the two eigenstates `n` and `m`. Note that in this nomenclature `wf[*,5]` corresponds to the sixth eigenvalue or `eigvals[5]`.

```
function animate_transition_prob,x,v,wf,eigvals,n,m
```

```
compile_opt idl2,hidden
; Displays the animation of the probability density
; corresponding to an initial wavefunction composed
; of two eigenstates <x|n> and <x|m> of the potential
; v.
e0 = eigvals[n] & e1 = eigvals[m]
w0 = e0/0.6528 & w1 = e1/0.6528
wf0 = wf[*,n] & wf1 = wf[*,m]
period = 2.*!pi/abs(w0 - w1)
; Set up the time discretization
nt = 300 & dt = period/(nt-1.0)
t = dt*findgen(nt)  ; time in pico-seconds
; Construct the time-dependent probability density
i = complex(0.0,1.0)
psi = (1./sqrt(2.0))*(wf0#(exp(-i*w0*t))+wf1#(exp(-i*w1*t)))
prob = abs(psi)^2
; Plot the animated sequence using a pixmap window
; so that the animation appears smooth.
angstrom = '!6!sA!r!u!9 %!6!n'
xtitle = '!6x ( '+angstrom+' )'
title = '!6P(x,t)'
winvis = 0
xsize = 500 & ysize = 400
window,0,xsize = xsize,ysize = ysize
window,/free,/pixmap,xsize = xsize,ysize = ysize
winpix = !d.window
sf = max(v)/max(prob)
for j = 0,nt-1 do begin
  wset,winpix
  plot,x,v,thick = 2.0,color = fsc_color('black'), $
    background = fsc_color('white'), $
    xtitle = xtitle,ytitle = '!6V (meV)',title = title, $
    yrange = [5.0,25.0],/ystyle,/nodata
  oplot,x,v,color = fsc_color('blue'),thick = 4.0
  oplot,x,e0+0.1*sf*prob[*,j],color = fsc_color('red'),thick = 2.0
  plots,!x.crange,[e0,e0],/data,linestyle = 2,$
    color = fsc_color('black')
  wset,winvis
  device,copy = [0,0,xsize,ysize,0,0,winpix]
endfor
wdelete,winpix
return,1B
end
```

Once we calculate the position space wavefunction $\psi(x,t)$ using one of our algorithms, it is also instructive to look at the momentum space wavefunction $\phi(k,t)$ or the momentum space probability density $|\phi(k,t)|^2$. It is straightforward to compute the momentum space wavefunction in IDL using the Fast-Fourier Transform (FFT). The function named `MOMENTUM_TRANSFORM`

performs this calculation for you, given the position coordinates, time vector, and position-space wavefunction defined for all `x` and `t`. This function returns a structure variable `str` whose fields are the `str.phi_k` and `str.k`. Note that the wavefunction in k-space $\phi(k,t)$ that is returned is properly normalized, (i.e. $\int_{-\infty}^{\infty} |\phi(k,t)|^2 = 1$).

```
function momentum_transform,x,t,wf
compile_opt idl2,hidden
; This function uses the FFT to compute the
; k-space wavefunction based on the position-
; space wavefunction.  The function returns a
; structure variable with a wavefunction field
; and a k field.
nx = n_elements(x) & nt = n_elements(t)
wfsize = size(wf)
dx = x[1] - x[0] & n21 = nx/2 + 1
k = indgen(nx)
k[n21] = n21-nx+findgen(n21-2)
k = 2.*!pi*k/(nx*dx) & k = shift(k,-n21)
if wfsize[0] eq 1 then begin
  ; Normalize the wavefunction
  c = int_tabulated(x,abs(wf)^2) & wf = wf/sqrt(c)
  phi_k = nx*fft(wf,-1)
  phi_k = shift(phi_k,-n21)
  ck = int_tabulated(k,abs(phi_k)^2)
  phi_k = phi_k/sqrt(ck)
endif else begin
  phi_k = complexarr(nx,nt)
  for j = 0,nt-1 do begin
    c = int_tabulated(x,abs(wf[*,j])^2)
    wf[*,j] = wf[*,j]/sqrt(c)
    phi_k[*,j] = nx*fft(wf[*,j],-1)
    ck = int_tabulated(k,abs(phi_k[*,j])^2)
    phi_k[*,j] = phi_k[*,j]/sqrt(ck)
  endfor
  phi_k = shift(phi_k,-n21,0)
endelse
str = {phi_k:phi_k,k:k}
return,str
end
```

In future chapters we will be calculating the time evolution of a probability density corresponding to a complex wavefunction. A function called `ANIMATE_PROBABILITY` has been written and is listed below. This is a simple utility whose input are the vectors `x`, `t`, and `v`, and the complex array `psi`.

```
function animate_probability,x,t,v,psi,offset = offset
```

```
compile_opt idl2,hidden
if n_params() ne 4 then return,0B
if n_elements(offset) eq 0 then offset = 0.0
nt = n_elements(t) & device,decomposed = 0
winvis = 0 & xsize = 500 & ysize = 400
window,winvis,xsize = xsize,ysize = ysize
window,/free,/pixmap,xsize = xsize,ysize = ysize
winpix = !d.window
vhi = max(v,min = vlo)
vmax = max([abs(vhi),abs(vlo)])
prob = abs(psi)^2 & sf = vmax/max(prob)
vlo = min(v,max = vhi) & dv = vhi-vlo
yr = [-vmax-0.2*dv,vmax+0.25*dv+offset] & xr = [min(x),max(x)]
xtitle = '!6X' & title = '!3|!6<X!3|!7W!6>(T)!3|!e2!n!6'
ytitle = '!7H'
for j = 0,nt-1 do begin
  wset,winpix
  plot,x,v,xrange = xr,yrange = yr,/xsty,/ysty,color = fsc_color('black'),$
    background = fsc_color('white'),xtitle = xtitle,title = title,$
    ytitle = ytitle,charsize = 1.5
  oplot,x,offset+sf*prob[*,j],thick = 2.0,color = fsc_color('red')
  wset,winvis
  device,copy = [0,0,xsize,ysize,0,0,winpix]
endfor
wdelete,winpix
return,1B
end
```

Note that there is an input keyword named `OFFSET`. When this is set to a value, the probability density will be offset from the x-axis by that amount. Often we would like for that offset value to be the kinetic energy of the wavepacket so that we can see how the wavepacket's energy compares with the potential from which it is scattering. A function called `TDSE_CALC_ENERGY` has been written that performs this calculation based on the discretized wavefunction. The equation used to calculate the kinetic energy from the wavefunction is

$$\langle T\rangle = \frac{\hbar^2}{2m}\int_{-\infty}^{\infty}\mathrm{d}x\left|\frac{\partial\psi}{\partial x}\right|^2. \tag{1.19}$$

```
function tdse_calc_energy,x,wf
compile_opt idl2,hidden
nx = n_elements(x) & dx = x[1] - x[0]
; Normalize the wavefunction
c = int_tabulated(x,abs(wf)^2) & wf = wf/sqrt(c)
wf_re = real_part(wf) & wf_im = imaginary(wf)
dwredx = 0.5*(wf_re[2:nx-1]-wf_re[0:nx-3])/dx
dwimdx = 0.5*(wf_im[2:nx-1]-wf_im[0:nx-3])/dx
energy = total(abs(dwredx)^2+abs(dwimdx)^2)*dx
```

```
return,energy
end
```

Likewise, a function called ANIMATE_WAVEFUNCTION has been written to display the real and imaginary components of the complex wavefunction.

```
function animate_wavefunction,x,t,v,psi,offset = offset
compile_opt idl2,hidden
if n_params() ne 4 then return,0B
if n_elements(offset) eq 0 then offset = 0.0
nt = n_elements(t)
winvis = 0 & xsize = 500 & ysize = 400
window,winvis,xsize = xsize,ysize = ysize
window,/free,/pixmap,xsize = xsize,ysize = ysize
winpix = !d.window
sf = max(v)/max(abs(psi))
psi_re = real_part(psi) & psi_im = imaginary(psi)
psi_env = abs(psi)
vlo = min(v,max = vhi) & dv = vhi-vlo
yr = [-max(v),vhi+0.25*dv+offset] & xr = [min(x),max(x)]
device,decomposed = 0
xtitle = '!6X' & title = '!6<X!3|!7W!6>(T)'
ytitle = '!7H'
for j = 0,nt-1 do begin
  wset,winpix
  plot,x,v,xrange = xr,yrange = yr,/xsty,/ysty,color = fsc_color('black'),$
    background = fsc_color('white'),xtitle = xtitle,title = title,$
    ytitle = ytitle,charsize = 1.5
  oplot,x,offset+sf*psi_re[*,j],thick = 2.0,color = fsc_color('red')
  oplot,x,offset+sf*psi_im[*,j],thick = 2.0,linestyle = 2,color = fsc_color('red')
  oplot,x,offset+sf*psi_env[*,j],thick = 2.0,color = fsc_color('red')
  oplot,x,offset-sf*psi_env[*,j],thick = 2.0,color = fsc_color('red')
  wset,winvis
  device,copy = [0,0,xsize,ysize,0,0,winpix]
endfor
wdelete,winpix
return,1B
end
```

The final utility that displays an animation of the time-evolution of a complex wavefunction is ANIMATE_PHASORS. This function displays a phasor representation of the complex valued wavefunction as a function of position. The phasor concept is commonly applied in the analysis of electrical circuits containing capacitive and inductive elements. Additionally they have been used to graphically construct the resulting interference pattern from light waves that pass through slit systems.

The concept is simple. Wave phenomena can be represented in the complex plane as an arrow whose length is equal to the magnitude of the wave at that point in space and at that time. The orientation of the arrow depends on the real and imaginary components at that point in space and at that time.

Consider the complex wavefunction,

$$\psi(x,t) = \psi_{re}(x,t) + i\psi_{im}(x,t) \tag{1.20}$$

where ψ_{re} and ψ_{im} are both real functions. Also note that

$$\begin{aligned} |\psi(x,t)|^2 &= \psi_{re}^2(x,t) + \psi_{im}^2(x,t) && (1.21) \\ &= \psi_{mag}^2(x,t). && (1.22) \end{aligned}$$

We can rewrite the wavefunction as

$$\psi(x,t) = \psi_{mag}(x,t)e^{i\phi(x,t)} \tag{1.23}$$

where the phase $\phi(x,t)$ is given by

$$\phi(x,t) = \tan^{-1}\left(\frac{\psi_{im}(x,t)}{\psi_{re}(x,t)}\right). \tag{1.24}$$

The graphical picture of this representation, equation 1.23, is that of an arrow whose tail is at the origin, whose head is pointing in a direction specified by $\phi(x,t)$ measured counter-clockwise from the positive x-axis, and rotating because it is a time-dependent quantity.

```
function animate_phasors,x,t,v,psi,wait = wait,offset = offset
compile_opt idl2,hidden
if n_params() ne 4 then return,0B
if n_elements(wait) eq 0 then wait = 0.0
if n_elements(offset) eq 0 then offset = 0.0
nt = n_elements(t) & nx = n_elements(x)
winvis = 0 & xsize = 500 & ysize = 400
window,winvis,xsize = xsize,ysize = ysize
window,/free,/pixmap,xsize = xsize,ysize = ysize
winpix = !d.window
sf = max(v)/max(abs(psi))
psi_re = real_part(psi) & psi_im = imaginary(psi)
psi_env = abs(psi)
psi_max = max(psi)
vlo = min(v,max = vhi) & dv = vhi-vlo
yr = [-max(v),vhi+0.25*dv+offset] & xr = [min(x),max(x)]
device,decomposed = 0
xtitle = '!6X' & title = '!6<X!3|!7W!6>(T)'
ytitle = '!7H'
for j = 0,nt-1 do begin
  wset,winpix
  pmag = abs(psi[*,j])/psi_max
  xlimin = where((pmag ge 0.005),count); and (x ge xr[0]) and (x le xr[1]),count)
  if xlimin[0] eq -1 then return,0B
```

```
  in = xlimin[where(~xlimin mod 3)]
  plot,x,v,xrange = xr,yrange = yr,/xsty,/ysty,color = fsc_color('black'),$
    background = fsc_color('white'),xtitle = xtitle,title = title,$
    ytitle = ytitle,charsize = 1.5
  arrow,x[in],offset+replicate(0.0,nx), $
     x[in]+1.0*sf*psi_re[in,j],$
     offset+1.0*sf*psi_im[in,j],$
     color = fsc_color('black'),thick = 1.0,/data,solid = 0B
  wset,winvis
  device,copy = [0,0,xsize,ysize,0,0,winpix]
  wait,wait
endfor
wdelete,winpix
return,1B
end
```

Chapter 2

Stationary States

A hallmark of a particle confined within some geometry is the existence of discrete–or quantized–energy levels. There are a number of simple systems that exhibit quantized energy levels whose solutions can be obtained analytically. These include the infinite square well, the finite square well, the linear well, and the parabolic potential (a.k.a. the simple harmonic oscillator). These idealized systems are usually treated in elementary quantum mechanics courses, and although they are idealized, they *have* been realized in practical physical systems. For example see [12] for a discussion of the applicability of a simple potential that can be used to model the energy levels of charge carriers confined by quantum dot structures. Nevertheless, for all but the simplest of systems and potentials, it is necessary to use some type of numerical algorithm to determine the energy levels and corresponding wavefunctions (eigenvalues and eigenfunctions).

One of the simplest methods to obtain eigenvalues and the stationary state wavefunctions of the Schrödinger equation for a static potential is the discrete variable approximation (DVA), a method that appeared early in the literature.[11] In this chapter we will describe the mathematics that underlies this method and provide the details on how this algorithm is implemented in a short IDL program in the case of one and two spatial dimensions.

2.1 The Discrete Variable Approximation in One Dimension

The equation that governs a system with mass m and under the influence of a static potential is the time-independent Schrödinger equation. In one spatial dimension the time-independent Schrödinger equation is written

$$\left(-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}+V(x)\right)\psi_E\left(x\right)=E\psi_E\left(x\right). \tag{2.1}$$

This differential equation is an eigenvalue equation and can be represented more closely to the matrix form of an eigenvalue equation in the following representation:

$$H\left|\psi_E\right\rangle = E\left|\psi_E\right\rangle \tag{2.2}$$

where $\psi_E(x) \equiv \langle x|\psi_E\rangle$, H is the Hamiltonian that represents the operation on $\psi_E(x)$ on the left-hand side of (2.1), and E is the eigenvalue corresponding to the eigenvector. We will also use the compact notation $\langle x|E\rangle$ to denote the eigenstates, $\psi_E(x)$.

As a first approximation, we can discretize the spatial dimension such that we represent the coordinate as $x_i = i\Delta_x + \min(x)$ for $i = 0, 1, 2, ..., N_x - 1$ where $\Delta_x = (\max(x) - \min(x)) / (N_x - 1)$, the potential as $V_i \equiv V(x_i)$, and the wavefunction as $\psi_i \equiv \psi(x_i)$. A half-step central difference approximation for the first derivative yields the following expression:

$$\frac{d\psi}{dx} \simeq \frac{\psi(x + \Delta_x/2) - \psi(x - \Delta_x/2)}{\Delta_x} \tag{2.3}$$

Extending this central difference approximation to the second derivative yields

$$\frac{d^2\psi}{dx^2} \simeq \frac{\psi(x + \Delta_x) - 2\psi(x) + \psi(x - \Delta_x)}{\Delta_x^2}. \tag{2.4}$$

In the discretization described above this expression becomes

$$\frac{d^2\psi}{dx^2} \simeq (1/\Delta_x)^2 \left(\psi_{i+1} - 2\psi_i + \psi_{i-1}\right). \tag{2.5}$$

Now we can rewrite the discretized version of the Schrödinger equation (2.1) as follows

$$-\frac{\hbar^2}{2m}\left(\frac{1}{\Delta_x}\right)^2 \left(\psi_{i+1} - 2\psi_i + \psi_{i-1}\right) + V_i\psi_i = E\psi_i. \tag{2.6}$$

We can make the substitution $\beta = \frac{\hbar^2}{2m\Delta_x^2}$ and (2.6) becomes

$$-\beta\left(\psi_{i+1} - 2\psi_i + \psi_{i-1}\right) + V_i\psi_i = E\psi_i. \tag{2.7}$$

We note here that β has the same dimensions as V and E, that is, energy. For our computations we will choose units of meV. Specifically, we use $\hbar c = 1973 \times 10^3$ meV $\cdot$ Å. We will also specify the mass of the particle governed by the Schrödinger equation in atomic mass units, u. In our expression for β then, $mc^2 = 931.5 \times 10^9$ u meV.

There is one more step necessary in order to cast this equation into the form of a matrix eigenvalue equation (i.e. $H_{i,j}\psi_j = E_j\psi_j$). We must transform this discrete equation such that the wavefunction evaluated at a single point, $\psi_i = \psi(x_i)$, is found in that equation. We can do this via the use of Kronecker delta function, δ_{ij} described in the previous chapter.

Noting that $\psi_i\delta_{i,j} = \psi_j\delta_{i,j}$ we can rewrite (2.7) as the following matrix equation:

$$-\beta\left(\delta_{i,j+1}\psi_i - 2\delta_{i,j}\psi_i + \delta_{i,j-1}\psi_i\right) + V_i\delta_{i,j}\psi_i = E\psi_i\delta_{i,j}. \tag{2.8}$$

Finally we can cast this into its final form of an eigenvalue equation $H_{ij}\psi_j = E_j\psi_j$ as follows:

$$\left[\left(2\beta + V_j\right)\delta_{i,j} - \beta\left(\delta_{i,j+1} + \delta_{i,j-1}\right)\right]\psi_j = E_j\psi_j. \tag{2.9}$$

The final result for the matrix elements of the Hamiltonian is:

$$H_{i,j} = \left(2\beta + V_j\right)\delta_{i,j} - \beta\left(\delta_{i,j+1} + \delta_{i,j-1}\right). \tag{2.10}$$

An alternative way to think about the discrete variable approximation is to consider the discrete points x_j (at which we seek the amplitudes ψ_j) as being an orthonormal basis. In this

perspective, we are working in an N_x-dimensional vector space where the unit vectors, $|x_j\rangle$ are column vectors that can be written as

$$|x_0\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, |x_1\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \cdots, |x_k\rangle = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \tag{2.11}$$

where the 1 appears in the kth position in the $|x_k\rangle$ unit vector. These vectors satisfy the required property of orthonormality, $\langle x_i|x_j\rangle = \delta_{ij}$. An expansion in this basis set of orthonormal unit vectors is an approximation to the wavefunction

$$|\psi\rangle = \sum_{j=0}^{N_x-1} |x_j\rangle\langle x_j|\psi\rangle, \tag{2.12}$$

where $\psi(x) \simeq \psi(x_j) \equiv \langle x_j|\psi\rangle$.

The IDL code that performs the numerical work in the equations above (i.e. constructing the Hamiltonian, equation 2.10, and then performing the diagonalization) is presented below.

```
function dva_solver,x,v,mass = mass
compile_opt idl2,hidden
if n_elements(mass) eq 0 then mass = 1.0
dx = x[1] - x[0] & nx = n_elements(x)
hbarc = 1973d3 & mc2 = mass*931.5d9
b = (hbarc^2)/(2.*mc2*dx^2)
i = rebin(indgen(nx),nx,nx,/sample)
j = transpose(i)
vj = rebin(v,nx,nx,/sample)
h = (2.*b+vj)*ds_delta(i,j)-$
  b*(ds_delta(i,j+1)+ds_delta(i,j-1))
evals = real_part(la_eigenql(h,/double,eigenvectors = evec))
esort = sort(evals) & evals = evals[esort]
psi = evec[*,esort]
str = {psi:psi,evals:evals}
return,str
end
```

This function has two required input parameters. A vector of equally-spaced x-values must be passed into the function via the parameter x. Note that the unit for the spatial coordinates is Å. The potential, defined at those x locations, must be passed into the function via the parameter `v`. Finally the user can specify the mass of the particle with the optional `mass` keyword. As described above, the mass is defined in terms of atomic mass units. If the user does not specify the particle's mass then the default value 1 is used. This function returns a structure variable with two fields: the wavefunction named `psi` and the energy eigenvalues `evals`. The dimension of the `psi` field is $N_x \times N_E$ where N_x is the number of x-values in the spatial discretization and N_E is the number

of energy eigenvalues. Of course the nature of this particular algorithm forces $N_E = N_x$ but we simply point out the previous point so that the user knows which index specifies space and which specifies the eigenstate. The wavefunctions are sorted in order of ascending energy eigenvalue so that, if we invoked our new IDL function `sol=DVA_SOLVER(x,v)`, the ground-state $\langle x|E_0\rangle$ and first excited state $\langle x|E_1\rangle$ wavefunctions are obtained in IDL via `sol.psi[*,0]` and `sol.psi[*,1]`. The corresponding energy eigenvalues are `sol.evals[0]` and `sol.evals[1]` respectively.

Since this is a numerical procedure, the natural question to ask is how good is it as an estimator of the eigenvalues and eigenvectors. Let's consider eigenvalues here for the purpose of illustrating its utility. In short, though, the algorithm works pretty well for the lowest-lying eigenstates but gets poorer for higher energy states. For example, for $V(x) = \frac{1}{2}x^2$, and $-10 \leq x \leq 10$, the errors[1] in the lowest 5 eigenvalues calculated using `DVA_SOLVER` are shown in table 2.1.

Table 2.1: Comparison of eigenvalues for different grid sizes

n	$N_x = 50$ (%)	$N_x = 100$ (%)	$N_x = 200$ (%)
0	0.51	0.13	0.03
1	0.86	0.21	0.05
2	1.3	0.33	0.08
3	1.9	0.45	0.11
4	2.4	0.57	0.14

The take-home message of this demonstration is that the results vary with how fine you discretize the spatial variable. If you are interested in accurate results, you should vary N_x and see how the results change before settling on a value for N_x. This is an effective qualitative algorithm but the user should exercise caution when attempting to obtain accurate numerical values.

The reader may no doubt suspect that there exist more accurate algorithms for computing the stationary states of a one-dimensional system. In fact there are and we will describe one such method that is more accurate but requires an analytical calculation of the matrix elements for the potential. In the *Expansion Method* (EM) in one-dimension we consider a potential that is bounded by an infinite square well on both sides. That is $V(x)$ is finite on $0 < x < L$ but it is infinite outside that region. The position-space eigenstates vanish outside of this region. The basis set which we use to calculate the eigenstates of a particle under the influence of $V(x)$ is that of the infinite square well, given by

$$\langle x|n\rangle = \sqrt{\frac{2}{L}}\sin\left(\frac{n\pi x}{L}\right), \quad n = 1, 2, 3, \cdots$$

and the energy eigenvalues are given by

$$E_n = \frac{n^2\pi^2\hbar^2}{2mL^2}.$$

Using this basis set we evaluate the matrix elements of the Hamiltonian $H_{m,n} = T_{m,n} + V_{m,n}$ where

$$\begin{aligned} H_{m,n} &= \langle m|H|n\rangle \\ &= \langle m|T|n\rangle + \langle m|V|n\rangle, \end{aligned}$$

the kinetic energy operator is given by

$$T = -\frac{\hbar^2}{2m}\frac{d^2}{dx^2},$$

[1]The error is defined as $100 \times \frac{|E_{exact} - E_{calc}|}{E_{exact}}$ where E denotes an eigenvalue.

and $V = V(x)$. Problem-specific knowledge is required to calculate the potential matrix element.

Let's consider a concrete example whose solution we know analytically: the harmonic oscillator $V(x) = k(x - L/2)^2/2$. We must calculate

$$\begin{aligned} V_{m,n} &= \int_0^L dx\ \langle m|x\rangle \frac{1}{2}\left(x - \frac{L}{2}\right)^2 \langle x|n\rangle \\ &= \frac{2}{L}\int_0^L dx\ \frac{1}{2}k\left(x - \frac{L}{2}\right)^2 \sin\left(\frac{n\pi x}{L}\right)\sin\left(\frac{m\pi x}{L}\right) \\ &= \frac{k}{L}\int_0^L dx\ \left(x - \frac{L}{2}\right)^2 \sin\left(\frac{n\pi x}{L}\right)\sin\left(\frac{m\pi x}{L}\right). \end{aligned}$$

This integral can be evaluated analytically and the final result is for $n = m$

$$V_{n,n} = \frac{1}{2}kL^2\left(\frac{1}{12} - \frac{1}{2n^2\pi^2}\right), \tag{2.13}$$

and for $n \neq m$

$$V_{m,n} = \frac{1}{2}\frac{kL^2}{\pi^2}\left[\frac{(-1)^{n-m}+1}{(n-m)^2} - \frac{(-1)^{n+m}+1}{(n+m)^2}\right] \tag{2.14}$$

Now we can calculate the eigenvalues for the lowest eigenstates for $k = 1$, $0 \leq x \leq 20$ and compare to the values calculated using the DVA (for the same size Hamiltonian) and the exact known values.

Table 2.2: Accuracy of eigenvalues for the SHO calculate with the DVA and the EM for a 50×50 Hamiltonian.

n	DVA (%)	EM (%)
0	0.51	4×10^{-4}
1	0.86	1×10^{-3}
2	1.3	1×9^{-6}
3	1.9	3×10^{-4}
4	2.4	2×10^{-5}

The results are shown in table 2.2. It is clear that the EM yields superior results for the same size Hamiltonian. However problem-specific knowledge is required to use the algorithm and it is necessary to calculate the matrix elements, usually by calculating (oftentimes) tedious integrals, a priori. This algorithm has been extended to two dimensions and used to calculate the eigenvalues and eigenfunctions for quantum billiard systems, discussed briefly in the next section.[13] The accuracy is superior to that of the DVA. Nevertheless we the DVA is used for the remainder of this chapter as the method of choice because of its ease-of-use. The fact that it requires that we never calculate any integrals when determining the matrix elements of the Hamiltonian makes it a very flexible algorithm.

** **Ex. 1** — Use the EM to calculate the lowest 5 eigenvalues and eigenfunctions for a linear well $V(x) = V_0|x - \frac{L}{2}|$ where $V_0 = 1$ over the range $0 < x < 20$. You will need to write your own program to perform the diagonalization after calculating an analytic expression for the potential matrix element. Compare your results to the DVA for a 50×50 Hamiltonian. Note: in the DVA this means that $N_x = 50$ and in the EM there should be 50 terms in the expansion.

** 1. Show that the potential matrix elements $V_{m,n}$ are given by

$$V_{n,n} = V_0 \left(\frac{L}{4} - \frac{L}{2n^2\pi^2} \left(1 - (-1)^n\right) \right)$$

for the diagonal elements ($n = m$) and the off-diagonal elements ($n \neq m$) are given by

$$\begin{aligned} V_{m,n} = & \frac{V_0 L}{\pi} \left[\frac{S_{n-m}}{n-m} - \frac{S_{n+m}}{n+m} \right] + \\ & \frac{L}{\pi^2 (n-m)^2} \left[1 + (-1)^{n-m} - 2C_{n-m} - (n-m)\pi S_{n-m} \right] - \\ & \frac{L}{\pi^2 (n+m)^2} \left[1 + (-1)^{n+m} - 2C_{n+m} - (n+m)\pi S_{n+m} \right] \end{aligned}$$

where $S_{n+m} \equiv \sin((n+m)\pi/2)$, $S_{n-m} \equiv \sin((n-m)\pi/2)$, $C_{n+m} \equiv \cos((n+m)\pi/2)$, and $C_{n-m} \equiv \cos((n-m)\pi/2)$.

To illustrate the use of the DVA algorithm in an interesting physical situation, let's use it to calculate the two lowest eigenvalues and eigenstates for a particle of mass 1 u confined in a potential

$$V(x) = \frac{1}{2}x^2 + 8e^{-\frac{1}{2}(x/2)^2}. \tag{2.15}$$

This potential is an harmonic oscillator with a Gaussian barrier separating it into two sides. The amplitude of the Gaussian obviously determines the extent to which the sides are separated.

```
pro dva_solver_ex1
; Configure the colors
device,decomposed = 0 & loadct,0,/silent
; Set up the grid
nx = 300 & xlo = -10.0 & xhi = -xlo
x = linspace(xlo,xhi,nx)
; Define the potential
v = 0.5*x^2+10.*exp(-0.5*(x/2.)^2)
; Solve the Schrodinger equation using the DVA
sol = dva_solver(x,v,mass = 1.0)
; Plot the results
angstrom = '!6!sA!r!u!9 %!6!n'
xtitle = '!6x ( '+angstrom+' !6)'
title = '!3|!6<x!3|!6E!d0!n>!3|!e2!n, !3|!6<x!3|!6E!d1!n>!3|!6!e2!n'
plot,x,v,thick = 4.0,xtitle = xtitle,$
  ytitle = '!6V (meV)',title = title, yrange = [5.0,20.0],$
  /ystyle
ls = [0,2]
for j = 0,1 do begin
  p = abs(sol.psi[*,j])^2
  oplot,x,sol.evals[j]+0.1*p*max(v)/max(p),$
    linestyle = ls[j],thick = 2.0
  plots,!x.crange,[sol.evals[j],sol.evals[j]],$
    linestyle = 2,/data
```

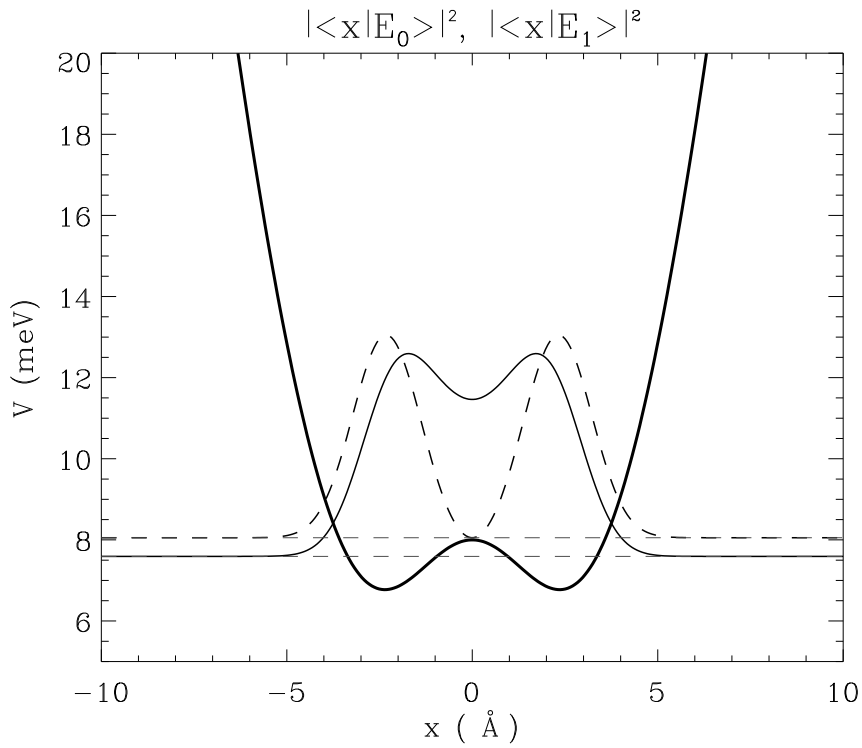


Figure 2.1: Probability densities corresponding to the lowest two eigenstates (solid and dashed curves) for a particle of mass 1 u confined in the potential $V(x) = \frac{1}{2}x^2 + 8e^{-\frac{1}{2}(x/2)^2}$ (thick solid line).

```
endfor
end
```

When we run this procedure, `DVA_SOLVER_EX1`, the probability densities corresponding to the lowest two eigenstates are plotted, as shown in figure 2.1. Note that these densities are offset in the potential well by the corresponding energy eigenvalue. It is worth noting that the magnitude of the probability density in the figure is meaningless compared with the magnitude of the potential. We have simply scaled the probability density so that it appears on the same scale as the potential.

One interesting thing to investigate is the effect of increasing the amplitude of the Gaussian barrier that separates one half of the well from the other. As the barrier height increases between the two sides of the parabolic well, the probability for the particle to be found *under* the barrier decreases and the states approach those of two independent parabolic wells. Another way to look at this is to examine the limit of an infinitely high barrier that separates the two parabolic wells. The ground state of the two independent wells is degenerate. By placing a finite barrier to separate the sides, this degeneracy is removed and the degenerate ground state is split into two states. These states' eigenvalues are close but not overlapping. As the barrier decreases in height, the eigenvalues for the two lowest states get further apart. Hence the difference between figures 2.1 and 2.2.

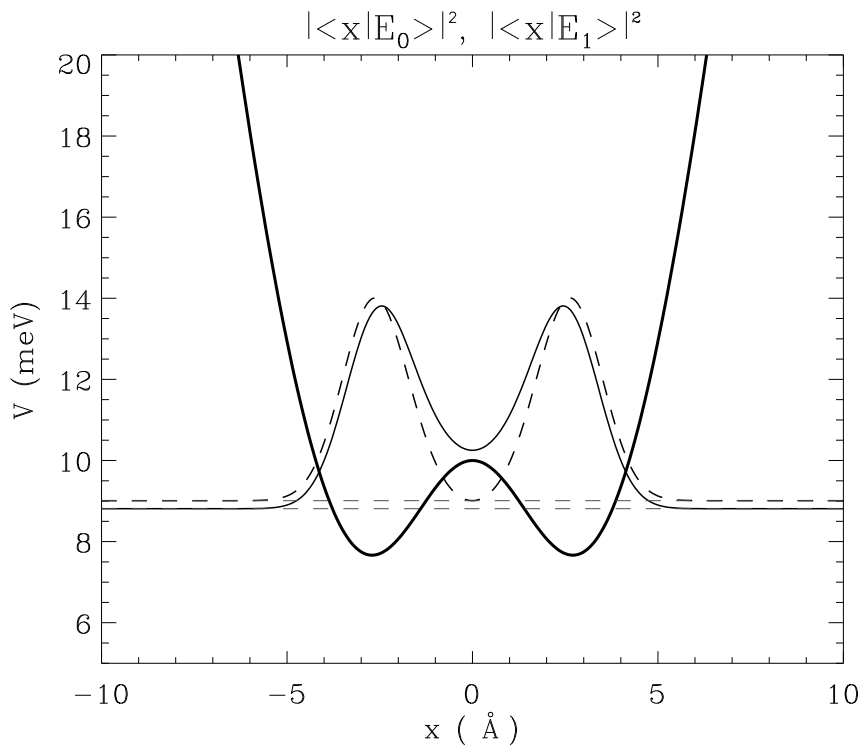


Figure 2.2: Probability densities corresponding to the lowest two eigenstates for a particle of mass 1 u confined in the potential $V(x) = \frac{1}{2}x^2 + 10e^{-\frac{1}{2}(x/2)^2}$. Note the differences with figure 2.1, specifically that the eigenvalues are closer together.

If we were to prepare the particle initially as a superposition of the two lowest energy eigenstates, then we would be able to observe the phenomenon of quantum tunneling. That is, the particle can be found with probability 1 in the right hand side of the potential well initially. But as time goes on, the wavefunction evolves and the probability density will slosh from one side of the well to the other, thus tunneling through the barrier.

We can visualize this phenomenon of quantum tunneling now that we know the eigenfunctions, $\langle x|E\rangle$, of the Hamiltonian. Physically we can imagine that two eigenstates are coupled through some interaction with radiation perhaps and that we have prepared the initial state as a superposition of the two lowest eigenstates.

$$\langle x|\psi\rangle(0) = \frac{1}{\sqrt{2}}\left(\langle x|E_0\rangle + \langle x|E_1\rangle\right), \tag{2.16}$$

where $\langle x|E_0\rangle$ and $\langle x|E_1\rangle$ are different eigenstates of the Hamiltonian. Then the wavefunction at any later time, t, is given by

$$\langle x|\psi\rangle(t) = \frac{1}{\sqrt{2}}\left(\langle x|E_0\rangle e^{-iE_0t/\hbar} + \langle x|E_1\rangle e^{-iE_1t/\hbar}\right), \tag{2.17}$$

where $i = \sqrt{-1}$. The time evolution of the probability density is given by $P(x,t) = |\langle x|\psi\rangle(t)|^2$ which, upon explicit calculation, yields

$$P(x,t) = \frac{1}{2}[|\langle x|E_0\rangle|^2 + |\langle x|E_1\rangle|^2 + \langle x|E_1\rangle^*\langle x|E_0\rangle e^{-i\omega_{01}t} + \langle x|E_0\rangle^*\langle x|E_1\rangle e^{i\omega_{01}t}] \tag{2.18}$$

where $\omega_{01} = \frac{E_0 - E_1}{\hbar}$. This time-dependent probability density, an observable quantity, is real.

The code that shows this time-evolution graphically, an animation, is shown below. Note that we don't explicitly use the expanded expression 2.18. Rather we calculate the wavefunction $\langle x|\psi\rangle(t)$ and then just use the definition $P(x,t) = |\langle x|\psi\rangle(t)|^2$ using the IDL code `prob=abs(psi)^2`. The explicit code that shows how the animation is performed is in the function `ANIMATE_TRANSITION_PROB` which is presented in chapter 1.

```
pro dva_animate_ex1
device,decomposed = 0
; Set up the spatial grid
nx = 300 & xlo = -10.0 & xhi = -xlo
x = linspace(xlo,xhi,nx)
; Define the potential
v = 0.5*x^2+20.*exp(-0.5*(x/2.)^2)
; Solve the Schrodinger equation using the DVA
sol = dva_solver(x,v,mass = 1.0)
wf = sol.psi & eigvals = sol.evals
n = 0 & m = 1
ret = animate_transition_prob(x,v,wf,eigvals,n,m)
end
```

Though it is not quite as satisfying as viewing a smooth animation, a sequence of six frames from this animation is shown in figure 2.3. This shows the time-evolution over one-half of the

period of the motion. This animation illustrates the phenomenon of quantum tunneling. Initially the particle is localized in the well on the right hand side. Classically it would remain there forever. However quantum-mechanically it can tunnel from one side to another. It does this periodically and the frequency with which it performs this motion is known as the tunneling frequency.

Ex. 2 — Compute the eigenvalues of the simple harmonic oscillator potential $V(x) = \frac{1}{2}x^2$ where the potential is defined over the range $-10 \leq x \leq 10$ and the mass of the particle is one atomic mass unit (i.e. $m = 1$ u).

1. Plot the lowest five eigenvalues of the harmonic oscillator as a function of N_x where N_x varies between 10 and 200.
2. Based on these results, what values for N_x are *good enough*?

Ex. 3 — Calculate and display the probability densities corresponding to the lowest 5 eigenstates of a particle of mass 1 u confined in the linear potential well $V(x) = |x|$.

* **Ex. 4** — For the linear well of the previous exercise, plot the eigenvalue, E_n, as a function of eigenvalue number, n, for the lowest 20 eigenvalues. Note that the accuracy of your result will depend on the value you choose for N_x as well as the bounds for x that you choose.

* **Ex. 5** — Calculate the time evolution of the wavefunction for a particle of mass 1 u initially in the state $\langle x|\psi\rangle(0) = \frac{1}{\sqrt{2}}(\langle x|E_0\rangle + \langle x|E_1\rangle)$ under the influence of the potential given by $V(x) = \frac{1}{2}x^2 + V_o e^{-\frac{1}{2}(x/2)^2}$. Note that $\langle x|E_0\rangle$ and $\langle x|E_1\rangle$ are eigenstates of this potential.

1. Specifically, for values of V_0 between 5 and 20, plot the transition energy $\Delta E = E_1 - E_0$ as a function of V_0.
2. Create animations of the real and imaginary parts of the wavefunction for a half period of motion for each of these values of V_0.

** **Ex. 6** — From your results from the previous exercise, describe the qualitative difference you observe in the time-development of the wavefunction components as the barrier height V_0 increases compared to the motion of the probability density (e.g. do they evolve faster or slower as compared to the probability density $P(x,t)$?). Explain this trend.

* **Ex. 7** — Which two eigenstates of the potential $V(x) = \frac{1}{2}x^2 + 20e^{-\frac{1}{2}(x/2)^2}$, when superposed as an initial wavefunction, display a probability density that sloshes from side to side **within** each of the two wells? Hint: you are looking for n and m in the following wavefunction $\langle x|\psi\rangle(0) = \frac{1}{\sqrt{(2)}}(\langle x|E_n\rangle + \langle x|E_m\rangle)$. Verify your response through construction and observation of the animated sequence.

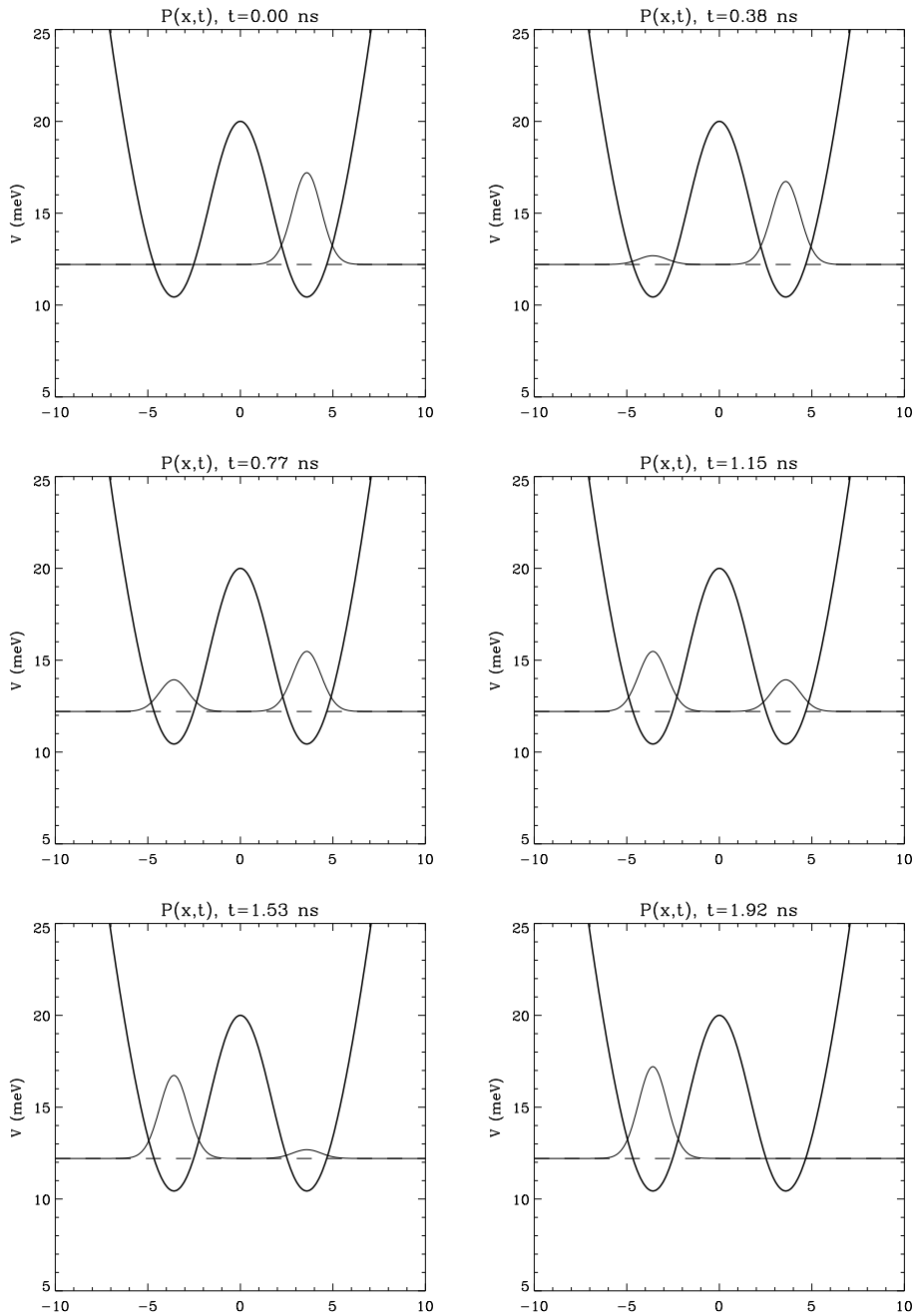


Figure 2.3: Time evolution of the probability density $P(x,t)$ through one half of a cycle. The initial state is given by $\langle x|\psi\rangle(0) = \frac{1}{\sqrt{2}}\left(\langle x|E_0\rangle + \langle x|E_1\rangle\right)$ evolving under the influence of the potential given by $V(x) = \frac{1}{2}x^2 + 20e^{-\frac{1}{2}(x/2)^2}$.

2.2 The Discrete Variable Approximation in Two Dimensions

The two-dimensional Schrödinger equation is written

$$\left(-\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)+V(x,y)\right)\psi(x,y)=E\psi(x,y)\,. \tag{2.19}$$

The extension of the discretization method that was presented in the previous section to two dimensions is straightforward. The discretization of x is identical and the discretization of y is $y_n = n\Delta_y + \min(y)$ for $n = 0, 1, 2, ..., N_y - 1$ where $\Delta_y = (\max(y) - \min(y)) / (N_y - 1)$. With this mesh in x and y we can write down the approximations to the second derivatives

$$\frac{d^2\psi}{dx^2} \simeq (1/\Delta_x)^2 (\psi_{i+1,n} + \psi_{i-1,n} - 2\psi_{i,n}) \tag{2.20}$$

and

$$\frac{d^2\psi}{dy^2} \simeq (1/\Delta_y)^2 (\psi_{i,n+1} + \psi_{i,n-1} - 2\psi_{i,n})\,. \tag{2.21}$$

Next we define the two parameters

$$\beta_x = \frac{\hbar^2}{2m}(1/\Delta_x)^2 \tag{2.22}$$

and

$$\beta_y = \frac{\hbar^2}{2m}(1/\Delta_y)^2\,. \tag{2.23}$$

The remaining steps to obtain the Hamiltonian matrix are similar to those of the one-dimensional system with the final result being

$$\begin{aligned} H_{i,j,n,m} \quad = \quad & (2(\beta_x+\beta_y)+V_{i,n})\,\delta_{i,j}\delta_{n,m} \\ & -\beta_x(\delta_{i,j+1}+\delta_{i,j-1})\,\delta_{n,m} \\ & -\beta_y(\delta_{n,m+1}+\delta_{n,m-1})\,\delta_{i,j}. \end{aligned} \tag{2.24}$$

The implementation of this algorithm in IDL is straightforward though the Hamiltonian matrix will be quite large. We expect that the evaluation of the matrix will be time-consuming and the diagonalization of the resulting Hamiltonian will be especially so. The four-dimensional Hamiltonian $H_{i,j,n,m}$ must be converted into a two-dimensional matrix in order to diagonalize it (IDL's eigenvalue/eigenvector function `LA_EIGENQL` requires a two-dimensional array). There are a couple of ways that we can approach this. The first is to group the indices i and n into one dimension and j and m into the other and compute the resulting two-dimensional Hamiltonian. If the four-dimensional Hamiltonian was N_x by N_y by N_x by N_y then the two-dimensional Hamiltonian will be $N_x \times N_y$ by $N_x \times N_y$. The other way to do this is to compute the four-dimensional Hamiltonian and then use the IDL function `REFORM` to reduce it to two-dimensions (but keeping the same number of matrix elements). This latter approach is the one that we adopt because, as you will see, the code for evaluating the matrix elements looks very similar to equation 2.24. The way that we've implemented results in fast evaluation of the matrix elements too.

The function, `DVA_SOLVER_2D`, calculates the eigenvalues and eigenfunctions for a two-dimensional potential. Even though we've reduced the dimensionality of the Hamiltonian we still must calculate just as many matrix elements. A brute-force approach involves looping over all four dimensions. When programming in IDL, looping over indices is oftentimes a losing proposition

(i.e. *slow*). But we can achieve some economy by looping over only two of the dimensions and invoking our vector Kronecker δ-function, `DS_DELTA`, in the evaluation. This works well because it takes advantage of the vector nature of the `EQ` operator. The way that we've written `DS_DELTA`, we can use as the first argument an array and a scalar as the second argument. The result will be an array with the same dimensions as the first argument. As equation 2.24 suggests, the hamiltonian is four-dimensional: $N_x \times N_y \times N_x \times N_y$. IDL's eigenvalue/eigenvector function `LA_EIGENQL` will only work on a two-dimensional matrix so after calculating our four-dimensional array, the IDL function `REFORM` is used to convert it into two dimensions. The remainder of the algorithm is simply extracting out the solution, the eigenvalues and eigenvectors, from the result of `LA_EIGENQL`.

```
function dva_solver_2d,x,y,v,mass = mass,range = range
compile_opt idl2,hidden
if n_elements(mass) eq 0 then mass = 1.0
if n_elements(range) eq 0 then range = [0,n_elements(x)-1]
dx = x[1] - x[0] & dy = y[1] - y[0]
nx = n_elements(x) & ny = n_elements(y)
hbarc = 1973d3 & mc2 = mass*931.5d9
bx = (hbarc^2)/(2.*mc2*dx^2) & by = (hbarc^2)/(2.*mc2*dy^2)
h = fltarr(nx,ny,nx,ny)
ivec = indgen(nx) & nvec = indgen(ny)
i = rebin(ivec,nx,ny,/sample)
n = rebin(transpose(nvec),nx,ny,/sample)
for m = 0,ny-1 do begin
  for j = 0,nx-1 do begin
    h[*,*,j,m] = (2.0*(bx+by)+v[i,n])*ds_delta(i,j)*ds_delta(n,m)-$
                 bx*(ds_delta(i,j+1)+ds_delta(i,j-1))*ds_delta(n,m)-$
                 by*(ds_delta(n,m+1)+ds_delta(n,m-1))*ds_delta(i,j)
  endfor
endfor
h = reform(h,nx*ny,nx*ny)
evals = la_eigenql(h,eigenvectors = evec,range = range)
esort = sort(evals)
evals = evals[esort]
evec = evec[*,esort]
; Now disentangle the eigenvectors into the more
; intuitive format [nx,ny,n_elements(evals)]
num_evals = n_elements(evals)
psi = complexarr(nx,ny,num_evals)
i = indgen(nx)
for k = 0,num_evals-1 do $
  for n = 0,ny-1 do psi[i,n,k] = evec[n*nx:(n+1)*nx-1,k]
str = {psi:psi,evals:evals}
return,str
end
```

The simple interface to the function makes it easy to use. Required input parameters include a vector of x-values (assumed to be equally spaced), a vector of y-values (also assumed to be equally

spaced), and a two-dimensional array V defining the potential over the range of x- and y-values. Optional input keywords include `mass`, the particle's mass in atomic mass units and `RANGE` which specifies the range of eigenvalues/eigenvectors to return in the diagonalization. This turns out to be a nice "knob to turn" in the numerical evaluation because it takes less time to calculate a few eigenvalues than it does for more eigenvalues, especially for large Hamiltonian matrices.

As a "warm-up" exercise with this function we can compute the lowest six eigenvalues for a two-dimensional isotropic harmonic oscillator. For a particle of mass 1 u the ground state energy $\hbar\omega_0$ is given by

$$\hbar\omega_0 = \hbar c\sqrt{\frac{k}{mc^2}}. \tag{2.25}$$

Assuming $k = 1$ (or equivalently $V(x, y) = \frac{1}{2}\left(x^2 + y^2\right)$), we have $\hbar\omega_0 = 2.044$ meV. The energy eigenvalues for the oscillator are given by $E_{n,m} = (n + m + 1)\,\hbar\omega_0$ for $n, m = 0, \pm1, \pm2, \cdots$. There is obvious degeneracy among the eigenvalues. For instance $E_{0,1} = E_{1,0} = 2\hbar\omega_0$, $E_{0,2} = E_{2,0} = E_{1,1} = 3\hbar\omega_0$, etc. For $-10 \le x \le 10$, $-10 \le y \le 10$, $N_x = N_y = 25$ the lowest six eigenvalues are shown in table 2.3.

Table 2.3: Comparison of the accuracy in the lowest six eigenvalues for the 2-D isotropic simple harmonic oscillator for $N_x = N_y = 25$

n + m	$E_{n,m}$ (DVA)	$E_{n,m}$ (exact)	% error
0	2.000	2.044	2.2
1	3.953	4.089	3.4
1	3.953	4.089	3.4
2	5.809	6.133	5.6
2	5.809	6.133	5.6
2	5.906	6.133	3.8

It is clear from the table that the function returns eigenvalues that have discrepancies of a few percent, and the difference increases as the eigenvalue index increases. Furthermore the degeneracy is not even maintained for higher eigenvalues.[2]

As in the one-dimensional DVA algorithm, we must calculate the accuracy of this algorithm. In order to do so we used the function `DVA_2D_SOLVER` to calculate the lowest three eigenvalues for the two-dimensional harmonic oscillator for a particle of mass 1 u. The potential is defined as $\frac{1}{2}\left(x^2 + y^2\right)$ over the range $-10 \le x \le 10$, $-10 \le y \le 10$. The results of the calculation, the percent error, are shown in table 2.4. The eigenvalues for this potential are known exactly so the comparisons are straightforward. Performance is certainly an issue when diagonalizing a matrix of this size. On my laptop computer it took 0.6, 19, and 192 seconds to solve the Schrödinger equation for $N_x = N_y = 25$, 50, and 75 respectively. Note that there are over 390K, 6.25M, and 31.6M matrix elements for the $N_x = N_y = 25$, 50, and 75 cases, respectively.

Table 2.4: Comparison of the accuracy in the lowest three eigenvalues for different grid sizes

n+m+1	$N_x = N_y = 25$ (%)	$N_x = N_y = 50$ (%)	$N_x = N_y = 75$ (%)
1	2.2	0.51	0.22
2	3.4	0.77	0.34
3	5.6	1.22	0.52

[2]This point is masked by the number of digits shown in the calculated eigenvalues in the second column.

Though the algorithm is clearly the most accurate for the smaller eigenvalues, we can use it for effective visualizations and with the same qualifying remark as we made for the function in one dimension, `DVA_SOLVER`. Accuracy in calculation of the larger eigenvalues requires diagonalization of large matrices. Given the inaccuracy of this algorithm we should briefly discuss why we chose to describe the implementation of the discrete variable approximation rather than one of the more accurate methods for computing the eigenvalues and eigenfunctions for a potential.

In short, the DVA is a very general technique that does not require more information about the physical system than the potential. The user simply supplies a vector or array of values for the potential defined over a spatial grid. It is worth mentioning one other algorithm that has been used in calculating the eigenstates for two-dimensional potentials with closed boundaries. An algorithm that is particularly accurate, especially for quantum billiards problems, is the *Expansion Method* (EM) as described in the paper by Kaufman et. al.[13] and applied to a one-dimensional system in the previous section. In this method, for computational purposes, the potential is inscribed in a rectangular region of space and the eigenstates for the potential are expressed as an expansion in the known eigenstates of the rectangular region. Determining the matrix elements of the Hamiltonian requires computing the overlap integral

$$V_{i,j,n,m} = \langle j,m|V|i,n\rangle \tag{2.26}$$

$$= \int\int dx\ dy\ \langle j,m|x,y\rangle\langle x,y|i,n\rangle V\left(x,y\right) \tag{2.27}$$

where the $|i,n\rangle$ are the eigenstates for the rectangular region in which the potential $V\left(x,y\right)$ is inscribed. Note that i is *not* used here as the imaginary unit but rather an index specifying the particular eigenvalue. The authors in [13] point out that the most benefit in terms of computational speed is obtained if the $V_{i,j,n,m}$ can be calculated analytically a priori. For a certain example given in [13] a 400×400 matrix was diagonalized resulting in the first 50 eigenvalues being within 0.13% of the exact values. Note that the DVA for the isotropic harmonic oscillator exhibited less accuracy for a 625×625 matrix ($N_x = 25,\ N_y = 25$). For accuracy, the interested reader is encouraged to consult the EM reference. For the remainder of this chapter we will perform calculations using the DVA for $N_x = N_y = 50$ with an emphasis on visualization rather than accuracy of tenths of a percent or less. Our interest here is in using a simple algorithm to get the correct qualitative behavior.

2.2.1 The Stadium Potential

In two dimensions we can consider many different types of potentials that have a closed boundary. Textbooks on quantum mechanics often treat the two dimensional infinite square well for instance. And we have already looked at the two dimensional isotropic harmonic oscillator. In the last twenty years considerable interest has focused on the so-called quantum billiards. Because the motion of a classical particle in a "stadium" potential yields chaotic motion, researchers investigated the quantum analog of the stadium potential well, hoping to gain insight into quantum chaos. The stadium potential has a boundary defined by the following:

$$\left(x+\frac{L}{2}\right)^2 + y^2 = R^2 \quad for \quad x \le -\frac{L}{2}$$

$$y = \pm R \quad for \quad -\frac{L}{2} \le x \le \frac{L}{2}$$

$$\left(x-\frac{L}{2}\right)^2 + y^2 = R^2 \quad for \quad x \ge \frac{L}{2}.$$

For our first visualization we calculate the lowest six eigenvalues and eigenfunctions for $-V_0 = -50$ meV, $L = 2$ Å, and $R = 9$ Å. The corresponding probability densities are shown in figure 2.4. The eigenvalues for these lowest states are closely spaced: -49.88, -49.70, -49.67, -49.44, -49.44, and -49.33 meV. These are not degenerate to the precision calculated (masked by rounding the numbers). The program that performs the calculation of the eigenvalues and eigenstates for the stadium potential is `DVA_SOLVER_2D_STADIUM`, shown below. In this function, the input parameters are N_x, N_y, and N_{eig}. N_x and N_y are the sizes of each of the two dimensions and N_{eig} is the eigenstate number to display.

```
pro dva_solver_2d_stadium, nx = nx, ny = ny,neig = neig
device,decomposed = 0
if n_elements(neig) eq 0 then neig = 0
if n_elements(range) eq 0 then range = [0,10]
loadct,1,/silent
if n_elements(nx) eq 0 then nx = 25
if n_elements(ny) eq 0 then ny = 25
x = linspace(-10.0,10.0,nx)
y = linspace(-10.0,10.0,ny)
xm = rebin(x,nx,ny,/sample)
ym = rebin(transpose(y),nx,ny,/sample)
v = 0.0*xm & vo = -50.0
l = 2.0 & r = 9.0
; Region I
xc1 = -0.5*l & yc1 = 0.0
r1_2 = (xm-xc1)^2+(ym-yc1)^2
condI = (r1_2 le r^2)
; Region III
xc3 = 0.5*l & yc3 = 0.0
r3_2 = (xm-xc3)^2+(ym-yc3)^2
condIII = (r3_2 le r^2)
; Region II
condII = (xm ge xc1) and (xm le xc3) and (ym ge -r) and (ym le r)
cond = (condI or condIII or condII)
indices = where(cond,count)
ind = array_indices(xm, indices)
if count gt 0 then v[ind[0,*],ind[1,*]] = vo
sol = dva_solver_2d(x,y,v,range = range)
tfinish = systime(/seconds)
prob = abs(sol.psi)^2
window,0 & plotimage,bytscl(prob[*,*,neig])
contour,v,xm,ym,/noerase,xrange = [min(x),max(x)],$
  ystyle = 5,xstyle = 5,yrange = [min(y),max(y)]
print,sol.evals[neig]
end
```

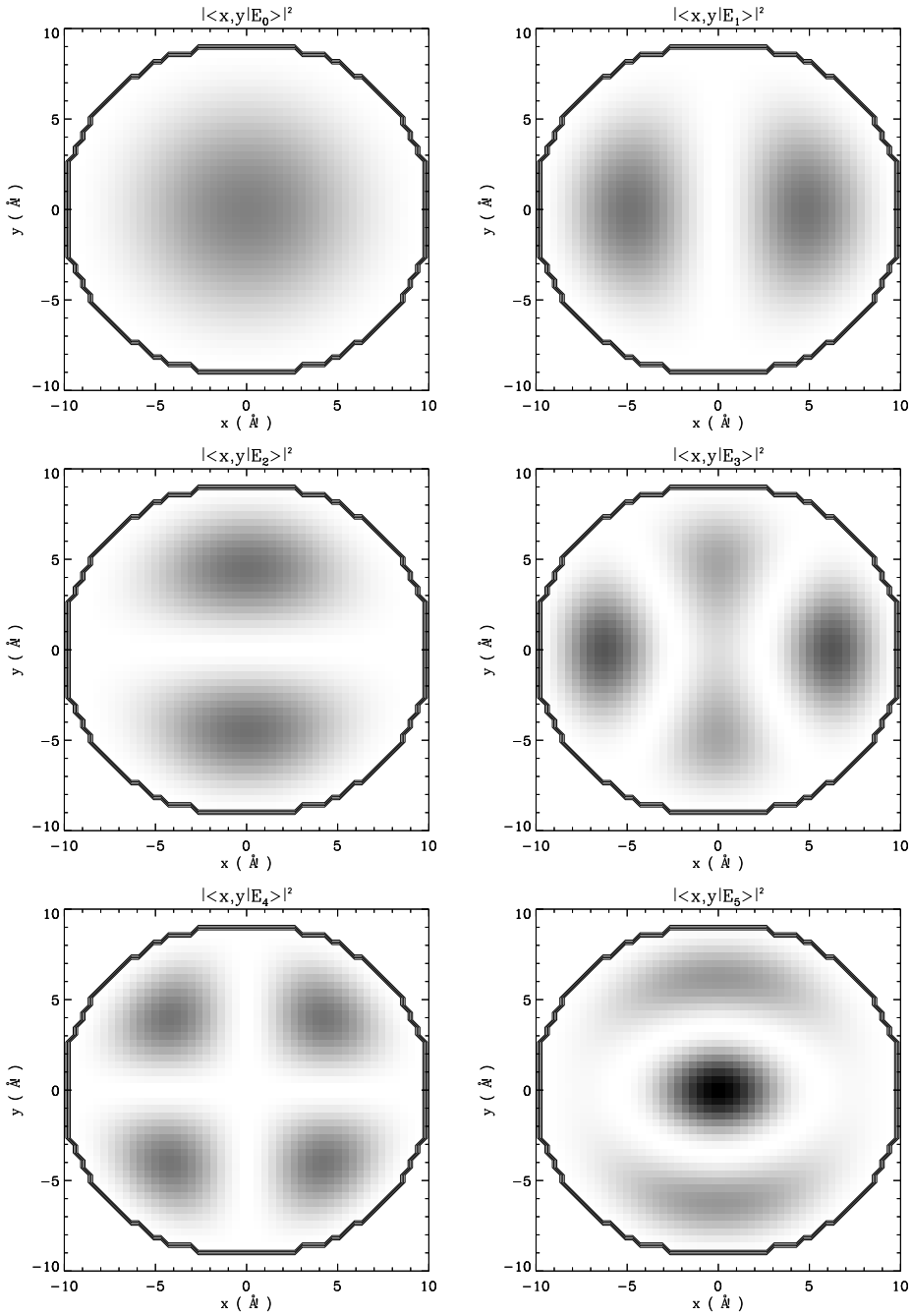


Figure 2.4: Probability densities corresponding to the lowest six eigenstates of the stadium potential. The calculation was done for a stadium depth of -50 meV and $N_x = N_y = 50$.

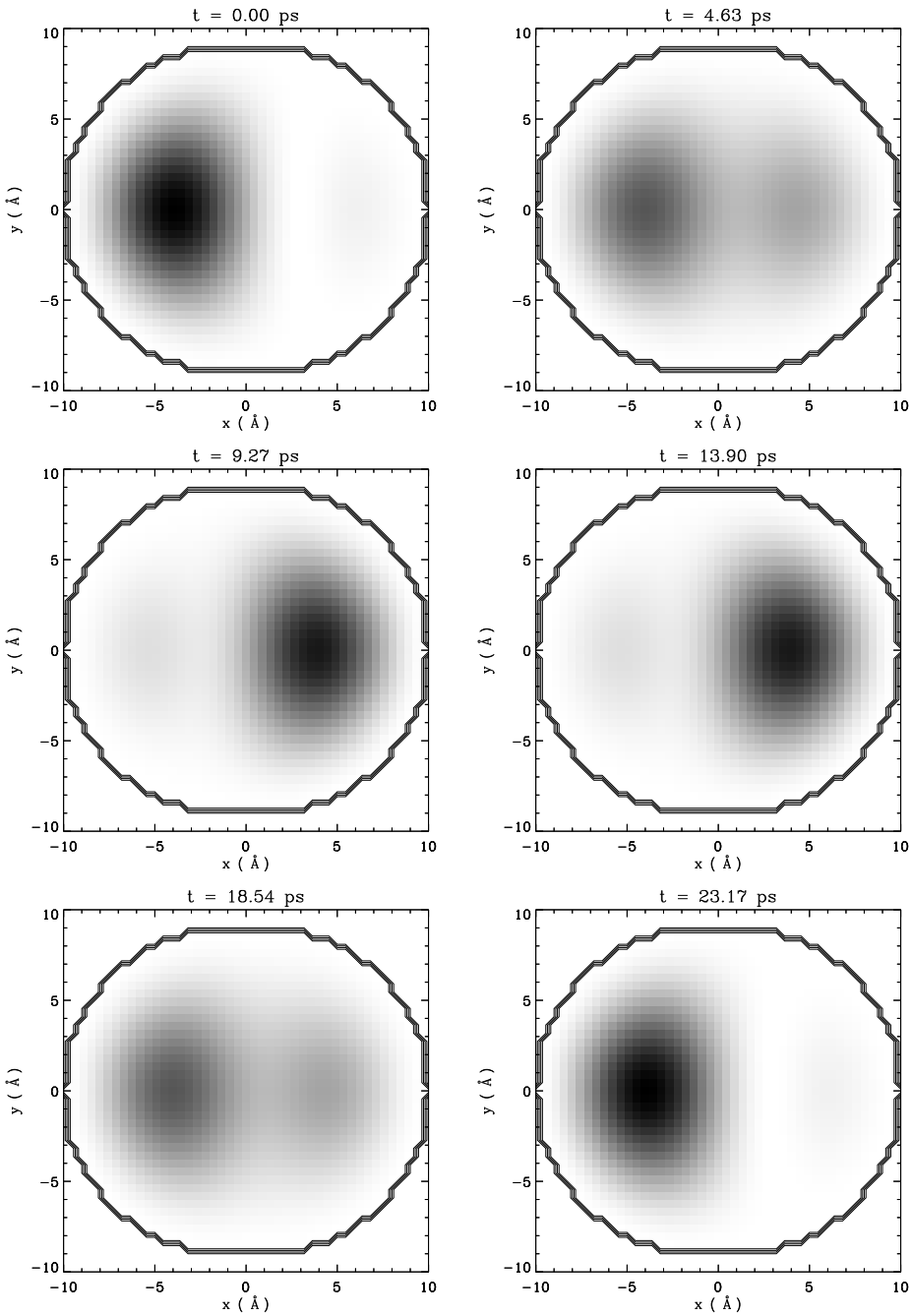


Figure 2.5: Time evolution of the probability density $P(x,t)$ for a state composed of the lowest two eigenstates for a stadium billiard potential. The initial state is given by $\langle x|\psi\rangle(0) = (\langle x|E_0\rangle + \langle x|E_1\rangle)/\sqrt{2}$. The calculation was done for a stadium depth of -50 meV and $N_x = N_y = 45$.

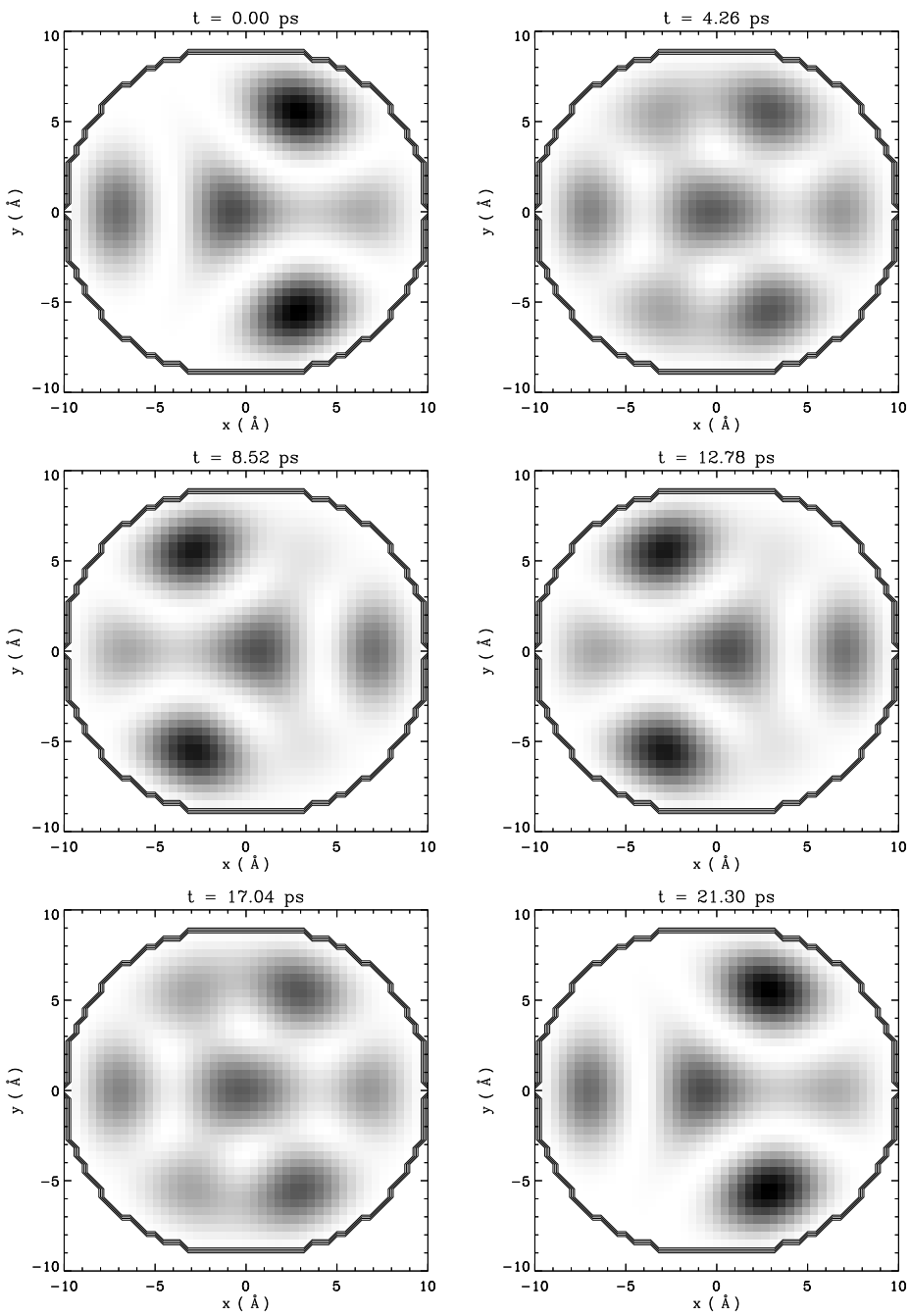


Figure 2.6: Time evolution of the probability density $P(x,t)$ for the initial state $\langle x|\psi\rangle(0) = (\langle x|E_5\rangle + \langle x|E_6\rangle)/\sqrt{2}$. The calculation was done for a stadium depth of -50 meV and $N_x = N_y = 45$.

2.2.2 The Cardioid Potential

Another family of boundaries that possess interesting eigenvalue properties are polar figures. One such boundary is the cardioid. This polar curve can be easily parameterized using the complex number $z = x + iy$ where $i = \sqrt{-1}$. The cardioid is defined by

$$z = pe^{it} + \frac{p}{2}e^{2it} + x_c + iy_c \tag{2.28}$$

where $t \in \{0, 2\pi\}$, the center is located at (x_c, y_c), and p is a scale factor for the shape. This yields

$$x(t) = \frac{p}{2}(2\cos t + \cos 2t) + x_c \tag{2.29}$$

and

$$y(t) = \frac{p}{2}(2\sin t + \sin 2t) + y_c. \tag{2.30}$$

The potential is defined as $-V_0$ within the boundary and zero outside. The probability densities corresponding to the lowest six eigenstates for $-V_0 = -50$ meV and $p = 7$ Åare shown in figure 2.7. The eigenvalues for these lowest states are closely spaced: -49.84, -49.63, -49.57, -49.31, -49.31, and -49.14 meV. These are not degenerate to the precision calculated (masked by rounding the numbers). The program that displays the probability density corresponding to a user-specified eigenstate for the cardioid potential is `DVA_SOLVER_2D_CARDIOID`. The input keywords are identical to those already discussed for `DVA_SOLVER_2D_STADIUM` so they will not be repeated.

```
pro dva_solver_2d_cardioid, nx = nx, ny = ny,neig = neig
device,decomposed = 0
if n_elements(neig) eq 0 then neig = 0
if n_elements(range) eq 0 then range = [0,10]
loadct,1,/silent
if n_elements(nx) eq 0 then nx = 25
if n_elements(ny) eq 0 then ny = 25
x = linspace(-10.0,10.0,nx)
y = linspace(-10.0,10.0,ny)
xm = rebin(x,nx,ny,/sample)
ym = rebin(transpose(y),nx,ny,/sample)
v = 0.0*xm
vo = -50.0
n = 100 & dt = 2.*!pi/(n-1.0) & t = dt*findgen(n)
i = complex(0.0,1.0)
p = 7.0
z = p*exp(i*t)+0.5*p*exp(2.*i*t)
xtrace = float(z)-2.0
ytrace = imaginary(z)
for k = 0,nx-1 do begin
  for j = 0,ny-1 do begin
    if inside(xm[k,j],ym[k,j],xtrace,ytrace) then v[k,j]=vo
  endfor
endfor
sol = dva_solver_2d(x,y,v,range = range)
prob = abs(sol.psi)^2
```

```
window,0 & plotimage,bytscl(prob[*,*,neig])
contour,v,xm,ym,/noerase,xrange = [min(x),max(x)],$
  ystyle = 5,xstyle = 5,yrange = [min(y),max(y)]
print,sol.evals[neig]
end
```

* **Ex. 8** — Compute the eigenfunctions and eigenvalues for a 150 meV deep well whose boundary is defined by an equilateral triangle with the vertices (-10,-10),(10,-10) and $(0,-10+\sqrt{300})$ (0 meV outside of the boundary).

1. Verify that there are no imaginary components to the wavefunction for the lowest six eigenstates.
2. Plot the lowest six eigenstates.
3. Does the symmetry of the well suggest any possible degeneracies and is this borne out in your visualizations?

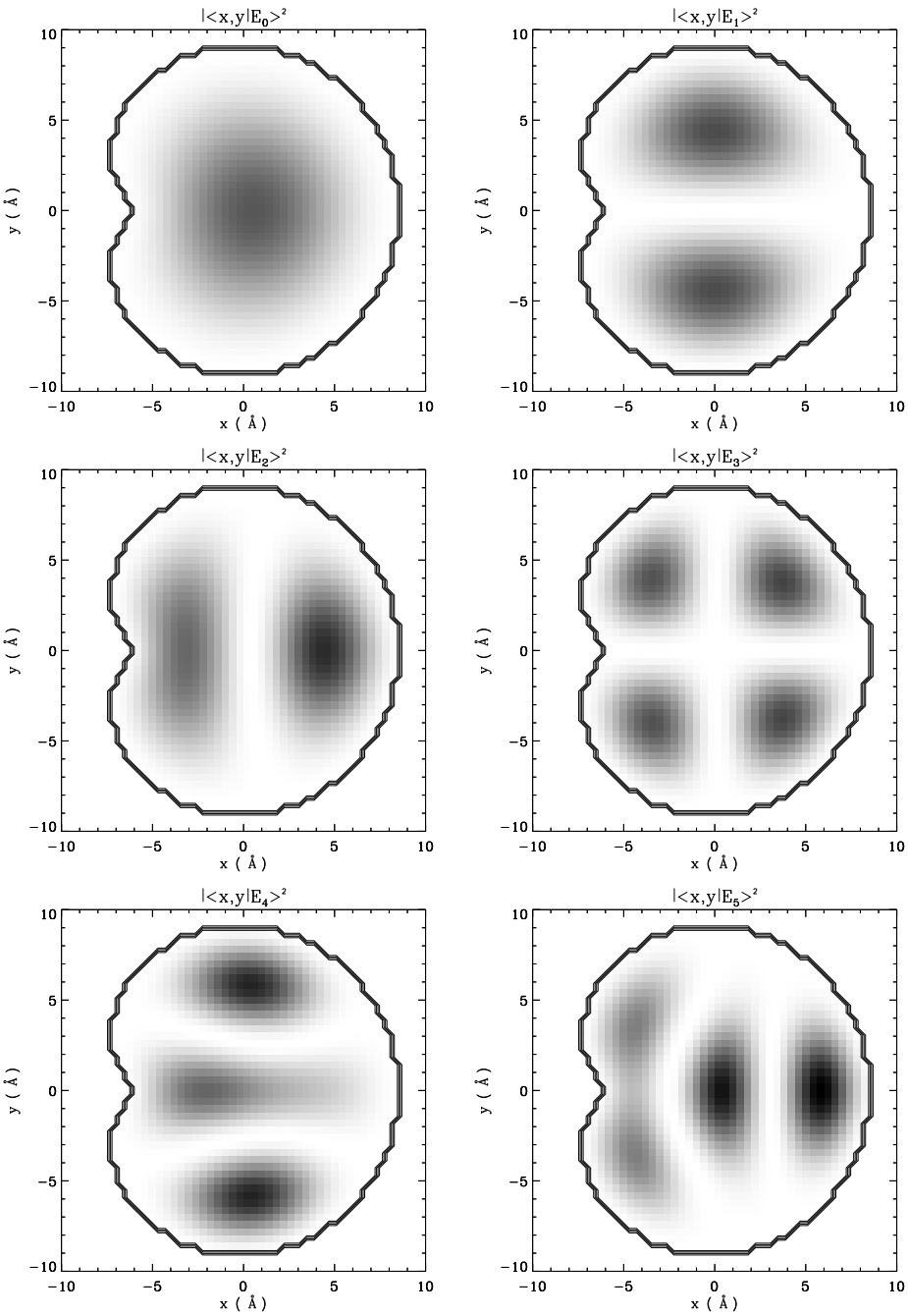


Figure 2.7: Probability densities corresponding to the lowest six eigenstates of the cardioid potential, equation 2.28, with $-V_0 = -50$ meV and $p = 7$ Å.

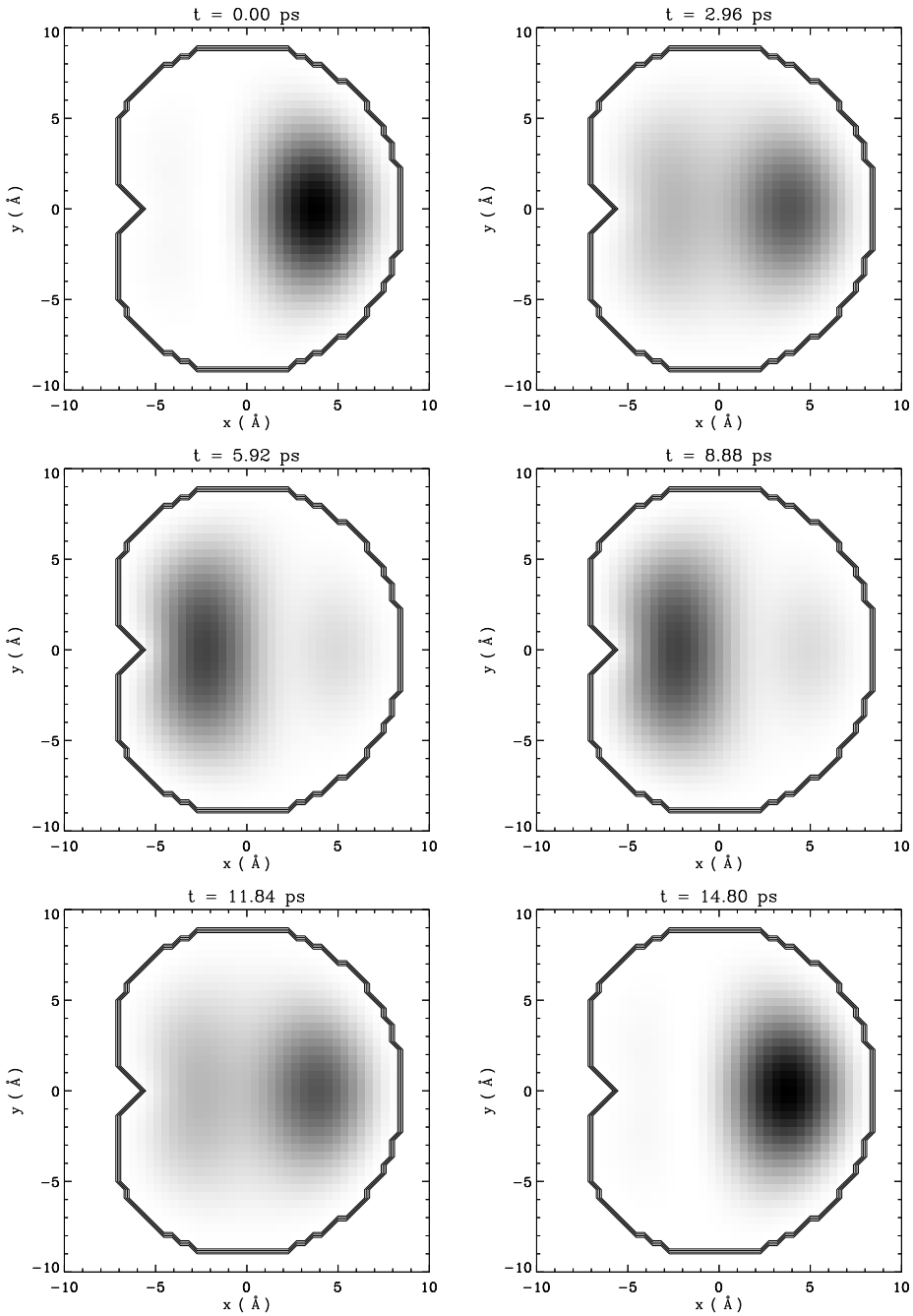


Figure 2.8: Time evolution of the probability density $P(x,t)$ for the initial state $\langle x|\psi\rangle(0) = (\langle x|E_0\rangle + \langle x|E_2\rangle)/\sqrt{2}$ in a potential well whose boundary is defined by a cardioid. The calculation was done for a stadium depth of -50 meV, $p = 7$ and $N_x = N_y = 50$.

Chapter 3

Quantum Rotations

The examples covered in the previous chapter assume that the wavefunction is constrained to a box. When the potential is periodic then this assumption no longer holds and we must consider a wavefunction constructed from the harmonics with the same periodicity as the potential. In this chapter we consider a few examples of quantum systems in which the rotational symmetry of the potential plays a role in the resulting eigenvalues and eigenvectors.

3.1 Hindered Methyl Rotations

Many chemical compounds contain molecular sub-units, common among these sub-units are the methyl groups. Composed of a single carbon atom and three hydrogen atoms, the motion of the methyl group can often be characterized as a *single particle*. That is, a single potential determines the overall motion of the CH_3 unit. In many cases the situation is simplified by the fact that the three-fold symmetry of the molecule itself causes the intermolecular potential to be three-fold symmetric. Thus the potential can be written as a sum of $3m$-fold symmetric terms, usually expressed as an expansion of the potential:

$$V(\theta) = \sum_{m=0}^{\infty} v_{3m} \cos(3m\theta + \alpha_m), \tag{3.1}$$

where θ is the coordinate that specifies the orientation of the three-fold rotor, and v_{3m} and α_m are the magnitude and phase, respectively, of the Fourier expansion of the potential. Often the lowest two terms dominate the expansion and the potential can be expressed as

$$V(\theta) = \frac{V_3}{2}\left(1 - \cos 3\theta\right) \tag{3.2}$$

where V_3 is the barrier height. Therefore the full Hamiltonian can be written

$$H = -\frac{\hbar^2}{2I}\frac{d^2}{d\theta^2} + \frac{V_3}{2}\left(1 - \cos 3\theta\right) \tag{3.3}$$

where I is the moment of inertia of the molecular sub-unit.

* **Ex. 9** — Derive an approximate expression for the lowest eigenvalue in the limit of a very high barrier. Hint: Consider a small angle expansion of the Schrödinger equation for a methyl rotor about $\theta = 0$ to second order in θ and compare with the Schrödinger equation for a simple harmonic oscillator.

3.2 Determining the Hamiltonian Matrix

In order to determine the Hamiltonian matrix, we will use a variant of the DVA. But rather than determining the amplitudes of the wavefunction at a specified set of coordinates, we will determine the amplitudes of a set of known eigenfunctions. We need to evaluate the matrix elements using a basis of eigenfunctions that make sense for this potential. In particular we can choose a basis of free-rotor eigenfunctions, $\exp(im\theta)$ which, of course, diagonalize the free-rotor Hamiltonian

$$H_{FR} = -\frac{\hbar^2}{2I}\frac{d^2}{d\theta^2}. \tag{3.5}$$

However it would be better to choose free-rotor eigenfunctions such that they reflect the symmetry of the potential *and* diagonalize the free-rotor Hamiltonian. These eigenfunctions are given by

$$\langle\theta|3m+l\rangle = \frac{1}{\sqrt{2\pi}}e^{i(3m+l)\theta}, m = 0, \pm1, \pm2, \cdots \tag{3.6}$$

for $l = [-1, 0, 1]$. It is important to point out here that we will need to diagonalize three Hamiltonians corresponding to each of the $l = [-1, 0, 1]$ terms.

We calculate the matrix elements of the free-rotor Hamiltonian by inserting the free-rotor basis states into the free-rotor Hamiltonian and calculate $\langle n|H_{FR}|m\rangle$ as follows

$$\begin{aligned}
\langle 3n+l|H_{FR}|3m+l\rangle &= \int_0^{2\pi} \mathrm{d}\theta\langle 3n+l|\theta\rangle H_{FR}\langle\theta|3m+l\rangle && (3.7)\\
&= \frac{1}{2\pi}\int_0^{2\pi} \mathrm{d}\theta e^{-i(3n+l)\theta}\left(-\frac{\hbar^2}{2I}\frac{d^2}{d\theta^2}\right)e^{i(3m+l)\theta} && (3.8)\\
&= \frac{\hbar^2}{2I}(3m+l)^2\,\delta_{m,n}. && (3.9)
\end{aligned}$$

Note that the quantity $\hbar^2/2I$ is frequently referred to as the rotational constant and is represented by B. In this derivation then the free-rotor energy is given by $E_{FR} = B(3n+l)^2$ where $n = 0, \pm1, \pm2, \cdots$ and l can take on values of -1,0, and 1.

We calculate the matrix elements of the potential similarly:

$$\begin{aligned}
\langle 3n+l|V|3m+l\rangle &= \int_0^{2\pi} \mathrm{d}\theta\langle 3n+l|\theta\rangle V(\theta)\langle\theta|3m+l\rangle && (3.10)\\
&= \frac{1}{2\pi}\int_0^{2\pi} \mathrm{d}\theta e^{-i(3n+l)\theta}\frac{V_3}{2}\left(1-2e^{3i\theta}-2e^{-3i\theta}\right)e^{i(3m+l)\theta} && (3.11)\\
&= \frac{V_3}{4}\left(2\delta_{m,n}-\delta_{m,n+1}-\delta_{m,n-1}\right). && (3.12)
\end{aligned}$$

So the matrix elements of the Hamiltonian $H_{m,n} = \langle 3n+l|H|3m+l\rangle$ are

$$H_{m,n} = B(3m+l)^2\,\delta_{m,n} + \frac{V_3}{4}\left(2\delta_{m,n}-\delta_{m,n+1}-\delta_{m,n-1}\right). \tag{3.13}$$

Note that had we selected the free-rotor basis set

$$\langle\theta|m\rangle = \frac{1}{\sqrt{2\pi}}e^{im\theta}, m = 0, \pm1, \pm2, \cdots . \tag{3.14}$$

then the Hamiltonian matrix would consist of diagonal elements (due to the free-rotor part of the Hamiltonian) and elements that are further off-diagonal than when we use those with the three-fold symmetry as in equation 3.6. The result of our choice is that it takes a smaller matrix—and hence shorter evaluation time—to get the same accuracy as the expansion in $\exp(im\theta)$, even though we must diagonalize three Hamiltonians.

** **Ex. 10** — Consider a potential with a three-fold and a six-fold term: $V(\theta) = \frac{V_3}{2}(1 - \cos 3\theta) + \frac{V_6}{2}(1 - \cos 6\theta)$. How does equation 3.13 change with the addition of this term?

3.2.1 Solving the Hamiltonian

The construction of the Hamiltonian described in the last section, equation 3.13, is as straightforward to implement in IDL as it is for the one-dimensional DVA discussed in the previous chapter. The function `METHYL_ROTATION` performs the relevant calculation and is shown below. This function has a number of input keywords and output keywords. Note that the function has been written so that they are all optional (though the function is not necessarily that useful if no output keywords are specified since the return value for the function is a logical 1 or 0). The input keywords are $\mathtt{V}_3$, the barrier height, `LMAX`, the maximum value for the expansion in equation 3.6 but here `LMAX` is the maximum value for m in that expansion. The rotational constant, `B`, is 0.654 meV for a methyl rotor but the user can re-define it via this keyword. The `EIGENVALUES` output keyword is a vector of eigenvalues from the solution of the Schrödinger equation with the Hamiltonian, equation 3.3. If you specify `THETA` then the function will return an equally-spaced grid of 200 values between 0 and 2π then the output keyword `PSI` returns a matrix of eigenvectors from the diagonalized Hamiltonian. `PSI` has the dimensions `PSI[N,3*(2*LMAX+1)]` where `N`, the number of points that define the angular grid, is 200 unless the user specifies otherwise as an input keyword. Finally, if the function fails for some reason, it will return a `0B` and the keyword `ERR_MESSAGE` will provide some information why the function failed.

```
function methyl_rotation,  V3 = v3,                    $
                           LMAX = lmax,                $
                           B = b,                      $
                           EIGENVALUES = eigenvalues,  $
                           PSI = psi,                  $
                           THETA = theta,              $
                           N = n,                      $
                           ERR_MESSAGE = err_message
compile_opt idl2,hidden
catch,the_error
ERR_MESSAGE = 'No errors detected'
if the_error ne 0 then begin
  catch,/cancel
  ERR_MESSAGE = !error_state.msg
  return,0B
```

```
endif
if n_elements(V3) eq 0 then v3 = 42.5
if n_elements(LMAX) eq 0 then lmax = 9
if n_elements(B) eq 0 then b = 0.654
if n_elements(N) eq 0 then n = 200
nsize = 2*lmax + 1  ; total number of terms in expansion
lo = indgen(lmax)+1
lvec = [-reverse(lo),0,lo]
l = rebin(lvec,nsize,nsize,/sample)
lp = transpose(l)
k = [-1,0,1] & nk = n_elements(k)
c = 1./sqrt(2.*!pi)
xlo = 0.0 & xhi = 2.*!pi
xo = linspace(xlo,xhi,n)
x = rebin(transpose(xo),nsize,n,/sample)
lm = rebin(lvec,nsize,n,/sample)
if arg_present(theta) then begin
  psi_re = fltarr(nk*nsize,n)
  psi_im = fltarr(nk*nsize,n)
endif
count = 0
for i = 0,nk-1 do begin
    h = ((3.*l+k[i])^2)*B*ds_delta(l,lp)+0.25*v3*$
      (2.0*ds_delta(l,lp) - $
      ds_delta(l,lp+1)-ds_delta(l,lp-1))
      evalso = real_part(la_eigenql(h,/double,eigenvectors = eveco))
      evalso = real_part(evalso)
    if arg_present(theta) then begin
;     Cycle through each of the eigenvalues and construct the
;     real and imaginary parts of the wavefunction. But only do
;     this if the user has asked for it.  It is an expensive
;     operation.
      for j = 0,nsize-1 do begin
        u = eveco[*,j]
        argm = (3.*lvec+k[i])#xo
        psi_re[count,*] = c*u#cos(argm)
        psi_im[count,*] = c*u#sin(argm)
        count++
      endfor
    endif
    if (i gt 0) then evals = [evals,evalso] else evals = evalso
endfor
esort = sort(evals)
eigenvalues = evals[esort]
if arg_present(theta) then begin
  psi = complex(psi_re,psi_im)
  psi = transpose(psi[esort,*])
  theta = xo
endif
```

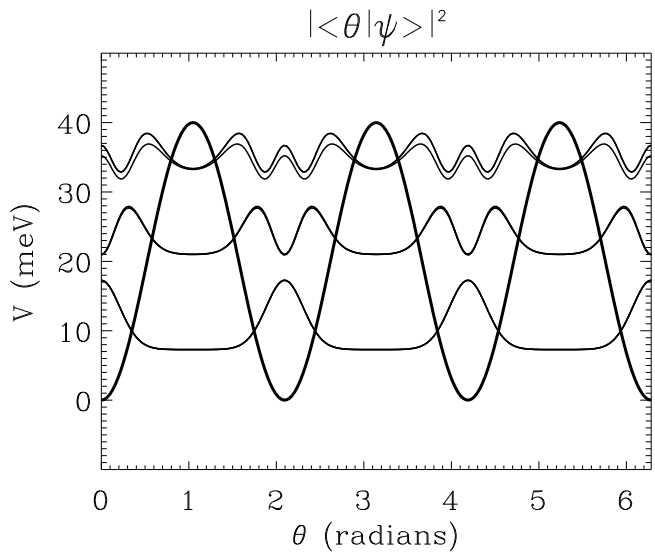


Figure 3.1: Probability densities corresponding to the lowest 9 eigenstates (thin, solid curves) for a methyl rotor confined in three-fold potential with a 40 meV barrier height (thick solid curve).

```
catch,/cancel
return,1B
end
```

The probability densities corresponding to the 9 lowest eigenstates for $V_3 = 40$ meV are displayed in figure 3.1. They are offset in the vertical direction by their corresponding eigenvalues in meV. Note that the probability densities for the lowest three eigenstates are indistinguishable from each other on this scale, as they are in the next three. Only the the three probability densities with eigenstates near the top of the well show distinction from each other. Indeed two of them lie on top of each other. The lowest 9 eigenvalues and their degeneracies are shown in table 3.1.

Table 3.1: Lowest eigenvalues for $V_3 = 40$ meV

eigenvalue (meV)	degeneracy
7.281	1
7.284	2
20.923	2
21.021	1
31.858	1
32.823	2

The spectrum of eigenvalues changes quite dramatically as the barrier height V_3 changes. Even without doing any numerical computations we can guess what the behavior is in two extremes: zero barrier and very high barrier. When the barrier is very small or zero, the eigenvalues

correspond to those of a free rotor

$$E_n = \frac{\hbar^2}{2I}(3n+l)^2 \tag{3.15}$$

$$= B(3n+l)^2. \tag{3.16}$$

For a very high barrier the potential can be approximated well by three adjacent parabolic wells—at least for the lowest lying eigenstates. The eigenstates are very nearly triply-degenerate harmonic oscillator states whose eigenvalues are given by

$$E_r = \left(r + \frac{1}{2}\right)\hbar\omega_0 \tag{3.17}$$

where

$$\omega_0 = \sqrt{\frac{9V_3}{2I}}. \tag{3.18}$$

For a derivation of this equation, see exercise 9.

The manner in which these two extremes are joined can be determined in a straightforward numerical calculation of the eigenvalues as a function of V_3. The calculation is illustrated in the code listing for `METHYL_EIGENVALUE_SPECTRUM` and the spectrum is displayed in figure 3.2.

```
pro methyl_eigenvalue_spectrum
; This procedure creates a plot of the eigenvalue
; spectrum as a function of barrier height.
nv3 = 101 & v3 = linspace(0.0,100.0,nv3)
lmax = 7
eigvals = fltarr(3*(2*lmax+1),nv3)
for j = 0,nv3-1 do begin
  ret = methyl_rotation(lmax = lmax,v3 = v3[j],eigenvalues = eout)
  eigvals[*,j] = eout
endfor
xsize = 800 & ysize = 600
window,0,xsize = xsize,ysize = ysize
yr = [0.0,100.0] & xr = [min(v3),max(v3)]
plot,v3,eigvals[*,0],yrange = yr,/ystyle,xrange = xr,/xstyle, /nodata,$
  psym = 0,linestyle = ls,background = fsc_color('white'),color = fsc_color('black'), $
  xtitle = '!6V!d3!n (meV)',ytitle = '!6Eigenvalues (meV)', $
  title = '!6Eigenvalues of the Methyl Rotor',charsize = 1.5
for j = 0,3*(2*lmax+1)-1 do oplot,v3,eigvals[j,*],linestyle = ls,psym = 0, $
  color = fsc_color('red'),thick = 2.0
end
```

Ex. 11 — What happens to the value of the lowest eigenvalue if you double the mass of the rotor (i.e. increase I, the moment of inertia, by a factor of 2)? Assume that the barrier height is very large.

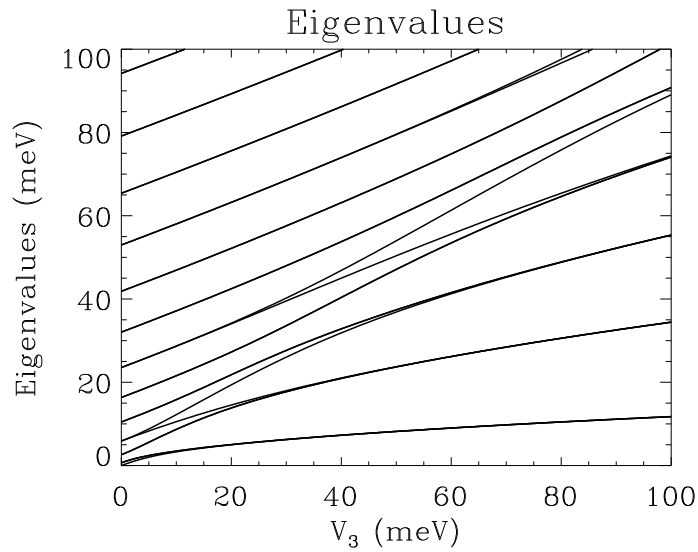


Figure 3.2: The eigenvalue spectrum as a function of barrier height V_3. Note how many of the lower lying branches merge as the barrier height V_3 increases. Though it appears that two branches merge into one, in reality one of these branches is actually doubly degenerate so that *three* branches really make up the single "merged" branch at high barrier heights.

Ex. 12 — Calculate the ground state energy (i.e. lowest eigenvalue) for barrier heights V_3 varying between 0 and 100 meV. Compare this (in a plot) with the approximate expression you derived previously in exercise 9 for the ground state energy in the harmonic approximation. How quickly do these converge, if at all?

Ex. 13 — Calculate the transition between the ground state and first excited state for $0 \leq V_3$ (meV) ≤ 100 for a methyl rotor. What is the approximate functional form for ΔE_{01} vs. V_3 for large V_3? (Hint: plot ΔE_{01} vs. V_3 on a semi-logarithmic scale.) Based on your result, do you think that rotational tunneling is a good or bad probe of the barrier height for methyl-containing systems?

** **Ex. 14** — Consider a potential with a three-fold and a six-fold term: $V(\theta) = \frac{V_3}{2}(1 - \cos 3\theta) + \frac{V_6}{2}(1 - \cos 6\theta)$. How do the eigenvalues shown in figure 3.2 change for different barrier heights, V_6?

3.2.2 Transitions

Transitions between eigenstates of the three-fold potential can be viewed easily with the methods developed so far. In the same way that we constructed an initial state composed of a superposition of two eigenstates in the last chapter, we can do the same here. Let's consider the initial state

given by

$$\langle\theta|\psi\rangle(0) = \frac{1}{\sqrt{2}}\left(\langle\theta|E_0\rangle + \langle\theta|E_1\rangle\right) \tag{3.19}$$

where $\langle\theta|E_0\rangle$ is the lowest energy eigenstate for the system and $\langle\theta|E_1\rangle$ is the first excited state. This is shown in figure 3.3 for $V_3 = 40$ meV.

The code to view the animation is shown below.

```
pro animate_methyl_transition,n,m
; Animates the transition
; between the nth and mth eigenstates for the
; hindered methyl rotor.
if n_params() ne 2 then begin
  n = 0 & m = 1
endif
v3 = 40.0  ; barrier height in meV
ret = methyl_rotation(v3 = v3,eigenvalues = eout, $
      psi = psi,theta = theta)
v = 0.5*(1.0-cos(3.*theta))
wfn = psi[*,n] & wfm = psi[*,m]
wn = eout[n]/0.6528 & wm = eout[m]/0.6528
i = complex(0.0,1.0)
period = 2.*!pi/abs(wn-wm)
nt = 300 & t = linspace(0.0,2*period,nt)
wf = wfn#exp(-i*wn*t) + wfm#exp(-i*wm*t)
prob = abs(wf)^2
xr = [-1.,1.] & yr = xr
sf = 0.75*max(v)/max(prob)
xsize = 500 & ysize = 500 & winvis = 0
window,winvis,xsize = xsize,ysize = ysize
window,/free,/pixmap,xsize = xsize,ysize = ysize
winpix = !d.window
f = '(f10.2)'
for j = 0,nt-1 do begin
  wset,winpix
  title = 'T = '+strtrim(string(1.e-3*t[j],format = f),2) + ' ns'
  plot,v,theta,psym = 0,thick = 2.0,color = fsc_color('black'),  $
    background = fsc_color('white'),/nodata,/polar, $
    xrange = xr,yrange = yr,xstyle = 5,ystyle = 5,$
    title = title,charsize = 2
  oplot,v,theta,thick = 2.0,color = fsc_color('blue'),/polar
  oplot,sf*prob[*,j],theta,thick = 2.0,color = fsc_color('red'),/polar
  r = sf*prob[*,j] & dr = 0.1*r
  polyfill,[0,r*cos(theta)],[0,r*sin(theta)],$
    color = fsc_color('red')
  wset,winvis
  device,copy = [0,0,!d.x_size,!d.y_size,0,0,winpix]
endfor
end
```

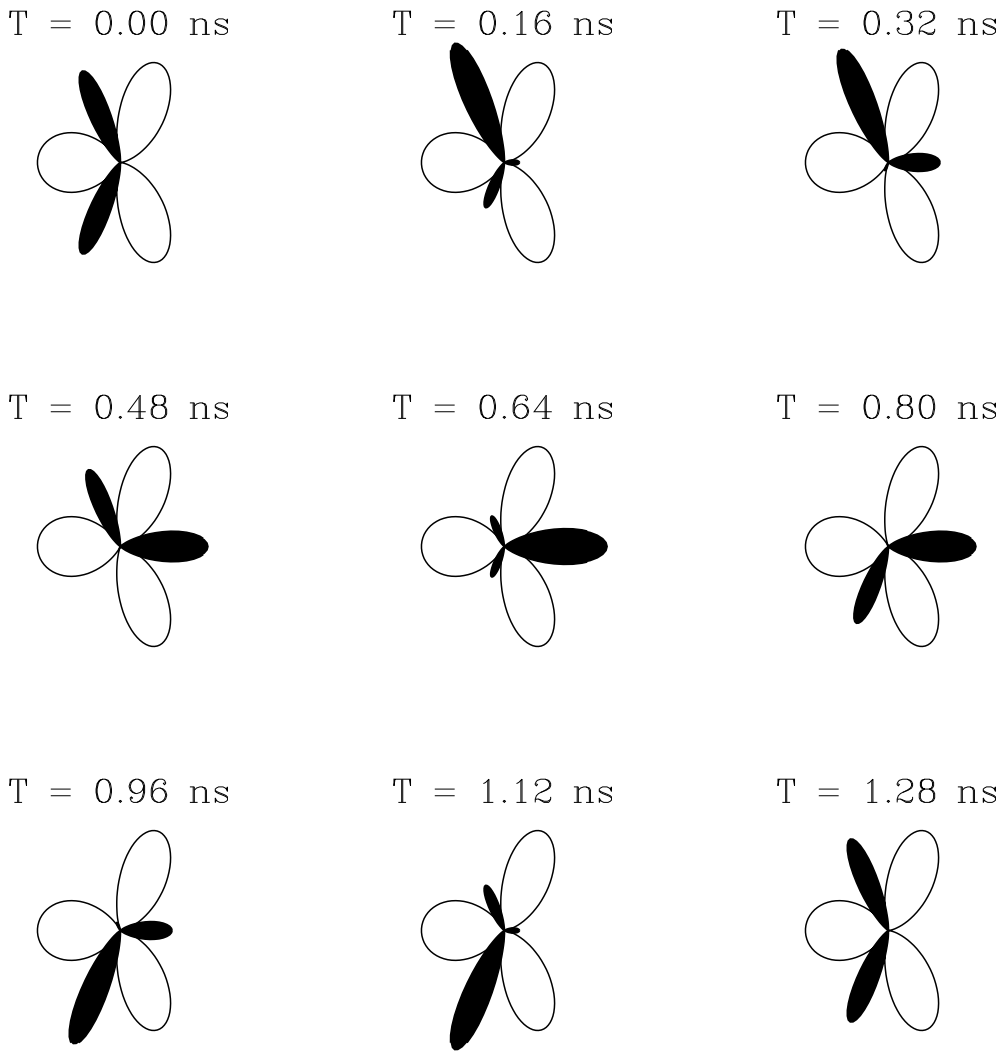


Figure 3.3: Polar plot of the time-evolution for transitions between the lowest two eigenstates for $V_3 = 40$ meV. The initial state is an equal mixture of the two lowest eigenstates. Over time the "particle" tunnels from one potential minimum to another. The potential is shown in blue and the probability density is shown in red.

Ex. 15 — View animations of the tunneling transitions $m = 0 \to 1$ and $m = 0 \to 2$ for $V_3 = 40$ meV. What is the primary difference you observe in the time evolution? Hint: Look at the time-evolution of the initial states $\frac{1}{\sqrt{2}}\left(\langle\theta|E_0\rangle + \langle\theta|E_1\rangle\right)$ and $\frac{1}{\sqrt{2}}\left(\langle\theta|E_0\rangle + \langle\theta|E_2\rangle\right)$ respectively.

The quantum analog of librational motion—also known as torsional oscillation—is shown in figure 3.4 (again, $V_3 = 40$ meV). It is clear from the figure that the probability density oscillates within each of the potential minima.

3.3 Hindered Dumbell Rotations

The hydrogen molecule, H_2, is composed of two hydrogen atoms that are nearly free to rotate in three dimensions. However there have been numerous experiments performed where researchers have adsorbed H_2 onto surfaces and into matrices (porous materials) where interactions with the surfaces or matrix leads to hindered rotations. Computation of H_2's energy spectrum is therefore quite useful so that one can compare the model's transitions between eigenstates to the experimentally-obtained energy spectrum.

H_2 molecules are spherical to an excellent approximation and, as such, form an idealized three-dimensional rotor. For more details on the development that follows, consult any standard undergraduate text on quantum mechanics such as [2]. The most natural way to describe the system is in spherical coordinates. For zero potential the Schrödinger equation in spherical coordinates is given by

$$-\frac{\hbar^2}{2m}\left[\frac{\partial^2}{\partial r^2} + \frac{2}{r}\frac{\partial}{\partial r} + \frac{1}{r^2}\left(\frac{\partial^2}{\partial\theta^2} + \cot\,\theta\frac{\partial}{\partial\theta} + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right)\right]\langle r,\theta,\phi|\psi\rangle = E\langle r,\theta,\phi|\psi\rangle. \tag{3.20}$$

This equation is separable so the wavefunction is separable: $\langle r,\theta,\phi|\psi\rangle = \langle r|f\rangle\langle\theta,\phi|g\rangle$. To a first approximation we can assume that the distance between nuclei in a diatomic molecule like H_2 is fixed. This means that the $r-$dependence is uninteresting. Furthermore we will only consider potentials for which there is no $r-$dependence and only orientational dependence. Since the Schrödinger equation in spherical coordinates is separable we will focus only on the orientational part.

The eigenvalue form of the orientational part of the Schrödinger equation is given by

$$-\hbar^2\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right)Y_{\ell,m}(\theta,\phi) = l\,(l+1)\,\hbar^2 Y_{\ell,m}(\theta,\phi) \tag{3.21}$$

where the eigenvalues are given by ℓ for $\ell = 0, 1, 2, \cdots;\ m = -\ell, -(\ell-1), \cdots, 0, \cdots, (\ell-1), \ell$; and $Y_{\ell,m}(\theta,\phi)$ are the spherical harmonics. The spherical harmonics are defined as

$$Y_{\ell,m}(\theta,\phi) = \sqrt{\frac{(2\ell+1)}{4\pi}\frac{(\ell-m)!}{(\ell+m)!}}P_\ell^m(\cos\,\theta)\,e^{im\phi} \tag{3.22}$$

where the P_ℓ^m is an associated Legendre function. Note that the mass factor has dropped out due to the way in which the Schrödinger equation has been separated. (Actually it is included in the "uninteresting" radial equation.) This equation is often written in operator form as

$$\mathbf{L}^2 Y_{\ell,m}(\theta,\phi) = l\,(l+1)\,\hbar^2 Y_{\ell,m}(\theta,\phi) \tag{3.23}$$

where

$$\mathbf{L}^2 \equiv -\hbar^2\left(\frac{1}{\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}\right) + \frac{1}{\sin^2\theta}\frac{\partial^2}{\partial\phi^2}\right). \tag{3.24}$$

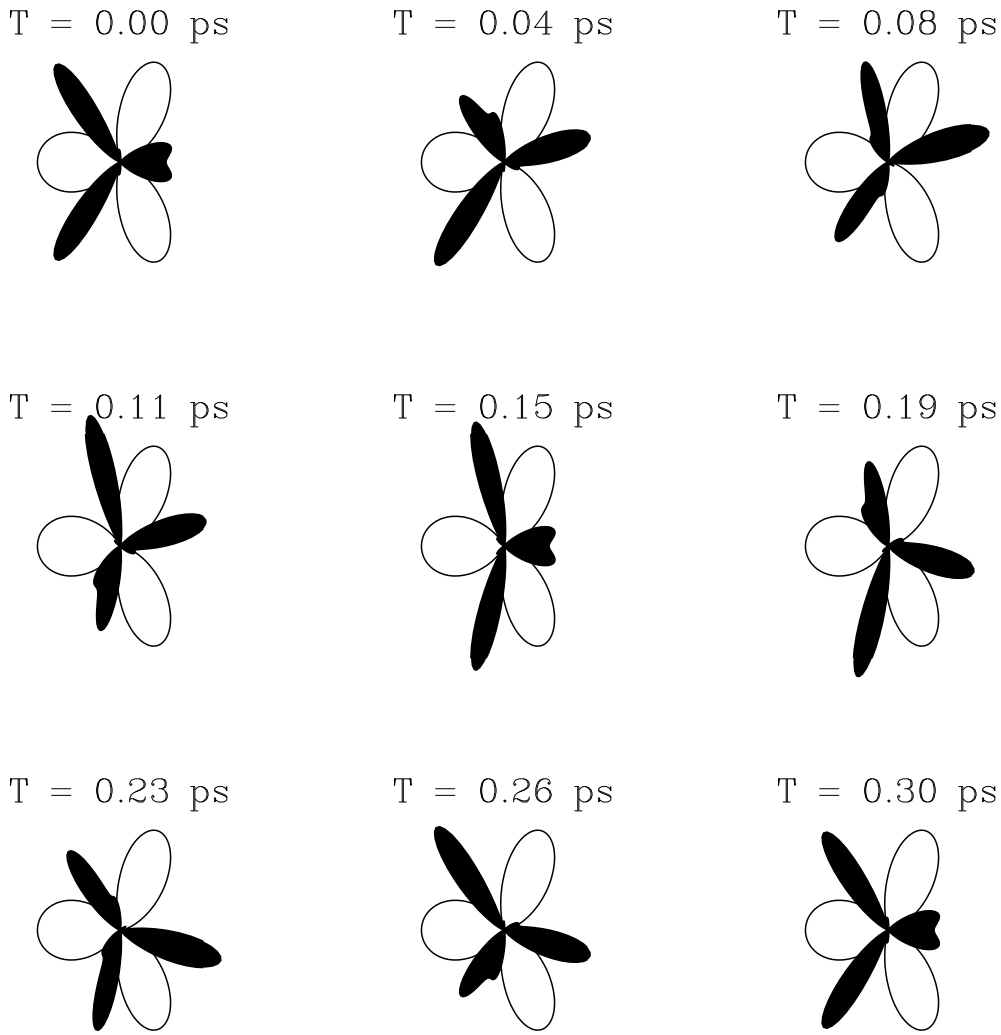


Figure 3.4: Polar plot of the time-evolution for transitions between the $m = 0$ and $m = 3$ eigenstates for $V_3 = 40$ meV. The initial state is an equal mixture of these two eigenstates. Note how the probability density oscillates within each of the three wells. This is the quantum analog of libration—torsional oscillation. The potential is shown in blue and the probability density is shown in red.

For an orientational potential, $V(\theta,\phi)$, we can assume that the wavefunction that satisfies the orientational part of the Schrödinger equation can be expanded in a series of the spherical harmonics, $Y_{l,m}(\theta,\phi)$. Therefore we can write down the matrix elements for the Hamiltonian as

$$\begin{aligned} H_{\ell,\ell',m,m'} &= T_{\ell,\ell',m,m'} + V_{\ell,\ell',m,m'} && (3.25)\\ &= \langle \ell', m'|\frac{\mathbf{L}^2}{2I}|\ell, m\rangle + \langle \ell', m'|V(\theta,\phi)|\ell, m\rangle && (3.26)\\ &= B\ell(\ell+1)\delta_{\ell,\ell'}\delta_{m,m'} + \langle \ell', m'|V(\theta,\phi)|\ell, m\rangle && (3.27)\end{aligned}$$

where the rotational constant, B is given by $B = \hbar^2/2I$. It is clear that we must evaluate the matrix elements of the potential. For the purposes of our calculation we choose a two-fold hindering potential,

$$V(\theta,\phi) = \frac{V_2}{2}(1-\cos 2\theta) \tag{3.28}$$

where V_2 is the barrier height. We will find it helpful to express the potential in terms of the spherical harmonics in order to calculate the matrix elements explicitly. Since

$$Y_{2,0}(\theta,\phi) = \sqrt{\frac{5}{4\pi}}\left(\frac{3}{2}\cos^2\theta - \frac{1}{2}\right), \tag{3.29}$$

a little algebra allows us to re-write the potential as

$$V(\theta,\phi) = \frac{2V_2}{3} - \sqrt{\frac{16\pi}{45}}V_2 Y_{2,0}(\theta,\phi). \tag{3.30}$$

Now we can calculate the matrix elements for the potential, $V_{\ell,\ell',m,m'}$.

$$\begin{aligned} V_{\ell,\ell',m,m'} &= \langle Y^*_{\ell',m'}|\frac{2V_2}{3} - \sqrt{\frac{16\pi}{45}}V_2 Y_{2,0}(\theta,\phi)|Y_{\ell,m}\rangle && (3.31)\\ &= \frac{2V_2}{3}\delta_{\ell,\ell'}\delta_{m,m'} - \sqrt{\frac{16\pi}{45}}V_2\langle Y_{\ell',m'}|Y_{2,0}|Y_{\ell,m}\rangle. && (3.32)\end{aligned}$$

The integral over three different spherical harmonics can be written in terms of the Wigner 3-j symbol [14]. See appendix A for more details on the 3-j symbol and the IDL implementation.

$$\begin{aligned} \langle Y_{\ell_1,m_1}|Y_{\ell_2,m_2}|Y_{\ell_3,m_3}\rangle &= (-1)^{m_1}\int_0^{2\pi} d\phi \int_0^{\pi} d\theta \sin\theta Y_{\ell_1,-m_1}Y_{\ell_2,m_2}Y_{\ell_3,m_3} && (3.33)\\ &= (-1)^{m_1}\left(\frac{(2\ell_1+1)(2\ell_2+1)(2\ell_3+1)}{4\pi}\right)^{1/2}\times && (3.34)\\ &\quad \begin{pmatrix}\ell_1 & \ell_2 & \ell_3\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\ell_1 & \ell_2 & \ell_3\\ -m_1 & m_2 & m_3\end{pmatrix}, \end{aligned}$$

where we have used the relationships

$$\begin{aligned} \int_0^{2\pi} d\phi \int_0^{\pi} d\theta \sin\theta Y_{\ell_1,m_1}Y_{\ell_2,m_2}Y_{\ell_3,m_3} &= \left(\frac{(2\ell_1+1)(2\ell_2+1)(2\ell_3+1)}{4\pi}\right)^{1/2}\times && (3.35)\\ &\quad \begin{pmatrix}\ell_1 & \ell_2 & \ell_3\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\ell_1 & \ell_2 & \ell_3\\ m_1 & m_2 & m_3\end{pmatrix} \end{aligned}$$

and $Y^*_{\ell,m} = (-1)^m Y_{\ell,-m}$. Using this we may write the matrix elements for the potential term in the Hamiltonian as

$$\begin{aligned} V_{\ell,\ell',m,m'} &= \frac{2V_2}{3}\delta_{\ell,\ell'}\delta_{m,m'} - \\ &\quad (-1)^{m'}\sqrt{\frac{16\pi}{45}}V_2\left(\frac{5\,(2\ell'+1)\,(2\ell+1)}{4\pi}\right)^{1/2}\begin{pmatrix}\ell' & 2 & \ell\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\ell' & 2 & \ell\\ -m' & 0 & m\end{pmatrix}. \end{aligned}$$

Collecting terms in the final Hamiltonian yields

$$\begin{aligned} H_{\ell,\ell',m,m'} &= \left(B\ell\,(\ell+1)+\frac{2V_2}{3}\right)\delta_{\ell,\ell'}\delta_{m,m'} - \\ &\quad (-1)^{m'}\sqrt{\frac{16\pi}{45}}V_2\left(\frac{5\,(2\ell'+1)\,(2\ell+1)}{4\pi}\right)^{1/2}\begin{pmatrix}\ell' & 2 & \ell\\ 0 & 0 & 0\end{pmatrix}\times \\ &\quad \begin{pmatrix}\ell' & 2 & \ell\\ -m' & 0 & m\end{pmatrix}. \end{aligned}$$

With this expression and the IDL function to calculate the Wigner 3-j symbol, `wigner_threeJ`, we can construct the Hamiltonian $H_{\ell,\ell',m,m'}$. However, because the Wigner 3-j symbol function we've implemented in IDL is not a vectorized function, we are not able to take advantage of the matrix operations that we did in the previous examples. We have to resort to a brute-force approach when constructing the Hamiltonian. The implementation is shown in the code `DUMBELL_SOLVER.PRO` where the eigenvalues and eigenvectors are computed for a particular value of V_2. In our implementation we must write the Hamiltonian as a square matrix (again, because the IDL eigensystem function `LA_EIGENQL` requires a square matrix) by including all of the values for ℓ and m in each of the two dimensions. Specifically we compute the matrix elements for $\ell = 0, 1, \cdots, \ell_{max}$; $m = -\ell, -(\ell-1), \cdots, 0, \cdots, (\ell-1), \ell$; $\ell' = 0, 1, \cdots, \ell_{max}$; and $m' = -\ell', -(\ell'-1), \cdots, 0, \cdots, (\ell'-1), \ell'$. For a two-dimensional matrix this means that the length of each dimension will be

$$\begin{aligned} \sum_{\ell=0}^{\ell_{max}}\sum_{m=-\ell}^{\ell} 1 &= \sum_{\ell=0}^{\ell_{max}}(2\ell+1) \\ &= 2\sum_{\ell=0}^{\ell_{max}}\ell + \sum_{\ell=0}^{\ell_{max}} 1 \\ &= \ell_{max}\,(\ell_{max}+1) + (\ell_{max}+1) \\ &= (\ell_{max}+1)^2 \end{aligned}$$

The IDL function that calculates the eigenvalues for a dumbell rotor is listed below. The interface allows the user to specify the rotational constant, B. For a H_2 molecule $B = 7.35$ meV. However often we would like to view the the eigenvalue spectrum as a function of reduced units (units of B). We can do this by setting $B = 1$ when invoking the function. The interface to this function also allows the user to specify the barrier height, V_2, and the maximum value for ℓ or ℓ_{max}. The default value for the maximum angular momentum quantum number is $\ell_{max} = 5$. Note that ℓ is not a good quantum number when $V_2 \neq 0$.

```
function dumbell_solver, B = B,      $
                         V2 = V2,    $
```

```
                          LMAX = LMAX
compile_opt idl2,hidden
if n_elements(B) eq 0 then B = 7.35  ; rotational constant (meV)
if n_elements(V2) eq 0 then V2 = 0.0 ; hindering barrier for two-fold
; symmetric potential (meV)
if n_elements(LMAX) eq 0 then LMAX = 5
; Create the Hamiltonian
nl = lmax+1 & hsize = nl*nl
l = indgen(lmax+1) & nl = n_elements(l)
m = -lmax+indgen(2*lmax+1) & nm = n_elements(m)
h2 = fltarr(hsize,hsize)
ix = 0L & iy = 0L
for j = 0,nl-1 do begin
  m = -j+indgen(2*j+1)
  for i = 0,2*j do begin
    iy = 0
    for jj = 0,nl-1 do begin
      mm = -jj+indgen(2*jj+1)
        for ii = 0,2*jj do begin
        term1 = (B*l[j]*(l[j]+1)+2.*V2/3.)*ds_delta(l[j],l[jj])*ds_delta(m[i],mm[ii])
        term20 = ((-1.)^mm[ii])*V2*sqrt(16.*!pi/45.)*$
          (sqrt(5.*(2.*l[jj]+1.0)*(2.*l[j]+1.0)/(4.*!pi)))*$
          wigner_threeJ(l[jj],2,l[j],0,0,0)*$
          wigner_threeJ(l[jj],2,l[j],-mm[ii],0,m[i])
        h2[ix,iy] = term1-term20
        iy++
      endfor
    endfor
    ix++
  endfor
endfor
; Find the eigenvalues of the Hamiltonian
evalso = la_eigenql(h2,/double)
esort = sort(evalso)
evalso = evalso[esort]
return,evalso
end
```

The procedure below, `DUMBELL1`, creates a plot of the eigenvalue spectrum for a broad range of values, V_2, as shown in the figure.

```
pro dumbell1
; Compute the eigenvalues for a range of barrier heights
nv = 20
v2 = linspace(0.0,20.0,nv)
for j = 0,nv-1 do begin
  evals = dumbell_solver(B = 1.0,v2 = v2[j])
```

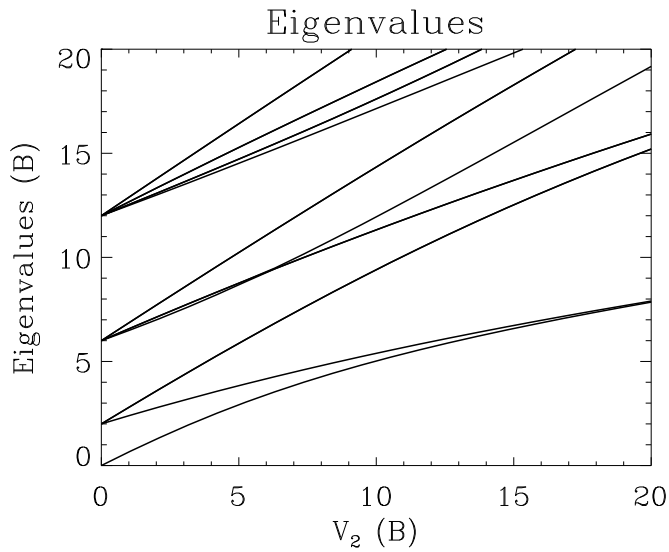


Figure 3.5: Eigenvalue spectra for a dumbell rotor as a function of barrier height V_2. The calculation was done for 10 values of V_2 between 0 and 20 with a rotational constant $B = 1$ and $\ell_{max} = 5$.

```
  if j eq 0 then eout = evals else eout = [[eout],[evals]]
endfor
es = size(eout,/dimension)
n = es[0]
eout = eout[0:n-1,*]
emax = max(eout,min = emin)
plot,v2,eout[0,*],yrange = [0.0,20],/ystyle,psym = 0, $
  color = fsc_color('black'),background = fsc_color('white'),$
  title = '!6Eigenvalues',xtitle = '!6V!d2!n (B)',$
  ytitle = '!6Eigenvalues/B',/nodata,charsize = 1.5
for j = 0,n-1 do oplot,v2,eout[j,*],psym = 0,thick = 2.0,$
  color = fsc_color('red')
end
```

Ex. 16 — Verify that the algorithm produces the expected free-rotor eigenvalues. Specifically for $V_2 = 0$, calculate the eigenvalues and show that they follow $E/B = \ell(\ell+1)$ for $\ell = 0, 1, \cdots, \ell_{max}$.

* **Ex. 17** — For $V_2 = 10B$ by what percentage does the 5th eigenvalue change when the eigenvalues are calculated for a Hamiltonian with $\ell_{max} = 5$ compared with a Hamiltonian calculated for $\ell_{max} = 6$? How much does the 17th eigenvalue change? Based on your result, should we be using more terms in calculating the Hamiltonian? Hint: remember to set $B = 1$ in order to ensure that you are calculating eigenvalues in units of B.

*** **Ex. 18** — Using the basis of spherical harmonics, $Y_{\ell,m}(\theta,\phi)$, determine the matrix elements $H_{\ell,\ell',m,m'}$ corresponding to a Hamiltonian with a four-fold symmetric potential energy, $V(\theta) = \frac{V_4}{2}(1-\cos 4\theta)$. Hint: first express the potential in terms of the spherical harmonics. Plot the eigenvalues for a range of V_4: $0 \leq V_4 \leq 20$.

Chapter 4

Quantum Dynamics in One Dimension

Until this chapter our discussions have focused on solving the time-independent Schrödinger equation. We described dynamic phenomena as resulting from an initial superposition of eigenstates. The time-evolution operator then provides the time dependence for the evolving state. In *real world* applications such as particle scattering, the particles are often represented as having a location and spread. Thus they have a finite extent and can be characterized by a peak position. The Gaussian wavepacket is a waveform that matches this description. In many cases it would be preferable to solve the time-dependent Schrödinger equation for a potential and specify a general form for the initial state, such as a propagating Gaussian wavepacket. In this chapter we will describe just such a method that allows the user to specify such a particle-like initial state, a propagating Gaussian wavepacket, and compute how this wavepacket interacts with a user-defined potential in time. Many observable characteristics of quantum scattering will be evident from our calculations, including dispersion, tunneling, and virtual or metastable states.

4.1 The Goldberg-Schey-Schwartz Algorithm

In 1967 Goldberg, Schey, and Schwartz published a paper in the American Journal of Physics that described an algorithm[1] to compute the time-development of a general wavepacket scattering from a one-dimensional potential.[9] This seminal paper spawned numerous follow-on papers and even a set of film loops depicting scattering phenomena.[15] It is still a very highly cited article. As an aside, one of the fascinating aspects of this paper is that it described the state-of-the-art in 1967 for converting the computer output on a cathode-ray tube to film. It is interesting to note some of the comments made describing the challenges that the authors encountered in doing this transfer.

> *The reader will note occasional breaks or discontinuities at various points in some of these pictures. This effect is due to an evidently inherent malfunction of the equipment involved in rendering the machine calculations into graphical form. At the present time there appears to be no simple way to avoid this problem; but, fortunately, the discontinuities, while annoying, by no means destroy the effect of the film. This situation*

[1] Hereafter referred to as the GSS algorithm

> *emphasizes the fact that the use of computers to illustrate time development in physical systems by motion pictures is still in a preliminary, if no longer rudimentary stage.*[9]

Not only did the film loops find great popularity in the classroom for a whole generation of physics students, but the computational technique described in that article is still of great value today. Moreover, this algorithm can be implemented in a straightforward way into IDL and, using a modern desktop computer, allows extremely rapid calculation of the dynamics of a wavepacket scattering from a potential.

4.1.1 Development of the GSS Algorithm

We will follow closely the algorithm development from [9]. We start with choosing a system of units in which $\hbar = 2m = 1$ so that the Schrödinger equation is

$$-\frac{\partial^2 \Psi}{\partial X^2} + \Theta\left(X\right)\Psi(X,T) = i\frac{\partial \Psi}{\partial T}. \tag{4.1}$$

We replaced the physical variables x with X, t with T, and V with Θ to distinguish between the two forms of the Schrödinger equation.

Next we discretize X in $(J+1)$ points equally-spaced by ϵ and discretize T in $(N+1)$ equally-spaced by δ. The action of the Hamiltonian on the discretized wavefunction, Ψ_j^n, is

$$H\Psi_j^n = -\frac{1}{\epsilon^2}\left(\Psi_{j+1}^n + \Psi_{j-1}^n - 2\Psi_j^n\right) + \Theta_j \Psi_j^n, \tag{4.2}$$

which we recall from our earlier discussion of the discrete variable approximation in chapter 2. We apply the time-development operator to the wavefunction in order to propagate our solution in time:

$$\Psi_j^{n+1} = e^{-i\delta H}\Psi_j^n. \tag{4.3}$$

For small time steps δ we might guess that we can make a small-argument expansion of the exponential operator and truncate it after the first-order term,

$$\Psi_j^{n+1} \simeq \left(1 - i\delta H\right)\Psi_j^n. \tag{4.4}$$

Unfortunately the approximation is not unitary and probability will not be conserved. We solve this issue by using the Cayley approximation to the exponential operator:

$$e^{i\delta H} \simeq \frac{1 - i\delta H/2}{1 + i\delta H/2}, \tag{4.5}$$

which is unitary. With this approximation we have

$$\Psi_j^{n+1} \simeq \frac{1 - i\delta H/2}{1 + i\delta H/2}\Psi_j^n, \tag{4.6}$$

or

$$\left(1 + i\delta H/2\right)\Psi_j^{n+1} \simeq \left(1 - i\delta H/2\right)\Psi_j^n. \tag{4.7}$$

From now on we will replace the $\simeq$ with $=$.

When we multiply through we get

$$\Psi_j^{n+1} + \frac{i\delta}{2}H\Psi_j^{n+1} = \Psi_j^n - \frac{i\delta}{2}H\Psi_j^n \tag{4.8}$$

which, upon factoring in the action of the Hamiltonian, can be expanded out

$$\Psi_j^{n+1} + \frac{i\delta}{2}\Theta_j\Psi_j^{n+1} - \frac{i\delta}{2\epsilon^2}\left(\Psi_{j+1}^{n+1} + \Psi_{j-1}^{n+1} - 2\Psi_j^{n+1}\right) = \Psi_j^n - \frac{i\delta}{2}\Theta_j\Psi_j^n + \frac{i\delta}{2\epsilon^2}\left(\Psi_{j+1}^n + \Psi_{j-1}^n - 2\Psi_j^n\right). \tag{4.9}$$

Letting $\lambda = 2\epsilon^2/\delta$ and collecting terms yields

$$\Psi_{j+1}^{n+1} + \left(i\lambda - \epsilon^2\Theta_j - 2\right)\Psi_j^{n+1} + \Psi_{j-1}^{n+1} = -\Psi_{j+1}^n + \left(i\lambda + \epsilon^2\Theta_j + 2\right)\Psi_j^n - \Psi_{j-1}^n. \tag{4.10}$$

This equation is an implicit equation, meaning that Ψ_j^n is not given only in terms of Ψ_j^{n-1}. This means that we must solve this equation using other means. Specifically we will use a *two-sweep* method as it is described in [9]. To begin, we make the following assignments:

$$\Omega_j^n \equiv -\Psi_{j+1}^n + \left(i\lambda + \epsilon^2\Theta_j + 2\right)\Psi_j^n - \Psi_{j-1}^n \tag{4.11}$$

and

$$\Psi_{j+1}^{n+1} = e_j^n\Psi_j^{n+1} + f_j^n. \tag{4.12}$$

We substitute equations 4.11 and 4.12 into equation 4.10 which yields

$$e_n^j\Psi_j^{n+1} + f_j^n + \left(i\lambda - \epsilon^2\Theta_j - 2\right)\Psi_j^{n+1} + \Psi_{j-1}^{n+1} = \Omega_j^n. \tag{4.13}$$

Collecting terms gives

$$\Psi_j^{n+1} = \left(2 + \epsilon^2\Theta_j - i\lambda - e_j^n\right)^{-1}\left(f_j^n - \Omega_j^n\right) + \left(2 + \epsilon^2\Theta_j - i\lambda - e_j^n\right)^{-1}\Psi_{j-1}^{n+1}. \tag{4.14}$$

Noting that we can rewrite equation 4.12 by decrementing the index j to get $\Psi_j^{n+1} = e_{j-1}^{n+1}\Psi_{j-1}^{n+1} + f_{j-1}^n$ and comparing this to the previous equation we see that

$$e_{j-1}^n = \left(2 + \epsilon^2\Theta_j - i\lambda - e_j^n\right)^{-1} \tag{4.15}$$

which can be rewritten as

$$e_j^n = 2 + \epsilon^2\Theta_j - i\lambda - 1/e_{j-1}^n. \tag{4.16}$$

Note that equation 4.16 suggests that e_j^n is time-independent so that we can drop the superscript n. Also, from comparing the equation 4.12 to equation 4.14 we see that

$$f_{j-1}^n = \left(f_j^n - \Omega_j^n\right)e_{j-1} \tag{4.17}$$

or

$$f_j^n = \Omega_j^n + f_{j-1}^n/e_{j-1}. \tag{4.18}$$

Equations 4.16 and 4.18 are the recursion relations that we'll need to solve the Schrödinger equation numerically but first we'll need to impose the boundary conditions to get the starting conditions for the recursions.

Since we are solving the Schrödinger equation in a box, we impose the boundary conditions at the ends $\Psi(0,t) = \Psi(L,t) = 0$ where the box is defined over $0 \leq x \leq L$. This means that $\Psi_0^n = \Psi_J^n = 0$ for all n. We can rewrite equation 4.13 as

$$\Psi_2^{n+1} = \left(2 + \epsilon^2\Theta_1 - i\lambda\right)\Psi_1^{n+1} + \Omega_1^n \tag{4.19}$$

and then compare with equation 4.12 to yield

$$e_1 = 2 + \epsilon^2\Theta_1 - i\lambda \tag{4.20}$$

and

$$f_1^n = \Omega_1^n. \tag{4.21}$$

With the boundary conditions 4.20 and 4.21 and the recursion relations 4.16 and 4.18 we can calculate e_j for all j and f_j^n for all j and n.

Next, substituting $j = J - 1$ into equation 4.12 and solving for Ψ_{J-1}^n we get $\Psi_{J-1}^n = -f_{J-1}^n/e_J$. We have all of the information on the right-hand side so that we can calculate Ψ_{J-1}^n. We can get the next value down the line (i.e. $j = J - 2$) simply. Solving equation 4.12 for Ψ_j^{n+1} yields $\Psi_j^{n+1} = \left(\Psi_{j+1}^{n+1} - f_j^n\right)/e_j$. Substituting $j = J - 2$ into that expression yields the next value down the line, $\Psi_{J-2}^{n+1} = \left(\Psi_{J-1}^{n+1} - f_{J-2}^n\right)/e_{J-2}$, for the wavefunction. Since we just calculated Ψ_{J-1}^n in the previous step, we have enough information on the right hand side of this expression to calculate Ψ_{J-2}^n. We can continue this process on down the line until we reach Ψ_1^n. Note that $\Psi_0^n = 0$ for all n due to the boundary condition.

Though this algorithm is not as straightforward as the DVA in one dimension, it allows one to compute the time-evolution of a wavefunction fairly rapidly. We will see this through numerous examples in the remaining sections of the chapter.

4.1.2 Implementation

The GSS algorithm is not as amenable to matrix or vector operations as the DVA algorithm. Nevertheless we find some economy in our implementation using some of IDL's built-in capabilities. The function that performs the time-evolution of the wavefunction given a potential is called `TDSE_SOLVER` and it is listed below. As the algorithm described in the previous section requires, we define the values for e_j for all time and keep that out of the loop over time. Where we find the most economy in implementation is in the calculation of Ω_j^n. However the recursions, particularly those that require that we step down the chain of spatial points from high to low, do not readily admit a vectorized operation. For those we must use a loop. Fortunately the loops are relatively quick to execute.

```
function tdse_solver,x,t,wf_init,v
compile_opt idl2,hidden
!except = 0
Y = wf_init[*,0]
nx = n_elements(x) & nt = n_elements(t)
dx = (x[1] - x[0]) & dt = t[1] - t[0]
lambda = complex(2.*(dx^2)/dt,0.0)
i = complex(0.0,1.0)    ; define the imaginary unit
                        ; remember not to use i as an array index or loop variable!
; Normalize the initial wavefunction
c = int_tabulated(x,(abs(Y))^2)
Y = Y/sqrt(c)
psi = complexarr(nx,nt) & psi[*,0] = Y
e = complexarr(nx) & omega = complexarr(nx,nt) & f = complexarr(nx,nt)
; Use the recursion relations to find the t = 0 parameters
e[1] = 2.+((dx)^2)*v[1]-i*lambda
for j = 2,nx-2 do e[j] = 2.+(dx^2)*v[j,0]-i*lambda-(1./e[j-1])
omega[1,0] = -Y[2] + (i*lambda+(dx^2)*v[1]+2.)*Y[1]-Y[0]
```

```
f[1,0] = omega[1,0]
; Now evolve the wavefunction in time...
for it = 1,nt-1 do begin
   omega[2:nx-2,it-1] = -psi[3:nx-1,it-1] + $
                        (i*lambda+(dx^2)*v[2:nx-2]+2.0)*psi[2:nx-2,it-1] - $
                        psi[1:nx-3,it-1]
   for j = 2,nx-2 do f[j,it-1] = omega[j,it-1] + f[j-1,it-1] / e[j-1]
   psi[nx-2,1] = -f[nx-2,it-1] / e[nx-2]
   for j = 3,nx-2 do psi[nx-j,it] = (psi[nx-j+1,it] - f[nx-j,it-1]) / e[nx-j]
endfor
return,psi
end
```

The function `TDSE_SOLVER` requires four input parameters and returns a two-dimensional complex array. The four input parameters include `x`, an equally-spaced vector of positions (of length `nx`); `t`, an equally-spaced vector of times (length `nt`); `wf_init` a complex array that contains the initial ($t = 0$) wavefunction (length `nx`); and `v`, the potential defined at each value of `x`. The function returns a complex array `psi` (`nx`×`nt`).

4.1.3 Stability and Selection of Parameters

It is important to choose the parameters in the calculation such that the results are accurate. The accuracy of the GSS algorithm depends on the mesh parameters (e.g. the space step and time step, the magnitude of the potential, and the magnitude of the propagation constant. We will not reproduce the reasoning that leads to the criteria that ensures accuracy but rather list them here and refer the reader to the discussion in the original GSS paper.[9]

The criteria are the following:

$$\left(k_m \epsilon\right)^2 / 12 << 1, \tag{4.22}$$

$$V_{max} \epsilon^2 / 12 << 1, \tag{4.23}$$

$$N \delta^3 \left(k_m^6 - k_0^6\right) / 12 << 1. \tag{4.24}$$

In these expressions, k_m is the maximum momentum component of consequence in the wavefunction, k_0 is the incident wavevector, and V_{max} is the maximum value of the potential. In the computations that follow in this book, we ensure that the conditions above hold. In practice we ensure that the left-hand sides of the three conditions above are less than or equal to 0.1.

A short function named `TDSE_CHECK_PARAMS` has been written to check that the parameters that you've selected meet the conditions described here. It uses the function `MOMENTUM_TRANSFORM` described in chapter 1. The inputs are the spatial grid values `x`, the time values `t`, the initial wavefunction `wf`, and the potential `v`. The function returns a logical 1 if the parameters satisfy the criteria and a 0 if they do not. The function is listed below.

```
function tdse_check_params,x,t,wf,v
compile_opt idl2,hidden
; This function determines whether the computation
```

```
; parameters will likely result in a stable and accurate
; result.  The criteria are based on that specified in
; GSS (1967).
nx = n_elements(x) & nt = n_elements(t)
dx = x[1] - x[0] & dt = t[1] - t[0]
; Transform the wavefunction to get the peak momentum component
str = momentum_transform(x,t,wf)
phi_k = str.phi_k & k = str.k
kenv = abs(phi_k)
kprob = kenv^2
; Find the peak in the momentum distribution, ko
peak = max(kprob,peak_index)
ko = k[peak_index]
; Find the highest momentum component (absolute value)
; Make the determination based on the highest
; momentum component with greater than 1% of the
; peak intensity.
kindex = where(kprob ge 0.05*peak)
kselect = k[kindex] & kprobselect = kprob[kindex]

lambda = 2.*(dx^2)/dt
km = max(abs(kselect))
vmax = max(v)
val1 = (km*dx)^2/12.0
val2 = vmax*(dx^2)/12.0
val3 = nt*(dt^3)*(km^6-ko^6)/12.0
val4 = km*dx/!pi
bound = 0.1
retval = (val1 lt bound) and (val2 lt bound) and (val3 lt bound) and (val4 lt bound)
return,retval
end
```

4.1.4 Getting the Units Right with the GSS Algorithm

We remind ourselves that the algorithm presented here solves the Schrödinger equation with units where $\hbar = 2\text{m} = 1$. It is a straightforward matter of scaling to determine the proper physical units corresponding to those used in the computations. We begin with the time-dependent Schrödinger equation:

$$-\frac{\hbar^2}{2m}\frac{\partial^2\psi}{\partial x^2} + V(x)\psi(x,t) = i\hbar\frac{\partial\psi}{\partial t}. \tag{4.25}$$

Next we assume the following scaling: $T = (\alpha/\hbar)t$ and $x = \beta X$. The chain rule applied to those scalings yield

$$\frac{\partial\psi}{\partial t} = \frac{\alpha}{\hbar}\frac{\partial\psi}{\partial T} \tag{4.26}$$

and

$$\frac{\partial^2\psi}{\partial x^2} = \frac{1}{\beta^2}\frac{\partial\psi}{\partial X^2}. \tag{4.27}$$

Substituting these into the time-dependent Schrödinger equation, equation 4.25 yields the scale-free Schrödinger equation

$$-\frac{\partial^2\Psi}{\partial X^2} + \Theta\left(X\right)\Psi(X,T) = i\frac{\partial\Psi}{\partial T}. \tag{4.28}$$

Using the scaling equations with equation 4.28 and comparing with 4.25 we find that

$$\frac{\hbar^2}{2m\beta^2\alpha} = 1 \tag{4.29}$$

so that $\beta = \frac{\hbar}{\sqrt{2m\alpha}}$ and $\Theta = V/\alpha$. This leads finally to the three scaling relationships:

$$\begin{aligned} x &= \left(\frac{\hbar}{\sqrt{2m\alpha}}\right)X \\ t &= \left(\frac{\hbar}{\alpha}\right)T \\ V &= \alpha\Theta \end{aligned}$$

A concrete example might help. Let's consider a particle of mass $1u$ and that we scale the potential energy such that $\alpha = 1$ so that $\Theta_{max} = 10$ corresponds to $V_{max} = 10$ meV. Let's further consider that our *box* in which we are solving the Schrödinger equation using `TDSE_SOLVER` is defined over $-30 \leq X \leq 30$. Noting that $\hbar c = 1973 \times 10^3meV\cdot$ Å we find that $\beta = 1.45$Å per X unit. Therefore our box is really defined over $-43.5 \leq x(\text{Å}) \leq 43.5$. The time scale is determined by $\hbar/\alpha$ which, using $\hbar = 0.6528$ meV$\cdot$ps, implies that a change in one unit of T corresponds to a change of 0.6528 ps.

* **Ex. 19** — Discuss the physical units of energy in terms of the scaling of the Schrödinger equation.

4.2 Scattering from a High Step Potential

Our first application of the GSS algorithm is to investigate the scattering of a propagating Gaussian wavepacket from a high step barrier. We define our computation box over $-30 \leq X \leq 30$ for 1000 equally-spaced locations and the time over $0 \leq T \leq 5$ for 1000 equally-spaced times. The incident wavefunction is a Gaussian wavepacket of the functional form

$$\langle x|\Psi\rangle\left(t\right) = Ce^{-\frac{1}{2}\left(\frac{x-x_0}{\sigma}\right)^2}e^{ik_0x}, \tag{4.30}$$

where C is a normalization constant, k_0 is the propagation constant, and x_0 is the center of the wavepacket at $t = 0$. There are a few points worth mentioning here. First, `TDSE_SOLVER` performs the normalization of the incident wavefunction so we can set $C = 1$ in our computation without a problem. Also, the propagation factor, e^{ik_0x}, causes a right-moving wavepacket whereas e^{-ik_0x} would be a left-moving wavepacket. In our calculation we make the following assignments: $x_0 = -15$, $k_0 = 3$,and $\sigma = 2$. The code that computes the time-evolution of this wavepacket and plots the animation is shown below in `TDSE_SOLVER_EX1`. Six frames from the animated sequence are shown in figure 4.1. We note, as we did in chapter 2, that the magnitude of the amplitude of the probability density in the animation and in the six-frame sequence is meaningless compared with the magnitude of the potential. We have simply scaled the probability density so that it appears on the same scale as the potential. This is all done in the routine `ANIMATE_PROBABILITY`.

```
pro tdse_solver_ex1
; Visualization of a Gaussian wavepacket colliding
; with a high step barrier.
; Define the space-time grid
nx = 1000 & x = linspace(-30.0,10.0,nx)
nt = 700 & t = linspace(0.0,5.0,nt)
dx = x[1] - x[0]
i = complex(0.0,1.0)  ; definition of the imaginary unit
; Note that we won't worry about normalizing this wavefunction
; because TDSE_SOLVER will take care of that for us.
ko = 3.0
wf_init = exp(-0.5*((x+15.0)/2.0)^2)*exp(i*ko*x)
; Define the potential as a barrier that is 20 units
; tall and starts at x=0.
v = fltarr(nx)
indhi = where(x gt 0) & v[indhi] = 20.0
ok = tdse_check_params(x,t,wf_init,v)
if ~ok then return
psi = tdse_solver(x,t,wf_init,v)
ret = animate_probability(x,t,v,psi,offset = tdse_calc_energy(x,wf_init))
end
```

The classical analog of this quantum scattering event is a ball bouncing off of a hard wall. The quantum effects of this collision are apparent in two striking ways. First the wavepacket spreads before it starts interacting with the step. This can be seen in the two frames before the collision. This is also clear in the structure of the wavepacket after the collision seen in the final two frames. The other quantum effect is the interference of the wavepacket with itself during the interaction with the wall. The oscillations are due to the superposition of the right-moving and left-moving components of the wavepacket interfering with each other during the interaction.

Another instructive way to view this collision is by representing the wavepacket by phasors, as described in chapter 1. Phasors, arrows whose magnitude corresponds to the magnitude of a complex quantity and whose direction depends on the magnitude of the real and imaginary components of that complex number, are an alternative representation of the dynamics of a quantum scattering event and can emphasize different aspects of the collision process. In order to see how a wavepacket looks in its phasor form, change the line of code in `TDSE_SOLVER_EX1` from `ret = animate_probability(x,t,v,psi)` to `ret = animate_phasors(x,t,v,psi)`. If the animation runs too quickly, you can set a keyword `wait` to a small value in the `ANIMATE_PHASORS` function which will slow it down. For example, try `ret = animate_phasors(x,t,v,psi,wait=0.02)`. Note that not all of the spatial points in the grid for `x` have a phasor associated with it. The reason for not including all of the phasors in the animated display is that it would be too cluttered. Only every third phasor is drawn. In addition, only phasors whose magnitude exceeds a certain value are drawn. So as the wavepacket propagates, you will see the trail of phasors track the motion of the center of the wavepacket. Figure 4.2 shows various aspects of the animated sequence.

When a continuous wave is represented by phasors, the effects of constructive interference are striking. For a wavepacket colliding with a high step barrier, there is a time period over which the constructive interference is apparent via adjacent phasors that are nearly parallel because they have nearly the same phase. This is evident in figure 4.2 which shows more time steps than figure

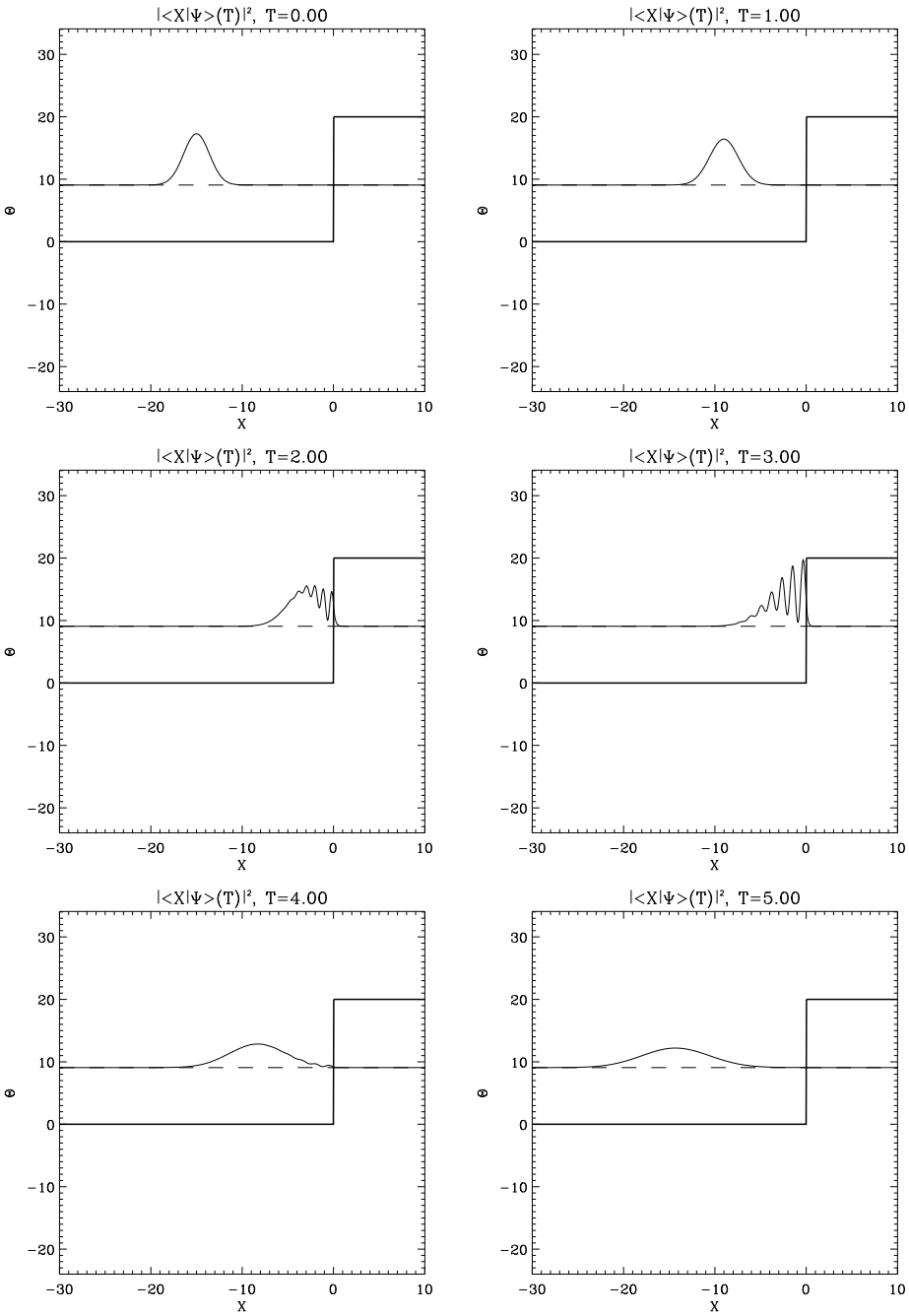


Figure 4.1: Scattering of a Gaussian wavepacket from a high step. The parameters used in this computation were $J = 1000$, $N = 700$, $x_0 = -15$, $k_0 = 3$, $\sigma = 2$, $-30 \leq X \leq 10$, and $0 \leq T \leq 5$. The probability density has been offset vertically by its kinetic energy.

4.1 for the time in which the wavepacket is "in contact" with the wall.

Another interesting way to view the collision of a wavepacket with a step barrier is in momentum space. Recall that the momentum of a particle is related to the wavevector of that particle through the relationship $p = \hbar k$. The probability density in wavevector space is given by $|\phi(k,t)|^2$ where $\phi(k,t)$ is related to the position-space wavefunction $\psi(x,t)$ through a Fourier transformation in space. The time-evolution of the probability density in k-space is shown in figure 4.3. Most aspects that you see in the momentum distribution should make qualitative sense. For instance, the incident wavepacket initially has a momentum distribution centered about $k = k_0 = 3$. During the collision (i.e. while the wavepacket is in contact with the barrier), a peak begins to build up centered about $k = -3$ while the intensity at $k = 3$ starts to diminish. Finally, after the collision is complete, only a peak centered about $k = -3$ remains.

Ex. 20 — If the step barrier height with a value of 20 corresponds to 20 meV for the computation displayed in figure 4.1 and the particle has mass of 1u, determine the correct physical units for the simulation. Specifically, how big is the computational box and at what time does the calculation end?

Ex. 21 — Create an animated sequence for a collision of the same wavepacket with a barrier that is 5 units high. What are the qualitative differences you observe with the collision shown in figure 4.1.

Ex. 22 — Create an animated sequence for a collision of a Lorentzian wavepacket (with propagation constant $k_0 = 3$) with a barrier that is 20 units high. Note that the Lorentzian wavepacket takes the functional form

$$\langle x|\psi\rangle = C\frac{e^{ik_0x}}{(x-x_0)^2+\Gamma^2} \tag{4.31}$$

where C is a normalization factor, Γ is the half-width at half-maximum, x_0 is the center of the wavepacket, and k_0 is the initial wavevector. What are any qualitative differences you see between a propagating Gaussian wavepacket and a propagating Lorentzian wavepacket interacting with the step barrier?

4.3 Scattering from a Square Barrier

If a wavepacket collides with a square barrier, it is possible that part of the wavepacket will penetrate into the barrier and even be found on the far side of the barrier. The conditions for this penetration, or tunneling through the barrier, depend on the energy of the incident wavepacket, the height of the barrier, and the width of the barrier.

Let's first consider an incident wavepacket with parameters identical to those in the previous section. That is, propagating with a propagation constant $k_0 = 3$ and with a Gaussian standard deviation $\sigma = 2$. For a potential barrier with height $\Theta_{max} = 10$ and a width of 3, the results of the scattering event are computed with `TDSE_SOLVER_EX2` and displayed in figure 4.4. From this figure we see a number of features that are indicative of the quantum nature of the process. First during the collision process with the left-most wall, there are numerous oscillations due to the interference of the wavefunction's incident and reflected components. Moreover there is clearly a component that is transmitted through the entire barrier. Perhaps one of the most interesting aspects of the

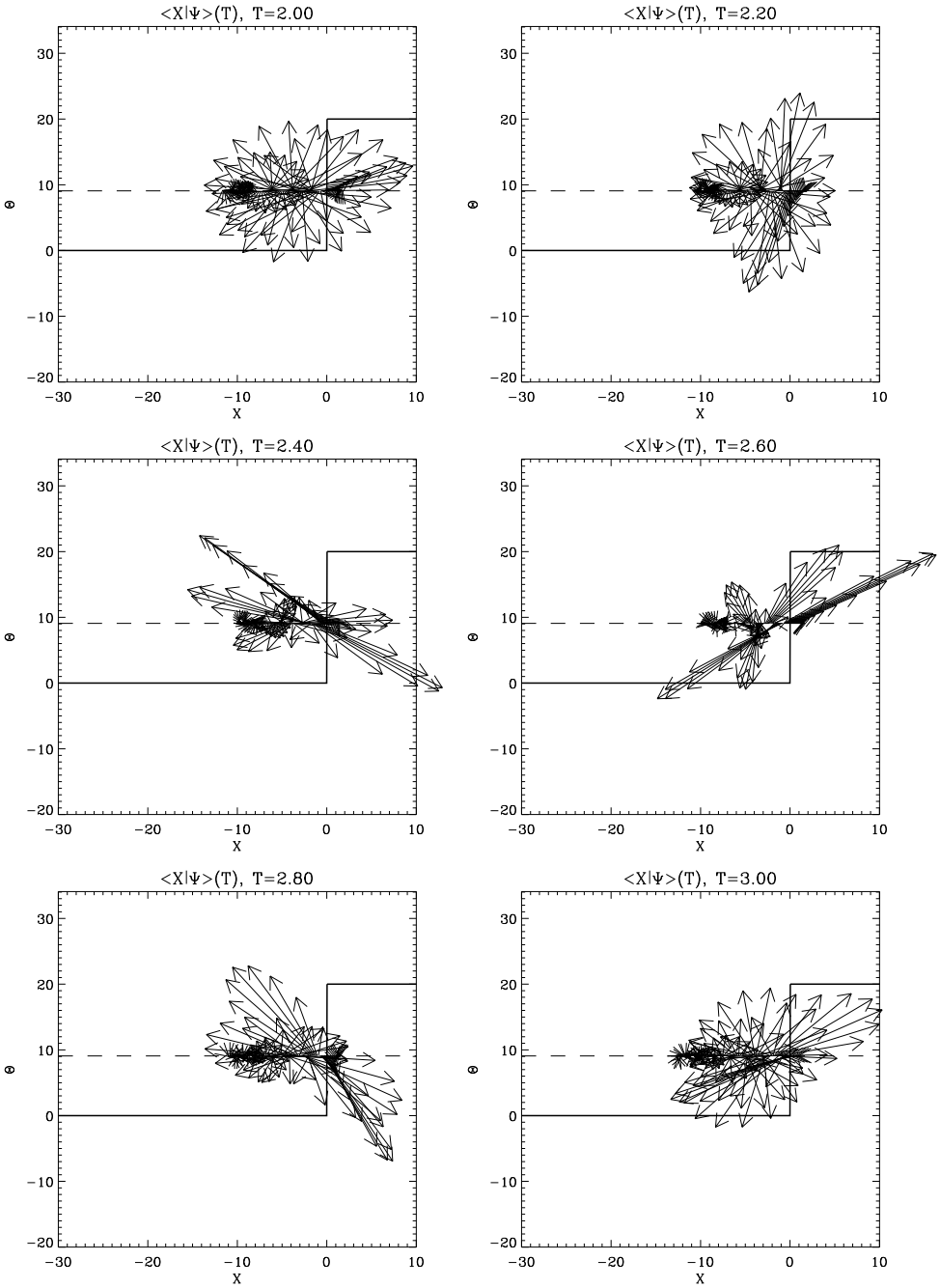


Figure 4.2: Same as figure 4.1 but wavefunction represented as a phasor and only shown over the range $2 \leq T \leq 3$ to emphasize the period of time in which the wavepacket interacts with the wall. Note that the phasors are moving largely in-phase during these frames. This manifests itself as the many phasors being parallel and moving synchronously which is better seen in the animation. The phasors have been offset vertically by the kinetic energy of the wavepacket.

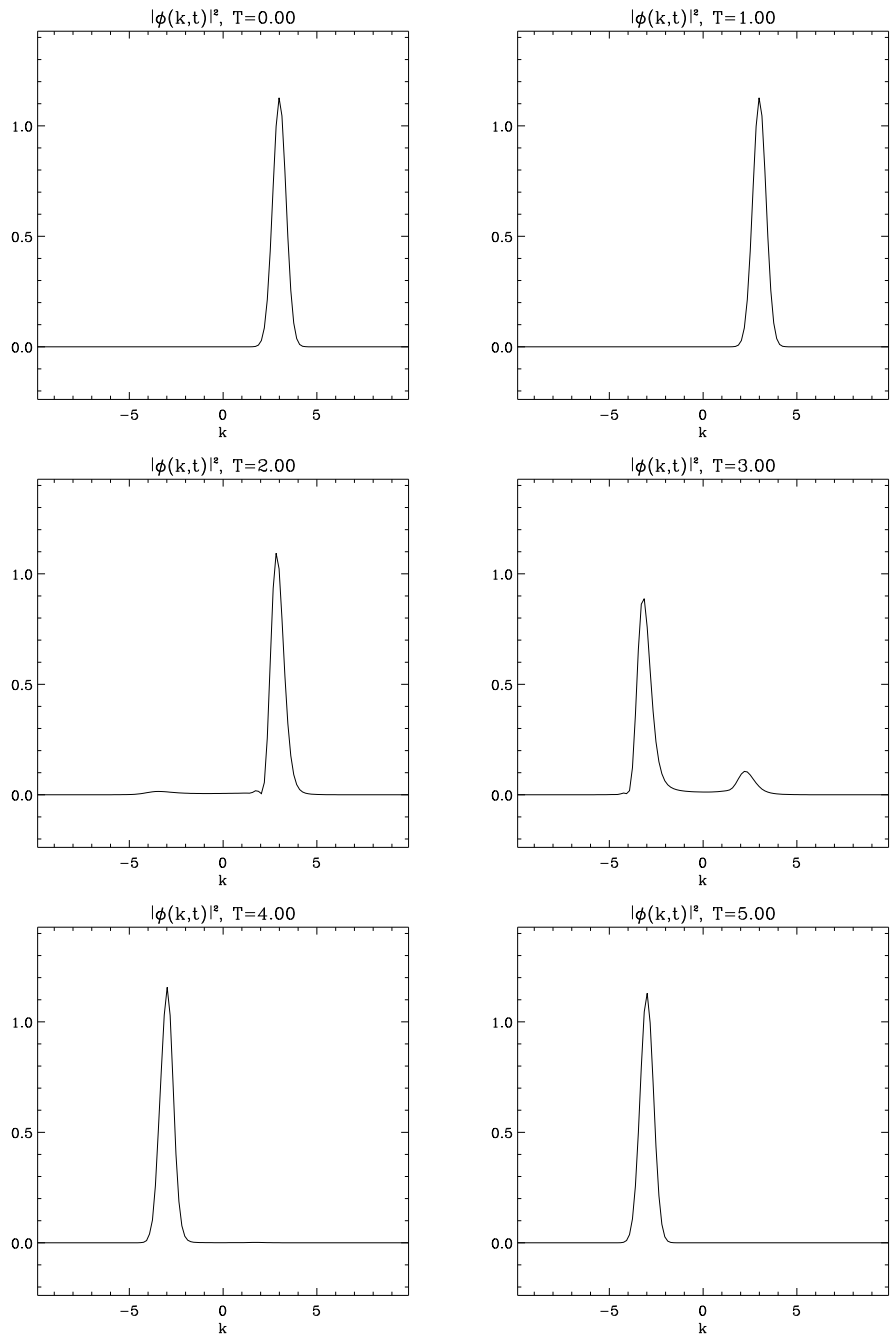


Figure 4.3: Same as figure 4.1 but for the probability density in momentum space.

scattering event is that there is a component that appears to be *trapped* within the barrier itself. This is a longer-lived component that still has a substantial amplitude even in the last frame of the animated sequence. This *trapped* component is a result of waves reflecting back and forth within the walls of the barrier that sum constructively. Eventually this component will die out but it is a resonant or metastable excitation of the barrier.

```
pro tdse_solver_ex2
; Visualization of a Gaussian wavepacket colliding
; with a low square barrier.
; Define the space-time grid
nx = 1000 & x = linspace(-30.0,30.0,nx)
nt = 700 & t = linspace(0.0,3.0,nt)
i = complex(0.0,1.0)  ; definition of the imaginary unit
; Note that we won't worry about normalizing this wavefunction
; because TDSE_SOLVER will take care of that for us.
wf_init = exp(-0.5*((x+10.0)/2.0)^2)*exp(i*3.*x)
v = fltarr(nx)
indhi = where((x gt -1.5) and (x le 1.5)) & v[indhi] = 10.0
ok = tdse_check_params(x,t,wf_init,v)
if ~ok then return
psi = tdse_solver(x,t,wf_init,v)
ret = animate_probability(x,t,v,psi,offset = tdse_calc_energy(x,wf_init))
end
```

** **Ex. 23** — Compute the transmission of the wavepacket on a square barrier as a function of barrier width for fixed height of $\Theta_{max} = 10$. Plot the transmission as a function of barrier width Γ for widths in the following interval: $0.5 \leq \Gamma \leq 5.0$.

** **Ex. 24** — Compute the transmission of the wavepacket on a square barrier as a function of barrier height for fixed width of $\Gamma = 3.0$. Plot the transmission as a function of barrier height Θ_{max} for heights in the following interval: $5.0 \leq \Theta_{max} \leq 30.0$.

4.4 Scattering from a Square Well

We saw in the previous section that it is possible for a wavepacket to penetrate through a barrier if the barrier is thin enough and/or short enough. In that example our intuition from classical mechanics would lead us to expect that there would only be a reflected component. Nevertheless we found that the wavepacket was also capable of penetrating through the barrier. Now we will consider a wavepacket scattering from a square well. In this case our classical intuition would lead us to expect that the wavepacket will transmit entirely across the well. Once again our classical intuition fails us.

The program that illustrates this scattering event is listed in `TDSE_SOLVER_EX3` below.

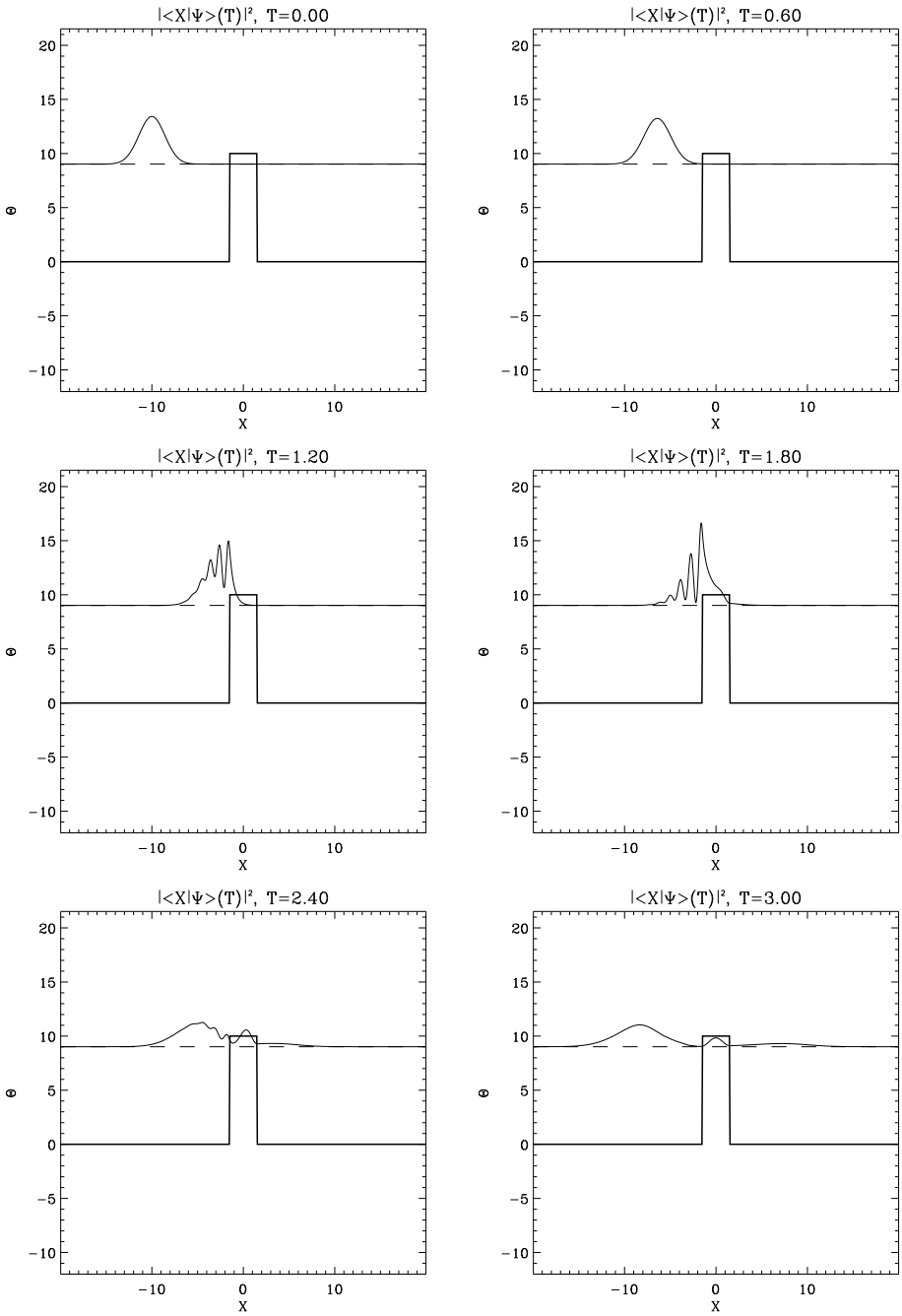


Figure 4.4: Scattering of a Gaussian wavepacket from a square barrier. The parameters used in this computation were $J = 1000$, $N = 700$, $x_0 = -10$, $k_0 = 3$, $\sigma = 2$, $-30 \leq X \leq 30$, and $0 \leq T \leq 3$. The probability density has been offset vertically by its kinetic energy.

```
pro tdse_solver_ex3
; Visualization of a Gaussian wavepacket colliding
; with square well.
; Define the space-time grid
nx = 1000 & x = linspace(-30.0,30.0,nx)
nt = 1000 & t = linspace(0.0,3.0,nt)
i = complex(0.0,1.0)
wf_init = exp(-0.5*((x+10.0)/2.0)^2)*exp(i*3.*x)
v = fltarr(nx)
w = 3.0
indhi = where((x gt -0.5*w) and (x le 0.5*w)) & v[indhi] = -50.0
ok = tdse_check_params(x,t,wf_init,v)
if ~ok then return
psi = tdse_solver(x,t,wf_init,v)
xr = [-20.0,20.0]
xind = where((x ge xr[0]) and (x le xr[1]))
ret = animate_probability(x[xind],t,v[xind],psi[xind,*],$
  offset = tdse_calc_energy(x,wf_init))
end
```

We can see from figure 4.5 that the wavepacket interacts with the square well in a complicated way. For instance, as the wavepacket hits the leading edge of the well, oscillations build up as a result of the interference of reflected wave components and the incident wave. Still, part of the wave moves into the region of negative potential where oscillations build up. A standing wave pattern builds up for a short time within the well as a result of right-moving and left-moving wave components interfering with each other. These left-moving and right-moving waves are the result of reflection from the edges of the well. Finally we can see that once the excitation within the well has decayed away, the only remaining components are a right-moving transmitted wavepacket and a left-moving reflected wavepacket. Both wavepackets have nearly the same amplitude and width.

The scattering of a wavepacket from a square well is also interesting to view in momentum space. A sequence of frames from the scattering event is shown in figure 4.6. Note that there is a great deal more structure in $|\phi(k,t)|^2$ during the excitation of the mode in the well as compared to the interaction of a wavepacket with a high step barrier. One specific aspect to observe is found in the two panels corresponding to T=0.8 and T=1.2 in figures 4.5 and 4.6. The standing wave of high-frequency oscillations in the well region seen in the two frames from the position-space plots manifest themselves in the momentum-space plots at the same time as two satellite peaks centered about $k_0 \simeq \pm 7.5$. We should expect there to be symmetric peaks in momentum space because it is a standing wave phenomenon that is constructed of a left-moving and right-moving plane wave pair.

** **Ex. 25** — Compute the transmission of the wavepacket on a square well as a function of barrier width for fixed depth of $\Theta_{min} = -50$. Plot the transmission as a function of barrier width Γ for widths in the following interval: $0.5 \leq \Gamma \leq 5.0$.

** **Ex. 26** — Compute the transmission of the wavepacket on a square barrier as a function of barrier depth for fixed width of $\Gamma = 3.0$. Plot the transmission as a function of barrier height Θ_{min} for heights in the following interval: $-50.0 \leq \Theta_{min} \leq -5.0$.

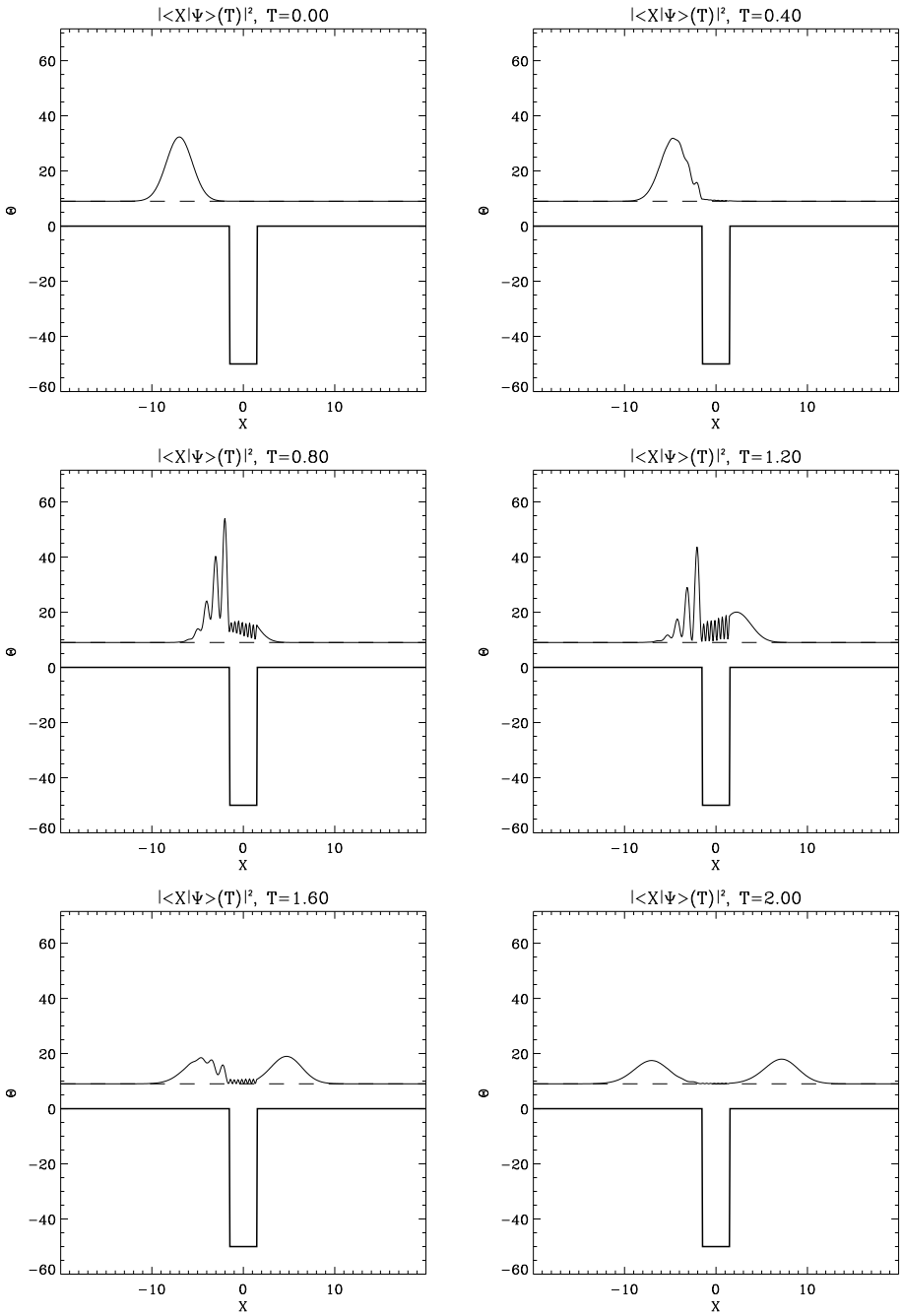


Figure 4.5: Scattering of a Gaussian wavepacket from a square well. The parameters used in this computation were $J = 1000$, $N = 700$, $x_0 = -7$, $k_0 = 3$, $\sigma = 2$, $-30 \leq X \leq 30$, and $0 \leq T \leq 2$. The probability density has been offset vertically by its kinetic energy.

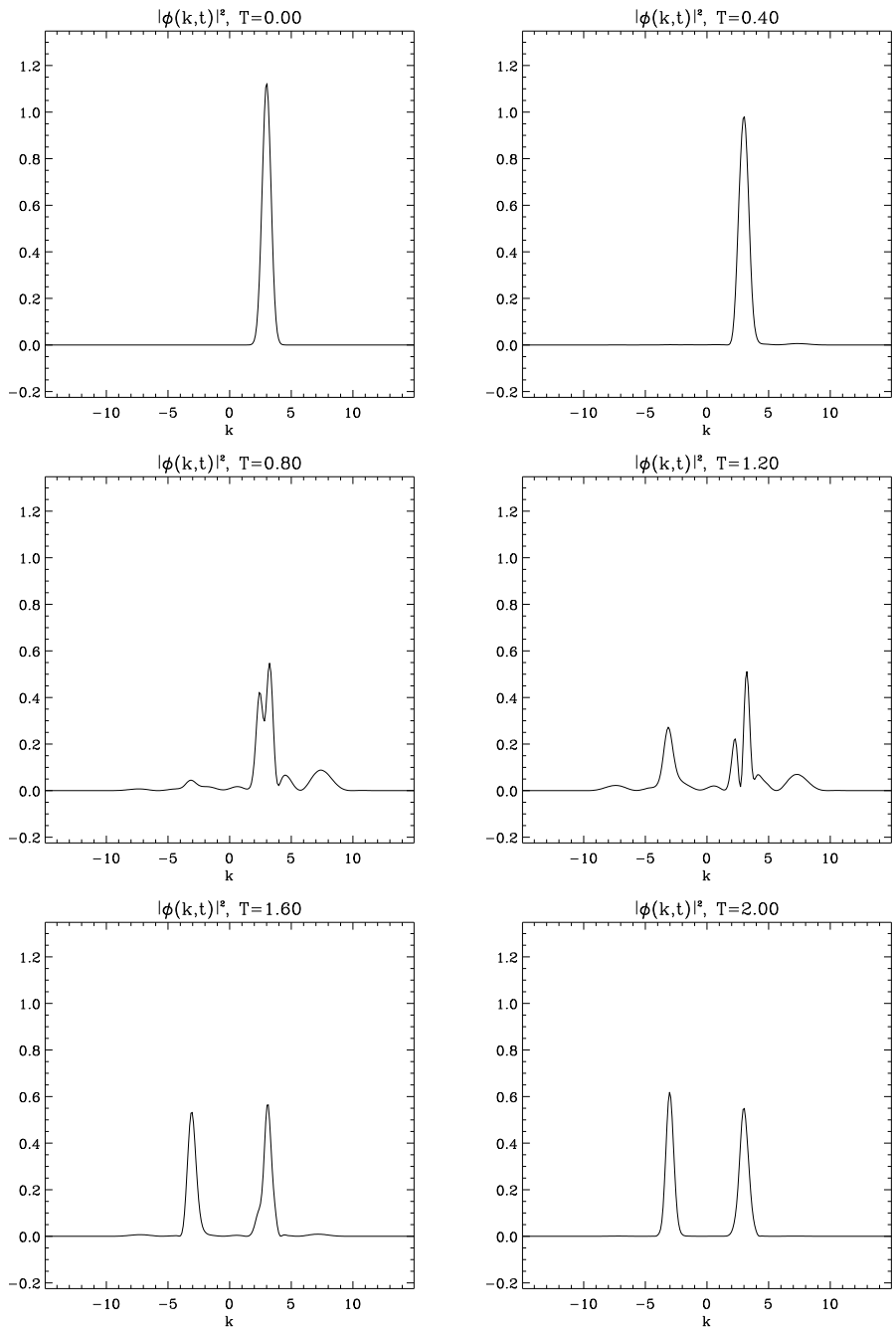


Figure 4.6: Same as figure 4.5 but for the probability density in momentum space.

4.5 Scattering from a Periodic Potential: Bragg's Law

One of the basic phenomena exploited in the field of crystallography, the study of the atomic-scale structure of materials, is diffraction. Beams of neutrons and x-rays are used to scatter from solids and the wave-like behavior of these particles results in interference intensity patterns in the detectors. These interference patterns arise from Bragg's law which is well-known to crystallographers. The basic phenomenon is that there are certain wavelengths λ of particles in the incident beam (and wavevectors k since $k = 2\pi/\lambda$) for which scattering from a periodic structure results in total reflection at the surface while other wavelengths result in partial transmission into the structure. We can model a one-dimensional "crystal" structure by the one-dimensional potential

$$V(x) = V_0 \sin(Qx) \tag{4.32}$$

where Q is called the reciprocal lattice vector for the "crystal". We can model our incident particle beam by a Gaussian wavepacket with incident wavevector, k_0. The result is worked out in numerous textbooks (see [16] for instance) but the result, the Bragg condition for total reflection by a one-dimensional "crystal", is given by

$$k_0 = \frac{Q}{2}. \tag{4.33}$$

Bragg's law can be illustrated through a simple animation. The code `TDSE_SOLVER_EX4` is the implementation of this scattering event and the sequence is shown in figure 4.7. The case where there is partial transmission is shown in figure 4.8. It is particularly interesting to see from figure 4.7 that the wavepacket appears to penetrate almost 5 Å into the "crystal" before reflecting out of it.

```
pro tdse_solver_ex4
; Visualization of a Gaussian wavepacket scattering
; from a periodic potential.
; Define the space-time grid
nx = 1000 & x = linspace(-20.0,20.0,nx)
nt = 1000 & t = linspace(0.0,4.5,nt)
i = complex(0.0,1.0)
k = 2.0
wf_init = exp(-0.5*((x+5.0)/3.0)^2)*exp(i*k*x)
energy = tdse_calc_energy(x,wf_init)
Q = 4.0 & vo = 4.2 & ncycles = 15.0 & L = 2.0*!pi*ncycles/Q
ok = where((x ge 0) and (x le L))
v = fltarr(nx) & v[ok] = vo*sin(Q*x[ok])
ok = tdse_check_params(x,t,wf_init,v)
if ~ok then return
psi = tdse_solver(x,t,wf_init,v)
ret = animate_probability(x,t,v,psi,offset = energy)
end
```

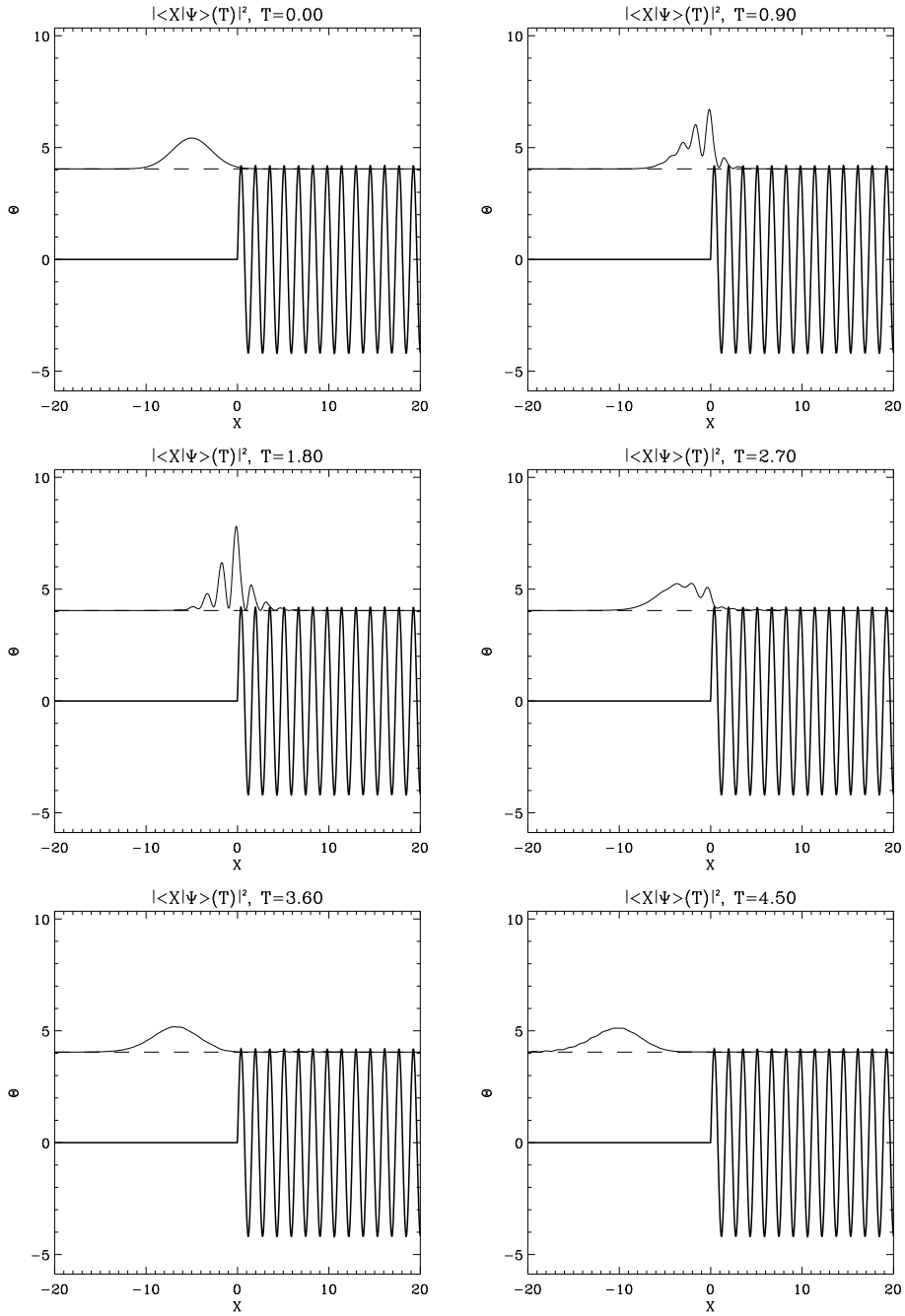


Figure 4.7: Scattering of a Gaussian wavepacket from a periodic potential $V(x) = 4.2\ \sin(4x)$. The parameters used in this computation were $J = 1000$, $N = 1000$, $x_0 = -5$, $k_0 = 2$, $\sigma = 3$, $-20 \leq X \leq 20$, and $0 \leq T \leq 4.5$. The probability density has been offset vertically by its kinetic energy.

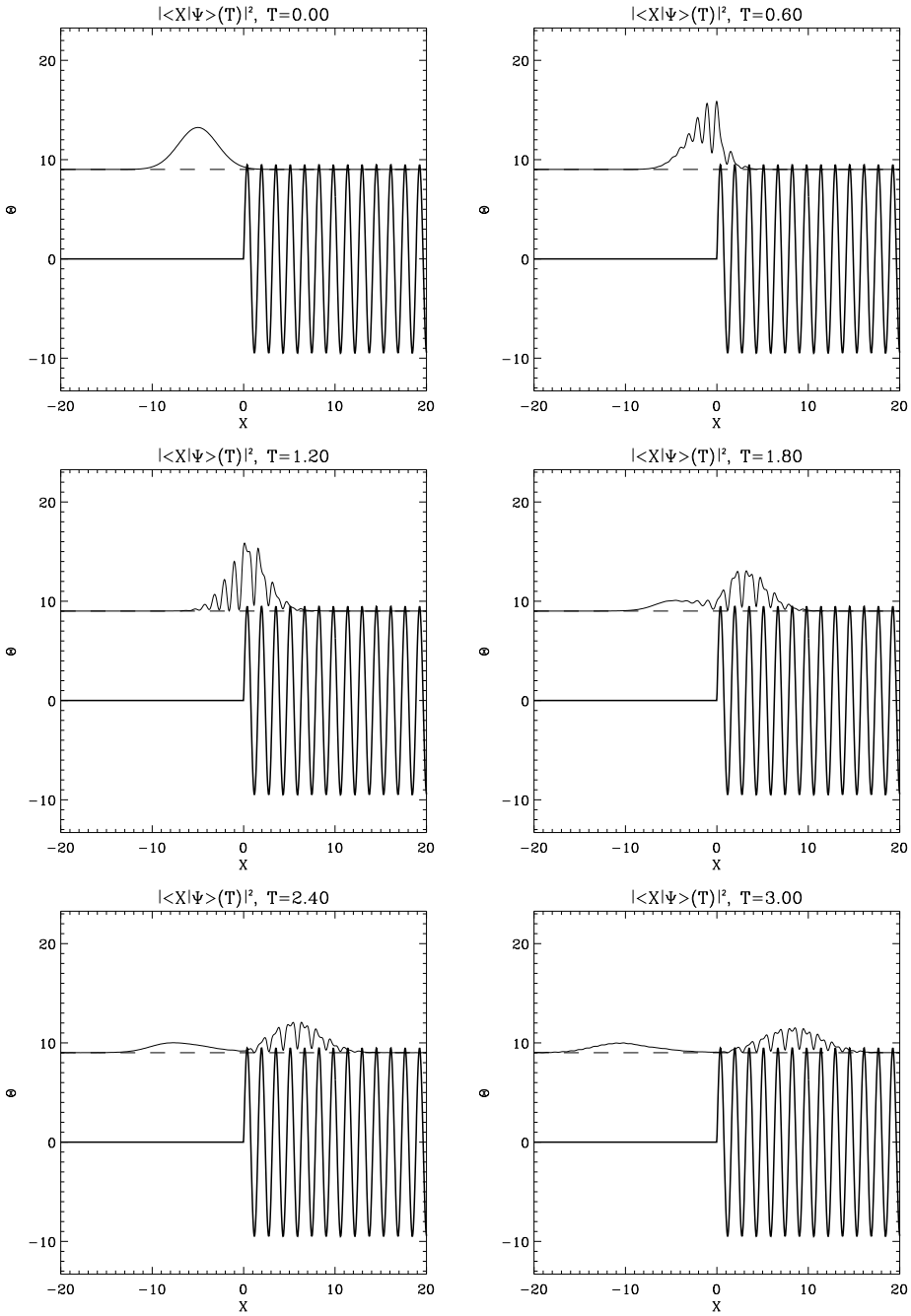


Figure 4.8: Scattering of a Gaussian wavepacket from a periodic potential $V(x) = 9.5\ \sin(4x)$. The parameters used in this computation were $J = 1000$, $N = 1000$, $x_0 = -5$, $k_0 = 3$, $\sigma = 3$, $-20 \leq X \leq 20$, and $0 \leq T \leq 3$. The probability density has been offset vertically by its kinetic energy.

Ex. 27 — You observed that the wavepacket penetrates a small distance into the one-dimensional "crystal" even when the Bragg condition for reflection is satisfied. How does this penetration depth vary with the width of the wavepacket? (A qualitative answer is sufficient.)

4.6 Scattering from a Well with a Lip: Metastable/Virtual States

When a wavepacket scatters from a square well, for certain values of the incident wavevector, a short-lived bound state of the well can be excited. This is known as a metastable or virtual state. If instead the well is surrounded by a thin barrier, then it is possible for an even longer lived state to be excited within the well region. There are specific conditions that must be met related to the wavelength content in the wavepacket and the width of the potential. If a "lip" exists on the well–as the barrier–and the kinetic energy of the wavepacket is just a bit smaller than the height of the lip, then a standing wave in the well can occur. A wavepacket can tunnel through the lip and part of it can get trapped in the well—bouncing between the insides of the lip causes a buildup of reflected waves and a long lived standing wave. In order to see this, consider a potential well with a half Gaussian barrier on either side of the well. This can be represented by the following function:

$$V(x) = \begin{cases} H\exp\left(-\frac{1}{2}\left(\frac{x+W/2}{r}\right)^2\right), & x \leq -W/2 \\ -V_0, & -W/2 \leq x \leq W/2 \\ H\exp\left(-\frac{1}{2}\left(\frac{x-W/2}{r}\right)^2\right), & x \geq W/2 \end{cases} \tag{4.34}$$

where H is the amplitude of the Gaussian barrier, V_0 is the depth of the well, W is the width of the well, and r is the thickness (standard deviation) of the Gaussian barrier. In order to obtain a long-lived state, we need to try to set up a standing wave within the well. To do this we can, as a first approximation, match an integer multiple of the wavepacket's propagation wavelength, $N\lambda$, to the well width, W, so that an integer number of standing waves fit into the well. In other words, we excite an eigenstate of the well. Since the well has a width W, the eigenstate's wavevector must satisfy $kW = 2N\pi$. This is the resonant matching condition. If we select $k = 3$ and $N = 3$, we obtain $W = 2\pi$. These are the values we use in the computations described below.

The result of sending a Gaussian wavepacket at the well is shown in figure 4.10. As shown in that sequence, the wavefunction in the well region starts to build up in intensity when the wavepacket collides with the left side of the well. Part of the wavepacket leaks through the lip/barrier and a standing wave begins to build up from the right-moving and reflected waves within the deep well region. Since the well is so deep, the standing wave has a large kinetic energy, evident from the numerous nodes in the well region. Note also that the wavepacket leaks out the far side of the well so that for later times the wavepacket has a large spatial extent.

We can focus on the portion of the wavepacket that is trapped in the well region by zooming into that region, as shown in figure 4.11. Note that the wavefunction builds up quickly but the amplitude decays away very slowly. We can observe this decay by integrating the probability density over the well region only and plotting its time-dependence. That is, we calculate the following:

$$a(T) = \int_{-W/2}^{W/2} \mathrm{dX} |\Psi(X,T)|^2. \tag{4.35}$$

For the situation shown in figures 4.10 and 4.11 the result is plotted in figure 4.12. Clearly there

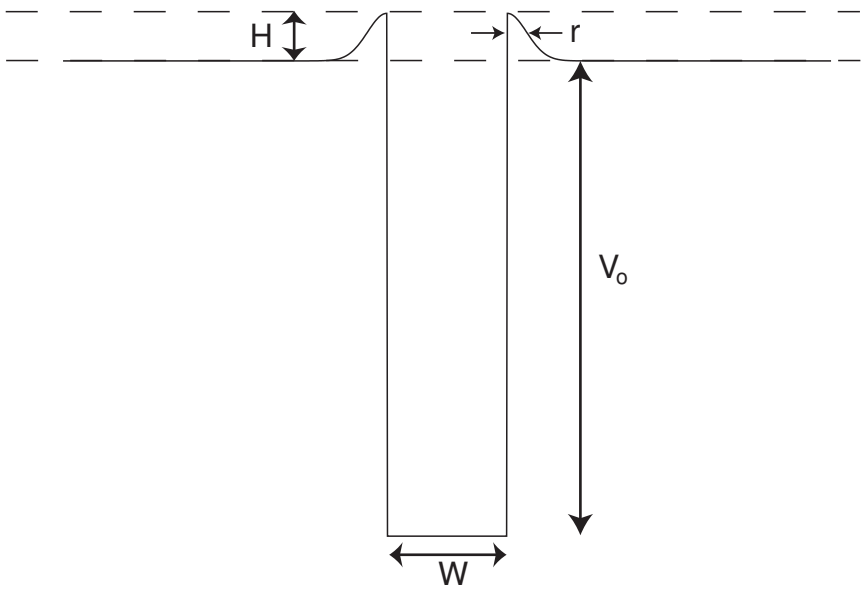


Figure 4.9: Well with a half Gaussian barrier on both sides corresponding to equation 4.34.

is rapid increase in probability density but then it decreases quite slowly. The time-dependence of this reduction is reminiscent of an exponential decay. The code that shows the wavepacket collision is listed in `TDSE_SOLVER_EX5`.

```
pro tdse_solver_ex5
; Visualization of a Gaussian wavepacket scattering
; from a deep well with a lip, illustrating a virtual or
; metastable state.
; Define the space-time grid
xlo = -30.0 & xhi = 30.0
nx = 1000 & x = linspace(xlo,xhi,nx)
nt = 1500 & t = linspace(0.0,4.5,nt)
i = complex(0.0,1.0) & k = 3.0
wf_init = exp(-0.5*((x+13.0)/3.0)^2)*exp(i*k*x)
energy = tdse_calc_energy(x,wf_init)
v = fltarr(nx)
vo = -100.0  ; well depth
w = 6.*!pi/k    ; width of the well
h = 10.0    ; height of the well
r = 1.0     ; decay of well lip
; Define region I
xo = 0.5*w
regI = where(x lt -xo)
regII = where((x ge -xo) and (x le xo))
regIII = where(x gt xo)
v[regII] = vo
```

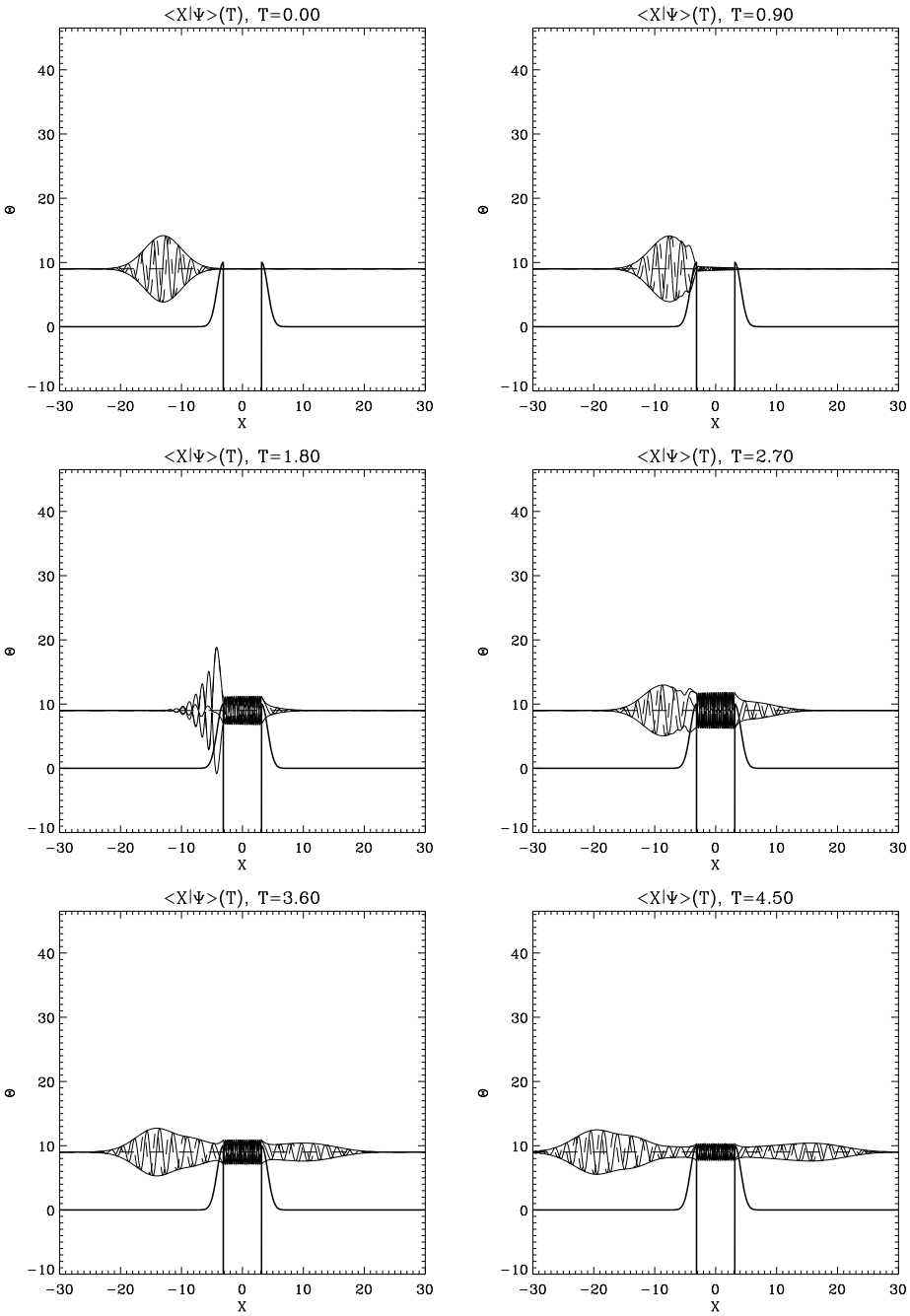


Figure 4.10: Scattering of a Gaussian wavepacket from a potential well with a Gaussian lip. The parameters used in this computation were $J = 1000$, $N = 1500$, $x_0 = -13$, $k_0 = 3$, $\sigma = 3$, $-30 \leq X \leq 30$, and $0 \leq T \leq 4.5$. The well has the parameters $H = 10$, $r = 1$, and $W = 2\pi$. The probability density has been offset vertically by its kinetic energy.

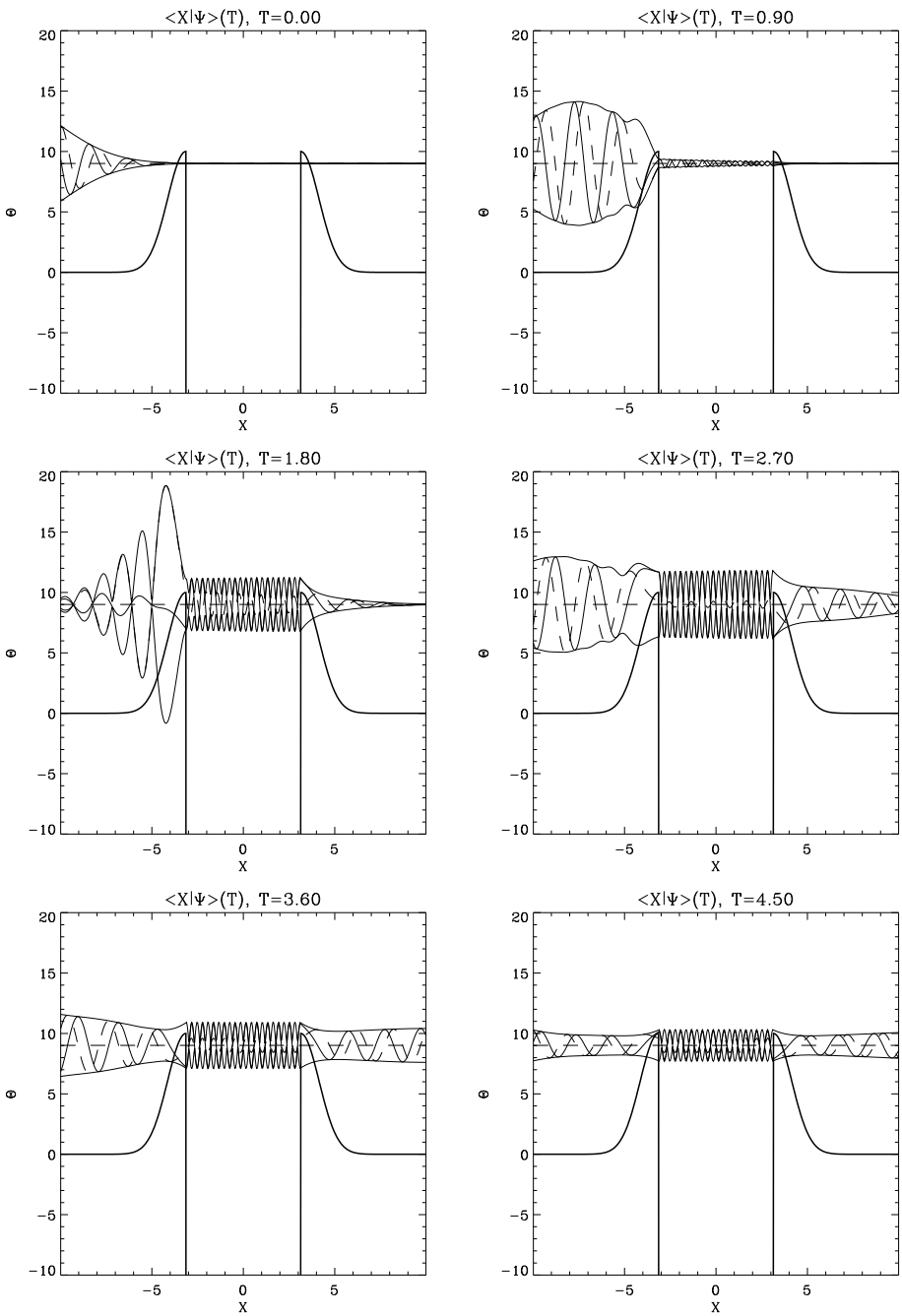


Figure 4.11: Same as 4.10 but with the well region expanded to see the standing wave detail.

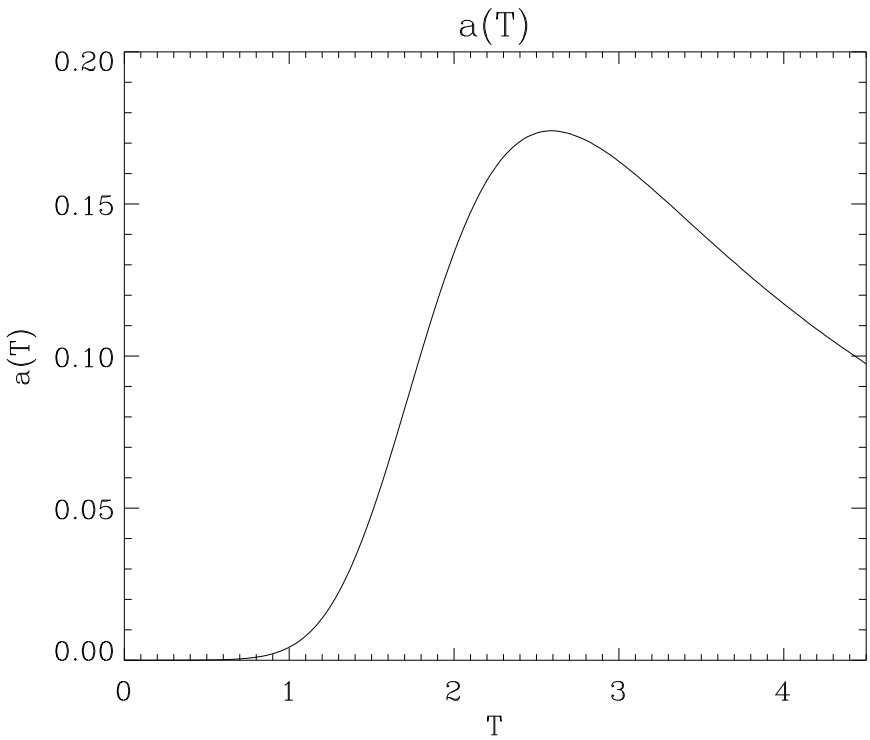


Figure 4.12: Time-dependence of the probability density integrated over the well region. Note the decay after the standing wave has achieved its maximum amplitude in the well.

```
v[regI] = h*exp(-0.5*((x[regI]+xo)/r)^2)
v[regIII] = h*exp(-0.5*((x[regIII]-xo)/r)^2)

ok = tdse_check_params(x,t,wf_init,v)
if ~ok then return
psi = tdse_solver(x,t,wf_init,v)
xr = [-5.,5.0] & ind = where((x ge xr[0]) and (x le xr[1]))
ret = animate_wavefunction(x[ind],t,v[ind],psi[ind,*],offset = energy)

; Now integrate the probability density over the "trapped" region
; and display the integrated probability as a function of time.
intensity = fltarr(nt) & prob = abs(psi)^2
xlim = x[regII,*]
problim = prob[regII,*]
for j = 0,nt-1 do intensity[j] = int_tabulated(xlim,problim[*,j])

window,/free
plot,t,intensity,background = fsc_color('white'),color = fsc_color('black'),$
    xtitle = '!6T',ytitle = '!6a(T)',title = '!6a(T)',xrange = [0,max(t)],/xstyle
end
```

* **Ex. 28** — For the example just discussed, construct and observe the animated sequence for the time-dependent momentum probability distribution, $|\phi(k,T)|^2$. Identify the feature(s) in momentum space corresponding to the standing wave in the well.

** **Ex. 29** — Create an animated sequence that illustrates the phenomenon of radioactive nuclear decay. As a simple model of a nucleus undergoing α-decay, consider a Gaussian wavepacket confined initially to a potential well, $V(x)$, and traveling to the right with $k = 3$. A simple potential that models this phenomenon is given by

$$V(x) = \begin{cases} \infty, & x < x_1 \\ -V_0, & x_1 \le x \le x_2 \\ H\exp\left(-\frac{1}{2}\left(\frac{x-x_2}{r}\right)^2\right), & x > x_2 \end{cases} \tag{4.36}$$

In this model your Gaussian wavepacket should initially be completely confined within $x_1 \le x \le x_2$. In addition to observing the animated sequence and watching the wavepacket leak out of the well, determine the decay curve, $a(T)$ as in figure 4.12.

Chapter 5

Quantum Dynamics in Two Dimensions

Though the animations of quantum scattering in one dimension are striking, seeing how a two-dimensional wavepacket scatters from even the simplest of potentials illustrates some of the most profound and visually arresting results from quantum mechanics. Perhaps one of the most intriguing results is that of *a particle interfering with itself* which becomes accessible in two dimensions in a manner arguably more impressive than in one dimension.

Another application of the two-dimensional Schrödinger equation is for two one-dimensional systems of interacting particles. A number of intriguing possibilities surface when we are able to show the dynamic evolution of two quantum particles whose motion is not only determined by their own independent potentials, but also an interaction term. We will illustrate a few cases of particles interacting, including two "free" particles (i.e. wavepackets with no independent potentials) colliding. In addition we will demonstrate a wavepacket interacting with an harmonic oscillator through a hard-core repulsive interaction.

5.1 The Algorithm of Askar and Cakmak

We begin with the Schrödinger equation in two dimensions:

$$H = -\frac{\hbar^2}{2m}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\langle x, y|\psi\rangle + V(x, y)\langle x, y|\psi\rangle = E\langle x, y|\psi\rangle. \tag{5.1}$$

In our derivation of the GSS algorithm in chapter 5 we found that a straightforward application of finite differences in the time-domain resulted in an algorithm that was unstable and probability was not conserved. We used an approximation to the exponential time-development operator which resulted in an implicit algorithm. Askar and Cakmak came up with an algorithm that was both explicit *and* stable by also considering an approximation involving the time-development operator.[17] We will develop the one-dimensional algorithm of Askar and Cakmak first and then simply present the result of its extension to two dimensions.

Following the original development in Askar and Cakmak's paper we choose units in which $\hbar = m = 1$. Note the difference to the choice of the GSS algorithm ($\hbar = 2m = 1$). The approximation to the wavefunction in time and space is given by ψ_j^n where, similar to the development of

the GSS algorithm in the previous chapter, n is the time index and j is the index for the spatial coordinate. The Hamiltonian action on the discretized wavefunction is given by

$$H\psi_j^n = -\frac{1}{2\Delta_x^2}\left(\psi_{j+1}^n + \psi_{j-1}^n - 2\psi_j^n\right) + V_j\psi_j^n. \tag{5.2}$$

The time-development operator has the following action on the discretized wavefunction is

$$\psi_j^{n+1} = e^{-i\Delta t H}\psi_j^n \tag{5.3}$$

and

$$\psi_j^{n-1} = e^{i\Delta t H}\psi_j^n. \tag{5.4}$$

Subtracting equation 5.4 from equation 5.3 and taking a small argument expansion yields

$$\psi_j^{n+1} - \psi_j^{n-1} \simeq -2i\Delta t H\psi_j^n. \tag{5.5}$$

We can now substitute the action of the Hamiltonian on the discretized wavefunction into this equation, and solve for the wavefunction at time $n+1$:

$$\psi_j^{n+1} = \psi_j^{n-1} - 2i[(2\alpha_x + V_j\Delta t)\,\psi_j^n - \alpha_x\left(\psi_{j+1}^n + \psi_{j-1}^n\right)] \tag{5.6}$$

where $\alpha_x = \frac{1}{2}\Delta t/\Delta_x^2$.

This result is easily extended to two spatial dimensions with the final result, again an explicit method as in one dimension,

$$\psi_{j,k}^{n+1} = \psi_{j,k}^{n-1} - 2i[(2\,(\alpha_x + \alpha_y) + V_{j,k}\Delta t)\,\psi_{j,k}^n - \alpha_x\left(\psi_{j+1,k}^n + \psi_{j-1,k}^n\right) - \alpha_y\left(\psi_{j,k+1}^n + \psi_{j,k-1}^n\right)] \tag{5.7}$$

where $\alpha_x = \frac{1}{2}\Delta t/\Delta_x^2$ and $\alpha_y = \frac{1}{2}\Delta t/\Delta_y^2$.

Though this equation is explicit, it relies on knowledge of the wavefunction at *two* previous time steps. Since we assume that we only know the wavefunction at the initial time step, $n = 0$, we must determine the wavefunction at $n = 1$. The way in which we do this is that we apply the following approximation to the time-development operator:

$$\begin{aligned}\psi_{j,k}^n &= e^{-i\Delta t H}\psi_{j,k}^n \\ &\simeq (1 - i\Delta t H)\,\psi_{j,k}^n.\end{aligned}$$

Of course we know that this is an inherently unstable way to propagate the solution in time because errors will accumulate at each time step. However it is acceptable to do this for a single time step because there will not be a systematic accumulation of errors. Completing the algebraic steps yields the following approximation for the first time step

$$\psi_{j,k}^1 = \psi_{j,k}^0 - i\,(\Delta t V_{j,k} + 2\,(\alpha_x + \alpha_y))\,\psi_{j,k}^0 + i\alpha_x\left(\psi_{j+1,k}^0 + \psi_{j-1,k}^0\right) + i\alpha_y\left(\psi_{j,k+1}^0 + \psi_{j,k-1}^0\right). \tag{5.8}$$

From this point on it is straightforward to apply the explicit algorithm in equation 5.7 to calculate $\psi_{j,k}^n$ for all time steps from $n = 2$ forward.

Ex. 30 — Using the method outlined in the last chapter, determine the scaling of this algorithm for $\hbar = m = 1$. Specifically, determine the scaling of the spatial coordinates.

5.2 Two-Dimensional Scattering

The IDL implementation of the algorithm of Askar and Cakmak is found in the function `TDSE2D_SOLVER`. This function requires the following input parameters: `x` and `y` which are vectors of equally-spaced grid points, `t` which is a vector of equally-spaced times, `wf` which is the initial wavefunction defined over the spatial grid, and `v` which is the potential defined over the spatial grid. Optional inputs include the keywords `output`, `skip`, `winpix`, and `winvis`. If the `output` keyword is set then an animated sequence of images showing the time-evolution of the wavepacket will be displayed. The speed of the animation can be controlled by altering the number of frames to skip by setting the `skip` keyword equal to this number of frames to skip. If the function `TDSE2D_SOLVER` is being called from another program that already has an active display window and pixmap, they can be passed into the function using the `winvis` and `winpix` keywords. They will then be updated. If they are not passed into the function then the function creates its own pixmap and display window.

This function differs in functionality from the GSS algorithm described in chapter 4 with the inclusion of these display keywords. The reason that we include these keywords is so that we don't have to run the function and then store a large array. That was acceptable in one dimension but the arrays to store the full time-dependence of the various quantities can become too large to do this effectively. So the implementation of the Askar and Cakmak algorithm is necessarily broken up at each time step with an option to send the probability density to a display window. However, if the `output` keyword is not set then the function returns a structure with the probability density in the `prob` field of then structure.

```
function tdse_2dsolver,x,y,t,wf_init,v,  $
                       output = output,  $
                       skip = skip,      $
                       winpix = winpix,  $
                       winvis = winvis
compile_opt idl2,hidden
!except = 0
if n_elements(skip) eq 0 then skip = 1
skip = skip > 1 & wf = reform(wf_init[*,*,0])
nx = n_elements(x) & ny = n_elements(y)
xm = rebin(x,nx,ny,/sample) & ym = rebin(transpose(y),nx,ny,/sample)
nt = n_elements(t) & dx = (x[1] - x[0])
xlo = min(x,max = xhi) & ylo = min(y,max = yhi)
dy = (y[1] - y[0]) & dt = t[1] - t[0]
a_x = 0.5*dt/(dx^2) & a_y = 0.5*dt/(dy^2)
i = complex(0.0,1.0); the imaginary unit
; Normalize the initial wavefunction
c = total((abs(wf))^2)*dx*dy
wf = wf/sqrt(c)
maxprob = max((abs(wf))^2)
output = keyword_set(output)
if output then begin
    if n_elements(winvis) eq 0 then begin
        winvis = 0 & xsize = (ysize = 700)
        window,winvis,xsize = xsize,ysize = ysize
    endif
```

```
    if n_elements(winpix) eq 0 then begin
        window,/free,/pixmap,xsize = xsize,ysize = ysize
        winpix = !d.window
    endif
endif

temp_0 = 1d*wf
; First need to evaluate the wavefunction at time t = 1*dt (i.e. n = 1)
; We will use a forward difference to approximate the time derivative
temp_1 = temp_0
temp_1[1:nx-2,1:ny-2] = (1.-i*(dt*v[1:nx-2,1:ny-2]+2.*(a_x+a_y)))*temp_0[1:nx-2,1:ny-2]+ $
                        i*a_x*(temp_0[2:nx-1,1:ny-2]+temp_0[0:nx-3,1:ny-2])+ $
                        i*a_y*(temp_0[1:nx-2,2:ny-1]+temp_0[1:nx-2,0:ny-3])
temp_2 = temp_1
if ~output then begin
    prob = fltarr(nx,ny,nt)
    prob[*,*,0] = (abs(temp_0))^2
    prob[*,*,1] = (abs(temp_1))^2
endif else begin
    wset,winpix
    title = 'T='+strtrim(string(t[0],format = '(f15.2)'),2)
    maxprob = max((abs(temp_1))^2)
    plotimage,bytscl((abs(temp_1))^2,max = maxprob),imgxrange = [xlo,xhi], $
        imgyrange = [ylo,yhi], xtitle = 'x',ytitle = 'y', title = title, $
        xstyle = 5,ystyle = 5
    contour,v,xm,ym, /noerase,xrange = [xlo,xhi],ystyle = 5,xstyle = 5,yrange = [ylo,yhi]
    wset,winvis & device,copy = [0,0,!d.x_size,!d.y_size,0,0,winpix]
endelse

; Now evolve the wavefunction in time...
for n = 1L,nt-2 do begin
    temp_2[1:nx-2,1:ny-2] = (temp_0[1:nx-2,1:ny-2]) -                                  $
                    2.*i*((2.*(a_x+a_y)+v[1:nx-2,1:ny-2]*dt)*(temp_1[1:nx-2,1:ny-2]) - $
                    a_x*((temp_1[2:nx-1,1:ny-2])+(temp_1[0:nx-3,1:ny-2])) -            $
                    a_y*((temp_1[1:nx-2,2:ny-1])+(temp_1[1:nx-2,0:ny-3])))
    temp_0 = temp_1 & temp_1 = temp_2

    if output and ((n mod skip) eq 0) then begin
        wset,winpix
        title = 'T='+strtrim(string(t[n],format = '(f15.2)'),2)
        maxprob = max((abs(temp_1))^2)
        plotimage,bytscl(((abs(temp_1))^2),max = maxprob),imgxrange = [xlo,xhi], $
          imgyrange = [ylo,yhi],title = title,xstyle = 5,ystyle = 5
        contour,v,xm,ym, /noerase,xrange = [xlo,xhi],ystyle = 5,xstyle = 5
        wset,winvis & device,copy = [0,0,!d.x_size,!d.y_size,0,0,winpix]
    endif
    if ~output then prob[*,*,n+1] = (abs(temp_2))^2
endfor
return,output ? {void:0B} : {prob:prob}
```

```
end
```

As in the one-dimensional case, we wish to scatter Gaussian wavepackets. The function `GAUSS2DWP` creates a normalized, propagating wavepacket. The required input parameters are `xm` and `ym` which are arrays of equally-spaced grid values for each of the respective dimensions. The wavepacket's location is specified via the keywords `xo` and `yo`. The propagation wavevectors are specified with the keywords `px` and `py`. The widths of the wavepacket in each dimension are specified with the keywords `sigx` and `sigy`.

```
function gauss2dwp,xm,ym,        $
                   xo = xo,      $
                   yo = yo,      $
                   px = px,      $
                   py = py,      $
                   sigx = sigx,  $
                   sigy = sigy
compile_opt idl2,hidden
; Returns a normalized two-dimensional complex
; Gaussian wavefunction.
if n_elements(xo) eq 0 then xo = 0.0
if n_elements(yo) eq 0 then yo = 0.0
if n_elements(px) eq 0 then px = 0.0
if n_elements(py) eq 0 then py = 0.0
if n_elements(sigx) eq 0 then sigx = 1.0
if n_elements(sigy) eq 0 then sigy = 1.0
i = complex(0.0,1.0)
dx = xm[1,0] - xm[0,0] & dy = ym[0,1] - ym[0,0]
arg1 = ((xm-xo)/sigx)^2+((ym-yo)/sigy)^2
arg2 = -i*(px*(xm-xo)+py*(ym-yo))
psi = exp(-0.75*arg1)*exp(-0.75*arg2)
; Normalize the probability density
c = sqrt(total((abs(psi))^2)*dx*dy)
psi = psi/c
return,psi
end
```

5.2.1 Self-Interference: Transmission through Double-Slits

The double-slit experiment demonstrated one of the most profound results of quantum mechanics–namely, wave-particle duality. In the experiment an electron gun fires electrons at a phosphorescent screen. When an electron strikes the screen, a "hit" is registered as a light dot. An absorbing screen with two small slits separated by a small distance is then placed between the electron gun and the detection screen. The first few electrons that pass through the slit system are registered on the detection screen as dots, proving their particle nature. However as many more electrons are fired from the gun, a pattern emerges on the detection screen. This pattern is composed of alternating light and dark regions with a central maximum positioned in between the two slits. The intensity of

the bands decreases the further away you go from the central maximum. This pattern is identical to a two-slit interference pattern for a wave, thus illustrating the wave-like nature of electrons.

We define the potential of the two-slit system with the function `TDSE_2D_2SLITS`. Note that in this implementation we have made the slit system not out of absorbing material but out of reflecting material so that there should be backscatter as well as transmission.

```
function tdse_2d_2slits,xm,ym
compile_opt idl2,hidden
v = 0.0*xm
a = 1.0 ; width of each slit opening
w = 0.5 ; thickness of the barrier
s = 2.0 ; center-to-center slit distance
xcond = (xm ge 0.0) and (xm le w)
ycond1 = (ym ge 0.5*(s+a)) and xcond
ycond2 = (ym le -0.5*(s+a)) and xcond
ycond3 = (ym ge -0.5*(s-a)) and (ym le 0.5*(s-a)) and xcond
indices = where(ycond1 or ycond2 or ycond3,count)
ind = array_indices(xm,indices)
vo = 100.0
if count gt 0 then v[ind[0,*],ind[1,*]] = vo
return,v
end
```

The time-evolution of the wavepacket interacting with the two-slit system is performed in the program `TDSE2D_EX1` listed below. In this program, the interaction is represented as an image. Six frames from the animation are shown in figure 5.1. From the figure it is clear that there are reflected and transmitted components. The transmitted component of the probability density shows the interference phenomena in both but the result of the experiment described above is shown in the transmitted portion.

```
pro tdse2d_ex1
; Solves the time-dependent SE in 2-dims for
; a Gaussian wavepacket interacting with a 2 slit
; system.
device,decomposed = 0 & loadct,1,/silent
; Define the space-time mesh
nx = 130 & ny = 130
x = linspace(-7.0,10.0,nx) & y = linspace(-7.0,7.0,ny)
xm = rebin(x,nx,ny,/sample) & ym = rebin(transpose(y),nx,ny,/sample)
tlo = 0.0 & thi = 1.75 & nt = 2000
t = linspace(tlo,thi,nt)
v = tdse_2d_2slits(xm,ym)
wf = gauss2dwp(xm,ym,       $
               xo = -4.0,   $
               yo = 0.0,    $
               px = 6.0,    $
```

```
               py = 0.0,    $
               sigx = 1.0,  $
               sigy = 1.0   )
ret = tdse_2dsolver(x,y,t,wf,v,skip = 15,/output)
end
```

In a measurement on the detector screen after the two-slit system the intensity pattern will be a diffraction pattern. This is shown in figure 5.2 which is the result of taking a cut of the probability density at $X = 2.2$.

Ex. 31 — What are the real physical dimensions of the slit system modeled in the dynamics shown in figure 5.1 if the particle mass is $m = 1$ u and $\alpha = 1$?

5.2.2 Dynamics of Quantum Billiards

In chapter 2 we used the discrete variable approximation to find the stationary states and eigenvalues for various one-dimensional and two-dimensional potentials. For the two-dimensional potentials the potentials were boundaries that defined closed systems. These systems have attracted interest because classical particles in stadia potentials exhibit chaotic motion which motivated the search for chaotic behavior in the quantum analogs. We can use the algorithm of Askar and Cakmak to compute the time-evolution of a wavepacket confined to move within a stadium potential. The stadium is defined in the IDL function `TDSE2D_STADIUM_POTENTIAL` below. The code that displays the animated sequence of a wavepacket propagating in a stadium potential is listed below and selected frames from the animation are shown in figure 5.3. Note that only the inner boundary of the stadium potential is shown in figure 5.3.

```
function tdse2d_stadium_potential, xm,ym
compile_opt idl2,hidden
v = 0.0*xm
r = 3.0 & thick = 0.5 & l = 5.0
xo = 0.0 & yo = 0.0
vo = 80.0
; Define region II (the mid region)
cond_II_1 = ((xm ge -(0.5*L+thick)) and $
  (xm le (0.5*L+thick))) and ((ym ge R) and (ym le (R+thick)))
cond_II_2 = ((xm ge -(0.5*L+thick)) and $
  (xm le (0.5*L+thick))) and ((ym ge -(R+thick)) and (ym le -R))
cond_II = cond_II_1 or cond_II_2
; Define region III (the positive circular region)
cond_III = (xm ge 0.5*L) and ((((xm-0.5*L)^2 + (ym)^2) ge R^2) and $
  (((xm-0.5*L)^2 + (ym)^2) le (R+thick)^2))
; Define region I
cond_I = (xm le -0.5*L) and ((((xm+0.5*L)^2 + (ym)^2) ge R^2) and $
  (((xm+0.5*L)^2 + (ym)^2) le (R+thick)^2))
cond = (cond_I or cond_III or cond_II)
indices = where(cond,count)
ind = array_indices(xm, indices)
```

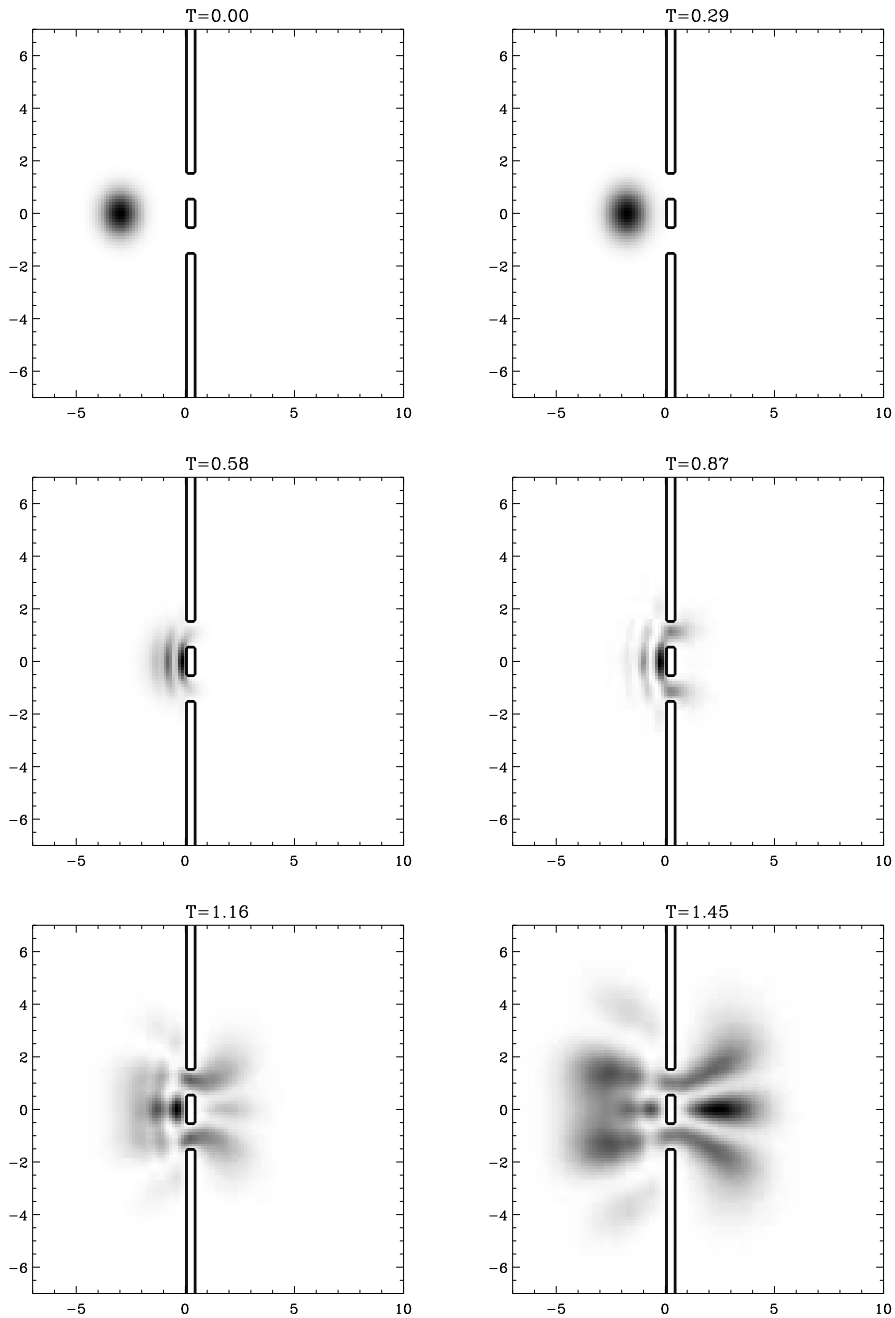


Figure 5.1: Time-evolution of the probability density for a wavepacket scattering from a two-slit system. Note the significant amount of "backscatter", especially at later times, due to reflection from the potential.

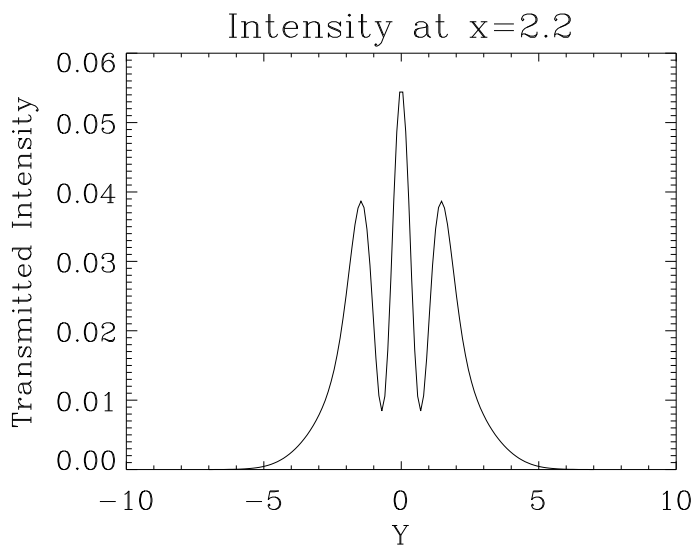


Figure 5.2: Intensity pattern found in a detector screen placed at $X = 2.2$ for the example displayed in figure 5.1.

```
if count gt 0 then v[ind[0,*],ind[1,*]] = vo
return,v
end



pro tdse2d_ex2
; Solves the time-dependent SE in 2-dims for
; a Gaussian wavepacket confined to a stadium
; potential.
device,decomposed = 0
loadct,5,/silent
; Define the space-time mesh
nx = 130 & ny = 130
x = linspace(-9.0,9.0,nx) & y = linspace(-5.0,5.0,ny)
xm = rebin(x,nx,ny,/sample) & ym = rebin(transpose(y),nx,ny,/sample)
tlo = 0.0 & thi = 5.0 & nt = 4000
t = linspace(tlo,thi,nt)
v = tdse2d_stadium_potential(xm,ym)
wf = gauss2dwp(xm,ym,       $
               xo = -2.0,   $
               yo = 0.0,    $
               px = -3.0,    $
               py = 0.0,    $
               sigx = 1.0,  $
               sigy = 1.0   )
```

```
skip = 15
result = tdse_2dsolver(x,y,t,wf,v,skip = skip,/output)
end
```

* **Ex. 32** — For the stadium potential can you select a set of initial conditions for the Gaussian wavepacket so that when $k_x = k_y = 0$ initially, the center-of-mass of the wavepacket will propagate? Hint: try modifying x_0 and y_0. Modify the program `TDSE2D_EX2` to see if you can obtain this behavior.

5.2.3 Dynamics of a Wavepacket in Archimedes' Spiral

In this final section on dynamics in two spatial dimensions we look at the time-evolution of a wavepacket initially at rest in the center of a curve called Archimedes' spiral. The fascinating aspect of this system is that the wavepacket does not remain at rest. Because of the quantum nature of the system, the wavepacket spreads and begins to interact with the curved wall. In so doing, it acquires angular momentum and begins to curl around the spiral until it leaks out.

The function that creates the potential boundary given by Archimedes' spiral is given in `TDSE2D_ARCHIMEDES_SPIRAL` below. This function uses a unique manner to obtain the potential. First, the spiral is generated with a complex parametric equation $z(t) = te^{it}$ where i is the imaginary unit. The x and y (real and imaginary) parts of $z(t)$ are used as the coordinates for the curve. Specifically,

$$\begin{aligned} z(t) &= te^{it} \\ &= t\cos t + it\sin t \\ &= x(t) + iy(t). \end{aligned}$$

Therefore the x and y components of the spiral are defined parametrically by $x(t) = t\cos t$ and $y(t) = t\sin t$. This curve is drawn to a pixmap in IDL—a window in memory that has the same size and dimensions as the mesh for the computation—and then the IDL function `TVRD` is used to obtain a byte image of the window's contents. The image is finally scaled into an appropriate range for the potential.

```
function tdse2d_archimedes_spiral,xm,ym
compile_opt idl2,hidden
xms = size(xm,/dimensions)
xsize = xms[0] & ysize = xms[1]
v = 0.0*xm & vo = 80.0
n = 100 & tlo = 0.5*!pi & dt = 3.*!pi/(n-1.0)
t = tlo+dt*findgen(n)
i = complex(0.0,1.0)
z = t*exp(i*t)
x = float(z) & y = imaginary(z)
im1 = bytarr(xsize,ysize)
window,/free,/pixmap,xsize = xsize,ysize = ysize
winpix = !d.window
xr = [-10.0,10.0] & yr = xr
```

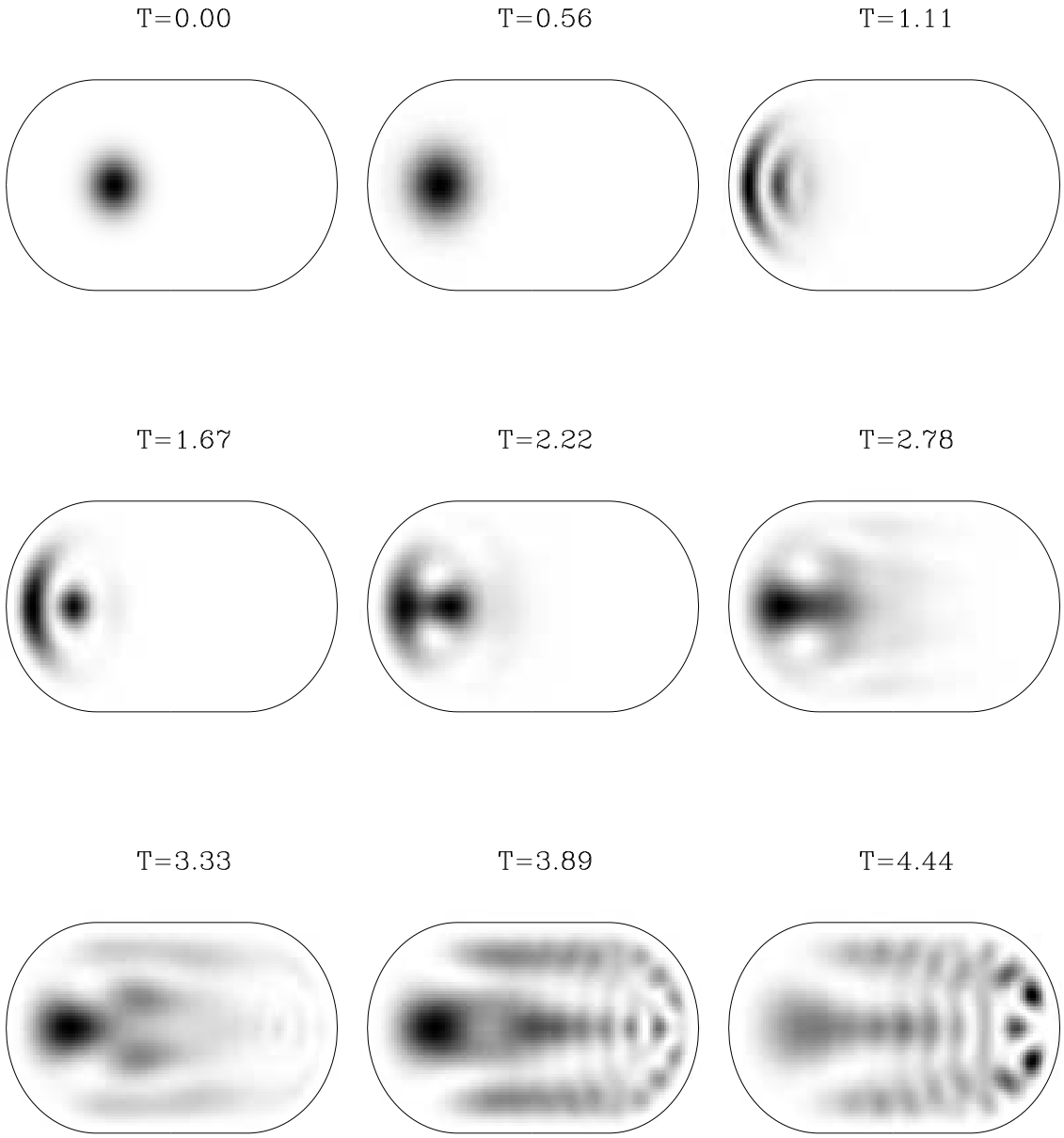


Figure 5.3: Time-evolution of the probability density corresponding to a propagating Gaussian wavepacket in a stadium potential. The wavepacket is given an initial wavevector $k_x = -3$, it is initially located at $x_0 = -2$, and has a standard deviation defined by $\sigma_x = \sigma_y = 1$.

```
tvlct,r,g,b,/get
device,decomposed = 0
loadct,0,/silent
plot,x,y,xrange = xr,yrange = yr,xstyle = 5,ystyle = 5,$
    thick = 4.0,xmargin = [0,0],ymargin = [0,0]
img = tvrd()
wdelete,winpix
device,decomposed = 1
tvlct,r,g,b
v = float(img)*(vo/255.0)
return,v
end
```

The program that illustrates the time-evolution of an initially stationary wavepacket at the center of the spiral is TDSE2D_EX3, shown below. Initally the Gaussian wavepacket has a width defined by $\sigma_x = \sigma_y = 0.5$. As time progresses, the center-of-mass of the wavepacket moves outward along the path of the spiral. Selected frames from the animation sequence are shown in figure 5.4. Note that only a trace of the spiral boundary is shown in figure 5.4 for clarity. The contour is shown in your animation TDSE2D_EX3.

```
pro tdse2d_ex3
; Solves the time-dependent SE in 2-dims for
; a Gaussian wavepacket confined initially to
; the center of Archimedes' spiral.
device,decomposed = 0
loadct,3,/silent
; Define the space-time mesh
nx = 100 & ny = 100
x = linspace(-4.0,4.0,nx) & y = linspace(-4.0,4.0,ny)
xm = rebin(x,nx,ny,/sample) & ym = rebin(transpose(y),nx,ny,/sample)
tlo = 0.0 & thi = 5.0 & nt = 5000
t = linspace(tlo,thi,nt)
v = tdse2d_archimedes_spiral(xm,ym)
wf = gauss2dwp(xm,ym,      $
               xo = 0.0,   $
               yo = -0.5,  $
               px = 0.0,   $
               py = 0.0,   $
               sigx = 0.5, $
               sigy = 0.5  )
skip = 15
result = tdse_2dsolver(x,y,t,wf,v,skip = skip,/output)
end
```

*** **Ex. 33** — Calculate the angular momentum as a function of space and time for a Gaussian wavepacket initially stationary in a potential with a boundary defined by Archimedes' spiral.

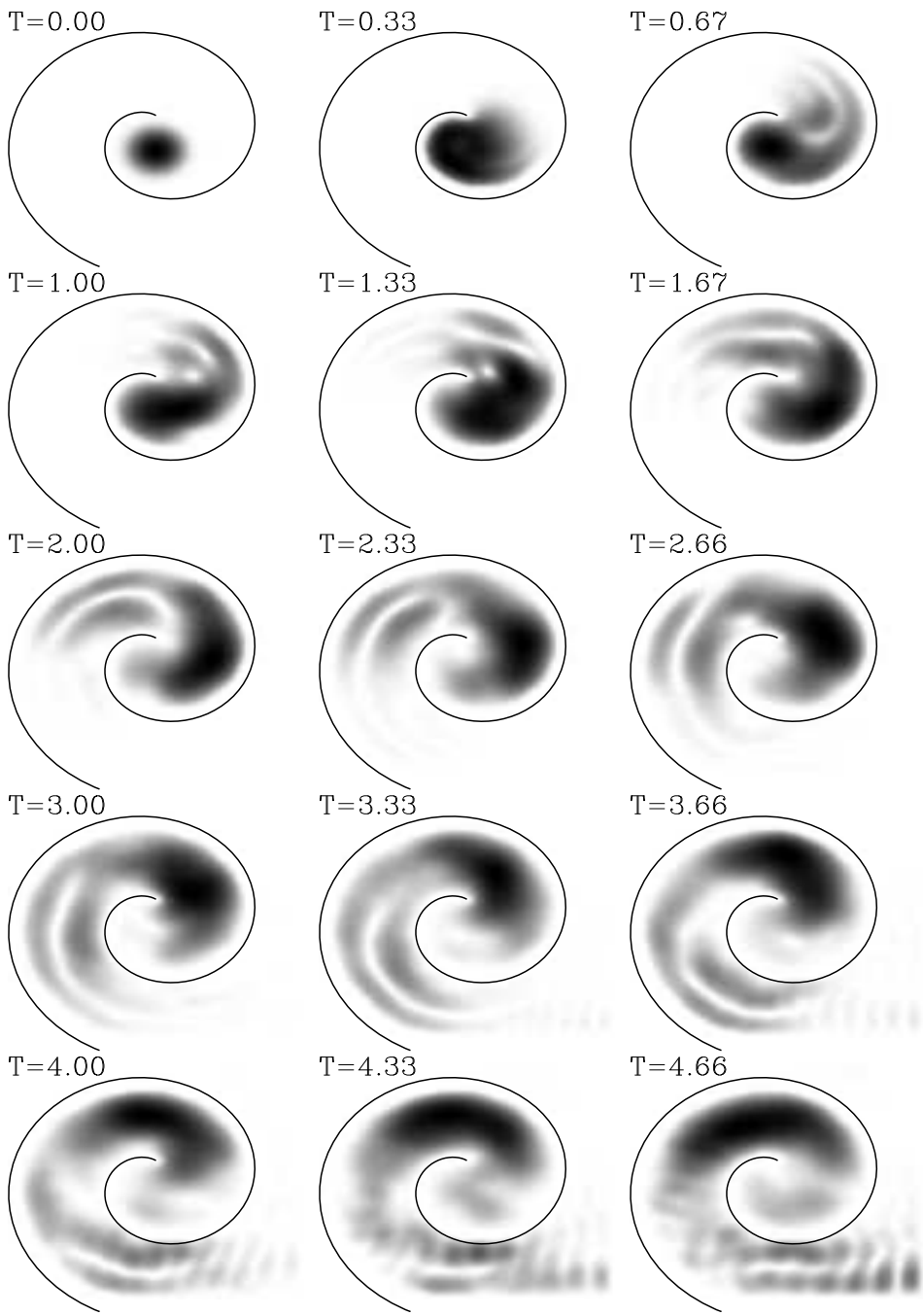


Figure 5.4: Time-evolution of the probability density corresponding to an initially stationary Gaussian wavepacket in a potential with a boundary defined by Archimedes' spiral. The wavepacket initially has a width defined by $\sigma_x = \sigma_y = 0.5$.

Display the time-evolution as a function of time.

5.3 Colliding Particles

Up until this point we have discussed a single particle interacting with a potential, either in one or two dimensions. Therefore, the collisions we have observed thus far have been elastic. In nature a particle does not generally interact with just a potential but rather with other particles–and the interaction is mediated through some potential. For instance, two particles can scatter from each other after interacting through a short-range repulsive interaction. This is often referred to as a billiard ball collision. If the target particle is significantly more massive than the probe particle then the system is indeed approximated well by a single particle interacting with a potential. In two-particle collisions, one particle can gain or lose energy to the other but energy is conserved for the overall system. In neutron scattering measurements—or any particle experiment where a particle or beam of particles scatters from a target particle—energy can be exchanged between the target and probe particles and this then provides direct information on the dynamics of the target. The researcher must figure out what the target was doing based on the behavior of the scattered particles. This can be challenging and our goal in this last section is to provide a space-time view of simple two-particle scattering events.

As described in chapter 1, we will consider two one-dimensional particles interacting through a pair potential $V(x_1, x_2)$. This pair potential can be composed of a perhaps three additive terms, two of which ($V_1(x_1)$ and $V_2(x_2)$) could govern the behavior of the individual particles and the third ($V_{1,2}(x_1, x_2)$) could describe the interaction between the two particles. This necessarily increases the dimensionality of the Schrödinger equation that describes the dynamics of the system to two dimensions. The equation that governs the dynamics, equation 1.12, is repeated below for reference:

$$H = -\hbar^2 \left(\frac{1}{2m_1} \frac{\partial^2}{\partial x_1^2} + \frac{1}{2m_2} \frac{\partial^2}{\partial x_2^2} \right) + V(x_1, x_2). \tag{5.9}$$

We will set $\hbar = 1$ but we will leave the flexibility of having different particle masses, m_1 and m_2. Furthermore, since the Hamiltonian above acts on a two-particle entangled wavefunction, $\langle x_1, x_2 | \psi \rangle$, we cannot in general disentangle the single-particle probability densities. However we can marginalize the probability densities as described in chapter 1 via

$$\rho_1(x_1) = \int_{-\infty}^{\infty} \mathrm{d}x_2 \left| \langle x_1, x_2 | \psi \rangle \right|^2 \tag{5.10}$$

and

$$\rho_2(x_2) = \int_{-\infty}^{\infty} \mathrm{d}x_1 \left| \langle x_1, x_2 | \psi \rangle \right|^2. \tag{5.11}$$

When we observe the dynamics long before or long after the collision then these expressions, $\rho_1(x_1)$ and $\rho_2(x_2)$, are good approximations to the probability densities for the individual particles.

The modification of the algorithm of Askar and Cakmak is straightforward. In order to calculate the dynamics of two colliding particles with different masses, m_1 and m_2, we need to slightly modify equation 5.7,

$$\psi_{j,k}^{n+1} = \psi_{j,k}^{n-1} - 2i[(2(\alpha_{x_1} + \alpha_{x_2}) + V_{j,k}\Delta t)\psi_{j,k}^n - \alpha_{x_1}\left(\psi_{j+1,k}^n + \psi_{j-1,k}^n\right) - \alpha_{x_2}\left(\psi_{j,k+1}^n + \psi_{j,k-1}^n\right)] \tag{5.12}$$

where

$$\alpha_{x_1} = \frac{1}{2m_1}\frac{\Delta t}{\Delta x_1^2} \tag{5.13}$$

and

$$\alpha_{x_2} = \frac{1}{2m_2}\frac{\Delta t}{\Delta x_2^2}. \tag{5.14}$$

The function that solves the two-particle Schrödinger equation is named `TDSE_2PARTICLE_SOLVER` and it is listed below. In this function, the wavefunction for each of the particles is assumed to be defined on the same x-coordinate. The manner in which we treat the true two-dimensionality of the problem is that we inflate the `x` vector into an array that is $N_x \times N_x$ to create x_1 and transpose it to create x_2. The required input parameters are `x`, `t`, `wf1`, and `wf2`. These are all one-dimensional vectors specifying the spatial coordinate, time, wavefunction for particle 1 and wavefunction for particle 2, respectively. The optional input parameters are `v1`, `v2`, `v12`, `m1`, `m2`, `output`, `skip`, `winpix`, and `winvis`. All of the potential terms, `v1`, `v2`, and `v12` are arrays specified on the grid defined by the outer product$x \times x^T$ which has dimensions $N_x \times N_x$. The masses of the particles are specified by `m1` and `m2`. The remaining keywords are identical to those in `TDSE2D_SOLVER`.

```
function tdse_2particle_solver,x,t,wf1,wf2,    $
                       v1 = v1,                $
                       v2 = v2,                $
                       v12 = v12,              $
                       m1 = m1,          $
                       m2 = m2,          $
                       output = output,  $
                       skip = skip,      $
                       winpix = winpix,  $
                       winvis = winvis
compile_opt idl2,hidden
!except = 0
if n_elements(skip) eq 0 then skip = 1
skip = skip > 1
if n_elements(m1) eq 0 then m1 = 1.0
if n_elements(m2) eq 0 then m2 = 1.0
nx = n_elements(x) & xm = rebin(x,nx,nx,/sample)
xlo = min(x,max = xhi) & dx = x[1] - x[0]
dt = t[1] - t[0] & nt = n_elements(t)
nframes = fix(float(nt)/float(skip))+1
a_1 = 0.5*dt/(dx^2) & a_2 = 0.5*dt/(m2*dx^2)
i = complex(0.0,1.0); the imaginary unit
output = keyword_set(output)
if output then begin
    if n_elements(winvis) eq 0 then begin
        winvis = 0 & xsize = (ysize = 400)
        window,winvis,xsize = xsize,ysize = ysize
    endif
    if n_elements(winpix) eq 0 then begin
        window,/free,/pixmap,xsize = xsize,ysize = ysize
        winpix = !d.window
```

```
    endif
endif
wf = wf1#wf2
c = total(abs(wf)^2) & wf = wf/sqrt(c)
if n_elements(v12) eq 0 then v12 = 0*xm
if n_elements(v1) eq 0 then v1 = 0*xm
if n_elements(v2) eq 0 then v2 = 0*xm
v = v1 + v2 + v12 & rho = abs(wf)^2
p1 = dx*total(rho,2) & p2 = dx*total(rho,1)
xr = [xlo,xhi] & yr = [0.0,1.1*max([max(p1),max(p2)])]
temp_0 = 1d*wf
; First need to evaluate the wavefunction at time t = 1*dt (i.e. n = 1)
; We will use a forward difference to approximate the time derivative
temp_1 = temp_0
temp_1[1:nx-2,1:nx-2] = (1.-i*(dt*v[1:nx-2,1:nx-2]+$
                        2.*(a_1+a_2)))*temp_0[1:nx-2,1:nx-2]+ $
                        i*a_1*(temp_0[2:nx-1,1:nx-2]+temp_0[0:nx-3,1:nx-2])+ $
                        i*a_2*(temp_0[1:nx-2,2:nx-1]+temp_0[1:nx-2,0:nx-3])
temp_2 = temp_1
counter = 0L
if ~output then begin
    rho1 = fltarr(nx,nframes) & rho2 = fltarr(nx,nframes)
    rho = (abs(temp_0))^2
    rho1[*,counter] = p1
    rho2[*,counter] = p2
endif else begin
    rho = abs(temp_0)^2
    rho1 = p1  ; P(x1)
    rho2 = p2  ; P(x2)
    wset,winpix
    title = '!6T='+strtrim(string(t[0],format = '(f15.2)'),2)
    plot,x,p1,xtitle = '!6X',psym = 0,thick = 2,xrange = xr,$
      title = title, yrange = yr,xstyle = 1,ystyle = 1,$
      ytitle = '!7q!6!d1!n(X!d1!n,T), !7q!6!d2!n(X!d2!n,T)', $
      color = fsc_color('black'),background = fsc_color('white')
    oplot,x,p2,psym = 0,thick = 2,color = fsc_color('black'),linestyle = 2
    wset,winvis & device,copy = [0,0,!d.x_size,!d.y_size,0,0,winpix]
endelse
; Now evolve the wavefunction in time...
for n = 1L,nt-2 do begin
    temp_2[1:nx-2,1:nx-2] = (temp_0[1:nx-2,1:nx-2]) -                                $
                    2.*i*((2.*(a_1+a_2)+v[1:nx-2,1:nx-2]*dt)*(temp_1[1:nx-2,1:nx-2])- $
                    a_1*((temp_1[2:nx-1,1:nx-2])+(temp_1[0:nx-3,1:nx-2])) -          $
                    a_2*((temp_1[1:nx-2,2:nx-1])+(temp_1[1:nx-2,0:nx-3])))
    temp_0 = temp_1 & temp_1 = temp_2
    rho = abs(temp_0)^2
    p1 = dx*total(rho,2)  ; P(x1)
    p2 = dx*total(rho,1)  ; P(x2)
    if output and ((n mod skip) eq 0) then begin
```

```
        wset,winpix
        title = '!6T='+strtrim(string(t[n],format = '(f15.2)'),2)
        plot,x,p1,xtitle = '!6X',psym = 0,thick = 2,xrange = xr,yrange = yr,$
          xstyle = 1, ystyle = 1,title = title,$
          ytitle = '!7q!6!d1!n(X!d1!n,T), !7q!6!d2!n(X!d2!n,T)',$
          color = fsc_color('black'),background = fsc_color('white')
        oplot,x,p2,psym = 0,thick = 2,color = fsc_color('black'),linestyle = 2
        wset,winvis & device,copy = [0,0,!d.x_size,!d.y_size,0,0,winpix]
    endif
    if ((~output) and ~(n mod skip)) then begin
      counter++
      rho1[*,counter] = p1
      rho2[*,counter] = p2
    endif
endfor
return,output ? {void:0B} : {prob1:rho1,prob2:rho2}
end
```

The interaction term that we will choose for the examples presented in the next sections will be repulsive and can be written as

$$V\left(x_1, x_2; \sigma\right) = V_0 1^+\left(\sigma - \left|x_1 - x_2\right|\right) \tag{5.15}$$

where σ is the interaction distance, V_0 is the interaction strength, and $1^+\left(x\right)$ is the Heaviside step function. This function, $1^+\left(x\right)$, is implemented in IDL in `HEAVISIDE_STEP` shown below.

```
function heaviside_step,x
compile_opt idl2,hidden
return,(x gt 0) + 0.0*(x lt 0) + 0.5*(x eq 0)
end
```

5.3.1 Colliding Billiard Balls

Perhaps the simplest example of two particles interacting is that of two free particles (i.e. $V_1\left(x_1\right) = V_2\left(x_2\right) = 0$ for each particle) that interact when they "touch." The notion of particles "touching" in quantum mechanics is not as plain as it is for classical particles where surface contact is a meaningful definition. We will assume that a "contact" potential like the repulsive interaction term in equation 5.15 is the only term in the potential and that the initial wavefunctions are Gaussian wavepackets. The program `FREE_PARTICLE_COLLISION` displays the time-evolution for a system composed of a wavepacket of mass $m_1 = 3$ at rest located at $x_0 = -5$ and a wavepacket of mass $m_2 = 1$ initially located at $x_0 = -15$ with a wavevector $k_x = 3$.

```
pro free_particle_collision
; This program shows a collision between
; two free particles interacting through a
; hard-core repulsion.
```

```
nx = 146 & nt = 5000
xlo = -25.0 & xhi = 25.0 & x = linspace(xlo,xhi,nx)
xm = rebin(x,nx,nx,/sample)
dt = 0.002 & t = dt*findgen(nt)
i = complex(0.0,1.0)
; Define the initial wavepackets
fwhm1 = 5.0 & cen1 = -15.0 & k1 = 3.0 & sig1 = fwhm1/2.354
arg1 = 0.5*((x-cen1)/sig1)^2
g1 = (1./sqrt(2.0*!pi*sig1^2))*exp(-arg1)*exp(i*k1*x)
fwhm2 = 5.0 & cen2 = -5.0 & k2 = 0.0 & sig2 = fwhm2/2.354
arg2 = 0.5*((x-cen2)/sig2)^2
g2 = (1./sqrt(2.0*!pi*sig2^2))*exp(-arg2)*exp(i*k2*x)
; Define the interaction potential
vo = 50.0 & sigint = 1.0
vint = vo*heaviside_step(sigint-abs(xm-transpose(xm)))
ret = tdse_2particle_solver(x,t,g1,g2,v12 = vint,$
  skip = 15,/output,m1 = 3.0,m2 = 1.0)
end
```

The marginalized densities, $\rho_1(x_1)$ and $\rho_2(x_2)$, are displayed in figure 5.5. The quantum nature of the particles is seen in a number of aspects of this collision. First, the second particle (dashed line) starts to move to the right before there is appreciable overlap of the wavepackets. For a classical particle we expect that the particles would move as soon as there was any overlap. Second, both wavepackets spread out a visible amount as the collision progresses, thus illustrating the dispersive nature of a quantum particle.

Ex. 34 — Change the values for m_1 and m_2 to determine the effects on the collision process. For the case of a wavepacket of low mass scattering from one with large mass, what do you expect to see in the collision process? Hint: use your intuition of classical scattering of two billiard balls of unequal mass. In particular try $m_1 = 1$ and $m_2 = 10$.

Ex. 35 — Change the values for σ and V_0, the interaction range and interaction strength in equation 5.15, and determine the effects on the collision process.

** **Ex. 36** — Write a program that transforms the two-particle, time-dependent wavefunction $\langle x_1, x_2|\psi\rangle(t)$ obtained using `TDSE_2PARTICLE_SOLVER` into momentum space, $\langle k_1, k_2|\phi\rangle(t)$. Next use marginalization to obtain $\rho_1(k_1, t)$ and $\rho_2(k_2, t)$ in the same way that we used it in position space as in equations 5.10 and 5.11. Watch the time-evolution of $\rho_1(k_1, t)$ and $\rho_2(k_2, t)$. Does the time-evolution of the momentum distributions make sense qualitatively based on your intuition of the classical collision process?

Ex. 37 — Explore the effects of making the interaction attractive (i.e. $V_0 = -50$).

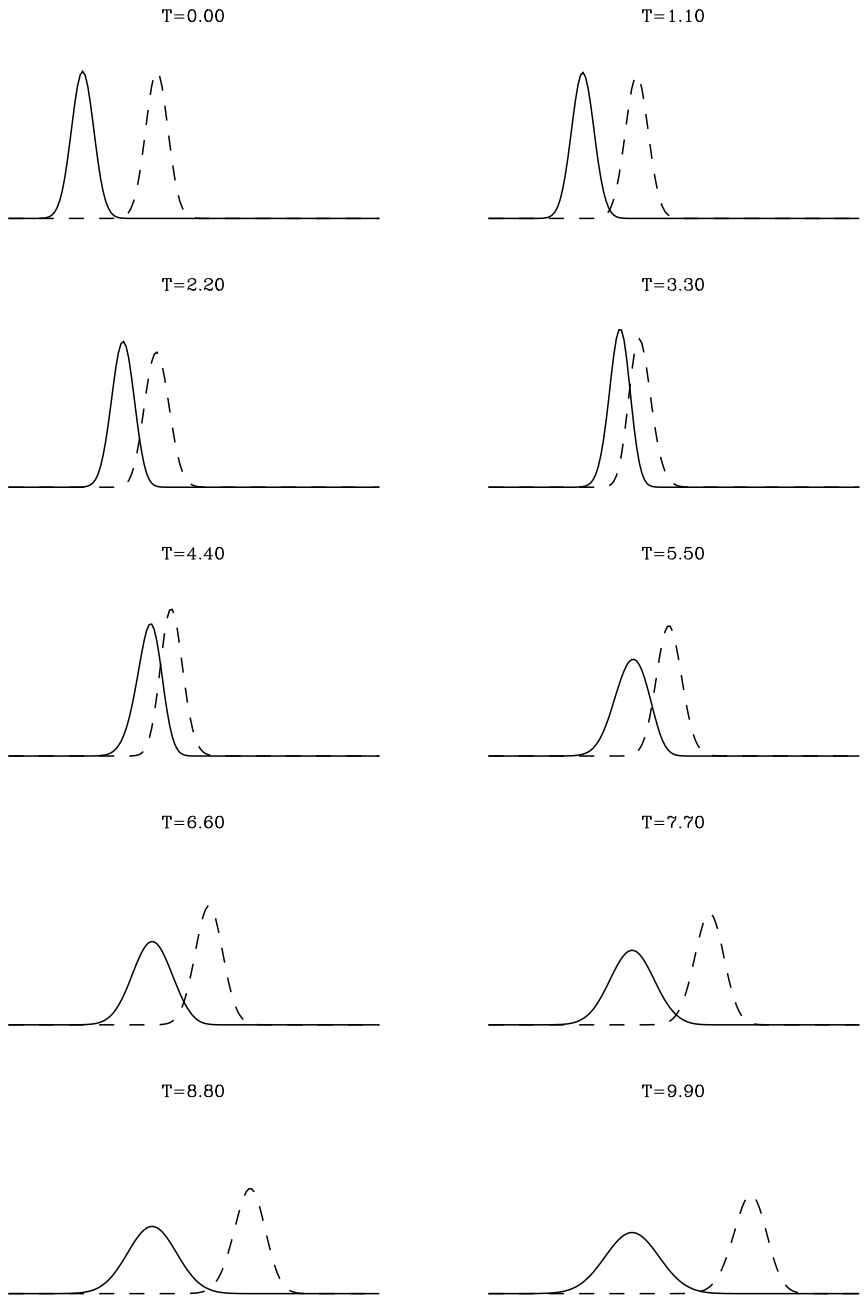


Figure 5.5: Time-evolution of the probability densities corresponding to a propagating Gaussian wavepacket ($k_1 = 3$) of mass $m_1 = 3$ colliding with a stationary Gaussian wavepacket of mass $m_2 = 1$. The interaction between the two particles is repulsive with $V_0 = 50$ and $\sigma = 1$. The wavepacket drawn with the solid line is initially propagating ($k = 3$) towards the stationary wavepacket, drawn with the dashed line.

5.3.2 Scattering from an Harmonic Oscillator

A problem that is often treated in introductory courses on classical mechanics is that of a particle colliding with a mass on a spring. This is an interesting example because it illustrates how the a free particle can give up its kinetic energy to the mass on the spring by compressing the spring and then give it back to the mass and set the mass on the spring into oscillations. This is a good example of a system exhibiting energy transfer between its components.

The quantum analog of this system is interesting and it is easy to observe the dynamics of this system using the programs we've developed so far. For the free particle we still use a Gaussian wavepacket. However we model the mass m_2 on the spring as a quantum simple harmonic oscillator whose natural frequency is given by ω_0. The energy eigenstates of this oscillator are given by $E_n = (n + 1/2)\hbar\omega_0$. We require the eigenstates of the simple harmonic oscillator and the IDL implementation of these eigenstates, `SHO_EIGENSTATES`, is presented in Appendix B. The oscillator eigenstates use Hermite polynomials whose implementation is also described in Appendix B. The program `SHO_PARTICLE_COLLISION` illustrates the collision event through an animated sequence. Selected frames from that sequence are shown in figure 5.6.

```
pro sho_particle_collision
; This program shows a collision between
; a free particle and a simple harmonic
; oscillator interacting through a
; hard-core repulsion.
nx = 146 & nt = 3900
xlo = -20.0 & xhi = 10.0 & x = linspace(xlo,xhi,nx)
xm = rebin(x,nx,nx,/sample)
dt = 0.002 & t = dt*findgen(nt)
i = complex(0.0,1.0)
; Define the initial wavepackets
m1 = 1.0 & m2 = 1.0
fwhm1 = 5.0 & cen1 = -5.0 & k1 = 3.0 & sig1 = fwhm1/2.354
arg1 = 0.5*((x-cen1)/sig1)^2
g1 = (1./sqrt(2.0*!pi*sig1^2))*exp(-arg1)*exp(i*k1*x)
wo = 1.0  ; oscillator (radial) frequency
wf2 = sho_eigenstates(0,x,m2,wo)
vsho = 0.5*m2*(wo*transpose(xm))^2
; Define the interaction potential
vo = 50.0 & sigint = 1.0
vint = vo*heaviside_step(sigint-abs(xm-transpose(xm)))
ret = tdse_2particle_solver(x,t,g1,wf2,v12 = vint+vsho,$
  skip = 15,/output,m1 = m1, m2 = m2)
end
```

In the animated sequence of the free wavepacket colliding with a ground state simple harmonic oscillator we see that the wavepacket gives up all of its energy, coming to rest when the oscillator reaches its maximum deviation to the right (i.e. it's maximally "compressed"). Clearly the oscillator does not remain in its ground state but instead develops motion due to transitions between the oscillator eigenstates induced by the interaction with the free wavepacket. After the

oscillator maximally "compresses", the wavepacket reverses direction while the oscillator continues its oscillation about its initial position. However it is clear that the wavepacket has excited additional oscillator modes in it's probability density evident in the additional structure observed in $\rho_2(x_2, t)$. This additional structure can only come from higher-order eigenstate components. The other noteworthy point seen in this animated sequence is that the free particle wavepacket broadens appreciably throughout the course of the collision process.

Next we can observe the consequences of preparing the simple harmonic oscillator in its first excited state with energy $E_1 = (3/2)\hbar\omega_0$. This is done by changing the line in the program `SHO_PARTICLE_COLLISION` from

```
wf2 = sho_eigenstates(0,x,m2,wo)
```

to

```
wf2 = sho_eigenstates(1,x,m2,wo).
```

Once again the free particle wavepacket starts with an initial speed corresponding to $k_1 = 3$. Select frames from the animated sequence are shown in figure 5.7. In this sequence we see a curious phenomenon occur which is a result of the wave nature of the system. During the period in which the particle is "in contact" with the oscillator, the free wavepacket undergoes fission and is composed of a leading and trailing edge for a short period. When the first of these two components is reflected and begins moving to the left, the second component comes to rest. But the second component gets a large boost from the oscillator moving to the left so that this second component ultimately catches up to the first reflected component.

Ex. 38 — Change the values for m_1 and m_2 to determine the effects on the collision process. You will have to change the discretization in space and time to ensure stability of the algorithm.

Ex. 39 — Explore the effects of making the interaction attractive (i.e. $V_0 = -50$).

* **Ex. 40** — Modify the program so that the oscillator's initial wavefunction is compposed of two eigenstates. Obviously this is not a stationary state so it will be moving. For the case of an equal mixture of $n = 0$ and $n = 1$ oscillator states, look at the time-evolution of the collision of a free wavepacket ($k_1 = 3$) with this moving oscillator.

** **Ex. 41** — Write a program that transforms the two-particle, time-dependent wavefunction $\langle x_1, x_2|\psi\rangle(t)$ obtained using `TDSE_2PARTICLE_SOLVER` into momentum space, $\langle k_1, k_2|\phi\rangle(t)$. Next use marginalization to obtain $\rho_1(k_1, t)$ and $\rho_2(k_2, t)$ in the same way that we used it in position space as in equations 5.10 and 5.11. Watch the time-evolution of $\rho_1(k_1, t)$ and $\rho_2(k_2, t)$ for a free particle wavepacket with $k_1 = 3$ and an oscillator in its ground state. Does the time-evolution of the momentum distributions make sense qualitatively based on your intuition of the classical collision process? How do the momentum distribution for the free particle and the oscillator differ?

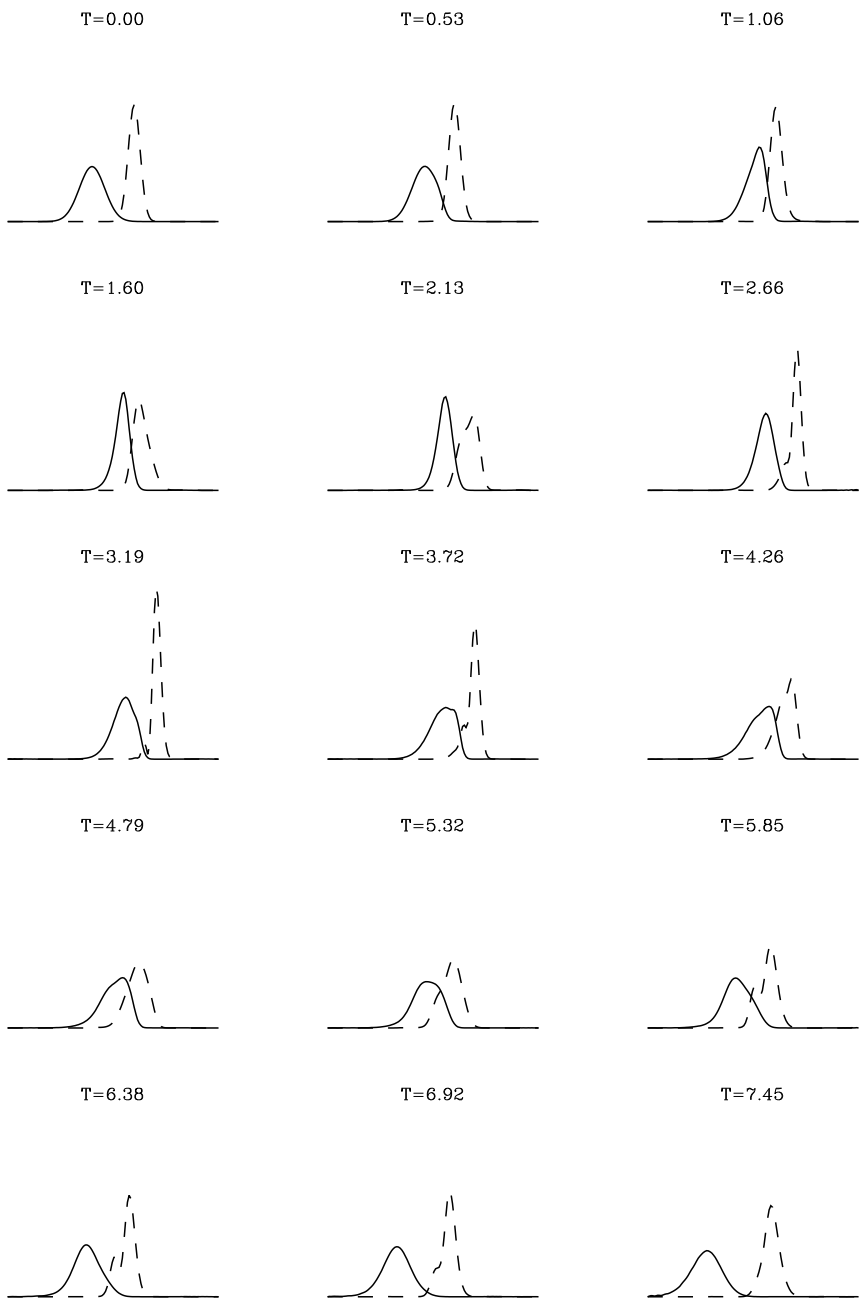


Figure 5.6: Time-evolution of the probability densities corresponding to a propagating Gaussian wavepacket of mass $m_1 = 1$ (solid line) and $k_1 = 3$ colliding with a simple harmonic oscillator of the same mass, initially at rest, in its ground state (dashed line).

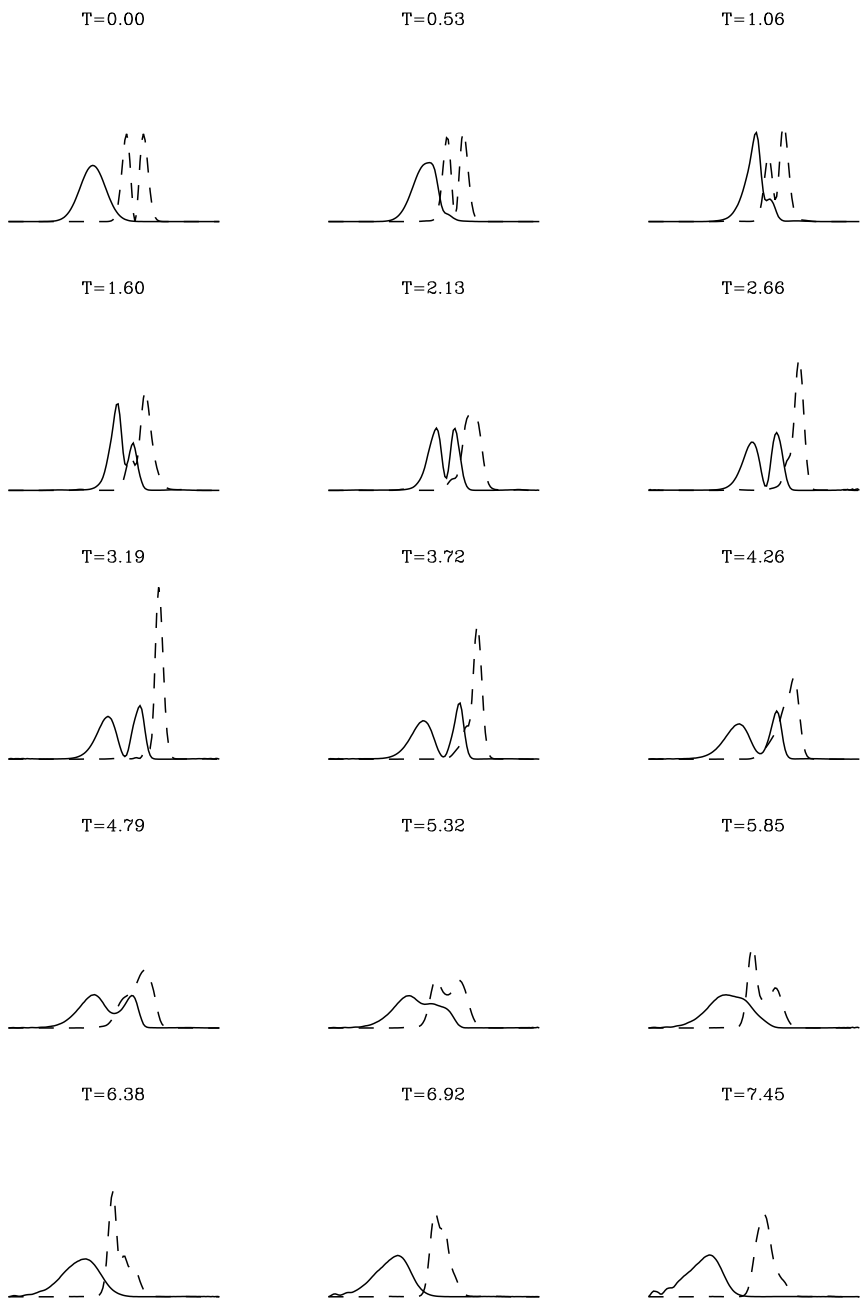


Figure 5.7: Time-evolution of the probability densities corresponding to a propagating Gaussian wavepacket of mass $m_1 = 1$ (solid line) and $k_1 = 3$ colliding with a simple harmonic oscillator of the same mass, initially at rest, in its first excited state (dashed line).

Chapter 6

Quantum Visualizations

In this concluding chapter we digress from the practical aspects of computing quantum phenomena. Here we present a number of different visualizations in one and two dimensions in color. One purpose of doing this is simply to illustrate how the algorithms presented in this book can be used to create interesting images. Closely related to that is that this presentation will stimulate the reader to create his/her own visualizations through experimentation.

In chapter 4 we described how to compute the effects of wavepackets undergoing collisions with potential barriers, wells, and combinations of the two in one dimension. Animations showing the time-evolution of the wavepackets were presented and frames from those animations were presented in many of the figures in that chapter. Another way to view the dynamics of the wavepacket scattering event is through a space-time image plot where the horizontal axis is the spatial dimension and the vertical axis is time. Even when we investigated the collision of two one-dimensional particles in chapter 5 we looked at those dynamics via animations of the two one-dimensional probability densities as well as sequences of frames from those animations. Those dynamics, too, can be viewed in an image representation of the space-time plane. However in this representation we plot the sum of the two marginal probability densities (as described in chapter 5): $\rho_1(x_1, t) + \rho_2(x_2, t)$.

The cases shown here are closely related—if not exactly related through the same potentials and initial conditions—to the cases presented in previous chapters.

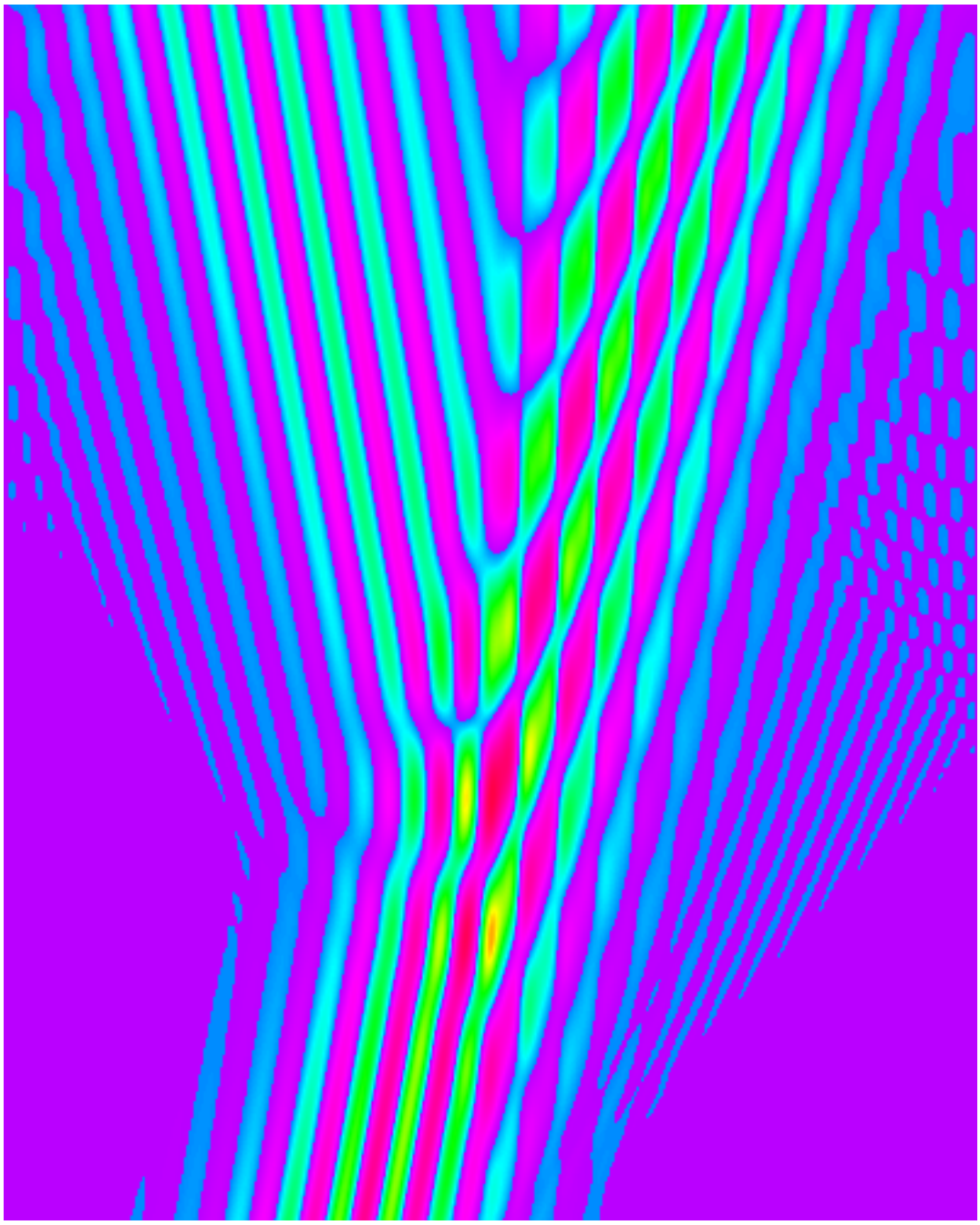


Figure 6.1: Space-time plot of the evolution of a Gaussian wavepacket scattering at the interface of a periodic potential. The position axis is horizontal and the time axis is vertical. The periodic potential begins at the center of the horizontal axis and continues to the right. The real part of the wavefunction is displayed.

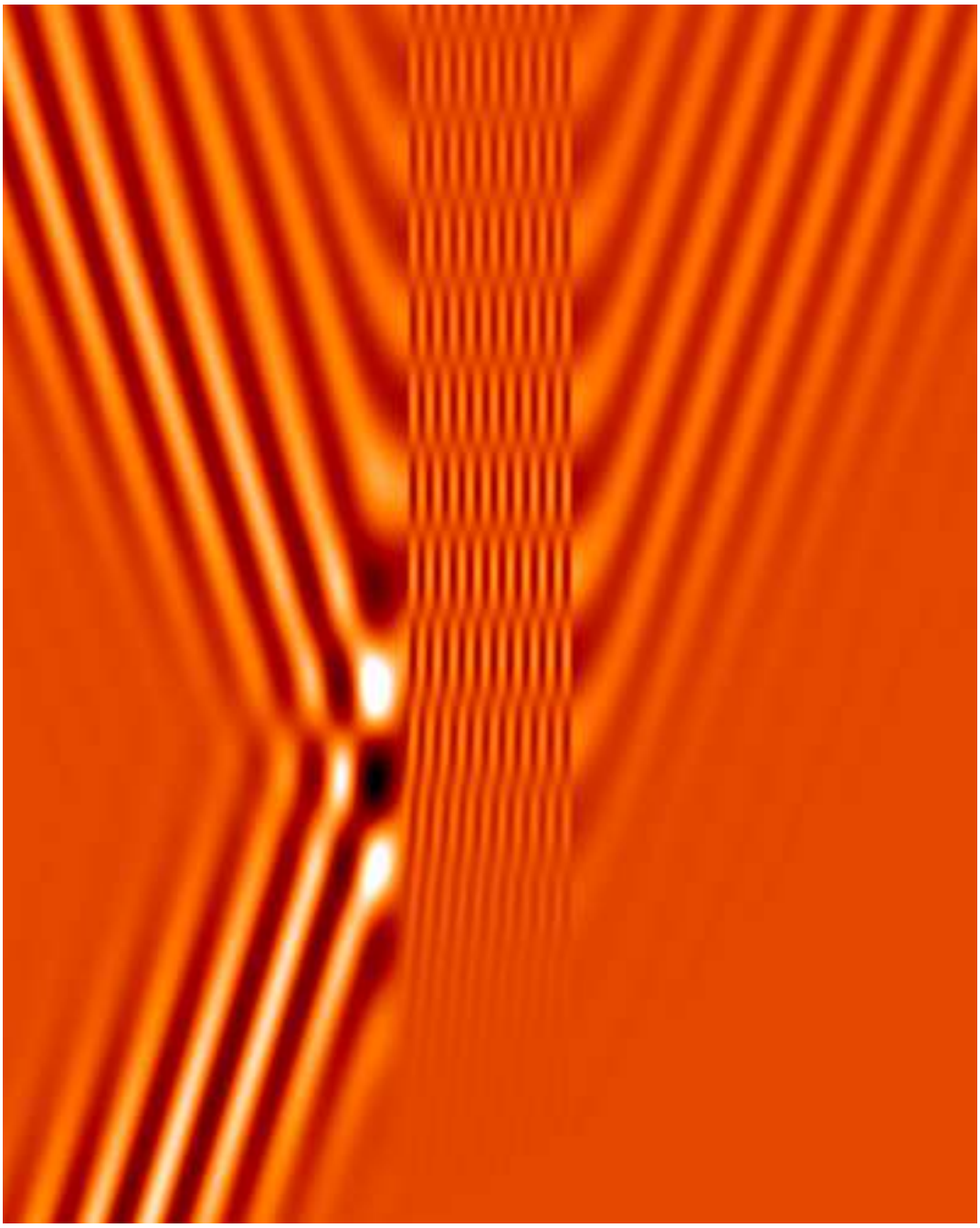


Figure 6.2: Space-time plot of the evolution of a Gaussian wavepacket scattering from a potential well with a small barrier surrounding it. A long-lived excitation of the well is evident in the persistent mode shown that runs vertically upwards along the center of the figure. The real part of the wavefunction is displayed.

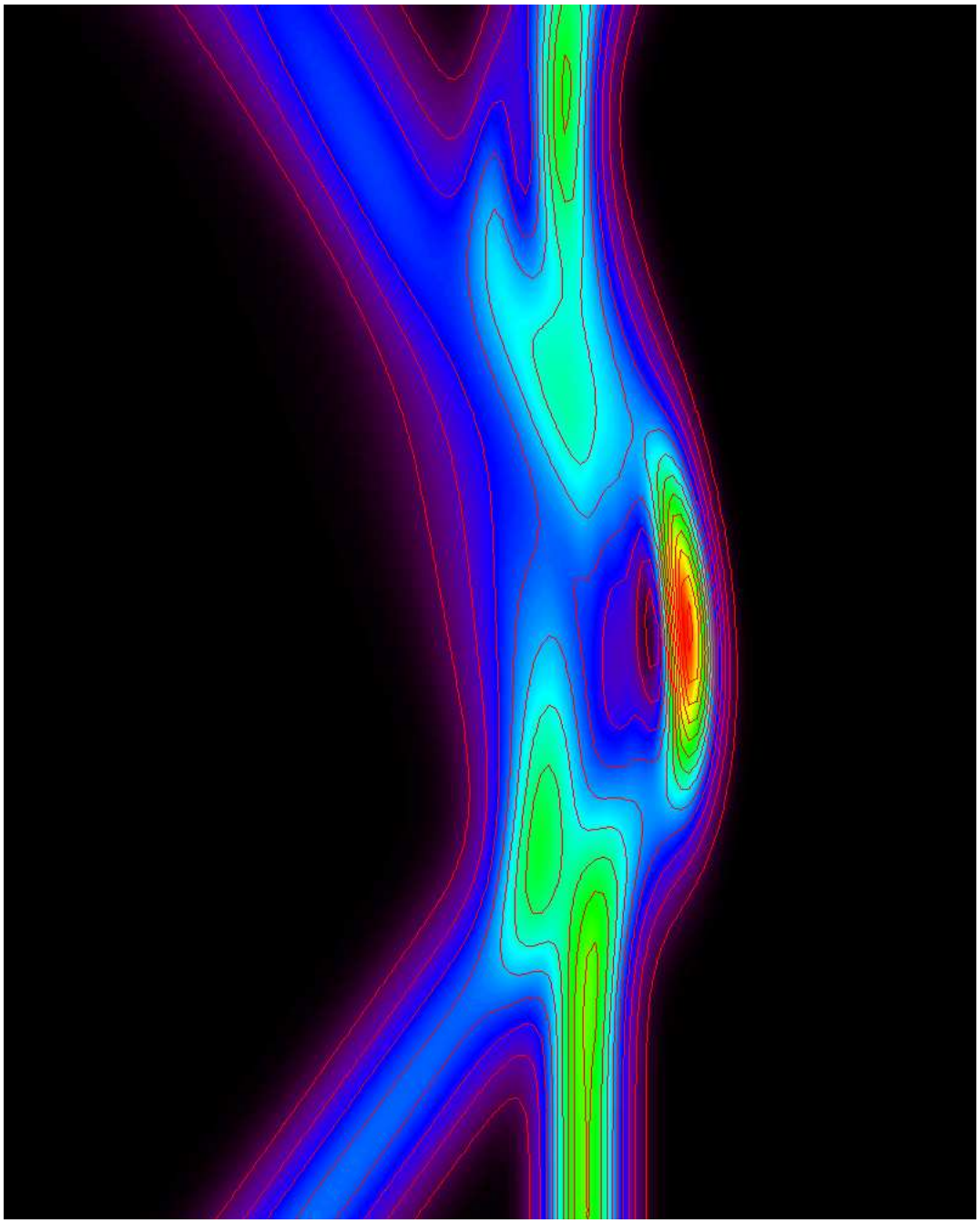


Figure 6.3: Space-time plot of the evolution of the collision of two Gaussian wavepackets. The first wavepacket is free and moves towards the second with an initial velocity. The second is a wavefunction corresponding to the ground state of a simple harmonic oscillator with the associated harmonic oscillator potential. The quantity displayed in this image is the sum of the marginal probability distributions, $\rho_1(x_1, t) + \rho_2(x_2, t)$. A countour of the same image is superimposed.

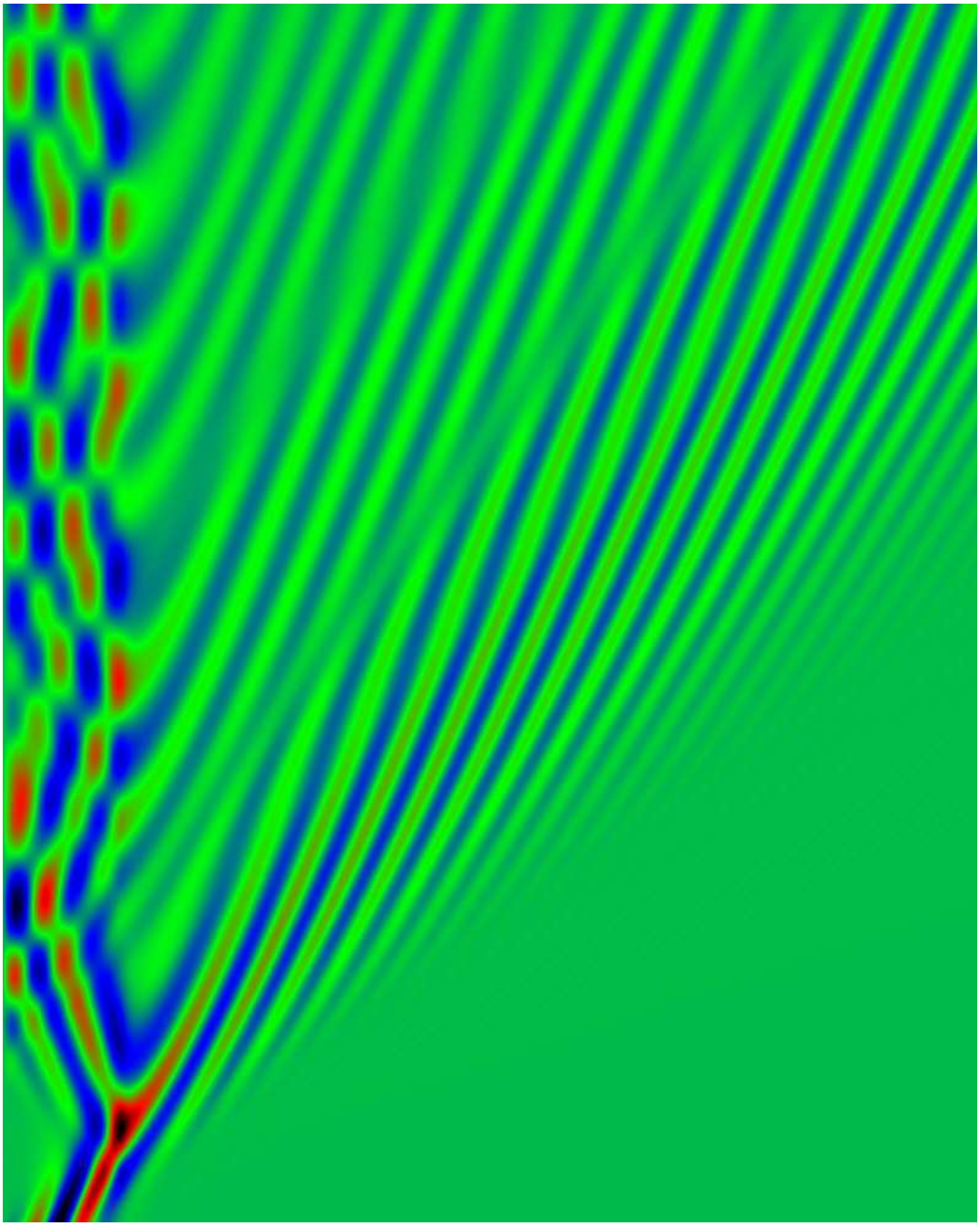


Figure 6.4: Space-time plot of a model of radioactive decay. A Gaussian wavepacket is "trapped" in a well on the left hand side of the figure. The wavepacket has an initial velocity to the right, collides with the wall, and part of the wavepacket tunnels through the confining potential. As time goes on, more of the wavepacket "leaks" out of the well. The real part of the wavefunction is displayed.

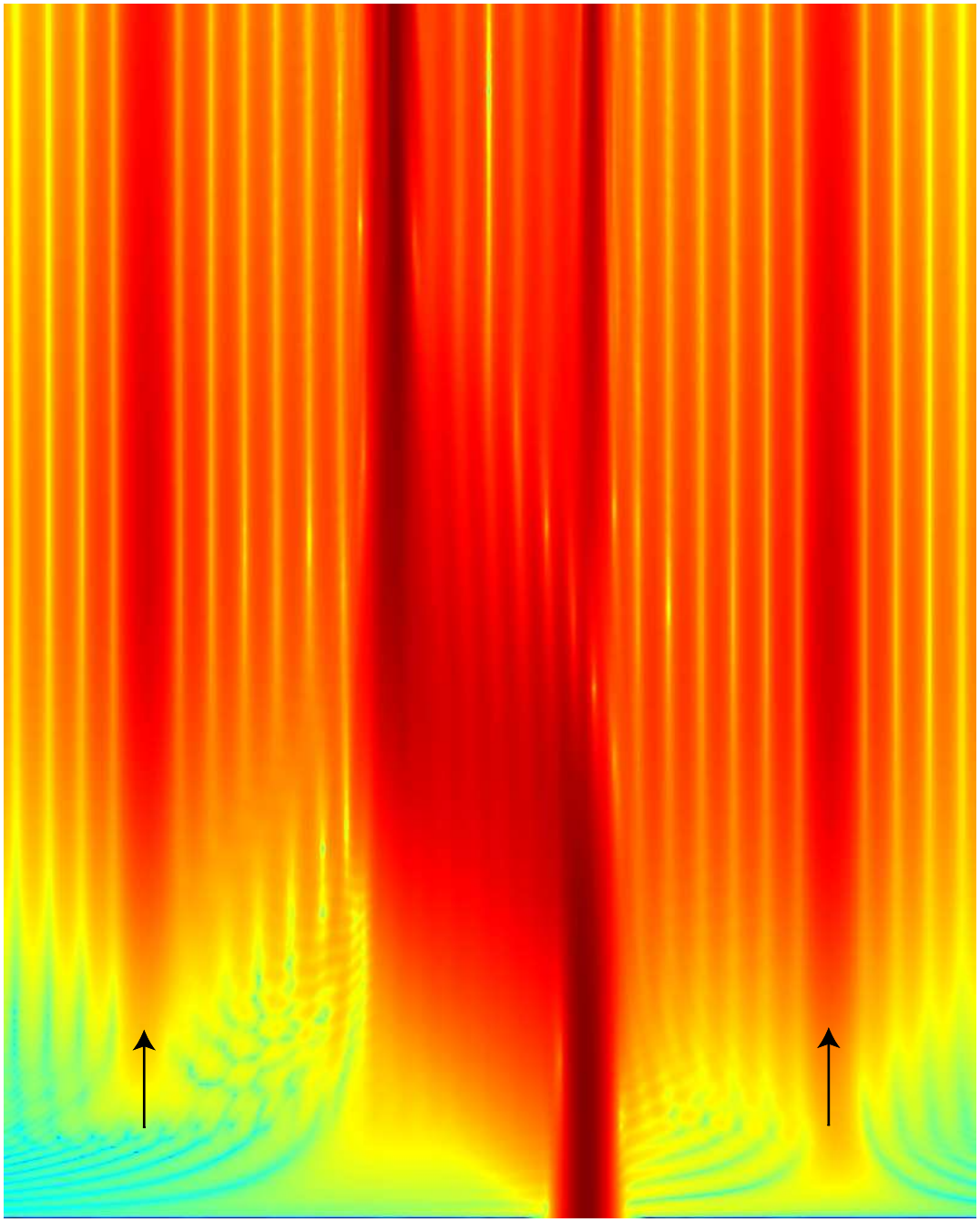


Figure 6.5: Momentum-time plot of a Gaussian interacting with a potential well with a small barrier surrounding it. The quantity displayed is $|\phi(k,T)|^2$. The horizontal axis is momentum and the vertical axis is time. Initially (i.e. at the bottom of the plot) there is a single momentum component due to the wavepacket moving to the right—this is the dark red band just right of center. During the interaction of the wavepacket with the well, a well-defined mode builds up and results in the higher momentum sidebands annotated with the arrows. See figure 4.10 for selected frames from the animated sequence in the position domain.

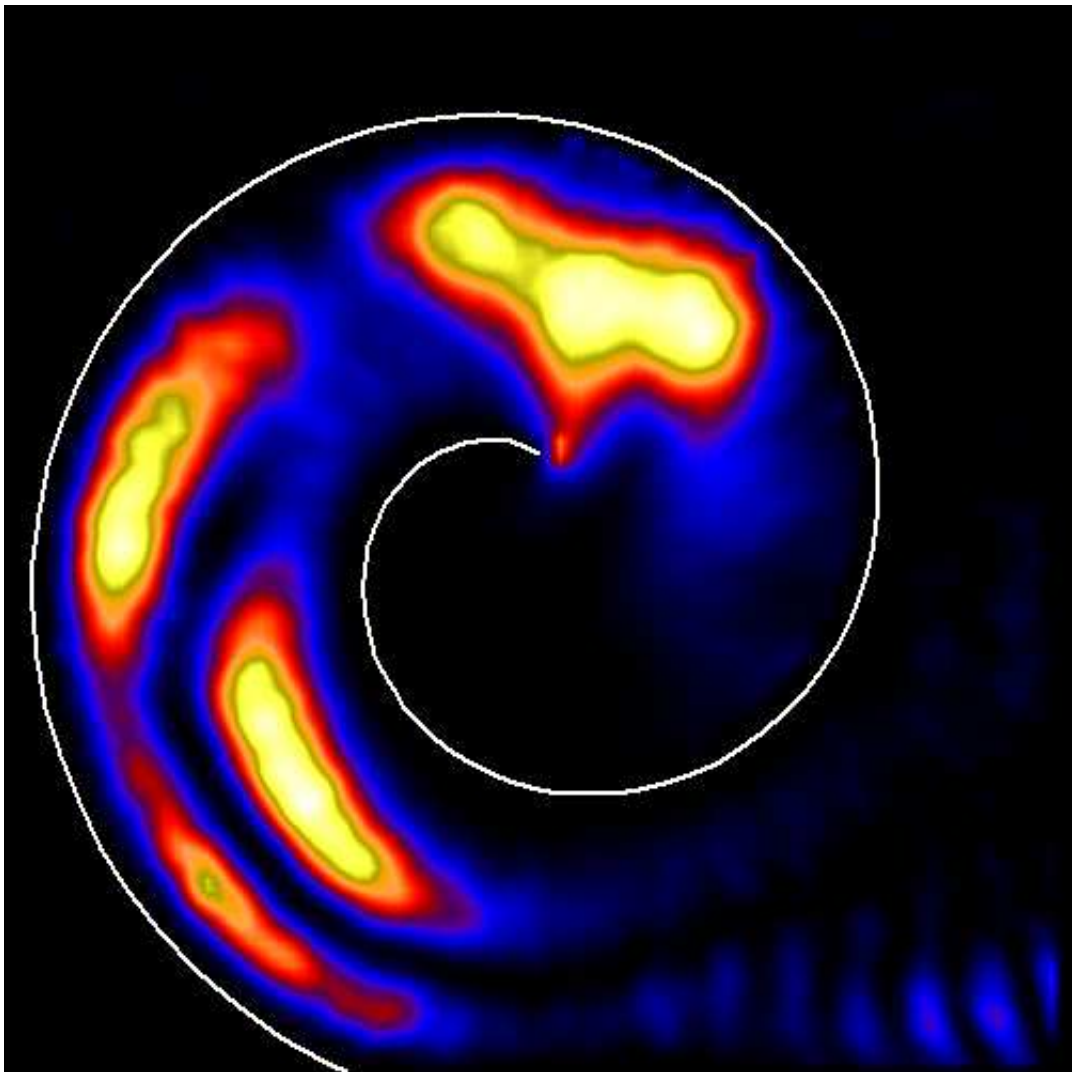


Figure 6.6: Image plot of a frame in the time-evolution of the angular momentum of a wavepacket, initially at rest, and evolving in a boundary defined by Archimedes' spiral. This image was created by plotting $|L_z \langle x, y|\psi\rangle (T)|$ at $T = 3.33$ where $L_z = -i\hbar \left(x\frac{\partial}{\partial y} - y\frac{\partial}{\partial x} \right)$.

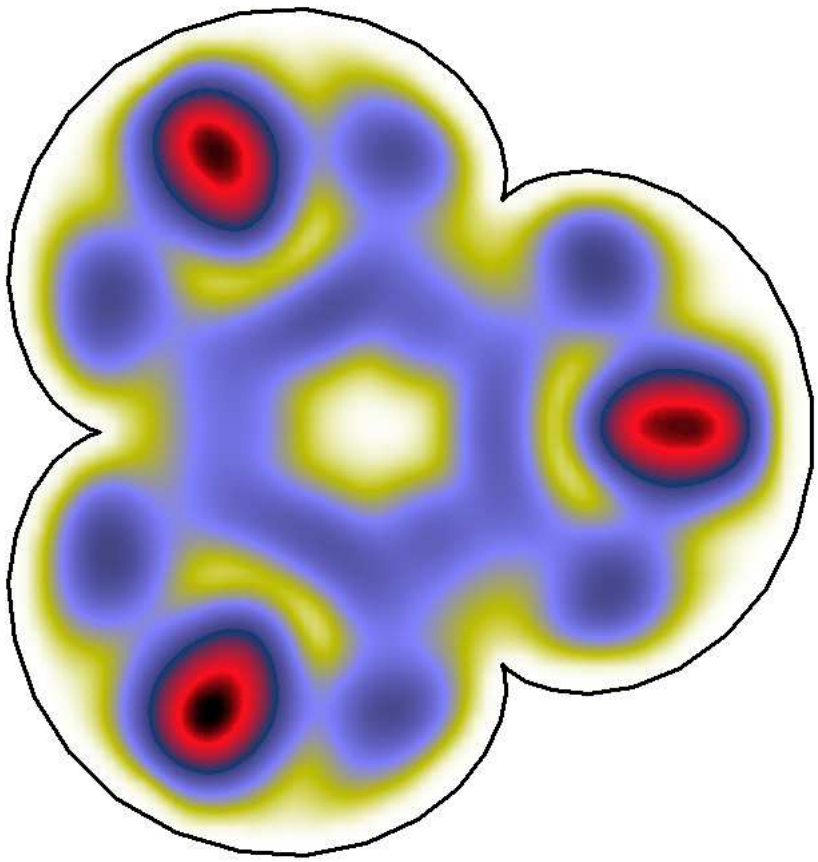


Figure 6.7: Image plot of a frame in the time-evolution of a wavepacket, initially at rest in the center of the figure, evolving in a potential well bounded by an epicycloid. The probability density, $|\langle x, y|\psi\rangle (T)|^2$ is displayed.

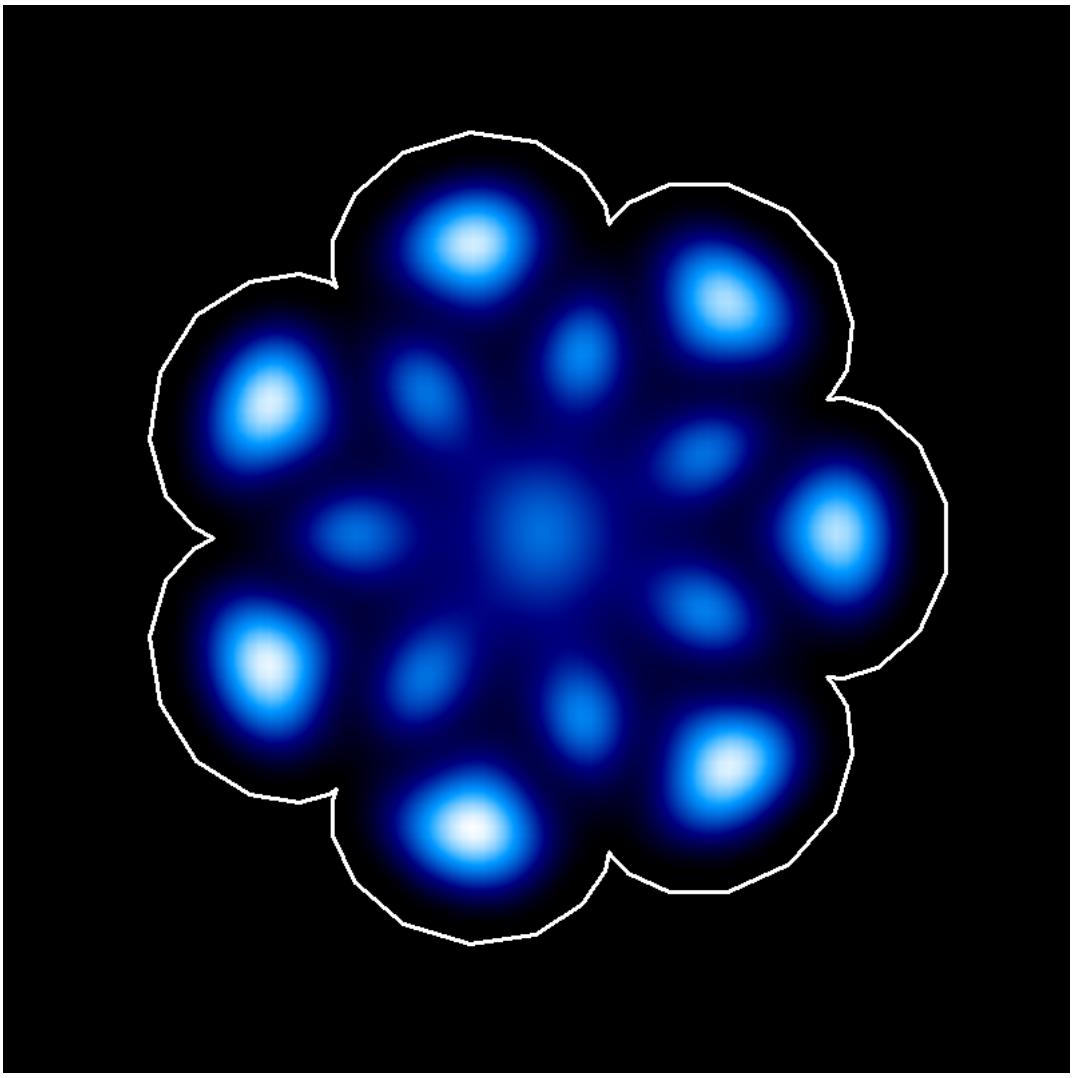


Figure 6.8: Image plot of a frame in the time-evolution of a wavepacket, initially at rest in the center of the figure, evolving in a potential well bounded by an epicycloid. The probability density, $|\langle x, y|\psi\rangle\,(T)\,|^2$ is displayed.

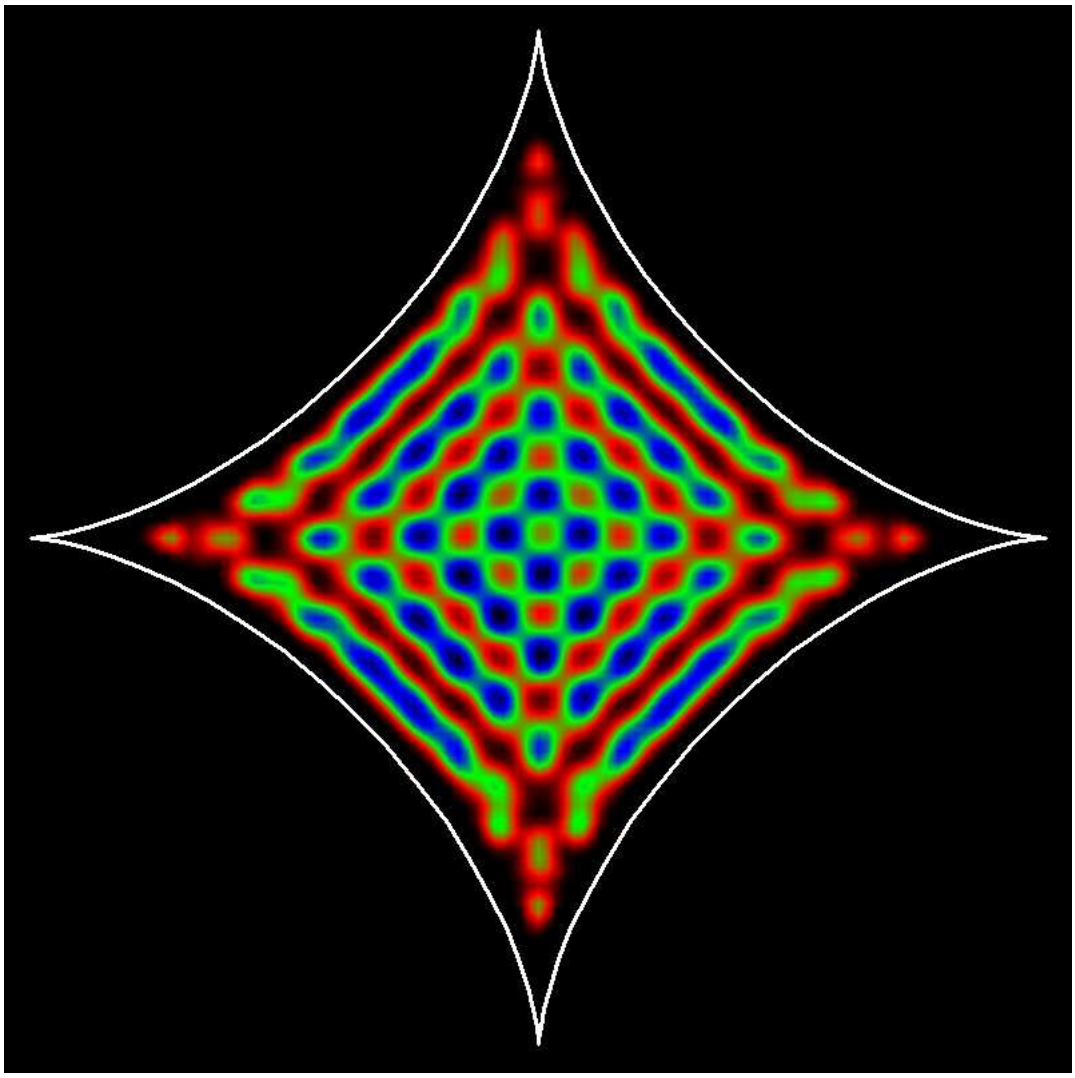


Figure 6.9: Image plot of a frame in the time-evolution of a wavepacket, initially at rest in the center of the figure, evolving in a potential well bounded by an astroid. The probability density, $|\langle x, y|\psi\rangle\,(T)\,|^2$ is displayed.

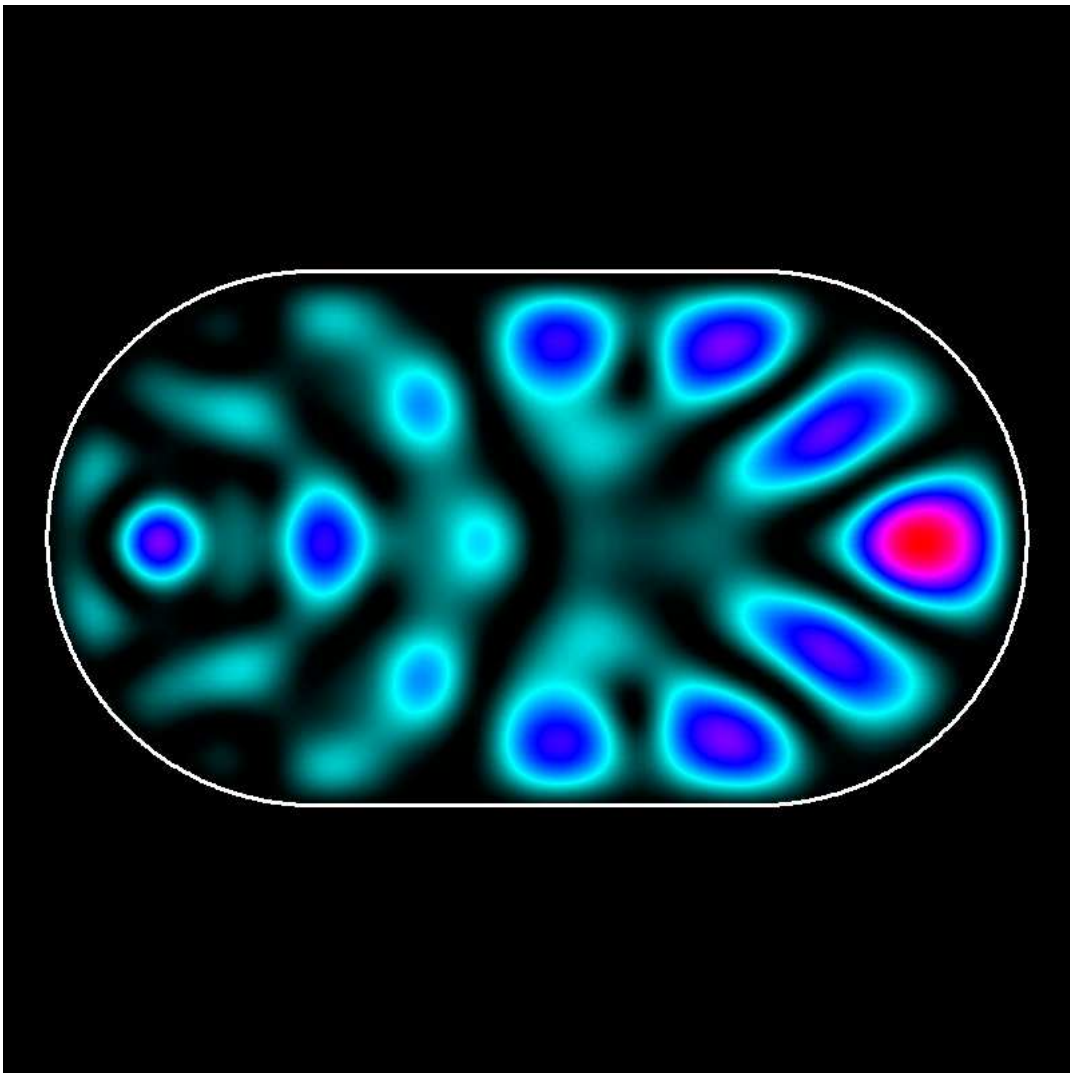


Figure 6.10: Image plot of a frame in the time-evolution of a wavepacket, initially placed at the center of the figure and moving to the left, evolving in a potential well bounded by a stadium curve. The probability density, $|\langle x, y|\psi\rangle (T)|^2$ is displayed.

Appendix A

The Wigner 3-j Symbol

As discussed in chapter 3 using the Wigner 3-j symbol can provide a compact way to represent a complicated expression involving coupling angular momentum. The implementation of the 3-j symbol in IDL is straightforward and based upon a set of equations motivated and presented in Edmond's book, *Angular Momentum in Quantum Mechanics* [14].

The 3-j symbol is given by

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = (-1)^{j_1-j_2-m_3}\,(2j_3+1)^{-1/2}\,(j_1\ m_1\ j_2\ m_2|j_1\ j_2\ j_3\ -m_3) \tag{A.1}$$

where the last term on the right-hand side of the equation above is given by

$$\begin{aligned}(j_1\ m_1\ j_2\ m_2|j_1\ j_2\ j\ m) \quad = \quad & \delta_{m_1+m_2,m}\left[\frac{(2j+1)\,(j_1-m_1)!}{(j_1+j_2+j+1)!\,(j_1-j_2+j)!}\right]^{1/2} \times \\ & \left[\frac{(j_2-m_2)!\,(j+m)!\,(j-m)!}{(-j_1+j_2+j)!\,(j_1+m_1)!\,(j_2+m_2)!}\right]^{1/2} \times \\ & \sum_s (-1)^{s+j_1-m_1}\,\frac{(j_1+m_1+s)!\,(j_2+j-m_1-s)!}{s!\,(j_1-m_1-s)\,(j-m-s)!\,(j_2-j+m_1+s)!}\end{aligned}$$

```
function wigner_threeJ,j1,j2,j3,m1,m2,m3
compile_opt idl2,hidden
jj1 = j1 & jj2 = j2 & jj3 = j3
mm1 = m1 & mm2 = m2 & mm3 = m3
t1 = (m1+m2+m3) ne 0
t2 = (j1+j2-j3) lt 0
t3 = (j1+j3-j2) lt 0
t4 = (j3+j2-j1) lt 0
t5 = 0
if (t1 or t2 or t3 or t4 or t5) then return,0.D0
m1 = double(m1) & m2 = double(m2) & m3 = -1.0*double(m3)
j1 = double(j1) & j2 = double(j2) & j3 = double(j3)
term1 = ds_delta(m1+m2,m3)
term2 = (2.0*j3+1)*nfact(j1+j2-j3)*nfact(j1-m1)*$
```

```
        nfact(j2-m2)*nfact(j3+m3)*nfact(j3-m3)
term2 = term2/(nfact(j1+j2+j3+1.0)*nfact(j1-j2+j3)*$
        nfact(-j1+j2+j3)*nfact(j1+m1)*nfact(j2+m2))
coef = term1*sqrt(term2)
sum = 0.0
nbig = 200
for s = 0,nbig-1 do begin
  z = double(s)
  den1 = j1-m1-z & den2 = j3-m3-z & den3 = j2-j3+m1+z
  if den1 ge 0.0 and den2 ge 0.0 and den3 ge 0.0 then begin
    num = nfact(j1+m1+z)*nfact(j2+j3-m1-z)
    den = nfact(z)*nfact(den1)*nfact(den2)*nfact(den3)
    sum = sum+((-1.0)^(z+j1-m1))*num/den
  endif
endfor
coef = coef*sum
newcoef = ((-1.0)^(j1-j2-m3))*(1.0/sqrt(2.0*j3+1))*coef
coef = newcoef
j1 = jj1 & j2 = jj2 & j3 = jj3
m1 = mm1 & m2 = mm2 & m3 = mm3
return,coef
end
```

This function requires the function `NFACT` which is the recursive factorial function listed below.

```
function nFact,n
; Recursive factorial function written by J.Copley
if (n gt 1) then begin
   z=double(n)*nFact(double(n-1))
endif else begin
   z=1
endelse
return,z
end
```

Appendix B

The Simple Harmonic Oscillator Eigenfunctions

The eigenfunctions in position-space for the simple harmonic oscillator are straightforward to implement in IDL. Since they involve Hermite polynomials, we first must implement a function that calculates the Hermite polynomial of any order. The recursion relationship for Hermite polynomials is

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x). \tag{B.1}$$

Knowing that $H_0(x) = 1$ and $H_1(x) = 2x$, we can use the recursion relationship above to calculate any higher order Hermite polynomial. The IDL function that computes the nth-order Hermite polynomial is `HERMITE.PRO` listed below. The input parameters are `n`, the order of the polynomial, and `x`, the point or points at which to evaluate the polynomial. It is useful to verify that the IDL function's output matches some of the higher order polynomials such as $H_2(x) = 4x^2 - 2$, $H_3(x) = 8x^3 - 12$, and $H_4(x) = 16x^4 - 48x^2 + 12$.

```
function hermite,n,x
compile_opt idl2,hidden
; This function returns the Hermite polynomial
; of order n and defined at argument values of
; x.  These are the "Physicists' Hermite Polynomials"
nx = n_elements(x)
m = (1+n) > 2
h = fltarr(m,nx)
h[0,*] = fltarr(nx) + 1.0
h[1,*] = 2.*x
if n le 1 then return,reform(h[n,*])
for j = 1,n-1 do h[j+1,*] = 2.*x*h[j,*] - 2.*j*h[j-1,*]
return,reform(h[m-1,*])
end
```

The eigenfunctions for the simple harmonic oscillator are given by

$$\langle x|n\rangle = \left(\frac{m\omega_0}{\pi}\right)^{1/4} \frac{1}{\sqrt{2^n n!}} e^{-\frac{1}{2}m\omega_0 x^2} H_n\left(x\sqrt{m\omega_0}\right) \tag{B.2}$$

where $\hbar\omega_0/2$ is the ground state energy for the oscillator and m is the mass of the particle. This expression can be converted into an IDL function using the `HERMITE.PRO` function. The function is called `SHO_EIGENSTATES.PRO`. Required input parameters are the oscillator level `n`, the spatial coordinates at which to evaluate the function `x`, the mass of the particle `m`, and the oscillator frequency `wo`.

```
function sho_eigenstates,n,x,m,wo
compile_opt idl2,hidden
b = sqrt(m*wo)
wf = ((b^2/!pi)^(0.25))*(1./sqrt((2.^n)*factorial(n)))*$
  (exp(-0.5*(b*x)^2))*hermite(n,b*x)
; Note that you must return a complex result if this
; is to be used by a routine that solves the Schrodinger
; equation.
return,complex(wf,0d*wf)
end
```

Answers to Selected Exercises

Answer (ex. 2) — The fifth eigenvalue changes by less than 1% for values of N_x greater than about 95.

Answer (ex. 6) — The wavefunction components move faster in time when compared with the motion of the probability density. The reason why the wavefunction components move faster is based on the fact that increasing the barrier height lowers the transition energy between the lowest two eigenstates, $\Delta E = E_1 - E_0$, but increases the absolute values of the ground state E_0 and first excited state E_1 energies. As given in equations 2.17 and 2.18, we note that the time-dependence of the wavefunction depends on E_0 and E_1 and the time-dependence of the probability density depends on the difference between these two, $\Delta E = E_1 - E_0$. Therefore, as the barrier increases, the wavefunction oscillations increase in frequency but the probability density decreases in frequency.

Answer (ex. 9) — Since the potential energy can be expanded

$$\begin{aligned} V(\theta) &= \frac{V_3}{2}(1-\cos 3\theta) \\ &= \frac{V_3}{2}\left(1-\left(1-\frac{1}{2}(3\theta)^2\cdots\right)\right) \\ &\simeq \frac{1}{2}\left(\frac{9V_3}{2}\right)\theta^2 \end{aligned}$$

then the approximate Schrödinger equation can be written as

$$-\frac{\hbar^2}{2I}\frac{\partial^2}{\partial\theta^2}\langle\theta|\psi\rangle + \frac{1}{2}\left(\frac{9V_3}{2}\right)\theta^2\langle\theta|\psi\rangle = E\langle\theta|\psi\rangle.$$

Comparing this to the Schrödinger equation for a simple harmonic oscillator

$$-\frac{\hbar^2}{2m_{eq}}\frac{\partial^2}{\partial x^2}\langle x|\psi\rangle + \frac{1}{2}k_{eq}x^2\langle x|\psi\rangle = E\langle x|\psi\rangle \tag{3.4}$$

we have for the equivalent mass $m_{eq} = I$ and the equivalent spring constant $k_{eq} = 9V_3/2$. Since the oscillation frequency for this simple harmonic oscillator is $\omega_0 = \sqrt{k_{eq}/m_{eq}}$ then the ground-state energy is given approximately by $E_0 = \hbar\sqrt{\frac{9V_3}{2I}}$

Answer (ex. 10) — The matrix element $\langle 3m+l|\cos 6\theta|3n+l\rangle$ is $\frac{1}{4}\left(\delta_{m,n+2}+\delta_{m,n-2}\right)$. The final Hamiltonian is the same as equation 3.13 with the following term added

$$\frac{V_6}{4}\left(2\delta_{m,n}-\delta_{m,n+2}-\delta_{m,n-2}\right).$$

Answer (ex. 11) — The lowest eigenvalue decreases by a factor of $\sqrt{2}$ when you double the moment of inertia. See equation 3.18.

Answer (ex. 13) — The tunneling energy ΔE_{01} depends approximately exponentially on the barrier height ($\Delta E_{01}\sim\exp\left(-cV_3\right)$) for large V_3, making it an excellent way to extract the barrier height from a tunneling measurement.

Answer (ex. 18) — First we note that the potential can be expressed in terms of the spherical harmonics as

$$V\left(\theta\right)=4V_4\left(\frac{2}{15}+\frac{2}{21}\sqrt{\frac{4\pi}{5}}Y_{2,0}-\frac{8}{25}\sqrt{\frac{4\pi}{9}}Y_{4,0}\right). \tag{3.36}$$

The final result is

$$\begin{aligned}
H_{\ell,\ell',m,m'} \quad = \quad & \left(B\ell\left(\ell+1\right)+\frac{8V_4}{15}\right)\delta_{\ell,\ell'}\delta_{m,m'}+ \\
& \frac{8V_4}{21}\sqrt{\frac{4\pi}{5}}\left(-1\right)^{m'}\left(\frac{5\left(2\ell+1\right)\left(2\ell'+1\right)}{4\pi}\right)^{1/2}\begin{pmatrix}\ell' & 2 & \ell\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\ell' & 2 & \ell\\ -m' & 0 & m\end{pmatrix}- \\
& \frac{32V_4}{35}\sqrt{\frac{4\pi}{9}}\left(-1\right)^{m'}\left(\frac{9\left(2\ell+1\right)\left(2\ell'+1\right)}{4\pi}\right)^{1/2}\begin{pmatrix}\ell' & 4 & \ell\\ 0 & 0 & 0\end{pmatrix}\begin{pmatrix}\ell' & 4 & \ell\\ -m' & 0 & m\end{pmatrix}
\end{aligned}$$

Answer (ex. 20) — $-43.5\leq x(\text{Å})\leq 14.5$ and $0\leq \text{t(ps)}\leq 3.26$.

Answer (ex. 27) — As the width of the wavepacket increases, the penetration depth gets smaller. In fact, in the limit of a plane wave ($\sigma\to\infty$), there will be no penetration.

Answer (ex. 30) — $x=\frac{\hbar}{\sqrt{m\alpha}}X$ and $y=\frac{\hbar}{\sqrt{m\alpha}}Y$.

Answer (ex. 31) — $a=2.04$Å, $w=1.02$Å, and $s=4.08$Å.

Bibliography

[1] IDL stands for the Interactive Data Language, a product from ITT Visual Information Solutions, `http://www.ittvis.com`.

[2] Richard W. Robinett, *Quantum Mechanics: Classical Results, Modern Systems, and Visualized Examples*, Oxford University Press (1997).

[3] Sigmund Brandt and Hans Dieter Dahmen, *The Picture Book of Quantum Mechanics*, Springer-Verlag (2001).

[4] John S. Townsend, *A Modern Approach to Quantum Mechanics*, McGraw-Hill, Inc. (1992).

[5] MATLAB is a product from the Mathworks, `http://www.mathworks.com`.

[6] R.M.Dimeo, *Application Development in IDL I*, available for download at `ftp://ftp.ncnr.nist.gov/pub/staff/dimeo/intro_to_idl.pdf`.

[7] David Fanning, *IDL Programming Techniques, 2nd Edition*, available from `http://www.dfanning.com`.

[8] `http://www.ncnr.nist.gov/staff/dimeo`.

[9] A. Goldberg, H.M. Schey, and J.L. Schwartz, *Computer-Generated Motion Pictures of One-Dimensional Quantum-Mechanical Transmission and Reflection Phenomena*, Am. J. Phys. **35**, 177 (1967).

[10] `http://cow.physics.wisc.edu/~craigm/idl/idl.html`

[11] G.E. Kimball and G.H. Shortley, *The Numerical Solution of Schrödinger equation's Equation*, Phys. Rev. **45**, 815 (1934).

[12] B.J. Riel, *An Introduction to Self-Assembled Quantum Dots*, Am. J. Phys. **76**, 750 (2008).

[13] D.L. Kaufman, I. Kosztin, and K. Schulten, *Expansion Method for Stationary States of Quantum Billiards*, Am. J. Phys. **67**, 133 (1999).

[14] A.R. Edmonds, *Angular Momentum in Quantum Mechanics*, Princeton University Press (1996).

[15] A. Goldberg, H.M. Schey, J.L. Schwartz and R. Whiddon, *Quantum Physics Series, Films 1-10*, reviewed by K. Gottfried in Am. J. Phys. **46**, 315 (1978).

[16] B.E. Warren, *X-Ray Diffraction*, Dover Publications (1990).

[17] A. Askar and A.S. Cakmak, *Explicit integration method for the time-dependent Schrodinger equation for collision problems*, J. Chem. Phys. **68**, 2794 (1978).

Index

1D harmonic oscillator, 16
2D Hamiltonian, 4, 79
2D isotropic harmonic oscillator, 26
3-fold rotor Hamiltonian, 37
3-fold rotor matrix elements, 38

Archimedes' spiral, 88

billiard balls, 95
Bragg's Law, 70

cardioid potential, 32

discrete variable approximation, 2, 13, 24

Expansion Method, 16, 27

free-rotor eigenfunctions, 38

harmonic oscillator, 98

probability current, 3

quantum billiards, 27, 85

radioactive decay, 78
repulsive interaction, 95

Schrödinger equation, 3, 13, 54
stadium potential, 27

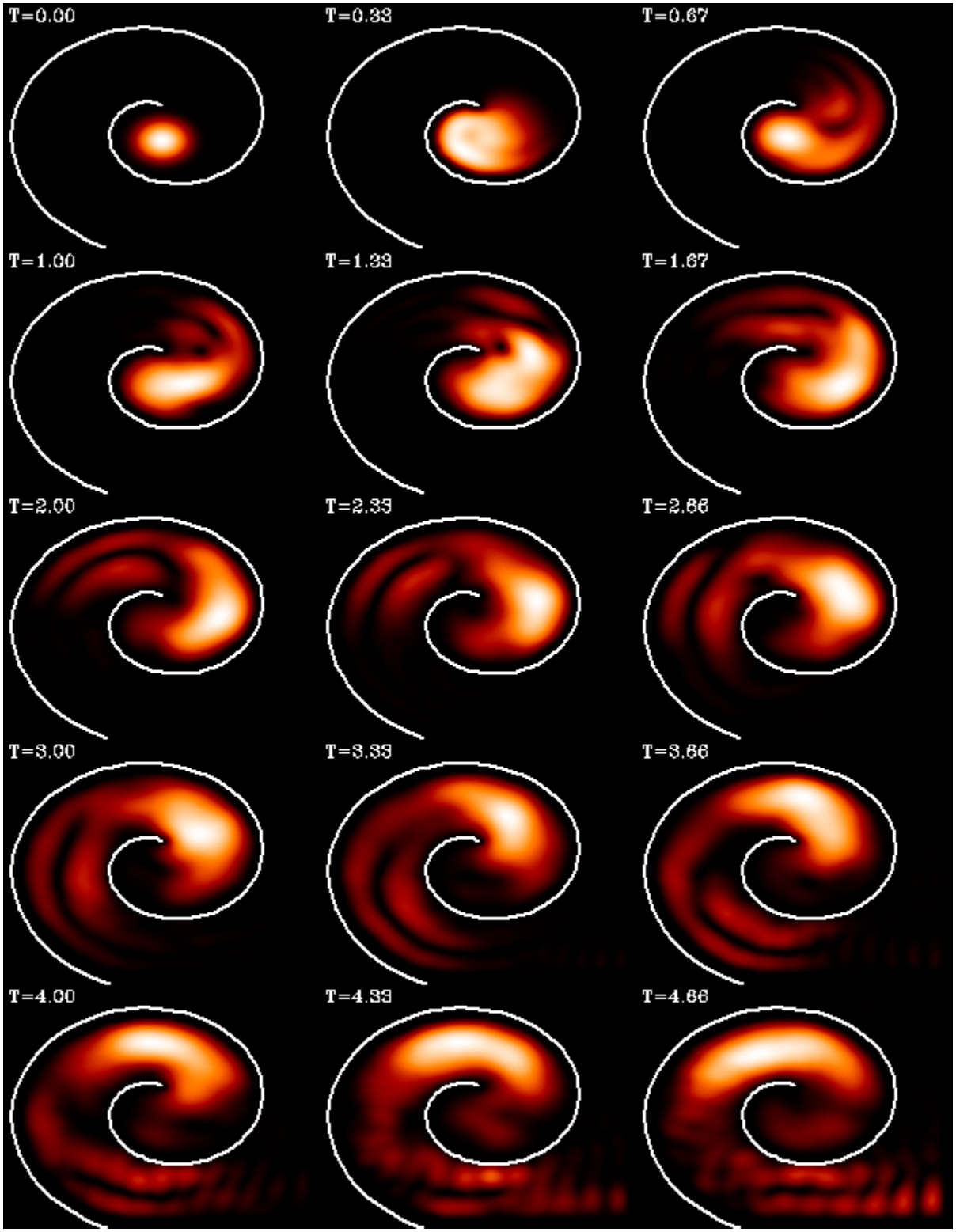
transition animation, 21, 23
two particle Hamiltonian, 4

virtual/metastable states, 73

wavepacket collisions, 92
Wigner 3-j symbol, 48, 115

About the Author

Dr. Robert Dimeo is the Deputy Director for the Center for Neutron Research at the National Institute of Standards and Technology and has worked at the NCNR since 1999. He has a Ph.D. in Physics and an M.S. in Electrical Engineering from the Pennsylvania State University. He has taught numerous courses in IDL programming, specifically for analysis applications in neutron scattering research. He lives in Germantown, MD with his wife and two children.



Front cover: Branches are the eigenvalues calculated from a numerical solution of the Mathieu differential equation for a three-fold symmetric potential used to model hindered methyl group dynamics. The image was created by finding the eigenvalues of a system with both a three-fold and a six-fold term, varying the amplitudes of the barrier terms and adding the transitions between the first and second eigenvalues and the fourth and fifth eigenvalues.

Back cover: The images displayed here are the frames from the time evolution of a Gaussian wavepacket in two dimensions initially at rest constrained to move within a potential boundary given by an Archimedes' spiral. Though the wavepacket is initially at rest, the spreading and interaction with the boundary cause the wavepacket to develop angular momentum and the center-of-mass of the wavepacket moves outward.