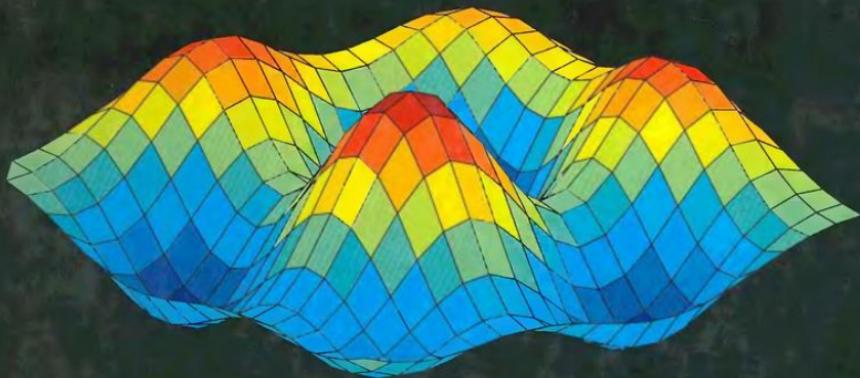


Computational Physics

SECOND EDITION



Nicholas J. Giordano
Hisao Nakanishi

Computational Physics

Second Edition

Nicholas J. Giordano

Department of Physics
Purdue University

Hisao Nakanishi

Department of Physics
Purdue University



Progressive International Agencies (Pvt) Limited

Karachi
Ph: 4525544-6

Islamabad
Ph: 2825906

Lahore
Ph: 6372806



Upper Saddle River, NJ 07458

Library of Congress Cataloging-in-Publication Data

Giordano, Nicholas J.

Computational physics / Nicholas J. Giordano, Hisao Nakanishi -- 2nd ed
p cm

Includes index

ISBN 0-13-146990-8

I. Physics--Data processing. I. Nakanishi, Hisao. II. Title

QC20.7 E4G56 2006

530.158--dc22

2005049248

Associate Editor: Christian Botting

Senior Editor: Erik Fahlgren

Editorial Assistant: Jessica Berta

Executive Managing Editor: Kathleen Schiaparelli

Assistant Managing Editor: Beth Sweeten

Production Editor: Edward Thomas

Manufacturing Manager: Alexis Heydt-Long

Manufacturing Buyer: Alan Fischer

Director of Marketing, Science: Linda Taft-MacKinnon

Senior Marketing Manager: Shari Meffert

Marketing Assistant: Laura Rath

Copy Editor: Barbara Booth

Production Assistant: Nancy Bauer

Art Director: Jayne Conte

Cover Designer: Bruce Kenselaar

© 2006, 1997 Pearson Education, Inc.

Pearson Prentice Hall

Pearson Education, Inc.

Upper Saddle River, NJ 0745

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and test of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Pearson Prentice Hall™ is a trademark of Pearson Education, Inc.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-146990-8

Pearson Education Ltd., London

Pearson Education Australia Pty., Limited, Sydney

Pearson Education Singapore, Pte. Ltd.

Pearson Education North Asia Ltd., Hong Kong

Pearson Education Canada, Ltd., Toronto

Pearson Educación de Mexico, S.A. de C.V.

Pearson Education—Japan, Tokyo

Pearson Education Malaysia, Pte. Ltd.

To my parents.
Nicholas Giordano

To Atsuko.
Hisao Nakanishi

Contents

Preface	ix
About the Authors	xii
1 A First Numerical Problem	1
1.1 Radioactive Decay	1
1.2 A Numerical Approach	2
1.3 Design and Construction of a Working Program: Codes and Pseudocodes	3
1.4 Testing Your Program	11
1.5 Numerical Considerations	12
1.6 Programming Guidelines and Philosophy	14
2 Realistic Projectile Motion	18
2.1 Bicycle Racing: The Effect of Air Resistance	18
2.2 Projectile Motion. The Trajectory of a Cannon Shell	25
2.3 Baseball: Motion of a Batted Ball	31
2.4 Throwing a Baseball: The Effects of Spin	36
2.5 Golf	44
3 Oscillatory Motion and Chaos	48
3.1 Simple Harmonic Motion	48
3.2 Making the Pendulum More Interesting: Adding Dissipation, Nonlinearity, and a Driving Force	54
3.3 Chaos in the Driven Nonlinear Pendulum	58
3.4 Routes to Chaos: Period Doubling	66
3.5 The Logistic Map: Why the Period Doubles	70
3.6 The Lorenz Model	75
3.7 The Billiard Problem	82
3.8 Behavior in the Frequency Domain: Chaos and Noise	88
4 The Solar System	94
4.1 Kepler's Laws	94
4.2 The Inverse-Square Law and the Stability of Planetary Orbits	101
4.3 Precession of the Perihelion of Mercury	107
4.4 The Three-Body Problem and the Effect of Jupiter on Earth	113
4.5 Resonances in the Solar System: Kirkwood Gaps and Planetary Rings	118
4.6 Chaotic Tumbling of Hyperion	123

5 Potentials and Fields	129
5.1 Electric Potentials and Fields: Laplace's Equation	129
5.2 Potentials and Fields Near Electric Charges	143
5.3 Magnetic Field Produced by a Current	148
5.4 Magnetic Field of a Solenoid: Inside and Out	151
6 Waves	156
6.1 Waves: The Ideal Case	156
6.2 Frequency Spectrum of Waves on a String	165
6.3 Motion of a (Somewhat) Realistic String	169
6.4 Waves on a String (Again): Spectral Methods	174
7 Random Systems	181
7.1 Why Perform Simulations of Random Processes?	181
7.2 Random Walks	183
7.3 Self-Avoiding Walks	188
7.4 Random Walks and Diffusion	195
7.5 Diffusion, Entropy, and the Arrow of Time	201
7.6 Cluster Growth Models	206
7.7 Fractal Dimensionalities of Curves	212
7.8 Percolation	218
7.9 Diffusion on Fractals	229
8 Statistical Mechanics, Phase Transitions, and the Ising Model	235
8.1 The Ising Model and Statistical Mechanics	235
8.2 Mean Field Theory	239
8.3 The Monte Carlo Method	244
8.4 The Ising Model and Second-Order Phase Transitions	246
8.5 First-Order Phase Transitions	259
8.6 Scaling	264
9 Molecular Dynamics	270
9.1 Introduction to the Method: Properties of a Dilute Gas	270
9.2 The Melting Transition	285
9.3 Equipartition and the Fermi-Pasta-Ulam Problem	294
10 Quantum Mechanics	303
10.1 Time-Independent Schrödinger Equation: Some Preliminaries	303
10.2 One Dimension: Shooting and Matching Methods	307
10.3 A Matrix Approach	323
10.4 A Variational Approach	326
10.5 Time-Dependent Schrödinger Equation: Direct Solutions	333
10.6 Time-Dependent Schrödinger Equation in Two Dimensions	345
10.7 Spectral Methods	349

11 Vibrations, Waves, and the Physics of Musical Instruments	357
11.1 Plucking a String: Simulating a Guitar	357
11.2 Striking a String: Pianos and Hammers	362
11.3 Exciting a Vibrating System with Friction: Violins and Bows	367
11.4 Vibrations of a Membrane: Normal Modes and Eigenvalue Problems	372
11.5 Generation of Sound	382
12 Interdisciplinary Topics	389
12.1 Protein Folding	389
12.2 Earthquakes and Self-Organized Criticality	405
12.3 Neural Networks and the Brain	418
12.4 Real Neurons and Action Potentials	436
12.5 Cellular Automata	445

APPENDICES

A Ordinary Differential Equations with Initial Values	456
A.1 First-Order, Ordinary Differential Equations	456
A.2 Second-Order, Ordinary Differential Equations	460
A.3 Centered Difference Methods	464
A.4 Summary	467
B Root Finding and Optimization	469
B.1 Root Finding	469
B.2 Direct Optimization	472
B.3 Stochastic Optimization	473
C The Fourier Transform	479
C.1 Theoretical Background	479
C.2 Discrete Fourier Transform	481
C.3 Fast Fourier Transform (FFT)	483
C.4 Examples: Sampling Interval and Number of Data Points	486
C.5 Examples: Aliasing	488
C.6 Power Spectrum	490
D Fitting Data to a Function	493
D.1 Introduction	493
D.2 Method of Least Squares: Linear Regression for Two Variables	494
D.3 Method of Least Squares: More General Cases	497
E Numerical Integration	500
E.1 Motivation	500
E.2 Newton-Cotes Methods: Using Discrete Panels to Approximate an Integral	500
E.3 Gaussian Quadrature: Beyond Classic Methods of Numerical Integration	504

E.4 Monte Carlo Integration	506
F Generation of Random Numbers	512
F.1 Linear Congruential Generators	512
F.2 Nonuniform Random Numbers	516
G Statistical Tests of Hypotheses	520
G.1 Central Limit Theorem and the χ^2 Distribution	521
G.2 χ^2 Test of a Hypothesis	523
H Solving Linear Systems	527
H.1 Solving $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, $\mathbf{b} \neq 0$	528
H.1.1 Gaussian Elimination	528
H.1.2 Gauss-Jordan elimination	530
H.1.3 LU decomposition	531
H.1.4 Relaxational method	533
H.2 Eigenvalues and Eigenfunctions	535
H.2.1 Approximate Solution of Eigensystems	537
Index	541

Preface

This book is based on the Computational Physics course that has been developed and taught by the authors at Purdue for more than a decade. The goal of this course is to introduce students to some basic numerical techniques and then apply these techniques to a number of *modern* topics, that is, problems of current interest to physicists. Students with some experience in differential and integral calculus can readily grasp rather sophisticated computational techniques. These students can use computers as tools with which to attack and solve problems that they would not ordinarily encounter in the undergraduate curriculum. We have used this approach to try to convey the excitement of physics, with a variety of problems of current interest.

While there are many texts with the terms “computers” and “physics” in their titles, most of the books in this area tend to focus heavily on numerical methods rather than physics. Since our goal is to teach a course on *physics*, rather than numerical methods, these books are not a good match for our course. While there are a few books that emphasize the physics that can be done with numerical methods, they are either too advanced for use by undergraduates, or (more commonly) they fail to deal with the types of problems that can profit most from a numerical approach. In too many cases they tend to simply treat the standard problems, which are already dealt with in many traditional texts using analytic methods. Hence the basic motivation for creating this book.

The material for our book is taken from a wide variety of “primary” sources, as will become clear from the references at the end of each chapter. In many cases we started with papers from the recent physics literature and then distilled them to produce problems suitable for an undergraduate class. While it is necessary for this book to introduce a variety of numerical methods of interest to physicists, the overriding emphasis is on the physics that can be done with these methods. The majority of the problems described in this book cannot be solved with purely analytic techniques. A computational approach is required in most cases, and we have tried to use the computer to make the *physics* as clear and as interesting as possible.

As readers scan through this new edition, they will notice a number of changes. Perhaps the most important is that there are now two authors. Besides simply sharing the workload of preparing the new edition, this additional expertise has allowed us to add many new topics (and improve old ones!). In fact, we have added much new material on subjects ranging from diffusion on fractals and cellular automata, to the physics of musical instruments (a new chapter) and a new algorithm for doing time-dependent quantum mechanical problems.

This book has also been reorganized in several ways. The first edition gave programming examples in the *True Basic* language. Now, in this new edition the reliance on the programming language *True Basic* has been removed. We recognize that present-day students will likely be using many different languages, so we have chosen to employ a very general pseudocode to illustrate the algorithms. This

pseudocode can easily be translated into virtually any language, and hence support the work of students in a wide variety of programming languages. However, for those students who prefer to see programs or routines in a “real” programming language, the *True Basic* programs from the first edition are still available at our website,¹ www.physics.purdue.edu/~giordano/comp-phys.html. Our plan is to add more programs, in other languages, to this website in the future, so that it can serve as a useful resource for students (and teachers). Another change in this new edition is that we have moved much of the discussion of the algorithms themselves into the appendices, and have added considerably to the depth and rigor of these discussions. It is our hope that the appendices can serve as reference material for students as they work their way through the physics that is covered in the chapters. This separation also allows the chapters to focus even more on the physics of the various topics.

How to Use This Book.

The first edition contained more than could easily be covered in a single course. The second edition contains even more, so it is clearly not possible to cover it all in one semester. The first few chapters rely mainly on elementary mechanics, and can be appreciated with a background at the freshman level. This material can be augmented with selected topics from later chapters (such as on random processes in Chapter 7 and molecular dynamics in Chapter 9) to produce a full-semester course. There is also ample material for a course aimed at advanced undergraduates and beginning graduate students. For example, the material on random processes (Chapter 7) and phase transitions (Chapter 8) can be added to the work on quantum mechanics (Chapter 10) to fill most of a semester at this level. A third way to use the material in this book is for an interdisciplinary course, in which case the chapters on waves (Chapter 6), musical instruments (Chapter 11), and interdisciplinary topics (Chapter 12) could form the core of a course.

The first edition would not have been possible without the help of many people, and we would like to thank them again. The support of Arnold Tubbis and the Department of Physics at Purdue, along with that of the National Science Foundation, made our course, and hence this book, possible. Many graduate students helped us teach early versions of this course, including Miguel Castro, Chris Parks, Jan Spitz, Stuart Burnett, Todd Jacobs, and Dan Lawrence. Of course, the undergraduate students who have willingly submitted to the course have provided much useful feedback; there are too many to mention them all here, although Mike Pennington deserves a special thanks. Many colleagues have provided essential advice and encouragement on the manuscript, including Todd Jacobs, Mark Haugan, Paul Muzikar, along with the reviewers Wolfgang Christian, Alejandro Garcia, Jan Tobochnik, and Rodney L. Varley, who were very polite and constructive. The support of the first edition editors Ray Henderson and Alison Reeves was much valued, while the final impetus to actually begin this book was provided by the well-timed encouragement of Earl Prohofsky and Betsy Beasley.

¹The URL listed for the first edition, www.physics.purdue.edu/~ng/comp-phys.html, will take you to this new website.

The second edition owes much to our many colleagues who have sent us suggestions and new ideas for the new edition. In particular, we would like to thank James Behrens, Bob Delaney, Denis Donnelly, Eamin Jamshidi, Michael Oczkowski, Steve Turcotte, and Kobus Visser for alerting us to errors in the first edition, Aaron Montgomery for spotting (and correcting) a mistake in a draft of the second edition, and Eduardo Cuansing, Harvey Gould and Jan Tobochnik for their various contributions and general support. We also greatly appreciate the many constructive comments and suggestions from reviewers Gus Hart, James MacDonald, Micha Tomkiewicz, Thomas Vojta, and Matt Wood concerning drafts of the second edition. And of course, we are grateful to our Editors at Prentice-Hall, Erik Fahlgren and Christian Botting, for patiently guiding (and prodding) us through the preparation of this new edition.

In closing we would like to reaffirm that our goal has been to write a book that uses computational methods to do interesting *physics*. While numerical methods can be fun, they are *not* our primary purpose. We hope that this book helps the student in all of us learn about and enjoy doing physics.

Nicholas J. Giordano and Hisao Nakanishi

CHAPTER 1

A First Numerical Problem

Many problems encountered in physics involve ordinary differential equations. Examples include projectile motion, harmonic motion, and celestial mechanics, topics we will be discussing extensively in the next few chapters. We therefore begin with a problem involving a first-order differential equation and use it to introduce some computational techniques that will be employed extensively in later chapters. We will also proceed step by step through the construction of a program to deal with this problem, so as to illustrate in detail how a numerical approach is translated into a (working) computer program.

In this chapter it is not possible to provide a complete introduction to programming for students who have no previous exposure to the subject. Rather, our goal is to enable students with some (even limited) experience in programming to begin writing programs to treat the physics that will be encountered in this book. However, those students with no prior experience should not give up hope! With some extra effort and access to a good instructor or book on computer programming (or both), such students should be able to handle the material in this and later chapters.

1.1 RADIOACTIVE DECAY

It is well known that many nuclei are unstable. A typical example is the nuclear isotope ^{235}U (the uranium nucleus that contains 143 neutrons and 92 protons, for a total of 235 nucleons), which has a small, but not insignificant, probability for decaying into two nuclei of approximately half its size, along with an assortment of protons, neutrons, electrons, and alpha particles. This process of radioactive decay is random in the following sense. If you were given a single ^{235}U nucleus, you would not be able to predict precisely when its decay would take place. The best you could do would be to give the *probability* for decay. An equivalent way to describe such a process would be to give the average time for decay; for ^{235}U the mean lifetime is approximately 1×10^9 years.

It is useful to imagine that we have a sample containing a large number of ^{235}U nuclei, which would usually be the case if we were actually doing an experiment to study radioactive decay. If $N_U(t)$ is the number of uranium nuclei that are present in the sample at time t , the behavior is governed by the differential equation

$$\frac{dN_U}{dt} = -\frac{N_U}{\tau}, \quad (1.1)$$

where τ is the “time constant” for the decay. You can show by direct substitution that the solution to this differential equation is

$$N_U = N_U(0) e^{-t/\tau}, \quad (1.2)$$

where $N_U(0)$ is the number of nuclei present at $t = 0$. This solution may be familiar to you; similar equations and similar solutions are found in many other contexts.¹ We note that at time $t = \tau$ a fraction e^{-1} of the nuclei that were initially present has not yet decayed. It turns out that τ is also the mean lifetime of a nucleus.

1.2 A NUMERICAL APPROACH

While the differential equation (1.1) can be solved without resorting to a numerical approach, this problem is useful for introducing several computational methods that will be used extensively in later chapters. With that in mind we now consider a simple method for solving this problem numerically. Our goal is to obtain N_U as a function of t . Given the value of N_U at one particular value of t (usually at $t = 0$), we want to estimate its value at later times. This is called an initial value problem, and various general approaches for solving such ordinary differential equations are discussed in Appendix A. Here we will describe one particularly useful line of attack that is based on the Taylor expansion for N_U ,

$$N_U(\Delta t) = N_U(0) + \frac{dN_U}{dt} \Delta t + \frac{1}{2} \frac{d^2 N_U}{dt^2} (\Delta t)^2 + \dots , \quad (1.3)$$

where $N_U(0)$ is the value of our function at time $t = 0$, $N_U(\Delta t)$ is its value at $t = \Delta t$, and the derivatives are evaluated at $t = 0$. If we take Δt to be small, then it is usually a good approximation to simply ignore the terms that involve second and higher powers of Δt , leaving us with

$$N_U(\Delta t) \approx N_U(0) + \frac{dN_U}{dt} \Delta t . \quad (1.4)$$

The same result can be obtained from the definition of a derivative. The derivative of N_U evaluated at time t can be written as

$$\frac{dN_U}{dt} \equiv \lim_{\Delta t \rightarrow 0} \frac{N_U(t + \Delta t) - N_U(t)}{\Delta t} \approx \frac{N_U(t + \Delta t) - N_U(t)}{\Delta t} , \quad (1.5)$$

where in the last approximation we have assumed that Δt is small but nonzero. We can rearrange this to obtain

$$N_U(t + \Delta t) \approx N_U(t) + \frac{dN_U}{dt} \Delta t , \quad (1.6)$$

which is equivalent to (1.4). It is important to recognize that this is an *approximation*, which is why it contains the \approx symbol, not the $=$ symbol. The error terms that were dropped in deriving this result are of order $(\Delta t)^2$, which makes them at least one factor of Δt smaller than any of the terms in (1.6). Hence, by making Δt small, we would expect that the error terms can be made negligible. This is, in fact, the case in many problems, but there are situations in which the error terms can

¹For example, an equation of this kind describes the time dependence of the voltage across a capacitor in an *RC* circuit.

parts of an algorithm, expressed in “common” language. The idea is to give enough detail so that you (the readers of this book) can see how to translate each piece of pseudocode into the specific instructions of your favorite programming language. In most of this book, we will give our examples only in pseudocode. However, in this chapter we will work through an example using pseudocode along with actual codes in two popular languages, Fortran and C, so that you can see how the translation from pseudocode to an actual programming language can be done.⁴ Working programs for many of the problems in this book are available in Fortran, C, and Basic at our Web site.⁵

While programming, like handwriting, is a highly individualized process, there are certain recommended practices. After all, as in handwriting, it is important that we be able to understand programs written by others, as well as those we ourselves have written! With that in mind, this book will try to promote proper programming habits. The (admittedly very loose) analogy between handwriting and programming can be carried one step further. The first thing you should do in writing any program is to *think*. Before writing any detailed code, construct an outline of how the problem is to be solved and what variables or parameters will be needed. Indeed, the pseudocode version of a program will often provide this outline. For our decay problem we have already laid the foundation for a numerical solution in our derivation of (1.7). This equation also contains all of the variables we will need, N_U , t , τ , and Δt . Our stated goal was to calculate $N_U(t)$, but since the numerical approximation (1.7) involves the values of N_U only at times $t = 0$, $t = \Delta t$, $t = 2\Delta t$, etc., we will actually calculate N_U at just these values of t . We will use an array to store the values of N_U for later use. An array is simply a table of numbers (which will be described in more detail shortly). The first element in our array, that is, the first entry in the table, will contain N_U at $t = 0$, the second element will be the value at $t = \Delta t$, and so on. Our general plan is then to apply (1.7) repetitively to calculate the values of $N_U(t)$.

The overall structure of the program consists of four basic tasks: (1) declare the necessary variables, (2) initialize all variables and parameters, (3) do the calculation, and (4) store the results.

EXAMPLE 1.1 Pseudocode for the main program portion of the radioactive decay problem

- Some comment text to describe the nature of the program.
 - ▷ Declare necessary variables and arrays.
 - ▷ *initialize* variables.
 - ▷ Do the actual *calculation*.
 - ▷ *store* the results.

⁴We are certainly not implying that everyone should use Fortran or C, but these are the authors’ favorites.

⁵www.physics.purdue.edu/~giordano/comp-phys.html

Note that this is only the *main program*; the individual tasks such as initialization of variables and the actual calculation, will be done in subroutines or functions, which we will discuss below. First we consider how this main program might look in Fortran.⁶

radioactive decay main program in Fortran

```
c Simulation of radioactive decay
c Program to accompany "Computational Physics" by N. Giordano/H. Nakanishi
      program decay
c     declare the arrays we will need
      double precision n_uranium(100), t(100)
c use subroutines to do the work
      call initialize(n_uranium,t,tau,dt,n)
      call calculate(n_uranium,t,tau,dt,n)
      call store(n_uranium,t,n)
      stop
      end
```

Our program begins with a few comment statements that identify the program and tell a little about what it is supposed to do. In Fortran, comment lines are indicated by the 'c' in the first character in a line. Similar features are present in most languages.

The first line *program decay* gives the name of the main program. The line *double precision n_uranium(100), t(100)* declares the two arrays that will be used to store N_U and t . This is done in the main program because the subroutines that do the work will pass the arrays among each other through the main program. The first array, *n_uranium()*, will contain the calculated values of the number of uranium nuclei, while *t()* will contain the corresponding values of the time. Here we have arranged for our arrays *n_uranium()* and *t()* to each hold 100 values, as we expect this to be enough for this problem (of course, many languages allow one to resize an array as needed during a calculation). Note that we have used the descriptive name *n_uranium* to make the program easier to read and understand. It is also tempting to use the name *time* for the other array (instead of *t*), but some languages use *time* as a "reserved name" (e.g., the name of the time-of-day function), so to be safe we will avoid using it.

The rest of the work is done in three subroutines, *initialize*, *calculate*, and *store* which are called in succession. The subroutine names describe the function of each; these tasks correspond directly to the general program outline mentioned above. The *call* statements also include the names of the variables that each routine needs to do its job. The subroutine *initialize* sets the initial values of the variables, *calculate* uses the Euler method to do the computation,

⁶In the example Fortran codes that we list in this chapter, we have chosen an old (some may say *ancient*) style for widest compatibility. Today's Fortrans (*Fortran90* and *Fortran95*), though backward compatible with these examples, allow codes to be written in much more robust and extensible ways. However, we are not teaching a computer language in this book, and thus have decided to stick with the style that is most compatible with whatever compiler the reader may have.

and store puts the results into a file for later use (such as a graphical display). We next consider these three routines.

EXAMPLE 1.2 Pseudocode for subroutine initialize

- Prompt for and assign $N_U(0)$, τ , and Δt .
- Set initial value of time, $t(0)$.
- Set number of time steps for calculation.

A Fortran version of this subroutine could read

Fortran version of subroutine initialize

```

subroutine initialize(nuclei,t,tc,dt,n)
c   Initialize variables
double precision nuclei(1),t(1)
print *,'initial number of nuclei -> '
read(5,*)
nuclei(1)
print *,'time constant -> '
read(5,*)
t(1)
print *,'time step -> '
read(5,*)
dt
print *,'total time -> '
read(5,*)
time
t(1) = 0
n=min(int(time/dt),100)
return
end

```

In Fortran all subroutines begin with the `subroutine name` statement where `name` is the subroutine name. As in most other languages, the variables listed in parenthesis after the word `initialize` are passed into and out of the subroutine from/to the calling routine. This variable list must be in correspondence with the list used when the subroutine is called from the main program (or anywhere else). Comparing the calling line and the first line of the subroutine, we see that some of the names in the variable list in the "calling" part of the program, `n_uranium`, `t`, `tau`, `dt`, `n`, and the list in the "receiving" part of the program, `nuclei`, `t`, `tc`, `dt`, `n` are quite different. For example, the array names `n_uranium` and `nuclei` do not agree. But, as in most languages, the array and variable names declared within a subroutine definition are *dummy* names, and only the corresponding arrays and variables in the calling program are actually used.

After getting the calling variables organized, the `initialize` subroutine sets the initial values of the number of nuclei and the time. These are just the first values in the arrays `nuclei(1)` and `t(1)`. The `print` statements prompt the user for input, while the `read` statements take in the values. The last job is to set the value of `n`, which is the number of time steps to be performed.

The real work of computing the number of remaining nuclei is done in the subroutine `calculate`.

EXAMPLE 1.3 Pseudocode for subroutine calculate

- For each time step i (beginning with $i = 1$), calculate N_U and t at step $i + 1$:
 - ▷ $N_U(t_{i+1}) = N_U(t_i) - (N_U(t_i)/\tau)dt$ (Use the Euler method, (1.7))
 - ▷ $t_{i+1} = t_i + \Delta t$.
 - ▷ repeat for $n - 1$ time steps
-

In Fortran this can be written as

Subroutine calculate in Fortran

```
c subroutine calculate(n_uranium,t,tau,dt,n)
c Now use the Euler method
c variable dimensioning is used for arrays n_uranium() and t()
double precision n_uranium(n),t(n)
do i = 1,n-1
    n_uranium(i+1) = n_uranium(i)-(n_uranium(i)/tau)*dt
    t(i+1) = t(i) + dt
end do
return
end
```

This routine loops through each time step using the `do` and `continue` statements (other languages have analogous facilities to write simple loops). The key statement is the one in which `n_uranium(i+1)` is calculated. This statement contains *all* of the physics of the program and is closely analogous to (1.7). As mentioned above, the array `n_uranium` corresponds to the variable $N_U(t)$, and the value stored in the i th element of `n_uranium` is the number of uranium nuclei present at time $t(i)$.

The final subroutine `store` writes the result to a file. It uses a standard Fortran approach to store the results in the file `decay.dat`. The values of t and N_U are written as pairs, with a separate line for each value of t . The results can then be read from this file, in order to plot the results, or use them in a subsequent calculation.

Subroutine store in Fortran

```
c subroutine store(n_uranium,t,n)
double precision n_uranium(n),t(n)
open(1,file='decay.dat')
do i=1,n
    write(1,20) t(i),n_uranium(i)
end do
close(1)
format(1x,1p,2(e12.5,2x))
return
end
```

For convenience, our complete Fortran program is listed in one piece below.

Fortran version of radioactive decay program

```

c Simulation of radioactive decay
c Program to accompany "Computational Physics" by N Giordano/H. Nakanishi
c program decay
c declare the arrays we will need
c      double precision n_uranium(100), t(100)
c use subroutines to do the work
c      call initialize(n_uranium,t,tau,dt,n)
c      call calculate(n_uranium,t,tau,dt,n)
c      call store(n_uranium,t,n)
c      stop
c      end

c
c      subroutine initialize(nuclei,t,tc,dt,n)
c      Initialize variables
c      double precision nuclei(1),t(1)
c      print *, 'initial number of nuclei -> '
c      read(5,*)
c      nuclei(1)
c      print *, 'time constant -> '
c      read(5,*)
c      tc
c      print *, 'time step -> '
c      read(5,*)
c      dt
c      print *, 'total time -> '
c      read(5,*)
c      time
c      t(1) = 0
c      n=min(int(time/dt),100)
c      return
c      end

c
c      subroutine calculate(n_uranium,t,tau,dt,n)
c      Now use the Euler method
c      Variable dimensioning is used for arrays n_uranium() and t()
c      double precision n_uranium(n),t(n)
c      do i = 1,n-1
c          n_uranium(i+1) = n_uranium(i)-(n_uranium(i)/tau)*dt
c          t(i+1) = t(i) + dt
c      end do
c      return
c      end

c
c      subroutine store(n_uranium,t,n)
c      double precision n_uranium(n),t(n)
c      open(1,file='decay.dat')
c      do i=1,n
c          write(1,20) t(i),n_uranium(i)
c      end do
c      close(1)
c      format(1x,ip,2(e12.5,2x))
c      return
c      end

```

We have spent a lot of time discussing a Fortran program for the radioactive decay problem, but the basic ideas — the basic structure of the program — can be implemented in many other languages. Below we give a program written in C that does the same calculation, using the same program structure with the same algorithm. It begins by declaring the necessary variables, then uses subroutines `initialize`, `calculate`, and `store`, as outlined in the pseudocode in Example 1.3. These subroutines are quite similar to those in the Fortran program given at left, and produce identical results.

Radioactive decay program in C

```

/* decay.c
 * Simulation of radioactive decay
 * Program to accompany "Computational Physics" by N. Giordano/H. Nakanishi
 */
#include <math.h>
#include <stdio.h>
#define MAX 100
double n_uranium[MAX]; /* number of uranium atoms */
double t[MAX]; /* store time values here */
double dt; /* time step */
double tau; /* decay time constant */
double t_max; /* time to end simulation */

main()
{
    initialize(n_uranium,t,&tau,&dt);
    calculate(n_uranium,t,tau,dt);
    store(n_uranium,t);
}

/* initialize the variables */
initialize(nuclei,t,tc,dt)
double *nuclei,*t,*tc,*dt;
{
    printf("initial number of nuclei -> ");
    scanf("%lf",&nuclei[0]); /* begin using arrays at index 0 */
    printf("time constant -> ");
    scanf("%lf",tc);
    printf("time step -> ");
    scanf("%lf",dt);
    t[0] = 0.0;
    return;
}

/* calculate the results and store them in the arrays t() and n_u() */
calculate(nuclei,t,tc,dt)
double *nuclei,*t,tc,dt;
{
    int i;
    for(i = 0; i < MAX-1; i++) {
        nuclei[i+1] = nuclei[i] - (nuclei[i] / tc) * dt;
        t[i+1] = t[i] + dt;
    }
}

```

```

    }
    return;
}
/*      save the results to a file  */
store(nuclei,t)
double *nuclei,*t;
{
    FILE *fp_out;
    int i;
    fp_out = fopen("decay.dat","w");
    for(i = 0; i < MAX; i++) {
        fprintf(fp_out,"%g\t%g\n",t[i],nuclei[i]);
    }
    fclose(fp_out);
    return;
}

```

We have given specific example programs in Fortran and C, as these are languages that we (the authors) use in our everyday work.⁷ However, this certainly does *not* mean that one of these languages is the best choice for you; that is a judgment that you must make.

Next we come to an *extremely* important point, that is a key issue in nearly all the problems we will discuss in this book. Our program `decay` calculates how N_U varies with time, and puts the results as numbers into a file. However, we still have the job of *understanding* the results. In this problem, and in most cases, this job of understanding is best done by examining the results in graphical form. While there will be times when the result of a calculation can be expressed or conveyed as one or two numbers, it will usually happen that our calculations produce a *lot* of numbers. Making sense of such results is almost always simplest when the results are displayed graphically. In our decay problem we will want to make a graph of N_U as a function of t . Methods for producing a graphical display, either on a video display or on paper, can vary a *lot* from one computer platform and language to another, and there is no way that we can possibly give a full discussion of how you should go about it on your particular system. Some languages have very powerful graphics capabilities “built-in” (e.g., Matlab and Mathematica), while in other cases you may need to use separate graphics programs to display the results in the file `decay.dat`.

The ability to easily display results in graphical form is *absolutely essential* for work in computational physics. Fortunately, virtually all computer systems have the programs you will need (so you will not have to write your own!). Before you go much further in this book we urge you to learn how to use the graphics capabilities of your particular system.

Figure 1.1 shows an example of the output produced by our radioactive decay program, along with the exact result, Equation (1.2). Here we have used the initial values `n_uranium(1) = 100` and `t(1) = 0`, along with a time constant of 1 s and a time step of 0.05 s. We will consider the choice of time step later. For now we note only that our calculated values compare well with the exact result.

⁷This may also give you some idea of when we first learned to program.

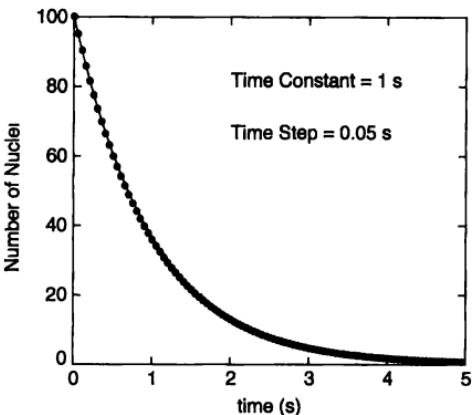


FIGURE 1.1: Circles indicate the numerical solution to the radioactive decay problem. The solid line is the exact solution Equation (1.2).

1.4 TESTING YOUR PROGRAM

The title of Section 1.3 was a little misleading. It was concerned with producing a program that runs without any errors in the syntax, etc., such as undeclared arrays, spelling errors, or variables omitted when calling a subroutine. However, we should not really consider it to be a working program until we are convinced that its output is correct! Checking a program is not always a trivial task, but there are some general guidelines. After a program has been debugged so that it can be run without any complaints about the syntax, there are several things you should do to verify that the results of the program are correct.

Does the output look reasonable? Before you perform any calculation you should always have at least a rough idea of what the result should be. The first thing you should do when considering the results from any program is ask whether or not they are consistent with your intuition and instincts. This exercise can also improve your overall understanding of the problem. When you show your result to someone else, you should always be able to convince them that it makes sense.⁸

Does your program agree with any exact results that are available? Since we knew the analytic solution for our radioactive decay problem, we were able to compare our numerical values with the exact result. While such a comparison will not be possible for most of the numerical calculations you will encounter, exact results are

⁸One definition of a physicist is “a person who can calculate anything to within an order of magnitude.” This is perhaps a bit overstated, but you should usually be able to anticipate an answer with this sort of accuracy.

sometimes available in certain limits, that is, for special values of the parameters. You should always run your program in those limits to check that it gives the correct answer. This is a necessary (but not sufficient) test that a program is correct in the general case.

Always check that your program gives the same answer for different “step sizes.” Our decay program involved a time-step variable, dt , and most other numerical calculations involve similar step- or grid-size parameters. Your final answer should be independent of the values of such parameters. This is another necessary (but not sufficient) test of a program’s accuracy.

Checking your program should not be viewed as a trivial, last-minute job. It is not unreasonable to spend as much time checking (and correcting) a program as it takes writing it. After all, a result is not much good if you don’t trust it to be correct.

1.5 NUMERICAL CONSIDERATIONS

The issue of numerical errors is central to the computational solution of any problem. Indeed, this is such an important topic that there are *many* books devoted to this area. Questions such as how to design or choose the best algorithm for a particular problem, and how to estimate the numerical errors associated with an algorithm, are central topics in many computer science and applied mathematics courses. We could easily spend *a lot* of space discussing the Euler method and all the other algorithms we will be using in this book. However, while this would certainly be an instructive thing to do, it would leave us with very little time to explore the *physics* that can be done with these algorithms. In this book our emphasis will be on the physics, rather than the numerical methods. This does not mean that we view numerical methods as unimportant, but just that there is only so much that we can do in one book. Nevertheless, we will be making some comments about the choice of algorithms, their numerical uncertainties, and their stability. We will attempt to blend these discussions with the physics as we go, particularly when they are closely related to the physics of the problem. The Appendices contain more detailed and systematic discussions of many of the numerical methods we use in this book.

With our radioactive decay program, errors were introduced by the *approximation* used to estimate the solution of the differential equation (1.7). Errors are also produced by the finite numerical precision in any programming language, including Fortran and C. These are known as round-off errors, and are almost always present when numbers are represented by a finite number of digits. Fortunately, this is not usually a severe problem, as most modern computer language systems employ a large number of significant digits. In order to be as safe as possible you should always use double precision variables in Fortran, C, and other languages where this is an option. However, this is no guarantee that these errors will be negligible, as the numerical solutions to certain types of problems are inherently sensitive to round-off errors. It is hard to give any general rules about what to watch for, but we will discuss such difficulties when they arise in several problems later in this book.

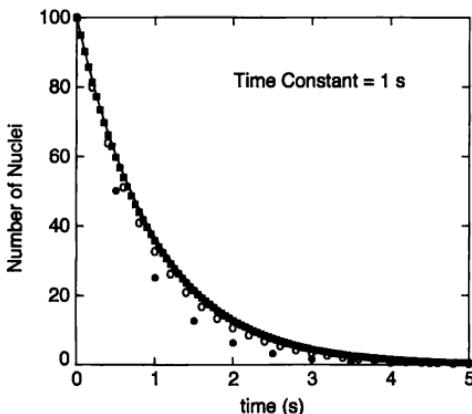


FIGURE 1.2: Numerical solution of the radioactive decay problem using the Euler method for different values of the time step. The time constant was 1 s, and the time steps were: filled circles, 0.5 s, open circles, 0.2 s, squares, 0.05 s. The solid curve is the exact solution

In attacking the radioactive decay problem we were led to treat time as a discrete variable; that is, we converted the differential equation (1.1) into a difference equation (1.7), which enabled us to compute estimates for N_U at the discrete times $n\Delta t$, where n is an integer. Such a “discretization” of time, or space, or both, is a common practice in numerical calculations, and brings up two related questions: (1) How do we know that the errors introduced by this discreteness are negligible, and (2) How do we choose the value of such a step size for a calculation? Again, there are no general answers to these questions. All we can do is give some guidelines, which can be illustrated using our radioactive decay problem.

The first guideline is that a calculation should *always* be repeated using several different values of the step size. Figure 1.2 shows the results from our radioactive decay calculation using three different values of the time step. The time constant τ was taken to be 1 s, and the symbols show the numerical results for time steps of 0.5, 0.2, and 0.05 s. Also shown is the exact solution (the solid line). Because of the nonzero step size, our program only yields results at discrete values of t , which is why the calculated points become more dense as the time step is reduced. We see that as the time step is made smaller, the calculated values converge quickly to the exact solution. This is not surprising, since intuition tells us that our approximations should become better as Δt is reduced. Being more quantitative, we already noted that the terms dropped in our derivation of (1.6) are of order $(\Delta t)^2$. In order to understand the accuracy of the calculated result at a specific value of t , we note that it takes $t/\Delta t$ time steps to reach this point (starting from $t = 0$). The total

error (often termed the *global* error) is proportional to the product of the number of time steps ($t/\Delta t$) and the error per step ($\sim (\Delta t)^2$), and is thus of order Δt . Hence, we expect the difference between the numerical (Euler) and exact results to be cut in half for each factor of 2 reduction of Δt . It is clear from Figure 1.2 that the numerical results become extremely close (on the scale of this figure) to the exact solution for time steps below about 0.05 s.

For our decay problem we have the exact solution, which makes it easy to evaluate the accuracy of our numerical results. However, we will not usually be this fortunate, so it is important to consider how to evaluate our results when the exact result is not available. In such cases you should always check that the calculated result converges to a fixed value (or curve) as the step size is made smaller. A closely related matter concerns the choice of step size in the first place; that is, what is a reasonable value? Here, yet again, there are no hard and fast rules. Ideally, you should use a step size that is small compared to any characteristic time scales in the problem. In our decay problem this time scale is τ , since its value determines how fast N_U varies. Our recommendation is that you pick the time step to be a small fraction of τ . We see from Figure 1.2 that a time step that is a few percent (or less) of this characteristic scale is a good choice.

One more important issue concerning numerical precision will be encountered later in this book. There are situations in which what appears to be a very reasonable numerical approach can be inherently unstable. For example, we will find in Chapter 3 that the Euler method, which we have found to work well for the radioactive decay problem, fails miserably when applied to problems involving oscillatory motion. The message here is that there is no one *best* method for solving ordinary differential equations or any other particular class of problems. You must apply your understanding of the theory behind an algorithm (as discussed, for example, in a numerical methods book such as Press et al. [1986]) and your ingenuity (and also this book!) to choose the appropriate method for any particular problem.

1.6 PROGRAMMING GUIDELINES AND PHILOSOPHY

In our construction of the radioactive decay program we noted that writing a program is necessarily an individualistic endeavor. Nevertheless, there are certain guidelines that are generally recommended, some of which were touched on above. All of these guidelines have essentially the same purpose: to make the program as easy to understand as possible.

- *Program structure.* Use subroutines to organize the major tasks and make the program more readable and understandable. The main program for the decay problem was basically an outline of the program; we recommend this style for *all* programs. Use subroutines and functions to perform any jobs that take more than a few lines of code, or that are required repeatedly.
- *Use descriptive names.* Choose the names of variables and subroutines according to the problem at hand. Descriptive names make a program easier to understand, as they act as built-in comment statements.

- *Use comment statements.* Include comment statements to explain program logic and describe variables. A short subroutine that uses descriptive variable names should not need a large number of comment statements.
- *Sacrifice (almost) everything for clarity.* It is often tempting to write a critical piece of code in a very compact or terse manner in the misguided belief that this will make the program run faster. While a compact code may occasionally run faster, this is not a reliable indication of computational efficiency. Moreover, such compact code will nearly always have a cost in terms of clarity and readability; there are many famous pieces of code in Fortran and C that come to mind here. It is almost always better to take a few more lines, or a few more variables, to do a job, if it makes the code more understandable.⁹ It is also a good idea to make the code look as much as possible like the associated physics equation. This will save time later when you are checking your program.

The execution speed becomes an important programming issue only when the overall computer time requirements become substantial (say, over several minutes). Even then, significant savings can usually be made only by algorithmic improvements and it is rare to see a noticeable gain in efficiency due to fiddling with a line of code here or there. In this book we will generally write our programs without worrying much about program efficiency. If you want to “tuneup” a program to make it run faster, this should be done only after it is known to be correct. Moreover, modern language compilers often do a good job of optimizing code without the programmer even knowing it!

- *Take time to make graphical output as clear as possible.* Think carefully about what quantities to plot and in what manner.¹⁰ The axes should be labeled clearly (including units, where appropriate), and parameter values given directly on the graph.

EXERCISES

Note: Throughout this book, exercises labeled with an * are more challenging than others in that section

- 1.1. The velocity of a freely falling object near Earth’s surface is described by the equation

$$\frac{dv}{dt} = -g, \quad (1.8)$$

where v is the velocity and $g = 9.8 \text{ m/s}^2$ is the acceleration due to gravity. Write a program that employs the Euler method to compute the solution to (1.8); that is, calculate v as a function of t . For simplicity, assume that the initial velocity is zero—that is, the object starts from rest—and calculate the solution for times $t = 0$ to $t = 10 \text{ s}$. Repeat the calculation for several different values of the time step, and compare the results with the exact solution to (1.8). It turns out

⁹In very large-scale projects, there may perhaps come a time when *clarity* begins to compete with *computational efficiency*. If so, some compromises may need to be made; however, in all cases, *clarity* should receive an extremely high priority.

¹⁰Sometimes, plotting either or both axes logarithmically may be more appropriate than using linear scales, see Chapter 3 for some examples.

that for this case the Euler method gives the exact result. Verify this with your numerical results and prove it analytically.

- 1.2. The position of an object moving horizontally with a constant velocity, v , is described by the equation

$$\frac{dx}{dt} = v \quad (1.9)$$

Assuming that the velocity is a constant, say $v = 40$ m/s, use the Euler method to solve (1.9) for x as a function of time. Compare your result with the exact solution.

- 1.3. It is often the case that the frictional force on an object will increase as the object moves faster. A fortunate example of this is a parachutist; the role of the parachute is to produce a frictional force due to air drag, which is larger than would normally be the case without the parachute. The physics of air drag will be discussed in more detail in the next chapter. Here we consider a very simple example in which the frictional force depends on the velocity. Assume that the velocity of an object obeys an equation of the form

$$\frac{dv}{dt} = a - bv, \quad (1.10)$$

where a and b are constants. You could think of a as coming from an applied force, such as gravity, while b arises from friction. Note that the frictional force is negative (we assume that $b > 0$), so that it opposes the motion, and that it increases in magnitude as the velocity increases. Use the Euler method to solve (1.10) for v as a function of time. A convenient choice of parameters is $a = 10$ and $b = 1$. You should find that v approaches a constant value at long times; this is called the terminal velocity.

- *1.4. Consider a radioactive decay problem involving two types of nuclei, A and B , with populations $N_A(t)$ and $N_B(t)$. Suppose that type A nuclei decay to form type B nuclei, which then also decay, according to the differential equations

$$\begin{aligned} \frac{dN_A}{dt} &= -\frac{N_A}{\tau_A}, \\ \frac{dN_B}{dt} &= \frac{N_A}{\tau_A} - \frac{N_B}{\tau_B}, \end{aligned} \quad (1.11)$$

where τ_A and τ_B are the decay time constants for each type of nucleus. Use the Euler method to solve these coupled equations for N_A and N_B as functions of time. This problem can also be solved exactly, as was the case with our original nuclear decay problem (1.1). Obtain the analytic solutions for $N_A(t)$ and $N_B(t)$, and compare them with your numerical results. It is also interesting to explore the behavior found for different values of the ratio τ_A/τ_B . In particular, try to interpret the short and long time behaviors for different values of this ratio.

- *1.5. Consider again a decay problem with two types of nuclei A and B , but now suppose that nuclei of type A decay into ones of type B , while nuclei of type B decay into ones of type A . Strictly speaking, this is not a “decay” process, since it is possible for the type B nuclei to turn back into type A nuclei. A better analogy would be a resonance in which a system can tunnel or move back and forth between two states A and B which have equal energies. The corresponding

rate equations are

$$\begin{aligned}\frac{dN_A}{dt} &= \frac{N_B}{\tau} - \frac{N_A}{\tau}, \\ \frac{dN_B}{dt} &= \frac{N_A}{\tau} - \frac{N_B}{\tau},\end{aligned}\quad (1.12)$$

where for simplicity we have assumed that the two types of decay are characterized by the same time constant, τ . Solve this system of equations for the numbers of nuclei, N_A and N_B , as functions of time. Consider different initial conditions, such as $N_A = 100$, $N_B = 0$, etc., and take $\tau = 1$ s. Show that your numerical results are consistent with the idea that the system reaches a steady state in which N_A and N_B are constant. In such a steady state, the time derivatives dN_A/dt and dN_B/dt should vanish.

- 1.6.** Population growth problems often give rise to rate equations that are first-order. For example, the equation

$$\frac{dN}{dt} = aN - bN^2, \quad (1.13)$$

might describe how the number of individuals in a population, N , varies with time. Here the first term aN corresponds to the birth of new members, while the second term $-bN^2$ corresponds to deaths. The death term is proportional to N^2 to allow for the fact that food will become harder to find when the population N becomes large. Begin by solving (1.13) with $b = 0$ using the Euler method, and compare your numerical result with the exact solution. Then solve (1.13) with nonzero values b . Give an intuitive explanation of your results. Interesting values of a and b depend on the initial population N . For small $N(0)$, $a = 10$ and $b = 3$ is a good choice, while for $N(0) = 1000$ a good choice is $a = 10$ and $b = 0.01$.

REFERENCES

- [1] B. W. Kernighan and P. J. Plauger, 1978, *The Elements of Programming Style*, 2d ed., McGraw-Hill, New York. A very nice discussion of how to write clear, readable, and efficient programs in any language.
- [2] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. An excellent all-purpose reference on numerical methods and why they work.

Realistic Projectile Motion

In this chapter we consider several problems involving the motion of objects through the atmosphere. The problems are all described by ordinary differential equations in which initial values are given and all can be solved using the Euler method, which was introduced in the last chapter. These problems are good examples of interesting physics involving the mechanics of macroscopic objects, which can't be solved analytically, but can be easily tackled with a computer.

We begin with a discussion of the motion of a bicycle traveling on flat terrain. We will find that air resistance must be included if we are to obtain a realistic description of the problem, and this leads us to a simple but fairly accurate model of the drag force due to the atmosphere. Next we treat projectile motion in two dimensions as we consider the trajectory of a shell fired by a large cannon. Again, the effect of air resistance is important, but now its variation with altitude plays a key role. We proceed to the national pastime, baseball, and consider the trajectory of a batted ball and the motion of thrown balls (curve balls and knuckleballs). To model a batted ball we are led to consider air resistance a little more realistically than in the earlier problems, while for a thrown ball we must also include spin-dependent forces. These themes are developed further in our discussion of the motion of golf balls, where we answer the eternal question: "Why do golf balls have dimples?"

2.1 BICYCLE RACING: THE EFFECT OF AIR RESISTANCE

The bicycle is an extremely efficient form of transportation, a fact that is well known to anyone who rides one. Our goal in this section is to understand the factors that determine the ultimate speed of a bicycle and to estimate this speed for a realistic case. We will begin by ignoring friction; we'll have to add it eventually, of course, but let us first understand how to deal with the simpler case without friction.¹ Our equation of motion is Newton's second law, which can be written in the form

$$\frac{dv}{dt} = \frac{F}{m}, \quad (2.1)$$

where v is the velocity, m is the mass of the bicycle-rider combination, t is the time, and F is the force on the bicycle that comes from the effort of the rider (here we will assume that the bicycle is moving on flat terrain). Dealing properly with F

¹The astute reader might now ask "But could a cyclist really propel a bicycle if there were no friction?", and you are right if your own answer is *No*. Without friction between the tires and the road, the wheels would simply slip, and the bicycle would go nowhere. Friction is thus essential for bicycles and cars, and other cases when an object must roll without slipping. However, here we are using the term "friction" to describe the resistance to the translational motion of the bicycle, i.e., the forces that cause energy dissipation. This dissipation typically occurs due to air drag, along with deformations of the tires and losses internal to the bicycle.

is complicated by the mechanics of a bicycle, since the force exerted by the rider is transmitted to the wheels by way of the chainring, gears, etc. This makes it very difficult to derive an accurate expression for F . However, there is another way to attack this problem that avoids the need to know the force. This alternative approach involves formulating the problem in terms of the power generated by the rider. Physiological studies of elite racing bicyclists have shown that these athletes are able to produce a power output of approximately 400 watts over extended periods of time (~ 1 h). Using work-energy ideas we can rewrite (2.1) as

$$\frac{dE}{dt} = P, \quad (2.2)$$

where E is the total energy of the bicycle-rider combination, and P is the power output of the rider. For a flat road the energy is all kinetic, so $E = \frac{1}{2}mv^2$, and $dE/dt = mv(dv/dt)$. Inserting this into (2.2) yields²

$$\frac{dv}{dt} = \frac{P}{mv}. \quad (2.3)$$

If P is a constant, (2.3) can be solved analytically. Rearranging gives

$$\int_{v_0}^v v' dv' = \int_0^t \frac{P}{m} dt', \quad (2.4)$$

where v_0 is the velocity of the bicycle at $t = 0$. Integrating both sides and solving for v then leads to

$$v = \sqrt{v_0^2 + 2Pt/m}. \quad (2.5)$$

While this is the correct solution of the equation of motion (2.2), it cannot be the whole story, since it predicts that the velocity will increase without bound at long times. We will correct this “unphysical” result in a moment, when we generalize our model to include the effect of air resistance. The new term we will add to the equation of motion will require us to develop a numerical solution, so with that in mind we consider a numerical treatment of (2.3). We begin with the finite difference form for the derivative of the velocity

$$\frac{dv}{dt} \approx \frac{v_{i+1} - v_i}{\Delta t}, \quad (2.6)$$

where we have assumed small, discrete time steps of size Δt , and taken v_i to be the velocity at time $t_i \equiv i\Delta t$, where i is an integer (this should be familiar from Chapter 1). Inserting this into (2.3) we obtain, after a little rearranging

$$v_{i+1} = v_i + \frac{P}{mv_i} \Delta t. \quad (2.7)$$

With the approximation made in (2.6), the leading correction terms are proportional to $(\Delta t)^2$.

²This assumes implicitly that very little energy is lost to friction in the bicycle itself; we'll include other sources of energy loss in a moment.

Given the velocity at time step i (i.e., v_i), we can use (2.7) to calculate an *approximate* value of the velocity at the next step, v_{i+1} . Hence, if we know the initial velocity, v_0 , we can obtain v_1 , then v_2 , and so on, and thereby estimate the velocity at all future times. This is just the Euler method, which we encountered in Chapter 1. There are more sophisticated methods for numerically solving differential equations of this form, and a few of them are described in Appendix A. In this book we will only rarely encounter problems for which the Euler method, or simple variants, are not adequate. However, you should be aware that: (1) methods that are more powerful³ than the Euler method do indeed exist; (2) these more powerful methods can certainly be used to solve all of the problems in this book where we use the Euler method; and (3) if you are going to do this sort of thing for a living it is worth your while to learn about these other methods. In taking what some purists might consider a quick-and-dirty approach (by our use of the Euler method), we are *not* trying to minimize the importance of other algorithms. Our intent is merely to use the simplest numerical method (which will give the correct solution, of course) appropriate for the job, so that we can emphasize the *physics* of our problems and not let the numerical techniques get in the way. You are encouraged to follow their own taste here, and with all of the other problems discussed in this book, and use the methods with which you feel most comfortable.

The outline of a program that performs this calculation is shown below. The general program structure is similar to the nuclear decay program in Chapter 1, in which variables and arrays are first declared and initialized, the actual calculation is performed using the Euler approximation, and the results are then stored. Here we only sketch the main program and the calculation subroutine, and leave the rest for the reader.

EXAMPLE 2.1 Pseudocode for bicycle calculation

- **main program**
 - ▷ Declare necessary variables, including arrays $t()$ and $v()$ to store time and velocity.
 - ▷ Set values for the power P , mass m , and time step Δt , and total number of time steps N , along with the initial value for the velocity (v_0).
 - ▷ Do the actual calculation.
 - ▷ Store the results.
 - **calculate subroutine**
 - ▷ For each time step i calculate v and t at step $i + 1$:
 - $$v_{i+1} = v_i + \left(\frac{P}{mv_i} \right) \Delta t$$
 - $$t_{i+1} = t_i + \Delta t$$
 - ▷ Repeat for N time steps.
-

³By “more powerful” we mean that they yield more accurate solutions for a given amount of computer time, or that they are more stable in a numerical sense.

We are now ready to compute a numerical solution. We assume that the bicycle starts with a velocity of $v_0 = 4 \text{ m/s}$ (about 10 mph) and take $P = 400 \text{ W}$, the value obtained from physiological measurements of well-trained athletes, as mentioned above. The last point to consider is the choice of Δt . Roughly speaking, Δt should be sufficiently small that the velocity changes only a little during such an interval. What it means to be "sufficiently" small is hard to say, in general. A useful rule of thumb is to begin with a time step that is about 1 percent of any time scales in the problem, and then repeat the calculation with several smaller values.⁴ Smaller time steps will give smaller correction terms [e.g., in (2.7)] and thus more accurate results, but the calculation will take a computational time proportional to $(\Delta t)^{-1}$, so there is a trade-off here. For our bicycle problem it will turn out that time steps smaller than about 1 s are adequate. It is usually instructive to repeat a calculation using different values of the time step so as to observe how the solution converges to the correct one as Δt is reduced.

The results for our "frictionless" bicycle are shown in Figure 2.1, where we have used a mass of 70 kg for the bicycle-rider combination (elite bicycle racers tend to be rather slender). We see immediately that, as anticipated above, our model has a serious problem; the velocity reaches 45 m/s (about 100 mph) in less than 3 min. While this performance would not be particularly impressive for a car, it is certainly not within reach of any known bicycles (or bicyclists). We also see that v appears to grow indefinitely, and this makes the origin of our problem clear. We have not included any sources of dissipation, so given our assumption of constant power exerted by the rider, the kinetic energy will increase without limit. If we want to have a realistic model, we must add some mechanism for energy loss.

For a well-tuned bicycle traveling at more than about 5 or 10 mph, the energy lost to friction in the hubs and tires is negligible compared to that caused by air resistance, that is, atmospheric drag. Thus, a reasonably realistic model of the motion of a bicycle need only consider this one source of friction. As we will come to appreciate in later sections, the physics of air resistance is a very complicated problem. In general, this force can be written in the fairly innocent form

$$F_{\text{drag}} \approx -B_1 v - B_2 v^2. \quad (2.8)$$

You will no doubt notice that (2.8) bears a strong resemblance to a Taylor expansion. At extremely low velocities the first term dominates, and its coefficient B_1 can be calculated for objects with simple shapes. This is known as Stokes' law and is considered in most elementary mechanics texts. However, at any reasonable velocity the v^2 term in (2.8) dominates for most objects. Moreover, B_2 cannot be calculated exactly for objects even as simple as a baseball, and certainly not for a complicated object like a bicycle. We can, however, make an *approximate* estimate of B_2 , as follows. As an object moves through the atmosphere, it must push the air in front of it out of the way. The mass of air moved in time dt is $m_{\text{air}} \sim \rho A v dt$,

⁴While the "characteristic" time scale is often not a unique or precise quantity, in some cases it is easy to estimate. For example, in the radioactive decay problem in Chapter 1 it was the time constant τ of the decay. In the present problem, one natural choice is the time it takes to attain a velocity of the order of terminal velocity (see Figure 2.2).

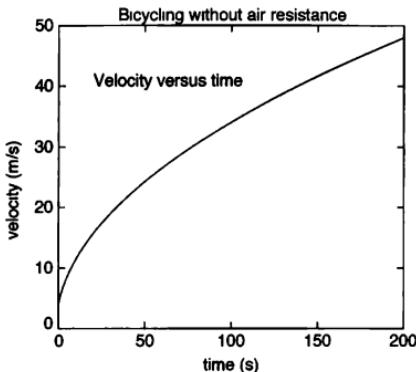


FIGURE 2.1: Velocity as a function of time for our bicycle problem, assuming no air resistance. The mass of the bicycle-rider combination was 70 kg, the initial velocity was 4 m/s, and the time step was 0.1 s. Here and in the other figures in this chapter, the results at each time step are connected to obtain an essentially smooth curve.

where ρ is the density of air and A the frontal area of the object. This air is given a velocity of order v , and hence its kinetic energy is $E_{\text{air}} \sim m_{\text{air}} v^2 / 2$. This is also the work done by the drag force (the force on the object due to air resistance) in time dt , so $F_{\text{drag}} v dt = E_{\text{air}}$. Putting this all together we find⁵

$$F_{\text{drag}} \approx -\frac{1}{2} C \rho A v^2. \quad (2.9)$$

C is known as the drag coefficient, and our simple argument predicts that it is equal to 1. However, we caution that our calculation was only approximate; while we expect it to give the correct functional dependence of F_{drag} on A and v , we certainly do not expect the precise value it predicts for C to be correct. Indeed, we should expect that C will depend on the “aerodynamics” of an object. The best way to determine the drag coefficient of any particular object is via wind tunnel measurements, or similar experiments.

It is easy to include this drag force in our calculation; Equation (2.9) contributes another term to the right-hand side of (2.7), which becomes

$$v_{i+1} = v_i + \frac{P}{mv_i} \Delta t - \frac{C\rho A v_i^2}{2m} \Delta t, \quad (2.10)$$

and we can use the Euler method to obtain v as a function of t as before. The

⁵Note that this expression differs by a factor of $\frac{1}{2}$ from the corresponding expression for F_{drag} in the first edition of this book. This change is made to conform better to the standard definition of the drag coefficient, and does not change any of the actual results.

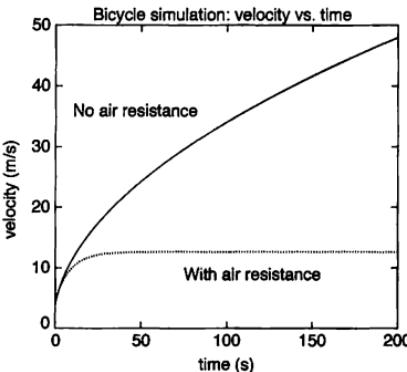


FIGURE 2.2: Velocity as a function of time for our bicycle problem, with and without air resistance. The drag coefficient was 0.5, the frontal area 0.33 m^2 , the mass of the bicycle-rider combination 70 kg, the initial velocity 4 m/s, and the time step 0.1 s

program sketched above can again be used, provided that we add the drag term to the Euler equation in the `calculate` subroutine.

The result obtained from solving (2.10) is shown in Figure 2.2, where we have assumed a frontal area of $A = 0.33 \text{ m}^2$, which is a typical estimate for a rider in a racing crouch. We obtain an ultimate (i.e., terminal) velocity of approximately 13 m/s (about 30 mph), which is in reasonable accord with the performance of an elite racer.⁶

Besides giving a good estimate of the speed of a bicycle racer, our model provides some insight into racing strategy. It is clear from Figure 2.2 that reducing the air resistance is the key to increasing the speed for a given power exertion or decreasing the power required to maintain a given speed. This is, of course, why bicycle racers often ride in a pack. The idea is to let the riders at the front of the pack break the wind, and thereby lower the effective frontal area for those in the pack who "draft" off those in front. For this reason, it is extremely difficult for a single rider to escape from a large group; the energy required of the single rider is much greater than that exerted by the riders in the pack (assuming they share time at the front).⁷ This also explains why the relatively recent development of "aerobars," handle bars that allow the rider to assume an extremely narrow profile and thereby reduce his frontal area, enables an isolated rider to go faster than with conventional handlebars. This is useful in individual events such as time trials and

⁶It is interesting to note that as of 2004 the record for the distance traveled by bicycle in 1 h is a little more than 56 km. This corresponds to an average velocity of approximately 15 m/s, which is within about 15 percent of our result for the terminal velocity.

⁷Estimates are that the effective frontal area is reduced by approximately 30 percent for a rider in the middle of a pack, as compared with a rider at the front.

triathlons, where following closely behind other riders is not allowed.

With a little work to include air resistance, we have developed a simple but reasonably accurate model of bicycle performance. Both our model and the numerical approach can be easily generalized to include other factors, which will be our job in the following sections.

EXERCISES

- 2.1. Compare the exact solution for the velocity as a function of time without air resistance with the numerical results in Figure 2.1 and show that they agree.
- 2.2. Investigate the effect of varying both the rider's power and frontal area on the ultimate velocity. In particular, for a rider in the middle of a pack, the effective frontal area is about 30 percent less than for a rider at the front. How much less energy does a rider in the pack expend than does one at the front, assuming they both move at a velocity of 13 m/s?
- 2.3. Estimate the effect of the Stokes term ($-B_1 v$ term) in (2.8). This term, neglected in all of the above discussions, represents *viscous drag*, that is, the dragging of air by the moving bicycle due to the viscosity of air.

The viscosity η is usually defined for a fluid contained between two parallel plates of area A by

$$F = \eta A \frac{\partial v}{\partial z}, \quad (2.11)$$

where F is the drag force and z is the transverse coordinate. As a rough approximation, let us replace $\frac{\partial v}{\partial z}$ by v/h where h is the height of the bicycle plus the bicyclist. For air, $\eta \approx 2 \times 10^{-5}$ Pa · s. Do the same also if the bicycle is being ridden *under water*. For water, $\eta \approx 1 \times 10^{-3}$ Pa · s, and don't forget that the density of water is much higher than air!

- 2.4. Generalize the model to treat motion through mountainous terrain. A steep hill is one with a 10 percent grade (that is, $\tan \theta = 0.1$, where θ is the angle the hill makes with the horizontal). Calculate how fast our bicyclist can travel up and down such a slope. Does racing strategy change in these situations? Determine what conditions (the steepness of the grade and the rider's frontal area) would be required for a bicycle to reach a velocity of 70 mph. This is reportedly the speed that professional riders sometimes attain on steep descents.⁸
- *2.5. You might wonder why we did not let our bicycle begin from rest, but instead gave it a nonzero initial velocity. The reason for this is that (2.7) breaks down when $v_i = 0$, since then the term involving P is infinite.⁹ If $v_i = 0$, then for a nonzero P the derivative dv/dt is, according to (2.3), infinite. This is difficult to handle in a numerical approach, and it also doesn't make sense from a physical point of view. The problem arises from our assumption that the bicyclist maintains a constant power output. This assumption must break down when the bicycle has a very small velocity, since it would then require that the rider exert extremely large forces (recall that the instantaneous power is the product of the force and the velocity). At low velocities it is more realistic to assume that the rider is able to exert a constant force. To account for this we can modify our bicycle model

⁸It is interesting to also include the effect of a tail wind in this calculation, as discussed in the section on the motion of a batted baseball.

⁹The problem is also evident from (2.3), and thus is not a product of the Euler method.

so that for small v there is a constant force on the bicycle, F_0 , which leads to the equation of motion

$$\frac{dv}{dt} = \frac{F_0}{m}. \quad (2.12)$$

The corresponding Euler equation is

$$v_{i+1} = v_i + \frac{F_0}{m} \Delta t, \quad (2.13)$$

and the difficulty that occurs when $v = 0$ is eliminated

Rewrite your bicycle program to incorporate (2.13). That is, use the Euler method with (2.13) when the velocity is small, and (2.7) when v is large. Let the crossover from small to large v occur when the power ($= F_0 v$) reaches P . Use the same parameters as in Figure 2.2, and take $F_0 = P/v^*$ where $v^* = 7$ m/s. (This corresponds to a force approximately twice that found when the bicycle is traveling at its maximum velocity, which seems like a reasonable approximation.)

2.2 PROJECTILE MOTION: THE TRAJECTORY OF A CANNON SHELL

The Euler method we used to treat the bicycle problem can easily be generalized to deal with motion in two spatial dimensions. To be specific, we consider a projectile such as a shell shot by a cannon. We have a very large cannon in mind, and the large size will determine some of the important physics. If we ignore air resistance, the equations of motion, which are again obtained from Newton's second law, can be written as

$$\frac{d^2x}{dt^2} = 0 \quad (2.14)$$

$$\frac{d^2y}{dt^2} = -g,$$

where x and y are the horizontal and vertical coordinates of the projectile, and g is the acceleration due to gravity. These are second-order differential equations, as opposed to the first-order equations we have encountered so far, so we must generalize our approach a bit. We have seen that with a first-order equation, such as (2.1), it is possible to use a finite difference approximation for the derivative (2.6) to obtain a simple algebraic equation containing the variable of interest at two adjacent time steps. However, if we were to take the same approach with one of the equations in (2.14) and write a finite difference approximation to the second derivative, we would obtain a more complicated algebraic equation involving our variable at three time steps. We will see how this works out in Chapter 6 (and later), as sometimes this is the best approach. However, in the present case it is possible to avoid this complication by recasting the differential equations in the following way.

Let us write each of these second-order equations as two first-order differential equations

$$\frac{dx}{dt} = v_x \quad (2.15)$$

$$\frac{dv_x}{dt} = 0$$

$$\frac{dy}{dt} = v_y$$

$$\frac{dv_y}{dt} = -g,$$

where v_x and v_y are the x and y components of the velocity. While we now have twice as many equations to deal with (four altogether), we can use our standard Euler approach to solve each one. To use the Euler method, we write each derivative in finite difference form, as in (2.6), which leads to

$$x_{i+1} = x_i + v_{x,i} \Delta t \quad (2.16)$$

$$v_{x,i+1} = v_{x,i}$$

$$y_{i+1} = y_i + v_{y,i} \Delta t$$

$$v_{y,i+1} = v_{y,i} - g \Delta t.$$

Given the initial values of x , y , v_x , and v_y , we can use (2.16) to estimate their values at later times. We emphasize yet again that these estimates are *approximate*, since there are correction terms to (2.16) that are of order $(\Delta t)^2$ and higher. By choosing Δt to be sufficiently small, we will usually be able to make these corrections negligible. However, their existence should not be forgotten.

In our treatment of the bicycle problem we found that air resistance was very important, so we now add that to the model. As was the case with a bicycle, we will assume that the magnitude of the drag force on our cannon shell is given by

$$F_{\text{drag}} = -B_2 v^2, \quad (2.17)$$

where $v = \sqrt{v_x^2 + v_y^2}$ is the speed of the shell. This force is always directed opposite to the velocity, so we must consider the vector components as illustrated in Figure 2.3. We find

$$F_{\text{drag},x} = F_{\text{drag}} \cos \theta = F_{\text{drag}} (v_x / v), \quad (2.18)$$

with a similar expression for $F_{\text{drag},y}$.

The components of the drag force are thus

$$F_{\text{drag},x} = -B_2 v v_x \quad (2.19)$$

$$F_{\text{drag},y} = -B_2 v v_y.$$

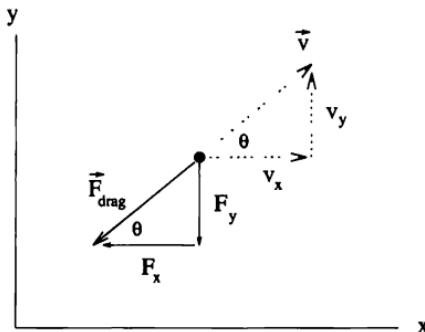


FIGURE 2.3: Components of the force due to air resistance on an object moving with a velocity \vec{v} . The direction of the drag force is opposite to \vec{v} .

Adding this force to the equations of motion leads to

$$\begin{aligned} x_{i+1} &= x_i + v_{x,i} \Delta t & (2.20) \\ v_{x,i+1} &= v_{x,i} - \frac{B_2 v v_{x,i}}{m} \Delta t \\ y_{i+1} &= y_i + v_{y,i} \Delta t \\ v_{y,i+1} &= v_{y,i} - g \Delta t - \frac{B_2 v v_{y,i}}{m} \Delta t . \end{aligned}$$

Below we give the subroutine (in pseudocode form) that does this calculation. The time evolution according to (2.20) is continued until y_{i+1} becomes negative, which means the shell struck the ground somewhere during the previous time step. We then exit the loop and interpolate between the last two calculated positions to estimate where the shell struck the ground.

EXAMPLE 2.2 Subroutine calculate for the cannon shell projectile using the Euler Method

- Store position as (x_i, y_i) and velocity as $(v_{x,i}, v_{y,i})$.
- For each time step i calculate position and velocity at step $i + 1$:

```

▷  $x_{i+1} = x_i + v_{x,i} \Delta t$ 
     $y_{i+1} = y_i + v_{y,i} \Delta t$ 
▷ Compute air drag force
 $F_{\text{drag},x} = -(B_2/m)v v_{x,i}$ 
 $F_{\text{drag},y} = -(B_2/m)v v_{y,i}$ 
```

$$\begin{aligned}\triangleright v_{x,i+1} &= v_{x,i} + (F_{\text{drag},x}/m)\Delta t \\ \triangleright v_{y,i+1} &= v_{y,i} + (-g + F_{\text{drag},y}/m)\Delta t\end{aligned}$$

\triangleright Stop when $y_{i+1} \leq 0$

- Estimate landing point by interpolating between the last point with $y > 0$ and the point that would have been “below ground.”
-

In this pseudocode we have kept the velocity components as arrays. However, we could just as well have kept only the “current” value of v_x and v_y as we looped through the calculation. Also, we suggest that you estimate the landing point of the shell (ℓ) by interpolating between the last point above ground (n) and the point that would have been below ground ($n+1$). If we let $r = -y_n/y_{n+1}$ then a linear interpolation gives

$$x_\ell = \frac{x_n + rx_{n+1}}{r+1}, \quad (2.21)$$

and $y_\ell = 0$.

Results for the trajectory are shown in Figure 2.4, where we have assumed an initial velocity of 700 m/s, which is appropriate for a large cannon shell. The plot on the left shows the results without air resistance,¹⁰ while on the right we show results with air resistance for $B_2/m = 4 \times 10^{-5}$ (m⁻¹), which is a value appropriate for a very large cannon shell.¹¹ There are two important points to note from these results. First, air resistance decreases the range *a lot*, approximately by a factor of 2. The second point concerns the behavior as a function of firing angle. You probably have learned in your introductory mechanics course that without air resistance the maximum range occurs for $\theta = 45^\circ$, and this is indeed observed in Figure 2.4. However, when air resistance is included as above, the maximum range occurs at a lower firing angle. For the parameters we have used here, the largest range with atmospheric drag is obtained with a firing angle of approximately 37° . This is in accord with our intuition that the longest range is obtained by shooting low into the “wind.” In this example, it is easy to estimate the launch angle that maximizes the range by simply repeating calculations at successively larger angles until the maximum range is found. However, in general, locating such a maximum (or minimum) can pose quite a numerical challenge. Appendix B describes some methods for dealing with this challenge.

Our calculations clearly show that air resistance plays a major role in the problem. However, there is another important piece of physics that we have not yet accounted for. From Figure 2.4 we see that the shell travels to a very high altitude, where the air density will be lower than at sea level. We saw in our

¹⁰For this case the trajectory can also be calculated analytically. We will leave it to the reader to verify that the trajectories we have obtained numerically agree with the analytic results.

¹¹See Chapter 5 of Bennett (1976) for a description of the cannon for which this value of B_2 was measured. The shells were approximately 10 cm in diameter. We emphasize that this value of B_2 was measured via “experimental” observations, and not estimated using the value $C = 1$ in (2.9). We will see in the next section that the drag force for objects that are moving very rapidly, such as cannon shells and baseballs, is more complicated than implied by (2.9).

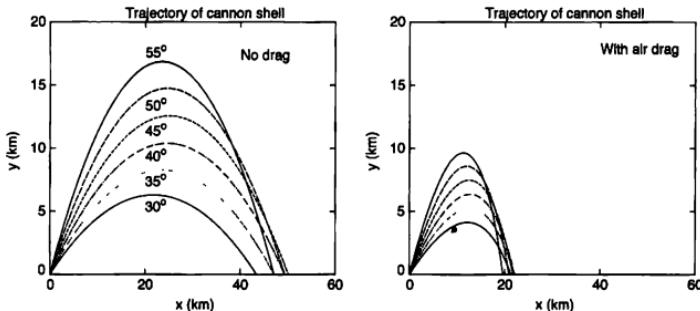


FIGURE 2.4: Left: trajectory of a cannon shell without air resistance. We assumed an initial speed of 700 m/s, and the firing angles are indicated. Right: trajectory of the same cannon shell, but now with air resistance included, with $B_2/m = 4 \times 10^{-5} \text{ m}^{-1}$. Note that the x and y scales are the same in (a) and (b), and that the firing angles used in (b) are also the same.

discussion of the drag on a bicycle that the force from air resistance is proportional to the density of the air, so the drag force at high altitudes will be less than that at sea level. To investigate the magnitude of this effect, we need to know how the air density depends on the altitude. This brings the statistical and thermal physics of the atmosphere into what had been a kinematic problem (albeit with our inclusion of air drag).

So how does the air density vary as a function of the altitude? This is a problem that is discussed in many texts on statistical mechanics (such as the one by Kittel and Kroemer in the references). The simplest approximation is to treat the atmosphere as an *isothermal* (constant temperature) ideal gas. One then finds that the pressure p depends on altitude according to

$$p(y) = p(0)e^{-mgy/k_B T} \quad (2.22)$$

where m is the average mass of an air molecule, y is the height from some reference point (say, the sea level), k_B is the Boltzmann's constant, and T is the absolute temperature. For an ideal gas the pressure is proportional to the density, so this leads to

$$\rho = \rho_0 \exp(-y/y_0), \quad (2.23)$$

where $y_0 = k_B T/mg \approx 1.0 \times 10^4 \text{ m}$, and ρ_0 is density at sea level ($y = 0$).

This isothermal model of atmosphere is perhaps not very realistic, since we know that the air temperature can vary quite a bit over altitude changes of a few kilometers (e.g., at the top of Mount Everest). A different approach is to assume that the air is a poor conductor of heat, and that convection is very slow. This leads to the so-called *adiabatic* approximation, which is much better for the troposphere (altitudes up to 10 km or so). Interested readers can find discussions of this case

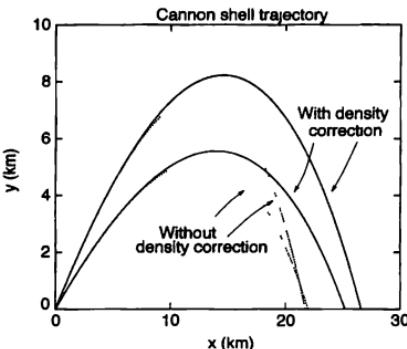


FIGURE 2.5: Trajectory of a cannon shell with (solid curves) and without (dotted curves) the effect of the lower air density at high altitudes taken into account. In all cases the air resistance was included with $B_2/m = 4 \times 10^{-5} \text{ m}^{-1}$, and the initial speed was 700 m/s. The lower two curves were for a firing angle of 35°, while for the top curves it was 45°. Note that the x and y scales are different.

in the book by Fermi (1937). The adiabatic approximation leads to a somewhat different dependence of the density on altitude

$$\rho = \rho_0 \left(1 - \frac{ay}{T_0}\right)^\alpha, \quad (2.24)$$

where $a \approx 6.5 \times 10^{-3} \text{ K/m}$ fits measured data fairly well. Here T_0 is the sea level temperature (in K), and the exponent $\alpha \approx 2.5$ for air. Comparing (2.23) and (2.24), we can expect substantial differences if y becomes a significant fraction of y_0 .

Whichever model of the air density we decide to use, however, the basic strategy of the numerical calculation is the same. The drag force due to air resistance is proportional to the density, so

$$F_{\text{drag}}^* = \frac{\rho}{\rho_0} F_{\text{drag}}(y=0), \quad (2.25)$$

where $F_{\text{drag}}(y=0)$ is the force at sea level [given by (2.17)], and F_{drag}^* is the drag force at altitude. Putting this into the calculation is straightforward; we replace B_2 in (2.20) with $B_2\rho/\rho_0$. Some results are shown in Figure 2.5 where we compare trajectories calculated using the isothermal model for ρ (2.23), with the results found assuming a constant density. We see that for this cannon, the decrease of ρ at high altitudes increases the range by about 20 percent. In addition, the maximum range now occurs for a firing angle slightly *larger* than 45°. Hence, to get the longest range for a projectile such as this, it is best to take advantage of the reduced drag at high altitudes.

EXERCISES

- 2.6. Use the Euler method to calculate cannon shell trajectories ignoring both air drag and the effect of air density (actually, ignoring the former automatically rules out the latter). Compare your results with those in Figure 2.4, and with the exact solution.
- 2.7. Use the *adiabatic* model of the air density (2.24) to calculate the cannon shell trajectory, and compare with the results found using (2.23). Also, one can further incorporate the effects of the variation of the ground temperature (seasonal changes) by replacing B_2 by $B_2^{ref} (T_0/T_{ref})^{\alpha}$, where B_2^{ref} is the value of B_2 at a reference temperature T_{ref} and T_0 is the actual ground temperature. The value quoted in the text is appropriate for $T = 300$ K. In particular, how much effect will the adiabatic model have on the maximum range and the launch angle to achieve it? How much do they vary from a cold day in winter to a hot summer day?
- 2.8. In our model of the cannon shell trajectory we have assumed that the acceleration due to gravity, g , is a constant. It will, of course, depend on altitude. Add this to the model and calculate how much it affects the range.
- 2.9. Calculate the trajectory of our cannon shell including both air drag and the reduced air density at high altitudes so that you can reproduce the results in Figure 2.5. Perform your calculation for different firing angles and determine the value of the angle that gives the maximum range.
- *2.10. Generalize the program developed for the previous problem so that it can deal with situations in which the target is at a different altitude than the cannon. Consider cases in which the target is higher and lower than the cannon. Also investigate how the minimum firing velocity required to hit the target varies as the altitude of the target is varied.
- *2.11. In warfare you generally want to hit a particular target (as opposed to having the cannon shells land indiscriminately). However, this is not easy, since very small changes in any of the parameters can lead to large changes in the landing site of the shell. Investigate this by calculating how much the range of the cannon shell considered in this section would change if the initial speed is increased by 1 percent. Also compute the change in the range if there is a slight (10 km/h) wind (this effect can be added using the methods developed in the next section). You should find that even these relatively small changes alter the landing site by quite significant amounts.
- *2.12. Add the effect of the Earth's revolution about its own axis, that is, consider the *Coriolis force*.

2.3 BASEBALL: MOTION OF A BATTED BALL

The game of baseball has fascinated fans of the game, and also physicists (some of us are both), for many years. There have probably been more books and papers written on the physics of baseball than any other sport, which is perhaps justified considering the wide variety of physical principles that are important in different aspects of the game. In this section we will consider the trajectory of a batted ball. Our goal is to understand how far a *real* baseball should be expected to travel when hit by a typical power hitter. The most important ingredient in our model will be the effect of atmospheric drag, and we will employ a somewhat more realistic model of the drag force than was used in previous sections. We will also see that

wind and altitude have smaller, but quite significant, effects on the game. In the next few sections we will depart from our usual exclusive use of SI units and use also the units of feet and miles per hour (mph), which are so closely tied with how many of us usually think about the sport.

The basic equations of motion for a baseball are the same as those of the cannon shell, (2.20), and we will again use the Euler method in our simulations. Before we go too far, however, it is interesting to estimate the distance a baseball would travel in a vacuum, that is, without atmospheric drag. This range can be calculated analytically, or numerically, by using (2.20) with $B_2 = 0$. A good power hitter can give the ball an initial speed of about 110 mph (49 m/s), and if it is hit at an initial angle of 45° starting from a height of 1 m, the range in a vacuum would be 248 m, or approximately 815 ft. So far as we know, no one has ever hit a baseball that far (except, perhaps, in the presence of a tornado). A typical outfield fence is 350–400 ft from home plate, and from practical experience we know that a 500-ft home run is an exceptionally long one.¹² Thus, it is clear that baseball in a vacuum would be a very different game than the one we are used to watching.

The force on a baseball due to air resistance is given by a form similar to (2.9). However, wind tunnel measurements with real baseballs show that the drag coefficient C is actually a strong function of v , as illustrated in Figure 2.6, which shows a plot of the drag coefficient as a function of the speed of the ball.¹³ At low speeds, C is close to $\frac{1}{2}$, which is not far from the value of unity predicted by the very simple argument we gave in the derivation of (2.9). However, as the speed increases, C drops substantially, and it is more than a factor of 2 smaller at high speeds. This does *not* mean that the drag force is lower at high speeds; F_{drag} is proportional to the product Cv^2 , and this quantity¹⁴ is a monotonically increasing function of v .

The behavior of C can be understood qualitatively as follows. At low speeds the air flow around the ball is “well behaved” and orderly. However, at high speeds the flow becomes turbulent, and it turns out that an object is able to “slip” through the air more easily in this case than when the flow is nonturbulent. This makes the drag coefficient smaller at high speeds when the flow pattern is turbulent, than at low speeds when it is nonturbulent. Such behavior is quite general and is important for other objects as well (we will consider the interesting case of a golf ball later in this chapter). We see from Figure 2.6 that this transition occurs at a speed of about 80 mph for an ordinary baseball. This is a common speed for both a batted and a pitched ball, so the velocity dependence of C must be considered in any realistic modeling. The figure also shows the variation of C for balls that are rougher or smoother than typical baseballs, and the behavior is quite different in these cases.

¹²Chapter 4 of Adair (1990) has a nice discussion of the distances traveled by the longest known home runs.

¹³Note that this plot copies results from the first edition of Adair’s book. Later editions of this book give a slightly different variation of C , but these do not lead to significantly different results for the range of a batted baseball.

¹⁴A plot of F_{drag} as a function of velocity is given in Chapter 2 of Adair (1990) (it could also be derived from the data given in Figure 2.6). While it does not appear to happen in the case of a baseball, it is possible that for certain objects the drag force might become smaller as the speed is increased. Some of the consequences of such behavior are discussed by Adair.

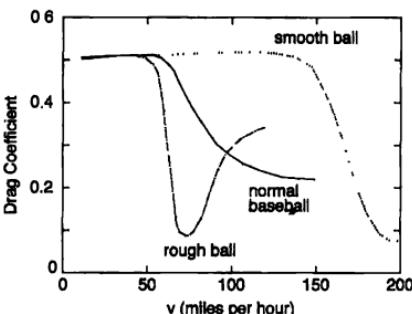


FIGURE 2.6: Variation of the drag coefficient, C , with velocity for normal, rough, and smooth baseballs. Note that for a normal ball the drag force is equal to the weight of the ball at a velocity of approximately 98 mph. After R. K. Adair, *The Physics of Baseball*, 1st Edition.

For a rough ball the transition to turbulent flow occurs at a lower velocity than for a regular ball, so the drop in C takes place sooner, while the opposite trend is observed for a smooth ball (a ball without stitches would qualify as smooth). This will turn out to have important implications for the motion of the pitch known as a knuckleball, since the stitches on the ball are equivalent to a sort of roughness.

In order to calculate the trajectory of a baseball, we need to solve the Euler equations (2.20), including the velocity dependence of the drag force. The variation of C observed in Figure 2.6 is clearly a complicated function. For our numerical work it is convenient to have an (admittedly approximate) analytic representation of this function. Taking the results for the drag coefficient from Figure 2.6 and adding in the appropriate factors of air density, and so on, it turns out that the drag factor for a normal baseball is described reasonably well by the function (now using SI units)

$$\frac{B_2}{m} = 0.0039 + \frac{0.0058}{1 + \exp[(v - v_d)/\Delta]}, \quad (2.26)$$

with $v_d = 35$ m/s and $\Delta = 5$ m/s. Using this parameterization of the drag we can construct a program to calculate the trajectory of our batted baseball. The program is very similar to the one given earlier for the cannon shell problem, so we will leave the details to the exercises.

A typical result for the trajectory is shown as the solid curve in Figure 2.7. Recalling that the range in a vacuum would be more than 800 ft, we see that air resistance has an enormous effect. The range of our power hitter is now approximately 120 m, or about 395 ft, which is in much better accord with the size of typical major league ballparks.¹⁵ It turns out that with air resistance described

¹⁵In writing a program to determine the range, it is advisable to locate the place where the ball strikes the ground by using an interpolation procedure, similar to that used in our work on the

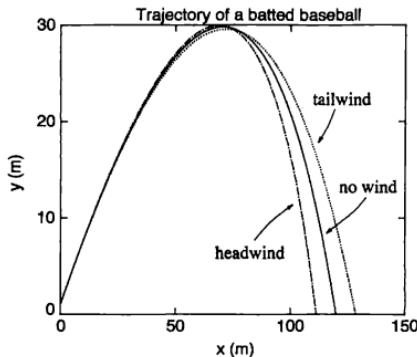


FIGURE 2.7: Calculated trajectory of a baseball hit at an initial velocity of 110 mph, with the effects of atmospheric drag included. Solid curve no wind; dotted and dot-dashed curves a tailwind and headwind of 10 mph. In all cases we assume that the initial velocity makes an angle of 35° with the horizontal. The other parameters used in the calculation are given in the text.

by (2.26), the maximum range is largest for angles near 35°. This is much lower than the 45° angle that gives the maximum range without air resistance. The trajectory is seen to be noticeably nonparabolic¹⁶ (in a vacuum it would be an exact parabola), which is also in agreement with our everyday experience.

Let us next add the effect of the wind. We will assume that it is blowing in a horizontal (x) direction and has a constant magnitude and direction during the flight of the ball.¹⁷ In this case the components of the drag force become

$$\begin{aligned} F_{\text{drag},x} &= -B_2 |\vec{v} - \vec{v}_{\text{wind}}| (v_x - v_{\text{wind}}) \\ F_{\text{drag},y} &= -B_2 |\vec{v} - \vec{v}_{\text{wind}}| v_y , \end{aligned} \quad (2.27)$$

where v_{wind} is the velocity of the wind, and a positive value corresponds to a tailwind. The derivation of (2.27) is understood most easily if you consider the drag force (2.19) in the reference frame at rest with respect to the wind. In that frame the drag force is just (2.19). Transforming into the batter's reference frame then requires that we subtract the velocity of the wind from \vec{v} , which yields (2.27).

cannon shell problem [see (2.21)]. The program “knows” that the ball has landed when $y_n < 0$. You could just take x_n to be the range, but this would overestimate the landing point by as much as $v_{x,n}\Delta t$, which can be significant when you want to consider small differences in the range. A better method is to interpolate between (x_n, y_n) and (x_{n-1}, y_{n-1}) , assuming the trajectory to be a straight line, to get a much improved estimate of where $y = 0$.

¹⁶This can be seen by noting that the ball reaches its maximum height at about $x = 70$ m while it strikes the ground at $x = 120$ m in the absence of any wind.

¹⁷It is not difficult to treat the case of a swirling wind like that found in most stadiums, but we'll leave that for the interested student. Such an effect would certainly be important in developing a model of judging and catching a fly ball.

This transformation also means that B_2 in (2.27) is calculated from (2.26) with v equal to the velocity of the ball relative to the wind.

We can include the wind in our program using (2.27), an exercise we will leave to the reader. Some results from such a calculation are shown in Figure 2.7, where we compare the trajectories calculated with a tail wind and a head wind (both 10 mph) with the result for still air. This relatively modest breeze alters the range by approximately ± 25 ft, which is more than enough to turn a routine fly ball out into a home run, or a home run into a long out. The wind can thus have a pronounced effect on the game, as any fan who has been to Wrigley Field or Candlestick Park can verify.¹⁸ It is interesting that major league parks are oriented so that the generally prevailing westerly breeze, which averages about 5 mph in the United States, is at the batter's back and thus favors a home run. As in most sports, offense is favored over defense; that is, the hitter over the pitcher.

It is also interesting to consider the effect of altitude. This can be added to the model just as we did in the case of the cannon shell, with the only difference being that we don't have to worry about density changes during the flight of the ball. The results for the range of our home-run hitter at several altitudes are shown in Table 2.1. The highest major league park is currently in Denver (altitude ≈ 5280 ft). Our long fly ball will travel about 31 ft farther there than at sea level. This is a substantial difference and suggests that home-run hitters should enjoy playing in Denver. The next highest major league park is in Atlanta (altitude ≈ 1000 ft), and here the effect of the altitude is to add about 6 ft to the range. We also note that if a ballpark is ever built on Pike's Peak, home runs there will be very interesting to watch.

TABLE 2.1: Calculated range of a batted ball at various altitudes. In all cases we have assumed an initial velocity of 110 mph, that there is no wind, and that the ball is hit at an initial angle of 35° with respect to the horizontal. (Pike's Peak is a mountain in Colorado.)

Location	Altitude (feet)	Range (feet)
sea level	0	390
Atlanta	1,000	396
Denver	5,280	421
Pike's Peak	14,110	473

One of the conclusions that can be drawn from our results for the range of a batted baseball is that atmospheric drag simply must be taken into account to get an accurate picture of the range and trajectory of a fly ball. Air resistance reduces the range of a home-run hitter by more than a factor of 2. For the statisticians, it seems clear that playing in Denver will significantly enhance a hitter's record and be unfair to a pitcher's earned-run average.¹⁹ There are a number of other effects that can be added to our model, and a nice qualitative discussion of many of

¹⁸These two baseball parks are notorious for being very windy. Note that Candlestick Park is now named 3Com Park. Alas, the Giants no longer use this ballfield.

¹⁹Although the reduced density at altitude will add slightly to the velocity of a fastball.

them is given by Adair (1990). One more comment concerning the “philosophy” of numerical simulations is appropriate here. It is not realistic to expect our model to be accurate in a truly quantitative sense. There are just too many complications. Besides the approximation involved in our parameterization of the drag factor, we must expect that this force will vary somewhat from ball to ball, and that it will also be a function of the humidity and other similar factors. Factors such as these will almost certainly add or subtract a few feet, or perhaps more, to the range. The usefulness of our model is *not* that we can use it to calculate the range of a home run with great precision. Rather, it should give us *insight* into the relative importance of different effects, and enable us to estimate changes in the range as a result of the wind, or other factors. We will have more to say concerning the philosophy of model building in physics during the course of this book. That said, the predictions of our model are certainly consistent with what is observed on the ballfield.

EXERCISES

- 2.13. Construct a program to calculate the trajectory of a baseball, as in Figure 2.7. Use it to investigate the following questions:
 - (a) Calculate the range at sea level, with no wind, of a ball hit at 110 mph for different initial angles. Determine to within 1° the angle that gives the maximum range
 - (b) Determine the initial angles that give the maximum range for a ball hit at 110 mph into a 25-mph head wind, and with a 25-mph tail wind. What are the maximum ranges in the two cases?
 - (c) In all of our calculations we have assumed that the initial velocity is 110 mph, a value typical for a good power hitter. How much does the range depend on this value? Calculate the range for initial velocities of 120 and 100 mph. How does the range scale with the initial kinetic energy of the ball and why?
 - (d) Calculate how much a fastball slows down on its way to home plate. Assume a pitch that leaves the pitchers hand at 100 mph and find its speed when it crosses home plate, which is 60.5 ft away.
- *2.14. Consider the effect of a crosswind on the trajectory of a fly ball. How much will a wind of 10 mph directed at right angles to the initial velocity alter the place where one of the fly balls in Figure 2.7 lands?
- *2.15. Calculate the range for the smooth and rough balls described by the drag functions in Figure 2.6.
- 2.16. Estimate the initial speed required to hit a home run 550 ft at sea level in the absence of any wind. How far would this ball travel on Pike’s Peak?

2.4 THROWING A BASEBALL: THE EFFECTS OF SPIN

To treat the problem of a thrown or pitched ball, we must deal with two different effects. One effect is the spin of the ball; this will turn out to dominate the motion of a curve ball. The other effect concerns the difference in the drag coefficients for rough and smooth balls, which we encountered in Figure 2.6, and this will be crucial for the behavior of a knuckleball. We begin with the curve ball.

The origin of the force that makes a spinning ball curve can be appreciated if we recall that the drag force has the form $F_{\text{drag}} \sim v^2$, where v is the speed of the object relative to the air. For a ball spinning about an axis perpendicular to

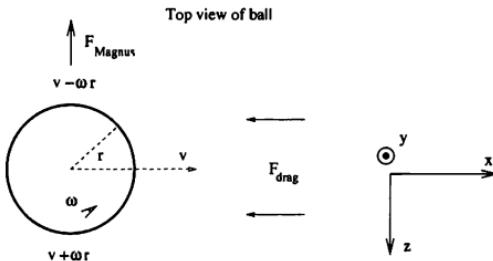


FIGURE 2.8: Forces acting on a spinning ball. The ball is moving from left to right, and the rotation axis is directed along y , out of the page. The lower edge of the ball has a speed relative to the air of $v + \omega r$, which is larger than that of the top edge, $v - \omega r$, where v is the relative speed of the center of mass of the ball with respect to the air just around the ball. (We assume that this speed is essentially equal to the horizontal speed of the ball relative to the ground here, but in practice, the viscous drag of air might reduce it.) So the drag force on the bottom part of the ball is larger. This figure corresponds to a sidearm curve ball (see Figure 2.9) as viewed from above. After Adair (1990).

the direction of travel, this speed will be different on opposite edges of the ball, as illustrated in Figure 2.8.

Because of the spin, the edge of the ball at the bottom in Figure 2.8 will have a larger velocity relative to the air than will the edge of the ball at the top of the figure. This will result in a larger drag force on the ball where the velocity is highest than on the opposite edge. This drag force will (due to the curved face of the ball) have a component that is directed towards the top in Fig. 2.8, since the air is pushed away partly in the $+z$ direction. Likewise the drag force on the upper edge of the ball in Fig. 2.8 will have a component that is directed downward in the figure. When these forces on the two sides of the ball are then added, there will be a component of the net force in the $-z$ direction (upward in Figure 2.8), perpendicular to the (center of mass) velocity. This is known as the Magnus force and is believed to make an important contribution for objects such as baseballs.²⁰

We expect this force to be proportional to B_2 in (2.17), but the precise value of the coefficient of proportionality is hard to estimate since it is determined from averaging the appropriate component of the drag force over the curved face of the

²⁰According to what is known as the Kutta-Joukowski theorem, the Magnus force arises from the relative motion of air around the ball, just as found for an airplane wing due to the air flow around the wing. Thus as long as there is a net air circulation around the moving ball, there will be a Magnus force however viscous (or not) the air might be, and however smooth or rough the surface of the ball might be. Also, the Bernoulli effect is sometimes mentioned as the origin of the spin-dependent force on baseballs. See the discussion by Adair (1990) for more on both this and the Magnus force. The bottom line is that there are several different contributions to the spin-dependent force on a baseball. One of them is explained in Fig. 2.8, and others are discussed in the references. However, in the end, they all have the functional form given in (2.29), and since we will take the coefficient S_0 from experiments, we have effectively included all of these different forces.

ball. Rather than trying to perform this averaging theoretically, we will simply argue for the general functional form of this force, and then resort to experimental measurements to determine the overall magnitude. To understand how it depends on the angular velocity ω of the ball, we first note that the upward component of the drag force on the lower half of the ball will be proportional to the square of the velocity of this surface of the ball relative to the air, i.e., $(v + r\omega)^2$. The downward component of the drag force on the upper half of the ball should then be proportional to $(v - r\omega)^2$. The Magnus force is equal to the difference between these two terms²¹

$$F_M \propto (v + r\omega)^2 - (v - r\omega)^2 \sim vr\omega . \quad (2.28)$$

Thus the net spin-dependent force have the general form²²

$$F_M = S_0 \omega v_x . \quad (2.29)$$

The numerical coefficient S_0 then takes care of averaging the drag force over the face of the ball, as just discussed; we will determine its value from experimental measurements. Since the drag coefficient of a baseball depends on its speed (see Figure 2.6), we must also expect that S_0 will depend on v . However, for simplicity we will assume that S_0 is a constant in the following. Over the velocity range that is usually of interest to a pitcher, 50–110 mph, we estimate from the data given in Adair (1990) that $S_0/m \approx 4.1 \times 10^{-4}$, where $m = 149$ g is the mass of the ball (note that S_0/m is unitless).²³ For a typical curve ball, the magnitude of the Magnus force (2.29) is about one-third the weight of the ball.

To calculate the trajectory of a curve ball, we have to consider motion in three dimensions. We will let x be the axis running from home plate to the pitcher, z be the horizontal direction perpendicular to x , and y be the height above the ground,

²¹This assumes that the two terms have the same coefficient; i.e., that the two surfaces of the ball are equally “smooth”. For a rapidly spinning ball, any differences in the smoothness (i.e., due to the stitch pattern on the ball) will be effectively averaged out.

²²The general vector form is $\vec{F}_M = S_0 \vec{\omega} \times \vec{v}$

²³In a sense we have swept all of our ignorance, that is, the complicated problem of averaging F_{drag} over the curved face of the ball, into the constant S_0 . By then obtaining S_0 from experiment, you might conclude that we are giving in to the problem, and in a sense we are. However, even in taking this approach it still seems fair to say that we “understand” the physics contained in S_0 ; it is only that the mathematics required to calculate its value is more complicated than we wish to tackle at this time.

as in Figure 2.8. The equations of motion for a sidearm curve ball are then

$$\begin{aligned}\frac{dx}{dt} &= v_x \\ \frac{dv_x}{dt} &= -\frac{B_2}{m} v v_x \\ \frac{dy}{dt} &= v_y \\ \frac{dv_y}{dt} &= -g \\ \frac{dz}{dt} &= v_z \\ \frac{dv_z}{dt} &= -\frac{S_0 v_x \omega}{m}.\end{aligned}\tag{2.30}$$

Here we assume that the axis of rotation is parallel to y , that is, perpendicular to the ground. These equations of motion include the effects of atmospheric drag on the largest component of the velocity (v_x), with a velocity dependent coefficient (2.26), but we haven't included it for v_y or v_z , since the forces are much smaller in these cases.

We can again use the Euler method to solve (2.30), and some typical results are shown in Figure 2.9. Here and below we assume that ω is a constant; that is, the ball's rotation rate does not decrease significantly during the course of the pitch. The ball begins on a trajectory that would carry it over the center of the plate ($z = 0$), but the Magnus force deflects it by nearly a foot, which would take it well outside. The dotted curve shows how much the pitch falls during its trip to the plate. We have taken the initial velocity to be purely horizontal, and it turns out that the vertical deflection due to gravity is so great that it would cause this pitch to bounce in front of the plate. One conclusion here is that gravity has a significant effect on the trajectory, which in this case is larger than the effect of the spin. The fact that a curve ball is so much harder to hit than a fastball thrown at the same velocity implies that a hitter is able to estimate very rapidly and accurately the deflection due to gravity. This is probably because the trajectories of all fastballs are basically similar.

In Figure 2.10 we show the trajectories of several other pitches. The solid line shows an overhand curve. The axis of rotation is now horizontal, so that the Magnus force is (to a good approximation) vertical. The dotted curve shows the trajectory of a similar pitch, the only difference being this one is *not* spinning, so the deflection due to the Magnus force alone is now evident. The additional break due to the Magnus force is seen to be a little less than 1 ft, which is similar to that found for the sidearm curve ball. Note that we have given these two pitches a small, upward, initial velocity so that unlike the example in Figure 2.9, they cross the plate in or near the strike zone. Also shown as the dot-dashed curve in Figure 2.10 is the trajectory of a 95-mph fastball. This has much less downward

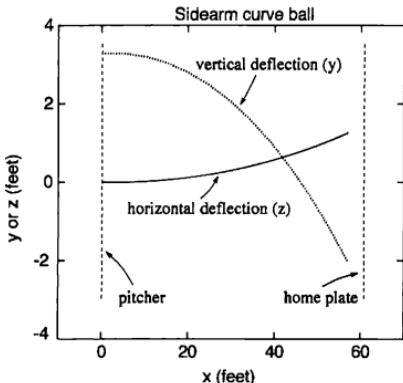


FIGURE 2.9: Trajectory of a sidearm curve ball. The pitch traveled from left to right, with the y axis vertical. The x axis is horizontal, and perpendicular to the line between the pitcher and home plate. The solid curve shows the horizontal (z) displacement, which is due to the Magnus force, while the dashed curve shows the vertical (y) displacement, which is due solely to gravity. The ball was thrown with an initial velocity of 70 mph in the z direction, and was spinning at 30 revolutions/s, with $\vec{\omega}$ parallel to the y axis

deflection due to gravity, since it travels to the plate in about 70 percent of the time taken in the other cases.

We next consider one of most entertaining pitches, the knuckleball. Unlike other pitches, this one is thrown with very little spin. Even so, its trajectory typically exhibits one or two rapid and large displacements in directions perpendicular to the line connecting the pitcher with home plate (the z direction). These displacements are different from pitch to pitch, making it extremely difficult for the hitter to judge where to swing in order to make contact with the ball. In fact, the motion is so irregular that even the pitcher cannot predict exactly how the pitch will move or curve.

The origin of the lateral force on a knuckleball can be appreciated from Figure 2.6. Suppose that a moving ball is not spinning at all and is oriented such that a stitch is exposed on one side while the other side is smooth (no stitches exposed). Since the drag force is greater for a smooth ball than for a rough one, there will be an imbalance of forces on the two sides of the ball, giving a net force in the direction of the rough side. It turns out that this force is large enough to make the pitch curve as much or more than a typical curve ball. Now, if the ball rotates just a little as it moves toward home plate, so that the exposed stitch moves to the opposite side of the ball, the lateral force will reverse direction and the ball will then curve in the *opposite* direction. This is a knuckleball.

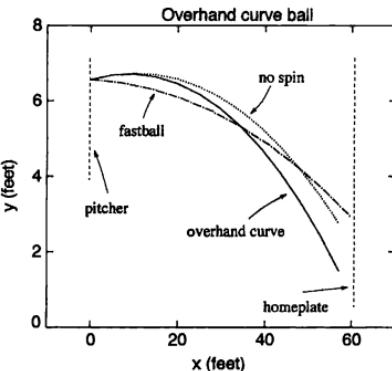


FIGURE 2.10: Trajectories of several pitches. The pitches travel from left to right, and for these pitches the deflection of the ball is purely in the vertical (y) direction. The solid curve shows the break of an overhand curve ball ($\vec{\omega}$ is along $-z$), which is due to both gravity and the Magnus force. We have assumed the same velocity and the same angular velocity as with the sidearm curve in Figure 2.9. The dotted curve shows the break due to gravity alone; that is, we have assumed a pitch thrown at the same velocity, but with no spin. The dot-dashed curve shows the trajectory of a 95-mph fastball for comparison.

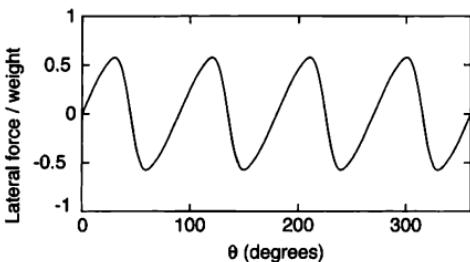


FIGURE 2.11: Approximate horizontal force on a baseball traveling without spinning at a speed of 65 mph as a function of the angular orientation of the ball. The angle θ is measured about an axis that is vertical, and perpendicular to the center of mass velocity of the ball, which is horizontal. The four oscillations of the force for each complete revolution of the ball correspond to the four times the stitches pass by an observation point. These results are estimated from the data given by Watts and Sawyer (1975). Here we have expressed the force in terms of the weight of the ball, and scaled it as appropriate for a pitch thrown at 65 mph (the measurements were performed at a lower speed).

A curve ball is difficult to throw, since it requires the pitcher to impart a large amount of spin on the ball. At the other extreme, a knuckleball is difficult to throw since it requires the pitcher to impart *very little* spin on the ball. A well-thrown knuckleball might complete only half of a revolution on its way to the plate, so a stitch (i.e., rough surface) will be exposed for a significant amount of time.²⁴

This force due to the effective roughness of the ball could in principle be estimated from the drag coefficients in Figure 2.6, but it is clearly preferable to obtain it from specially designed experiments. Fortunately Watts and Sawyer (1975) have performed wind tunnel measurements of this force, and a schematic of their results is shown in Figure 2.11. As anticipated, the lateral force is a function of the angular orientation of the ball. In this case the angle is measured about an axis such that stitches pass by any one point four times for each complete revolution of the ball. Hence there are four maxima and four minima in the lateral force for each revolution ($\Delta\theta = 360^\circ$). The magnitude of this force is approximately one-half the weight of the ball at a speed of 65 mph. In addition to the magnitude of the force, the non-sinusoidal shape of the curve is also worth noting (it is not a plotting error!). The rapid variation of the force at certain angles no doubt contributes to the irregular trajectory of this pitch. To calculate the trajectory of a knuckleball, we treat it as a projectile moving horizontally with a drag coefficient given by (2.26), and assume that it spins very slowly. We now have to keep track of the angular orientation of the ball, and we do this with the equation of motion $d\theta/dt = \omega$. For simplicity we will assume that the ball spins about a vertical axis so that the deflection due to the force in Figure 2.11 will be purely horizontal. Thus, the rotation angle θ is

²⁴If the ball completes too many (several) rotations on the way to the plate, this lateral force will oscillate rapidly in direction, and cancel out the effect.

measured with respect to this vertical axis. This leads to another Euler equation

$$\theta_{i+1} = \theta_i + \omega \Delta t, \quad (2.31)$$

with ω taken as constant during the course of the pitch (but adjustable from one pitch to the next). As the ball travels to home plate, θ varies with time, thereby altering the lateral force. The force in Figure 2.11 is given approximately by the function

$$\frac{F_{\text{lateral}}}{mg} = 0.5 [\sin(4\theta) - 0.25 \sin(8\theta) + 0.08 \sin(12\theta) - 0.025 \sin(16\theta)]. \quad (2.32)$$

This expression is simply an empirical choice²⁵ that has a form close to the wind tunnel results of Watts and Sawyer (1975). Putting this all together, the trajectory can now be calculated. Two parameters control the behavior, the angular velocity of the ball, ω in (2.31), and its initial angular orientation. Some typical trajectories calculated for $\omega = 0.2$ rev/s are shown in Figure 2.12, where we have considered three different initial orientations. Our model reproduces the expected behavior very well. Typical deflections are a foot or more, and are comparable to or even larger than those found for curve balls. The multiple deflections in the trajectory make it no surprise that catchers do not enjoy trying to stay in front of these pitches. The trajectories become more complicated if the rotation rate is increased, but the deflections are smaller since the direction of the lateral force then varies more rapidly.

EXERCISES

- 2.17. Investigate the trajectories of knuckleballs as a function of the angular velocity ω , the initial angular orientation, and the (center of mass) velocity.
- 2.18. Calculate the effect of backspin on a fastball. How much does an angular velocity of 1000 rpm (typical for a fastball) affect the trajectory?
- 2.19. Model the effect of backspin on the range of a batted ball. Assume an angular velocity of 2000 rpm.
- 2.20. Calculate the effect of the knuckleball force on a batted ball. Assume that the ball is a line drive hit at an initial speed of 90 mph and an angle of 20° and that the ball does not spin at all (clearly a gross assumption). Let the rough side of the ball always face one side, which is perpendicular to the direction the ball is hit, and use Figure 2.11 to estimate the magnitude of the lateral force. With all of these assumptions, calculate the lateral deflection of the ball. Such balls hit with very little spin are observed by outfielders to effectively "flutter" from side to side as they move through the air.
- *2.21. Calculate the trajectory of a batted ball hit with side spin. That is, let the rotation axis be vertical, corresponding to a ball that is hit so that it "hooks" or "slices". This is commonly encountered in a ball hit near one of the foul lines. Calculate how much a spin angular velocity of 2000 rpm would cause a line drive to curve. Is this consistent with your experiences? If not, calculate what angular velocity would be required.

²⁵Our choice of this particular function to describe the data of Watts and Sawyer (1975) was motivated by the form of a Fourier series that has a period of $\pi/2$. For more on this topic see Appendix C.

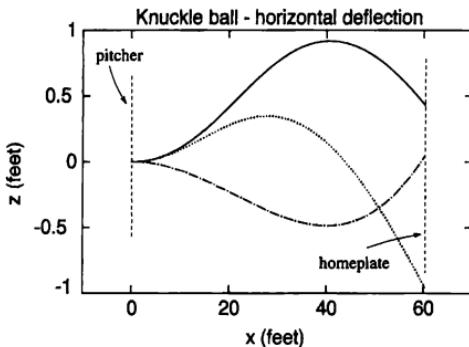


FIGURE 2.12: Calculated horizontal deflections of several knuckleballs, traveling from left to right. All had a velocity of 65 mph, with an angular velocity of 0.2 revolutions/s (with $\vec{\omega}$ parallel to the vertical direction, y). The different pitches had different initial orientations of the stitches

- *2.22. Estimate the value of S_0 in (2.29) by averaging F_{drag} as given in Figure 2.6 over the face of the ball. Don't be afraid to make some approximations, such as assuming that one side of the ball is completely rough while the other is smooth, in the sense of Figure 2.6. You might also want to perform your averaging numerically, using the integration methods discussed in Chapters 5 and 7.
- *2.23. Estimate the velocity dependence of S_0 from the data given in Adair (1990). Use your results to calculate what effect this has on the trajectory of a typical curve ball.

2.5 GOLF

We conclude this chapter with a treatment of the motion of a golf ball. While all of the ingredients we will need for our simulations have already been touched on in our discussions of cannon shells and baseballs, there are still some interesting lessons to be learned.

We again use a coordinate system in which y is the height of the ball above the ground, x is the horizontal direction in the same plane as the ball's initial velocity, and z horizontal and perpendicular to the velocity. The equations of motion for the golf ball are then

$$\begin{aligned}\frac{dv_x}{dt} &= - \frac{F_{\text{drag},x}}{m} - \frac{S_0 \omega v_y}{m} \\ \frac{dv_y}{dt} &= - \frac{F_{\text{drag},y}}{m} + \frac{S_0 \omega v_x}{m} - g,\end{aligned}\tag{2.33}$$

with the usual equations for dx/dt and dy/dt . Here we have assumed that the ball is hit with backspin with an angular velocity of ω , so that the spin axis is

along z and thus perpendicular to both x and y . We have encountered all of these variables before, including the atmospheric drag, F_{drag} , and the Magnus force, which is proportional to the product of ω and the velocity. The magnitudes of these forces have been estimated from measurements with real golf balls. The drag force is given by an expression of the form (2.9), and it turns out that the drag coefficient C varies with velocity in a manner that is qualitatively similar to what we observed for a baseball in Figure 2.6. At low speeds $C \approx \frac{1}{2}$, but this coefficient drops sharply with increasing speed, with C measured to be $\approx 7.0/v$ (with v in m/s) at high speeds (see Erlichson [1983]). We will include the behavior in these two limits in our modeling by assuming that

$$F_{\text{drag}} = -C \rho A v^2, \quad (2.34)$$

with $C = \frac{1}{2}$ for speeds up to $v = 14$ m/s, and $C = 7.0/v$ at higher velocities. Our drag force will thus be a continuous function of v , although it will have a discontinuous derivative (dF_{drag}/dv) at $v = 14$ m/s. This behavior is qualitatively similar to that of a baseball, but the transition to a reduced drag coefficient, that is, to turbulent flow, occurs at a much smaller velocity.

A golf ball is hit with a significant amount of backspin, and the associated Magnus force gives a large upward (vertical) force. In fact, the initial Magnus force can actually be larger than the force of gravity. This is apparent to any (physics) student of golf, since the initial trajectory of a well-hit drive has upward curvature. Using the measurements discussed by Erlichson, we estimate $S_0 \omega/m \approx 0.25 \text{ s}^{-1}$ for a typical case. It is believed to be a good approximation to assume that ω does not change significantly over the course of the trajectory. In our simulations we will assume that $\vec{\omega}$ is constant (in both magnitude and direction), so the Magnus force then varies only as \vec{v} changes.

The equations of motion (2.33) can (yet again) be solved with the Euler method. The programming is similar to what we have encountered previously in this chapter, so we will leave it to the exercises. Some results are shown in Figure 2.13, where we have assumed an initial velocity of 70 m/s (about 230 feet/s), which is a typical value for an average player. The maximum range is approximately 215 m (approximately 235 yards), in reasonable accord with our expectations, so the model seems to capture the essential physics. It turns out that the maximum range occurs with an initial angle of only 9° , much reduced from the 45° that would be found for a simple projectile in a vacuum. This shows the importance of the Magnus force.

We also consider several hypothetical cases in Figure 2.13. If we increase the spin on the ball by 50 percent, as might occur with an iron shot, for example, the trajectory is much higher even for our 9° initial angle, although for the parameters considered here the range is almost unchanged. If we put no spin on the ball, the range drops dramatically to about 112 m, again assuming an initial angle of 9° . Thus, as any golfer knows, the spin is extremely important.

It is especially interesting to consider the case of a smooth ball, that is, a ball *without* dimples. This can be modeled with a constant drag coefficient, $C = \frac{1}{2}$,

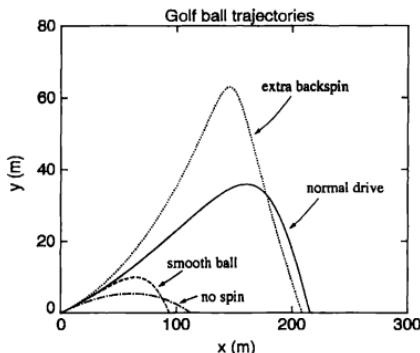


FIGURE 2.13: Several calculated trajectories of golf balls. In all cases the initial velocity was 70 m/s, at an angle 9° from the horizontal. Solid curve: result for a normal drive (the calculation includes the effect of air resistance and backspin), dotted curve same as a normal drive, but with 50 percent more backspin to provide more lift, dot-dashed curve, without any backspin; dashed curve, normal amount of backspin, but for a smooth ball.

independent of the speed.²⁶ This case has the shortest range of all, only about 94 m. (Note that we have still assumed that the ball is spinning, with the same Magnus force, the same value of S_0 , as above.) Now we can understand why a golf ball has dimples. They cause the transition to turbulent flow to occur at very low velocities, with a corresponding drop in the drag coefficient. This, in turn, allows a golf ball to be hit much farther than would otherwise be possible (thereby inflating the ego of the golfer). While it is probably too late to change such a familiar aspect of the game, the use of a smooth ball would have some advantages. For example, golf courses could be made much smaller and the players wouldn't have to walk as far.

EXERCISES

- 2.24.** Model the behavior of a Ping-Pong ball. The drag force can be estimated from (2.9) with $C = \frac{1}{2}$ (since the ball is smooth). The Magnus acceleration of a Ping-Pong ball is $\approx S_0 \vec{\omega} \times \vec{v}/m$, where $S_0/m = 0.040$ (in SI units, with ω in rad/s and v in m/s; see the discussion in Bennett [1976]). Assume $v = 3$ m/s, and angular velocities in the range 1–10 revolutions per second (better yet, try to measure ω in your own experiment).

²⁶This amounts to assuming that the transition to turbulent flow occurs at a velocity above that used in our calculation, 70 m/s. As with baseballs, we would expect that even for a smooth ball there must eventually be a transition to turbulent flow, but we do not know of any accurate studies of where this transition occurs. Thus, our modeling of a “smooth” golf ball is less quantitative than our simulations for a ball with dimples. Even so, these uncertainties do not affect our main conclusion, which is that the dimples substantially increase the range of the ball.

- 2.25. Calculate the trajectory for a hooking or slicing golf ball by giving the ball sidespin, with the same angular velocity used in the calculations in Figure 2.13.
- 2.26. Investigate the trajectory of the topspin lob in tennis. Topspin is also important for ground strokes and serves. See Štěpánek (1988) for a discussion of the relevant parameters.
- 2.27. Calculate the effects of air drag, wind, or spin (or all three) for your favorite sports projectile. De Mestre (1990) gives a handy table of the relevant parameters for a wide variety of objects.

REFERENCES

- [1] R. K. Adair, 1990, *The Physics of Baseball*, Harper and Row, New York, and also Physics Today, May 1995, p. 26. For the arm-chair physicist baseball fan.
- [2] A. Armenti, Jr., 1992, *The Physics of Sports*, American Institute of Physics, New York. A useful collection of articles on a wide variety of sports.
- [3] W. R. Bennett, Jr., 1976, *Scientific and Engineering Problem-Solving with the Computer*, Prentice-Hall, Englewood Cliffs. An entertaining discussion of trajectories in a windy environment plus more.
- [4] N. De Mestre, 1990, *The Mathematics of Projectiles in Sport*, Cambridge University Press, Cambridge.
- [5] H. Erlichson, "Maximum Projectile Range with Drag and Lift, with Particular Application to Golf," Am. J. of Phys. **51**, 357 (1983). Essential numbers for the drag and Magnus forces on a golf ball. Although his interpretation is a bit different than ours, the results are the same.
- [6] E. Fermi, 1937, *Thermodynamics*, Dover, New York.
- [7] C. Kittel and H. Kroemer, 1980, *Elementary Statistical Physics*, Freeman, San Francisco.
- [8] A. Štěpánek, "The Aerodynamics of Tennis Balls—the Topspin Lob," Am. J. of Phys. **56**, 138 (1988). A scholarly look at one of the authors' favorite games.
- [9] R. G. Watts and E. Sawyer, "Aerodynamics of a Knuckleball," Am. J. of Phys. **43**, 960 (1975). Very useful wind tunnel results for the forces on a knuckleball.

Oscillatory Motion and Chaos

Examples of oscillatory phenomena can be found in many areas of physics, including the motion of electrons in atoms, the behavior of currents and voltages in electronic circuits, and planetary orbits. Perhaps the simplest mechanical system that exhibits such motion is a pendulum, consisting of a mass that is connected by a string to some sort of support so that it is able to swing freely in response to the force of gravity. In the idealized case, ignoring friction and assuming that the angle the string makes with the vertical is small, such a pendulum undergoes what is known as simple harmonic motion. The main features of this motion are common to many types of oscillating systems, making it a paradigm in elementary physics textbooks. However, elementary-level treatments usually do not consider the behavior of a *real* pendulum, that is, a pendulum that has some friction, that is driven by an external force, and that is allowed to swing to large angles. The motion of such a pendulum exhibits many features not found in the simple harmonic case. Perhaps the most important of these features is the possibility of chaotic behavior, which is the central topic of this chapter.

In the next several sections we will explore some of the interesting effects that occur in real oscillatory systems. We begin with a simple pendulum and consider how to treat simple harmonic motion numerically. We then generalize our pendulum model to include the effects of friction and nonlinearities and find that they give rise to the possibility of chaotic behavior. While the term chaos probably has some intuitive meaning for all of us, it is not easy to give a precise definition of this notion (especially one that would make a physicist or a mathematician happy). We will, therefore, spend some time in formulating such a definition, and this will lead us to consider several key issues. The remainder of the chapter is devoted to the study of a variety of other systems that exhibit chaotic behavior.

3.1 SIMPLE HARMONIC MOTION

One example of a simple pendulum is a particle of mass m connected by a massless string to a rigid support. We let θ be the angle that the string makes with the vertical and assume that the string is always taut, as in Figure 3.1. We also assume that there are only two forces acting on the particle, gravity and the tension of the string. It is convenient to consider the components of these forces parallel and perpendicular to the string. The parallel forces add to zero, since we assume that the string doesn't stretch or break, while the force perpendicular to the string is given by

$$F_\theta = -m g \sin \theta, \quad (3.1)$$

where g is the acceleration due to gravity, and the minus sign reminds us that the force is always opposite to the displacement from the vertical, where $\theta = 0$.

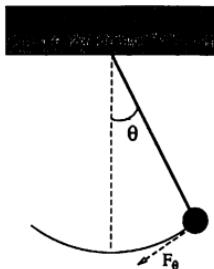


FIGURE 3.1: A simple pendulum, consisting of a mass connected by a string to a fixed point on a support, such as a ceiling.

Newton's second law tells us that this force is equal to the mass times the acceleration of the particle along the circular arc that is the particle's trajectory, that is, $F_\theta = m d^2 s / dt^2$. The displacement along this arc is $s = \ell \theta$, where ℓ is the length of the string. If we now assume that θ is always small so that $\sin \theta \approx \theta$, we obtain the (probably familiar) equation of motion

$$\frac{d^2 \theta}{dt^2} = -\frac{g}{\ell} \theta. \quad (3.2)$$

This is the central equation of simple harmonic motion. It is easy to verify that it has the general solution

$$\theta = \theta_0 \sin(\Omega t + \phi), \quad (3.3)$$

where $\Omega = \sqrt{g/\ell}$, and θ_0 and ϕ are constants that depend on the initial displacement and velocity of the pendulum. We see that the motion of a simple pendulum is indeed very simple. The oscillations are sinusoidal with time and continue forever without decaying, which is not surprising since there is no friction in the model. The oscillations have an angular frequency, Ω , which is a function of the length of the string (and g , of course), but is independent of m and the amplitude of the motion.

We now consider a numerical approach to this problem. You might wonder why this is necessary, since the analytic solution is already in hand. However, this is just a warm-up exercise, as we will soon be turning our attention to slightly more complicated oscillatory problems for which exact results are not available. Our basic equation of motion is the second-order differential equation (3.2), which we want to solve for θ as a function of t . It is convenient to rewrite this as two first-order differential equations

$$\begin{aligned} \frac{d\omega}{dt} &= -\frac{g}{\ell} \theta, \\ \frac{d\theta}{dt} &= \omega, \end{aligned} \quad (3.4)$$

where ω is the angular velocity of the pendulum. This should look familiar, as we took a very similar approach in the projectile problems in Chapter 2. There we used the Euler method to solve a similar set of equations, so let us consider the same algorithm here. We convert (3.4) into difference equations, using a time step Δt so that time is discretized with $t = i\Delta t$, where i is an integer. Letting θ_i and ω_i be the numerically approximated angular displacement and velocity of the pendulum at time step i , the equations in (3.4) become

$$\begin{aligned}\omega_{i+1} &= \omega_i - \frac{g}{\ell} \theta_i \Delta t, \\ \theta_{i+1} &= \theta_i + \omega_i \Delta t.\end{aligned}\quad (3.5)$$

These are closely analogous to the difference equations we obtained when using the Euler method in Chapters 1 and 2. As in those problems, we use θ_i and ω_i (i.e., values at time step i) to estimate the values at step $i+1$, then repeat the process for steps $i+2, i+3, \dots$

In constructing a program to implement this approach, we can use arrays to hold the values of θ , ω , and t , or we might store these results in a file as they are calculated. Either way, the heart of the calculation is the subroutine `euler_calculate` given below. This routine computes θ and ω with the Euler method in (3.5).

EXAMPLE 3.1 Pseudocode for subroutine `euler_calculate`

- For each time step i calculate ω and θ at time step $i+1$.

```

▷  $\omega_{i+1} = \omega_i - (g/\ell)\theta_i \Delta t$ 
▷  $\theta_{i+1} = \theta_i + \omega_i \Delta t$ 
▷  $t_{i+1} = t_i + \Delta t$ 
▷ Repeat for the desired number of time steps.
```

Some results from a program based on the Euler method are shown in Figure 3.2. The behavior we see there is very peculiar; while the motion is basically oscillatory, the amplitude of the oscillations *grows* with time. This is contrary to the exact solution, (3.3), and should also be at odds with your intuition. Since the model does not contain any source of energy nor does it include any friction, the total energy of the pendulum should remain constant. However, our program doesn't seem to appreciate this fact!

It turns out that the difficulty lies with our use of the Euler method. As was emphasized when we first introduced this method in Chapter 1, it provides us with an approximate estimate of the solution to our differential equation, (3.2). Again, the emphasis is on the word *approximate*. We have already mentioned that the errors involved in using the Euler method become smaller as the time step is made smaller. You might, therefore, think that the erroneous behavior seen in Figure 3.2 could be corrected by simply using a smaller value of Δt in our program. Reducing Δt will indeed make the errors smaller, but it turns out that the energy of the pendulum will increase with time for *any* nonzero value of Δt .

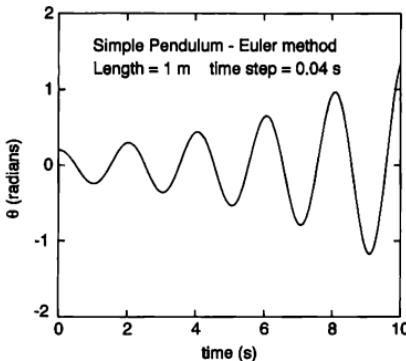


FIGURE 3.2: θ as a function of time for our simple pendulum, calculated using the Euler method with program `pendulum`. Here, and in most of the other figures in this chapter, we have connected the calculated values with solid lines.

At this point you might sense a contradiction. If the errors get smaller when Δt is reduced, how can the calculated behavior be defective for all values of Δt ? The results in Figure 3.2 show that for our Euler solution both the amplitude of the oscillations and the total energy increase with time. If the calculation were repeated with a smaller value of Δt (a job we will leave for the exercises), the rate of this increase would be smaller. However, for all nonzero values of Δt we would find that the energy increases with time, and hence the Euler solution is, in that sense, always incorrect.

This is our first encounter with a numerical method that is inherently unstable. To see how this instability comes about we consider the total energy of the pendulum. This energy, E , is given by

$$E = \frac{1}{2}m\ell^2\omega^2 + mg\ell(1 - \cos\theta). \quad (3.6)$$

The first term is the kinetic energy, and is just $\frac{1}{2}mv^2$ where $v = \omega\ell$. The second term in (3.6) is the gravitational potential energy, and is equal to mgh where h is the height of the mass above the lowest point on its trajectory. In the limit of small θ the energy reduces to

$$E = \frac{1}{2}m\ell^2(\omega^2 + \frac{g}{\ell}\theta^2). \quad (3.7)$$

Substituting ω_{i+1} and θ_{i+1} from the Euler method (3.5) into ω and θ in (3.7), the Euler result for the energy at time t_{i+1} is

$$E_{i+1} = E_i + \frac{1}{2}mg\ell \left(\omega_i^2 + \frac{g}{\ell}\theta_i^2 \right) (\Delta t)^2. \quad (3.8)$$

Since the second term on the right is always positive, the energy of the Euler “solution” *always* increases with time, no matter how small we make Δt .

But if the Euler method is unstable in this way, how did we manage to get away with using it for the problems in Chapters 1 and 2? The answer to this is that a method can only be termed “suitable” or “unsuitable” (i.e., “good” or “not good”) in the context of a particular problem. In the problems treated in Chapters 1 and 2 the Euler method also yielded (in most cases) solutions that did not quite conserve energy.¹ However, in those situations the errors were negligible on the scales relevant for those particular problems. In contrast, for problems involving oscillatory motion we often want to consider the behavior for many periods of oscillation. In order to be useful in such cases, a numerical method must conserve energy over the long haul.² The Euler method is not a good choice for these types of problems.

We are thus led to consider other methods for solving ordinary differential equations. This is such an important class of problems that we have devoted Appendix A to a discussion of several different numerical approaches, including the Runge-Kutta and Verlet methods. While those work well in dealing with the oscillatory problems encountered in this chapter, it turns out that a simple modification of the Euler method yields an algorithm that is also quite suitable. This modification yields what is known as the Euler-Cromer method, as Cromer was the first to carefully discuss the algorithm and show why it works. To appreciate the Euler-Cromer method, we reconsider our pendulum program. Given below is a *modified* version of the `calculate` subroutine.

EXAMPLE 3.2 Subroutine `euler_cromer_calculate` with the Euler-Cromer method

- For each time step i calculate ω and θ at time step $i + 1$.
 - ▷ $\omega_{i+1} = \omega_i - (g/\ell)\theta_i\Delta t$
 - ▷ $\theta_{i+1} = \theta_i + \omega_{i+1}\Delta t$ (Note that ω_{i+1} is used to calculate θ_{i+1} .)
 - ▷ $t_{i+1} = t_i + \Delta t$
 - ▷ Repeat for the desired number of time steps.

The *only* change in this subroutine is emphasized by the added note in the above pseudocode, and can be appreciated as follows. With the Euler method, the *previous* value of ω and the *previous* value of θ are used to calculate the new values of both ω and θ . However, with the Euler-Cromer method, the *previous* values of ω and θ are used to calculate the new value of ω , but the *new* value of ω is used to calculate the new value of θ . This is such a minor change in the algorithm that you are probably surprised that it makes any difference. Indeed, for

¹A notable exception was the constant acceleration problem considered in Chapter 1, for which the Euler method gives the exact solution.

²More precisely, the numerical algorithm must *itself* not contribute or remove energy from the system.

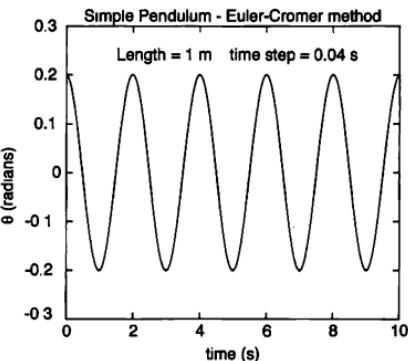


FIGURE 3.3: θ as a function of time for a simple pendulum calculated using the Euler-Cromer method.

many problems it makes no significant difference. However, for problems involving oscillatory motion the Euler-Cromer method conserves energy over each complete period of the motion,³ and thus avoids the difficulties we found in Figure 3.2. Figure 3.3 shows a stable oscillation obtained with the Euler-Cromer algorithm (the parameters used here were the *same* as those used in the calculation shown in Figure 3.2). This numerical solution is very stable; these calculated oscillations would persist until our patience runs out. There are other numerical methods besides the Euler-Cromer algorithm that yield accurate solutions for oscillatory problems. Appendix A contains a description and discussion of the advantages and disadvantages of several other methods.

EXERCISES

- 3.1. Numerically investigate the stability of the Euler-Cromer method. Calculate the total energy, kinetic plus potential, of the pendulum as a function of time. Show that the energy is conserved over each complete cycle of the motion.
- 3.2. Repeat the previous problem using the Runge-Kutta method described in Appendix A. Compare the accuracy of the Runge-Kutta method with that of the Euler-Cromer algorithm using the same time step.
- 3.3. Use the Euler method to simulate the motion of a pendulum as in Figure 3.2. Study the behavior as a function of the step size, Δt , and show that the total energy always increases with time.
- 3.4. For simple harmonic motion, the general form for the equation of motion is

$$\frac{d^2x}{dt^2} = -k x^\alpha, \quad (3.9)$$

³For more on this see the exercises.

with $\alpha = 1$. This has the same form as (3.2), although the variables have different names. Begin by writing a program that uses the Euler-Cromer method to solve for x as a function of time according to (3.9), with $\alpha = 1$ (for convenience you can take $k = 1$). The subroutine described in this section can be modified to accomplish this. Show that the period of the oscillations is independent of the amplitude of the motion. This is a key feature of simple harmonic motion. Then extend your program to treat the case $\alpha = 3$. This is an example of an *anharmonic oscillator*. Calculate the period of the oscillations for several different amplitudes (amplitudes in the range 0.2 to 1 are good choices), and show that the period of the motion now depends on the amplitude. Give an intuitive argument to explain why the period becomes longer as the amplitude is decreased.

- *3.5. For the anharmonic oscillator of (3.9) in the previous exercise, it is possible to analytically obtain the period of oscillation for general values of α in terms of certain special functions. Do this, and describe how the relationship between the period and the amplitude depends on the value of α . Can you give a physical interpretation to the finding? Hint: If you multiply both sides of (3.9) by $\frac{dx}{dt}$ you can integrate with respect to t . This then leads to a relation between the velocity and x .

3.2 MAKING THE PENDULUM MORE INTERESTING: ADDING DISSIPATION, NONLINEARITY, AND A DRIVING FORCE

Let us now consider how we might make the simple pendulum in the previous section a bit more realistic. The equation of motion (3.2) describes a frictionless pendulum, so we begin by adding some damping. The manner in which friction enters the equations of motion depends on the origin of the friction. Possible sources of friction include the effective bearing where the string of the pendulum connects to the support, air resistance, etc. In many cases this damping force is proportional to the velocity, and that is the assumption we will make here.⁴ The frictional force we will employ thus has the form $-q(d\theta/dt)$, since the velocity of the pendulum is $\ell(d\theta/dt)$. Here q is a parameter that measures the strength of the damping, and the minus sign guarantees that this force will always oppose the motion of the pendulum. Thus, the equation of motion for our damped pendulum has the form

$$\frac{d^2\theta}{dt^2} = -\frac{q}{\ell}\theta - q\frac{d\theta}{dt}, \quad (3.10)$$

where the second term on the right models the friction. Since this differential equation is still linear, we can solve it analytically just as we solved (3.2). The details of the solution can be found in many standard mechanics texts (such as the one by Marion and Thornton in the references), but in short, there are three regimes of distinct physical behavior. The first regime, called *underdamped*, occurs for sufficiently small friction, where the solution

$$\theta(t) = \theta_0 e^{-qt/2} \sin(\sqrt{\Omega^2 - q^2/4} t + \phi) \quad (3.11)$$

⁴Note, however, that other functional forms are possible. For example, in the case of air resistance, we noted in Chapter 2 that while the drag from air resistance is proportional to v for very small velocities, it varies as v^2 in many cases of practical interest. We will leave the investigation of the behavior with other forms for F_{friction} to the inquisitive reader.

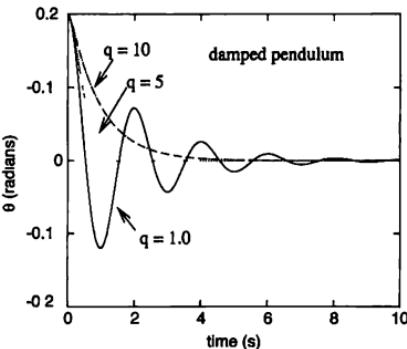


FIGURE 3.4: θ as a function of time for a damped pendulum for several different values of the damping, q , calculated using the Euler-Cromer method. Here we have taken $g = 9.8$ and $\ell = 1.0$. For $q = 5$ the system is close to being critically damped.

shows an oscillatory behavior with a frequency $\sqrt{\Omega^2 - q^2/4}$, where $\Omega = \sqrt{g/\ell}$, and an amplitude that decays with time. At the other extreme, when the damping is large, the behavior is *overdamped*. Here the solution is

$$\theta(t) = \theta_0 e^{-(q/2 \pm \sqrt{q^2/4 - \Omega^2}) t}, \quad (3.12)$$

which is a monotonic, exponential decay of θ . At the boundary between the underdamped and overdamped regimes, the pendulum is *critically damped* with

$$\theta(t) = (\theta_0 + Ct)e^{-qt/2}. \quad (3.13)$$

Numerical results for $\theta(t)$ in all three cases are shown in Figure 3.4.

While the behavior of our damped pendulum is a bit different from the undamped case considered in the previous section, the results are still not very exciting. We therefore consider the addition of a driving force to the problem. The form of this force will depend on how the force is applied. A convenient choice is to assume that the driving force is sinusoidal with time, with amplitude F_D and angular frequency Ω_D (which is not to be confused with the natural frequency of the simple pendulum, Ω). This might arise, for example, if the pendulum mass has an electric charge and we apply an oscillating electric field. This leads to the equation of motion

$$\frac{d^2\theta}{dt^2} = -\frac{g}{\ell}\theta - q\frac{d\theta}{dt} + F_D \sin(\Omega_D t), \quad (3.14)$$

where the last term represents the external driving force. This driving force will pump energy into (or out of) the system, and the externally imposed frequency,

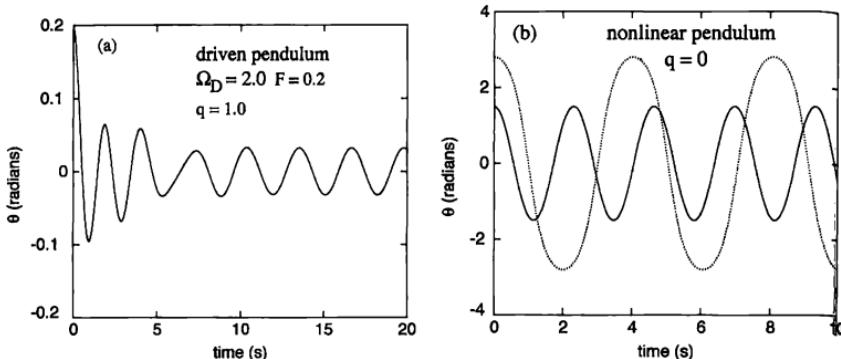


FIGURE 3.5: (a) θ as a function of time for a driven pendulum. (b) θ as a function of time for an undamped pendulum, for two different values of $\theta(0)$. Note that the period is significantly longer when $\theta(0)$ is increased (the dotted curve). For both calculations we used the Euler-Cromer method, with $g = 9.8$ and $\ell = 1.0$.

Ω_D , will “compete” with the natural frequency of the pendulum, Ω , so one might hope to find much richer behavior than we have seen so far.

Alas, it turns out that (3.14) can also be solved analytically, and the behavior is again not very exciting. The steady state solution is

$$\theta(t) = \theta_0 \sin(\Omega_D t + \phi), \quad (3.15)$$

where the amplitude θ_0 is given by

$$\theta_0 = \frac{F_D}{\sqrt{(\Omega^2 - \Omega_D^2)^2 + (q \Omega_D)^2}}. \quad (3.16)$$

This behavior is shown in Figure 3.5(a), which gives some numerical results obtained with the Euler-Cromer method.

The driven, damped pendulum thus undergoes a simple harmonic oscillation with the angular frequency of the driving force, so in some sense its behavior is even simpler than the non-driven case. There is an interesting situation, though, when the driving frequency Ω_D matches the natural angular frequency of the pendulum Ω . In this case of *resonance*, the amplitude θ_0 can become large, especially if the friction is small.

So far, we have assumed that the amplitude of the oscillation is small, and this allowed us to expand the $\sin \theta$ term in (3.1). We will now not make this assumption; this will enable us to treat situations in which the mass swings to large angles or

even all the way around the pivot point of the pendulum.⁵ Therefore, we now consider the equation of motion

$$\frac{d^2\theta}{dt^2} = - \frac{g}{l} \sin \theta. \quad (3.17)$$

That is, we consider a nonlinear pendulum without friction and without a driving force. Since there is no means of adding to or removing energy from this system, the total mechanical energy is conserved and the pendulum executes a periodic motion. Its period, however, is no longer independent of amplitude. This can be demonstrated analytically, but the mathematics is not simple.⁶ However, a numerical approach is straightforward, and some results are shown in Figure 3.5(b). Here we have started our pendulum with some initial displacement (some initial angle $\theta(0)$), and then let it go ($\omega(0) = 0$). The motion is periodic, but it is no longer described by a simple sine or cosine function. You will also notice that (as we just mentioned) the period now depends on the amplitude. When the amplitude is large, the pendulum spends a longer time at its turning points (the points where θ is largest), and this makes the period longer than found for small amplitudes.

All of these results for a more realistic pendulum are in accord with our intuition of how a pendulum should behave. The real fun begins when we put all three of these effects — damping, a driving force, and the nonlinearity — together at the same time. This will be the subject of the next section.

EXERCISES

- 3.6. Investigate the three regimes of linear pendulum with dissipation (3.10) using the Euler-Cromer (or any other suitable) method. In all three regimes, the dissipation due to friction leads to an exponential damping of the overall energy. Locate the boundary between the overdamped and underdamped regimes numerically, and compare with the analytic result (you can either derive the latter yourself, or find it in a mechanics text such as Marion and Thornton).
- 3.7. Numerically investigate the linear, forced pendulum with friction of (3.14). Show numerically the existence of the resonance, and confirm the dependence of the resonant amplitude on the driving angular frequency Ω_D , and on the friction parameter q .
- 3.8. In the nonlinear pendulum of (3.17), use the Euler-Cromer or another suitable method to investigate the relationship between the amplitude and period numerically. Can you give an intuitive argument supporting your results?

⁵To allow such motion in a real pendulum we would have to replace the string in Figure 3.1 with a rigid rod.

⁶This problem can be solved using the conservation of energy and an approach similar to that given in the hint for the anharmonic oscillator of Exercise 3.5. Interested readers can find details in Marion and Thornton (1995). It turns out that the solution involves *elliptic integrals*. These are well-known special functions, but are nonetheless not further reducible in terms of elementary functions. In particular, the period of this system can be expressed as $4\sqrt{l/g}K(\sin(\theta_m/2))$, where θ_m is the maximum angle made by the pendulum and $K(a) = \int_0^{\pi/2} \frac{dx}{\sqrt{1-a^2 \sin^2 x}}$ is the complete elliptic integral of the first kind. Its values can be either found in a table (such as Abramowitz and Stegun in the references), or evaluated numerically using some of the methods discussed in Appendix E. Elliptic integrals occur fairly frequently in physics, particularly in problems related to electricity and magnetism.

3.3 CHAOS IN THE DRIVEN NONLINEAR PENDULUM

Now that we have a numerical method that is suitable for various versions of the simple pendulum problem, and armed with some understanding of what might or might not happen when dissipation, an external driving force, and/or nonlinearity is present, we are ready to take on a slightly more complicated and also more interesting situation. That is, we add all three ingredients which were previously only discussed separately. First, we do not assume the small-angle approximation, and thus do not expand $\sin \theta$ term in (3.1). Second, we include friction of the form $-q(d\theta/dt)$. Third, we add to our model a sinusoidal driving force $F_D \sin(\Omega_D t)$. Putting all of these ingredients together, we have the equation of motion⁷

$$\frac{d^2\theta}{dt^2} = -\frac{g}{\ell} \sin \theta - q \frac{d\theta}{dt} + F_D \sin(\Omega_D t). \quad (3.18)$$

We will call this model for a nonlinear, damped, driven pendulum, (3.18), the *physical pendulum*. It contains some very rich and interesting behavior. We will only be able to touch on a few of its intriguing properties here, although you can explore others through the exercises. Let us begin by examining the behavior of θ as a function of time for several typical cases. First we must construct a program to calculate a numerical solution, since there is no known exact solution to (3.18). Our program is similar in form to the one we used to study the simple pendulum. The only major difference is that we must use a slightly more complicated equation for ω . We again rewrite (3.18) as two first-order differential equations and obtain

$$\begin{aligned} \frac{d\omega}{dt} &= -\frac{g}{\ell} \sin(\theta) - q \frac{d\theta}{dt} + F_D \sin(\Omega_D t), \\ \frac{d\theta}{dt} &= \omega. \end{aligned} \quad (3.19)$$

These can be converted into difference equations for θ_i and ω_i as we did in (3.5). These difference equations can then be translated into a program. Below we sketch the subroutine that does the work of calculating θ and ω .

EXAMPLE 3.3 Subroutine euler_cromer_calculate for the physical pendulum model

- For each time step i (beginning with $i = 1$), calculate ω and θ at time step $i + 1$.
 - ▷ $\omega_{i+1} = \omega_i - [(g/\ell) \sin \theta_i - q\omega_i + F_D \sin(\Omega_D t_i)] \Delta t$
 - ▷ $\theta_{i+1} = \theta_i + \omega_{i+1} \Delta t$
 - ▷ If θ_{i+1} is out of the range $[-\pi, \pi]$, add or subtract 2π to keep it in this range.

⁷Note that, strictly speaking, the parameter F_D in (3.18) is not the actual force since other factors of m and l also enter. However, F_D is proportional to the driving force, so when we speak of larger drive forces this will mean larger values of F_D , etc.

- ▷ $t_{i+1} = t_i + \Delta t$
 - ▷ Repeat for the desired number of time steps.
-

This subroutine is organized much like the `euler_cromer_calculate` routine for our simple pendulum program, but there are several differences of note. First, the equation for ω_{i+1} is more complicated since we have a different equation of motion. Second, we adjust the value of θ after each iteration so as to keep it always between $-\pi$ and π . Recall that our pendulum can't now swing all the way around its pivot point, which corresponds to $|\theta| > \pi$. Since θ is an angular variable, values of θ that differ by 2π correspond to the *same* position of the pendulum. This is done because for plotting purposes it is convenient to keep θ in the range $-\pi$ to π . If θ becomes less than $-\pi$, then its value is increased by 2π ; likewise, if it becomes greater than $+\pi$, its value is decreased by 2π . This procedure⁸ keeps $-\pi \leq \theta \leq +\pi$. Finally, note that we again use the Euler-Cromer method, but the Runge-Kutta or Verlet methods would also be suitable.

Some typical results for θ and ω as functions of time, as calculated with our program, are shown in Figure 3.6 where we plot the behavior for several different values of the driving force, with all of the other parameters held fixed. With a driving force of zero, the motion is damped and the pendulum comes to rest after at most a few oscillations. These damped oscillations have a frequency close to the natural frequency of the undamped pendulum, Ω , and are a vestige of simple harmonic motion and its damped cousin (3.10). With a small driving force, $F_D = 0.5$, we find two regimes. The first few oscillations are affected by the decay of an initial transient as in the case of no driving force. That is, the initial displacement of the pendulum leads to a component of the motion that decays with time and has an angular frequency of $\sim \Omega$. After this transient is damped away, the pendulum settles into a steady oscillation in response to the driving force. The pendulum then moves at the driving frequency, Ω_D , *not* at its natural frequency, with an amplitude determined by a balance between the energy added by the driving force and the energy dissipated by the damping. This is very similar to the driven, damped, but linear oscillator of (3.14). In a sense, the motion of the pendulum can be viewed as an interplay of the two frequencies Ω and Ω_D , the natural frequency of the pendulum and the frequency of the driving force.

The behavior changes radically when the driving force is increased to $F_D = 1.2$. Now the motion is no longer simple, even at long times. The vertical jumps in θ are due to our resetting of the angle to keep it in the range $-\pi$ to π and thus correspond to the pendulum swinging "over the top." To make this clear, the center plot in Figure 3.6 shows $\theta(t)$ for $F_D = 1.2$, with and without this resetting of the angle. We see that the pendulum does not settle into any sort of repeating steady-state behavior, at least in the range shown here. We might suspect that we have not waited long enough for the transients to decay and that a steady oscillation might be found if we simply waited a little longer. This is not the case; for this

⁸Later on, we will see that this resetting of θ is also useful when we examine the frequency spectra of these and similar results.

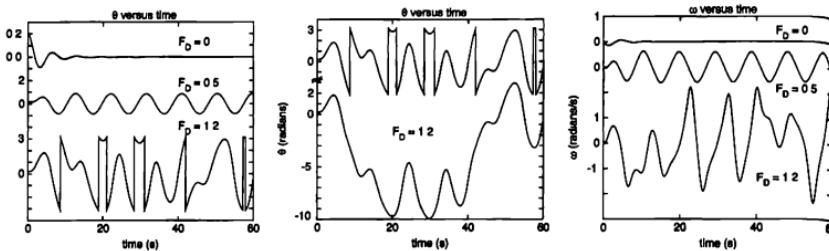


FIGURE 3.6: Left: behavior of θ as a function of time for our driven, damped, nonlinear pendulum, for several different values of the driving force. The vertical “jumps” in θ occur when the angle is reset so as to keep it in the range $-\pi$ to $+\pi$; they do not correspond to discontinuities in $\theta(t)$. Center: behavior of $\theta(t)$ for $F_D = 1.2$ with and without these “resets”. Right: corresponding behavior of the angular velocity of the pendulum, ω . The parameters for the calculation were $q = 1/2$, $\ell = g = 9.8$, $\Omega_D = 2/3$, and $dt = 0.04$, all in SI. The initial conditions were $\theta(0) = 0.2$ and $\omega(0) = 0$.

value of the driving force the behavior *never* repeats. This is an example of *chaotic* behavior, which will be our main concern for the rest of this chapter.

It is important to appreciate the behavior illustrated in Figure 3.6. At low drive the motion is a simple oscillation (after the transients have decayed), which would, if we were sufficiently patient, repeat forever. On the other hand, at high drive the motion is chaotic; it is a very complicated nonrepeating function of time. But what does it really mean to be chaotic? Your intuition probably tells you that chaotic behavior is random and unpredictable, and the behavior of our pendulum at high drive certainly has that appearance. However, if the behavior is truly unpredictable, then how was our program able to calculate it? This conundrum can be put another way when we realize that the behavior of our pendulum is described by the differential equation (3.18). From the theory of such equations we know that once the initial conditions (at $t = 0$, for example) are specified, the solution for θ is then *completely determined* for all future times. Indeed, we took this for granted in constructing our program. But how can the behavior be both deterministic and unpredictable at the same time?

We are thus faced with an apparent contradiction between analytic theory (the theory of differential equations) and numerical calculations (our program). Since our only evidence, so far, that the pendulum can be chaotic is from our numerical results, we might be tempted to suspect that we have made some sort of programming error, or that we have somehow misinterpreted the meaning of the numerical results. For example, we could imagine that if we waited long enough, a (predictable) pattern might emerge even at high drive, including at $F_D = 1.2$ in Figure 3.6. Indeed, how could such behavior ever be ruled out? Rather than pursue the question in this way, let us raise another possibility; namely, that the behavior is deterministic *and* unpredictable at the same time! This may seem to be impossible, but we will now show how to reconcile these two, apparently contradictory notions.

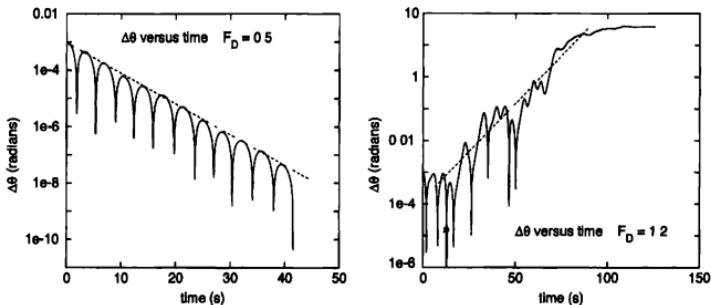


FIGURE 3.7: Results for $\Delta\theta$ from our comparison of two identical pendulums. The parameters were the same as in Figure 3.6. The initial values of θ for the two pendulums differed by 0.001 rad. On the left are results for low drive, while the results on the right were obtained from the chaotic regime. The dashed lines indicate the overall trends, that is, that $\Delta\theta$ decreases approximately exponentially for low drive, and increases roughly exponentially at high drive.

Let us consider the stability of the solutions to our pendulum equation of motion. We imagine that we have two *identical* pendulums, with exactly the same lengths and damping factors. We set them in motion at the same time, with the same driving forces. The only difference is that we start them with *slightly* different initial angles. We thus must calculate the angular positions of two pendulums, θ_1 and θ_2 , and we can do this using a program very similar to the one described above (an exercise we will leave to the reader). Some results for $\Delta\theta \equiv |\theta_1 - \theta_2|$ are shown in Figure 3.7 for two different values of the drive amplitude. The smaller value of F_D is the one for which we found simple oscillatory motion in Figure 3.6. To understand these results we first call your attention to the very sharp dips that occur approximately every 3 s. These dips in $\Delta\theta$ occur when one of the pendulums reaches a turning point. $\Delta\theta$ will vanish near each turning point since the trajectories $\theta_1(t)$ and $\theta_2(t)$ must then cross. It is more useful to focus on the plateau regions away from these dips in Figure 3.7. These plateau values of $\Delta\theta$ exhibit a steady and fairly rapid decrease with t . This means that the motion of the two pendulums becomes more and more similar, since the difference in the two angles approaches zero as the motion proceeds. (Indeed, this angular difference decreased by *six orders of magnitude* after about a dozen oscillations.) This in turn means that the motion is *predictable*. If, for some reason, we did not know the initial conditions of one of the pendulums, we could still predict its future motion since our results show that $\theta(t)$ converges to a particular solution (that of the other pendulum). This is what our intuition tells us to expect for predictable, nonchaotic motion.

In contrast, for the larger value of F_D we find that $\Delta\theta$ increases rapidly and irregularly with t ; this trend is indicated by the dashed line on the right in Figure 3.7. This is usually described by saying that the two trajectories, $\theta_1(t)$ and

$\theta_2(t)$, diverge from one another. Note that this divergence is extremely rapid at short times ($t < 75$ in Figure 3.7). $\Delta\theta$ saturates (i.e., stops changing) at long time periods, but this is only because it has reached a value of order 2π and simply can't get any larger! We have used a logarithmic scale in Figure 3.7, so the increase of $\Delta\theta$ with time is very rapid. The irregular variation of $\Delta\theta$ cannot be described by any simple function. However, if we were to repeat this calculation for a range of different initial values of θ_1 (keeping $\Delta\theta(0)$ fixed) and average the results, we would find a much smoother behavior, such as the dashed line in Figure 3.7. This line corresponds to the relation $\log(\Delta\theta) \sim \lambda t$, which implies

$$\Delta\theta \approx e^{\lambda t}. \quad (3.20)$$

It turns out that this functional form for $\Delta\theta$ is very common and the parameter λ is known as a Lyapunov exponent.⁹

For our pendulum the numerical results show that λ is positive at high drive, which means that two pendulums that start with nearly, but not exactly, the same initial conditions will follow trajectories that diverge exponentially fast.¹⁰ Since we can never hope to know the initial conditions or any of the other pendulum parameters exactly, this means that the behavior at $F_D = 1.2$ is for all practical purposes unpredictable. Our system is thus both deterministic and unpredictable. Put another way, a system can obey certain deterministic laws of physics, but still exhibit behavior that is unpredictable due to an extreme sensitivity to initial conditions. This is what it means to be chaotic. One more point should be noted from Figure 3.7. The behavior of $\Delta\theta$ can be described by a Lyapunov exponent in both the chaotic and nonchaotic regimes. In the former case $\lambda > 0$, while in the latter, $\lambda < 0$. The transition to chaos thus occurs when $\lambda = 0$.

Now that we have seen how $\theta(t)$ for our pendulum can be unpredictable, you might give up all hope of developing a useful theoretical description of the chaotic regime. However, it turns out that this view is too pessimistic. It is possible to make certain accurate predictions concerning θ , even in the chaotic regime! To demonstrate this we need to consider the trajectory in a different way. Instead of plotting θ as a function of t , let us plot the angular velocity ω as a function of θ . This is sometimes referred to as a phase-space plot. Since we have already constructed a program to calculate θ and ω , it is straightforward to modify it to make the desired plot; results for two values of the drive amplitude are shown in Figure 3.8.

⁹In general, systems such as a pendulum possess several different Lyapunov exponents. In order to be chaotic, at least one of these exponents must be positive. The accurate extraction of the values of the Lyapunov exponents from results such as those computed in this section is a somewhat complicated procedure. For one approach to this extraction see Wolf et al. (1985). Also note that logarithmic scales are used in the y -axes in Figure 3.7. This is because of the expectation of exponential behaviors. An exponential function would be displayed as a straight line when a logarithmic y -axis is used and thus it would be easier to spot an exponential dependence on such a plot. In addition, such a plot can clearly display y values that vary over many decades. This is an example of a good plotting strategy mentioned in Chapter 1.

¹⁰Similar results would be found if the two pendulums had slightly different lengths or driving forces, or if any other of the parameters were different.

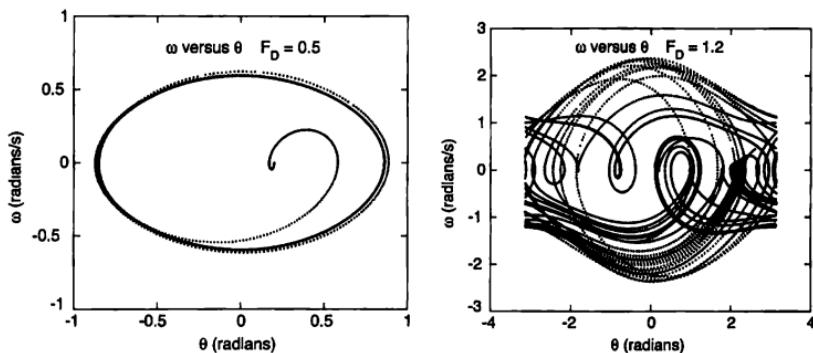


FIGURE 3.8: Results for ω as a function of θ for a pendulum. The parameters were the same as in Figure 3.6. For high drive (on the right), many trajectories go beyond $|\theta| = \pi$ and thus "jump" from $\theta = \pi$ to $\theta = -\pi$, or vice-versa in this plot.

With a small driving force the trajectory in phase space (ω - θ space) is easy to understand in terms of the behavior we found earlier for $\theta(t)$. For short times there is a transient that depends on the initial conditions (in this case we started at $\theta = 0.2$ with $\omega = 0$), but the pendulum quickly settles into a regular orbit in phase space corresponding to the oscillatory motion of both θ and ω . It can be shown that this final orbit is independent of the initial conditions; this is also what our results for the Lyapunov exponent imply. The behavior in the chaotic regime is a bit more surprising. The phase-space trajectory exhibits many orbits that are nearly closed and that persist for only one or two cycles. While this pattern is certainly not a simple one, it is not completely random, as might have been expected for a chaotic system. This is a common property of chaotic systems; they generally exhibit phase-space trajectories with significant structure.

If we examine these trajectories in a slightly different manner we find a very striking result. In Figure 3.9 we show the same type of phase-space graph, but here we plot ω versus θ only at times that are *in phase* with the driving force. That is, we only display the point when $\Omega_D t = 2n\pi$, where n is an integer.¹¹ This is an example of what is known as a *Poincaré section* and is a very useful way to plot and analyze the behavior of a dynamical system. The motivation for plotting the results in this way can be appreciated from an analogy with the function of a stroboscope. It sometimes happens that we want to examine an object that is rotating at a high rate. A good example is an old-fashioned (vinyl) record as it

¹¹When constructing this plot numerically you must be careful to account for the fact that time increases in steps of size Δt . Thus, the points in Figure 3.9 were actually plotted when $|t - 2n\pi/\Omega_D| < \Delta t/2$. Similarly, we could also plot at times corresponding to a fixed phase difference relative to the driving force for analogous results.

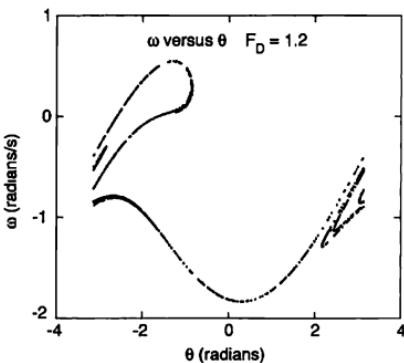


FIGURE 3.9: More results for ω as a function of θ for a pendulum, here we only plot points at times that are in phase with the driving force. The parameters were the same as in Figure 3.6. This surface of points is known as a strange attractor.

is rotated on a record player. When in operation, the record rotates too fast for a human eye to read the label. However, if the record is illuminated with a light source (a stroboscope), that is turned on and off at the frequency of the record player, our eye will receive input only when the record has a particular orientation; as a result we will be able to read the label as if the record were not moving at all. The key point is that things will look simpler when we observe them at a rate (i.e., frequency) that matches the problem. This lesson can be applied to the pendulum by observing the behavior, that is, recording the values of θ and ω , at a rate that matches a characteristic frequency of the system. For a driven pendulum, the obvious candidate for a qualifying frequency is that of the driving force, and this is effectively what we have done in the Poincaré section shown in Figure 3.9. If we had constructed this plot in the nonchaotic regime, with $F_D = 0.5$ for example, it would yield a single point (after allowing the initial transient to decay), since at any particular point of the drive cycle we would always find the same values of θ and ω .

The result of such a stroboscopic plot is very different in the chaotic regime, Figure 3.9. It turns out that except for the initial transient this phase-space trajectory is the same for a wide range of initial conditions. In other words, even though we cannot predict the behavior of $\theta(t)$, we do know that the system will possess values of ω and θ , which put it on this surface of points. The trajectory of our pendulum is drawn to this surface, which is known as an attractor. Actually, there are attractors in both the nonchaotic and chaotic regimes; the single point that would be found with $F_D = 0.5$ would also be an attractor. While the attractors have simple forms in the nonchaotic case, they have a very complicated structure

in the chaotic regime. The “fuzziness” of the chaotic attractor in Figure 3.9 is not due to numerical uncertainties or plotting errors. It is a property of the attractor. Chaotic attractors have a *fractal* structure and are usually referred to as *strange attractors*.¹² We will discuss the nature of fractals at some length in Chapter 7.

There is much more that the damped, nonlinear, driven pendulum can tell us about chaos, and we will explore a few of these lessons in the exercises. Our key results are: (1) it is possible for a system to be both deterministic and unpredictable—in fact, this is what we mean by the term chaos; and (2) the behavior in the chaotic regime is not completely random, but can be described by a strange attractor in phase space. We will amplify and expand on these themes in the following sections.

EXERCISES

- 3.9. Study the effects of damping by starting the pendulum with some initial angular displacement, say $\theta = 0.5$ radians, and study how the motion decays with time. Use $q = 0.1$ and estimate the time constant for the decay. Compare your result with approximate analytic estimates for the decay time. *Note:* Any of the exercises in the section can be conveniently done with either the Euler-Cromer algorithm or the Runge-Kutta method described in Appendix A.
- 3.10. Calculate $\theta(t)$ for $F_D = 0.1, 0.5$, and 0.99 , with the other parameters as in Figure 3.6. Compare the waveforms, with special attention to the deviations from a purely sinusoidal form at high drive.
- 3.11. For the three values of F_D shown in Figure 3.6, compute and plot the total energy of the system as a function of the time and discuss the results.
- 3.12. In constructing the Poincaré section in Figure 3.9 we plotted points only at times that were in phase with the drive force; that is, at times $t \approx 2\pi n/\Omega_D$, where n is an integer. At these values of t the driving force passed through zero [see (3.18)]. However, we could just as easily have chosen to make the plot at times corresponding to a maximum of the drive force, or at times $\pi/4$ out-of-phase with this force, etc. Construct the Poincaré sections for these cases and compare them with Figure 3.9.
- 3.13. Write a program to calculate and compare the behavior of two, nearly identical pendulums. Use it to calculate the divergence of two nearby trajectories in the chaotic regime, as in Figure 3.7, and make a qualitative estimate of the corresponding Lyapunov exponent from the slope of a plot of $\log(\Delta\theta)$ as a function of t .
- 3.14. Repeat the previous problem, but give the two pendulums slightly different damping factors. How does the value of the Lyapunov exponent compare with that found in Figure 3.7?
- 3.15. Study the shape of the chaotic attractor for different initial conditions. Keep the drive force fixed at $F_D = 1.2$ and calculate the attractors found for several different initial values of θ . Show that you obtain the same attractor even for different initial conditions, provided that these conditions are not changed by too much. Repeat your calculations for different values of the time step to be sure that it is sufficiently small that it does not cause any structure in the attractor.
- *3.16. Investigate how a strange attractor is altered by small changes in one of the pendulum parameters. Begin by calculating the strange attractor in Figure 3.9.

¹²Physicists like to use provocative names.

Then change either the drive amplitude or drive frequency by a small amount and observe the changes in the attractor.

- *3.17. Construct a very high-resolution plot of the chaotic attractor in Figure 3.9, concentrating on the region $\theta > 2$ rad. You should find that there is more structure in the attractor than is obvious on the scale plotted in Figure 3.9. In fact, an important feature of chaotic attractors is that the closer you look, the more structure you find. We will see later that this property is related to fractals. It turns out that a strange attractor is a fractal object. *Hint:* In order to get accurate results for a high resolution plot of the attractor, it is advisable, in terms of the necessary computer time, to use the Runge-Kutta method.

3.4 ROUTES TO CHAOS: PERIOD DOUBLING

We have seen that at low driving forces the damped, nonlinear pendulum exhibits simple oscillatory motion, while at high drive it can be chaotic. This raises an obvious question: Exactly how does the transition from simple to chaotic behavior take place? It turns out that the pendulum exhibits transitions to chaotic behavior at several different values of the driving force. We have already observed in Figure 3.6 that one of these transitions must take place between $F_D = 0.5$ and 1.2. However, this transition is not the clearest one to study numerically, so we will instead consider the behavior at somewhat higher driving forces.¹³

Figure 3.10 shows results for θ as a function of time for several values of the driving force calculated using the Euler-Cromer program described earlier. At these high values of the drive the pendulum often swings all the way around its support; this can be seen from the vertical steps in θ as our program resets¹⁴ this angle to keep it in the range $-\pi$ to π . These steps notwithstanding, the behavior in Figure 3.10 is a periodic, repeating function of t in all three cases (after the initial transients have damped away). The drive frequency used here was $\Omega_D = 2/3$ so the period of the driving force was $2\pi/\Omega_D = 3\pi$, and this is precisely the period of the motion found at $F_D = 1.35$. Hence, in this case the pendulum moves with the same frequency as the driving force.

The behavior at $F_D = 1.44$ is a bit more subtle. While we again have periodic motion, the period is now *twice* the drive period. This can be seen most clearly by comparing the θ - t waveforms at $F_D = 1.35$ and 1.44, and noticing that in the latter case the bumps alternate in amplitude. Our pendulum has already surprised us on several occasions, and we might be tempted to add this behavior to our list of pendulum puzzles. However, this surprise is a very special and important one. When a nonlinear system is excited or driven by a single frequency stimulus, the response is, in general, not limited to the driving frequency. If Ω_D is the drive frequency, the nonlinear response will usually contain components at $2\Omega_D$, $3\Omega_D$, etc., at all harmonics. This process is known as mixing and is manifest by the generation of responses at integer multiples of the driving frequency. Such a

¹³The pendulum exhibits an extraordinarily rich behavior, some of which we discuss later. Unfortunately, we only have time to explore a small portion of the chaotic regime here. We will, therefore, limit ourselves to one value of both the drive frequency and damping, and only certain ranges of the drive amplitude. The interested reader is encouraged to investigate other parameter values. Additional results are also described by Baker and Gollub (1990).

¹⁴This distraction can be avoided by plotting ω instead of θ . We will leave this to the exercises.

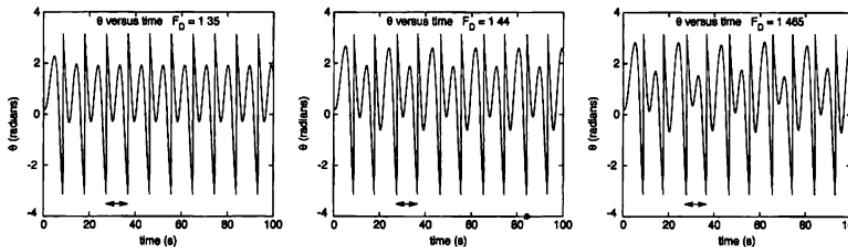


FIGURE 3.10: Results for θ as a function of time for our pendulum for several different values of the drive amplitude. The other parameters were the same as in Figure 3.6 except that here we used a time step of 0.01. The horizontal arrows show the period of the driving force. In the middle plot the period is twice the drive period, since the values at the maxima alternate between $\theta \approx 1.9$ and 2.6. On the right the period is four times the drive period, as the maxima alternate between the values $\theta \approx 1.69$, 2.82, 1.52, and 2.72.

nonlinear response is standard and well understood, and its key property is that it contains frequencies that are equal to or greater than the drive frequency. Hence, the periods of these harmonics will be smaller than the drive period. In contrast, our pendulum is now exhibiting a response at $\Omega_D/2$ (a lower frequency!), a *subharmonic*, which is unlike any standard mixing effect.

Returning to Figure 3.10, a careful look at the results for $F_D = 1.465$ shows that they exhibit a period that is four times the driving period. The pattern should now be evident. If we were to increase the drive amplitude further, the period would double again as the pendulum would switch to a motion that has a period eight times that of the drive. This period-doubling cascade would continue if the drive were increased further.

But if the period keeps on doubling, what about the transition to chaos? A nice way to appreciate how this transition comes about is with what is known as a *bifurcation diagram*. In Figure 3.11 we show a bifurcation diagram for θ as a function of drive amplitude, which was constructed in the following manner. For each value of F_D we have calculated θ as a function of time. After waiting for 300 driving periods so that the initial transients have decayed away, we plotted θ at times that were in phase with the driving force¹⁵ as a function of F_D . Here we have plotted points up to the 400th drive period. This process was then repeated for the range of values of F_D shown in the figure.¹⁶

To understand the bifurcation diagram we start at $F_D = 1.35$. We have already seen that in this case the motion has the same period as the drive, so if we observe θ at a particular time in the drive cycle we will always find the same

¹⁵Just as we did in constructing the Poincaré section in Figure 3.9.

¹⁶We have chosen to study this range of F_D because it exhibits period-doubling in an especially clear manner. Results for other values of F_D are shown by Baker and Gollub (1990), or you can calculate them for yourself.

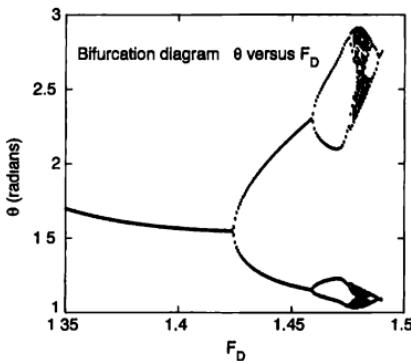


FIGURE 3.11: Bifurcation diagram for our pendulum. The parameters used for the calculation were the same as in Figure 3.10.

value. Our bifurcation diagram thus consists of a single point; there is just one value of θ for this value of F_D , although that point will be plotted many times. We will refer to this as period-1 behavior, since the θ - t waveform has the same period as the driving force. If the motion is period-2, then the values of θ that are plotted will alternate between two values. This is just the alternation we saw at $F_D = 1.44$ in Figure 3.10, and it leads to two points on the bifurcation diagram. The pattern should now be clear: Motion that is period n will yield n points on the bifurcation diagram for that value of F_D . From Figure 3.11 we see that the behavior is period 1 up to approximately $F_D = 1.424$, where there is a transition to period-2 motion. This persists up to the transition to period-4 behavior at $F_D \approx 1.459$. This process continues, although the resolution of our diagram makes it difficult to follow the behavior past period 8. This period-doubling cascade ultimately ends in a transition to chaotic behavior.

We have now obtained at least a qualitative understanding of *how*¹⁷ the transition from regular to chaotic behavior occurs for one particular system, the pendulum, in one particular range of parameters (i.e., drive force, drive frequency, damping strength, etc.). But how general is this behavior? Is it only found in the pendulum, or does it occur in other systems? These are the kinds of questions that physicists often ask. We tend to look for (and like to discover) patterns and principles that occur widely and apply to many different systems. For example, the motion of macroscopic objects can, with only a few exceptions, be described

¹⁷In this section we have concentrated on the *how* of the period-doubling-route to chaos. In the next section, we will discuss more of the *why*. We will do this using somewhat simpler models, that are often called *iterative maps*. Though perhaps further removed from real physical systems, they offer unique opportunity to investigate the mathematics behind period doubling in great detail.

by Newton's laws of motion. Likewise, classical (as opposed to quantum) electromagnetic phenomena can be described by Maxwell's equations. Because of their near-universal applicability we feel that these "laws" of physics provide us with a better understanding of the world. We don't want to get too philosophical here, and we also don't want to be pressed into a discussion of what the term "understand" really means, but we hope that this gives the reader some appreciation for why it is important to look hard for universal aspects in any problem or result. With that in mind, we now restate the question posed at the beginning of this paragraph. Is the period-doubling route to chaos universal in any way, or do all systems have their own particular way of making this transition?

While there is not yet a complete theory of chaos in the sense of Newton's laws or Maxwell's equations, the answer to this question seems to be the following. Many systems have been found to exhibit chaotic behavior, but there appear to be only a few ways in which the transition from simple to chaotic behavior can occur. The periodic-doubling scenario that we have observed with the pendulum is one of these few known routes to chaos. You can then ask if this periodic-doubling procedure itself has any properties that are universal. The answer to this is yes (why else would we raise the question?!). In the next section we will study a mathematically "stripped down" model called the logistic map, which also exhibits the period doubling route to chaos. The mathematical simplicity of this model will enable us to see why this route to chaos is so common and why many features of the transition are "universal." However, before we discuss the logistic map, it is useful to consider an important property of the period doubling transitions in our pendulum.

Returning to our bifurcation diagram, Figure 3.11, we note that the spacing between period-doubling transitions becomes rapidly smaller as the order of the transition increases. For example, the period-2 regime extends from $F_D \approx 1.424$ to 1.459, while the period-4 regime extends only from about 1.459 to 1.476, and the same trend is found for higher periods.¹⁸ Let us define F_n to be the value of the driving force at which the transition to period- 2^n behavior takes place. The shrinkage of the size of the periodic windows can be described by a parameter δ_n , where

$$\delta_n \equiv \frac{F_n - F_{n-1}}{F_{n+1} - F_n}. \quad (3.21)$$

The observation that the windows become smaller as n increases means that $\delta_n > 1$. It has been found that as n becomes large, δ_n approaches a constant that is known as the Feigenbaum δ . That is, the rate of shrinkage approaches a constant in the limit $n \rightarrow \infty$. Moreover, it turns out that essentially all systems that exhibit the period-doubling route to chaos appear to possess the same value of $\delta \approx 4.669\dots$. This is one of the universal parameters and features associated with the transition to chaos. There are, as we have hinted above, several other known routes to chaos, and a few of them can also be found in the pendulum. However, rather than making this chapter the story of the pendulum, we will next consider several other chaotic systems.

¹⁸In order to see this clearly we would need to examine the bifurcation on a much finer scale than in Figure 3.11.

EXERCISES

- 3.18. Calculate Poincaré sections for the pendulum as it undergoes the period-doubling route to chaos. Plot ω versus θ , with one point plotted for each drive cycle, as in Figure 3.9. Do this for $F_D = 1.4$, 1.44 , and 1.465 , using the other parameters as given in connection with Figure 3.10. You should find that after removing the points corresponding to the initial transient the attractor in the period-1 regime will contain only a single point. Likewise, if the behavior is period n , the attractor will contain n discrete points.
- *3.19. Calculate Poincaré sections for the pendulum but for a frequency which has *nothing* to do with those intrinsic to the system. For example, you might calculate the Poincaré sections using an angular frequency Ω' that is *irrationally* related to the driving frequency Ω_D (as well as to the natural frequency Ω) for your stroboscopic snapshots. Do the results tell you anything about the system? Hint: you will find that the results *look* chaotic even in the periodic regimes. The point is that it is crucial to interrogate the system at a rate that matches one of the intrinsic frequencies of the problem.
- 3.20. Calculate the bifurcation diagram for the pendulum in the vicinity of $F_D = 1.35$ to 1.5 . Make a magnified plot of the diagram (as compared to Figure 3.11) and obtain an estimate of the Feigenbaum δ parameter.
- *3.21. Investigate the bifurcation diagrams found for the pendulum with other values of the drive frequency and damping parameter. Warning: This can easily become an ambitious project!

3.5 THE LOGISTIC MAP: WHY THE PERIOD DOUBLES

In the past few sections we have spent a lot of time studying the chaotic behavior of a pendulum. We now want to look at this problem from a broader perspective and ask the following question: "What does it really take for a system to be chaotic?" That is, what properties must a system have to be chaotic, and can we learn how to predict ahead of time if a particular system is capable of exhibiting chaos? This is a difficult, and extremely important question. In the case of the pendulum, it turns out that chaos is only found when the pendulum is damped, is nonlinear, and is subject to a driving force. All three of these must be present. However, while this fact is useful, it is also important to consider what other simple equations of motion are capable of chaotic behavior.

With these questions in mind, we now consider a rather different type of "system" that is called the *logistic map*. This system can be interpreted as a model for population growth in a collection of animals.¹⁹ The model is defined by the relation

$$x_{n+1} = \mu x_n(1 - x_n). \quad (3.22)$$

One can think of x_n as being the size of the population in generation n , while μ is a parameter related to the amount of food that is available (hence, the value of μ will control the rate of population growth or loss). This "equation of motion" is a *discrete map*, and thus looks rather different from the continuous equations

¹⁹It also has many other interpretations, as discussed in the article by May (1976). The book by Baker and Gollub (1990) contains a nice introduction to the chaotic behavior of the logistic map.

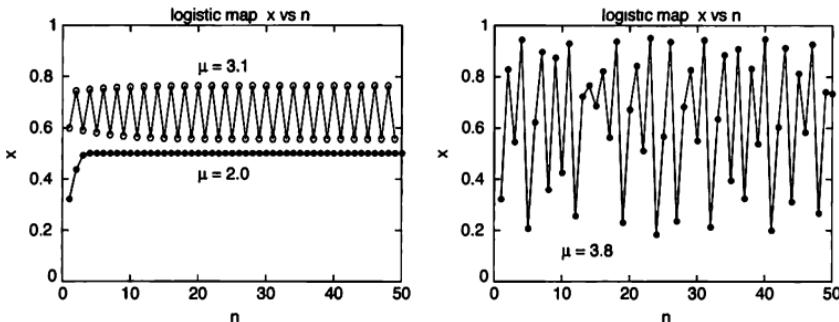


FIGURE 3.12: Behavior of x as a function of n for the logistic map, for several different values of μ . For $\mu = 3.1$ the system is in the period-2 regime, while for $\mu = 3.8$ the behavior is chaotic. Note that the values of x_n are given by the symbols, the connecting lines are simply guides to the eye.

of motion what we are used to seeing in physics. However, you should recognize that in our numerical approaches, such as the Euler method (3.5), we have actually converted a continuous equation of motion (e.g., Newton's law) into a discrete map, so (3.22) is really not that new for us! One advantage of studying the logistic map is that the simplicity of (3.22) will allow us to gain some deep insights into the problem of chaos. However, there is also a potential danger in this general approach. The danger is that the conversion to a discrete map may alter some of the fundamental physics of the problem. For example, it is conceivable that a discrete map may be chaotic, even with the original continuous system is not. This is not the case for any of the systems we study in this book, and we will not pursue this issue here (you can find more on this problem in advanced treatments of chaotic systems). However, the general message (yet again) is that our numerical “solutions” are only approximations to the true behavior.

Let us now consider the general behavior of the logistic map. Figure 3.12 shows the behavior of x_n for several different values of μ . Here we have simply started with a random value of x_0 , and used (3.22) to calculate x_1 , and then x_2 , etc.²⁰ We see from Figure 3.12 that the logistic map exhibits nonchaotic behavior for small μ , a regime of period-2 behavior for intermediate μ , and chaotic behavior for large μ . We will leave it to the exercises to explore the period doubling route to chaos in this system, and to calculate the associated bifurcation diagram (which bears a striking resemblance to the bifurcation diagram for the pendulum in Figure 3.11). Here we want to explore how the very simple function in (3.22) is able to give such interesting and complex results.

²⁰Note that the values of x are restricted to the range between 0 and 1, while μ must lie between 0 and 4.

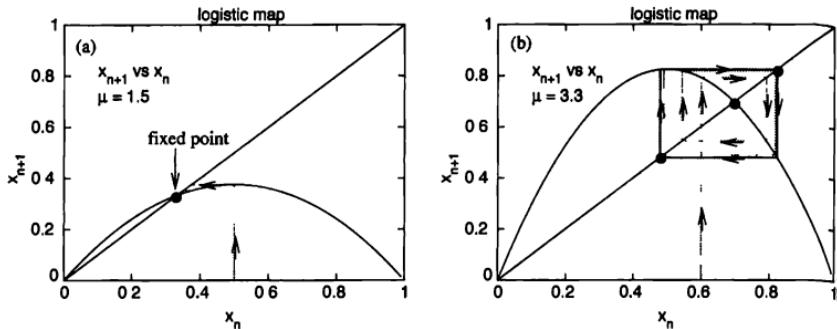


FIGURE 3.13: (a) The logistic map function $f(x)$ for $\mu = 1.5$. The fixed point of the map, x^* , is the point where this function intersects the diagonal line. The dotted lines show the “trajectory” of the system, starting from an initial value $x_0 = 0.5$; the arrows indicate the “direction” that the system moves. Note how the system rapidly approaches the fixed point. (b) For $\mu = 3.3$ the system is in the period-2 regime. There is a fixed point at $x \approx 0.70$, but it is unstable (there is also a trivial fixed point at $x = 0$ for all values of μ). Unless x has this special fixed point value, the system will oscillate between two values of x , as shown here. The dotted lines show the corresponding system trajectory

Let us first consider the behavior of the logistic map for small values of μ , as found for $\mu = 2.0$ in Figure 3.12. Here the system rapidly approaches a particular value of x that is called a *fixed point*. That is, after many iterations the map “repeats” the same value of x , over and over. If we define the logistic function²¹

$$f(x) = \mu x(1 - x), \quad (3.23)$$

then the fixed points x^* of the map function satisfy the relation

$$x^* = f(x^*). \quad (3.24)$$

You can easily combine (3.23) and (3.24) to find the exact fixed point value(s) x^* as a function of μ . However, it is very instructive to consider the fixed point behavior graphically as shown in Figure 3.13. Here we plot the map function $f(x)$ with $\mu = 1.5$, along with a line with a slope of unity; that is, we have plotted both sides of (3.24). The fixed point solutions are then the values of x at which these two curves intersect. If the value of x somehow gets to a fixed point value, i.e., $x_n = x^*$, then x_{n+1} will repeat the value, and then x_{n+2} will repeat ...

Also drawn as the dotted line in Figure 3.13 is the trajectory of the system starting from an initial value $x_0 = 0.5$. The arrows indicate the path that is taken.

²¹The logistic function $f(x)$ also depends on μ , so it would be more precise to call it $f(x, \mu)$. However, this leads to a somewhat cumbersome notation in what follows, so we will not indicate this dependence explicitly.

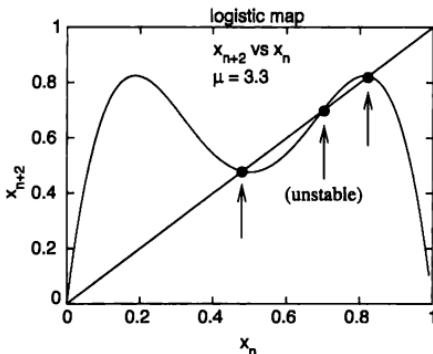


FIGURE 3.14: Second iterate of the logistic map, $f_2(x)$, for $\mu = 3.3$, showing the fixed points. The fixed point near $x \approx 0.70$ is the unstable fixed point associated with the first iterate, $f(x)$.

Starting from the point $[(x_0, 0)]$ on the horizontal axis, the system first moves to the point $[x_0, f(x_0)]$. It then continues on, passing through the points $[x_1, f(x_1)]$, $[x_2, f(x_2)]$, etc., eventually approaching the fixed point at $[x^*, f(x^*)]$.

We can use the same approach to understand the period-2 behavior found for $\mu = 3.3$ in Figure 3.13. The corresponding plot of $f(x)$ and the system trajectory is shown in Figure 3.13(b). There are now three fixed points of the map, and but the system only passes through two of them as it travels along its trajectory. The fixed point at $x^* \approx 0.70$ is unstable. The system avoids this point, and instead cycles between two other values, with one near $x = 0.48$ and the other near $x = 0.82$. These correspond to the period-2 oscillations seen in Figure 3.12.

Deeper insight into the origin of this period-2 behavior can be obtained by considering the so-called “second iterate” of the logistic map. That is, we consider how x_{n+2} depends on x_n . In terms of the logistic function, the second iterate function is

$$f_2(x) = f(f(x)) , \quad (3.25)$$

where $f(x)$ is defined in (3.23). When the behavior is precisely period-2, this corresponds to a fixed point of $f_2(x)$, and we can solve for these fixed points just as we found the fixed points of the first iterate, $f(x)$. Figure 3.14 shows a plot of $f_2(x)$ along with the line $x_{n+2} = x_n$. We now find three fixed points, just as in Figure 3.13. One of these (the fixed point in the middle) is the original unstable fixed point of the first iterate $f(x)$. The other two fixed points are stable ones, and describe the period-2 behavior.

This method can be used to study the period-4 regime, by considering the fourth iterate of $f(x)$, etc., for period-8, period-16, and so on. In addition to providing a nice way to analyze these period doubling transitions, this approach

gives insight into the underlying nature of these transitions. Let us examine again the behavior of the first iterate in Figure 3.13, and consider the behavior of $f(x)$ in the vicinity of a fixed point. Near a fixed point we can write

$$f(x^* + \delta x) \approx f(x^*) + f'(x^*)\delta x, \quad (3.26)$$

where $f'(x)$ is the derivative of $f(x)$. If the system starts a short distance from the fixed point, i.e., we start at a value of x for which $\delta x = x - x^*$ is small, we can apply (3.26). We can see that if the derivative $|f'(x^*)| < 1$, then the system will move towards x^* as the map is iterated, while if $|f'(x^*)| > 1$, the system will move away from the fixed point. Hence, the behavior is controlled by the magnitude of this derivative. As μ is varied, the value of $f'(x^*)$ will also vary, so we can now see how, in a mathematical sense, the fixed point can be made unstable. At the same time, this leads to two additional fixed points for $f_2(x)$, and this is why the system is driven into a period-2 regime. Thus, we are now able to see *why* the system undergoes a period-doubling transition. Indeed, this general line of attack can be extended to the higher period transitions, and is described in the paper by Feigenbaum (1978). Moreover, we can now see why these period doubling transitions can be *universal*; i.e., why many details of the behavior can be the same in different systems. The idea here is that the logistic function, $f(x)$, is representative of the map function for a very general system. Near the transitions, such functions will generally have a smooth (i.e., Taylor expansion) form that is the same as that of $f(x)$. Hence, the nature of the fixed points, their stability, the second iterate functions, etc., will all be basically the *same* for many different systems. This is why the chaotic properties of very different systems, including the logistic map and the nonlinear pendulum, can be *universal*.

EXERCISES

- 3.22.** The logistic map undergoes successive period doubling as μ is increased from below 3 and finally develops a fully chaotic behavior at $\mu \approx 3.56$. It is interesting to compute the bifurcation diagram for the logistic map (the analog of Figure 3.11 for the pendulum). Use numerical results for the logistic map to estimate the value of the Feigenbaum δ parameter and compare it with that quoted above for the pendulum. The article by May (1976) and the book by Baker and Gollub (1990) contain good introductions to the logistic map.
- 3.23.** Calculate the trajectories of the logistic map, as shown in Figure 3.13, for different values of μ . It is especially interesting to do this in the period-4, period-8, ... regimes, and also when the system is chaotic.
- *3.24.** Iterative maps are often used to produce what are called “pseudo-random” numbers. In this exercise, you will look at the behavior of such maps and investigate what makes them suitable for this purpose
- (a) First, consider the following,

$$x_{i+1} = [\alpha x_i 2^n + \beta] (\text{mod } 2^n) / 2^n, \quad (3.27)$$

where $\text{mod}(2^n)$ indicates the remainder after division by 2^n . Here $0 \leq x_i < 1$ and n , α , and β are suitable integers. Fix n (say, to 8) and investigate the properties of the maps for various values of α when the

- $\beta = 0$ and again when $\beta = 1$. Also consider the effect of different initial values x_0 .
- (b) Next, consider a variation of the above where discrete steps are introduced by using the $\text{int}(x)$ function (which takes the greatest integer not exceeding x):

$$x_{i+1} = [\text{int}(x_i 2^n) \alpha + \beta] (\text{mod } 2^n) / 2^n. \quad (3.28)$$

Do the same as in part (a). Try values of α which are even, and also odd numbers of the forms $8m \pm 1$ and $8m \pm 3$. You might be interested to know that this is very similar to the standard random number generator function found in many software packages; and is called the multiplicative congruential method. In most practical generators one takes advantage of the integer arithmetic overflows, but here you should avoid them. The quality of the random numbers depends on the choice of α , β , and n . While there are some theoretical results for how to choose these parameters to produce "good quality" random numbers, much is only known empirically. See Appendix F and also Knuth (1998) and Hamming (1987) for more on this.

3.6 THE LORENZ MODEL

The two models that we have considered so far in this chapter — the pendulum and the logistic map — are both very simple systems, yet they exhibit extremely rich behavior. It is thus not surprising that other slightly more complicated systems are also capable of chaotic behavior. When we think of chaotic or unpredictable behavior, an example that naturally comes to mind is the weather. Because of the economic importance of having accurate weather predictions, a good deal of effort has been devoted to this problem. While much of this effort has gone into computer modeling of Earth's atmosphere, much has also been devoted to understanding the weather problem from a more fundamental point of view. It was work of this kind by the atmospheric scientist E. N. Lorenz (1963) that provided a major contribution to the modern field of chaos.

Lorenz was studying the basic equations of fluid mechanics, which are known as the Navier-Stokes equations; they can be thought of as Newton's laws written in a form appropriate for a fluid. These are a complicated set of differential equations that describe the velocity, temperature, density, etc., as functions of position and time, and they are very difficult to solve analytically in cases of practical interest. Of course, this is just the type of problem where a computational approach can be useful, and that is precisely what Lorenz did. The specific situation he considered was the Rayleigh-Bénard problem, which concerns a fluid in a container whose top and bottom surfaces are held at different temperatures. It had long been known that as the difference between these two temperatures is increased, the fluid can undergo transitions from a stationary state (no fluid motion) to steady flow (nonzero flow velocities that are constant in time, also referred to as convection) to chaotic flow. Lorenz did his work more than 40 years ago, so the computational power available to him was not very impressive by today's standards. This prompted him to consider a greatly simplified version of the Navier-Stokes equations as applied to this particular problem. Indeed, he grossly oversimplified the problem as he

reduced it to only three equations

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= -xz + rx - y, \\ \frac{dz}{dt} &= xy - bz.\end{aligned}\tag{3.29}$$

These are now known as the Lorenz equations (or equivalently, the Lorenz model).²² As noted above, the Navier-Stokes equations, from which those in (3.29) are derived, involve the state of the fluid as a function of position and time. Hence, a complete description of the Rayleigh-Bénard problem must involve a *large* number of variables. The Lorenz variables x , y , and z are derived from the temperature, density, and velocity variables in the original Navier-Stokes equations, and the parameters σ , r , and b are measures of the temperature difference across the fluid and other fluid parameters. However, it is not particularly useful to insist on interpreting x , y , and z in that manner, since the simplifications made in reducing the problem to only three variables means that we cannot expect our results to apply to any real system. Rather, the behavior exhibited by these equations is indicative of the *type* of behavior that could be expected of the Rayleigh-Bénard problem or any other problem involving the Navier-Stokes equations. For brevity we shall often refer to the latter as the weather problem. Any behavior we find in the Lorenz model will certainly be found in the weather problem. Moreover, it makes strategic sense to attack the Lorenz model first, since if we can't solve that, we would have no hope of making headway on the weather problem.

With this in mind, we plunge ahead and consider solutions to the Lorenz equations. They are just three, coupled, differential-equations that are very similar to those we have already encountered in connection with the pendulum [for example, (3.4)]. We will, therefore, not discuss the program construction in detail; it is basically the same approach we employed in the previous sections, but with the pendulum equation of motion replaced by the Lorenz equations. However, we will make a few general comments about the numerical aspects of the problem. We saw in our work on the pendulum that the Euler method does not conserve energy for oscillatory problems. The Lorenz equations exhibit oscillatory solutions for certain parameter values, which could cause us to worry about using the Euler method. We might, therefore, want to use the Euler-Cromer method here, but since that algorithm is designed for second-order differential equations, it is not directly applicable to the Lorenz model. However, it turns out that the Euler algorithm can actually be used to treat the Lorenz problem, for the following reason. Several of the terms in the Lorenz equations play the same role as the damping term in the pendulum equations of motion, while other terms are analogous to the driving

²²While these do not appear to resemble our pendulum's equation of motion, the two systems are both nonlinear (note the terms involving products such as xz and xy), and as we saw with the pendulum, this is very important.

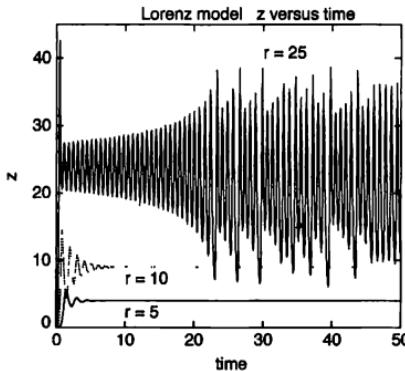


FIGURE 3.15: Variation of the Lorenz variable z as a function of time. The calculation was performed using the Euler method with a time step of 0.0001. The initial conditions were $x = 1$, $y = z = 0$

force. If the time step in the Euler algorithm is sufficiently small, the energy lost (or added) through the error terms associated with the Euler method can be made much smaller than the energy lost to the effective damping, or added by the effective driving force. In this situation the Euler method provides an accurate solution, as can be verified directly by simply repeating the Euler calculation with different time steps, or by comparing with the solution generated with the Runge-Kutta method. This exercise, which we will leave to the reader, reveals that the Euler solution is accurate for the time steps we have employed in the computations described below. This is another example of a point we made earlier; the suitability of an algorithm depends upon the problem.

Returning to the Lorenz problem, there are three parameters in (3.29), σ , b , and r , and the behavior one finds depends on their values. We will follow custom (and also Lorenz [1963]) and use $\sigma = 10$ and $b = 8/3$. According to some authors these values correspond to cold water, but given the highly simplified nature of the model you shouldn't take this claim seriously! The parameter r is a measure of the temperature difference between the top and bottom surfaces of the fluid. For small r the effective force on the fluid is small in the sense that there is very little heat carried by the fluid. As r increases this force increases, so r plays a role analogous to the drive amplitude, F_D , in the pendulum problem. Results for z as a function of time are shown in Figure 3.15, which shows the behavior at three different values of the force r (x and y exhibit qualitatively similar behavior). At $r = 5$ there is an initial transient, and after it decays away z is a constant, independent of t . The same behavior is seen at $r = 10$, although the transient takes a little longer to decay. These two cases correspond to steady convective motion in the original fluid; in this process the warm fluid produced at the bottom surface of the container rises

and the cooler fluid returns from the top. This steady convection is the analog of regular nonchaotic motion of the pendulum. The behavior is completely different at $r = 25$. Here the initial transient is roughly periodic, but it gives way to an irregular, that is, chaotic time dependence after $t = 20$ or so. There are not many exact results available for the Lorenz model, but it is known that the transition from steady convection to chaotic behavior takes place at $r = 470/19 \approx 24.74$. This is consistent with our numerical results, although we have not attempted to locate this transition accurately here; we leave that for the exercises!

The chaotic behavior seen at $r = 25$ after the initial transient has decayed certainly appears to be very random and unpredictable. However, we learned in dealing with the pendulum that looks can be deceiving. In that case we found that a Poincaré section can reveal underlying regularities that are not obvious from the time dependence alone. With this motivation we consider how to construct such a phase-space plot for the Lorenz model. The situation here is a little more complicated than that of the pendulum, since we now have three variables to deal with, x , y , and z , as opposed to only two in the pendulum problem (θ and ω). There are several ways to proceed. Perhaps the simplest way is to imagine that x , y , and z are coordinates in some abstract space and recognize that we are dealing with a trajectory in this space. We can then obtain a projection of this trajectory²³ by simply plotting z as a function of x (for example); this gives a projection onto the x - z plane. Such a projection for the chaotic case $r = 25$ is shown in Figure 3.16. In this trajectory the system undergoes approximately periodic oscillations (roughly circular orbits) on one side of the line $x = 0$, then moves to the opposite side of this line and undergoes a new series of oscillations, etc.

The phase-space plot in Figure 3.16 certainly gives some hints of an underlying regularity, but we would like to have more than hints. With the pendulum we saw that the Poincaré sections in the chaotic regime reveal the attractors in a particularly clear way, as with the strange attractor in Figure 3.9. Might there be a similar sort of underlying attractor for the Lorenz model? You will recall that to construct a Poincaré section for the pendulum we plotted ω as a function of θ at times that were in phase with the driving force. Hence, we essentially used the driving force as a timekeeper, which told us when to make the measurement. However, in the Lorenz model there is no direct analog to this drive force with its sinusoidal time dependence, so we must take a slightly different approach. We already mentioned that the Lorenz variables x , y , and z can be viewed as specifying the trajectory of a particle moving in three dimensions. It is then natural (or at least it was to Lorenz!) to consider two-dimensional slices through this trajectory. To be more specific, we show on the left in Figure 3.17 a plot of z versus y when

²³It is difficult to present a full three-dimensional trajectory in this book, but nowadays it is not very hard to find ways to do so on a video monitor. In addition to using one of the many 3D visualization applications, some computer languages even have specialized extensions to accommodate 3D plotting. Many of these will even allow you to rotate and zoom around the plot at will. A case in point is a system called *VPython* (the Visual module of a language called *Python*). It is highly recommended that you try to plot and view the Lorenz model trajectory in full three dimensions using one of these methods, particularly as the time evolves and the butterfly-like trajectory gradually becomes filled in.

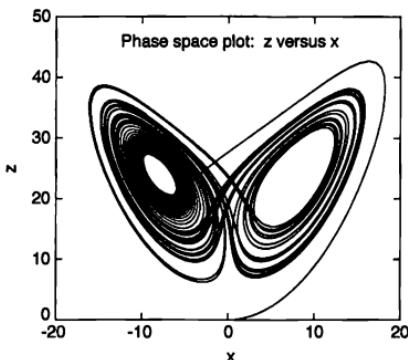


FIGURE 3.16: Trajectory of the Lorenz model projected onto the x - z plane, with $r = 25$. This was calculated using the Euler method with a time step of 0.0001 and the initial conditions $x = 1$, $y = z = 0$.

$x = 0$.²⁴ In our trajectory language, we are simply plotting the places where the trajectory intersects the y - z plane. The results in Figure 3.17 were obtained with $r = 25$, which, as we have already seen, places us in the chaotic regime. We see from the figure that even though the behavior is strongly chaotic, there is a very high degree of regularity in the phase-space trajectory. This attractor surface in phase space can be shown to be independent of the initial conditions. Hence, while the time-dependent behavior [e.g., $z(t)$] is unpredictable, we *can* predict with certainty that the system will be found somewhere on the attractor surface in phase space.

In the previous section we studied the period-doubling route to chaos in the pendulum. The Lorenz model exhibits this route to chaos, and several others as well. A complete study of the model would take us too long here, so we refer the interested reader to Sparrow (1982), an entire *book* devoted to the Lorenz model. Here we will only explore one of the chaotic transitions exhibited by the model. Figure 3.18 shows the behavior of z at two values of r . At $r = 160$ we see periodic oscillations. While the waveform of these oscillations is certainly not simple, they are stable and persist forever. This corresponds to period-1 behavior, and is analogous to the results we found for the pendulum. If we were to examine the behavior as r is made smaller, we would observe period-doubling and eventually chaos (see the exercises for more on this). However, let us instead consider what happens as r is increased; this is also shown in Figure 3.18. The motion is seen to be approximately periodic for many cycles of oscillation, but these periodic stretches

²⁴Since t can take on only discrete values in our simulations, we can only determine that the system has crossed the plane $x = 0$ at some instant between two adjacent time steps. We then interpolate the position to construct the phase-space plots.

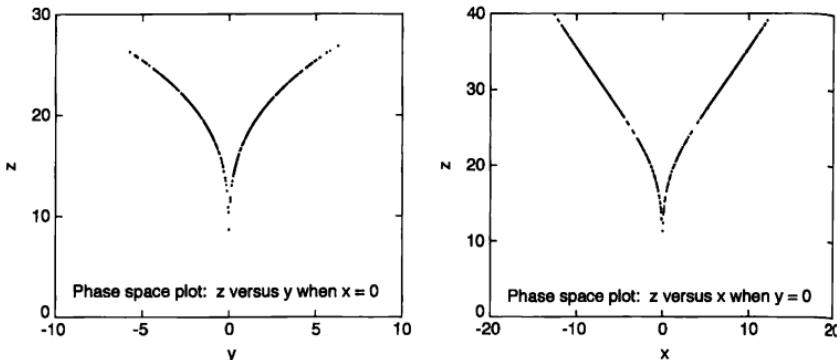


FIGURE 3.17: Phase-space plots for the Lorenz model with $r = 25$, calculated with a time step of 0.0001. Left: z versus y , with points plotted only when $x = 0$. Right: z versus x , with points plotted only when $y = 0$. The points were recorded only after $t = 30$, to allow for the decay of initial transients.

are interrupted by chaotic interludes (hiccup?). The behavior is thus chaotic, but in a sense it is only barely chaotic.

This is known as the intermittency route to chaos, and can be pictured as follows. Let r_c be the value of r at which this transition occurs. For r less than r_c the behavior is perfectly periodic, as found with $r = 160$. For values of r that are just a little beyond the transition, in our case just a little larger than r_c , the behavior is almost periodic with only an occasional chaotic interlude. As we go farther and farther into the chaotic regime these interludes occur more and more often, until eventually the underlying periodic behavior is unrecognizable. As we approach r_c , but still stay inside the chaotic regime, there are fewer and fewer interludes spaced farther and farther apart. At the transition the spacing between the interludes becomes infinitely long; that is, they no longer occur, and we are left with periodic motion.

The Lorenz model is a very rich and interesting chaotic system, and we will explore it in further detail in the exercises. Before we leave this topic we have a few comments concerning what the behavior of the Lorenz model implies about the weather problem. We have seen that the Lorenz model can exhibit chaotic behavior. Since it is a special case of the Navier-Stokes equations, we must expect that chaos is a general property of virtually all fluid systems. The weather problem is concerned with a complicated fluid system, our atmosphere, so we reach the not very surprising conclusion that the weather is a chaotic problem. From what we have learned about the extreme sensitivity of chaotic systems to initial conditions, we know that predicting the weather must therefore be an *inherently* difficult problem. Suppose

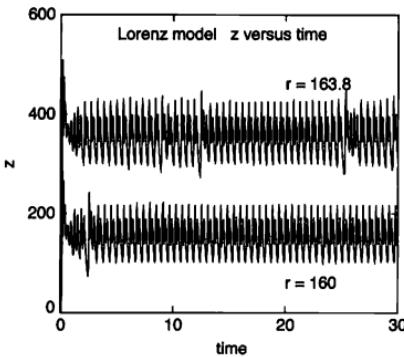


FIGURE 3.18: Variation of the Lorenz variable z as a function of time, for high values of r . Calculated using the Euler method with a time step of 0.0001. The results here show that the system is periodic at $r = 160$, and that it starts to develop some irregular features at $r = 163.8$. The threshold value of r at which this transition to chaos is observed is very sensitive to the accuracy of the approximation. So, a more accurate calculation (e.g., with a higher order Runge-Kutta method) can yield a slightly different threshold value of r . However, the general result is the same, with periodic behavior at small r , changing to chaotic behavior as r is increased.

that we construct an extremely detailed computer model of the atmosphere (this problem is not in the exercises). The model would take all atmospheric conditions, including the temperature, wind velocity, etc., as functions of position around the world at some particular time as input parameters, and then calculate how these conditions vary with time. However, assuming that the atmosphere is in its chaotic regime (and it appears that it is), any slight error in any of the initial conditions would rapidly lead to an enormous error in the predictions. This conclusion should be no surprise; we all know how bad current weather forecasts can be, and these are based largely on computer models (or the human equivalent!). This situation led to the butterfly metaphor that appeared in the title of a talk by Lorenz, “Predictability: Does the Flap of a Butterfly’s Wings in Brazil Set Off a Tornado in Texas?”²⁵ This is just another reference to the extreme sensitivity of chaotic systems to their initial conditions.

EXERCISES

- *3.25. Study period-doubling in the Lorenz model by examining the behavior for $r \leq 160$. Calculate the bifurcation diagram and extract the value of Feigenbaum’s δ parameter. You should find a value similar to that calculated for the pendulum.

²⁵Given at a meeting of the American Association for the Advancement of Science in 1972. It seems fitting that the trajectory in Figure 3.16 bears some resemblance to a butterfly.

Hint: While this problem can be done using the Euler method, it is probably advisable, in order to conserve computer time, to use the Runge-Kutta algorithm.

- 3.26. Continue the previous problem, and construct the phase-space plots as in Figures 3.16 and 3.17 in the different regimes.
- 3.27. Show that the Poincaré sections in Figure 3.17 are independent of the initial conditions. For example, compare the attractor found for $x(0) = 1$, $y(0) = z(0) = 0$ with that found for $x(0) = 0$, $y(0) = z(0) = 1$.
- *3.28. Estimate qualitatively the Lyapunov exponent for a few trajectories of the Lorenz model near the transition to chaos at $r = 24.74\dots$. Try to observe this exponent change from negative in the nonchaotic regime, to positive in the chaotic regime.
- *3.29. Explore the intermittency route to chaos for $r \geq 163$ in more detail. Begin by calculating z as a function of time for different values of r . Try $r = 163$ (which should be in the nonchaotic regime), and several larger values up to $r = 165$ or so. For the larger values of r you should observe chaotic “hiccups” like those found in Figure 3.18. Next calculate the average time between these hiccups and study how it diverges as the transition to chaos is approached. While the idea here is easy to explain, writing a program to detect hiccups is a bit tricky. One way to accomplish this is to construct a histogram of times between adjacent maxima in $z(t)$. In the oscillatory (nonchaotic) regime these times will all be the same. An odd value signals a hiccup.

3.7 THE BILLIARD PROBLEM

So far we have considered two different chaotic systems, and you are probably willing to believe that there are many more. To get an appreciation for the different kinds of behavior that can be found, and also the common threads that run through this behavior, we will consider one more chaotic model in this chapter. Here we consider the problem of a ball moving without friction on a horizontal table. We imagine that there are walls at the edges of the table that reflect the ball perfectly and that there is no frictional force between the ball and the table. We can think of this as a billiard ball that moves without friction on a perfect billiard table.²⁶ The ball is given some initial velocity, and the problem is to calculate and understand the resulting trajectory. This is known as the stadium billiard problem.

Except for the collisions with the walls, the motion of the billiard is quite simple. Between collisions the velocity is constant so we have

$$\frac{dx}{dt} = v_x , \quad (3.30)$$

$$\frac{dy}{dt} = v_y ,$$

where v_x and v_y change only through collisions with the walls. These equations can be solved using our usual Euler algorithm. Note that since the velocity is constant (except during the collisions), the Euler solution gives an exact description of the

²⁶We will ignore any complications associated with the angular momentum of the ball, so it is better to think of this as a particle sliding on a frictionless sheet of ice. It would thus be more accurate to term this the “hockey puck” problem, but the name billiard is already firmly attached to the model.

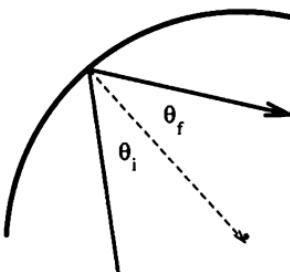


FIGURE 3.19: Geometry for perfect reflection of the billiard from a wall. The angle of incidence is equal to the angle of reflection, $\theta_i = \theta_f$.

motion across the table. The most difficult part of the calculation is the treatment of the collisions. Since we have assumed that they are perfectly elastic, the reflections will be mirrorlike, which means that the angle of incidence will be equal to the angle of reflection. These angles are defined in terms of the incoming and outgoing velocity vectors, and the vector normal to the wall at the location of the collision. This geometry is shown in Figure 3.19, where we have drawn a curved wall; our arguments apply just as well to a straight wall.

A numerical solution for the billiard's motion thus consists of two parts. First, when the billiard is away from the wall its motion is described by (3.30). These equations of motion are used to integrate forward in time, calculating x and y as functions of time. After each time step we must check to see if there has been a collision with one of the walls, that is, if the newly calculated position puts the billiard off the table. When this happens the program must backtrack to locate the position where the collision occurred. There are several ways to do this. One way is to back the billiard up to the position at the previous time step and then use a much smaller time step [for example, a factor of 100 smaller than the time step used to initially integrate (3.30)], so as to move the billiard in much smaller steps. When the billiard then goes off the table (again), we take the location after that iteration to be the point of collision.²⁷ After locating the collision point, we need to do a little vector manipulation. The initial velocity vector $\vec{v}_i \equiv (v_x, v_y)$ is already known, since v_x and v_y are known. We must next obtain the unit vector normal to the wall at the point of collision, \hat{n} . It is then useful to calculate the

²⁷This approach will always yield a collision point that is off the table by a small amount. We can imagine other ways to locate the collision point, but in most cases they will never locate the collision point exactly. We have found that the approach described here yields results that are essentially identical to other methods for locating the collision point. A different method will be considered in the exercises.

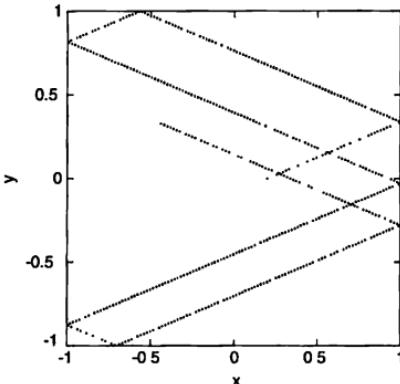


FIGURE 3.20: Trajectory of a billiard on a square table. Only the first few reflections are shown. The billiard started at $x = 0.2$, $y = 0$ with a speed of unity and with the velocity directed toward the upper right of the table. The time step was 0.01.

components of \vec{v}_i parallel and perpendicular to the wall. These are just

$$\begin{aligned}\vec{v}_{i,\perp} &= (\vec{v}_i \cdot \hat{n}) \hat{n}, \\ \vec{v}_{i,\parallel} &= \vec{v}_i - \vec{v}_{i,\perp}.\end{aligned}\quad (3.31)$$

Once we have the components of \vec{v}_i we can reflect the billiard. A mirrorlike reflection reverses the perpendicular component of velocity, but leaves the parallel component unchanged (we'll leave it to the reader to show that this makes $\theta_f = \theta_i$ in Figure 3.19). Hence, the velocity after reflection from the wall is

$$\begin{aligned}\vec{v}_{f,\perp} &= -v_{i,\perp}, \\ \vec{v}_{f,\parallel} &= v_{i,\parallel}.\end{aligned}\quad (3.32)$$

Some results are given in Figure 3.20, which show the first few bounces for a billiard on a square table. When (if) you write your own program for this problem, a graphical display of the trajectory is extremely useful in finding, and fixing, any errors. Two strong tests of the program are that the reflections should indeed be mirrorlike (usually referred to as *specular*), and that the energy, which is all kinetic for this problem, should be conserved. The trajectory on a square table is, as we might infer from Figure 3.20, very regular; it has a very simple predictable pattern. This is confirmed in Figure 3.21, which shows such a trajectory over a much longer time period.

Another way to graphically capture the regularity of the trajectory is with yet another phase-space plot. In Figure 3.21 we show a plot of v_x versus x , but

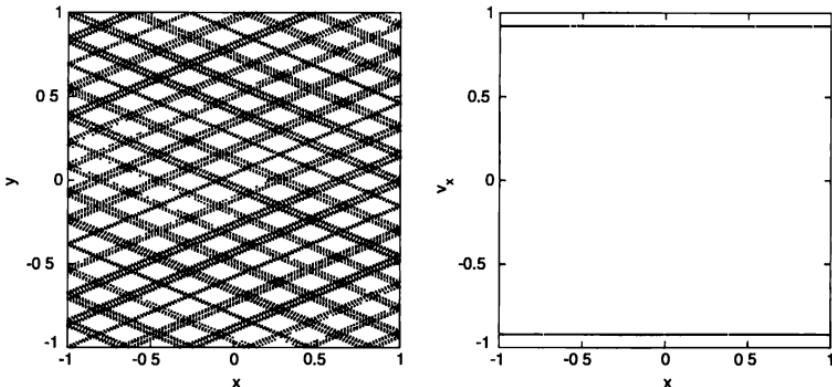


FIGURE 3.21: Left, trajectory of a billiard on a square table. This is a continuation of the trajectory shown in Figure 3.20. Right, corresponding Poincaré section derived from the trajectory shown on the left. Note that the phase-space plot was obtained from a much longer run of the program than was used for the trajectory plot. There are a few gaps visible in the phase-space plot; these would be filled in if the program were run for a longer period of time.

here we have not plotted every point of the phase-space trajectory. Rather, we have constructed another type of Poincaré section by plotting the points only when the billiard crosses the $y = 0$ axis.²⁸ We find two horizontal lines here; the billiard trajectories are all parallel to one of two different directions, so only two different values of v_x occur. Since the billiard can cross the $y = 0$ axis anywhere, the values of x in this plot vary continuously from -1 to $+1$.

The behavior of the billiard gets more interesting when we consider other table shapes. There are many possibilities; here we will consider only one, the so-called stadium shape, which can be described as follows. Imagine a circular table of radius $r = 1$, as shown on the left side of Figure 3.22. Now cut the table along the x axis, and pull the two semicircular halves apart (along y), a distance $2\alpha r$. Then fill in these two open sections with straight segments. Thus $\alpha = 0$ yields a circular table, while nonzero values of α give a table with a more traditional stadium shape. Figure 3.22 compares trajectories for a circular table with those for a table with $\alpha = 0.01$. While the trajectories depend on the initial conditions (the initial values of x , y , and \dot{v}), the results for the circular table are always highly symmetric. On the other hand, the trajectory for the $\alpha = 0.01$ stadium is much more complicated and is definitely not symmetric, except for very special initial conditions (such conditions were not used in Figure 3.22).²⁹ This should remind you of chaotic motion.

²⁸This should remind you of how we dealt with the Lorenz model.

²⁹Examples of such special, nonchaotic initial conditions are $x(0) = y(0) = 0$, with \dot{v} parallel to either the x or y axis.

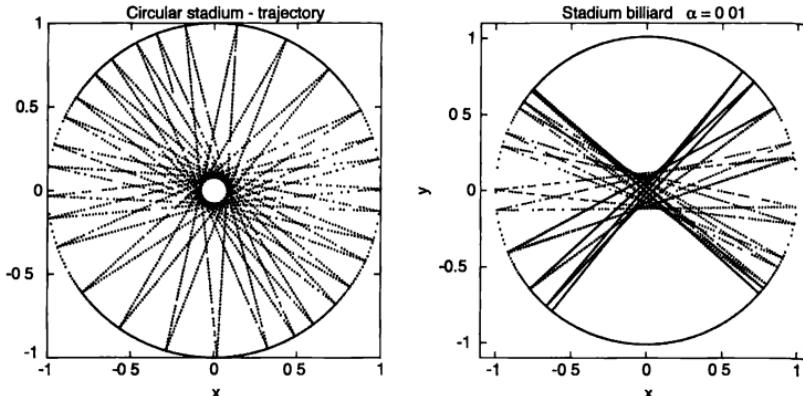


FIGURE 3.22: Left: trajectory of a billiard on a circular table, right. trajectory of a billiard on a stadium-shaped table with $\alpha = 0.01$.

The corresponding phase-space plots of v_x versus x (constructed as in Figure 3.21) are shown in Figure 3.23. The very ordered pattern for the circular table confirms our impression from the trajectories, that this is a nonchaotic system. However, for the $\alpha = 0.01$ stadium the phase-space plot is somewhat reminiscent of the chaotic attractor we found for the pendulum problem; it is indeed chaotic. Two more phase-space plots for other stadium shapes are shown in Figure 3.24, and both are seen to be chaotic.

A hallmark of a chaotic system is an extreme sensitivity to initial conditions. This property is also found in the billiard problem, as can be seen if we calculate the trajectories of two billiards with slightly different initial conditions. An example is shown in Figure 3.25 where we plot the distance between two billiards as a function of time. The billiards were on a chaotic table ($\alpha = 0.01$) and were given the same initial velocities, but were started a distance 1×10^{-5} apart (recall that the table has a radius of approximately 1 unit). The billiard separation shows a very sharp dip after about every one time unit. These dips occur when the billiards collide with the walls, as this causes their trajectories to cross. The overall separation is seen to increase very rapidly with time (note the logarithmic scale). The divergence of these trajectories can be described by a Lyapunov exponent, as we found for the pendulum.³⁰

A remarkable feature of our results for the billiard problem is that the chaotic behavior is evident even for very small values of α . It turns out that the stadium billiard is chaotic for *any* nonzero value of α . In fact, only tables with very high

³⁰To calculate the Lyapunov exponent quantitatively we would have to average the behavior over different initial conditions, so as to smooth out the irregularities in Figure 3.25

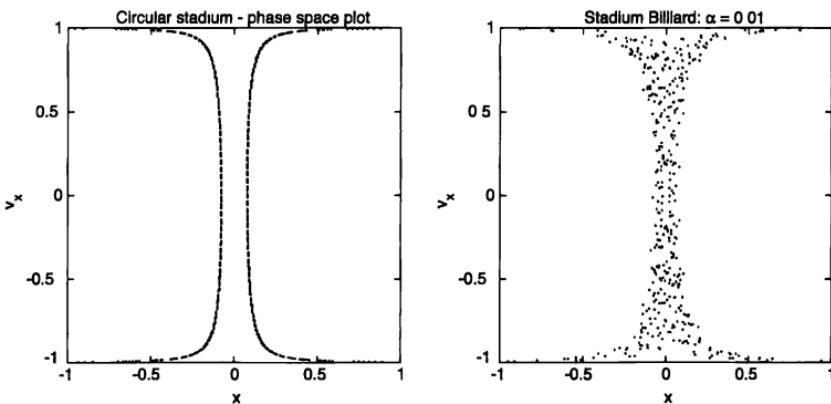


FIGURE 3.23: Phase-space plots for the trajectories shown in Figure 3.22. Left: for a circular-shaped table; right: for a stadium-shaped table with $\alpha = 0.01$. These were constructed by plotting points only when $y = 0$.

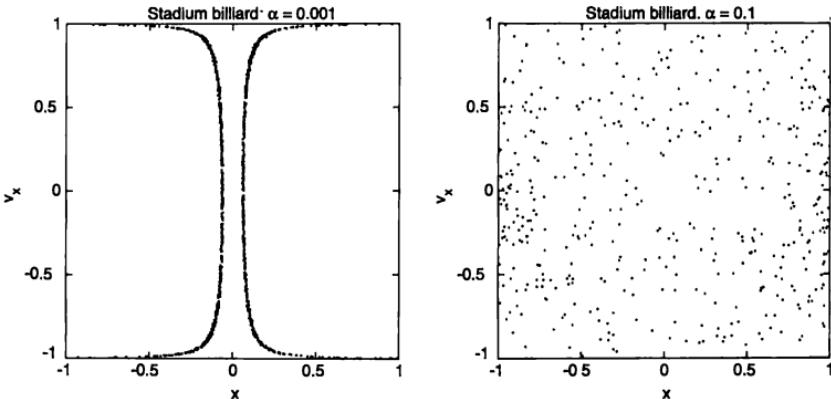


FIGURE 3.24: Phase-space plots for two more stadium-shaped tables. Left: for a table with $\alpha = 0.001$; right: table with $\alpha = 0.1$.

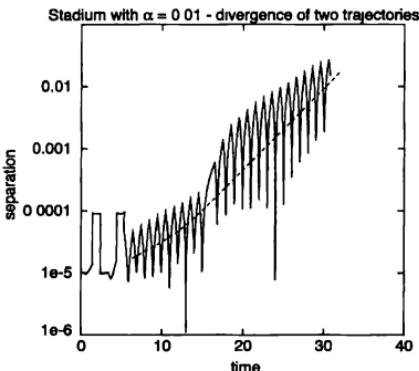


FIGURE 3.25: Divergence of the trajectories of billiards started at slightly different initial conditions, for a stadium-shaped table with $\alpha = 0.01$. The dashed line is drawn to emphasize the rapid overall increase of the separation with time. The initial separation of the billiards was 1×10^{-6} .

symmetry are nonchaotic. The billiard problem may be relevant for describing the motion of gas molecules in a container. Our results suggest that for any realistically shaped container (i.e., any shape that is not extremely symmetric, such as the perfectly circular table) such motion is likely to be chaotic and thus unpredictable. This finding will be relevant to our discussion of entropy and the approach to equilibrium in Chapter 7.

EXERCISES

- 3.30. Investigate the Lyapunov exponent of the stadium billiard for several values of α . You can do this qualitatively by examining the behavior for only one set of initial conditions for each value of α you consider, or more quantitatively by averaging over a range of initial conditions for each value of α
- *3.31. Study the behavior for other types of tables. One interesting possibility is a square table with a circular interior wall located either in the center, or slightly off-center. Another possibility is an elliptical table.
- *3.32. The key part of a program for the billiard problem is the treatment of collisions with the wall of the stadium, and one way of doing this was described above. Another way is to use the exact solution of (3.30) to compute the trajectory and then solve analytically (using the equation that specifies the perimeter of the stadium) for the location of the collision. Write a program that uses this method and compare your results with those given in this section.

3.8 BEHAVIOR IN THE FREQUENCY DOMAIN: CHAOS AND NOISE

Our intuitive ideas concerning what it means to be chaotic usually include some connection with terms such as *random*, *unpredictable*, and *noisy*. We have already

explored the first two notions; in this section we consider how chaotic behavior is connected with noise. For this we require several tools for dealing with time-dependent signals. These tools are discussed in Appendix C and rely on the Fourier transform.³¹

Our goal in this section is to Fourier analyze the time-dependent signals obtained in our simulations of the damped, nonlinear pendulum in Section 3.3. We will consider only the signals associated with the angular position of the pendulum, $\theta(t)$, although the same sort of analysis could be used with the angular velocity, $\omega(t)$. Such a signal can, in general, be a complicated function of time. Nevertheless, we show in Appendix C how it can always be decomposed into component waveforms that are simple sines and cosines. If you choose to write your own program to calculate this Fourier decomposition, we recommend that you use the fast Fourier transform (FFT) algorithm described in Appendix C (another option is to use the FFT routine in your favorite software package).

In this section we will be concerned only with the frequency spectrum associated with $\theta(t)$, which can be derived using its power spectrum. Here the term *power* is used in the following sense. Typical signals of interest include the amplitudes of pressure waves, electrical voltages, and light waves. In such cases the square of the amplitude of a particular frequency component of the signal is proportional to the *intensity* of the signal at that frequency, which is in turn proportional to the power carried by the signal. An examination of the intensity as a function of frequency leads to the *power spectrum* discussed in Appendix C. This term is commonly used, even in cases (such as the present one) where the connection with power and energy is not direct.³²

The power spectra of several pendulum waveforms³³ are shown in Figures 3.26 and 3.27, where we show the power spectrum of $\theta(t)$ as a function of frequency for different values of F_D . The area under each peak in the spectrum is proportional to the effective power, that is, the sum of the squares of the corresponding Fourier components of the signal. At low drive, $F_D = 0.5$, the pendulum is in a period-1 state, in which the $\theta(t)$ waveform is very close to a simple sine wave. The FFT result shows a single peak at the frequency of this sine wave, that is, at the drive frequency. This is completely analogous to what we found for the FFT of a sine wave in Appendix C. At somewhat higher drive $F_D = 0.95$, the behavior is again period-1, and we see that the power spectrum is again dominated by a single peak at the drive frequency. We also notice a very small peak at approximately three

³¹Readers who are not familiar with the concept of Fourier analysis should review Appendix C before tackling this section.

³²Although with the pendulum, the power spectrum of $\theta(t)$ will be closely connected with the kinetic energy of the system.

³³In calculating these power spectra, it is useful to use the time series data for $\theta(t)$ which have been shifted to keep the pendulum angle in the range $|\theta| < \pi$, as explained in connection with Figure 3.6. Using the unshifted values of θ from Section 3.3 can lead to large low frequency components in the power spectrum (you might think about why this is so). Also, in this and other situations where you are using Fourier transforms, it is essential to first formulate a strategy about what value of the time step Δt should be used, and over how many time steps you should sample the time series. Although a small Δt may be required to accurately track the time evolution of the nonlinear system (making the Nyquist frequency very high), it might be useful to obtain a high frequency resolution, which requires a long time series.

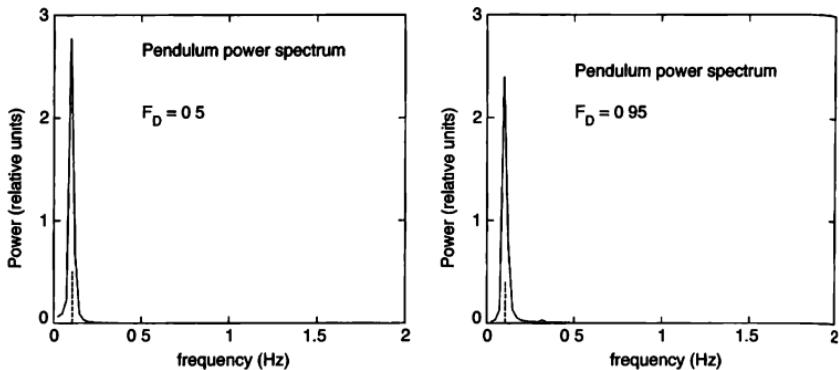


FIGURE 3.26: Fourier analysis of the results for $\theta(t)$ for the nonlinear pendulum at different values of the driving force. At $F_D = 0.5$ and $F_D = 0.95$ the pendulum is in the period-1 regime. The dashed lines indicate the drive frequency

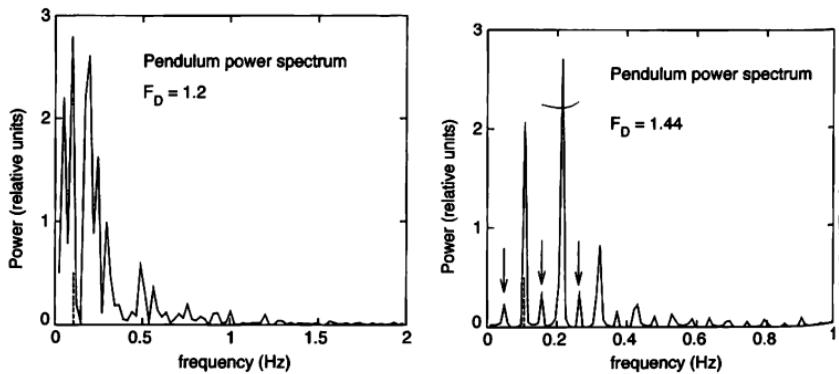


FIGURE 3.27: Left: With $F_D = 1.2$ the pendulum is chaotic. Right: When $F_D = 1.44$ the pendulum is in the period-2 regime. The dashed line indicates the drive frequency. The arrows on the right indicate the subharmonic (period-doubled) component, which appears at half the drive frequency, and some of its harmonics.

times the drive frequency (≈ 0.3 Hz). This peak is not a plotting error! It is produced by the nonlinearity of the waveform at this drive and is an example of the phenomenon of nonlinear mixing we mentioned earlier. The most interesting result is found in the chaotic regime, $F_D = 1.2$ (on the left in Figure 3.27). The spectrum is now very complicated as the power is broadly distributed over a wide range of frequencies. This is just the noise that we intuitively expect to find in a chaotic system.

It is also interesting to examine the power spectra when the pendulum undergoes period-doubling on its way to the chaotic regime. Results from the period-2 regime are shown on the right in Figure 3.27. There is a large peak at the drive frequency $\Omega_D/2\pi \approx 0.1$ Hz, with additional peaks at integer multiples of this frequency, as we again have the nonlinear mixing observed earlier. However, there is now a strong component at half this frequency, ≈ 0.05 Hz. This corresponds to a component with twice the period and is, therefore, the result of *period-doubling*. A similar analysis can be used to examine the behavior in the period-4, period-8, etc., regimes. If the behavior is period- n , there will be a spectral component at a frequency $1/n$ times the drive frequency. Hence, spectral analysis reveals the period-doubling route to chaos in an extremely clear manner.

The key result of this section is that much can be learned about the behavior of a system by examination of its frequency spectrum. Here we have used this approach to study the pendulum in and near its chaotic states. In later chapters we will use the same method in connection with several other problems.

EXERCISES

- 3.33. In Figure 3.26 we saw that at a relatively high drive, $F_D = 0.95$, there was a small, but noticeable response of the pendulum at three times the frequency of the driving force. Calculate the size of this component as a function of the drive force in the range $F_D = 0.95 - 1.00$. Try also to observe a component at five times the drive frequency. The process in which these signals at multiples of the drive frequency are produced is an example of mixing.
- 3.34. Analyze the behavior of the nonlinear pendulum in the period-4 regime and show that the spectral component with the lowest frequency has a frequency of one-fourth the drive frequency.
- *3.35. We saw in connection with Figures 3.10 and 3.11 that every time a period-doubling threshold is crossed a new subharmonic component is added to the $\theta(t)$ waveform. The size of this component can be readily extracted using the Fourier transform. Calculate $\theta(t)$ for values of the drive amplitude near the period-2 transition in Figure 3.11. Then use the FFT to obtain the amplitude of the period-2 component as a function of F_D . Try to determine the functional form that describes the way in which this amplitude vanishes at the transition.
- 3.36. Analyze the power spectrum of $\omega(t)$ of the nonlinear pendulum for different values of the driving force.
- 3.37. Calculate the frequency spectra for the waveforms $z(t)$ for the Lorenz model. Compare the behavior in the chaotic, nonchaotic, intermittent, and period-doubled regimes.

The Solar System

In Chapter 2 we found that atmospheric drag plays a major role in the behavior of projectiles moving near the earth's surface. In some respects, this drag and other types of "friction" obscure the fundamental physical principles that govern the behavior.¹ If we want to study the essential consequences of these principles, it thus seems best to consider a system in which frictional forces are as small as possible. The solar system is just such a laboratory, and not surprisingly, studies of the motion of planets and moons provided major inspiration to the founders of classical mechanics.

In the present chapter we will consider several problems that arise in the study of planetary motion. We begin with the simplest situation, a sun and a single planet, and investigate a few of the properties of this model solar system. While a computational approach is not required in this case, the algorithm we develop will prove useful for later problems. Additionally, comparisons of the numerical results obtained with this algorithm with exact solutions will provide valuable insight into the nature of our approximations.

4.1 KEPLER'S LAWS

Figure 4.1 shows our hypothetical solar system. There is one planet, which we will refer to as Earth, in orbit around the Sun, and the only force in the problem is gravity. According to Newton's law of gravitation the magnitude of this force is given by

$$F_G = \frac{G M_S M_E}{r^2}, \quad (4.1)$$

where M_S and M_E are the masses of the Sun and Earth, r the distance between them, and G is the gravitational constant. Let us assume for now that the Sun's mass is sufficiently large that its motion can be neglected.² Our goal is to calculate the position of Earth as a function of time. From Newton's second law of motion

¹For example, Newton's first law is most clearly observed with systems in which friction is negligible.

²For a two-body system like the Sun-Earth solar system this is not a real problem since the treatment is essentially the same even when the Sun's finite mass is considered (through the notion of a reduced mass). However, for systems containing more than two bodies, such a reduction is not possible and this becomes a potential problem. Even so, in the real solar system, the Sun is so much more massive than the planets and other bodies, that this approximation is very good. Moreover, numerically it is straightforward to also allow for the Sun's motion. This will be explored later.

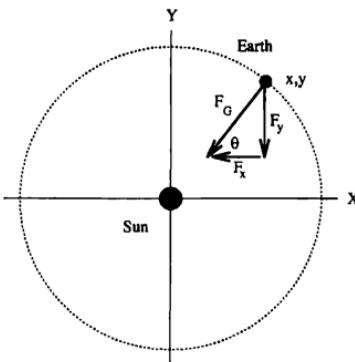


FIGURE 4.1: Coordinate system for describing the motion of Earth in orbit around the Sun. The Sun is at the origin and Earth is located at coordinates (x, y) .

we have

$$\begin{aligned}\frac{d^2x}{dt^2} &= \frac{F_{G,x}}{M_E} \\ \frac{d^2y}{dt^2} &= \frac{F_{G,y}}{M_E},\end{aligned}\quad (4.2)$$

where $F_{G,x}$ and $F_{G,y}$ are the x and y components of the gravitational force. From Figure 4.1 we have

$$F_{G,x} = -\frac{G M_S M_E}{r^2} \cos \theta = -\frac{G M_S M_E x}{r^3}, \quad (4.3)$$

with a similar result for $F_{G,y}$. Here the negative signs remind us that the force is directed toward the Sun, which is located at the origin of our coordinate system.

We now follow our usual approach and write each of the second-order differential equations in (4.2) as two first-order differential equations

$$\begin{aligned}\frac{dv_x}{dt} &= -\frac{G M_S x}{r^3} \\ \frac{dx}{dt} &= v_x \\ \frac{dv_y}{dt} &= -\frac{G M_S y}{r^3} \\ \frac{dy}{dt} &= v_y.\end{aligned}\quad (4.4)$$

These can be converted into difference equations that can be solved numerically. However, before we proceed with that, it is useful to consider the choice of units. One option is to simply use SI units. There is no difficulty with this, except that meters and seconds do not match the scale of the problem. For example, the radius of Earth's orbit is $\sim 1.5 \times 10^{11}$ m. If we were to use SI units, a graph showing this orbit around the Sun would have labels of 1×10^{11} , 2×10^{11} m, etc. This would be awkward, though not impossible so. It is much more convenient to use *astronomical units*, AU, which are defined as follows. One astronomical unit of length, known simply as 1 AU, is the average distance between the Sun and Earth ($\approx 1.5 \times 10^{11}$ m). It is convenient to measure time in years (1 year $\approx 3.2 \times 10^7$ s) since this unit matches the solar system better than, say, seconds. For the remainder of this chapter we will, therefore, use astronomical units for distance, and measure time in years, unless specifically noted otherwise.

To complete our system of units we also need the corresponding unit of mass. This is easily derived if we recall that Earth's orbit is, to a very good approximation, circular. For circular motion we know that the force must be equal to $M_E v^2/r$, which leads to

$$\frac{M_E v^2}{r} = F_G = \frac{G M_S M_E}{r^2}, \quad (4.5)$$

where v is the velocity of Earth. Rearranging we find

$$G M_S = v^2 r = 4 \pi^2 \text{AU}^3/\text{yr}^2, \quad (4.6)$$

where we have used the fact that (since the orbit is circular) the velocity of Earth is $2\pi r/(1 \text{ yr}) = 2\pi (\text{AU}/\text{yr})$ (recall that $r = 1 \text{ AU}$). Since G and M_S appear only as a product in (4.4) there is no need to express either term separately.

We next convert the equations of motion (4.4) into difference equations in preparation for constructing a computational solution. We find

$$\begin{aligned} v_{x,i+1} &= v_{x,i} - \frac{4\pi^2 x_i}{r_i^3} \Delta t \\ x_{i+1} &= x_i + v_{x,i+1} \Delta t \\ v_{y,i+1} &= v_{y,i} - \frac{4\pi^2 y_i}{r_i^3} \Delta t \\ y_{i+1} &= y_i + v_{y,i+1} \Delta t, \end{aligned} \quad (4.7)$$

where, as usual, Δt is the time step and the factors of $4\pi^2$ ($= GM_S$) signal that we are using astronomical units. Note that we have used the Euler-Cromer method. That is, we use the *previous* values of position and velocity to update the velocities, while the *previous* values of position and the *new* values of velocity are used to update the positions. As we discussed in Chapter 3, the Euler method is not a good choice for oscillatory problems, and planetary motion is just such a problem. If we were to use the Euler method here we would find that the energy of the planet would grow with time, and it would spiral away from the Sun. This difficulty is

avoided with the Euler-Cromer method, since it conserves energy exactly over the course of each orbit.

We can use the techniques developed in previous chapters to translate (4.7) into a suitable program. The result for the subroutine `planet` which performs the time evolution of the position and velocity of Earth is given below.

EXAMPLE 4.1 Pseudocode for subroutine `planet`

- At each time step i calculate the position (x, y) and the velocity (v_x, v_y) for time step $i + 1$ using the Euler-Cromer method.
 - ▷ Calculate the distance r_i from the sun: $r_i = (x_i^2 + y_i^2)^{1/2}$.
 - ▷ Compute $v_{x,i+1} = v_{x,i} - \frac{4\pi^2 x_i}{r_i^3} \Delta t$ and $v_{y,i+1} = v_{y,i} - \frac{4\pi^2 y_i}{r_i^3} \Delta t$.
 - ▷ The Euler-Cromer step: calculate x_{i+1} and y_{i+1} using $v_{x,i+1}$ and $v_{y,i+1}$:
 $x_{i+1} = x_i + v_{x,i+1} \Delta t$, $y_{i+1} = y_i + v_{y,i+1} \Delta t$.
 - ▷ Record the new position or plot it as it becomes available.
 - ▷ Repeat for desired number of time steps.

In analyzing planetary motion, it is very useful to visualize the movement of the planet graphically. This can be done by plotting the position of the planet as it becomes available during the calculation (i.e., in "real time," so to speak). Though this is often not simple to do (e.g., if you are using programming languages such as C or fortran), with some effort you can almost always find graphic subroutine libraries that can be combined with your program to make this possible. This is one case where you will be glad you went to the extra trouble. When plotting planetary orbits, some details are worth particular attention. One such detail is the screen aspect ratio. It is very helpful to arrange for a circular orbit to appear as truly circular. However, most computer displays are not "square," that is, they have different "intrinsic" length units along x and y . We strongly suggest that you compensate for this when displaying planetary orbits. Another important detail in such animated plotting is to make the "current" position of the planetary bodies readily apparent. This can easily be done by using color to distinguish the current position of a planet from previously plotted points along the trajectory. It is also useful to show a running display of the elapsed time beside the orbit.

The routine `planet` can be used to simulate the motion of the Earth or any other real or hypothetical planet in orbit about the Sun. For your convenience, Table 4.1 lists orbital data for the planets in our solar system. The entry under "eccentricity" refers to the shape of each orbit and will be discussed further in the next section.

We have used our planetary orbit program to simulate the motion of Earth, and the results are shown in Figure 4.2. To perform this calculation we need to specify the initial conditions. As we have already noted, Earth's orbit is circular to a very good approximation. From Table 4.1 we see that the radius of the orbit is 1.00 AU (which we already knew from the definition of astronomical units), so for

TABLE 4.1: Some useful planetary data For all of the planets except Mercury and Pluto the orbits are very nearly circular. The orbits for Mercury and Pluto are noticeably elliptical (i.e., they deviate substantially from circular), and in these cases the entry under radius gives the semimajor axis of the ellipse (see Figure 4.3) Note that the mass of the Sun is 2.0×10^{30} kg.

planet	mass (kg)	radius (AU)	eccentricity
Mercury	2.4×10^{23}	0.39	0.206
Venus	4.9×10^{24}	0.72	0.007
Earth	6.0×10^{24}	1.00	0.017
Mars	6.6×10^{23}	1.52	0.093
Jupiter	1.9×10^{27}	5.20	0.048
Saturn	5.7×10^{26}	9.54	0.056
Uranus	8.8×10^{25}	19.19	0.046
Neptune	1.03×10^{26}	30.06	0.010
Pluto	$\sim 6.0 \times 10^{24}$	39.53	0.248

our simulation we have chosen an initial x coordinate of 1.0 and a y coordinate of zero. It is crucial that we also choose the proper initial velocity to obtain a circular orbit.³ To estimate the value required to yield a circular orbit we recall that Earth completes one orbit in one year. The velocity is thus $2\pi r/1 = 2\pi$ AU/yr. Using these initial conditions we obtain the nicely circular orbit shown in Figure 4.2. If we had let the program run for many orbits, we would have found that the calculated path of the Earth repeats itself to within the size of the points in the figure. We will see below that this repeatability is a general feature of closed orbits in a two-body solar system; that is, a system with one planet and one sun. The result in Figure 4.2 also demonstrates that the Euler-Cromer method is quite suitable for solar system calculations, and we will use it throughout this chapter.⁴

We can use our program to investigate Kepler's laws for planetary motion. You may recall that Johannes Kepler was a 17th century astronomer who performed a very careful study of the planetary data acquired earlier by Tycho Brahe. Kepler showed that Brahe's observations were consistent with the following three statements:

- All planets move in elliptical orbits, with the Sun at one focus.
- The line joining a planet to the Sun sweeps out equal areas in equal times.
- If T is the period and a the semimajor axis of the orbit (Figure 4.3), then T^2/a^3

³As we know from Kepler's Laws, any velocity (as long it is less than the escape velocity¹) will produce a stable, elliptical orbit. However, only one value of the initial velocity will yield a circular orbit. When studying elliptical orbits, the size and orientation of the ellipse will depend on the magnitude and direction of the initial velocity. If you choose it to be in the y direction, you will get an elliptical orbit oriented with its major axis along the x axis with Sun (the origin) at one of its foci. Further, if you choose a value of v that is too small, you will get the Sun at the far focus, while a value that is too large will have the Sun at the near focus.

⁴Other methods, such as the Runge-Kutta and Verlet methods discussed in Appendix A, also work well for this problem.

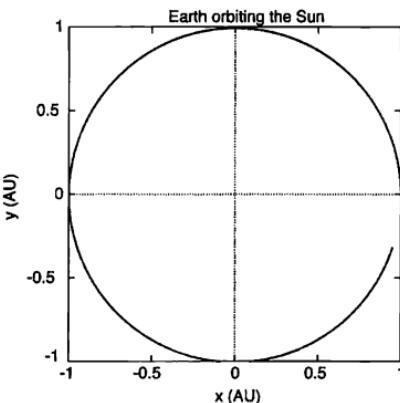


FIGURE 4.2: Results for a simulation of Earth orbiting around the Sun. The Sun was located at the origin ($x = 0$, $y = 0$), and the points on a circle with radius 1 AU (which are so close together that they partially overlap) show the calculated path of Earth. The time step was 0.002 yr, and the initial conditions were $x = 1$, $y = 0$, $v_x = 0$, $v_y = 2\pi$ (in AU and AU/yr, respectively). The program was stopped just before one orbit was completed.

is a constant (i.e., the same constant for all of the planets). Note that for a circular orbit, a is just the radius of the orbit.

It can be shown analytically that all three of Kepler's laws are consequences of the fact that the gravitational force follows an inverse-square law, (4.1), and we will have more to say about such matters in Section 4.2. Here we want to consider how our planetary orbit program can be used to confirm Kepler's third law. For simplicity we consider only the planets whose orbits are circular (we'll deal with the case of elliptical orbits in a later section). Table 4.1 gives the distance from the Sun, and by choosing the initial velocity appropriately⁵ we can obtain circular orbits and thus simulate the motion of any of the planets. It is convenient to modify the program slightly to print out the value of the time when a planet passes a particular point on the orbit (such as where it crosses the x axis), as this makes it easier to accurately determine the period of an orbit. We'll leave this job to the reader. Table 4.2 lists the calculated values of T^2/a^3 for the planets Venus through Saturn. These planets all have orbits that are very nearly circular, so a is just the

⁵As we have already noted, choosing the proper initial conditions is essential. Here we have used the initial positions given in Table 4.1 and chosen the initial velocity by trial and error so as to give a circular orbit, in accord with the actual orbits of these planets. The same values of the initial velocities could have been obtained from arguments like those used to obtain (4.5), but in a way this would be begging the question in a "test" of Kepler's third law. Our point here is simply to show that our planetary motion program correctly reproduces Kepler's third law for the particular case of circular orbits.

TABLE 4.2: Confirmation of Kepler's third law. The results were obtained using the planetary orbit program to calculate the orbital periods of several planets.

planet	T^2/a^3 (yr 2 /AU 3)
Venus	0.997
Earth	0.998
Mars	1.005
Jupiter	1.010
Saturn	0.988

orbital radius listed in Table 4.1. According to Kepler's third law, T^2/a^3 should be a constant, and in astronomical units this constant is unity. Our numerical results are in agreement with this prediction and thus confirm Kepler's third law. The time steps used in these calculations were 0.001 yr for Venus, Earth, and Mars and 0.005 yr for Jupiter and Saturn, and this is the cause of the (small) deviations of the calculated values of T^2/a^3 in Table 4.2 from the ideal value of unity. We will leave it to the exercises to confirm Kepler's third law for noncircular orbits.

EXERCISES

- 4.1. Investigate the results obtained from the planetary motion program with different values of the time step. Show that for simulations of Earth, time steps greater than about $\Delta t = 0.01$ yr do not lead to satisfactory results. For such large time steps the orbits are not stable (i.e., not repeating). This is in accord with our general rule of thumb (Chapter 1) that the time step should be no larger than 1 percent or so of the characteristic time scale of the problem. In this case the characteristic time scale is the period of one orbit. Now use other initial speeds, both larger and smaller than the value 2π which was needed for a circular orbit and again study the stability of the elliptical orbit to different values of the time step. (Try values like 4 and 8.) Do you see differences in the sensitivity to increasing time steps?
- 4.2. Change the planetary motion program so that it uses the Euler method. Show that Earth always spirals away from the Sun no matter how small the time step is made. Compare your results with those obtained using the Euler-Cromer method.
- 4.3. Modify the planetary orbit program so that you can use it to verify Kepler's second law for the case of an elliptical orbit. As part of this problem you might also check quantitatively that the orbits are indeed elliptical.
- 4.4. Investigate the orbit of Halley's comet. This comet has an orbital period of 76 yr and a distance of closest approach to the Sun of 0.59 AU. Use trial and error to determine its maximum orbital speed and maximum distance from the sun. How does its maximum distance compare with the orbit of Pluto?
- 4.5. Extend your (Euler-Cromer) orbit program so that it calculates the energy (kinetic, potential, and total) of the planet, and also the angular momentum. Consider the following issues.
- 4.6. Begin with a circular orbit and show that both the kinetic and potential energies are constants. The angular momentum should also be a constant.

- 4.5. Consider an elliptical orbit. An orbit with an initial position 1 AU from the Sun and a velocity of 5 AU/yr is a convenient choice. Show that while the kinetic and potential energies now vary as the planet moves through its orbit, their sum (the total energy) is a constant. Also show that the angular momentum is a constant during the course of the orbit. Prove that these results are direct consequences of the fact that the force of gravity is conservative, and is a central force (i.e., it acts along the line that runs between the Sun and the planet).
- 4.6. Consider a planet that begins a distance of 1 AU from the Sun. By trial and error, determine what its initial velocity must be in order for it to escape from the Sun. Compare your estimate with the exact result (which you should also calculate).
- *4.7. Consider a hypothetical solar system consisting of a sun and one planet in which the mass of the sun is not much greater than the mass of the planet. Now you must allow for the motion of both the planet and the sun. Extend your planetary motion program to include this effect. You will have to deal with a set of equations such as those in (4.7) for both objects. Investigate the possible types of orbital motion found in such a system. Begin with a double star system in which the two objects are of equal mass. Then explore the behavior when the masses are unequal. *Hint:* In order to obtain the simplest orbits, it is best to pick initial conditions such that the total linear momentum is zero. While this problem can be handled with a stationary sun together with the concept of a reduced mass, this calculation is a necessary prelude to the study of orbits of planets in binary star systems, which we will consider in a later exercise.

4.2 THE INVERSE-SQUARE LAW AND THE STABILITY OF PLANETARY ORBITS

In this section, we begin with a quick review of a few analytic results concerning the trajectory of a body (such as a planet or a comet) in a solar system assumed to contain only the Sun and the body. If you are not interested in brushing up on conic sections and the details of elliptical orbits, you can simply use our basic results, and pick up the physics below after (4.11).

We first note that for a two-body system, all three of Kepler's laws are consequences of the fact that the gravitational force follows an inverse-square law, (4.1). The details of such a calculation can be found in, e.g., Marion and Thornton (1995), but we will sketch some of the analytic results here to lay the ground for further numerical investigations.

We consider a two-body system in which the interaction force depends only on the separation r . The relative motion in this system can be studied as if it were a one-body system; i.e., as if one of the bodies is at rest, while the other orbits about it. The moving body in this equivalent system has a mass equal to the so-called reduced mass $\mu \equiv m_1 m_2 / (m_1 + m_2)$ where m_1 and m_2 are the masses of the two original bodies. The position of the equivalent body is given by the relative displacement $\vec{r} \equiv \vec{r}_2 - \vec{r}_1$ of the original two bodies. Of course, if $m_1 \gg m_2$, such as if body 1 were our Sun and body 2 one of its planets, $\mu \approx m_2$ and \vec{r} would be the position of the planet relative to an almost stationary Sun.

The orbital trajectory for a body of reduced mass μ is given in polar coordinates by

$$\frac{d^2}{d\theta^2} \left(\frac{1}{r} \right) + \frac{1}{r} = - \frac{\mu r^2}{L^2} F(r), \quad (4.8)$$

where $L = \mu r^2 \dot{\theta}$ is the angular momentum and $F(r)$ is the force acting on the body. For the solar system with Sun's mass M_S and a planet's mass M_P , the force is given by $F(r) = -GM_S M_P / r^2$. The angular momentum L is conserved since the system is invariant under rotation.

Since $F(r)$ has the inverse square form (i.e., $F(r) \propto 1/r^2$), (4.8) can be readily solved. For our solar system case, the solution can be expressed as

$$\frac{1}{r} = \left(\frac{\mu G M_S M_P}{L^2} \right) [1 - e \cos(\theta + \theta_0)], \quad (4.9)$$

or, choosing $\theta_0 = 0$ (which defines the axes of the ellipse),

$$r = \left(\frac{L^2}{\mu G M_S M_P} \right) \frac{1}{1 - e \cos \theta}. \quad (4.10)$$

This result does not give us the position of the planet as a function of time. Rather, it gives the shape of the orbital trajectory, $r(\theta)$. Equation 4.10 describes a *conic section*; it is a circle if $e = 0$, an ellipse if $0 \leq e < 1$, a parabola if $e = 1$, and a hyperbola if $e > 1$, with a focus at the origin. The value of e is known as the eccentricity. Thus, any closed orbit in a two-body system with an inverse square interaction must be elliptical with the location of one of the bodies at a focus, as in Kepler's first law. Kepler's second law amounts to the conservation of angular momentum L , a fact that we have already used.

Equation (4.10) is useful in calculating quantities such as the planet's closest approach (called the *perihelion*) at distance $r_{\min} = a(1 - e)$, the farthest point (called the *aphelion*) at $r_{\max} = a(1 + e)$, and the corresponding speeds v_{\max} and v_{\min} (see also Figure 4.3). These quantities often provide handy starting points in numerical calculations, e.g., as initial conditions. Note that the length of the semimajor axis of the ellipse, a , is given by $a = L^2 / [\mu G M_S M_P (1 - e^2)]$. Also, since the angular momentum $L = \sqrt{\mu G M_S M_P a (1 - e^2)}$ is conserved, we may set this equal to $\mu r_{\min} v_{\max}$ and to $\mu r_{\max} v_{\min}$. This leads to

$$v_{\max} = \sqrt{G M_S} \sqrt{\frac{(1 + e)}{a(1 - e)}} \left(1 + \frac{M_P}{M_S} \right) \quad (4.11)$$

$$v_{\min} = \sqrt{G M_S} \sqrt{\frac{(1 - e)}{a(1 + e)}} \left(1 + \frac{M_P}{M_S} \right).$$

The orbital period T can be obtained by dividing the area enclosed by the elliptical orbit by the constant rate at which area is swept out according to Kepler's second law. From this, Kepler's third law follows as $T^2/a^3 = 4\pi^2/[G(M_S + M_P)] \approx 4\pi^2/(GM_S)$. Thus, all of the Kepler's laws follow directly from the inverse-square functional form of the gravitational force.

Let us now consider the inverse square law from a somewhat different, and more "philosophical" perspective. In addition to Newton's law of universal gravitation, an inverse-square dependence is also found for the electric force between two charges (Coulomb's law), and this raises some interesting questions. Is there anything special or profound about the inverse-square dependence, or is it just an accident? Does the force vary exactly as $1/r^2$, or might it really go as $1/r^2 0000000001$? To answer the first question it is necessary to consider how the force "gets from one of the objects to the other." Or, to put it another way, how do the masses know about each other? A useful classical picture involves lines of force or (equivalently) field lines.⁶ Here we imagine that gravitational field lines emanate from all objects that have mass and that these lines radiate outward to infinity, that is, they never terminate. The number of such lines is proportional to the mass of the object, and the force felt by a second, nearby object is proportional to the number of lines that intersect it. As the separation between the two objects is increased, the number of intersecting field lines drops because the lines are spreading out over an ever larger surface area as they move away from their source.

The inverse-square law follows naturally from this field-line picture. The area of a sphere of radius r that surrounds an object is proportional to r^2 . Since a fixed number of lines, say N , emanate from an object, the density of these field lines as they cut through such a sphere is proportional to N/r^2 . The inverse-square law is thus a direct consequence of the field-line picture together with the geometry of the Euclidean space in which we live.⁷

These arguments concerning the inverse-square law have all been theoretical. We should also consider what the real world has to say about this matter. Since, as noted above, Kepler's laws are predicted to be a direct consequence of the inverse-square law, we can use experimental tests of Kepler's laws to determine if gravity really does follow an inverse-square law. One way to accomplish this can be seen in Figure 4.3, which shows the geometry of a hypothetical elliptical orbit. Among other things, Kepler's laws tell us that the orientation of such an orbit does not change with time. More precisely, if we have a solar system consisting of a sun and one planet, and that planet follows an elliptical orbit, it is predicted that the orientation of the axes of the ellipse does not change with time.

Now let us suppose that the force law deviates slightly from an inverse-square dependence.⁸ To be specific, suppose that the gravitational force is of the form⁹

$$F_G = \frac{G M_S M_E}{r^\beta}. \quad (4.12)$$

If $\beta = 2$, then we have an inverse-square law, but we also want to consider the motion of our planet for values of β that are different from 2. The behavior of

⁶A quantum mechanical picture would, of course, be somewhat different. However, an inverse-square law has profound implications in that case as well.

⁷Again, this a classical picture.

⁸As we will discuss shortly, general relativity leads to such deviations, although they have a functional form that is different from (4.12). We will consider the effect of general relativity on elliptical orbits in the next section.

⁹Note that if the value of β is different than 2, this will alter the units of G . For simplicity we will assume here that r is always measured in AU and use $GM_S = 4\pi^2$ throughout.

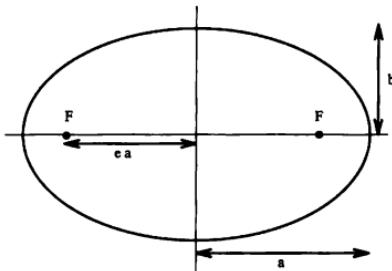


FIGURE 4.3: Hypothetical elliptical orbit. A sun lies at one of the foci of the ellipse, both foci are labeled F . The semimajor and semiminor axes are a and b , and the eccentricity is e . This drawing is not to scale.

elliptical orbits with this force law can be simulated with the planetary motion program given above by simply changing the exponent of r in the equations for the velocity.

Figure 4.4 shows a result for $\beta = 2$, that is, an inverse-square law. We have chosen the initial conditions to give an elliptical orbit, and it is seen that, as advertised, the sun is *not* at the center of the orbit. This is expected, since according to Kepler's first law the sun should be at one focus of the ellipse. The other point to note about this result is that we have plotted the position over the course of many orbits. The calculated orbit is seen to accurately retrace itself, demonstrating that the orientation of the ellipse does not change with time. This confirms some of the things we have claimed concerning Kepler's laws. Note that we used a time step smaller than was the case for the circular orbit shown in Figure 4.2. This is to make sure that any apparent precession is not a result of numerical errors. As it turns out, such problems occur much more easily for elliptical orbits than for circular ones.

Interesting things happen when we make β different from 2. Results for $\beta = 3.00$, an *inverse-cube* law, are shown in Figure 4.5. The behavior is *very* different from the inverse-square case. A planet in this solar system would not even follow a stable (i.e., repeating) orbit.¹⁰ The hypothetical planet in Figure 4.5 passes very near its sun and is then, because of numerical errors (due to the time step being too large during the close approach with the sun), ejected from the solar system. The result for $\beta = 2.50$ is not quite as dramatic; the planet is seen to follow an approximately elliptical path, with the axes of the ellipse rotating through an angle of nearly 360° after only three orbits.

As β is made closer to 2 the orbits become somewhat more stable, although

¹⁰You might think that if the orbit was perfectly circular with $r = 1$ AU, then, since $r^2 = r^3$ in this case, such an orbit with $\beta = 3$ would be stable. However, it turns out that for this value of β , any small deviations of r away from unity (as from round-off errors in a calculation, or the effects of other planets in a real solar system) are rapidly amplified, and such an orbit is still unstable.

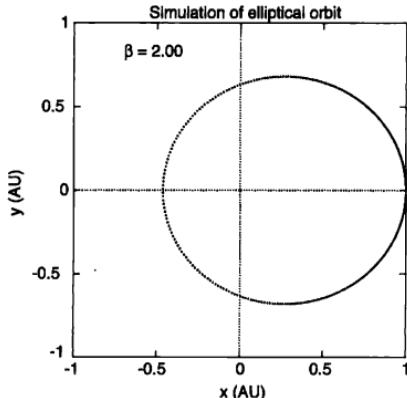


FIGURE 4.4: Elliptical orbit calculated for a force law with $\beta = 2$. The time step here, and in all of the calculations shown in this section, was 0.001 yr. We also used the same initial conditions in all of these simulations. The sun is at the origin.

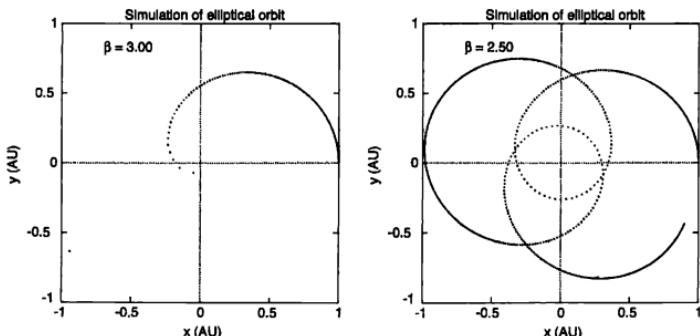


FIGURE 4.5: Elliptical orbits calculated for a force law (4.12) with $\beta = 3$ (left) and $\beta = 2.50$ (right).

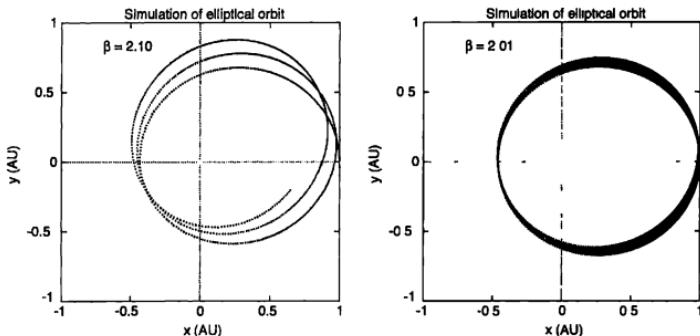


FIGURE 4.6: Elliptical orbits calculated for a force law (4.12) with $\beta = 2.1$ (left) and $\beta = 2.01$ (right)

even for $\beta = 2.01$ the ellipse still rotates significantly after only a few orbits. Indeed, comparing these results with those for $\beta = 2$ we see that the behavior is *extremely* sensitive to deviations from an inverse-square law (see Figure 4.6). This suggests that we might be able to use observations of planetary orbits to determine by what amount (if any) nature deviates from a perfect inverse-square law.

Such an experiment would not be as simple as our calculations here suggest. You may have noticed that in our discussions of Kepler's laws we have often referred to a two-body solar system consisting of a sun and just one planet. Of course, the real world is not that simple. There are nine planets to worry about, and any complete simulation would have to include the gravitational force of each planet on all of the others. We will study this problem in later sections. Here we only note that the forces from the other planets can also cause an elliptical orbit to rotate with time.¹¹ Hence, an experiment aimed at determining β through deviations from Kepler's laws would have to account carefully for the effects of the other planets.

EXERCISES

- 4.8. Verify Kepler's third law for elliptical orbits. Run the planetary motion program with initial conditions chosen to give orbits that are noncircular. Calculate T^2/a^3 and compare with the values given in Table 4.2.
- *4.9. In this section we saw that orbits are unstable for any value of β that is not precisely 2 in (4.12). A related question, which we did not address (until now), is *how* unstable an orbit might be. That is, how long will it take for an unstable orbit to become obvious. The answer to this question depends on the nature of the orbit. If the initial velocity is chosen so as to make the orbit precisely circular, then the value of β in (4.12) will make absolutely no difference. Of

¹¹This should not be surprising, since all of the planets orbit in the same direction (i.e., their angular momentum vectors are parallel), so the mutual attraction from gravity tends to drag the lighter planets along with the heavier ones.

course, in practice it is impossible to construct an orbit that is exactly circular, so the instabilities when $\beta \neq 2$ will always be apparent given enough time. Even so, orbits that start out as nearly circular will remain almost stable for a longer period than those that are highly elliptical. Investigate this by studying orbits with the same value of β (say, $\beta = 2.05$) and comparing the behavior with different values of the ellipticity of the orbit. You should find that the orientation of orbits that are more nearly circular will rotate more slowly than those that are highly elliptical.

4.3 PRECESSION OF THE PERIHELION OF MERCURY

In Section 4.2 we mentioned the possibility of using the solar system in an experiment to test the accuracy of the inverse-square law. In fact, such an experiment essentially has already been performed. Astronomy is a precise science. We mentioned earlier that Kepler inferred his laws from observations made by Tycho Brahe. What we didn't mention was that Brahe's observations were all made *by eye*; the telescope had not yet been invented! With the availability of telescopes and improvements in time-keeping, astronomical observations became even more precise.¹² This prompted impressive advances in celestial mechanics, the branch of physics concerned with the motion of planets and other objects in the universe. You might think that Kepler's laws provide all of the information a celestial mechanic would ever require, but this is far from the case. In a solar system with more than one planet there will be deviations from Kepler's laws. For our solar system these deviations are small, so Kepler's laws form an extremely useful starting point for discussing the planets we know. Nevertheless, there are deviations from these laws. These deviations come from a number of sources, including the effects of the planets on each other. It turns out that this is a problem for which very few exact results are known (even today). One of the jobs of a celestial mechanic is to calculate such effects.

We have noted several times that most of the planets have orbits that are very nearly circular. The planets whose orbits deviate the most from circular are Mercury and Pluto, and this leads to interesting consequences in both cases. For Mercury it was known by the early part of the 19th century that the orientation of the axes of the ellipse that describes its orbit rotate with time. This is known as the precession of the perihelion of Mercury (the perihelion is the point in an orbit when a planet is nearest the Sun). The magnitude of this precession is approximately 566 arcseconds per century (an arcsecond is $1/3600$ of a degree). That is, Mercury's perihelion makes one complete rotation every $\approx 230,000$ years. That such a small effect can be measured so accurately is a striking demonstration of the precision of astronomical measurements. This deviation from Kepler's law was of great interest to celestial mechanics, and by the middle of the 19th century it had been calculated that the gravitational forces of the other known planets¹³ lead to a precession of 523 arcseconds/century. Jupiter, which is by far the largest planet, is responsible for most of this. Perhaps most amazing is the fact that this calculation of the precession

¹²Astronomers have to measure not only *where* an object is located, but also *when* it is there.

¹³At that time only eight planets were known. Pluto was not discovered until 1930, but it is too small and too far away to have a significant effect on Mercury.

was performed "by hand" in a numerical computation involving log tables and the like, long before computers were available!

While the precision of both the experimental measurement and the theoretical calculation of the precession of Mercury's perihelion are very impressive, the fact remained that they did not agree. It was realized, of course, that this disagreement might be evidence for some interesting new physics, and various solutions to this puzzle were proposed. One of the most popular (for a while at least) was the suggestion that there might be another planet whose orbit was inside that of Mercury. However, such a planet was never found. It was also suggested that there might be a large amount of dust orbiting near the Sun, whose gravitational attraction was affecting Mercury, but again this proposal was never confirmed.

This troubling discrepancy was not explained until 1917 when Einstein developed the theory of general relativity. That theory deals with the geometry of space and views gravity in a much more complicated manner than our simple picture of Euclidean space and the inverse-square law (4.1). Nevertheless, general relativity leads to a similar prediction for the force due to gravity, provided that the two objects are not too close together (or equivalently, not too massive). However, if the separation between the two objects is made small enough, general relativity predicts deviations from the inverse-square law. It turns out that the Sun and Mercury are close enough for these deviations to just be significant, and Einstein showed that they precisely account for the previously unexplained 43 arcseconds of precession. This was one of the first triumphs of the theory of general relativity.

The precession due to general relativity can be calculated analytically, although to do so is fairly complicated. However, it is actually very straightforward to deal with this problem computationally (which is why we have devoted a section to it!). All we have to do is simulate the orbital motion using the force law predicted by general relativity and measure the rate of precession of the orbit, much as we did in the previous section. However, the precession rate is fairly small, so we have to design our simulation with that in mind (we don't want to have to compute the motion for 230,000 years).

The force law predicted by general relativity is

$$F_G \approx \frac{G M_S M_M}{r^2} \left(1 + \frac{\alpha}{r^2}\right), \quad (4.13)$$

where M_M is the mass of Mercury and¹⁴ $\alpha \approx 1.1 \times 10^{-8}$ AU². The force is thus an inverse-square law with a very small additional piece¹⁵ that is proportional to $1/r^4$. The effects of this tiny deviation from an inverse-square law are too small to measure easily in a computer simulation. The approach we take here is to calculate the rate of precession as a function of α , with values of α that are much larger than the actual value for Mercury. It will turn out that the rate of precession is given

¹⁴ α can be expressed in terms of the speed of light, the mass of the Sun, the eccentricity of the orbit, and other similar parameters (see Goldstein [1990], Chapter 11). This general relativistic effect is most noticeable for Mercury because it is the planet closest to the Sun.

¹⁵The term involving α in (4.13) is actually just the first of a series of such terms, which can be written as an expansion in powers of r^{-1} . For Mercury it is accurate to stop with the first correction term, as we have done here.

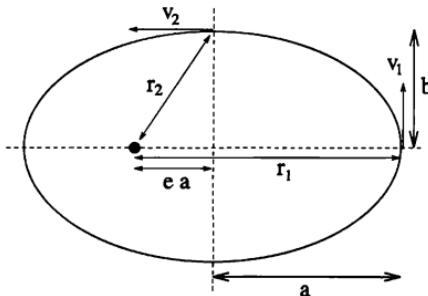


FIGURE 4.7: Definition of the parameters necessary for a calculation of the initial conditions needed for a simulation of Mercury's elliptical orbit. Points 1 and 2 are the places where the orbit crosses the x and y axes, respectively.

by $C\alpha$, where C is a constant that we will calculate. After we have obtained the value of C , we can then estimate the rate of precession for $\alpha = 1.1 \times 10^{-8}$ AU 2 , which is the case that we are really interested in.

To carry out this approach we first need to modify our planetary motion program to employ the force law (4.13), with α as an adjustable parameter. We will leave this job for the exercises. Next we need to determine the initial conditions required to obtain the orbit of Mercury. This is important, since the behavior depends on both the size of the orbit and also its eccentricity. The length of the semimajor axis (see Figure 4.7) for Mercury's orbit is $a = 0.39$ AU. The corresponding velocity, v_1 , which we require as an initial condition for the simulation, can be estimated in either of two ways. One is to calculate the motion for different trial values of v_1 and adjust its value until the eccentricity of the resulting orbit agrees with the known value, $e = 0.206$. A second approach is to make use of the conservation of both energy and angular momentum over the course of an orbit to calculate v_1 . This approach can be carried out with the help of Figure 4.7.

Conservation of total energy (kinetic plus potential of the planet, ignoring the kinetic energy of the Sun) implies that the energies at points 1 and 2 in Figure 4.7 are the same. Thus

$$-\frac{G M_S M_M}{r_1} + \frac{1}{2} M_M v_1^2 = -\frac{G M_S M_M}{r_2} + \frac{1}{2} M_M v_2^2. \quad (4.14)$$

The terms on the left-hand side of this equation are just the potential and kinetic energies at point 1 in Figure 4.7, and the terms on the right are the corresponding energies at point 2 (here we don't need to worry about the extremely small contribution of the general relativistic term in the potential). Since the force of gravity is a central force, it exerts no torque on the planet. Hence, the angular momentum

at point 1 is equal to that at point 2, which yields

$$r_1 v_1 = b v_2 . \quad (4.15)$$

We thus have two equations involving the unknowns v_1 and v_2 . After several lines of algebra (we'll leave that to you) we find

$$\begin{aligned} v_1 &= \sqrt{2 G M_S \left[\frac{b^2}{a^2(1+e)^2 - b^2} \right] \left[\frac{1}{\sqrt{e^2 a^2 + b^2}} - \frac{1}{a + ea} \right]} \\ &= \sqrt{\frac{G M_S (1-e)}{a (1+e)}}, \end{aligned} \quad (4.16)$$

where in the last step we have used the fact that $b = a\sqrt{1-e^2}$. Note that this is essentially identical to our earlier result for the parameters associated with a general elliptical orbit (4.11). Inserting the values of a and e given above yields $v_1 = 8.2$ AU/yr, and this is one of our initial conditions. The other is the distance from Mercury to the Sun, which is $r_1 = (1+e)a = 0.47$ AU.

Now that the initial conditions are known, we can simulate the motion of Mercury. The result obtained using the force law (4.13) with $\alpha = 0.01$ is shown in Figure 4.8. This value of α is *much* larger than the true value for Mercury, and it is seen that the ellipse precesses very noticeably in this case. The lines drawn from the Sun to the orbit show the orientation of the long axis of the ellipse. They are drawn from the origin (which is where the Sun is located) to the points on each orbit that are farthest from the Sun. These points are found by monitoring the distance of Mercury from the Sun and noting when its time derivative changes from positive to negative. These lines are very useful for our calculation, since the angles they make with the x axis are the amounts the orbit has precessed.

The next step is to calculate the angle of precession as a function of time for a particular value of α . A plot of this angle, which is the angle the radial lines in Figure 4.8 make with the x axis, as a function of time is shown in Figure 4.9. We see that the precession angle θ varies linearly with time. This means that the precession rate is a constant, which is what we have assumed implicitly all along.

The rate of precession, $d\theta/dt$, is the slope of the line in Figure 4.9. To obtain this slope we have two choices. One is to simply draw in a line by hand through the points and estimate its slope. A much better approach is to calculate the equation of the best-fit line. By this we mean the line that comes closest to passing through all of the points. If we had somehow been able to perform a perfect calculation of θ as a function of t , all of our points would lie exactly on a line. Alas, things are not this simple. Numerical errors (due to the nonzero value of the time step, etc.) will cause the points to deviate from such an ideal result. We would, therefore, like to somehow choose a line that comes as close as possible to all of the points (sort of a grand compromise). The conventional way to choose this line is with the method of least squares, which is discussed in Appendix E. This is a very powerful method of "fitting" straight lines (or, more generally, smooth curves) to noisy data; here the data are our results for θ . This least-squares line is determined in the following

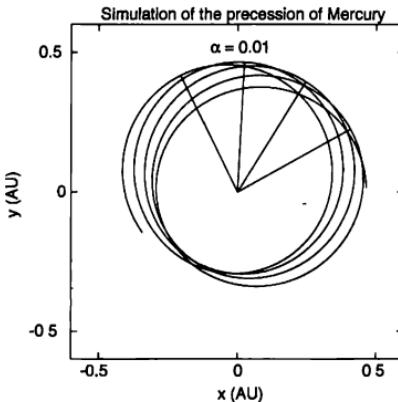


FIGURE 4.8: Simulated orbit for Mercury orbiting the Sun. The force law (4.13) was used, with $\alpha = 0.01$. The time step was 0.0001 yr. The program was stopped after several orbits. The solid lines emanating from the Sun (i.e., the origin) are drawn to the points on the orbit that are farthest from the Sun, so as to show the precession of the orientation of the orbit.

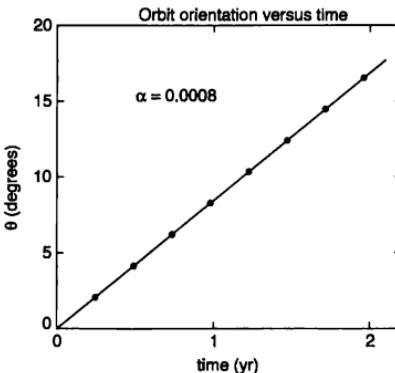


FIGURE 4.9: Precession of the axis of Mercury's orbit as a function of time, calculated for $\alpha = 0.0008$. The time step was 0.0001 yr. The solid line is a least-squares fit.

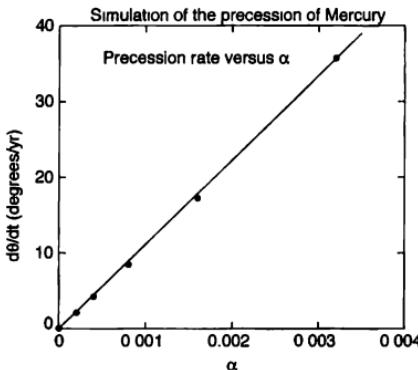


FIGURE 4.10: Precession rate of Mercury as a function of α . The solid line is a least-squares fit, which yielded a slope of 1.11×10^4 degrees per year per unit α .

way. Since, as we have already mentioned, the data do not all fall precisely on a single line, any line we imagine will deviate somewhat from at least some of the points. In our specific case, the values of θ defined by any line will in general deviate somewhat from the calculated values in Figure 4.9. With the method of least squares we choose a line so as to minimize the sum of the squares of these deviations. This approach is discussed at greater length in Appendix E, where we show how to calculate the slope and intercept of the least-squares line. It turns out that there is a unique solution for this problem. That is, there is one and only one least-squares line. Moreover, it is not hard to calculate.

Let us now return to the precession of the perihelion of Mercury. The solid line in Figure 4.9 is the least-squares line for these calculated values of $\theta(t)$. We see from Figure 4.9 that the least-squares line does indeed provide an excellent fit to the results for $\theta(t)$. The slope of this best-fit line then gives the precession rate for this particular value of α . We can now repeat the calculation with different values of α and obtain the best-fit precession rate in each case. The results of such a calculation are shown in Figure 4.10 where we plot the precession rate as a function of α . We see that the precession rate itself varies linearly with α . We can again use the method of least squares to describe these results, and the best-fit line is also shown in Figure 4.10. The slope of this line is 1.11×10^4 degrees per year per unit α . Now we are finally ready to finish off our calculation. Since we have found that the precession rate varies linearly with α and we have calculated the coefficient of proportionality, we can extrapolate to the case $\alpha = 1.1 \times 10^{-8}$ AU² predicted by the theory of general relativity. This yields a precession rate of $1.1 \times 10^{-8} \times 1.11 \times 10^4 \approx 1.2 \times 10^{-4}$ degrees/year, which is also equal to ≈ 43 arcseconds/century, in agreement with the experimental result mentioned at the beginning of this section.

Our study of the precession of the perihelion of Mercury has illustrated several useful techniques. One is the use of extrapolation to deal with situations in which the effects of interest are too small to conveniently estimate directly with a numerical approach. Of course, you must have some confidence in how the extrapolation should be made. In the present case we were able to show that a linear extrapolation as a function of α is appropriate. The second useful technique we have introduced is the method of least squares. We will use it again in later chapters.

EXERCISES

- 4.10.** Calculate the precession of the perihelion of Mercury, following the approach described in this section.
- *4.11.** Investigate how the precession of the perihelion of a planet's orbit due to general relativity varies as a function of the eccentricity of the orbit. Study the precession of different elliptical orbits with different eccentricities, but with the same value of the perihelion. Let the perihelion have the same value as for Mercury, so that you can compare it with the results shown in this section.

4.4 THE THREE-BODY PROBLEM AND THE EFFECT OF JUPITER ON EARTH

To this point all of our planetary simulations have involved two-body solar systems. It is now time to consider some of the things that can happen when there are three or more objects in the solar system. The problem of two objects interacting through the inverse-square law (4.1) can be solved exactly (as we have already mentioned), leading to Kepler's laws. However, if we add just one more planet to give what is known as the *three-body problem*, an analytic theory becomes *much* more difficult. In fact, there are very few exact results in this case, even though it has been studied extensively for several centuries. Indeed, the three-body, or more generally the *n*-body problem, is the problem of celestial mechanics.

In this section we consider one of the simplest three-body problems, the Sun and two planets, which we will take to be Earth and Jupiter. We know that without Jupiter, Earth's orbit is stable and unchanging with time.¹⁶ Our objective is to observe how much effect the gravitational force from Jupiter has on Earth's motion. We consider Jupiter since, at about 0.1% of the solar mass, it is by far the largest planet in the solar system.

To carry out this simulation we must modify our planetary-motion program to include two planets and the gravitational force between them. The magnitude of the force between Jupiter and Earth is given by our now familiar inverse-square law, with the Sun replaced by Jupiter

$$F_{EJ} = \frac{G M_J M_E}{r_{EJ}^2}, \quad (4.17)$$

where M_J is the mass of Jupiter and r_{EJ} the distance between Earth and Jupiter. We assume that the orbits of the two planets are coplanar, and we take this to be

¹⁶Ignoring general relativity, whose effect is much smaller for Earth than it is for Mercury.

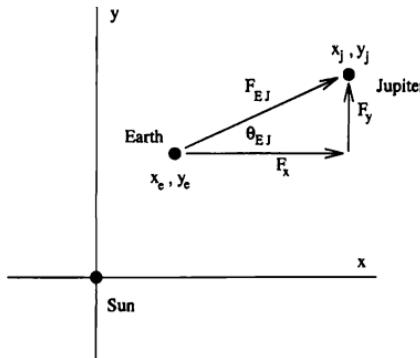


FIGURE 4.11: Components of the gravitational force due to Jupiter, located at x_j, y_j , with Earth at x_e, y_e . The Sun is at the origin.

the xy -plane. Writing $F_{E,J}$ in terms of components yields (see Figure 4.11)

$$F_{EJ,x} = - \frac{G M_J M_E}{r_{EJ}^2} \cos \theta_{EJ} = - \frac{G M_J M_E (x_e - x_j)}{r_{EJ}^3}, \quad (4.18)$$

for the x component of the force, with a corresponding result for the y component. Here x_e and x_j are the coordinates of Earth and Jupiter (the Sun remains at the origin), and θ_{EJ} is the angle defined in Figure 4.11. The total force on Earth in the x direction will be the sum of the forces of gravity from the Sun (4.3) and Jupiter (4.18), yielding the equation of motion for the x component of Earth's velocity, $v_{x,e}$

$$\frac{dv_{x,e}}{dt} = - \frac{G M_S x_e}{r^3} - \frac{G M_J (x_e - x_j)}{r_{EJ}^3}, \quad (4.19)$$

where r is again the distance from Earth to the Sun. We can convert this and the corresponding result for $v_{y,e}$, into difference equations, just as we did in (4.7). Since M_S is much greater than M_J or M_E , we may treat the Sun to be stationary at the origin and just calculate the positions of Jupiter and Earth. Thus, the only thing left to do is calculate GM_J in the appropriate units. Here it is simplest to use the result $GM_J = GM_S(M_J/M_S) = 4\pi^2(M_J/M_S)$, with M_J and M_S given in Table 4.1. A program to calculate the orbits of two planets in this approximation is a straightforward extension of the one we sketched for the two-body solar system previously in Example 4.1. We now need to update the positions and velocities of both planets at each step through the main loop. A sketch of such a calculation is given in Example 4.2.

EXAMPLE 4.2 Subroutine jupiter-earth for a two-planet solar system

- At each time step i , calculate the positions (x_e, y_e) (Earth) and (x_j, y_j) (Jupiter) as well as the velocities $(v_{e,x}, v_{e,y})$ and $(v_{j,x}, v_{j,y})$ for the time $i+1$ using the Euler-Cromer method.
 - ▷ Calculate the distances among Earth, Jupiter, and the Sun:

$$r_e(i) = \sqrt{x_e(i)^2 + y_e(i)^2},$$

$$r_j(i) = \sqrt{x_j(i)^2 + y_j(i)^2},$$

$$r_{EJ} = \sqrt{[x_e(i) - x_j(i)]^2 + [y_e(i) - y_j(i)]^2}.$$
 - ▷ Compute the new velocity of Earth

$$v_{e,x}(i+1) = v_{e,x}(i) - \frac{4\pi^2 x_e(i)}{r_e(i)^3} \Delta t - \frac{4\pi^2 (M_J/M_S)[x_e(i) - x_j(i)]}{r_{EJ}(i)^3} \Delta t,$$
 and similarly for the y -component, $v_{e,y}(i+1)$.
 - ▷ Compute the new velocity of Jupiter

$$v_{j,x}(i+1) = v_{j,x}(i) - \frac{4\pi^2 x_j(i)}{r_j(i)^3} \Delta t - \frac{4\pi^2 (M_E/M_S)[x_j(i) - x_e(i)]}{r_{EJ}(i)^3} \Delta t,$$
 and similarly for the y -component, $v_{j,y}(i+1)$.
 - ▷ Use the Euler-Cromer method to calculate the new positions of Earth and Jupiter:

$$x_e(i+1) = x_e(i) + v_{e,x}(i+1) \Delta t // y_e(i+1) = y_e(i) + v_{e,y}(i+1) \Delta t //$$
 and similarly for Jupiter.
 - ▷ Record or plot the new positions as they become available.
 - ▷ Repeat for desired number of time steps.
-

A note about program efficiency should also be made here. You may have noticed that the Euler-Cromer equations used above contain factors such as $4\pi^2 M_J/M_S$. The way we have written the code, these factors may be recomputed each time through the loop. However, we could have arranged to calculate them just once at the beginning of the program, thereby saving some time during each iteration. As we have noted in previous chapters, it is usually our policy to sacrifice speed for clarity. You might argue that with the proper use of comment statements the reduction of clarity can be made very small. While this is correct, we would argue in response that the increase in program speed is probably also very small. Indeed, most modern compilers and interpreters would probably recognize that our factors of $4\pi^2 M_J/M_S$ do not have to be recomputed each time and would store them accordingly. Hence, we would have both our clarity and efficiency. In any case, if you do insist on expending extra effort to make a program execute as fast as possible, we have two recommendations. First, don't spend much time at this until *after* the program is working. Second, use a profiling tool or other means to determine which part(s) of the program is limiting the overall speed. Chances are that only one routine or loop will need to be tuned, so you might as well concentrate your efforts on it.

Some results from this simulation are shown in Figure 4.12. Using parameters appropriate for Earth and Jupiter, we find that both of the planets follow stable

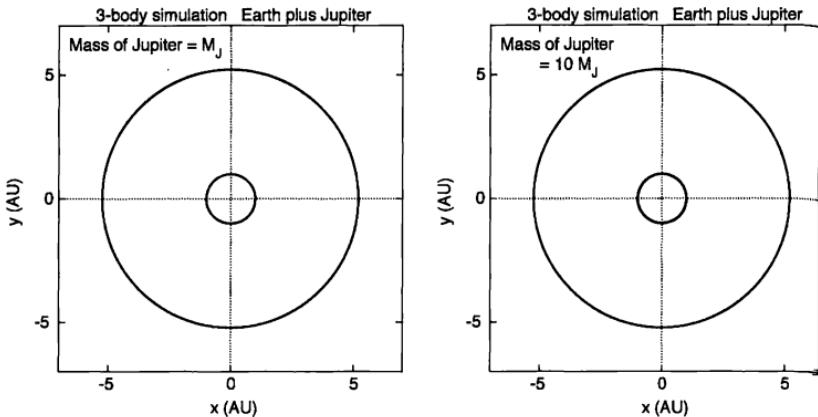


FIGURE 4.12: Simulation of a solar system with two planets, Earth and Jupiter. Left: Jupiter has its true mass; right: the mass of Jupiter has been set to 10 times its true mass.

circular orbits. Thus, Jupiter has a negligible effect on Earth (at least on this scale). This should not be terribly surprising; we know that Earth has been orbiting the Sun for several billion years, so the orbit must be fairly stable! We can also use our simulation to calculate what would happen if the mass of Jupiter were somehow increased. Giving Jupiter a mass of $10M_J$, that is, 10 times its actual value, has no discernible effect on Earth (at least on the scale of this figure).

It is tempting to see what would happen if the mass of Jupiter were increased to $100M_J$ or even to $1000M_J$, using the same program. However, since $1000M_J$ is about equal to the mass of the Sun, the perturbation by Jupiter on the Sun would then be significant and would have to be taken into account. If we ignore this detail(!), and simply use $1000M_J$ as the mass of the second planet in routine `jupiter-earth`, we find the interesting trajectory shown on the left in Fig. 4.13. We see that Earth's orbit becomes completely unstable, as it is eventually ejected from the solar system! To investigate this problem more carefully, we need to perform a true three-body calculation, in which the motion of *all three* bodies – the Sun, Jupiter, and Earth – are computed. The results of such a simulation are very sensitive to the initial conditions, and often quite dramatic. For example, it is possible for Earth to switch back and forth between approximate orbits centered on Jupiter and the Sun. An example of such a true three-body simulation is shown on the right side of Figure 4.13. We will let you explore this problem further in the exercises.

The conclusion from this simulation is that Jupiter is (fortunately) too small to have a major influence on Earth. However, Jupiter is much closer to Mars,

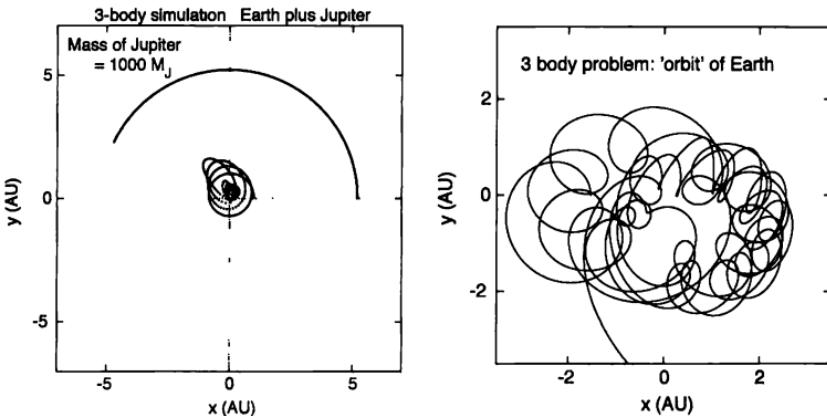


FIGURE 4.13: Simulation of a solar system with two planets, Earth and Jupiter. Left: the mass of Jupiter has been set to 1000 times its true mass, and we have used the routine `jupiter-earth`, which does *not* take into account the motion of the Sun. Here we stopped the simulation before Jupiter had completed even half an orbit, as the motion of Earth was unstable. Right: Typical results for a true 3-body simulation, in which the motions of Earth, Jupiter, and the Sun were all computed. Here we show only the motion of the Earth. The origin is now the center of mass of the 3-body system.

so there will be a larger effect in that case. We will leave it to the exercises to investigate this issue. In the next section we will consider the effect of Jupiter on the asteroids that orbit the Sun in the region between Mars and Jupiter.

EXERCISES

- 4.12.** Investigate the effect of Jupiter on Mars
- *4.13.** Explore the orbits of a planet in a double-star system Write a program that computes the motion of both stars along with that of the planet, including the gravitational forces of both stars on the planet and on each other. Explore the nature of the planetary orbits, and try to discover stable, repeating ones. *Hint:* First consider the case of two stars of equal mass, and pay special attention to planetary orbits that are especially symmetric with respect to the orbits of the stars.
- 4.14.** Simulate the orbits of Earth and Moon in the solar system by writing a program that accurately tracks the motions of both as they move about the Sun Be careful about (1) the different time scales present in this problem, and (2) the correct initial velocities (i.e., set the initial velocity of Moon taking into account the motion of Earth around which it orbits)
- *4.15.** In our discussion of the precession of the perihelion of Mercury we mentioned that the other planets cause most of the observed precession. As the largest planet, Jupiter is responsible for most of this. Calculate the precession of the perihelion of Mercury due to Jupiter. We suggest that you perform the calculation by giving Jupiter a mass that is much larger than its true value, and then extrapolate to obtain the final result.
- *4.16.** Carry out a true three-body simulation in which the motions of Earth, Jupiter, and the Sun are all calculated Since all three bodies are now in motion, it is useful to take the center of mass of the three-body system as the origin, rather than the position of Sun. We also suggest that you give Sun an initial velocity which makes the total momentum of the system exactly zero (so that the center of mass will remain fixed). Study the motion of Earth with different initial conditions. Also, try increasing the mass of Jupiter to 10, 100, and 1000 times its true mass.

4.5 RESONANCES IN THE SOLAR SYSTEM: KIRKWOOD GAPS AND PLANETARY RINGS

Table 4.1 gives the distances between the nine planets and the Sun. It is interesting to note that these distances grow in a seemingly regular manner, with the spacing between adjacent planets increasing as we go outward from the Sun. In fact, this pattern can be described by what is known as the Titius-Bode formula (see Peterson [1993]). According to that formula, the distances from the planets to the Sun are closely related to the sequence of integers

$$0, 3, 6, 12, 24, \dots \tag{4.20}$$

which is obtained by starting with the integers 0 and 3 and letting each succeeding term be just twice the term before. To derive the distance from a planet to the Sun, the corresponding term in this sequence is added to 4 and the result divided by 10 (in AU). In this formula, the first planet is Mercury, the second Venus, etc.

TABLE 4.3: Comparison of the actual semimajor axes of the planetary orbits, a , with the predictions of the Titus-Bode formula.

Planet	a (actual) (AU)	Titus-Bode (AU)
Mercury	0.39	0.40
Venus	0.72	0.70
Earth	1.00	1.00
Mars	1.52	1.60
???	—	2.80
Jupiter	5.20	5.20
Saturn	9.54	10.00
Uranus	19.19	19.60
Neptune	30.06	38.80
Pluto	39.53	77.20

The values calculated from the Titus-Bode formula are given in Table 4.3, where we also list the actual planetary distances for comparison.

The agreement is quite respectable for the planets Mercury through Uranus, although we might well be skeptical that the Titus-Bode formula simply describes a trend, but doesn't contain any real physics. On the other hand, it is interesting to note that Uranus was discovered *after* the Titus-Bode formula was proposed. The predictions for Neptune and Pluto are not as good, but nothing is perfect. Perhaps the most striking result in Table 4.3 is that a planet is missing. We don't mean this literally, of course; the point is that the formula predicts a planet between Mars and Jupiter, but none is known in that region. Now, if you don't believe in the Titus-Bode formula, you would not find this to be disturbing. However, more than 200 years ago, when the formula was first proposed, some astronomers did take it seriously. It was, therefore, the source of some excitement when, in about 1800, a "planet" with an orbital radius of approximately of 2.8 AU was discovered. Unfortunately, many other objects with about the same orbital radii were subsequently discovered. It was eventually realized that these orbiting objects are much smaller than planets, and they were given the name *asteroids*.

As more and more asteroids were discovered, an interesting pattern in *their* orbital radii was found. In the mid-1800s, the astronomer Daniel Kirkwood plotted the number of asteroids as a function of their distance from the Sun¹⁷ and found a result like that sketched in Figure 4.14. There are many gaps in this distribution plot. That is, for some values of the orbital radius there are essentially no asteroids. These are now known as the Kirkwood gaps.

Kirkwood also showed that the gaps are associated with Jupiter. He did this by noting that a (hypothetical) asteroid that has an orbital radius placing it in one of the gaps, would be in *resonance* with Jupiter. For example, the gap at

¹⁷More precisely, he plotted the probability density for finding an asteroid as a function of the orbital radius.

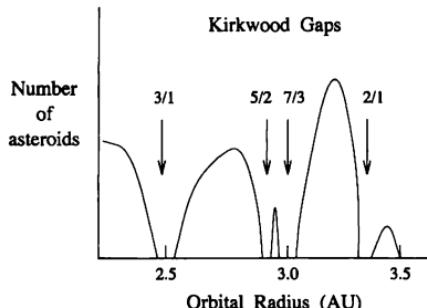


FIGURE 4.14: Schematic plot of the number of asteroids as a function of their distance from the Sun. The distances at which there are no asteroids are the Kirkwood gaps. For comparison, the orbital radii of Mars and Jupiter are 1.52 and 5.20 AU, respectively.

approximately 3.3 AU corresponds to an orbital period¹⁸ that is one half that of Jupiter's period. This is known as the 2/1 gap, since an asteroid placed there would complete two orbits every time Jupiter completes one. Similarly, there are Kirkwood gaps corresponding to 3/1, 5/2, and 7/3 resonances, and others as well.

The phenomenon of resonance is probably familiar to you, as it is found in many mechanical and electrical systems. For example, a lightly damped pendulum exhibits a resonance at its natural frequency of oscillation. If you were to excite a pendulum at this frequency, its amplitude of oscillation would become much larger than it would if another frequency was used. The same behavior would be found if an asteroid were somehow placed in one of the Kirkwood gaps. We have already discussed the gravitational force exerted by Jupiter on the other objects in the solar system. This force will be largest when the asteroid is closest to Jupiter and will tend to perturb the orbit of the asteroid. In most cases the asteroid will be in a completely different part of its orbit when Jupiter makes its closest approach, and the perturbations due to Jupiter will tend to (at least roughly) cancel out. However, if the asteroid is in resonance with Jupiter, it will be at the same point (or points) in its orbit each time Jupiter comes closest. For example, an asteroid in the 2/1 gap will always be at one of two points on its orbit when Jupiter makes its closest approach, and these points will be at opposite ends of the orbit. As a result, the effects of Jupiter during each approach will accumulate and eventually lead to a *large* perturbation of the orbit.

This resonant effect can be studied with our planetary-motion program. We have already discussed routine `jupiter-earth`, which allows the simulation of two planets. The same procedure can be used to simulate any number of planets. We will leave the programming for the exercises, but do have a few suggestions.

¹⁸The orbital period can be calculated using Kepler's third law.

TABLE 4.4: Initial positions and velocities used for three hypothetical asteroids in the vicinity of the 2/1 Kirkwood gap. The initial conditions for Jupiter are also given.

Object	Radius (AU)	Velocity (AU/yr)
Asteroid number 1	3.000	3.628
Asteroid number 2	3.276	3.471
Asteroid number 3	3.700	3.267
Jupiter	5 200	2.755

It is simplest to use arrays to store the positions and velocities of the planets and asteroids. The first element in each array would then contain the position or velocity of the first object, etc., for whatever number of objects you want to place in orbit. Also, since Jupiter is by far the largest orbiting object in this problem and the mass of an asteroid is very small by comparison, it is not necessary to include the gravitational force of each asteroid when computing Jupiter's motion (you can also ignore the interactions between asteroids). This approximation amounts to what is known as the *restricted n-body problem* and will make your program run faster without significantly affecting the physics of the problem.

To study the resonant behavior of asteroids near the 2/1 Kirkwood gap we have simulated the motion of three hypothetical asteroids, all placed in circular orbits. The initial positions and velocities of these asteroids, along with those of Jupiter, are given in Table 4.4.

The parameters for asteroid number 2 have been chosen to place it in the 2/1 gap, while the other asteroids are adjacent to this gap. The reason for including them in the simulation will become clear as we discuss the results, which are shown in Figure 4.15. The simulation was run for approximately 10 orbits of Jupiter (which is not shown in the figure), and only a portion of the calculated positions (about every 10th one) are shown. We see that the orbits of all three asteroids are somewhat "smeared out" in our plot. This means that the force of Jupiter has caused all three orbits to be elliptical, and the axes of all three ellipses precess with time. The interesting result is that the asteroid in the 2/1 gap, the one that is in resonance with Jupiter (shown on the left in Figure 4.15), is the one affected *most* strongly by Jupiter. This must be due to the resonance since the outermost asteroid (one of those shown on the right), which experiences the largest force from Jupiter, is affected *less* than the asteroid in the gap region.

Now that we have confirmed the importance of the resonance, it is interesting to consider how an asteroid which is initially in the gap region might be ejected from this orbit. One possibility is that the disturbance of the orbit seen in Figure 4.15 would simply grow with time, until the orbit becomes so elliptical that the asteroid has a close encounter with Mars, or some other planet. However, extensive numerical simulations carried out in the last 30 years or so give a somewhat different picture (more on this can be found in the references at the end of the chapter). These calculations show that an asteroid that begins with an orbit in a Kirkwood gap will follow an orbit that, though somewhat elliptical because of

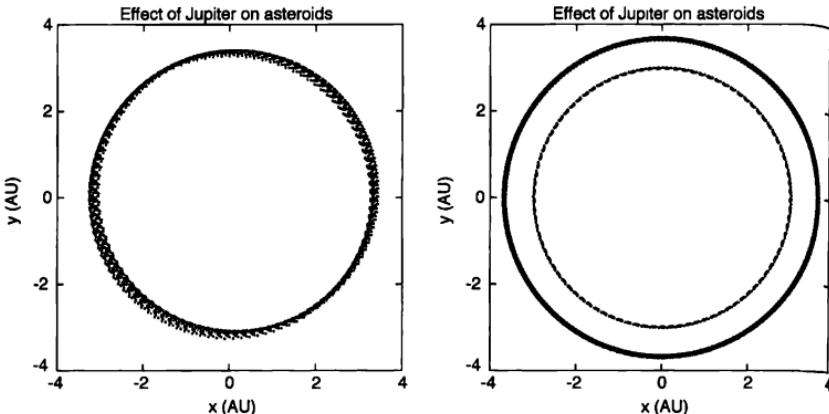


FIGURE 4.15: Simulation of the effect of Jupiter on three asteroids, showing only the asteroids (Jupiter's orbital radius is 5.2 AU). Left: an asteroid (number 2 in Table 4.4), which was chosen to be in the 2/1 Kirkwood gap, and thus was in resonance with Jupiter. Right: two asteroids (number 1 and number 3 in Table 4.4), chosen to be closer to and farther from Jupiter than the asteroid in the gap. The orbit of the asteroid in the gap is much more "smeared out," that is, much more affected by Jupiter, than are the other asteroids. The time step was 0.005 yr. Here we have not connected the calculated positions, which is why the trajectories appear as discrete points.

the effect of Jupiter, remains in resonance with Jupiter for a long period of time. Then, seemingly without warning, the orbit changes drastically over a relatively short period, taking the asteroid near Mars or Earth, which then pulls it far from its initial orbit.

Such strange and erratic behavior is actually very similar to what we studied in Chapter 3 (recall the intermittency route to chaos), and the motion of these gap asteroids is believed to be chaotic. This is a very intriguing result for the following reason. We have already seen that the two-body problem is governed by Kepler's laws. This *always* implies a perfectly regular, predictable orbital motion. However, the addition of one more object can lead to motion that is completely unpredictable, that is, chaotic. It is believed that there are a number of other instances of chaotic motion in our solar system. One of these will be considered in the next section, while others are discussed in the references.

The physics associated with the Kirkwood gaps is not limited to asteroids. You have probably seen pictures of the rings around Saturn. These rings are made up of a large collection of particles that orbit the planet. These particles have a distribution of orbital radii and hence effectively form a disk. However, for certain orbital radii there are essentially *no* orbiting particles. These gaps are analogous to the Kirkwood gaps in the asteroid distribution. In accord with that analogy, each

gap in the ring system is associated with one of Saturn's moons. If a particle could somehow be placed in one of these gaps, it would be in resonance with a moon and would exhibit the behavior we have observed in our asteroid simulations.

EXERCISES

- 4.17. Simulate the motion of asteroids near the one of the Kirkwood gaps. The 2/1 and 7/3 gaps are good choices.
- *4.18. Investigate how close an asteroid must be to the precise gap resonance in order to be effectively "in" the gap. That is, study the effective width of the resonances in Fig. 4.14. Do this by performing the calculation of Fig. 4.15 for several asteroids whose (circular) orbits are near that of asteroid number 2 in Table 4.4. Calculate how much the orbit is "smeared out" as a function of the initial orbital radius.
Note: You will need to develop a quantitative way to measure this "smearing."

4.6 CHAOTIC TUMBLING OF HYPERION

We mentioned in the previous section that the motion of asteroids located near the Kirkwood gaps is believed to be chaotic. Unfortunately, we were not able to demonstrate this explicitly; to do so would require a simulation of the motion for several million years or longer. Such a simulation is very involved (don't try this at home) and best left for a professional. It turns out that there is also good evidence, based largely on computer simulations, that the motion of Pluto is chaotic. This is another difficult simulation that we will not attempt here (nor will we leave it for the exercises). However, there is one case of chaos in our solar system that is accessible to a fairly simple simulation. It involves the motion of Hyperion, one of Saturn's smaller moons. Hyperion is extremely small in comparison to other moons. Its mass is about 3×10^{-8} of Saturn's mass; this is much smaller than our Moon, whose mass is about 1% of the Earth's mass.

Moons are very interesting objects. Most planets have them, with the larger planets such as Jupiter and Saturn having a great many each. You are probably most familiar with Earth's moon and know that it spins about its axis in such a way that it always keeps one particular side facing Earth. That is, its spin is synchronized with its orbital motion about Earth. It is believed that this synchronization came about in the following way. When first formed, a typical moon (such as Earth's moon) probably spun at a rate larger than one revolution per orbit. However, as time passed, a slight asymmetry in the Moon's mass distribution coupled with the fact that the gravitational force of Earth falls off with distance, led to a very small "stretching" and "compressing" of the Moon during each orbit. This dissipated a small amount of energy (essentially from friction), which caused the spin angular velocity to decrease. Eventually, this angular velocity matched the orbital motion, at which point an effect known as spin-orbit resonance came into play and added enough energy to the Moon's spinning motion during each orbit to counter the frictional losses. The spin and orbital motion were then synchronized, and this situation has persisted to the present time. It turns out that all of the moons in the solar system *except one* exhibit such synchronization. The exception is Hyperion.

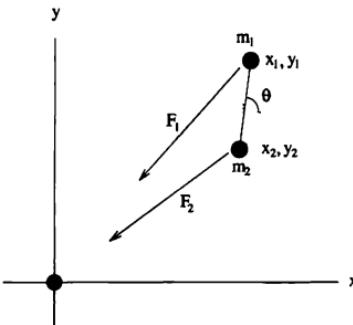


FIGURE 4.16: Simplified model of Hyperion in orbit around Saturn. Our “moon” consists of two particles, m_1 and m_2 , connected by a massless rigid rod. The dotted lines are parallel to the x and y directions, and intersect the rod at the center of mass. θ is the angle that the rod makes with x . Saturn is at the origin.

The reason for Hyperion’s different behavior is its very unusual shape, together with its highly elliptical orbit. While other moons (and planets) are approximately spherical, Hyperion is shaped more like an egg with dimensions of about $205 \times 130 \times 110$ km³. Hence, it is roughly like an orbiting dumbbell, and anyone who has ever thrown a dumbbell, or something with a similar shape, knows that they can spin in a very erratic manner. Indeed, this is precisely what Hyperion seems to do. Not only is Hyperion’s spin out of synchrony with its orbital motion, but careful astronomical observations suggest that Hyperion tumbles chaotically as it orbits Saturn.

To simulate the motion of Hyperion we will first make a few simplifying assumptions. Our goal will *not* be to perform a realistic simulation of Hyperion (you can read about such work in the references). Rather, our objective is simply to show that the motion of such an irregularly shaped moon can be chaotic. With that goal in mind we consider the model shown in Figure 4.16. We have two particles, m_1 and m_2 , connected by a massless rod in orbit around a massive object located at the origin. To simulate this we have to extend our original planetary-motion program to include the rotation of the object about the axis perpendicular to the plane of the figure.

We let θ be the angle that the rod makes with the x axis and define the associated angular velocity $\omega = d\theta/dt$. Our program must calculate the variation of θ and ω , along with the position of the center of mass of the object, x_c and y_c . The motion of the center of mass can be calculated in precisely the manner we used in our first planetary motion program, as the equations of motion for x_c and y_c are unchanged. To calculate the behavior of θ and ω we will use the Euler-Cromer method. We, therefore, need the equation of motion for ω , which can be obtained from the torque on the moon.

There are two forces acting on each of the masses, the force of gravity from Saturn and the force from the rod. Since we are interested in the motion (and thus the torque) about the center of mass, the force from the rod does not contribute. The gravitational force on m_1 can be written as

$$\vec{F}_1 = - \frac{G M_{\text{Sat}} m_1}{r_1^3} (x_1 \hat{i} + y_1 \hat{j}), \quad (4.21)$$

where M_{Sat} is the mass of Saturn, r_1 is the distance from Saturn to m_1 , and \hat{i} and \hat{j} are unit vectors in the x and y directions. The coordinates of the center of mass are (x_c, y_c) , so that $(x_1 - x_c)\hat{i} + (y_1 - y_c)\hat{j}$ is the vector from the center of mass to m_1 . The torque on m_1 is then

$$\vec{\tau}_1 = [(x_1 - x_c)\hat{i} + (y_1 - y_c)\hat{j}] \times \vec{F}_1, \quad (4.22)$$

with a similar expression for $\vec{\tau}_2$. The total torque on the moon is just $\vec{\tau}_1 + \vec{\tau}_2$, and this is related to the time derivative of ω by¹⁹

$$\frac{d\omega}{dt} = \frac{\vec{\tau}_1 + \vec{\tau}_2}{I}, \quad (4.23)$$

where $I = m_1 |r_1|^2 + m_2 |r_2|^2$ is the moment of inertia. Putting this all together yields, after some algebra,²⁰

$$\frac{d\omega}{dt} \approx - \frac{3GM_{\text{Sat}}}{r_c^5} (x_c \sin \theta - y_c \cos \theta) (x_c \cos \theta + y_c \sin \theta), \quad (4.24)$$

where r_c is the distance from the center of mass to Saturn.

To construct a program for this problem we must add a few lines to our planetary motion program so as to calculate ω according to (4.24), and θ using $d\theta/dt = \omega$. We will leave this to the exercises. Some results for θ and ω as functions of time are shown in Figures 4.17 and 4.18. Since we are not trying to do a quantitative simulation of the actual motion of Hyperion (our simple dumbbell model is too crude for that), we have chosen some slightly more convenient orbital parameters. After all, our goal is only to get a feeling for the kind of behavior that is possible. A truly quantitative simulation is, as they say, beyond the scope of this book.

Figure 4.17 shows the behavior when the orbit is circular. The abrupt vertical jumps in θ are simply due to the program “resetting” θ to keep it in the range $-\pi$ to π (as we did in our pendulum simulations). The behavior in Figure 4.17 is seen to be regular and repeatable; this is especially clear from the results for ω . We thus conclude that the motion is not chaotic when the orbit is circular. However, the results obtained for an elliptical orbit, Figure 4.18, are very different. The behavior seen in this case is very complicated and erratic, and certainly appears to be chaotic.

¹⁹The vector signs are not really needed here since all of the torques and angular momenta in this problem are along the z axis, that is, perpendicular to the plane of Figure 4.16.

²⁰The details of which we will leave to the inquisitive reader. Equation (4.24) assumes that the length of the rod connecting the two masses is much smaller than the Saturn-Hyperion distance r_c .

EXERCISES

- 4.19.** Study the behavior of our model for Hyperion for different initial conditions. Estimate the Lyapunov exponent from calculations of $\Delta\theta$, such as those shown in Figure 4.19. Examine how this exponent varies as a function of the eccentricity of the orbit.
- 4.20.** Our results for the divergence of the two trajectories $\theta_1(t)$ and $\theta_2(t)$ in the chaotic regime, shown on the right in Figure 4.19, are complicated by the way we dealt with the angle θ . In Figure 4.19 we followed the practice employed in Chapter 3 and restricted θ to the range $-\pi$ to $+\pi$, since angles outside this range are equivalent to angles within it. However, when during the course of a calculation the angle passes out of this range and is then “reset” (by adding or subtracting 2π), this shows up in the results for $\Delta\theta$ as a discontinuous (and distracting) jump. Repeat the calculation of $\Delta\theta$ as in Figure 4.19, but do *not* restrict the value of θ . This should remove the large ($\Delta\theta \sim 2\pi$) jumps in $\Delta\theta$ in Figure 4.19, but the smaller and more frequent dips will remain. What is the origin of these dips? *Hint:* Consider the behavior of a pendulum near one of its turning points.

REFERENCES

- [1] H. Goldstein, 1980, *Classical Mechanics*, Addison-Wesley, Reading. Gives the form for the perturbation of the gravitational inverse-square law due to general relativity.
- [2] J. J. Klavetter, “Rotation of Hyperion. I. Observations.” *Astronomical Journal* **97**, 570 (1989); “Rotation of Hyperion. II. Dynamics.” *Astronomical Journal* **98**, 1855 (1989). Research articles that describe a detailed analysis of the motion of Hyperion.
- [3] J. B. Marion and S. T. Thornton, 1995, *Classical Dynamics of Particles and Systems*, Brooks-Cole, Belmont. Detailed discussions of central force systems such as the planetary motion in the solar system are given in Chapter 8.
- [4] I. Peterson, 1993, *Newton’s Clock: Chaos in the Solar System*, Freeman, New York. A very entertaining account of the history of astronomical studies of the solar system.
- [5] J. Wisdom, “Chaotic Dynamics in the Solar System.” *Icarus* **72**, 241 (1987). A readable, though advanced, discussion of simulations of several cases of chaotic motion in the solar system.

Potentials and Fields

5.1 ELECTRIC POTENTIALS AND FIELDS: LAPLACE'S EQUATION

In regions of space that do not contain any electric charges, the electric potential obeys Laplace's equation

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = 0. \quad (5.1)$$

From a numerical perspective the situation here is a bit different from anything we have encountered so far in this book. All of our problems to this point have involved differential equations for which several initial conditions were given, and we were able to use the Euler method or something like it to calculate the behavior for later times. However, with Laplace's equation we are generally given some boundary conditions for V , which specify its value on a surface in x - y - z space. Alternatively, the boundary conditions might be given in terms of the electric field, which is proportional to the gradient of V . In either case our problem is to find the function $V(x, y, z)$ that satisfies both Laplace's equation and the specified boundary conditions. Most importantly, satisfying the boundary conditions will not be as easy as with the ordinary differential equations encountered in previous chapters.

While most ordinary differential equations can be dealt with using methods such as the Euler or Runge-Kutta approaches, there are no comparable "general purpose" algorithms for partial differential equations (PDEs). Instead there are many different algorithms, each designed for a particular class of PDEs. We will explore one such method in this and the following section, while others will be encountered in later chapters. The approach we will use is the relaxation method; it is convenient for dealing with the class of PDEs known as elliptic equations, of which Laplace's equation is one example.

As usual we discretize the independent variables, in this case x , y , and z . Points in our space are then specified by integers i , j , and k , with $x = i\Delta x$, $y = j\Delta y$, $z = k\Delta z$. Our goal is to determine the potential $V(i, j, k) \equiv V(i\Delta x, j\Delta y, k\Delta z)$ on this lattice of points.¹ The first step in reaching this goal is to rewrite (5.1) as a difference equation. We already know how to write a first derivative in finite-difference form. For example, at the point (i, j, k) the derivative with respect to x may be written as

$$\frac{\partial V}{\partial x} \approx \frac{V(i+1, j, k) - V(i, j, k)}{\Delta x}, \quad (5.2)$$

¹Here we depart with what has been our usual convention (so far) and write $V(i, j, k)$ instead of $V_{i,j,k}$. The latter is a bit more cumbersome when there are two or more indices. In the remainder of this book we will use both of these notations (though not in the same problem!), with the choice being determined by convenience and (we hope) clarity.

where the \approx sign is there to remind you that this is only an approximation. This is not the only conceivable way to express the first derivative in finite-difference form. We could just as easily have written

$$\frac{\partial V}{\partial x} \approx \frac{V(i, j, k) - V(i-1, j, k)}{\Delta x}, \quad (5.3)$$

or

$$\frac{\partial V}{\partial x} \approx \frac{V(i+1, j, k) - V(i-1, j, k)}{2 \Delta x}. \quad (5.4)$$

The “best” choice depends on the particular problem at hand.

Since here we are dealing with Laplace’s equation (5.1), we are really after an expression for the second derivatives, so it is worth noting that (5.2) is effectively centered about the “imaginary” location $i + \frac{1}{2}$, while (5.3) is centered at $i - \frac{1}{2}$. Thus, it is natural to write the second partial derivative as

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{1}{\Delta x} \left[\frac{\partial V}{\partial x}(i + \frac{1}{2}) - \frac{\partial V}{\partial x}(i - \frac{1}{2}) \right], \quad (5.5)$$

where the notation here is intended to indicate that the derivatives are to be evaluated at the locations $(i \pm \frac{1}{2})$. This leads to

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{1}{\Delta x} \left[\frac{V(i+1, j, k) - V(i, j, k)}{\Delta x} - \frac{V(i, j, k) - V(i-1, j, k)}{\Delta x} \right], \quad (5.6)$$

and a little rearranging yields

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V(i+1, j, k) + V(i-1, j, k) - 2V(i, j, k)}{(\Delta x)^2}. \quad (5.7)$$

This expression is nicely symmetric in the way it treats $V(i+1, j, k)$ and $V(i-1, j, k)$, which will turn out to reduce the overall errors in our computations. The results for the other second partial derivatives have similar forms. Inserting them all into Laplace’s equation and solving for $V(i, j, k)$ we find

$$V(i, j, k) = \frac{1}{6} [V(i+1, j, k) + V(i-1, j, k) + V(i, j+1, k) \\ + V(i, j-1, k) + V(i, j, k+1) + V(i, j, k-1)], \quad (5.8)$$

where we have assumed² that the step sizes along x , y , and z are all the same ($\Delta x = \Delta y = \Delta z$). In words, (5.8) simply says that the value of the potential at any point is the *average* of V at all of the neighboring points. The solution for $V(i, j, k)$ is the function that manages to satisfy this condition at *all* points simultaneously.

We now require a numerical strategy for determining this function, assuming only that V is known at the boundaries. We can’t just start at one of the boundaries and work our way into and across the system, since according to (5.8) we need to

²This assumption is not necessary, but does make the form of (5.8) a little simpler.

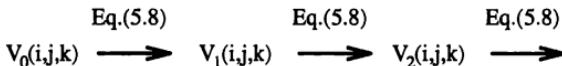


FIGURE 5.1: Schematic flowchart for the relaxation algorithm.

know V at *all* of the neighbors in order to calculate its value at any particular point. The approach we take is to begin with some initial guess for the solution; call it $V_0(i,j,k)$. In general, unless we are extremely clever, the guess we make will not satisfy (5.8) everywhere. To obtain an improved guess, we use (5.8) to calculate new values of V , using V_0 on the right-hand side. This is illustrated pictorially in Figure 12.47. The guess V_0 , together with (5.8) yields, a new and we hope improved guess, $V_1(i,j,k)$. We then repeat the procedure with V_1 to obtain an even better guess, V_2 , etc. This iterative process is continued until our result satisfies some convergence criteria, which we will discuss shortly. The main point is that we can iterate with (5.8) to obtain a better and better solution. This general approach is called the *relaxation method*, and is a useful way to deal with several important classes of partial differential equations. There are different ways to implement the relaxation method, and some are much better than others with respect to speed of convergence, etc. The particular algorithm we have just described is known as the Jacobi method, and we will begin by applying it to several problems, considering some other, more efficient relaxation algorithms a bit later. As it turns out, even the simplest relaxation algorithm, the Jacobi method outlined above, works reasonably well provided that the initial guess, V_0 , is not too bad. In fact, as we will see shortly, even rather simple choices for V_0 are usually adequate.

Before we consider some specific examples, it is useful to consider the relaxation algorithm in a more general context. In essence, we have converted a time *independent* partial differential equation (5.1), into a time *dependent* equation whose solutions evolve in time according to the relaxation procedure, as contained schematically in Fig. 12.47. We also require that the long-time, steady-state solution of the time dependent partial differential equation should be the solution of our time independent equation. This conversion from a time independent to a time dependent problem can be done in a number of ways, but a particular clear approach is to do so via a *diffusion equation*. With this in mind, we consider a function $\bar{V}(x,y,z,t)$, that obeys

$$\frac{\partial \bar{V}(x,y,z,t)}{\partial t} = D \left(\frac{\partial^2 \bar{V}}{\partial x^2} + \frac{\partial^2 \bar{V}}{\partial y^2} + \frac{\partial^2 \bar{V}}{\partial z^2} \right). \quad (5.9)$$

This equation describes diffusion,³ a dynamic process in which a particle moves about randomly, with $\bar{V}(x,y,z,t)$ being the probability density of finding the particle at (x,y,z) at time t . The connection between diffusion and Laplace's equation (5.1) works as follows. The process of diffusion has an equilibrium limit; that is,

³You may also recognize this as the *heat* equation, which describes the spreading of heat.

starting from almost any initial distribution (and with appropriate boundary conditions), $\bar{V}(x, y, z, 0)$ will evolve in time to a time-independent, steady-state solution, $\bar{V}(x, y, z, t \rightarrow \infty)$. Since it describes a time-independent system, the time derivative of $\bar{V}(x, y, z, t \rightarrow \infty)$ must then vanish, and $\bar{V}(x, y, z, t \rightarrow \infty)$ will then also satisfy Laplace's equation. What may come as a surprise to you is that the Jacobi method effectively evolves the system in time according to the diffusion equation, so as to reach this time-independent state.⁴

Let us now use the relaxation method to attack a few specific problems. We begin with the problem of calculating the electric potential inside the square box in Figure 5.2. For boundary conditions⁵ we assume that the face of the box at $x = -1$ is held at $V = -1$, while $V = +1$ on the face at $x = +1$. We will assume that the sides of the box at $y = \pm 1$ are nonconducting, so V on those faces will vary as we move from $x = -1$ to $x = +1$. From the symmetry it is natural to suppose that the potential on these surfaces of the box will vary linearly with position⁶ as we move from $x = -1$ to $x = +1$. We assume further that the box is infinite in extent along $\pm z$, so $V(i, j, k)$ is independent of k . We thus have only a two-dimensional problem. Our goal is to find the potential function $V(i, j)$ that satisfies

$$V(i, j) = \frac{1}{4} [V(i+1, j, k) + V(i-1, j, k) + V(i, j+1, k) + V(i, j-1, k)] . \quad (5.10)$$

Next we consider how to implement the Jacobi relaxation method. The core of such a program must repeatedly update $V(i, j)$ until convergence is achieved. So, there are several tasks involved:

Suitable initial values must be set for $V(i, j)$.

The boundary conditions must be ensured.

Given an estimate for $V(i, j)$ for all points (i, j) , we need to compute an improved estimate using (5.10). With this in mind, we will denote $V_n(i, j)$ as the estimate at iteration n . The right-hand side of (5.10) then contains V_n , while the left-hand side is the result at the next iteration V_{n+1} .

Updating must be repeated until a convergence criterion is satisfied.

We separate these tasks into three separate routines. The first routine (that could be called `initialize-V`) sets the initial values for $V(i, j)$. A second routine, `update-V`, uses the values in $V(i, j)$ to compute an improved estimate for $V(i, j)$, while a third routine, `laplace-calculate`, takes care of calling `update-V` and testing for convergence. The heart of the calculation is `update-V`; this routine uses (5.10) to compute $V(i, j)$ at the next iteration, and also keeps track of the cumulative changes in V . This cumulative change is used by `laplace-calculate` to judge convergence.

⁴In Chapter 7 we will consider the connection between relaxation methods, diffusion, and random walks in a little more detail.

⁵For convenience we will let V and x be unitless, although we could just as well give them units of volts and meters.

⁶For walls that are perfect insulators, the variation of V need not be linear as we assume here. However, if the walls have some small but nonzero conductivity, a linear variation would be found.

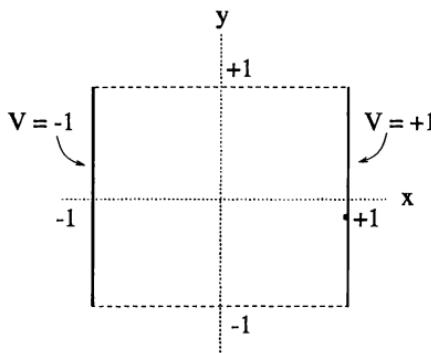


FIGURE 5.2: Geometry and boundary conditions (at $x = \pm 1$) assumed by our program to solve Laplace's equation. The solid lines are the metallic walls of the box, while the dashed lines (at $y = \pm 1$) are insulating walls.

EXAMPLE 5.1 Routine update-V for using V_n to find V_{n+1}

- Let ΔV be the total (cumulative) change in V during this update loop.
- Set initial value of $\Delta V = 0$.
- Loop through all points (i, j) *except* the boundary, where values of V_n are fixed by the boundary conditions.
 - ▷ $V_{n+1}(i, j) = [V_n(i - 1, j) + V_n(i + 1, j) + V_n(i, j - 1) + V_n(i, j + 1)] / 4$
 - ▷ Add $|V_n(i, j) - V_{n+1}(i, j)|$ to ΔV .
- Return $V_{n+1}(i, j)$ and ΔV to the calling program.

Note that in subroutine update-V it is critical to update V *only* at points other than the boundary. This is perhaps obvious, but easy to forget. In addition, since V_n is used to construct V_{n+1} for the next iteration, we must use a different array to store the updated values. Also, we use the variable ΔV to store the sum of the magnitudes of the amount by which each value of V has been changed. This is an obvious measure of how much V_{n+1} is different from V_n , and we clearly expect it to get smaller as we approach convergence to the final, equilibrium value of V . The value of ΔV will be used to check for the degree of convergence by the routine `laplace-calculate`.

EXAMPLE 5.2 Routine laplace-calculate: implement the Jacobi method

- Begin with a guess for $V_n(i, j)$. This could be an initial guess, set by the initialization routine, or the result of the previous iteration of this loop.
 - Call `update-V` so that $V_n(i, j)$ is used to compute an improved guess, $V_{n+1}(i, j)$.
 - Call `update-V` again so that $V_{n+1}(i, j)$ is used to compute an even better guess, that will be stored back in the original array $V(i, j)$.
 - Check if ΔV is small enough to be acceptable. If so, exit the routine; otherwise continue from the start with the current $V(i, j)$ as the new guess.
-

Subroutine `laplace-calculate` performs an iterative updating of V . In this implementation, we actually call `update-V` twice in each loop through the routine. This is done for the following reason. The first call to routine `update-V` will take an array of values $V(i, j)$ as input and return a different array $V_{n+1}(i, j)$ as output. We then call `update-V` a second time, now with $V_{n+1}(i, j)$ as the *input*, and use the original array to hold the output. This saves us the work of swapping values between separate arrays in `laplace-calculate`. We will probably need many iterations so these calls to `update-V` will not be wasted.

Iteration in `laplace-calculate` is stopped when ΔV is sufficiently small – but how small is sufficient? In this particular problem, the boundary values of $V = \pm 1$ set a scale. Thus if the average deviation between V_2 and V is much smaller than this value of 1 per site, then we can assume that we are reasonably close to equilibrium. A suggested value here is 1×10^{-5} per site, and we must not forget to compare ΔV with $1 \times 10^{-5} \times N$ where N is the number of sites.

We have described `laplace-calculate` and `update-V`, since these are the most difficult for this problem. We will leave it to the reader to implement the initialization routine. Here we only note that even the simple initial guess $V(i, j) = 0$ everywhere, except on the boundaries, is sufficient. Of course, the values of V on the boundaries should match the boundary conditions in Fig. 5.2.

As a final piece of advice, it is often necessary to make sure that the relaxation algorithm has sufficient time to get started by requiring that a certain number of iterations, say 10, have been performed before the program is allowed to exit the loop. This is because, with our initial guess of $V = 0$ everywhere, the condition on ΔV could be satisfied trivially very early, when things are still very far from equilibrium.

We have already mentioned the need to set some convergence limit for ΔV , and we suggested the value 1×10^{-5} as a suitable choice for this problem. It is crucial to realize that this does *not* mean that the final result will be within, say, 1×10^{-5} of the true solution. It only means that on average the potential function is changing by less than that amount with each iteration. Of course, using a smaller limit in the convergence test will make the final answer more accurate (and also require more computing time), but we must expect the true error in that answer to be somewhat larger than the error limit used to judge convergence. We will explore this point in the exercises.

Finally, if we wish to obtain the electric field, this can be calculated by differentiating $V(i,j)$. To do this we use the fact that the component of E in the x direction is

$$E_x = - \frac{\partial V}{\partial x}, \quad (5.11)$$

with corresponding relations for E_y and E_z . These derivatives can be estimated using our usual finite difference expressions, such as (5.2) or (5.4). In this case we can use a symmetric form for this derivative

$$E_x(i,j) \approx - \frac{V(i+1,j) - V(i-1,j)}{2 \Delta x}, \quad (5.12)$$

but a less symmetric form [such as (5.2) or (5.3)] would give essentially the same results. One note of caution is that E at the boundary needs to be calculated using a one-sided difference equation since, clearly, using values of V at sites *beyond* the boundary makes no sense.

To give you some feeling for how the relaxation algorithm works, the results for the potential $V(i,j)$, at several stages of the calculation, are printed out below. In this particular case we have taken 7 rows and 7 columns in the array $V(i,j)$. The layout is the same as in Figure 5.2, so the left-most column corresponds to $x = -1$, the right-most column to $x = +1$, and the top and bottom rows to $y = \pm 1$. The initial guess, V_0 , satisfies the boundary conditions and assumes $V = 0$ in the interior of the box.

Initial guess: V_0

```
-1.00 - .67 - .33 .00 .33 .67 1.00
-1.00 .00 .00 .00 .00 .00 1.00
-1.00 .00 .00 .00 .00 .00 1.00
-1.00 .00 .00 .00 .00 .00 1.00
-1.00 .00 .00 .00 .00 .00 1.00
-1.00 .00 .00 .00 .00 .00 1.00
-1.00 - .67 - .33 .00 .33 .67 1.00
```

After the first call to update-V, that is, one updating of our guess for the solution, we obtain V_1 . We see that some of the zeros in the interior of $V(i,j)$ have been replaced by values that bring it closer to solving Laplace's equation. That is, the values of $V(i,j)$ are now closer to being equal to the average of the values in the neighboring sites. However, we still have a long way to go, so update-V must be used several more times.

After 1 call to update: V_1

```
-1.00 - .67 - .33 .00 .33 .67 1.00
-1.00 - .42 - .08 .00 .08 .42 1.00
-1.00 - .25 .00 .00 .00 .25 1.00
-1.00 - .25 .00 .00 .00 .25 1.00
-1.00 - .25 .00 .00 .00 .25 1.00
-1.00 - .42 - .08 .00 .08 .42 1.00
-1.00 - .67 - .33 .00 .33 .67 1.00
```

After nine calls to *update* the relaxation algorithm has produced a very good guess for the potential. We will leave it to you to verify that each value of V in the interior is indeed very close to the average of the values at the neighboring sites, as required by (5.10).

After 9 calls to update: V_9

```
-1.00 - .67 - .33 .00 .33 .67 1.00
-1.00 - .66 - .32 .00 .32 .66 1.00
-1.00 - .65 - .32 .00 .32 .65 1.00
-1.00 - .65 - .31 .00 .31 .65 1.00
-1.00 - .65 - .32 .00 .32 .65 1.00
-1.00 - .66 - .32 .00 .32 .66 1.00
-1.00 - .67 - .33 .00 .33 .67 1.00
```

The final results for the potential are plotted as equipotential contours in Figure 5.3. These contours are simply lines drawn through the locations where the potential has a particular value (as given in Figure 5.3) and are not very exciting to look at. They are just evenly spaced, straight (vertical) lines, as we should have expected from the symmetry of the problem. These results for V can be used to obtain the electric field as explained earlier.

The results for \vec{E} are also given in Figure 5.3, where we show a vector plot. Here each arrow is oriented in the direction of \vec{E} at that location, and the length of each arrow is proportional to the magnitude of the field. We find a uniform field directed from high potential ($x = +1$) to low potential ($x = -1$), as expected.

The simple problem treated above was intended to demonstrate how the relaxation method produces a solution to Laplace's equation. The method is very general and can be readily used in more complicated cases where an analytic solution is difficult or impossible. We now consider two such cases. The first is illustrated in Figure 5.4. It is an infinitely long, hollow prism, with metallic walls and a square cross-section. Inside this prism is a metal bar, also with a square cross-section. We

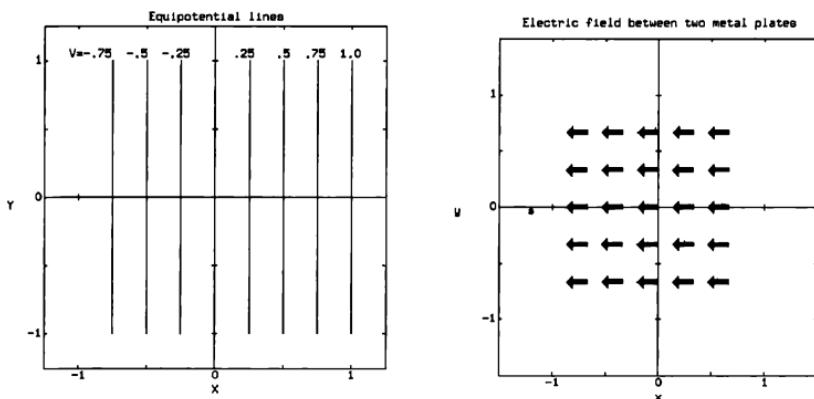


FIGURE 5.3: Left: equipotential lines inside the metal box in Figure 5.2. The value of V at each of the lines is given. Right: the corresponding electric field. Note that here and in later plots the size of each arrow is proportional to the magnitude of the field at the tail of the arrow.

assume that a voltage is applied between the bar and the outer walls, and we want to calculate the potential in the space between them. The program we developed above can be used in this case; all we need to do is modify the initialization and update- V routines so as to take into account the different geometry. We will leave this task to the reader.

The results are given in Figure 5.5. Here we show the equipotential contours, a “three-dimensional” perspective plot of the potential, and the corresponding electric field. The results are consistent with our expectations based on the symmetry of the problem. These considerations tell us that the results should, after the appropriate rotation, be precisely the *same* in each quadrant in the x - y plane. The fact that our solution displays the expected symmetry gives us some extra confidence that we have not made any programming errors. However, we could have taken advantage of this symmetry to make the program execute faster. Since the result must be the *same* in all four quadrants, we really only need to consider one quadrant in the calculation. After obtaining a solution in one quadrant, the values for the other quadrants could then be obtained from the symmetry requirements. This approach would speed-up our program by (approximately) a factor of 4. Actually, for this problem we could do even better. Within each quadrant the solution must also be symmetric with respect to *reflection* about the line $y = \pm x$. Hence, our program really only needs to deal with half of one quadrant, leading to a speed-up by a factor of 8. However, this increased efficiency does come at a cost. Such a program would have to deal carefully with the values of V at the boundaries between the quadrants.

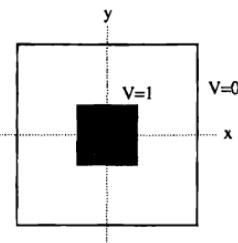


FIGURE 5.4: Schematic cross-section of a hollow metallic prism with a solid, metallic inner conductor. The prism and inner conductor are presumed to be infinite in extent along z . The inner conductor is held at $V = 1$ and the walls of the prism at $V = 0$

and along the lines $y = \pm x$. We will explore this problem in the exercises.

Another interesting use of the relaxation algorithm is the problem of the potential between two parallel capacitor plates. The case of infinite plates can, of course, be handled analytically using Gauss' law. However, here we are interested in what happens when the plates are *finite* in extent. The geometry of the problem is sketched in Figure 5.6. Again, we can modify our earlier program to handle this case. All we need to do is set up the proper boundary conditions for V . We set the plates to $V = \pm 1$, and the square boundary defined by $x = \pm 1$, $y = \pm 1$ surrounding the plates is set to $V = 0$. In analytic calculations we would generally apply the condition $V = 0$ at $x, y = \infty$, but that is usually not practical in a numerical treatment. Here, for simplicity, we apply this condition on the square boundary just described; we will have more to say about how to choose such boundary regions and their effects in the next section. After specifying the boundary conditions on V , the application of the relaxation algorithm is the same as that outlined above. The results are given in Figure 5.7. They are again in accord with our expectations based both on symmetry⁷ and the very schematic drawings given in many textbooks. The electric field is seen to be largest between the two plates. In that region the field is approximately uniform (although this is hard to verify with the rather coarse scale used for this plot) and is directed from high to low potential. The fringing fields at the edges of and outside the plates are also evident.

All of our calculations with the relaxation method have to this point used the Jacobi algorithm, according to which (5.8) is iterated as suggested in Figure 12.47. While this method is simple to describe and implement in a program, it does not have desirable convergence properties for large systems. We will not discuss speed of convergence and related issues in any detail here (see Press et al. [1986]). However, we do note that when the Jacobi method is applied to a hyper-cubic grid⁸

⁷As in the case of the field inside a square prism, this symmetry could be used to speed up the calculation

⁸A hyper-cubic grid of d -dimensions is a d -dimensional grid with nearest neighbors arranged

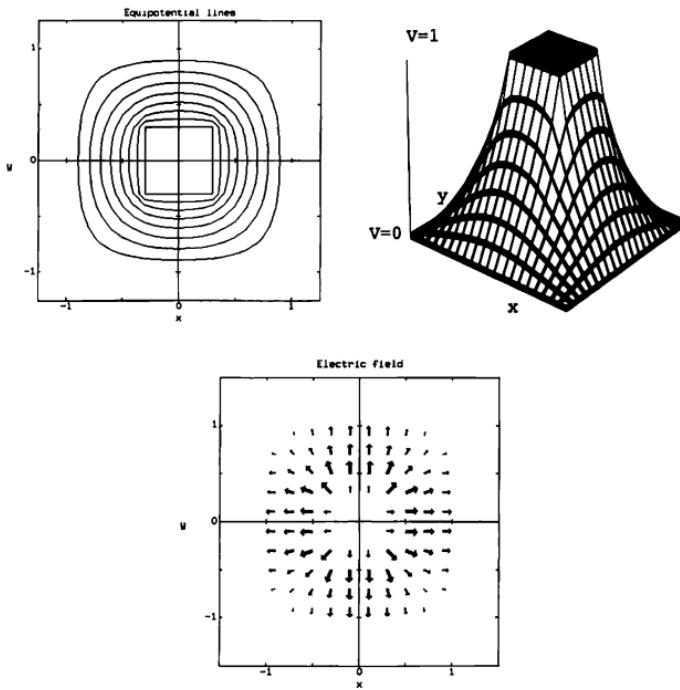


FIGURE 5.5: Electric potential and field inside the prism in Figure 5.4. The sides of the prism in the $x-y$ plane had lengths of 2 units, and the inner conductor had an edge length of 0.6 units. The spatial grid size was 0.1. Upper left: equipotential contours; upper right: perspective plot of the potential; bottom: electric field

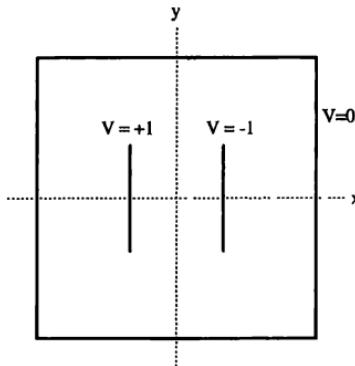


FIGURE 5.6: Schematic of two capacitor plates held at $V = 1$ (left plate) and $V = -1$ (right plate). The square boundary surrounding the plates is held at $V = 0$

of side length L (the examples considered above are all two-dimensional examples of this, being an $L \times L$ grid), the number of iterations required for a given level of convergence can be shown to scale as L^2 . Hence, if the number of grid points is increased by a factor of 2 in each direction in our two-dimensional problems, as might be desirable if we wanted to obtain a solution with better spatial resolution, the computational time increases by a factor of 16. That is, the computational requirement for a given level of accuracy increases as L^4 . Here one factor of L^2 comes from the increase in the required number of iterations, while another factor of L^2 arises from the increased number of grid points that must be dealt with in each iteration. This is a rather large computational cost, so the Jacobi method is generally not employed in large-scale work.⁹

While the Jacobi method is certainly not the last word in relaxation algorithms, it can be used to understand a number of other more efficient approaches. Mathematically the Jacobi method can be expressed as

$$V_{\text{new}}(i,j) = \frac{1}{4} [V_{\text{old}}(i+1,j) + V_{\text{old}}(i-1,j) + V_{\text{old}}(i,j+1) + V_{\text{old}}(i,j-1)], \quad (5.13)$$

where for simplicity we have assumed a two-dimensional problem and taken the grid size to be the same along x and y . Here V_{old} is the potential from the previous iteration, while V_{new} is the new value. In words, (5.13) states that only the old values of the potential are used to compute the values in the next iteration. One

in d orthogonal directions with equal lattice spacing. This is a generalization of the square and simple cubic lattices.

⁹For this reason, Press et al [1986] refers to it as mainly of “academic” interest.

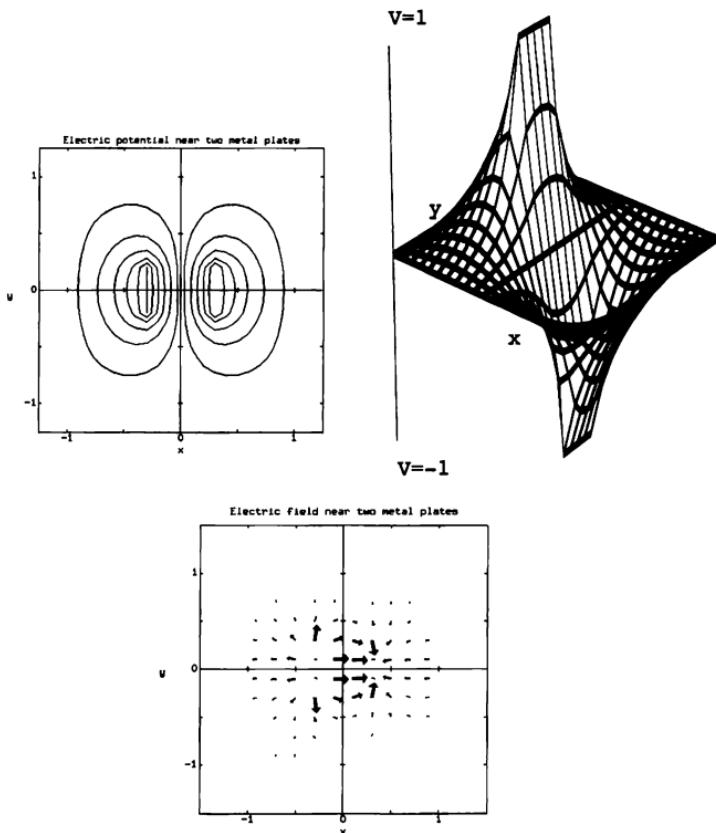


FIGURE 5.7: Electric potential and field near two capacitor plates. The plates were held at $V = +1$ (left plate) and $V = -1$ (right plate), while the square boundary surrounding the plates was held at $V = 0$. The plates are located at $x = \pm 0.3$ and the spatial step size was 0.1. For simplicity the box and plates were assumed to be infinite in extent along the z direction. Upper left: equipotential lines; upper right: perspective plot of the potential; bottom: electric field.

simple (though modest) improvement is the Gauss-Seidel method. The Gauss-Seidel method differs in that it uses the new values as they become available. The order in which they become available depends on how you loop through the grid. If you move along each column (i) from small i to large i , beginning with the top row (small j), the Gauss-Seidel method can be written as

$$V_{\text{new}}(i, j) = \frac{1}{4} [V_{\text{old}}(i+1, j) + V_{\text{new}}(i-1, j) + V_{\text{old}}(i, j+1) + V_{\text{new}}(i, j-1)] . \quad (5.14)$$

While the performance of the Gauss-Seidel algorithm is better than that of the Jacobi method, the improvement is only modest. It turns out that the number of iterations required for convergence with the Gauss-Seidel method is smaller, but only by a factor of 2. A factor of 2 is certainly useful, but we might have hoped for bigger things, such as a factor of L . A bonus of the Gauss-Seidel method is that it does not require that we use two separate arrays to store the potential. Since the new values can be used as soon as they are calculated, there is no need to save the old values separately. In other words, the updating can be performed *in place* with $V(i, j)$ using an algorithm of the form

$$V(i, j) = [V(i-1, j) + V(i+1, j) + V(i, j-1) + V(i, j+1)] / 4 .$$

The problem with the slow convergence of the Jacobi and Gauss-Seidel methods can be overcome with a method known as *simultaneous over-relaxation* (SOR), which can be appreciated as follows. Let $V^*(i, j)$ be the new value of the potential calculated using the Gauss-Seidel method.¹⁰ We can then think of

$$\Delta V(i, j) \equiv V^*(i, j) - V_{\text{old}}(i, j) , \quad (5.16)$$

as the change recommended by the Gauss-Seidel algorithm. However, we have seen in our previous examples that this choice of $\Delta V(i, j)$ is too conservative. So to speed up convergence we will change the potential by a larger amount calculated according to

$$V_{\text{new}}(i, j) = \alpha \Delta V(i, j) + V_{\text{old}}(i, j) , \quad (5.17)$$

where α is a factor that measures how much we “over-relax.” Choosing $\alpha = 1$ yields the Gauss-Seidel method. It turns out that if $\alpha \geq 2$, the method does not converge, while $\alpha < 1$ corresponds to “under-relaxation.” The algorithm known as simultaneous over-relaxation employs (5.17) with a value of α between 1 and 2. It remains for us to determine the best value of α ; that is, the value that yields the fastest convergence. This is a question that requires a more detailed analysis than we can discuss here (see Press et al. [1986]), so we will only give a few pointers. For a problem on a two-dimensional square grid like the ones we have considered and with fixed-value boundary conditions (called *Dirichlet* boundary conditions) on all four sides, the best choice for α is

$$\alpha \approx \frac{2}{1 + \pi/L} . \quad (5.18)$$

¹⁰Note that applying SOR to the Jacobi method results in an unstable algorithm.

If your problem uses a different grid geometry or boundary conditions you should check Press et al. (1986), and the references they list on the subject, for guidelines on how to choose α . The SOR method is a bit more work to implement since, whereas the Gauss-Seidel method can use in-place updating of V , for SOR we need both the so updated V^* and the original one in order to calculate the new SOR-updated V . However, for this extra effort we gain in speed of convergence. The number of required iterations (for a given accuracy) now scales as L , so for problems with large grids the computational savings can be substantial. We will explore the simultaneous over-relaxation method in the exercises. As a final note we should point out that simultaneous over-relaxation is itself not the last word in algorithms, so if you need to solve really large problems of this type, it would pay to read more about these issues in the references.

EXERCISES

- 5.1. Solve for the potential in the prism geometry in Figure 5.4.
- *5.2. Use the symmetry of the problem described in Figure 5.4 to write a program that solves for V by calculating the potential in only half of one quadrant of the x - y plane.
- *5.3. Use the symmetry of the capacitor problem (Figure 5.6) to write a program that obtains the result by calculating the potential in only one quadrant of the x - y plane.
- 5.4. Investigate how the magnitude of the fringing field of a parallel plate capacitor, that is, the electric field outside the central region of the capacitor in Figure 5.6, varies as a function of the plate separation.
- 5.5. Study the accuracy of the relaxation method by solving any of the problems considered in this section with several different values of the convergence (error) limit. Compare the results for V and \vec{E} , and estimate how the actual error in either of these quantities compares to the convergence limit. Theoretically,¹¹ the number of iterations required to achieve p significant digits should be proportional to p for the Jacobi, Gauss-Seidel, and SOR methods. Compare your results above to this theoretical expectation.
- 5.6. Calculate the electric potential and field near a lightning rod. Model this as a very long and narrow metal rod held at a high voltage, with one end near a conducting plane. Of special interest is the field near the tip of the rod.
- *5.7. Write two programs to solve the capacitor problem of Figures 5.6 and 5.7, one using the Jacobi method and one using the SOR algorithm. For a fixed accuracy (as set by the convergence test) compare the number of iterations, N_{iter} , that each algorithm requires as a function of the number of grid elements, L . Show that for the Jacobi method $N_{\text{iter}} \sim L^2$, while with SOR $N_{\text{iter}} \sim L$.

5.2 POTENTIALS AND FIELDS NEAR ELECTRIC CHARGES

In the last section we learned how to compute the electric potential and field in regions of space that are free of any electric charges. We next consider how to

¹¹See Press et al. (1986).

include charges in the problem. Laplace's equation applies only in charge-free regions. If we include charges we obtain Poisson's equation, which is given by

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho}{\epsilon_0}, \quad (5.19)$$

where ρ is the charge density and the permittivity constant ϵ_0 takes care of the units (in SI units). To deal with this numerically we must add the charge density to the right-hand side of (5.8). The result is

$$\begin{aligned} V(i, j, k) = & \frac{1}{6}[V(i+1, j, k) + V(i-1, j, k) + V(i, j+1, k) + V(i, j-1, k) \\ & + V(i, j, k+1) + V(i, j, k-1)] + \frac{\rho(i, j, k)(\Delta x)^2}{6\epsilon_0}, \end{aligned} \quad (5.20)$$

where we have added the spatial indices to ρ to remind us that the charge density is, in general, a function of position. We have also assumed that the spatial grid size is the same in all directions ($\Delta x = \Delta y = \Delta z$).

Any of the different versions of the relaxation method that we described in the previous section can be used to deal with (5.20). To generalize our program in order to handle this case, we employ a three-dimensional array to hold the charge density. In the program listing given in the previous section, the line of update-V that calculates the new values of the potential for a three-dimensional problem then becomes something like

$$\begin{aligned} V_2(i, j, k) = & \frac{1}{6}[V(i-1, j, k) + V(i+1, j, k) + V(i, j+1, k) + V(i, j-1, k) \\ & + V(i, j, k+1) + V(i, j, k-1)] + \frac{\rho(i, j, k)\Delta x^2}{6}, \end{aligned}$$

where the array $\rho(i, j, k)$ contains the charge density per grid element and Δx is the spatial step size.¹²

The relaxation algorithm works here just as it did with Laplace's equation. An instructive case is that of a single-point charge in three dimensions. For our calculations we need to have some boundary conditions for the potential, so we will place this charge at the center of a metal box whose walls are held at $V = 0$ (Figure 5.8). The charge density is zero, except at the grid location at the center of the box where it is given by $\rho(0, 0, 0) = q/dx^3$. In the limit of a very large box we should recover the result familiar to us from Coulomb's law. Results for the potential and electric field are shown in Figure 5.9, where we have assumed a point charge of magnitude $q/\epsilon_0 = 1$. Both the electric field and the potential fall off very rapidly as we move away from the charge.

Before we compare these results with the exact answer, we should comment on the overall symmetry of the equipotential lines. From the symmetry of the problem we would expect the potential to be spherically symmetric, except near the faces of the box. However, even if we were to make the box very large, our

¹²Here we have chosen units so that the factor ϵ_0 is unity, for convenience.

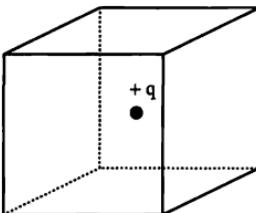


FIGURE 5.8: Schematic of a point charge $+q$ located at the center of a metal box. The faces of the box are all held at $V = 0$

numerical solution would still impose a sort of cubic symmetry on the system, since the spatial grid is cubic. This distorts the equipotential lines and gives them a distinctly noncircular shape when projected as in Figure 5.9. This effect could be reduced by choosing a smaller grid size. An alternative approach would be to use the symmetry of the problem. For example, the metal box could be replaced with a spherical metal shell. Since the problem would then have full spherical symmetry (assuming we keep the point charge at the center of the sphere), we could then rewrite Poisson's equation in terms of the radial coordinate and obtain $V(r)$ with the relaxation algorithm. This approach would greatly speed up our program and will be explored in the exercises. However, before we get too carried away with exploiting all of the possible symmetries in each problem, we have a cautionary note. Highly symmetric problems are generally the ones most amenable to analytic solution. The kinds of problems you are likely to encounter in numerical work will be those with very little, if any, symmetry. Even so, recognition of symmetries is important since they can at least provide a check on the correctness of a numerical solution.

Returning to the results in Figure 5.9, it is instructive to compare them quantitatively with the exact result for this case (Coulomb's law)

$$V(r) = \frac{q}{4\pi\epsilon_0 r}, \quad (5.21)$$

where $r = \sqrt{x^2 + y^2 + z^2}$ is the distance from the charge. This comparison is shown in Figure 5.10 where we plot the numerical solutions for boxes with edge lengths of 2 and 4. Near the charge, that is, as $x \rightarrow 0$, the numerical results agree with the exact solution. However, as x increases, the numerical results fall systematically below the exact result. This is because of the different boundary conditions assumed in the two cases. The analytic result, (5.21), assumes that $V = 0$ at an *infinite* distance away, while our numerical solution takes $V = 0$ on the faces of the box, which are much closer. Indeed, we see that the numerical result for V goes smoothly to zero at the face of the box. Thus, the presence of the grounded box forces V to vanish faster than it would otherwise. In short, the numerical and analytic solutions differ because they are not dealing with the same problem.

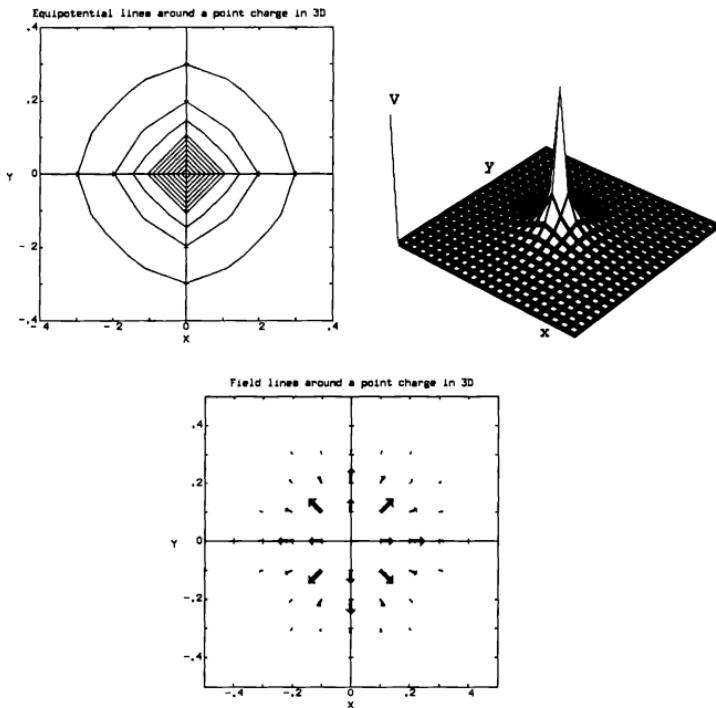


FIGURE 5.9: Results for the electric potential and field near a point charge located at the center of a metal box. The box had an edge length of 2 (x , y , and z all ran from -1 to $+1$) and the spatial step size was 0.1. Upper left: equipotential lines; upper right: perspective plot of the potential in the $z = 0$ plane; bottom: electric field (note that the length of each arrow is proportional to the field strength at the point where the base of the arrow is located). Also note that the arrows closest to the origin are "distorted", the singularity at the origin together with the finite difference expression used to compute E makes the value smaller than it should ideally be. The plots on the upper left and right show only the potential and field near the center of the box.

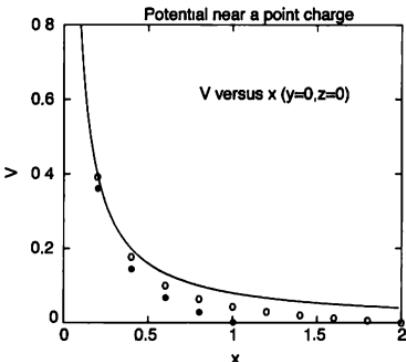


FIGURE 5.10: Numerical results for the potential near a point charge obtained using the Jacobi relaxation method compared with the exact result (the solid line). The open circles were obtained with a box of edge length 4 (so the faces of the box were at $x = \pm 2$), and the filled circles with a box of length 2. The magnitude of the charge was such that $q/\epsilon_0 = 1$ and the spatial grid size was 0.2 for both calculations.

This illustrates something that must always be kept in mind in numerical work. For many problems we would like to impose conditions on the behavior at "infinity," but this is often difficult to do in a numerical treatment. Therefore we impose these conditions at the boundaries of some suitable region; for the problem here this is just the box. We then hope that the finite size of the box will not distort or otherwise significantly affect the numerical solution near the center of the box, but we must always be on guard for such effects. In our problem involving the potential near a point charge, we have observed that the influence of the finite size of the box is felt even fairly near the center. The reason for this is the relatively slow spatial dependence in Coulomb's law. From (5.21) we find a $1/r$ dependence which means that the effect of the box will be felt at rather long distances from the face of the box. Fortunately, boundary effects often fall off much faster than this (the rate of fall off depends on the problem), so they aren't always as large as in the present case.

EXERCISES

- 5.8.** Extend our treatment of a point charge in a metal box to deal with the case in which the charge is located near one face of the box. Study how the equipotential contours are affected by the proximity of a grounded surface (the face of the box).
- *5.9.** In spherical coordinates Poisson's equation has the form

$$\frac{1}{r} \frac{\partial^2}{\partial r^2} (r V) = - \frac{\rho}{\epsilon_0}, \quad (5.22)$$

where we have assumed a spherically symmetric problem so that V is a function only of the distance from the origin. Solve this equation numerically using the relaxation method for a point charge at $r = 0$, imposing $V = 0$ some large distance away. Compare your result with Coulomb's law, (5.21). *Hint:* This problem is made difficult by the factor of $1/r$ on the left side of (5.22) and its effect on constructing a numerical solution, especially when the charge distribution is a singular function at $r = 0$, as is the case for a point charge. One way to deal with this problem is to instead give the "point" charge a small but nonzero spatial size; that is, assume that there is a uniform charge density inside a small sphere of radius r_{\min} . If you take this approach, be sure to pick a grid size smaller than r_{\min} . Convenient parameter choices are $r_{\min} = 0.2$ with a grid size of 0.025, and $V = 0$ imposed at $r = 5$, but you should also try other values. Compare your result for $V(r)$ with Figure 5.10.

- *5.10. Investigate the performance of the simultaneous over-relaxation algorithm for a point charge in two and three dimensions. *Hint:* In two dimensions we know the optimum choice of the over-relaxation parameter, α in (5.18). In three dimensions you should determine the optimum choice of this parameter by observing the speed of convergence for different values of α . How sensitive is the convergence to the value of α ?

5.3 MAGNETIC FIELD PRODUCED BY A CURRENT

We now turn to several problems associated with magnetic fields. Our goal is to calculate the magnetic field produced by an electric current for several different geometries. Besides illustrating how to compute the magnetic field in cases that cannot easily be dealt with analytically, these problems will also introduce a few of the ideas involved with numerical integration.

We start with the simplest possible problem, the magnetic field produced by a straight wire. The geometry is sketched in Figure 5.11 where we define several variables that enter the Biot-Savart law for the magnetic field produced by a current I flowing in a wire segment $d\vec{z}$

$$d\vec{B} = \frac{\mu_0 I}{4\pi} \frac{d\vec{z} \times \vec{r}}{r^3}. \quad (5.23)$$

Here \vec{r} is the vector that runs from the segment carrying the current to the point in question, and μ_0 takes care of the units (in SI). For this particular geometry the cross product can be written in terms of the angle θ in Figure 5.11 yielding

$$dB = \frac{\mu_0 I}{4\pi} \frac{dz \sin \theta}{r^2}, \quad (5.24)$$

where we have dropped the vector symbols, since the field from each segment of the wire is directed perpendicular to the plane in Figure 5.11.

The total field is the integral of dB over the length of the wire. To estimate this integral we follow our usual pattern of converting differentials into small differences; dz becomes Δz and the integral becomes a sum from one end of the wire to the other

$$B \approx \sum \frac{\mu_0 I}{4\pi} \frac{x \Delta z}{(z^2 + x^2)^{3/2}}, \quad (5.25)$$

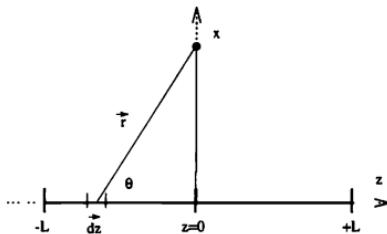


FIGURE 5.11: Geometry for computing the magnetic field near a straight wire. The wire is of length $2L$ and we consider the field at a point on the x axis. This axis is perpendicular to the wire and intersects the wire at its center. The current is assumed to flow from left to right, so at the point in question (the dot on the x axis) the field is perpendicular to the plane of the drawing and directed out of the plane.

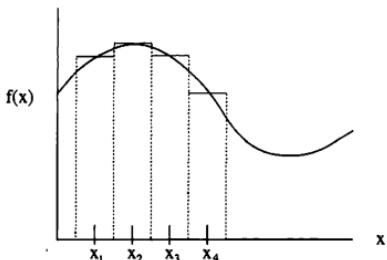


FIGURE 5.12: Geometrical interpretation of how the sum (5.26) approximates the area under a function $f(x)$.

where we have now expressed r and $\sin \theta$ in terms of x and z . This sum yields an *approximate* value for B , where the numerical error can be reduced by making Δz smaller. This simple method can be quite effective in computing an integral numerically. However, there are other more accurate methods of numerical integration, which are not much more complicated. A few of these methods are described in Appendix E, where we consider the general problem of numerical integration.

Integrating a function $f(x)$ can be viewed as the problem of finding the area under the corresponding curve, as illustrated in Figure 5.12. The approximation to

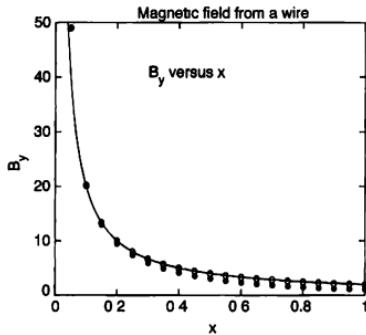


FIGURE 5.13: Magnetic field near a straight wire for a current I such that $\mu_0 I = 1$. Filled circles for a wire of length 1, open circles, for a length of 10. The solid curve is the exact result for an infinite wire. The step size for the integration was $\Delta z = 0.1$.

the integral we used in deriving (5.25) yields in the general case¹³

$$\int f(x) dx \approx \sum_i f(x_i) \Delta x , \quad (5.26)$$

where Δx is the size of the grid step along x and $x_i = i\Delta x$. Each term in this sum is the area of one of the rectangles in Figure 5.12. This approximation for the integral thus amounts to approximating the region of interest by a sequence of rectangles of width Δx and height $f(x_i)$. As Δx is made smaller these rectangles provide a better and better approximation to the true area.¹⁴ In Appendix E we introduce various higher order methods which achieve considerably better accuracy. A common approach is known as Simpson's rule, which employs the same basic approach, except that it uses parabolic arcs to approximate the top edges of the "rectangles."

Returning to the problem of a magnetic field produced by a straight wire, the sum in (5.25) can be evaluated numerically. Figure 5.13 shows results for the field strength as a function of distance from the wire. Here we consider wires of two different lengths and also show the exact result for an infinitely long wire as obtained from Ampere's law, $B = \mu_0 I / (2\pi r)$. The results for the longer wire are seen to be in agreement with the infinite wire values, except at locations nearest the wire. These slight deviations are due to the finite grid size, Δz . As we approach the wire (i.e., for small x), we eventually reach the regime where the grid size is no longer small compared to x , and errors due to the finite grid become important. We will investigate in the exercises how these errors vary with grid size.

¹³Here, and in Fig. 5.12 we have not specified how to treat the ends of the integration region (the first and last rectangles). This and other important details are discussed in Appendix E.

¹⁴Assuming, of course, that the function is not pathological.

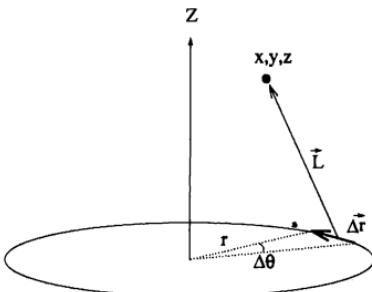


FIGURE 5.14: Geometry of the current loop problem.

EXERCISES

- 5.11. Calculate the field from a straight wire using Simpson's rule (see Appendix E), and compare it with the result obtained from (5.25) for the same grid size, Δz .
- 5.12. Evaluate (5.25) for different grid sizes and compare the results with Ampere's law. Derive a rule of thumb concerning how small the grid size must be in comparison with the distance from the wire, in order for the calculated field to be within 5 percent of the exact result.
- 5.13. Calculate the value of π by using numerical integration to estimate the area of a circle of unit radius. Observe how your estimate approaches the exact value ($3.1415926\dots$) as the grid size in the integration is reduced.

5.4 MAGNETIC FIELD OF A SOLENOID: INSIDE AND OUT

We next consider the magnetic field produced by somewhat more complicated current arrangements. The first case we consider is a circular current loop. The field along an axis perpendicular to the plane of the loop and passing through its center, can be calculated analytically. However, the field at points off this axis cannot be evaluated in closed form. To calculate the field at a general location, we take the approach employed in the previous section and adapt the sum (5.24) to the case of a wire with a circular shape. We consider a circle of radius r lying in the x - y plane (Figure 5.14) and find

$$\Delta \vec{B} \approx \frac{\mu_0 I}{4\pi} \frac{\Delta \vec{r} \times \vec{L}}{L^3}. \quad (5.27)$$

The total field is the sum of these terms as we traverse the loop. This can be done by letting θ vary from 0 to 2π with $\Delta r = r\Delta\theta$.

Implementing this calculation with a program requires a careful consideration of the terms in the cross product, since $\Delta \vec{r} \times \vec{L}$ generally will have nonzero components along x , y , and z . The calculation thus yields the three components of \vec{B} as a function of position, so the question of how to best display the result requires a little thought. In Figure 5.15 we show the field lines in the plane perpendicular to

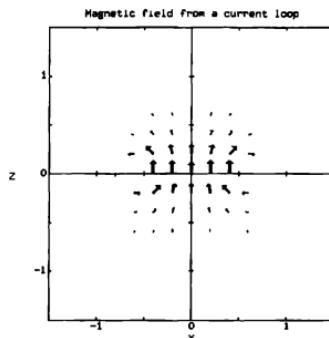


FIGURE 5.15: Magnetic field near a current loop (this is the $y = 0$ plane in Figure 5.14). The radius of the loop was 0.5 and the step size $\Delta\theta$ was $2\pi/10$.

plane of the loop, which cuts through the center of the loop. We see that the field lines circulate around the wire, as expected.

As noted above, these results for the field cannot be expressed analytically except on the axis of the loop (z). It is thus useful to check our calculations against the exact result in this case. For points on the z axis the magnetic field is directed along z , and the Biot-Savart law can be evaluated in closed form

$$B_z = \frac{\mu_0 I r^2}{2(r^2 + z^2)^{3/2}}. \quad (5.28)$$

This exact answer is compared with our numerical results in Figure 5.16. The results agree, which gives us some confidence in our program. As we have emphasized in connection with previous problems, it is always advisable to check numerical results against analytic calculations whenever possible. A program always gives an answer (once the syntax errors are removed), but that is no guarantee that it is correct!

Another interesting case is that of a solenoid. While the field along the axis of a solenoid can be calculated analytically using the Biot-Savart law, the field off the axis cannot be obtained in closed form. This field can be computed numerically using a program very similar to the one for the current loop. The only difference is that now the current follows a helical path, so $\Delta\vec{r}$ has three nonzero components. We will leave the programming for you. Some results are given in Figure 5.17, which shows \vec{B} in both the x - z and x - y planes (the solenoid is centered on the z axis).¹⁵

¹⁵In situations like this, three-dimensional visualization can be very useful. You may save the computed results for the components of \vec{B} into files and then use your favorite three-dimensional plotting program, or it may be a good time to explore programming languages with built-in

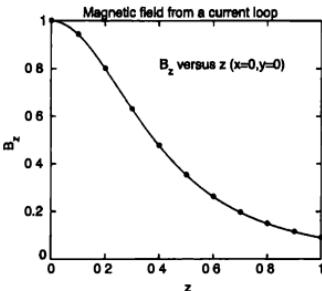


FIGURE 5.16: Magnetic field produced by a current loop, on the axis of the loop. The field is directed along the z axis in Figure 5.14. The radius of the loop was 0.5 and the step size $\Delta\theta$ was $2\pi/10$. For convenience, the current was chosen such that $\mu_0 I = 1$. The solid line is the exact result.

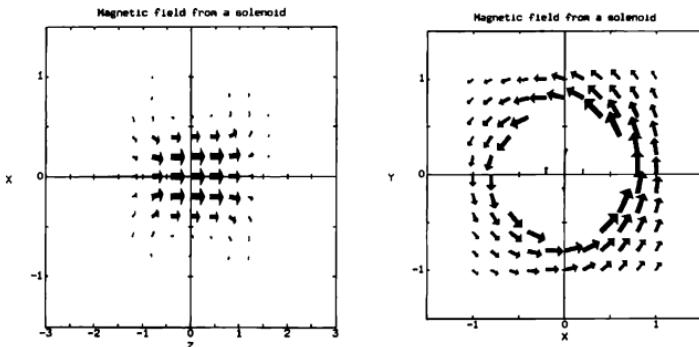


FIGURE 5.17: Magnetic field produced by a solenoid. Left: \vec{B} in the x - z plane. The solenoid runs along the z axis and is 2 units long, its radius is 0.5 units. Right: \vec{B} in the x - y plane ($z = 0$). The magnitude of B here is much smaller than in the plot on the left. The number of turns in the solenoid was 80 and the step size was $\Delta\theta = 2\pi/20$.

The field near the z axis inside the solenoid is shown on the left. It is seen to be (relatively) large, uniform, and parallel to the axis of the solenoid (here the z axis), which is what we are all taught in our courses on electricity and magnetism.¹⁶ The field outside the solenoid is often taken to be zero, since it is much smaller than the field inside (note that the effective scale on the right in Figure 5.17 is greatly magnified). That there is a field outside can be understood if we note that since the solenoid winding follows a helical path, there is also a component of the current *along* the z axis. This component is much smaller than the component that cuts the x - z plane, since there are many windings that cut this plane, but only one that cuts through the x - y plane. Nevertheless, the current along z is not zero, so there is a small field whose direction can be roughly estimated by modeling this current by a single wire that runs along z and using the right hand rule. Hence, the field outside the solenoid should circulate counterclockwise as viewed from the large z end of the solenoid; this is precisely what is seen in Figure 5.17. It is interesting to note that the field outside the solenoid is *not* quite symmetric about the z axis. We have chosen the geometry of the solenoid such that the helix carrying the current pierces the x - y plane at $y = 0$, $x > 0$, making the field slightly larger on that side of the solenoid in the plot on the right in Figure 5.17.

EXERCISES

- 5.14. Write a program to calculate the magnetic field for your favorite current distribution. One possibility is a pair of loops of radius r , with one loop lying in the x - y plane and the other in the y - z plane. Another possibility is the solenoid considered in Figure 5.17.
- 5.15. Consider the magnetic field produced by a set of two coils that are both centered on the z axis, are both parallel to the x - y plane, and are both of radius r (see Figure 5.18). Let the separation between the coils also be r . These are called Helmholtz coils and are noteworthy because they produce a particularly uniform field near the point centered between them. Calculate numerically the field produced by these coils both on the z axis (where you can compare with the exact result from Biot-Savart law), and along the x axis.
- 5.16. Calculate the magnetic field both inside and outside a coil wrapped on a torus. Be sure to compare your result for B on the axis of the torus with the exact answer.

visualization functions. An excellent example of this is VPython. You can find information on this language at the following web site: www.vpython.org.

¹⁶The magnitude of the field at the center of the solenoid has been checked against the analytic result (they agree).

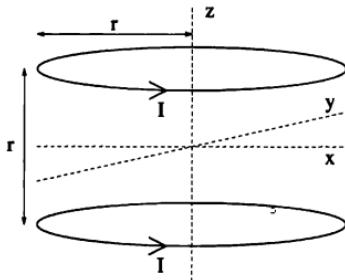


FIGURE 5.18: Helmholtz coils: two current loops that are parallel and whose separation is equal to their radius. This geometry is noteworthy because the field at the center (the origin) is particularly uniform

REFERENCES

- [1] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapter 17 discusses the relaxation method and analyzes its convergence properties. They also give a good description of Simpson's rule and related algorithms.

CHAPTER 6

Waves

In this chapter we consider several topics associated with wave motion. While the central ideas in this discussion are applicable to virtually all types of waves, we will find it convenient (and interesting, we hope) to consider the particular case of waves on a string. After introducing and developing a solution for the wave equation in the ideal case (that is, for a perfectly flexible, frictionless string), we then extend our modeling to deal with waves in several more realistic situations. This will lead us to some interesting issues that arise in connection with musical instruments.¹

6.1 WAVES: THE IDEAL CASE

The central equation of wave motion is

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2}, \quad (6.1)$$

which is usually referred to as the wave equation. This equation arises in many situations, including waves on a string, electromagnetic waves, waves on the surface of a lake, and sound waves. Here we will use a language appropriate for waves on a string, although our methods and conclusions will apply to other cases as well. y is then the displacement of the string from its equilibrium (undisturbed) form, x is distance measured along the string, t is the time, and c is a parameter that turns out to be the speed with which a wave moves on the string.

It will be helpful for some of our later development to review how this equation comes about for wave motion on a string. Figure 6.1 sketches a portion of a long string. The string runs (on average) along the horizontal direction (x), and we wish to determine the equation of motion for displacements along the vertical (y). Consider now the motion of the segment of the string denoted by the heavy line and labeled y_i in Figure 6.1. This segment has a length Δx and a mass $\mu \Delta x$, where μ is the mass per unit length of the string. We denote the vertical displacement of this i -th segment of the string by y_i .

The force on segment i will come from the neighboring segments, $i \pm 1$. The vertical components of these forces will be equal to the tension in the string T , multiplied by the $\sin \theta$, where θ is the angle that the string makes with the horizontal, as defined in Fig. 6.1. If we assume that the string displacement is small, so that the angles θ_i are also small, we can write the equation of motion of segment i as

$$(\mu \Delta x) \frac{d^2 y_i}{dt^2} = T \sin \theta_{i+1} - T \sin \theta_i, \quad (6.2)$$

¹We will develop a few of these themes in more detail in Chapter 11.

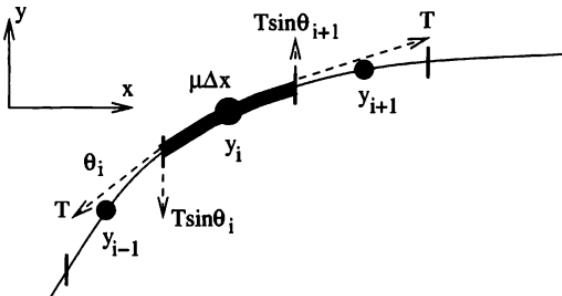


FIGURE 6.1: Forces on a short segment of a string

and then use a finite difference approximation for the angles

$$\sin \theta_i \approx \frac{y_i - y_{i-1}}{\Delta x}. \quad (6.3)$$

We obtain

$$\frac{d^2 y_i}{dt^2} \approx \left(\frac{T}{\mu}\right) \frac{y_{i+1} - 2y_i + y_{i-1}}{(\Delta x)^2} \approx \left(\frac{T}{\rho}\right) \frac{\partial^2 y}{\partial x^2}, \quad (6.4)$$

which is precisely of the form of (6.1).

From direct substitution you can verify that *any* function of the form $y = f(x \pm ct)$ is a solution to (6.1) (provided, of course, that the function is not so pathological that the necessary derivatives do not exist).² The forms $y = f(x \pm ct)$ describe waves traveling in either direction ($+x$ or $-x$) along the string. The function f describes the shape of the wave, and we will see shortly that it is determined by the initial conditions (that is, how the string is excited) and the boundary conditions (how the end points are held).

Our main job in this section is to develop a numerical scheme for solving (6.1). Superficially, (6.1) bears some resemblance to Laplace's equation, as both contain a second-order spatial derivative. However, with (6.1) we wish to solve for the time-dependent solution $y(x, t)$ rather than a stationary solution. Thus, the relaxation methods described in Chapter 5 are not suitable. We must therefore attack the wave equation with rather different numerical treatments than those that we employed in our work with Laplace's equation. After we develop an algorithm suitable for (6.1), we will use it to explore several aspects of wave motion. In later sections we move on to consider the spectral analysis of wave motion and also the behavior of more realistic models, including the stiffness of the string (which leads to an effect known as dispersion) and friction (i.e., damping).

²Moreover, since the wave equation is linear, the sum of any two solutions is also a solution. This fact will be essential to our work in Section 6.4.

To construct a numerical approach to the wave equation, we will, as usual, write it in finite difference form. We treat both x and t as discrete variables with $x = i\Delta x$ and $t = n\Delta t$. The displacement of the string is then a function of both i and n , which we write as $y(i, n) \equiv y(x = i\Delta x, t = n\Delta t)$ so that the first index of y specifies the spatial coordinate and the second index corresponds to time. We have already derived the needed expression for the second partial derivative (see the discussion of Laplace's equation in Chapter 5), and inserting it into the wave equation, (6.1), yields³

$$\frac{y(i, n+1) + y(i, n-1) - 2y(i, n)}{(\Delta t)^2} \approx c^2 \left[\frac{y(i+1, n) + y(i-1, n) - 2y(i, n)}{(\Delta x)^2} \right]. \quad (6.5)$$

This problem is in some respects similar to the initial-value problems we encountered in Chapters 1–4. In those cases we typically considered the time-dependent motion of an object using an equation of motion that was (usually) derived from Newton's second law. In order to obtain a solution for those differential equations we required some initial conditions, which were often the position and velocity at an initial time. To construct a solution of (6.5) we will also require some initial conditions. For simplicity we will assume that the displacement of the string at times prior to and including $t_n = n\Delta t$ is known. We then want to derive an expression for the displacement at the next time step, $t_{n+1} = (n+1)\Delta t$. Rearranging (6.5) we can express $y(i, n+1)$ in terms of y at previous time steps, with the result

$$y(i, n+1) = 2[1 - r^2]y(i, n) - y(i, n-1) + r^2 [y(i+1, n) + y(i-1, n)], \quad (6.6)$$

where $r \equiv c\Delta t/\Delta x$. Thus if we know the string configuration at time steps n and $n-1$, we can calculate the configuration at step $n+1$. Typically, we might know the string configuration at $t=0$, which we will refer to as $y_0(x)$. Since we are dealing with a second-order differential equation, we require two initial conditions.⁴ Knowledge of the string configuration at two consecutive time steps is thus sufficient.⁵ One natural choice, and the one that we will make here, is to assume that the string is held fixed with shape $y_0(x)$ prior to $t=0$.

There is one more important issue that must be dealt with, namely how to treat the ends of the string. We have several options; perhaps the simplest choice is to treat the ends as fixed, that is, tied down so that they cannot move ($y(0, n) \equiv y(M, n) \equiv 0$), and then restricting the updating of $y(i, n)$ to the interior of the string (the region extending from $i=1$ to $i=M-1$) using (6.6). However, we might also want to consider different boundary conditions, such as free ends, or ends that are intermediate between fixed and free. For example, to see how to implement free ends, let us look again at (6.2). The first term on the right is the force exerted on segment i by the string segment $i+1$ on its right, while the second term is the force from segment $i-1$ on the left. If we now consider the motion of the left end, $i=0$, which is assumed to be free, only the first term in (6.2) will

³Note that (6.5) applies in the “interior” of the string, and not at the ends.

⁴Actually, we require two initial conditions for each (discrete) spatial element of the string.

⁵The initial position and velocity of the string would also suffice.

be present. Carrying this on to (6.4), we see that the corresponding equation of motion for the end segment y_0 must have the form⁶

$$\frac{d^2y_0}{dt^2} \approx \left(\frac{T}{\mu}\right) \frac{y_1 - y_0}{(\Delta x)^2}. \quad (6.7)$$

We will see shortly that the boundary conditions greatly affect the manner in which waves are reflected when they reach the ends of the string.

The design of a routine `propagate` that implements (6.6) with fixed end boundaries is shown below.

EXAMPLE 6.1 Routine propagate for simulating waves on a string. Use (6.6) to update $y(i, n)$ through one time step.

- Set the parameter combination $r = c\Delta t/\Delta x$ (this can be done once, at the start of the calculation).
- Loop through the interior points $i = 1$ through $i = M - 1$. Update according to (6.6):
 - ▷ $y(i, n+1) = 2[1 - r^2]y(i, n) - y(i, n-1) + r^2[y(i+1, n) + y(i-1, n)].$
- The ends at $i = 0$ and $i = M$ are fixed, so $y(0, n) = y(M, n) = 0$ for all time steps n .

During the course of a calculation, the time index n will run from the initial time ($n = 0$) to some final value corresponding to the time interval of interest. Since the string is modeled as a large number of discrete elements (M might be 1000 or more), we will quickly generate a lot of “data.” Fortunately, we don’t actually have to permanently keep all of this information. Since our algorithm (6.6) requires only the values of y at the current and previous time step, we really only need to store information for 3 consecutive time steps (the “previous,” “current,” and “next,” time steps). We thus only need three values of the time index, and might let $n = 1$ correspond to the previous time step, $n = 2$ be the current time step, and $n = 3$, be the next time step. The routine `propagate` would then loop through the spatial index as it calculates the string displacement at the next time step, placing the result in $y(i, 3)$ (which could then be stored in a file, etc.). In preparation for the next iteration we would then shift the values in $y(i, 2)$ into $y(i, 1)$, and those in $y(i, 3)$ into $y(i, 2)$.⁷

Some results obtained with our routine `propagate` are shown in Figure 6.2, where we have plotted the string displacement at different times. The uppermost plot is the initial ($t = 0$) displacement of the string. Here we have assumed a simple

⁶See Problem 6.1 for more on how to interpret (6.7).

⁷This shifting could be avoided by simply treating the time index n in a “cyclical” manner. This is not hard to do, but we will leave it for the curious reader. We could easily spend a great deal of time streamlining and otherwise tinkering with our programs, but if we did, we would probably still be in Chapter 1.

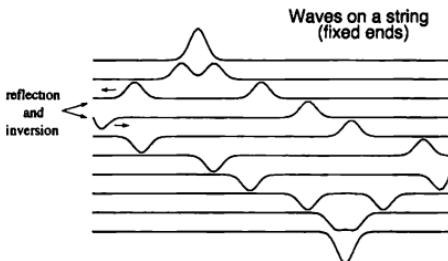


FIGURE 6.2: Waves propagating on a string with fixed ends. The string had a length of 1 m and the simulation used the values $c = 300 \text{ m/s}$, $\Delta x = 0.01 \text{ m}$, and $\Delta t = \Delta x/c$. The initial string profile is given at the top, and successive traces (moving from top to bottom) show the string at progressively later times. An example of reflection and inversion of a wavepacket when it reaches the left end of the string is indicated.

“Gaussian pluck” of the string. That is, we have taken the initial string profile to be

$$y_0(x) = \exp[-k(x - x_0)^2], \quad (6.8)$$

where the displacement is centered at x_0 , and k is a factor that determines the width of the Gaussian envelope. In Figure 6.2 we use $x_0 = 0.3 \text{ m}$ and $k = 1000 \text{ m}^{-2}$, and the string runs from $x = 0$ to 1 m . We have included units here as we will be applying some of these ideas to (semi-) realistic piano strings in the following sections.

While the initial displacement has a (single) Gaussian profile, it quickly splits into two separate wavepackets, or pulses, which propagate in opposite directions along the string. This can be seen from the second and third traces in Figure 6.2, where the two wavepackets have the same shape (as they must, by symmetry) as they move apart. As t increases, the wavepacket on the left reaches the end of the string at $x = 0$, where it finds that the string is held fixed [$y(\text{end}) = 0$]. This wavepacket is reflected, but the reflection process also *inverts* the wave so that the displacement is now negative. This inverted wavepacket then travels toward the right. Meanwhile, the wavepacket on the right eventually encounters the opposite end of the string, and it too is inverted on reflection (the seventh trace from the top in Figure 6.2). These two inverted wavepackets then come together, and in the bottom trace they have overlapped so as to reform the initial string displacement (although it is now inverted).

The results in Figure 6.2 show that our numerical approach gives an accurate solution of the wave equation. Indeed, things seem almost too easy. It turns out that there are, in fact, some subtle numerical issues that we have managed to avoid so far. We have in previous chapters made several comments concerning the accuracy of finite difference approximations, and as a general rule we know that

smaller step or grid sizes usually lead to smaller numerical errors. The problem we are dealing with here involves two step sizes, one for t and one for x , and the question now is how best to choose them. You may have noticed that the step sizes we used in Figure 6.2 satisfied the condition $c\Delta t/\Delta x = 1$; in other words, the factor $r \equiv c\Delta t/\Delta x$ in (6.6) was exactly 1. This was no accident. It turns out that when $r = 1$ the higher-order terms that are dropped in deriving (6.6) are largely (but not completely) canceled out. If we were to repeat the calculation with a smaller time step and the same value of Δx , the results would actually be *less* accurate; we will illustrate the nature of such errors shortly. However, reducing the spatial step size while keeping Δt fixed leads to a real disaster. For example, if we were to cut Δx in half but keep Δt fixed so that $r = 2$, the algorithm becomes *unstable*. In this case a wavepacket like the one in Figure 6.2 would grow with rapidly with time, and the calculated string displacement would *diverge*!

There are some numerical lessons to be learned here, and a full discussion of these issues could easily fill this chapter. We refer readers to the sources in the references (as well as the exercises) for further information, and will make only a few general observations here. Let us first consider what is so special about $r = 1$ and why larger values of r , that is, smaller values of the spatial grid size Δx , lead to numerical instabilities. Imagine that our string is initially at rest with $y = 0$ everywhere. We then arrange things so that at time step $n = 1$ the value of y at one of the spatial grid sites, $i = i_0$, is given a nonzero value, while $y = 0$ at all of the other sites. You can think of this as a very special way of plucking the string. The finite difference expression, (6.6), tells us that at time step $n = 2$ this "disturbance" will be felt at sites $i_0 \pm 1$. The string displacements $y(i_0, 2)$ and $y(i_0 \pm 1, 2)$ will be nonzero, but the displacements at all other values of i will still be zero. Similarly, at time step 3, the disturbance will propagate to sites $i_0 \pm 2$, etc. With our numerical scheme the disturbance will propagate precisely one spatial step for each time step. The velocity of the disturbance is thus $\Delta x/\Delta t$ and is, therefore, limited by our choice of step sizes. It is not possible for our algorithm to yield a disturbance that moves more than one spatial element per time step. Now, we know that a wave on the string will move at a speed of c , so if our numerical solution has any hope of describing such a wave, it must be able to handle disturbances that travel at this speed. However, if the step sizes are such that $\Delta x/\Delta t < c$, our algorithm will simply be incapable of yielding a wave moving at the proper velocity. This corresponds to $r > 1$, and so in this case our numerical algorithm must fail. This failure is manifested by the instability mentioned above; it turns out that a simulation with $r > 1$ yields waves whose amplitudes diverge rapidly with time. We will leave a study of this behavior to the exercises.⁸

This simple argument involving propagation velocities explains why our algorithm requires $r \leq 1$. To analyze the performance of the algorithm for this range of r requires a bit more work, which can be reviewed in the sources in the references. There it is shown that when $r = 1$ the higher-order terms neglected in deriving (6.6) are (as we already noted) largely canceled. Thus, as a practical point, it is

⁸This instability is analogous to the one discussed in connection to the SOR method in Chapter 5.

usually better to choose the step sizes such that $r = 1$ rather than, say, $r = 0.5$. However, values of r smaller than 1 still yield stable solutions, as we will illustrate in our next example.

Consider a string that is composed of two segments on which the waves have different velocities. This would arise, for example, if we tie together two strings of different thicknesses (i.e., different values for the mass per unit length). If we were dealing with electromagnetic (light) waves, this would correspond to light traveling from a material with one index of refraction into a material with a different index. We can model this problem with (6.6) and the program developed above by letting c be dependent on position. To be specific, we consider a string 1 m long described by a wave velocity c_1 for $0 \leq x \leq 0.5$, and c_2 elsewhere. This means that r in (6.6) has different values on the two halves of the string.

Some results from such a simulation are shown in Figure 6.3. Here the left-hand part of the string (the part to the left of the dotted line) has a wave velocity c_1 , which is a factor of 2 larger than the velocity on the right. Hence, we have a light string on the left and a heavy string on the right (we won't worry about the knot in the middle). The wavepacket is initiated on the left side of the string, and immediately splits into left- and right-going components. When the right-going wavepacket hits the interface with the heavy string, it is partially reflected and partially transmitted. The reflected part is inverted, since the heavy string acts a little like a fixed boundary. The transmitted part is not inverted and is seen to travel more slowly on the heavy string, which is expected since $c_2 = c_1/2$. Careful examination of the shape of the wave packet in the heavy string shows a small "blip" trailing the main part of the packet. It turns out that this is due to the numerical errors, mentioned above, that arise when $r < 1$ (we will justify this claim in the exercises). While we are able to use $r = 1$ for the left part of the string, the algorithm forces us to then use $r = 1/2$ on the right side of the system, so these numerical errors cannot be avoided (at least with our simple algorithm).⁹ Nevertheless, these errors are not serious and our wave simulations give a nice picture of wave transmission and reflection at a boundary.

The program we have developed for wave propagation can easily be extended to study a variety of other wave phenomena. Examples include reflection from a free end, superposition of waves, and standing waves. Some of these are explored in the exercises.

EXERCISES

- 6.1. Write a program to simulate wave motion on a string with free ends. Do this by either using boundary conditions that always give the ends of the string the same displacement as the points that are one spatial unit in from the ends, or by employing (6.7) (these two approaches are equivalent). Study how the waves are reflected from the ends of the string and compare the results with the behavior with fixed ends. You should find that the reflected wave packets are not inverted.
- 6.2. Compare the behavior calculated with the algorithm (6.6) different values of r . Observe the instabilities associated with $r > 1$ and the errors that arise when

⁹Of course it is possible to retain $r = 1$ on both parts of the string if we are willing to use different values of Δx on the two parts. See Problem 6.6.

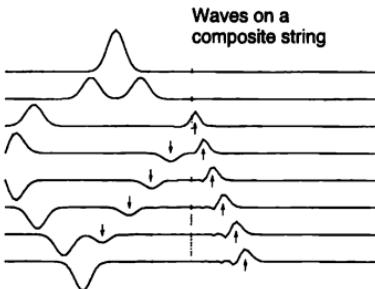


FIGURE 6.3: Waves propagating on a composite string with fixed ends. The string was 1 m long, with $c_1 = 300$ m/s on the left half of the string and $c_2 = 150$ m/s on the right half (the dotted line shows where the change in c occurs). We used $\Delta x = 0.01$ m and $\Delta t = \Delta x/c_1$. The arrows indicate pieces of the wavepacket that are reflected by, and transmitted through, the point where the propagation velocity changes.



FIGURE 6.4: Realistic excitation profile for a guitar string that is plucked. The initial string displacement consists of two straight lines, which join with different slopes at the excitation (plucking) point.

$r < 1$. It is especially instructive to compare the reflection of a wavepacket from a fixed end for different values of $r \leq 1$. The small errors that are introduced when $r < 1$ are then clearly visible as extra “bumps” (like the ones noted in connection with Figure 6.3) that are generated during each reflection.

- *6.3. Set up a wavepacket that doesn't split up into two pieces (as we observed with our Gaussian packet in Figure 6.2), but moves uniformly in one direction. *Hint:* In the simulations discussed so far in this section we have assumed that the string is at rest prior to $t = 0$. In order to construct a single wavepacket that does not immediately split you will have to properly specify both the initial displacement and velocity of the string.
- 6.4. Study the propagation of wavepackets with other shapes. For example, a guitar string that is plucked has an initial profile like that shown in Figure 6.4. Calculate how this excitation splits and moves with time.
- *6.5. Investigate the motion of a string for which one end is held fixed, while the other is made to oscillate. Do this by letting the string element at one end

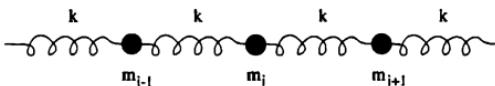


FIGURE 6.5: Model for compressional waves on a rod. Each small piece of the rod is modeled as a small mass m , connected to adjacent masses by springs with force constants k

move according to $y(i=0) = A \sin(\omega t)$. You should find that this generates a wave that propagates toward the opposite end of the string. This wave is then reflected, and it interferes with the initial wave. Confirm that the frequency of this wave ($f = \omega/2\pi$) and its wavelength are consistent with the parameter c in the wave equation. Also determine the values of ω that lead to standing waves.

- 6.6. An important feature of a linear equation is that the sum of two solutions is also a solution. One consequence of this is that two wavepackets will travel independently of each other. An especially clear way to demonstrate this is to set up a string with an initial profile such that there are two Gaussian wavepackets, located at different places on the string. These wavepackets (or components of them) may then propagate toward each other and collide. Show that the wavepackets are unaffected by these collisions. That is, show that two such wavepackets pass through each other without changing shape or speed.

- *6.7. The origin of the wave equation (6.1) is different for different types of waves. For electromagnetic waves it is a result of Maxwell's equations, while for waves on a string it follows from Newton's second law. One way to derive the wave equation for compressional waves on a solid rod is as follows. Model the rod as a collection of particles of mass m , as shown in Figure 6.5. Each mass is a small piece of the rod and is connected to adjacent pieces (masses) by springs. Let the position of mass m_i be y_i . The springs are unstretched and uncompressed when the masses are at their equilibrium positions, which we will take to be $y_i = 0$. Each mass experiences forces from the two springs to which it is connected (don't worry about the ends). The force from the spring to the left is just $F_i(\text{left}) = -k(y_i - y_{i-1})$ where k is the force constant of the spring, and the force from the spring on the right has a similar form. Write Newton's second law for mass m_i and show that it can be cast in the form (6.1), with $x = i\Delta x$ where Δx is the distance between masses. Find the value of c in terms of m , k , and Δx . *Hint:* The finite difference form for a second partial derivative will be useful. Also, note that this derivation assumes implicitly that the displacements y_i are all small compared to Δx .

- *6.8. Perform a simulation of waves on a composite string, as in Fig. 6.3, but now use different spatial step sizes Δx on the two parts of the string, so that the factor $r = 1$ on both sides of the string. Study how this removes the numerical artifacts noted in Fig. 6.3. *Hint:* since you will have different spatial steps on the two parts of the string, you must be careful in your treatment of the forces at the interface

FREQUENCY SPECTRUM OF WAVES ON A STRING

Vibrating strings are used in a variety of musical instruments, which makes it interesting to consider the frequency spectrum of waves on a string. We have already employed Fourier analysis of a time-dependent signal in connection with our study of oscillatory motion in Chapter 3, and the same sort of spectral analysis can be used to examine waves on a string.¹⁰ Ideally we might want to perform such an analysis in connection with a model of a guitar or piano, in which case we would analyze the sound wave caused by plucking (in the case of a guitar) or striking (for a piano) a string. However, for a *real* musical instrument this is a very complicated process, since the sound is actually produced by the vibration of a wooden plate connected to the string.¹¹ A full treatment of the vibrations of such a plate coupled to a vibrating string is too involved to include here, so we will now take a simplified approach. (We will consider to a somewhat similar problem in Chapter 11.)

We consider a string with fixed ends and imagine that we record the displacement of a particular point on the string as a function of time. An example of such a signal is given in Figure 6.6, where we show some results calculated using the routine `propagate` that we developed in Section 6.1. We see a series of pulses, each of which is produced when one of the wavepackets observed in Figure 6.2 moves past our observation point. Here the observation point, that is, the place at which the string position was recorded, was 5 percent from one end of the string. The pulses alternate in sign, since the wave is inverted when it is reflected. The pulses occur in closely spaced (inverted) pairs because the observation point is very near to one end.

As was discussed in Chapter 3, and in more detail in Appendix C, this signal can be decomposed into Fourier components. That is, we can write the signal as a sum of sines and cosines with different frequencies and amplitudes. The results of such an analysis can be displayed in several ways. For this problem it is most convenient to consider the power spectrum. This is just the sum of the squares of the Fourier components (the sine and cosine components) at each frequency, as a function of the frequency.¹² The power spectrum calculated for the signal in Figure 6.6 is shown in Figure 6.7, where we find a series of regularly spaced peaks.

These peaks can be understood in terms of the standing waves that are found for a string with fixed ends, as illustrated in Figure 6.8. The standing wave with the longest wavelength is the one shown at the top, and has a wavelength $\lambda_1 = 2L$, where L is the length of the string. The other standing waves have wavelengths of L , $2L/3$, etc. These standing waves can be thought of as the basic *spatial* Fourier components of the string motion, just as the time-dependent sine waves discussed in Chapter 3 and Appendix C are the basic Fourier components in a frequency analysis. Each of these standing waves has a specific wavelength, which means

¹⁰ Readers who are not familiar with Fourier analysis should examine Appendix C before attempting this section.

¹¹ The body of the guitar or the soundboard in a piano.

¹² As a first approximation, the sound wave produced by a vibrating string will have an amplitude proportional to that of the string (there will also be a frequency-dependent coefficient of proportionality that we won't worry about here). With these approximations, the power carried by the sound wave is proportional to the power spectrum considered here (hence the name).

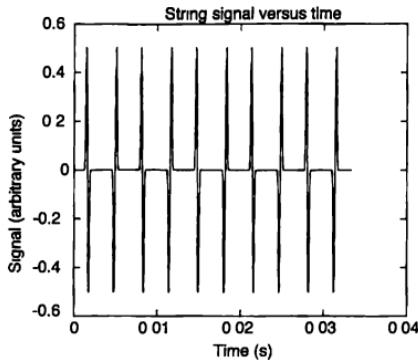


FIGURE 6.6: Signal from a vibrating string. The string was excited with a Gaussian initial pluck centered at the middle of the string, and the displacement a distance 5 percent from one end was recorded. The other parameters in the simulation were the same as in Figure 6.2.

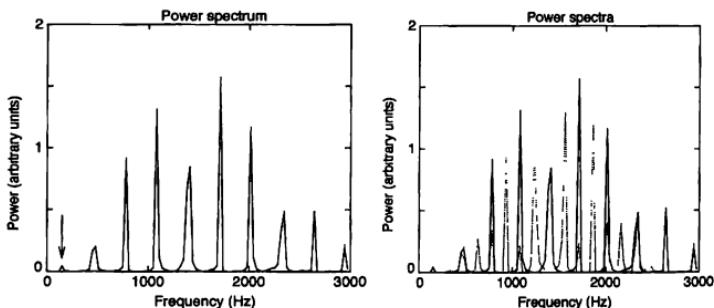


FIGURE 6.7: Left power spectrum of the signal in Figure 6.6. Here the string was excited at the center. The arrow indicates the peak corresponding to the fundamental frequency of the string, which in this case is 150 Hz. Right the dotted curve is the power spectrum obtained when the string was excited 5 percent from its center. For comparison, the solid curve shows the power spectrum found when the string was excited at the center (this is the result shown on the left).

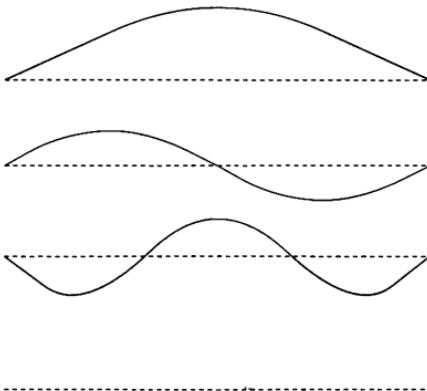


FIGURE 6.8: Solid curves show possible standing waves on a string with fixed ends. The dotted curve at the bottom shows one choice for the initial wavepacket, a Gaussian function centered at the middle of the string.

that they also have a specific *frequency*, since the wavelength and frequency of a wave (even a standing wave) are related by $\lambda f = c$. The allowed wavelengths are $\lambda = 2L/m$, where m is an integer, so the possible frequencies are just $mc/(2L)$. This is why the peaks seen in the spectral analysis in Figure 6.7 occur at regularly spaced frequencies. Each of these peaks corresponds to a particular value of m .

For the parameters used in this simulation, the lowest frequency peak in the spectrum should occur at $c/(2L) = 150$ Hz, and we do, indeed, find a peak at this frequency (albeit a small one), as shown on the left in Figure 6.7. There are also substantial peaks at 450, 750, 1050 Hz, etc., so there is certainly a very regular pattern. However, our standing wave argument suggests that the peaks should be spaced by $c/(2L)$, but the “peaks” at 300, 600, 900 Hz, etc., are either missing or extremely small compared to the adjacent peaks. To understand this behavior we need to consider how the string is excited, that is, how the initial wavepacket is setup. As we have already mentioned, the standing waves are the spatial Fourier components for our problem. We can, therefore, write our initial wavepacket, the Gaussian curve at the bottom in Figure 6.8, as a sum of these standing waves. In fact, for our ideal string the magnitudes of these components do not change with time, so once we have determined the Fourier components for the initial wavepacket, we also have them for all later times. The wave moves, of course, because the different Fourier components have different frequencies. However, the relative *magnitudes* of these components do not change with time (we will develop these ideas further in Section 6.4).

Now let us consider in a little more detail the spatial Fourier components for our initial Gaussian wavepacket. This wavepacket is located at the center of the string and is thus *symmetric* with respect to the center of the string. This symmetry will be preserved in the Fourier components. That is, the only Fourier components with significant amplitudes will be those that are nonzero at the center of the string and are symmetric about the center. Hence the Fourier components, which have zero amplitude (nodes) at the center, will not be present in our initial wavepacket. The standing waves that are not present have wavelengths L , $L/2$, $L/3$, ... and for the particular simulation in Figure 6.7, their frequencies are 300, 600, 900, ... Hz. This is why these frequencies are missing in our spectrum.

The argument, then, is that the *symmetry* of the initial wavepacket causes certain frequencies to be suppressed. To convince you that this is indeed the case, we have repeated the simulation with an initial wavepacket whose peak is located 5 percent away from the center of the string, and the results are shown for comparison on the right in Figure 6.7. The peaks are now (almost) all spaced by 150 Hz, as expected for the string in this simulation. The only peaks missing here are those with nodes located 5 percent away from the center of the string. The message of this exercise is that the frequencies that are excited will be those whose spatial modes have the same *symmetry* as that of the initial wavepacket.

It turns out that a situation of this kind occurs in pianos. When you press a key on a piano, a complicated mechanical linkage is engaged that sets a "hammer" into motion. These hammers are essentially felt-covered mallets that strike one or more steel strings, the vibrations of which lead to the desired musical tone. The lowest frequency vibration of the string(s) is referred to as the frequency of the note, although as we have seen, many higher frequencies will also be generated. These harmonics are, for an ideal string such as the one considered here, integer multiples of the lowest, or fundamental, frequency.¹³

If we were to perform a Fourier analysis of the sound signal associated with a particular note, we would find a spectrum similar to that seen in Figure 6.7. There would be many peaks at regularly spaced frequencies. While the fundamental frequency is just the frequency of the first peak, the harmonics play an absolutely crucial role. It is not hard to arrange for a piano, a guitar, and a violin to all play the same note, that is, a note with the same fundamental frequency. However, most people would have no trouble in distinguishing the different instruments. This is because each produces a distinctive pattern of harmonic components. The relative strengths of the harmonics, and how they vary with time during the course of the note, are different for different instruments. Our ear (really our brain) makes use of these differences to identify the instruments.

Taking this argument one more step, when a piano designer assembles plans for a piano, the placement of the hammers is a crucial issue. This is because the striking point (the place where the hammer contacts the string) affects the spectrum of the vibrations. Nowadays, in most pianos the strike points are approximately 1/8 of the way from one end of the string. Using the arguments connected with Figure 6.8, this implies that vibrations that have a node at this location will not be

¹³We will consider these harmonics in more detail in Section 6.3.

excited by the hammer. Piano designers, and listeners, have apparently decided that minimizing these particular harmonics leads to the most pleasing tone. Precisely why that is the case is not obvious.

EXERCISES

- 6.9. Perform a spectrum analysis of waves on a string in which one end is free to move while the other is held fixed. Assume an initial Gaussian wavepacket located 40 percent from one end. Explain the peaks in the spectrum in terms of the allowed standing waves. Note that because the ends are free, these standing waves will be different from those found with fixed ends (Figure 6.8).
- 6.10. Devise an initial string displacement that gives rise to a vibration in which the only frequency present is the one corresponding to the Fourier component with a wavelength of $2L/3$, where L is the length of the string.
- 6.11. Devise an initial string displacement that gives rise to vibrations in which the vibrational frequencies are all higher than the frequency of the Fourier component with a wavelength of $2L/5$, where L is the length of the string.
- 6.12. Gaussian initial string displacements are convenient for the calculations of this section, but are not very realistic. When a real string, such as a guitar string, is plucked, the initial string displacement is more accurately described by two straight lines that start at the ends of the string (we assume fixed ends) and end at the excitation point, as illustrated in Figure 6.4. Compare the power spectrum for a string excited in this manner with the results found above for a Gaussian initial wavepacket.
- 6.13. Consider the power spectra for waves on a string as a function of where the string vibration is observed, x_0 . Some standing waves will have nodes at x_0 , and the corresponding frequencies will not appear in the power spectrum, even though these waves may be excited on the string as a whole. Demonstrate this by comparing spectra obtained for values of x_0 , which are 10 percent, 40 percent, and 50 percent from the end of the string. *Hint:* It is helpful in each case to consider initial Gaussian excitations located at different spots along the string.
- *6.14. The peaks in the power spectra in Figure 6.7 vary greatly in size, with the peaks at the lowest and highest frequencies being much weaker than the peaks at intermediate frequencies ($\sim 1000\text{--}2000\text{ Hz}$). Explain qualitatively why this is the case

6.3 MOTION OF A (SOMEWHAT) REALISTIC STRING

In our studies of vibrating strings we have so far treated only the case of an ideal string. Of course, real strings will have some frictional losses (damping), as well as some stiffness. To get a feeling for how these affect the vibrations, we will in this section consider the effect of stiffness (leaving damping to the exercises). The phenomena of damping and stiffness are not limited to waves on a string. For example, the same physics (and a similar equation) is encountered when dealing with electromagnetic waves in a lossy medium or sound in the atmosphere.

The physical origin of stiffness in the case of waves on a string can be understood as follows. In deriving the wave equation, (6.1), we assumed that the string is perfectly flexible. This means that the only force on a particular segment of the string is due to the tension, and this force is directed along the string. While strings

can be made to be very close to this ideal limit, there will always be a stiffness force that opposes bending of the string. This is the case, for example, with a thick metal bar; the vibrations of such an object are very different from those of a flexible wire.

Stiffness can be added to the model by adding a term¹⁴ to our original wave equation (6.1)

$$\frac{\partial^2 y}{\partial t^2} = c^2 \left(\frac{\partial^2 y}{\partial x^2} - \epsilon L^2 \frac{\partial^4 y}{\partial x^4} \right), \quad (6.9)$$

where ϵ is a dimensionless stiffness parameter, and L is the length of the string. The fourth partial derivative can be written in a finite difference form, using the same approach that we used to obtain the second partial derivative in Chapter 5. The result is

$$\frac{\partial^4 y}{\partial x^4} \approx \frac{y(i+2,n) - 4y(i+1,n) + 6y(i,n) - 4y(i-1,n) + y(i-2,n)}{(\Delta x)^4}. \quad (6.10)$$

The fourth derivative thus involves the value of y at distances up to ± 2 spatial units away from the point where the derivative is centered. Inserting this into (6.9) leads, after a little algebra, to

$$\begin{aligned} y(i,n+1) &= [2 - 2r^2 - 6\epsilon r^2 M^2] y(i,n) - y(i,n-1) \\ &\quad + r^2 [1 + 4\epsilon M^2] [y(i+1,n) + y(i-1,n)] \\ &\quad - \epsilon r^2 M^2 [y(i+2,n) + y(i-2,n)], \end{aligned} \quad (6.11)$$

where $M = L/\Delta x$ is the number of spatial units along the string. Equation (6.11) thus yields the string displacement at time step $n+1$ in terms of the displacement at previous time steps. Extending our program to handle this equation is straightforward; we need to modify one line in the `propagate` routine to include the terms proportional to ϵ . There is a slight complication, however. Equation (6.11) involves the displacement at sites ± 2 units away from the site in question, whereas without stiffness only sites displaced by ± 1 unit were involved. Now, when the `propagate` routine approaches the end of the string, that is, the last mobile section of the string, we need to know the displacement at the end and also the displacement one unit *past* the ends. The simplest way to deal with this problem is to assume that the ends of the string are “hinged.” That is, we take the displacement at each end to be zero and assume that there are phantom locations one unit beyond the ends which have displacements that are opposite the displacements at locations one unit inside the ends. This is illustrated in Figure 6.9. Here the real string terminates

¹⁴The origin of the 4-th order derivative term in (6.9) is connected with the potential energy of the string due to bending. For an ideal, flexible string, the potential energy is all associated with the stretching of the string, and is of the form $T \left(\frac{\partial y}{\partial x} \right)^2$. The potential energy associated with bending has the form $Y \left(\frac{\partial^2 y}{\partial x^2} \right)^2$, where the proportionality constant Y is proportional to one of the elastic constants of the string material. Those who are familiar with the Lagrangian approach to mechanics should see that one can insert these potential energy terms into the Lagrangian density L , and thus derive (6.9) from the Euler-Lagrange equation. More detail on this term is given by Chaigne and Askenfelt (1994).

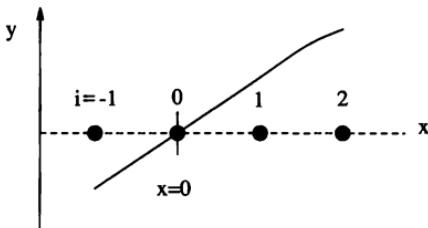


FIGURE 6.9: Hinged boundary conditions at the end of a string. The "real" string ends at $x = 0$ (spatial step $i = 0$) The displacement at $i = -1$ is only imaginary and is taken to be $-y(i = +1)$.

at $x = 0$ (corresponding to spatial step $i = 0$ and a fixed end), but the program allows it to effectively extend to $i = -1$. The displacement of this phantom point is $y(i = -1) = -y(i = +1)$. This simple treatment of the boundary conditions will be adequate for our work and is actually the procedure used in detailed modeling of piano strings (see the references for more on this).¹⁵

The simulations of our stiff string can be carried out as before. Here we will focus our attention on an issue that is relevant to musical instruments. Figure 6.10 shows the power spectra calculated for several cases that correspond roughly to piano strings (this note would be in the upper bass region of the keyboard).¹⁶ Here we compare results for three separate simulations; a perfectly flexible string (stiffness = $\epsilon = 0$), a string with a stiffness similar to that found in a good grand piano ($\epsilon = 1 \times 10^{-5}$), and a string with a somewhat higher stiffness. Overall, the spectra are very similar; the peaks are regularly spaced and the spacing is very close to the expected value, which is 150 Hz for this string. However, careful examination of the region above about 2000 Hz shows that the peaks for the stiff strings are shifted systematically to higher frequencies when compared with those of the ideal string.

We have already seen that for an ideal string the frequencies of the standing waves are spaced by $c/(2L)$, independent of the frequency. However, for the stiff string this separation increases as we move to higher frequencies. This effect is known as *dispersion* and occurs because the stiffness causes the wave speed to now be *frequency dependent*. Physically this can be understood as follows. The higher

¹⁵You might object that such hinged boundary conditions are not very realistic. In our defense we have the following comments. First, we could assume some sort of clamped boundary conditions, which would amount to conditions on the spatial derivatives of y . The most realistic way to do this would depend on precisely how the string is fastened to its support. Second, in many instruments hinged boundary conditions seem to be a reasonable approximation (see Chaigne and Askenfelt [1994]). Third, if we want to construct a truly accurate model of a real musical instrument (or other type of string), life can quickly get very complicated.

¹⁶Note that in this simulation we used $r < 1$. It turns out that the numerical stability condition involving r is altered by the stiffness term in the wave equation. Values of r less than unity are now required for stability; this is discussed in the references.

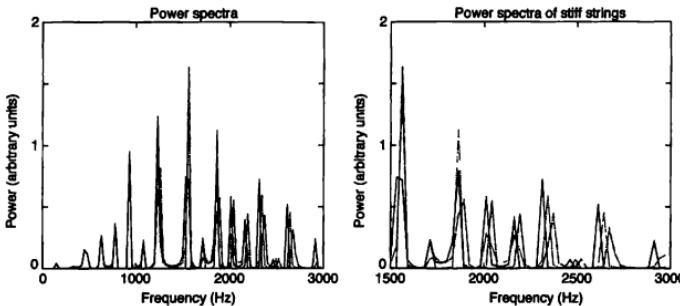


FIGURE 6.10: Power spectra for “realistic” strings. Solid curves no stiffness; dotted curves, $\epsilon = 1 \times 10^{-5}$; dot-dashed curves, $\epsilon = 2 \times 10^{-5}$. In all cases the string was excited 5 percent off center, the displacement 5 percent from one end was analyzed, and we used $\Delta t = \Delta x/(4c)$ to ensure stability for all values of ϵ employed. The plot on the right shows the high frequency region on an expanded scale.

frequencies correspond to standing waves with shorter wavelengths.¹⁷ In order for it to move with a shorter wavelength, the string must be bent more severely. A stiff string resists bending and this extra resistance makes the string effectively tighter, leading to a higher wave velocity and a higher frequency.

Thus, for a real string the “harmonics” are *not* arranged harmonically. Indeed, they are usually called *partials* for this reason. The effect seen in Figure 6.10 is a significant effect in pianos and other stringed instruments. The frequencies of the partials deviate by a small amount from the ideal harmonic spacing, and this deviation contributes to the characteristic piano sound. Perhaps most importantly, it affects chords since these involve notes that, in the ideal case, are harmonically related. For example, a chord might involve two notes for which the second harmonic of one is very close in frequency to the third harmonic of the other. For a real string these harmonics, that is, partials, will not have quite the same frequencies, and this will affect the way the chord sounds.¹⁸

There are three other points that we should mention regarding this simulation. First, the shift of the partials to higher frequencies (higher than pure harmonics) is known as *octave stretching*, and is well known to piano designers and technicians.

¹⁷It is important to note that the concepts of normal modes and superposition, which are at the heart of these arguments, still apply here since even with the stiffness term included the wave equation (6.9) is linear.

¹⁸Our discussion here is a bit oversimplified, since we are ignoring how the strings are tuned (i.e., equal versus just temperament), etc. However, these complications do not change our basic point.

This term arises in the following way. Two notes that are an octave apart would, in the perfectly harmonic case, differ in frequency by precisely a factor of 2. For a well-tuned piano the second partial of the lower note will have approximately the same frequency as the fundamental (first partial) of the upper note. However, because of the shift produced by the string stiffness, the second partial of the lower note will be slightly more than a factor of 2 higher than its fundamental frequency. Hence, the effective size of an octave separation is stretched slightly and is greater than the value of 2 that would be found for a perfectly flexible string.

A second point concerns piano design. The effects of string stiffness depend on the length of the string. Essentially, this is because a short string will (other factors being similar) need to bend more than a long one. For this reason the shifting of the partials away from the harmonic "ideal" is greater for shorter strings. Since smaller pianos have shorter strings, the effect is generally largest in small pianos, and this is one reason why a large piano is preferred over a small one.¹⁹ Interestingly, however, we would *not* like to see octave stretching eliminated altogether. This stretching contributes an essential element to the characteristic piano sound. After all, in an organ there is no octave stretching (there is no problem with stiff strings in that case), and most of us have no problem discerning an organ from a piano.

Our third point is a numerical one. We saw earlier that when solving the wave equation it is best from a numerical standpoint, to choose the parameter r to be unity, as this led to the best and most stable solutions. However, when the wave equation is generalized to include stiffness, the value of r at which our algorithm becomes unstable is reduced. Accordingly we chose $r = 1/4$ in the simulation of Figure 6.10. The problem of numerical stability is discussed further in Chaigne and Askenfelt (1994).

EXERCISES

- 6.15. Derive the finite difference approximation for the fourth partial derivative (6.10).
- *6.16. Perform the calculations described in this section. One interesting possibility is to compare the size of the octave stretching, that is, the magnitude of the deviations from a purely harmonic spectrum, for short (treble) and long (bass) strings. The relevant string parameters for a good grand piano are given in Table 6.1.
- *6.17. None of the wave models we have considered so far include friction (i.e., damping), hence none of these waves would decay away with time. To make our simulations more realistic, add damping to the wave equation. This can be accomplished by adding a term $-2b(\partial y/\partial t)$ to the right-hand side of (6.9), corresponding to a frictional force proportional to the velocity of the string. Derive a new difference equation that includes this term and calculate the wave pattern as a function of time. Warning: This is not a trivial calculation, but it is

¹⁹Note that the stiffness parameter ϵ in (6.9) was defined as a dimensionless parameter by explicitly separating out the factor L^2 . The argument given in footnote 14, where we discussed this parameter in terms of a Lagrangian density, also leads to the identification of $\epsilon L^2 = Y/T$ where Y is an intrinsic, material-specific measure of stiffness (called the Young's modulus) and T is the tension. Y has the dimensions of energy times length, so ϵ is a measure of the effect of stiffness normalized per unit length. Thus, contrary to the way it may first appear from this equation, the effect of stiffness is actually smaller with a longer string (all else being the same).

TABLE 6.1: Some parameters describing the properties of a strings in a typical grand piano. The note $C4$ is middle C , while $C2$ is two octaves lower and $C7$ is three octaves higher in frequency. The parameter b is associated with damping, as discussed in problem 6.17. After Chaigne and Askenfelt (1994).

Note	f (Hz)	L (m)	c (m/s)	ϵ (unitless)	b (s^{-1})
$C2$	65.4	1.9	250	7.5×10^{-6}	0.5
$C4$	262	0.62	330	3.8×10^{-5}	0.5
$C7$	2093	0.09	380	8.7×10^{-4}	0.5

straightforward; for some background see Chaigne and Askenfelt (1994). You can also listen to your calculated signals if you have access to the appropriate hardware. They don't sound too bad, but they aren't quite the same as the real thing.

6.4 WAVES ON A STRING (AGAIN): SPECTRAL METHODS

In this section we reexamine the problem of waves on a string using a different computational approach. This approach relies heavily on ideas associated with the Fourier transform and is usually placed under the general heading of "spectral methods." These methods avoid the use of a finite difference expansion and do most of their work in Fourier space instead.

We have already seen several instances in which it is useful to decompose a time-dependent signal into its Fourier components. A typical example was the displacement of a vibrating string at a particular location along the string, $y(x, t)$. We learned how to use a Fourier transform to write such a function in the frequency domain as a collection of sines and cosines with different frequencies. This is a central theme of Appendix C, and was an essential computational tool in Section 6.2. As we have hinted in our discussion of standing waves on a string, the concept of Fourier analysis is not limited to time-dependent functions, but can be applied in a natural way to *space-dependent* functions, as well. When discussing the string displacement *at a particular time*, it is useful to consider a Fourier decomposition in term of sines and cosines that are functions of *space* instead of time. This is illustrated in Figure 6.11, which shows at the top a hypothetical string displacement given by our usual Gaussian form (6.8). This function can be written as a sum of sines and cosines, the largest of which are plotted in Fig 6.11. We have already seen how to use Fourier techniques to write a function in the time domain. Here we have used the same techniques to write the initial string displacement $y_0(x)$ in terms of sines and cosines that are functions of x . Mathematically this decomposition has the form

$$y_0(x) = \sum_i \left[A_i \sin\left(\frac{2\pi x}{\lambda_i}\right) + B_i \cos\left(\frac{2\pi x}{\lambda_i}\right) \right], \quad (6.12)$$

where the wavelengths λ_i determine the spatial variations of our sines and cosines. The values of λ_i for the Fourier components of our Gaussian displacement function are given next to the components in Figure 6.11.

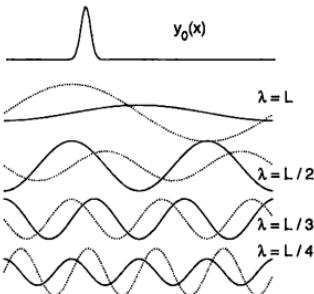


FIGURE 6.11: Top: string displacement at $t = 0$, given by the function $y_0(x)$ in (6.8); bottom: the first few spatial Fourier components of $y_0(x)$. The solid lines are the cosine components, while the dotted lines are the sine components. The corresponding Fourier wavelengths are indicated. The string was 1.0 m long, the spatial step size was $\Delta x = 0.01$ m, and there were $N = 128$ points used in the FFT.

This expansion is completely analogous to the Fourier sums involving time-dependent functions that we have employed in our work with the pendulum and the vibrating string. The spatial transform we need here can also be computed using the fast Fourier transform (FFT) algorithm. Indeed, that is how we calculated the components in Figure 6.11. Computationally the problem is essentially the same as when dealing with a time-dependent function. You will recall that for functions of time we had N values of our signal that were obtained at points spaced in time by Δt . The FFT then yielded the Fourier components at frequencies $n/(N\Delta t)$, where the integer n varied from 0 to $N/2 - 1$. For a function of space we require values of the signal at N points that are evenly spaced in space. For a spatial step size Δx the FFT then gives the Fourier components at wavelengths $\lambda_n = N\Delta x/n$, where n runs from 0 to $N/2 - 1$. Figure 6.11 shows the spatial Fourier components calculated using the FFT for the Gaussian pluck plotted at the top of the figure. In this case we employed $N = 128$ points in our FFT, so the Fourier decomposition yielded the sine and cosine components at $N/2 = 64$ wavelengths. Figure 6.11 shows only the first few components, that is, those with the longest wavelengths. We will consider how the magnitudes of these components vary with λ in a moment.

Now that we have split up our signal into its Fourier components, the next problem is deciding what to do with them. Each component has a particular value of λ . For an ideal string the waves have a speed c , so each of these components will also have a particular frequency, $f_n = c/\lambda_n$. Suppose that we started the string off with a shape that was given exactly by one of the Fourier components in Figure 6.11, say the sine component with $\lambda = L$. The string would then vibrate with the frequency of this Fourier component, $f = c/L$. Each point on the string

would undergo simple harmonic motion about $y = 0$ with this frequency, and the amplitude of vibration would be the initial displacement of the string²⁰ at that value of x . The string would be just a collection of simple harmonic oscillators, one for each point on the string, all oscillating at the same frequency, f . If instead, the string were given an initial displacement with a shape corresponding to a different Fourier wavelength, say $\lambda = L/3$, the frequency would then be $f = c/\lambda = 3c/L$, but the basic behavior would be the same. Each point on the string would undergo simple harmonic motion at this frequency.

In general, the initial displacement of the string will not be so simple as to be given by only a single Fourier component. We have already seen that for our Gaussian pluck there are many components present. To describe this more complicated case we appeal to the principle of superposition, which tells us that for an equation that is *linear*, the sum of two solutions is also a solution. The wave equation for our problem (6.1) is, indeed, linear since it contains only factors proportional to the first power of y . It is easy to verify in this case by direct substitution that the sum of two or more solutions is also a solution. Now imagine that we have all of the Fourier components for our Gaussian initial shape. Each of these is a solution to the wave equation, so their sum is also. If we move forward in time, letting each component vibrate at its particular frequency, and then sum up the displacements corresponding to each component, we will again have a solution to the wave equation. This solution will, in fact, describe the way our string vibrates in response to the initial Gaussian displacement. We have thus solved the wave equation.

This approach is sometimes referred to as a spectral method and is most useful if the time dependence of each Fourier component is easy to calculate. This time dependence is required in order to effectively propagate each component forward in time. The solution is then the sum of these individual components at the time (or times) of interest. For our problem of a wave on a string, the mathematical expression of these ideas is as follows. If $Y_r(k)$ and $Y_i(k)$ are the real (cosine) and imaginary (sine) Fourier components for wavelength $\lambda_k = L/k$, each will have the corresponding frequency $f_k = c/\lambda_k$. The string displacement at time t will then be²¹

$$y(x, t) = \sum_{k=0}^{N/2-1} \left[Y_r(k) \cos\left(\frac{2\pi x}{\lambda_k}\right) + Y_i(k) \sin\left(\frac{2\pi x}{\lambda_k}\right) \right] \cos(2\pi f_k t). \quad (6.13)$$

At $t = 0$ the factor $\cos(2\pi f_k t)$ is unity and we simply have the sum of our original Fourier components. For $t > 0$, each component vibrates according to its individual frequency f_k .

To illustrate how this method works in practice we reconsider the problem of a vibrating string. We have already described in Figure 6.11 the FFT of an initial displacement given by a Gaussian function. Performing the sum (6.13) gives the results for a vibrating string shown in Figure 6.12. The behavior is seen to be

²⁰This assumes that the string was initially at rest.

²¹We assume here that the string is initially at rest. To allow for a nonzero initial velocity, a phase factor would have to be added to the last (time-dependent) cosine factor.

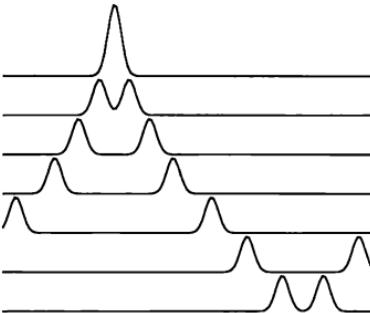


FIGURE 6.12: Solution for a vibrating string obtained using the spectral method. The top curve shows the string at $t = 0$, while the other traces (moving down the plot) show the string profile at progressively later times. Note that when the left traveling pulse reaches the end of the string (in the plot which is third from the bottom), it reappears at the far right (in the plot second from the bottom) due to the effectively periodic boundary conditions.

essentially identical to that which we obtained using finite difference methods to solve the wave equation in Figure 6.2. The initial Gaussian displacement splits into two pulses that propagate in opposite directions. The only difference between this solution and the finite difference result occurs when one of the pulses reaches the end of the string. In Section 6.1 we assumed that the ends of the string were held fixed, and this caused the pulses to be reflected (and inverted) when they reached the ends. However, these boundary conditions were not put into our spectral calculation. In fact, in using a Fourier decomposition we have implicitly assumed that the string has *periodic* boundary conditions. That is, we have assumed that the two ends of the string are connected as if the string formed a large closed loop. This assumption can be seen mathematically from the spatial periodicity of the sines and cosines in (6.13). For our calculated solution in Figure 6.12 we observe that when the left-going pulse reaches the (left) end of the string, it simply reappears at the right end, just as if the two ends were connected. Spectral methods can be extended to include such boundary effects, but that is a problem we will leave for the exercises.²²

Now that we have described spectral methods and illustrated them in one case, you might ask “Why bother?” Have we gained anything from this approach? The answer is “maybe.” For the particular calculation illustrated in Figure 6.12 we took $N = 128$, which means we had to calculate the displacements of 128 places along the string. Since we also had N Fourier components, evaluating (6.13) every

²²The spectral method depends on superimposing solutions to a linear equation, so it is most convenient when the boundary conditions are also “superimposable,” that is, when the boundary conditions are such that each Fourier component satisfies the same boundary conditions as the overall solution. This is indeed the case for the usual boundary conditions for a string; e.g., fixed ends, periodic boundary conditions, or free end (zero-slope) boundary conditions.

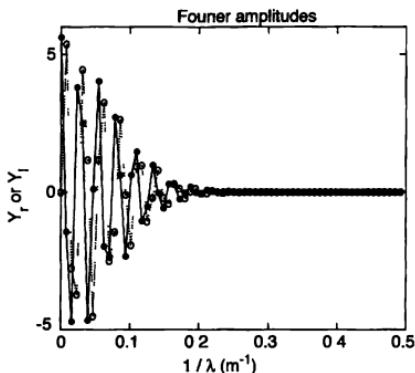


FIGURE 6.13: Fourier amplitudes for the Gaussian displacement in Figure 6.11. The solid symbols (and solid line) are the real (cosine) components, and the open symbols (and dotted line) are the imaginary (sine) components. Note that the horizontal axis is the *inverse* of λ , so the long wavelength components are on the left

time step was still a tall order (although we could imagine using an FFT to do this calculation). The utility of spectral methods arises in situations where some of these Fourier components are very small and can be neglected. It turns out that in many cases of physical interest the Fourier amplitudes decrease rapidly with N . Figure 6.13 shows these amplitudes for our Gaussian displacement, and we see that they become very small at short wavelengths. For λ^{-1} greater than about 0.2 m^{-1} the amplitudes are extremely small compared to the first few components. The reason these components are so small is that their wavelengths are *much* smaller than the scale over which the initial string displacement varies. Since these short wavelength components have very small Fourier amplitudes, we can, to a very good approximation, simply ignore them in computing the spectral sum (6.13). In fact, we already did this (without telling you) in Figure 6.12; in that calculation we used only half of the Fourier components.

This simplification is very common when dealing with spectral methods. We can often ignore all but the first few Fourier components, leading to a great simplification and speed up of the calculation. This will, of course, lead to some numerical errors in the final result. However, these errors are not as serious as some of the others we have encountered in this chapter, as they do not grow with time. In Figure 6.12 these errors were very small since the Fourier components that were neglected were themselves very small. However, we can only push things so far. To illustrate what happens when we use too few Fourier components, Figure 6.14 shows the results calculated using only $N/4$ and $N/8$ spectral components. The basic behavior is the same as that found with the more accurate calculation, Fig-

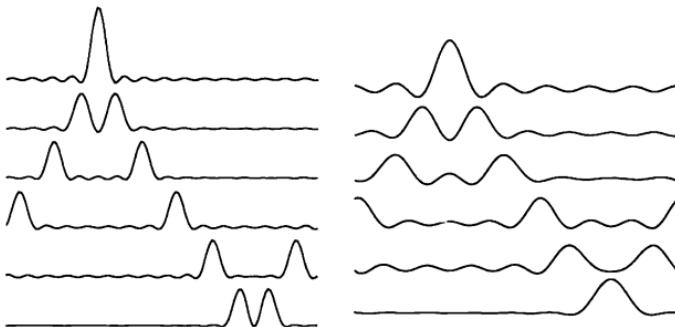


FIGURE 6.14: Effect of limiting the number of Fourier components in the spectral treatment of a vibrating string. Left: using $N/4$ spectral components, right: using $N/8$ spectral components. In both graphs the top curves show the string at $t = 0$, while the other traces (moving from top to bottom) show the string profile at progressively later times

ure 6.12. However, we now see undulations in the string well away from the two pulses. The Fourier components we have dropped here were small, but not completely negligible, and this is the kind of error introduced when they are omitted.

Spectral methods are potentially useful in problems that are linear, that is, in which the principle of superposition applies. Fortunately, this is the case in many problems in physics. We will employ them again in Chapter 10 in our dealings with quantum mechanics.

EXERCISES

- 6.18. Write a program that uses the spectral approach to solve the problem of waves on a string. Reproduce the results in Figure 6.12.
- 6.19. Use a broader initial wavepacket than we employed in the calculation in Figure 6.14, and show that the errors introduced by using a (relatively) small number of Fourier components are reduced. That is, show that for a given desired error we can use fewer Fourier components if the initial wavepacket is made broader. Explain why this is the case.
- *6.20. Build fixed-end boundary conditions into the spectral method. *Hint:* Use only Fourier components that automatically yield a displacement of zero at the ends of the string.

REFERENCES

- [1] A. Chaigne and A. Askenfelt, “Numerical simulations of piano strings. I. Physical model for a struck string using finite difference methods,” *J. Acoust. Soc. Am.* **95**, 1112 (1994); “Numerical simulations of piano strings. II. Comparisons with measurements and systematic exploration of some hammer-string

parameters," J. Acoust. Soc. Am. **95**, 1631 (1994). These papers describe state-of-the-art simulations of the motion of piano strings.

- [2] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapter 17 discusses the problem of numerical stability with regards to the wave equation.

Random Systems

7.1 WHY PERFORM SIMULATIONS OF RANDOM PROCESSES?

In all of our work to this point we have considered systems that are *deterministic*. These are systems that are described by some mathematical rule, such as a differential equation with boundary conditions, which has a uniquely determined solution. For example, with projectile motion the rules are the differential equations derived from Newton's second law. Given some initial conditions, the motion of the projectile is completely determined, that is, predictable, for all future times.¹ Similarly, for problems involving the electric potential the relevant rule is Poisson's equation. Once the boundary conditions on the electric potential and the charge distribution are specified, there is a unique solution for V .

In this chapter we consider a class of systems in which randomness plays a central role. These are called random or *stochastic* systems. Typically, these systems consist of a very large number of "degrees of freedom," which might be associated with particles (e.g., in a macroscopic sample of a liquid) or perhaps spins (in a piece of a ferromagnet). Randomness can then arise in several different ways. For example, it might be impossible to obtain complete knowledge of the positions and velocities of all of the particles. Or, our system may interact with a thermal reservoir in the complicated manner that is best described using probabilities and averages as familiar from statistical mechanics.² Hence, even though the "underlying" physics of the system may be deterministic, our incomplete knowledge may force us to resort to a statistical, i.e., stochastic, description. Fortunately, as we will see, a statistical description is often extremely useful, and indeed, most appropriate in these problems.

A typical stochastic problem is diffusion; this describes such important processes as the spreading of a drop of cream in your morning coffee. Let us consider what happens if we start with a cup of black coffee and then delicately place a drop of cream at the center. Now, assuming that you resist the urge to stir the coffee, the white mass of cream will slowly spread to fill the cup, and eventually the coffee will take on a uniform brownish color. At a more microscopic level this process would be described in the following way. The drop of cream consists of a large number of "cream particles."³ If we were somehow able to watch or follow one particular

¹In this sense, the time evolution of *any* system that obeys Newton's laws of mechanics is deterministic, although we have seen in Chapter 3 that such systems can still be chaotic. However, we will see below that even for nonchaotic (and deterministic) systems it is sometimes better and/or appropriate to work with a probabilistic description.

²Such problems involving statistical mechanics and thermal physics will be the focus of the next chapter.

³Each of these particles consists of many fat molecules, and even calling them a "particle" is a bit of an approximation (our chemist friends will not be happy). Even though we know that

cream particle as it moved through the coffee, we would find that it undergoes a complicated trajectory. Roughly speaking, it would move for a short period in a straight line (according to Newton's first law), between collisions with other cream particles and with coffee particles. Each collision would cause an abrupt change in our cream particle's velocity, and it would then move on with this new velocity until the next collision.

Our goal is to construct a useful theoretical description of the way the cream mixes with the coffee. We could *in principle* accomplish this by writing down the equations of motion for all of the particles, or even for all of the individual molecules that make up the cream and coffee particles, in the cup. This would give a very large number ($\sim 10^{23}$ on the molecular level) of differential equations which we could *in principle* solve (using, perhaps, the Euler method) to tell us everything there is to know about our cup of coffee. However, this approach is flawed for two reasons. First, while this calculation would be possible in principle, it certainly is not possible *in practice*. It seems highly unlikely that we will have a computer powerful enough for this computation any time soon. Second, even if we did have such a computer, the results of the calculation would be the positions and velocities of all our particles or molecules as a function of time. While these numbers would describe how the cream mixes with the coffee in great detail, it is not obvious that they would give us any real understanding of the process. By "understanding" we mean an intuitive picture that would allow us to apply what we have learned from the calculation to other similar situations. For example, if our computation showed that the cream was well mixed in, say, 20 seconds, could we then predict how long the mixing would take in a cup that was twice as large?

Our point is that such a complete computational solution of the problem would give far more information than we want or need to understand the mixing process. What we really require is a *statistical* description, or theory, of the behavior. We don't care about the detailed trajectory of every cream particle. It is enough for us to know the average properties of a particle trajectory. Since the number of particles involved is very large, these averages will be very sharply defined, and if we are careful we can also estimate the fluctuations from the averages using statistical arguments.

This discussion of the cream-in-your-coffee problem was intended to introduce you to a situation in which a statistical approach provides precisely the type of information we are after. Since we seek average properties, we will *model* the (real) deterministic process by a random or stochastic one that is designed to have the same average properties. Each cream particle follows a complicated trajectory as it collides repeatedly with other particles. Such a trajectory can be modelled by what is known as a "random walk." This is a process in which a particle (the walker) moves one step at a time, according to certain rules. Here the walker's steps correspond to the motion of the cream particle between collisions. Each collision changes the direction of the velocity of the particle, and this is modeled by letting the direction of each step in the walk be random. Hence, our random

a proper chemical and structural description of the cream particles would be more complicated, this simpliminded point of view is all we need here

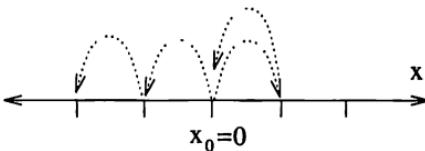


FIGURE 7.1: Sketch of a random walk in one dimension. The walker began at $x = x_0 = 0$, and each step is indicated schematically by a dotted arrow. Here the first step happened to be to the right, while the next three steps were to the left.

walker follows a zigzag path that is similar to the effectively random trajectory of a cream particle. Of course, the trajectory of a particle is not really random since, in principle, we could have calculated the motion of all the other particles in the cup and then predicted the precise times of the collisions, etc. However, we have already seen that this is not feasible, so we will treat these collisions as effectively *random* processes. The large number of particles (and collisions) in the problem will make this a very good approximation; we will return later to the question of how good this approximation really is. In any case, we hope that you are now convinced that it makes sense to model the motion of a particle of cream in a cup of coffee by a random walk.

7.2 RANDOM WALKS

In the previous section we argued that the motion of a particle or molecule in solution is analogous to a random walk. It turns out that such walks arise quite often in simulations of random processes, and in this section we will consider the simulation of several different types of random walks.

The simplest situation involves a walker that is able to take steps of length unity along a line. This one dimensional random walk is illustrated schematically in Figure 7.1. The walker begins at the origin, $x = 0$, and the first step is chosen at random to be either to the right or left, each with probability $1/2$. For the specific walk in Figure 7.1 the first step was to the right, so the location after the first step was $x_1 = +1$. The next step was then chosen, and again the probabilities for stepping left or right are both $1/2$. In this example the step went left, so $x_2 = 0$. This process can be repeated, and the position as a function of step number will be obtained. In a physical process such as the motion of a molecule in solution, the time between steps is approximately a constant, so the step number is roughly proportional to time. We will, therefore, often refer to the walker's position as a function of time.

A routine that implements a random walk in one dimension is illustrated below. Here we generate a random number in the range between 0 and 1 and compare its value to $1/2$. If it is less than $1/2$, our walker moves right, otherwise it steps to the left. This process is then repeated to generate x_n as a function of n . Any suitable pseudo-random number generator can be used (see Appendix F), but here we use a function we choose to call `rnd`.

EXAMPLE 7.1 Routine rwalk for simulating one-dimensional random walks and calculating the mean-squared displacement

- Initialize a random number generator (say, `rnd`) (if necessary).
 - Initialize an array $x2ave(i) \equiv 0$ ($i = 1, 2, \dots, n$) which will hold the squared displacements at time step i .
 - Loop through the desired number of walkers ($j = 1, \dots, m$).
 - ▷ Initialize the initial location of the walker to $x = 0$. (Note that you need not keep the locations of all of the walkers at all steps. You can simply accumulate the squared displacement at step i into the array $x2ave(i)$ as soon as it becomes available and update x for the $i + 1$ -st step, etc.)
 - ▷ Loop through the number of steps ($i = 1$ through n) in each walk.
 - Get a random number $r = rnd$ between 0 and 1.
 - If $r < 0.5$, update x to $x + 1$, otherwise to $x - 1$.
 - Accumulate the squared displacement: $x2ave(i) = x2ave(i) + x^2$.
 - Normalize the squared displacements to get the *mean* squared displacement averaged over all of the walkers: $x2ave(i) = x2ave(i)/m$ for all $i = 1, \dots, n$.
-

Figure 7.2 shows the results for two different walkers, obtained in two separate runs of this algorithm. The two walkers move erratically, which is why this process is sometimes referred to as a drunkard's walk. Several quantitative results can be obtained from these simulations. Perhaps the most basic is the mean displacement of a walker after n steps. Since a walker is as likely to step left as right, this average, which we denote by $\langle x_n \rangle$, must be zero.⁴ Here the angular brackets indicate an average over different walkers and is calculated by simulating a large number (m in the above sketched codes) of independent walkers and averaging their values of the quantity within the brackets (in this case, x_n).

A more interesting and informative quantity is $\langle x_n^2 \rangle$, the average of the square of the displacement after n steps (denoted by $x2ave$ in our `rwalk` routine). Some results for this quantity are shown on the right in Figure 7.2. We see that they are well described by a straight line, that is

$$\langle x^2 \rangle = 2 D t , \quad (7.1)$$

where t is the time, which here is just equal to the step number; the factor D is known as the diffusion constant.⁵ It is useful to compare this result with the behavior of a free particle, that is, one that is moving at a constant velocity and is not impeded by collisions with other particles. For such a particle we know

⁴This is why we did not specifically calculate it in `rwalk` above.

⁵The factor of 2 is inserted here so that the continuum limit of this random walk will reduce to the standard diffusion equation (see a later section for this connection). For a hypercubic lattice of d dimensions, $2D$ in this equation should be replaced by $2dD$. Note that the first edition of this book did not include this factor of 2.

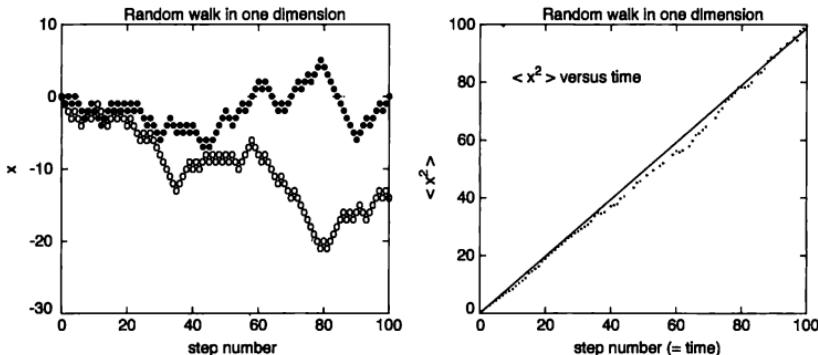


FIGURE 7.2: Left: x versus step number, that is, time, for two random walks in one dimension Right $\langle x^2 \rangle$ as a function of step number (which is proportional to time) for a collection of one-dimensional random walks. The step length was unity and the results for 500 walkers were averaged The points are the calculated values and the straight line is a least-squares fit to the form (7.1).

that $x = vt$, so its distance from the origin (its starting point) grows linearly with time. A random walker behaves differently; according to (7.1) its root-mean-square distance from the origin grows only as $\sqrt{\langle x^2 \rangle} \sim t^{1/2}$. Hence, a random walker escapes from the origin much more slowly than would a free particle.⁶

Motion of the sort described by (7.1) is also known as *diffusion*, and we will explore the connection between random walks and diffusion in somewhat more detail later in this chapter. However, a few comments regarding (7.1) should be made here. First, to follow up on a discussion that we began in Section 7.1, it turns out that this result tells us a lot about how fast a drop of cream will mix with coffee if we change the size of the cup. Mixing will be roughly complete⁷ when $\sqrt{\langle x^2 \rangle}$ is comparable to the diameter of the cup. If we double that diameter, then from (7.1), we see that it will take *four* times as long for the cream to mix. A second interesting point concerns the value of the diffusion constant, D . According to (7.1), the value of $2D$ is the slope of the $\langle x^2 \rangle$ versus t plot in Figure 7.2, which we see is approximately unity. This result for D can, in fact, be obtained analytically. Writing the position after n steps, x_n , as a sum of n separate steps gives

$$x_n = \sum_{i=1}^n s_i , \quad (7.2)$$

⁶The reader might object to this conclusion on the grounds that for small t the random-walk result $\sim t^{1/2}$ is larger than the constant velocity result $\sim t$. However, a comparison of these expressions at very small t is not appropriate. At such very short times the walker's first step has not yet "finished," so it, too, moves at a constant velocity. Hence, (7.1) applies only after several steps.

⁷We will make these ideas more quantitative below.

where s_i is the displacement of the i th step. For this problem s_i is a random variable, with $s_i = \pm 1$ with equal probabilities. We can then write

$$x_n^2 = \sum_{i=1}^n \left(\sum_{j=1}^n s_i s_j \right) . \quad (7.3)$$

Since the steps are independent of each other, the terms $s_i s_j$ with $i \neq j$ will be ± 1 with equal probability. If we average x_n^2 over a large number of separate walks this will leave only the terms with $i = j$ or s_i^2 . Thus we find

$$\langle x_n^2 \rangle = \sum_{i=1}^n s_i^2 = n , \quad (7.4)$$

where we have used the fact that $s_i^2 = 1$. Since n is also equal to time, this is identical to (7.1) with $D = 1/2$.

Let us now turn briefly to the fluctuations of x_n^2 , and consider how much this quantity varies from one walker to the next. These fluctuations correspond to the difference between the two random walkers on the left in Fig. 7.2. We would like to understand how these fluctuations vary with n ; in particular, how will the values of x_n^2 for two different (and typical) walkers diverge (or converge) as n becomes large? It is sometimes stated that these fluctuations decrease as n increases, but that is not the case, as can be seen from the following argument. We wish to calculate the fluctuation of x_n^2 with respect to its mean, and this will be given by

$$\sqrt{\langle (x_n^2 - \langle x_n^2 \rangle)^2 \rangle} = \sqrt{\langle x_n^4 \rangle - \langle x_n^2 \rangle^2} , \quad (7.5)$$

so we need to evaluate $\langle x_n^4 \rangle$. From (7.2) we have

$$\langle x_n^4 \rangle = \sum_{i,j,k,l=1}^n \langle s_i s_j s_k s_l \rangle . \quad (7.6)$$

Since the values of s_i for different i are independent, the only non-vanishing terms $\langle s_i s_j s_k s_l \rangle$ are those where either all the indices i, j, k, l are the same, or they occur in pairs of the same values. This leads to

$$\langle x_n^4 \rangle = \sum_{i=1}^n s_i^4 + 3 \sum_{i=1}^n \left[s_i^2 \sum_{j \neq i} s_j^2 \right] = 3n^2 - 2n , \quad (7.7)$$

and the root-mean-square fluctuation of x_n^2 is given by

$$\sqrt{\langle (x_n^2)^2 \rangle - \langle x_n^2 \rangle^2} = \sqrt{2n^2 - 2n} \sim \sqrt{2}n , \quad (7.8)$$

for large n . Thus the root-mean-square fluctuation of x_n^2 is itself proportional to the mean of x_n^2 and increases as n does. That is, the separation of two typical walkers will grow as n increases. In a sense, no *single* random walk behaves like the *average* random walk however large n may be.

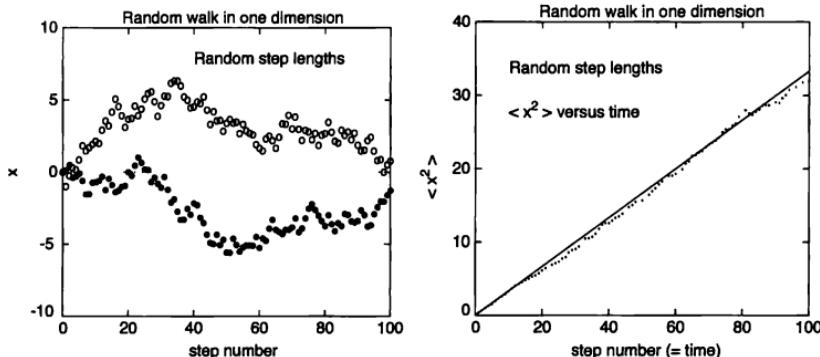


FIGURE 7.3: Left. x versus step number (that is, time) for two random walks in one dimension. Here the steps were of random lengths in the range -1 to 1 . Right: $\langle x^2 \rangle$ as a function of time for a collection of these one-dimensional random walks. The results for 500 walks were averaged.

Up to this point we have considered the simplest random-walk model. There are many ways to generalize the model to make it more realistic. One way is to allow the steps to be of random length.⁸ Some results for this case, again with a one-dimensional walker, are shown in Figure 7.3. We again find diffusive behavior, that is, $\langle x^2 \rangle$ is described by (7.1), but with a different value of the diffusion constant. The value of D in this case can again be calculated analytically, a job we will leave for the exercises.

Another obvious generalization is to allow the walker to move in three dimensions, and results for this case are shown in Figure 7.4. For this simulation we have restricted the steps to be of unit length along either $\pm x$, $\pm y$, or $\pm z$. Diffusive behavior is again found. There are many other interesting generalizations of the random-walk model. We will explore a few of them in the exercises, and also later in this chapter.

EXERCISES

- 7.1. Calculate the diffusion constant analytically for the random-walk simulations in Figures 7.3 and 7.4. These simulations were carried out in one dimension and on a three-dimensional simple-cubic lattice (respectively). Can you generalize your analytic results to a wider class of lattice types such as a two-dimensional triangular lattice, or a three-dimensional face-centered or body-centered cubic lattice?

⁸Here we have in mind randomly chosen step lengths within some fixed, and finite, range. If, on the other hand, we allow steps of arbitrary length, an entirely different behavior can result. One such case is called *Lévy flight*, and is described in the book by Mandelbrot listed in the references.

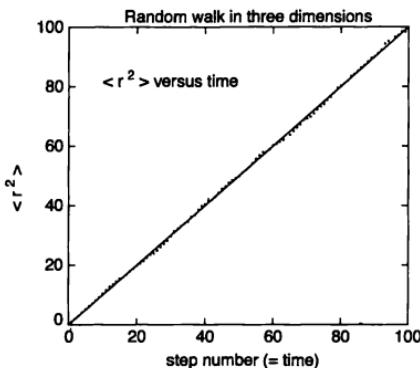


FIGURE 7.4: $\langle r^2 \rangle$ as a function of step number (that is, time) for a collection of three-dimensional random walks ($r^2 \equiv x^2 + y^2 + z^2$). The step length was unity and the results for 500 walks were averaged

- 7.2. Simulate a random walk in three dimensions allowing the walker to make steps of unit length in random directions; don't restrict the walker to sites on a discrete lattice. Show that the motion is diffusive, that is, $\langle r^2 \rangle \sim t$. Find the value of the proportionality constant.
- *7.3. Investigate the behavior of a random walk in which the probabilities for different step directions are not equal. For example, consider a one-dimensional walk with $p_{\text{left}} = 0.25$ and $p_{\text{right}} = 0.75$. In what sense is the motion still diffusive?
- *7.4. The calculated values for $\langle r^2 \rangle$ in Figure 7.2 do not agree perfectly with the theoretical prediction (7.1) with $D = 1/2$. Instead, these values exhibit statistical fluctuations about the theoretical curve. Show that these fluctuations follow a Gaussian form. Apply the χ^2 test to demonstrate that their magnitude is consistent with the expected statistical fluctuations (see Appendix G for a discussion of the χ^2 test).

7.3 SELF-AVOIDING WALKS

In the random-walk models we have considered so far, each step was completely independent of all prior steps. Indeed, this is what we would expect for truly random, diffusive behavior. However, in some physical processes this assumption is not appropriate. Consider a long flexible molecule, such as a polymer, which is made up of a large number of repeating units called monomers. A typical polymer in solution is not stretched out like a straight rod, but loosely "coiled up" with many turns into a three-dimensional form. Suppose we want to construct a model to describe the shape of such a molecule. A random walk is an obvious candidate for such a model. Each link (or monomer) in the polymer chain corresponds to one step in the walk, and since the polymer is flexible (the angles between successive

links are usually not rigidly fixed, but can take on several different values), each step is independent of the one immediately before it.⁹ However, for this problem a random walk ignores an important piece of physics. The path followed by our polymer molecule must not be allowed to intersect itself. Only one segment of the polymer is allowed to occupy any particular region of space. A random walk that is subject to this constraint is called a self-avoiding walk, or SAW. Each distinct SAW of the same number of steps (links) must occur with equal probability. The collection of all such SAWs is often called the SAW *ensemble*, and the study of various statistical properties of this ensemble is termed the SAW problem.

The SAW problem is commonly studied using two approaches: *enumeration* and *simulation*. "Enumeration" refers to actually listing all possible configurations of a given number of steps. On the other hand, "simulation" refers to sampling from the ensemble of all possible SAWs as we did in our studies of random walks in the previous section. We will discuss the important topic of SAW enumeration a bit later, but first we consider the simulation approach.

A simulation of SAWs is similar to that of ordinary random walks, but with one very important difference: we must keep track of all prior steps and make sure that configurations that would revisit a previously trampled site are not included. At first glance, this would appear to be accomplished by simply growing them one step at a time, randomly choosing the location to take the next step from the available (i.e., still unvisited) neighboring sites, terminating the walk when no further steps consistent with the self-avoiding constraint are possible.¹⁰ However, this approach contains a subtle flaw. It is known that for polymers in an equilibrium solution, all possible SAW configurations will occur with *equal* probability. However, if we generate a collection of SAWs by simply growing them as described above, the resulting ensemble will *not* satisfy this condition on the probabilities. This point is illustrated by the two different 3-step SAWs on the square lattice shown in Figure 7.5. Both of these occur with equal frequency if we generate SAWs step by step (i.e., by our "growth" approach), taking all possible directions of the next step with equal probability. However, in (a), there are 3 possible ways to generate the 4th step, while in (b), there are only 2 possible ways. Thus, if we choose only from the *possible* (i.e., non-intersecting) directions, then each 4-step extension of the SAW in (a) has 2/3 the probability of occurrence as those in (b). As a result, an ensemble generated in this way will not contain all of the possible SAWs with equal probability.

⁹Actually, we are simplifying the situation a great deal (as usual!). In a real, three-dimensional polymer, the bond angle between two successive monomers is usually fixed, but the azimuthal angle that is defined by the relative orientation of three successive monomers will vary. This is called *isomeric rotation*, and it is this azimuthal angle that can have several different values. In addition, some polymers are *stiff*, that is, the probability for one of the preferred azimuthal angles is much greater than the other ones, resulting in long stretches of monomers in a straight, rod-like configuration. In such cases, it is a better approximation to consider a link (sometimes referred to as the *persistence length*) to be consisting of many monomers rather than of a single monomer. In a realistic simulation of polymers, such details are taken into account. However, we will ignore these complications and use an isotropic random walk model on a lattice.

¹⁰Walks generated by this type of algorithm are sometimes called *growing* or *kinetic* self-avoiding walks. This is an attractive way to model certain types of growth processes. See, e.g., Vicsek in the references.

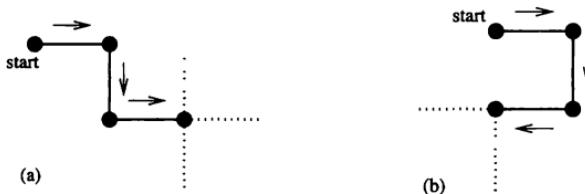


FIGURE 7.5: Two 3-step SAWs are shown. In (a), there are 3 possible ways to extend it to a 4-step SAW while, in (b), there are only 2 possible ways

The simplest way to ensure the correct statistical weight in the ensemble is to grow a walk step by step, randomly selecting the next *attempted* step from *all* directions,¹¹ and if that attempt ends up in self-intersection, the whole walk is discarded and a new one started from scratch. Returning to the illustration above, if we use this algorithm, then all surviving 4-step extensions in (a) and (b) occur with equal probability, as required.

This approach to SAW simulation is very inefficient, as self-intersections cause many of the walks to be discarded.¹² In fact, this attrition problem is so severe that a very large amount of CPU time is needed in typical calculations. This is in contrast to the simulation of ordinary random walks where there is absolutely no attrition. However, even this attrition observed in simulations can be used to deduce certain statistical characteristics of the SAWs (see the exercises).

The self-avoidance constraint (often called the excluded-volume effect) also has an obvious effect on the size of SAW. As compared with an ordinary random walk, a self-avoiding walker will, on average, get farther away from its starting point in a given number of steps. Avoidance of self-intersections forces a SAW to stretch out more rapidly into previously unvisited regions of space. This is illustrated in Figure 7.6, which shows that for a two-dimensional SAW the mean-square distance from the starting point, $\langle r^2 \rangle$, does not vary linearly with time as is the case for an ordinary random walk. We see that $\langle r^2 \rangle \sim t^{1.5}$ for this SAW, so the behavior is intermediate between that of a random walk (diffusion), for which $\langle r^2 \rangle \sim t$, and a free (i.e., ballistic) particle, for which $\langle r^2 \rangle \sim t^2$. Thus, if we define the exponent ν (called the Flory exponent) by

$$\sqrt{\langle r^2 \rangle} \sim A t^\nu, \quad (7.9)$$

for large t , the value of ν describes the asymptotic large t behavior of various walks. Its value for the SAW problem in two dimensions¹³ is $3/4$, while it is approximately

¹¹Except immediately reversing the last step. Try to convince yourself that this exception is still consistent with (II).

¹²There are a number of simulation methods developed to partially counter the high attrition. Most of them use some form of *enrichment*, an idea first introduced by Wall and Erpenbeck (see the references).

¹³This is actually an exact result, though we demonstrate it here only approximately with our simulations.

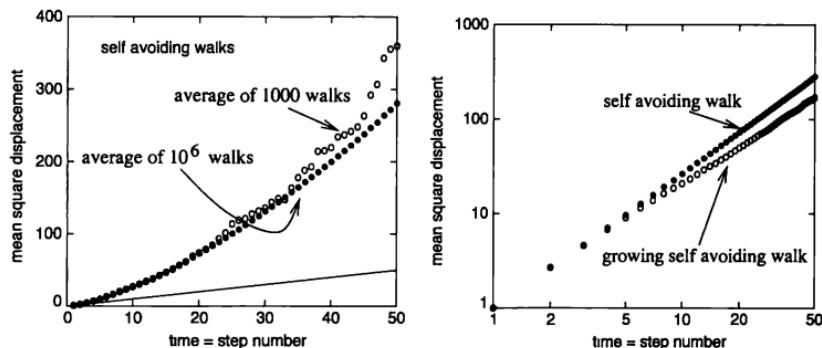


FIGURE 7.6: Left results for $\langle r^2 \rangle$ as a function of step number (i.e., time) for a self-avoiding walk in two dimensions. The results of simulations starting 1000 walks (open circles) and 10^6 walks (filled circle) are shown. Note that $\langle r^2 \rangle$ increases much more rapidly here than it does for a simple random walk, which would follow the solid line. The “scatter” apparent in the plot for the results from starting 1000 walks (but not in the one for starting 10^6 walks) is due to the statistical fluctuations of the calculated mean. Although the fluctuations of the squared-displacements from walk to walk are always of the order of the mean-squared displacement itself, the fluctuations of the means can be reduced by averaging the results for more walkers. Right: results for the average of 10^6 SAWs now plotted on log-log scales (filled circles, same data as on the left). Also shown (open circles) are the average mean-squared displacement for 1000 growing self-avoiding walks. The filled circles follow the functional form $\langle r^2 \rangle \sim t^{2\nu}$, with the result $\nu \approx 0.735$ (the exact result is $3/4$), while open circles give $\nu \approx 0.661$ for the growing SAW. For a simple random walk, as in Figure 7.2, $\nu = 1/2$.

$3/5$ in three dimensions.

The very high attrition rate in SAW simulations makes it worthwhile to investigate other approaches. The method we consider now is called enumeration. In this approach, *all* SAWs of a given number of steps, say, n of steps are generated and their average properties such as the $\langle r_n^2 \rangle$ are calculated *exactly* for that n . Of course, these exact results can be calculated only for relatively small n because of the exponentially growing CPU time that is required, but we can then *extrapolate* them to $n \rightarrow \infty$ to obtain the asymptotic large n results. In addition, the numerical techniques used in such enumerations are useful in many other contexts, so we will discuss such methods both here and a little later in this chapter.

We can produce a SAW by letting it “grow” starting from some initial seed. Actually, this produces an entire sequence of SAWs, as we let the number of steps, n , grow. The point of enumeration is to get a count of *all* possible SAWs, and this can be accomplished in two general ways. One is to grow many SAWs, but do so one at a time, letting all grow to the same length n . This is called a *depth-first* approach. In the language of computer science, we are dealing with a “tree” of SAWs, and we are traveling (i.e., searching) through this tree to a “depth” n . We

now sketch one depth-first search algorithm for the SAW problem. This algorithm proceeds by successively choosing and extending the SAW in the most preferred direction; when this fails by a self-intersection, it backtracks one step and searches other less preferred directions.¹⁴

EXAMPLE 7.2 Depth-first search algorithm for SAW enumeration

- Decide on a fixed *local* search order around a site. Initialize all sites as *unvisited*.
 - Mark the starting point as *visited* and call it the *current* site $n = 1$. Set the search direction index at this site as $\text{dir}(1) = 1$. (The assigned value 1 signifies that the next direction to search is the first one in the predefined order of search directions.)
 - *Test* if all directions around the current site have already been searched.
 - ▷ If yes:
 - Mark the current site n as *unvisited*.
 - Decrement the current site index $n \rightarrow n - 1$ (backtrack to previous step).
 - If the new $n = 0$, you are done. Otherwise, repeat the test around the revised current site.
 - ▷ If not, test whether the target site in the direction $\text{dir}(n)$ from the current site n is available (i.e., *unvisited*):
 - If it is:
 - ▷ Extend SAW to this site and add this walk to the enumeration collection.
 - ▷ Increment $\text{dir}(n)$ (i.e., update the next direction to search when you later return to site n for more SAWs).
 - ▷ Test if $n + 1$ is the predetermined length of the SAW.
 - If it has, we have reached the desired length. Just enumerate other directions around site n by going back to the *test*.
 - If not:
 - ▷ Update the current site index by incrementing $n \rightarrow n + 1$.
 - ▷ Mark the new current site as *visited*.
 - ▷ Set $\text{dir}(n) = 1$.
 - ▷ Repeat the *test*.
 - If it is not available, increment $\text{dir}(n)$ and repeat the *test*.
-

¹⁴In a later section we will describe a *breadth-first search* in the context of the percolation problem.

TABLE 7.1: Exact results (obtained by enumeration) for the number of SAWs on a square lattice, as a function of the number of steps n . Note that the number of walks increases *extremely* rapidly with n . However, the mean square displacement $\langle r^2 \rangle$ grows much more slowly, and the fluctuations in the displacement ($\sqrt{\langle (\Delta r^2)^2 \rangle}$, listed as the standard deviation) are of order $\langle r^2 \rangle$.

number of steps (n)	$\langle r^2 \rangle$	standard deviation	Number of SAWs
1	1.00	0.00	4
2	2.67	0.94	12
3	4.56	2.27	36
4	7.04	3.54	100
5	9.56	5.21	284
6	12.57	6.90	780
7	15.56	8.93	2,172
8	19.01	10.94	5,916
9	22.41	13.25	16,268
10	26.24	15.55	44,100
11	30.02	18.10	120,292
12	34.19	20.65	324,932
13	38.30	23.43	881,500
14	42.79	26.21	2,374,444
15	47.22	29.20	6.417×10^6
16	51.99	32.19	1.726×10^7
17	56.72	35.37	4.647×10^7
18	61.77	38.56	1.247×10^8
19	66.77	41.93	3.351×10^8
20	72.08	45.29	8.977×10^8

While this algorithm is relatively sophisticated in concept, actual programs implementing it can be very compact. It is one of the shortest and sweetest programs that you will find in this book. Table 7.3 gives the results for the SAWs of up to 20 steps on the square lattice.

Once we have generated all SAWs up to some desired number of steps, n , and have the exact values of $\langle r_n^2 \rangle$, we can use the fact that¹⁵

$$\frac{\langle r_{n+1}^2 \rangle}{\langle r_n^2 \rangle} \sim (1 + \frac{1}{n})^{2\nu} \sim 1 + \frac{2\nu}{n}, \quad (7.10)$$

for asymptotically large n . Thus, the plot of this ratio vs $1/n$ should approach a line¹⁶ with the slope equal to 2ν . Figure 7.7 shows such an extrapolation, based on the data of the Table 7.3.

¹⁵This result can be derived from (7.9)

¹⁶There are various ways to improve on this simple-minded treatment but we will leave them for you to explore in the references

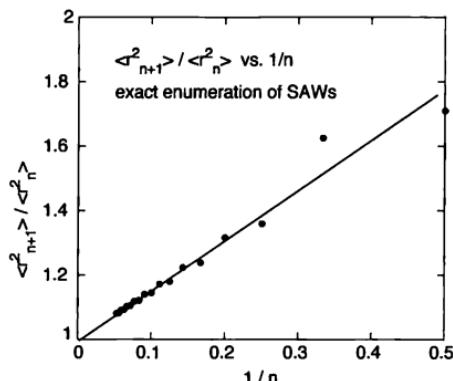


FIGURE 7.7: Results for $\langle r_{n+1}^2 \rangle / \langle r_n^2 \rangle$ from the exact enumeration of the SAWs on a square lattice, as discussed in text. The solid line is a linear least-squares fit to $n = 4$ through 19 and has a slope of approximately 1.47, corresponding (to within the statistical errors) to a Flory exponent of $\nu = 0.735$.

EXERCISES

7.5. Let us denote by C_n the total number of all distinct SAWs of n steps, say, on a square lattice. According to the SAW simulation procedure suggested above, the probability that a SAW started in a simulation is still surviving after n steps should be equal to

$$P(n) = \frac{C_n}{3^n}, \quad (7.11)$$

which describes the attrition of simulated SAWs. That is, if you start simulating, say, 10^6 SAWs, then on average $P(n) \times 10^6$ of them will survive to n steps. Calculate $P(n)$ up to $n = 50$ by simulation.

The asymptotic, large n behavior of C_n is known to be

$$C_n \sim C_0 \mu^n n^{\gamma-1}, \quad (7.12)$$

where μ (< 3) is the effective average number of places you can extend a SAW by another step and γ is another exponent defined by this relation. Substituting this into (7.11), we get

$$\frac{P(n+1)}{P(n)} \sim \frac{\mu}{3} \left(1 + \frac{\gamma-1}{n}\right). \quad (7.13)$$

Use your results for $P(n+1)/P(n)$ as a function of $1/n$ to estimate μ and γ .

- 7.6. Simulate SAWs in three dimensions. Determine the variation of $\langle r^2 \rangle$ with step number and find the value of ν , where this parameter is defined through the relation (7.9). Compare your results with those in Figure 7.6. You should find

that ν decreases for successively higher dimensions. (It is 1 in one dimension and 3/4 in two dimensions.) Can you explain this trend qualitatively?

- *7.7. Simulate SAWs in four dimensions using a four-dimensional, hyper-cubic lattice. Obtain the Flory exponent ν by comparing your simulation results of $\langle r^2 \rangle$ with (7.9).

Further work: You will find that the observed fit gives a value slightly larger than the random walk value of 1/2. For the SAW problem, however, four dimensions can be shown to be the borderline between the regime where the excluded-volume effect is important and one where it is not. At such borderline (or *marginal*) dimensions, the defining relation (7.9) acquires a *logarithmic* correction:

$$\sqrt{\langle r^2 \rangle} \sim A t^\nu (\ln t)^\theta, \quad (7.14)$$

where θ is yet another exponent. Use multiple linear regression discussed in Appendix D (or any other suitable method) to estimate ν and θ from (7.14). The exact value of ν in four dimensions is 1/2.

- 7.8. Enumerate exactly all SAWs of up to 20 steps on the triangular lattice and tabulate the results for $\langle r_n^2 \rangle$ and C_n . Estimate the Flory exponent ν from these results. How does it compare with the theoretically expected exact result of 3/4 for all lattices in two dimensions?

7.4 RANDOM WALKS AND DIFFUSION

We have mentioned several times that random walks are equivalent to diffusion. In this section we will explore this connection in a little more detail. We will again adopt the cream-in-your-coffee analogy in which we have a large number of particles (cream) moving in solution (coffee). The goal is to calculate how these particles are spatially distributed as a function of time. In our discussion of random walkers we have, up to this point, focused on the motion of individual walkers. An alternative way to describe the same physics involves the density of particles, $\rho(x, y, z, t)$, which can be conveniently defined if the system contains a large number of particles (walkers). The idea, known as *coarse graining*, is to consider regions of space that are big enough to contain a large number of particles so that the density (\equiv mass/volume) can be meaningfully defined. The density is then proportional to the probability per unit volume per unit time, denoted by $P(x, y, z, t)$, to find a particle at (x, y, z) at time t . Thus, ρ and P obey the same equation.

To find this equation, we focus back on an individual random walker. We assume that it is confined to take steps on a simple-cubic lattice, and that it makes one “walking step” each time step. $P(i, j, k, n)$ is the probability to find the particle at the site (i, j, k) at time n . Since we are on a simple cubic lattice, there are 6 different nearest neighbor sites. If the walker is on one of these sites at time $n - 1$, there is a probability of 1/6 that it will then move to site (i, j, k) at time n . Hence, the total probability to arrive at (i, j, k) is

$$\begin{aligned} P(i, j, k, n) = & \frac{1}{6} [P(i+1, j, k, n-1) + P(i-1, j, k, n-1) + P(i, j+1, k, n-1) \\ & + P(i, j-1, k, n-1) + P(i, j, k+1, n-1) + P(i, j, k-1, n-1)]. \end{aligned} \quad (7.15)$$

Rearranging this equation, we get

$$\begin{aligned} P(i, j, k, n) - P(i, j, k, n-1) &= \\ \frac{1}{6} &\{[P(i+1, j, k, n-1) - 2P(i, j, k, n-1) + P(i-1, j, k, n-1)] \\ &+[P(i, j+1, k, n-1) - 2P(i, j, k, n-1) + P(i, j-1, k, n-1)] \\ &+[P(i, j, k+1, n-1) - 2P(i, j, k, n-1) + P(i, j, k-1, n-1)]\}. \quad (7.16) \end{aligned}$$

Apart from a constant factor ($1/\Delta t$), the left side of this equation is just the finite difference approximation for the time derivative of P , while the right-hand side is proportional to a second order space derivative. This suggests taking the continuum limit, which leads to

$$\frac{\partial P(x, y, z, t)}{\partial t} = D \nabla^2 P(x, y, z, t), \quad (7.17)$$

where $D = (1/6)(\Delta x)^2/\Delta t$ in this approximation.¹⁷ Equation (7.17) is the diffusion equation, and our derivation shows the close connection between the random walks and diffusion. The density ρ obeys the same equation

$$\frac{\partial \rho}{\partial t} = D \nabla^2 \rho. \quad (7.18)$$

We encountered a similar differential equation in our studies of waves in Chapter 6, and the numerical approach we used there can be extended to treat the diffusion equation. For ease of notation we will assume that ρ is a function of only one spatial dimension, x , although everything we do below can readily be extended to two or three dimensions. We can then write $\rho(x, t) = \rho(i\Delta x, n\Delta t) = \rho(i, n)$, so that the first index corresponds to space and the second to time. Converting (7.18) to one dimension yields

$$\frac{\partial \rho}{\partial t} = D \frac{\partial^2 \rho}{\partial x^2}, \quad (7.19)$$

and the finite-difference version of this (as in (7.16)) is

$$\frac{\rho(i, n+1) - \rho(i, n)}{\Delta t} = \frac{\rho(i+1, n) - 2\rho(i, n) + \rho(i-1, n)}{(\Delta x)^2}. \quad (7.20)$$

Rearranging to express the density at time step $n+1$ in terms of ρ at step n we find

$$\rho(i, n+1) = \rho(i, n) + \frac{D \Delta t}{(\Delta x)^2} [\rho(i+1, n) + \rho(i-1, n) - 2\rho(i, n)]. \quad (7.21)$$

If we are given the initial distribution of the cream particles, $\rho(x, t=0)$, we can use (7.21) to solve for ρ at future times. A program to implement this can be constructed along the lines we developed in Chapter 6 to deal with waves on a string, so we will leave the details to the exercises. While the programming is

¹⁷This particular value of D is for a simple cubic lattice, and is not universal.

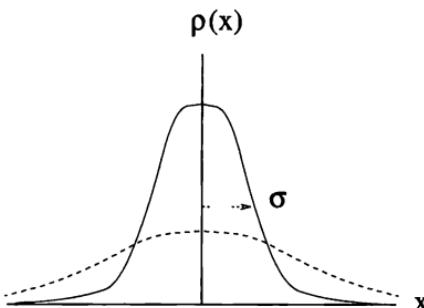


FIGURE 7.8: Schematic solutions of the diffusion equation at two different times. The solid curve shows the density at an early time, while the dashed curve shows it at some time later. The Gaussian distribution broadens with time, but the area under the curve, which is equal to the total number of particles, does not change.

straightforward, there remains the choice of spatial and temporal step sizes. As you might have guessed, the numerical instabilities we encountered when solving the wave equation can also arise here.

While it is not easy to provide a general analytic solution to the diffusion equation, one special case is very instructive. You can verify by substitution that the function

$$\rho(x, t) = \frac{1}{\sigma} \exp \left[-\frac{x^2}{2\sigma^2} \right], \quad (7.22)$$

satisfies (7.19), provided that σ is *time dependent*, with $\sigma = \sqrt{2Dt}$. This result can be understood intuitively from Figure 7.8, which shows sketches of the density at two different times. At any particular time the spatial distribution has a Gaussian form whose half-width σ is, roughly speaking, the spatial size occupied by the clump of particles. As time passes, the density maintains a Gaussian form with the only change being that the width increases as $\sigma \sim \sqrt{t}$. That this is also just the root-mean-square distance traveled by an average particle can be seen as follows. At $t = 0$ the clump of particles will, according to our assumptions concerning how the drop is deposited, be very small ($\sigma \sim 0$). At a later time the cream distribution will be of order σ in extent, so this must also be the distance traveled by a typical diffusing cream particle. Hence, the distance traveled by a particle as it diffuses varies as $t^{1/2}$. This behavior¹⁸ is precisely what we found in our studies of random walkers (7.1).

¹⁸This result is also an example of the central-limit theorem, which is mentioned in Appendix G in connection with the distribution functions associated with random processes. In the present problem many random values (steps in the diffusion/random-walk process) combine to yield a Gaussian distribution. The width of this distribution is proportional to $N^{1/2}$, where N is the number of steps. Here $N \sim t$.

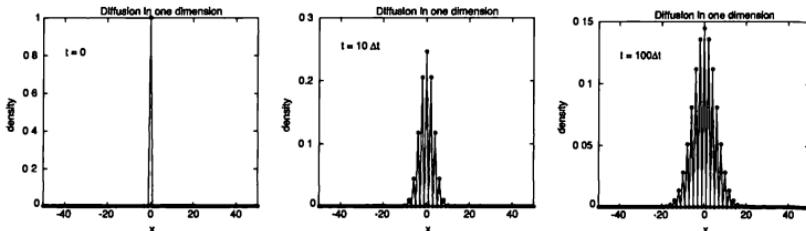


FIGURE 7.9: Time evolution calculated from the diffusion equation in one dimension at $t = 0$ (left), $t = 10\Delta t$ (center), and $t = 100\Delta t$ (right). The spatial step was $\Delta x = 1$ so the density was calculated only at the locations $x = 0, \pm 1, \pm 2$, etc. The lines are simply drawn to connect these points. We chose $D = 1$ and $\Delta t = 0.5$ so as to satisfy the stability condition (7.23) as an equality. Note that the vertical scales are different in the different plots; the maximum value of the density decreased as t increased, so as to keep the total number of particles fixed.

This analytic result tells us that a disturbance such as our particle distribution can be expected to spread by an amount as large as $\sim \sqrt{2D\tau}$ during time τ , or equivalently, it takes the distribution a time of order $\ell^2/(2D)$ to spread a distance ℓ . Thus, to guarantee numerical stability we must make sure that the space and time steps satisfy

$$\Delta t \leq (\Delta x)^2/(2D), \quad (7.23)$$

since larger values of the time step size would not allow the computed distribution to spread as quickly as we know that it must. This potential instability is similar to that found in connection with waves on a string, which should not be surprising since we are dealing with a similar equation.

A numerical solution of the diffusion equation obtained using (7.21) is shown in Figure 7.9. Here we have assumed that the initial density is zero everywhere except at the origin, $x = 0$; thus, our drop of cream is all initially located in one very small region. At $t = 10\Delta t$ the density profile has broadened, as the particles have spread over the range $x \approx \pm 6$. The amount of spreading has increased further at $t = 100\Delta t$. Qualitatively this distribution has spread out an additional factor of ≈ 3 for a tenfold increase in time, which is what we expect for diffusion.

A curious feature of the results for $t > 0$ is that the density alternates between zero and nonzero values. This behavior is due to the initial density profile which we assumed. Our initial profile had all of the density situated at a *single* grid site. This violates our usual rule-of-thumb that step sizes should always be smaller than any of the characteristic scales in the problem. Here one characteristic length scale is the spatial extent of the density profile. By allowing all of the density to be located at a single grid site we are effectively taking the density distribution to be a singular function.¹⁹ The price we pay for this is that the finite difference equation (7.21) produces these spurious zeros.²⁰

¹⁹This is much like a Dirac delta function or a “point” charge

²⁰This can be appreciated by evaluating $\rho_i(1)$ by hand, using (7.21), for the initial profile

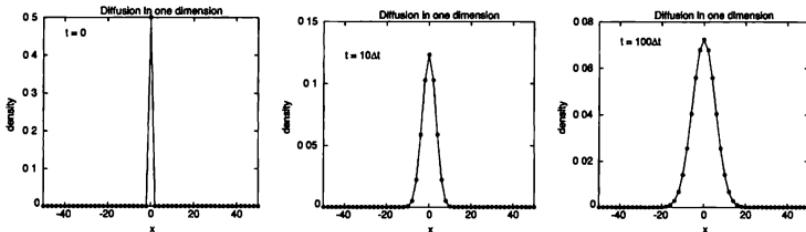


FIGURE 7.10: Time evolution calculated from the diffusion equation in one dimension at $t = 0$ (left), $t = 10\Delta t$ (center), and $t = 100\Delta t$ (right). Here we have averaged the values from Figure 7.9 over adjacent spatial grid sites to minimize discreteness effects.

One way to overcome this problem would be to use a smaller spatial step size so that the initial density profile is spread out over many grid sites. We will explore this approach in the exercises. Another, perhaps a bit more expedient (but equivalent) solution is to simply average the results in Figure 7.9 over adjacent grid elements. That is, we can average the results for adjacent spatial sites so as to “smooth over” the points where the density was zero. Doing this yields the results in Figure 7.10. The density profiles at $t = 10\Delta t$ and $100\Delta t$ now have the expected Gaussian shapes. The amplitude of the Gaussian distribution decreases and its width increases as we follow the motion to larger times. We will leave it for the reader to show that the width of the distribution, which is the scale over which the particles are spread, varies as \sqrt{t} . This is, of course, a defining feature of diffusion.

This numerical approach to the diffusion equation is very general and can be used just as well in two or three dimensions. We will explore the derivation of the corresponding finite difference equations (which are analogous to (7.21)) in the exercises. The stability condition becomes somewhat more restrictive on Δt as the dimensionality increases, a topic we will leave for the references. Figure 7.11 shows some results for a two-dimensional case. Here we have assumed initial conditions as in the cream-in-your-coffee problem, with all of the particles confined to a square region surrounding the origin. The density profile then spreads with time in an approximately spherical manner. This numerical approach can also be used to study other types of problems. All that is needed is the initial density profile. The finite difference algorithm (7.21) can then be used to calculate the profile at all future times.

To make the connection between the diffusion equation and random walks even more explicit, we now consider the same cream-in-your-coffee problem using a random-walk approach. We do this by considering a large number of walkers

in Figure 7.9. If you do this you will also see that in order to get a density of precisely zero at alternating grid sites, the spatial and temporal grid sizes must be chosen according to the stability condition, as we have done in our simulation. Other choices of the grid sizes would not necessarily lead to such exact cancellations of the density, but would still yield values that alternate in magnitude.

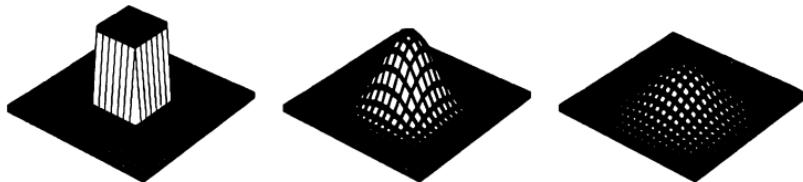


FIGURE 7.11: Time evolution calculated from the diffusion equation in two dimensions. At $t = 0$ (left) all of the particles were confined to “clump” at the center of the system. The distributions at $t = 6\Delta t$ (center), and $t = 20\Delta t$ (right) are also shown. The parameters used in the calculation were $\Delta z = 1$, $D = 1$, and $\Delta t = 0.25$. The region shown here covers the range $|z, y| \leq 10$.

that all start at the origin. This corresponds to the cream particles just after they have been deposited into the coffee. To obtain the density profile at time $t = n\Delta t$, we calculate the *probability distribution* of the walkers after n steps. That is, we let every walker take n steps, each of length unity, and record their positions. A histogram of the number of walkers that end up at location x , as a function of x , is then constructed. This is the desired probability distribution, or equivalently, density profile.

Some results for the one dimensional case are shown in Figure 7.12, and at first sight they might seem a bit strange as the probability of finding a walker at the odd-numbered grid sites is zero. However, this can be understood by recognizing that if a walker that starts from the origin and takes an even number of steps, it must end up at an even numbered site. Thus, we should really average the results over adjacent grid sites, much as we did with the diffusion-equation results.²¹ In any case, the random-walk results exhibit the Gaussian spreading of the particle distribution that we have come to expect. Indeed, our derivation of (7.17) shows that the analogy with diffusion is exact. The general approach is to consider a large number of walkers distributed in space according to the particular initial conditions of the problem. Each walker is then allowed to move and the distribution of walkers is monitored as a function of step number, that is, time. We will employ this approach to bring out another aspect of the cream-in-your-coffee problem in Section 7.7.

EXERCISES

- 7.9.** Write a program to solve the finite-difference form of the diffusion equation in one dimension, (7.21). Use it to confirm that (7.22) is, indeed, a solution. To do this, begin with an initial density profile that is sharply peaked at $x = 0$, but choose the grid size such that this profile extends over at least several grid sites. Then show that at later times the density distribution satisfies (7.22).

²¹We could also start with walkers distributed over several grid sites, or let the number of steps be randomly distributed over some range.

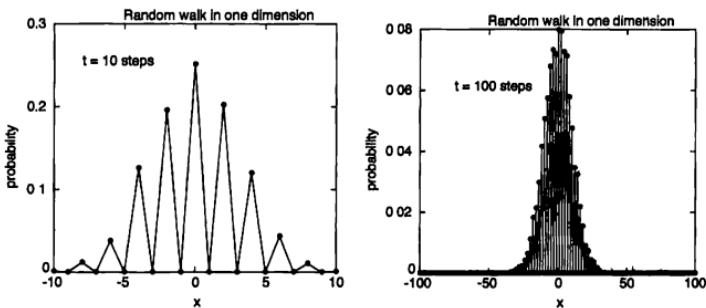


FIGURE 7.12: Random-walk distributions in one dimension A large number of walkers began at the origin, and their positions after 10 (shown at left) and 100 steps (shown at right) were used to obtain the probability distribution as a function of position

- *7.10. Repeat the calculation of the previous problem in either two or three dimensions (or both). Begin by deriving the finite-difference equation corresponding to (7.21), suitably generalized to the desired dimensionality. Then show that the density distribution spreads in time according to (7.22) with a half-width that grows as $t^{1/2}$
- *7.11. Use the program developed for the previous problem to investigate how more complicated initial-density distributions evolve with time. For example, consider an initial distribution that is a constant along the x axis and zero everywhere else. Study how this distribution spreads with time

7.5 DIFFUSION, ENTROPY, AND THE ARROW OF TIME

We introduced the cream-in-your-coffee problem at the beginning of this chapter when we were trying to motivate an interest in random processes. Now we want to reconsider it from the point of view of nonequilibrium statistical mechanics and use it to illustrate how a system approaches equilibrium.

Our initial conditions are, again, a cup of black coffee containing a drop of cream at its center. For simplicity we consider a two-dimensional cup with an initial cream distribution as shown in Figure 7.13. The black dots, which form a square black mass at $t = 0$, are the cream particles. For the simulation we assume that each of these particles executes a random walk on a two-dimensional square lattice and allow multiple occupancy of a lattice site (although our results would not change qualitatively if we were to limit occupancy to only one particle per site). At each time step we choose a particle at random and let it take one step in its random walk. The distributions after 10^4 , 10^6 , and 10^8 time steps are shown in Figures 7.13 and 7.14. As expected, the cream spreads with time in a manner that appears by eye to be diffusive (we will leave quantitative verification of this claim to the exercises). We have assumed that there are walls at $x = \pm 100$ and $y = \pm 100$, so the particles are constrained to stay in the region shown here.

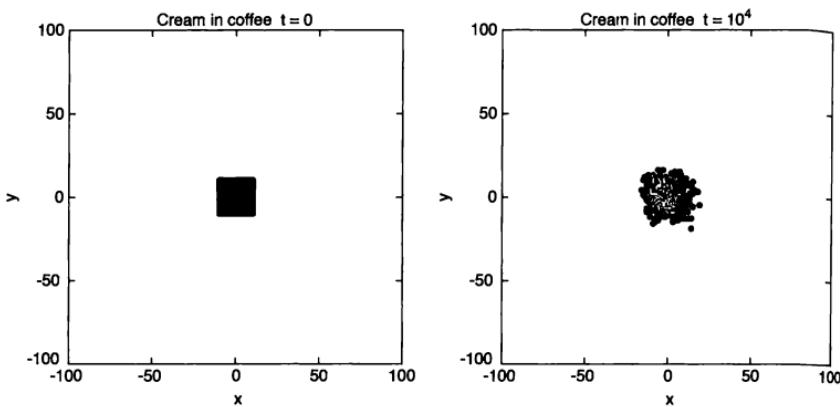


FIGURE 7.13: Random-walk simulation of diffusion of cream in coffee. Left: the initial ($t = 0$) cream distribution in which all of the particles were near the center of the cup. Right: after $t = 10^4$ time steps, only a little spreading has taken place. There were 400 particles constrained to a 200×200 square lattice.

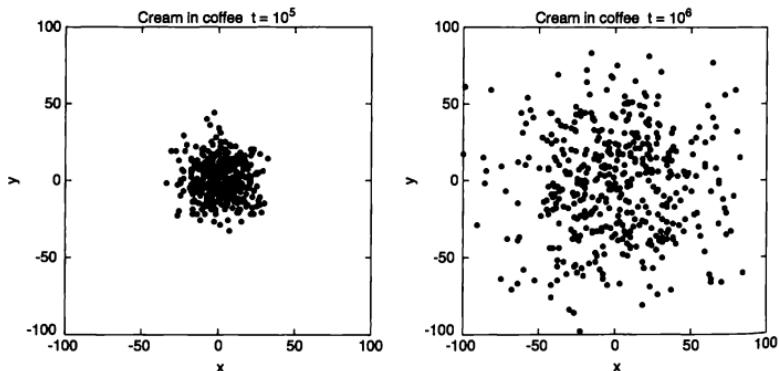


FIGURE 7.14: Diffusion of cream in coffee, continuation of the random-walk simulation in Figure 7.13. Left: after $t = 10^5$ time steps, right: after $t = 10^6$ time steps.

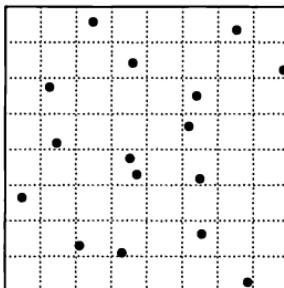


FIGURE 7.15: Schematic division of our coffee cup into grid cells, with a few particles distributed throughout the cup. P_i is the probability of finding a particle in cell i .

These results are equivalent to our solution of the two dimensional diffusion equation in the previous section. Here we want to carry this example one step further, and discuss how it is related to the second law of thermodynamics and the manner in which systems approach equilibrium. For this it is useful to consider the entropy of the system. Roughly speaking, entropy is a measure of the amount of disorder. A perfectly ordered system has zero entropy, while a disordered one has a large entropy. Furthermore, statistical physics tells us that the entropy of a closed system will either remain the same or increase with time.

Our cream-in-your-coffee simulation illustrates these ideas very nicely. Initially, all of the cream particles are packed tightly into a small region of the cup, so the system is highly ordered and has a small value of the entropy. As time passes the particles spread to fill the cup and their arrangement becomes more disordered. We can make this description quantitative by calculating the entropy explicitly. To do this we recall that the statistical definition of entropy S is

$$S = - \sum_i P_i \ln P_i , \quad (7.24)$$

where the sum is over all possible states of the system and P_i is the probability of finding the system in state i . To apply this definition to our problem we imagine that the system is divided into a square grid, as shown in Figure 7.15. Note that this grid is *not* related to the square lattice occupied by our walkers. It is just a convenient way of partitioning space; each of these partitions is a distinct state in which a particle might be found. To appreciate the meaning of (7.24) it is useful to first imagine a system containing only a single cream particle (we'll add the others from Figures 7.13 and 7.14 in a moment). The state we label i then corresponds to the particle being located in grid cell i , and P_i is the probability of finding the particle in this cell at any particular time. The sum over i in (7.24) is, then, a sum over all of the cells in the grid. The simulations in Figures 7.13 and 7.14 involve a large number of particles, and we can use all of them in the computation of P_i .

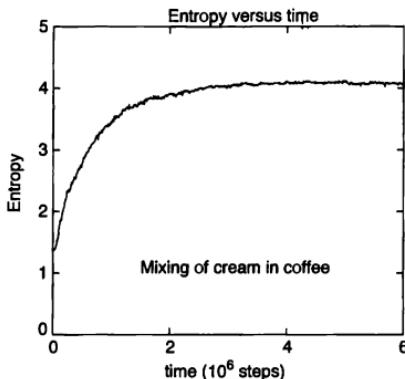


FIGURE 7.16: Entropy as a function of time (total number of random-walk steps) for cream mixing in coffee, calculated from the simulation that yielded the snapshots in Figures 7.13 and 7.14. The grid used to calculate the entropy had $8 \times 8 = 64$ cells

We have used the particle positions from the random-walk simulation in Figures 7.13 and 7.14 to calculate these probabilities and evaluate S , and some results are shown in Figure 7.16. The behavior is in complete accord with our intuitive definition of entropy as it applies to the cream. The system is initially in a highly ordered state, with a low value of S . As time passes, the entropy increases and eventually levels off; it approaches a constant value at long times, signaling that the system has reached equilibrium.

This illustrates how a closed system approaches equilibrium. The particles spread to fill the available states (in this case the available space) uniformly, thereby maximizing the entropy. This tendency to maximize the entropy is not built into the microscopic equations of motion. Rather, it occurs because the system explores (or spends time in) all of the available states with equal probability. This eventually makes the probability of finding a cream particle the same in all of the grid cells. The equilibrium condition is one in which all available states have equal probabilities.

According to the *ergodic hypothesis*, all of the available states of a closed system in equilibrium will be occupied with equal probabilities. This is not a result of Newton's laws or any other microscopic equations of motion. Rather, it is a *hypothesis* that plays a key role in statistical physics. While it has not been possible to derive this hypothesis from microscopic laws or principles, it has been shown to hold (rigorously) for certain systems. The difficulty with deriving the ergodic hypothesis in a completely general way can be appreciated from the snapshots of the cream in Figures 7.13 and 7.14. The picture at $t = 10^6$ shows a fairly random distribution of particles, which contrasts greatly with the completely ordered picture at $t = 0$. If we were to follow the system beyond $t = 10^6$, we would

expect to find the randomness of this distribution to either stay the same or increase; we would certainly not expect it to decrease. However, such a decrease would not violate any microscopic laws of nature. These laws leave open the possibility that a cream distribution such as the one at $t = 10^6$ might evolve with time into a perfectly ordered arrangement like that at $t = 0$. That is, the cream could "unmix" and all flow to the center of the cup. However, this is extremely unlikely and, so far as we know, has never been observed in nature.

In the present example the ergodic behavior is a result of the rules of our random walk. The fact that each step is independent of the previous steps leads the particles to explore all parts of the cup with equal probability. However, this assumption of independence is just that, an *assumption*. We know that on a microscopic level the trajectory followed by a particle is not independent of its prior history, but could in principle be calculated (using, for example, molecular dynamics as we will discuss in Chapter 9). So how can we explain why the ergodic hypothesis is so widely applicable? The answer to this question is not completely settled, but one attractive possibility can be seen from our work on chaotic systems. There we saw that deterministic systems can behave chaotically and exhibit an extreme sensitivity to initial conditions.²² This sensitivity leads to essentially random behavior of systems such as the pendulum, asteroids near a Kirkwood gap, etc. It may be that this essentially random behavior is responsible for the ergodic behavior observed in nature. With this in mind, it is intriguing to reconsider the billiard problem discussed in Chapter 3. That can be viewed as a (classical) model for the motion of gas molecules in a closed container and is thus very similar to our cream-in-your-coffee problem. You may recall that except for very specially shaped containers, the motion of the billiard is chaotic. Thus it would not be surprising to find that the motions of our cream particles are also chaotic, making the system ergodic.

EXERCISES

- 7.12. Calculate the entropy for the cream-in-your-coffee problem, and reproduce the results in Figure 7.16.
- 7.13. Calculate S as a function of time for the cream-in-your-coffee problem for containers with different sizes. Show that the time necessary to reach equilibrium varies as the square of the size.
- *7.14. Perform the random-walk simulation of Figures 7.13 and 7.14 and show that the size of the drop of cream increases as $t^{1/2}$ (our familiar diffusive behavior), so long as the drop is smaller than the size of the container. Show that the behavior changes when the drop has spread so much that it uniformly fills the container. The time at which the size of the drop stops increasing should be the same as the time at which the system reaches equilibrium as determined by the entropy. Hint: A convenient measure of the size of the drop of cream is the root-mean-square distance of the particles from the origin, $\sqrt{(\sum r_i^2)/N}$.
- *7.15. Perform the random-walk simulation of spreading cream (Figures 7.13 and 7.14), and let one of the walls of the container possess a small hole so that if a cream

²²They are also extremely sensitive to changes in external parameters, which would also contribute to ergodicity

particle enters the hole, it leaves the container. Calculate the number of particles in the container as a function of time. Show that this number, which is proportional to the partial pressure of the cream particles varies as $\exp(-t/\tau)$, where τ is the effective time constant for the escape. Hint: Reasonable parameter choices are a 50×50 container lattice and a hole 10 units in length along one of the edges.

- 7.16. Carry out an analysis of the entropy for the nonlinear damped pendulum studied in Chapter 3. Consider the behavior of $\theta(t)$ and divide the possible range for θ into a number of cells (try 100). Simulate the pendulum and calculate a histogram of the number of times the pendulum angle falls into a cell as a function of θ ; sample $\theta(t)$ in synchrony with the drive force, as we did in calculating the Poincaré sections. Calculate the entropy using (7.24) as a function of the driving force. You should find that S is small in the periodic regime and large when the pendulum is chaotic. What is S in the period-2 and period-4 regimes?

7.6 CLUSTER GROWTH MODELS

We have spent a good deal of time in this chapter exploring random walks and their connection with diffusion and the approach to equilibrium. Another interesting random process, which turns out to be closely related to random walks, concerns the growth of clusters, such as snowflakes and soot particles. In this section we will examine two different models of cluster growth. The first is known as the Eden model and operates according to the following rules. Consider a two dimensional lattice of points (x, y) , where x and y are both integers. These are the allowed locations for the particles that will make up the cluster. We begin by placing a seed particle at the origin $(x = 0, y = 0)$; this is our initial cluster. A cluster grows by the addition of particles to its perimeter. Our initial cluster has nearest-neighbor points on the lattice at $(\pm 1, 0)$ and $(0, \pm 1)$. We will refer to such still unoccupied sites that are nearest neighbors of occupied sites as the perimeter sites of the cluster. We next choose one of these perimeter sites at random and place a particle at the chosen location. The cluster now contains two particles and a correspondingly larger perimeter. This process is then repeated; a perimeter site is chosen at random (i.e., all perimeter sites have the same probability of being chosen), and a particle added at that location. We continue this process until a cluster of the desired size is obtained. This is the Eden model of cluster growth.

A typical Eden cluster is shown in Figure 7.17. While it is a little rough around the edges, it is basically a circular disc with a few holes. Note that as the cluster grows these holes tend to fill in, since they are treated on the same footing (they are equally likely to be occupied by the next particle) as the exterior perimeter sites.

The Eden model is sometimes referred to as a “cancer” model, because the clusters grow from within by expanding their borders. However, not all clusters in nature grow in this manner. For example, snowflakes and soot particles grow by the addition of new particles that originate from outside the cluster.²³ This process is captured by a different cluster model, which is known as diffusion-limited aggregation, or DLA.

²³More precisely, the places where new particles are added depend on processes that take place outside the cluster

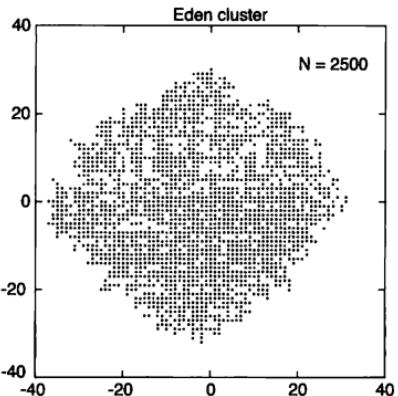


FIGURE 7.17: Eden cluster containing 2500 particles Note that there are very few "holes" inside the main body of the cluster

The growth rules for DLA clusters are as follows. We again start with a seed particle at the origin. We then release a particle at a randomly chosen location (x, y) that is some distance away from the seed and let it perform a random walk. If (or when) this walker lands on a perimeter site, it sticks there and becomes part of the cluster.²⁴ This process is repeated with many walkers until a large cluster is grown. One way to motivate (or justify!) the choice of these growth rules is to consider how a large particle might be built up from smaller particles or molecules in a solution. If the cluster is located well away from any other objects, such as walls or other clusters, small particles will approach it from all directions. In addition, it seems reasonable to assume that small particles will move diffusively as they travel through the solution around the cluster. This process is captured in the DLA growth rules since, as we have already seen, diffusion is equivalent to a random walk. Of course, we can imagine situations in which these rules would not be appropriate. For example, there might be some localized source of the small particles, or perhaps a prevailing current, that would give an overall drift velocity in addition to the random walk. These are perfectly reasonable models and each could be interesting depending in part on possible connections to real systems.

A cluster grown using the DLA rules is shown in Figure 7.18 (we will consider the programming associated with generating such clusters in the exercises). Comparing our DLA cluster with the Eden cluster Figure 7.17, it is obvious that

²⁴We may also let the particle stick permanently at a perimeter site with some specified probability that is less than unity (or only after a certain number of visits to that site). Such a model may better describe a situation where the sticking process represents e.g., a chemical reaction, which occurs probabilistically. A problem of this type is explored in the exercises

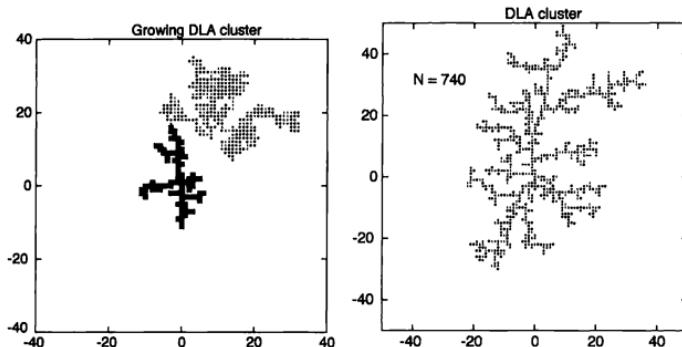


FIGURE 7.18: Left growing DLA cluster. The filled squares are sites in the cluster and the dots are the lattice sites visited by a particular random walker as it approached. This walker eventually touched the perimeter at the top edge of the cluster and became attached there. Right DLA cluster containing 740 particles.

they have very different properties. The Eden cluster is, as we have already noted, essentially a solid disk with very few holes and a fairly smooth perimeter. In contrast, the DLA cluster contains many large open spaces and the perimeter is very irregular. These differences are directly connected with the growth rules. For the Eden clusters all perimeter sites, even the interior ones, are equally likely to be filled by the next particle. This tends to fill in any holes or cracks, since those were likely formed long before the outermost parts of the cluster. For DLA it is extremely unlikely that such crevices will be filled in, as the probability that a random walker will manage to navigate past the outermost parts of the cluster on its way deep into a crack is very low. A walker is much more likely to first make contact with the outer edges of the cluster.

This intuitive explanation of the difference between Eden and DLA clusters is useful, but we would like to have a quantitative measure of this difference. This brings us to consider objects that are known as *fractals*, which will be our primary topic for the remainder of this section and the next one as well. Rather than try to give a very general definition of what it means to be a fractal, we will instead introduce a few terms and concepts associated with these objects. A definition will gradually emerge as we proceed.

Let us consider how we might measure the dimensionality of an object. At first this may seem like a silly exercise. Your intuition tells you that straight lines are one-dimensional objects, flat disks are two dimensional, etc. But what about a piece of spaghetti, or a string that is tangled, or a piece of crumpled paper? While your intuition probably would still feel comfortable in these cases, it is instructive to construct an *operational* definition for dimensionality. There are several ways to

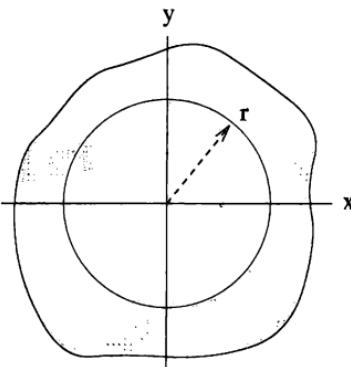


FIGURE 7.19: Method for calculating the effective dimensionality of a cluster, shown as the shaded region $m(r)$ is the mass contained within a circle of radius r .

approach this problem. In the next section we will discuss simple curves and other objects that are close to being one dimensional. Here we will consider the problem for our Eden and DLA clusters.

Suppose we have a large disk of uniform density that lies in the x - y plane, as illustrated in Figure 7.19. If we consider the mass of the disk that is contained within a circle of radius r , it is easy to see that this is given by

$$m(r) = \sigma \pi r^2 , \quad (7.25)$$

where σ is the mass per unit area and r is small enough that the test circle is entirely contained within the disk. The key point is that the mass scales as r^2 , and this 2 is also the dimensionality of the object. If we instead had a straight line or a similar type of curve, the mass would be

$$m(r) = 2 \lambda r , \quad (7.26)$$

where λ is the mass per unit length and r is again small enough that the circle does not go beyond the ends of the line. The mass now scales as r^1 , and 1 is again the spatial dimensionality of the object.²⁵

These observations form the basis of an operational definition that we can use to calculate the effective dimensionality of a cluster. Our definition is

$$m(r) \sim r^{d_f} , \quad (7.27)$$

²⁵Strictly speaking, objects must be of infinite size for the dimensionality obtained in this way to be correct to all scales of r . In that sense, any finite object becomes zero-dimensional at a sufficiently large scale of r .

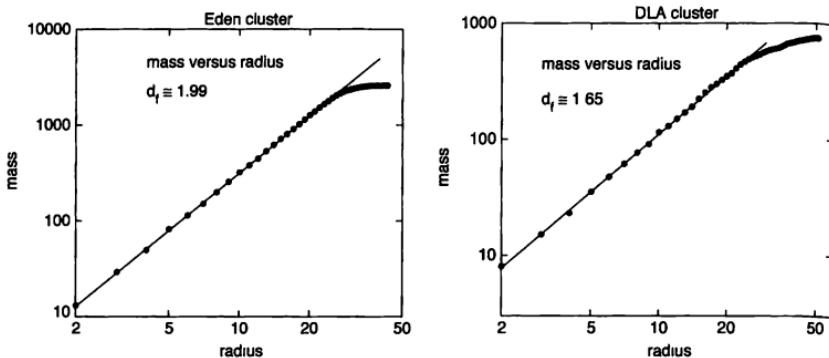


FIGURE 7.20: Left: plot of $\log m$ versus $\log r$ for the Eden cluster in Figure 7.17. The solid line is a least-squares fit whose slope is the fractal dimensionality, which for this cluster was $d_f \approx 1.99$, it was equal to 2 to within the statistical uncertainties. Right: plot of $\log m$ versus $\log r$ for the DLA cluster in Figure 7.18. The solid line is again a least-squares fit whose slope is the fractal dimensionality, which for this cluster was $d_f \approx 1.65$.

where d_f is the effective or *fractal* dimensionality of the object. We have already seen cases that yield $d_f = 1$ (a simple line or curve), and $d_f = 2$ (a solid disk); a solid sphere would be described by $d_f = 3$. It remains for us to devise objects for which d_f is not an integer.

To apply this definition to one of our clusters, we need to calculate the mass inside a circle of radius r , which is centered inside the cluster. For convenience we choose the position of the initial seed particle as the center and call this point the origin.²⁶ Assuming that all of the particles have the same mass, we can find $m(r)$ by counting the number of particles within a distance r of the origin. Values for the mass as a function of r for our Eden and DLA clusters are shown in Figure 7.20, where we have plotted the results on logarithmic scales. Such plots are useful, since taking the logarithm of both sides of (7.27) yields

$$\log m \sim d_f \log r , \quad (7.28)$$

so the slope of a log-log plot is equal to the fractal dimensionality.

For both types of clusters the results for $\log m$ versus $\log r$ are consistent with a straight line and thus with the relation (7.28) for small r . However, the curves flatten out for large r . This is due to the finite size of the clusters. For very large

²⁶Strictly speaking, the definition (7.27) should be applied with many test circles, with centers chosen at all possible locations within the cluster. The notion and value of an effective dimensionality should not depend on the choice of a specific "center." However, for simplicity in our numerical calculations, we will use the seed particle location as the center. The interested (or skeptical) reader is encouraged to calculate the dimensionality using other choices.

measuring circles (see Figure 7.19) the entire cluster will be inside the circle, and in this case $m(r)$ will be independent of r . For the same reason, when r is only a little less than the maximum "radius" of the cluster, r_{\max} , $m(r)$ will be suppressed below its value for the ideal case; that is, for an extremely large (infinite) cluster. In practice, a particular cluster can only be used to estimate $m(r)$ for distances up to about $r_{\max}/2$.

The solid lines in Figure 7.20 are least-squares fits of (7.28) to the results for $m(r)$ out to $r_{\max}/2$. The slopes of these lines are the fractal dimensionalities, and we find $d_f \approx 1.99$ for the Eden cluster and 1.65 for the DLA cluster. To within the statistical errors the Eden cluster has a dimensionality of 2. This is in accord with our intuition; the Eden cluster is essentially just a solid disk. However, the DLA cluster has a fractal dimensionality much less than 2 (and also much greater than 1). Indeed, this is why it is known as a fractal.

In order for a cluster to have an effective dimensionality d_f , which is not an integer, its mass must increase more slowly²⁷ than r^2 . This means that it must contain holes or cracks, as we have observed in the DLA clusters. However, simply containing such open spaces is not enough. A planar object that has a certain, constant fraction of open space would still have $d_f = 2$. In order to have $d_f < 2$, the sizes of these open spaces must *increase* with r . Evidently, DLA clusters have just this property.

EXERCISES

- 7.17. Write a program to generate DLA clusters and calculate their fractal dimensionality. Here are a few programming suggestions for this problem. We have already seen that random walkers can take a long time to move an appreciable distance and this can make the generation of a DLA cluster very slow if you are not careful. Initially, start new walkers a distance r_{start} away from the origin, by choosing the initial position of a walker at random on a circle of radius r_{start} (but make sure that they are on the lattice).²⁸ If the walker wanders too far from the cluster, say farther than $1.5 \times r_{\text{start}}$, it may never hit the cluster, so a new walker should be started. As the cluster grows, r_{start} should be increased so that the walkers don't begin too close to the cluster. Try keeping r_{start} at least 5 units larger than the maximum cluster size (that is, the point on the cluster which is farthest from the origin). Also, when the walker is far from the cluster you can let it take steps of length 2 (to speed up the walk), then decrease the step length as it approaches the cluster.
- 7.18. Grow a DLA cluster using the algorithm described in the previous problem, but instead of letting the walkers start from points on a circle that surrounds the cluster, have all of the walkers begin at a location on the x axis. How does this affect the shape and structure of the cluster?
- 7.19. Generate a DLA structure using an initial "seed," which is the entire x axis. That is, begin with all of the sites on the x axis occupied and let the walkers begin some distance above this axis. The resulting structure is sometimes used

²⁷We assume here that the cluster is grown on a planar lattice and not a three-dimensional one.

²⁸Be sure that you pick the initial location of the walker at random from possible locations on the circle. This is most easily done by choosing an angle at random in the range 0 – 2π and using it to specify the starting point of the walker.

- to model the paths followed by electric discharges in a gas (that is, lightning bolts).
- *7.20. Repeat the previous problem, but allow your random walkers to move on a *three-dimensional* lattice. You should find a value of $d_f \sim 2.5$ in this case.
 - 7.21. Generate a DLA cluster as above but let the walker stick to the cluster only with a probability $p < 1$, say, $p = 0.1, 0.3, 0.6$, etc. Describe any trends that you find in the fractal structures as p is varied.
 - 7.22. Generate a DLA cluster using walkers that perform a biased random walk. That is, let your walkers have a higher probability for walking in one particular direction (along the $+x$ direction, for example) than in other directions. This is a biased random walk, as we have considered in an earlier exercise. Study how both d_f and the overall shape of the cluster depend on the magnitude of this drift velocity.
 - *7.23. An interesting variation on DLA is to begin with a lattice in which some fraction of sites are occupied with particles, and then let the cluster diffuse and pick up particles as it makes contact with them. Use a square lattice and place particles on sites at random with some probability ($p = 0.1$ is a good choice). Let the cluster perform a random walk and whenever a perimeter site is occupied by a particle, that particle then becomes part of the cluster. Generate clusters in this way and calculate their fractal dimensionality. You should find $d_f \sim 1.7$, which is about the same as a DLA cluster. Interestingly, d_f for this cluster-diffusion model seems to vary with p . Calculate d_f for other values of p and show that d_f becomes larger (it should approach ~ 1.95) for large values of p . This calculation was first performed by Voss (see Voss [1984]).

7.7 FRACTAL DIMENSIONALITIES OF CURVES

While DLA clusters may be nice to look at, we must still ask what it is about fractals that makes them interesting from a *physics* point of view. We will discuss this question in due course, but it is useful to first to consider another problem concerning fractal objects. It is convenient to introduce this problem using a class of regular fractals that are known as Koch curves. In contrast to the fractal clusters grown using the DLA model, Koch curves are generated by *deterministic* rules. While such regular fractals do not have a direct connection with physics, they are useful for learning more about fractals, as we will now see.

Perhaps the simplest way to define a Koch curve is through the examples in Figure 7.21, which shows a family of such curves. The first member of the family is shown at the bottom and is just a straight line of length L ; we will refer to this as a Koch curve of order one. The second-order Koch curve (the second curve from the bottom) is derived from the first-order curve by replacing the straight section with four segments of length $L/3$, oriented with respect to the original (first-order section) as shown. The third-order curve is obtained from the second-order curve by replacing *each* of its straight sections by four more segments, with lengths $L/9$. The fourth and higher-order curves are obtained in an analogous manner. This procedure can be used to obtain curves of arbitrary order.

The Koch curves are thus defined *recursively*. A member of the series is generated from the preceding member by replacing each of its straight sections by four new segments, as described above. We can clearly generate other types of

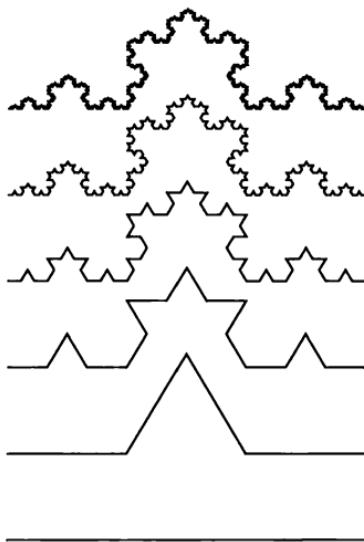


FIGURE 7.21: A family of Koch curves generated at several different levels of recursion. The first-order curve is shown at the bottom, the second-order curve is the second curve up from the bottom, etc.

Koch curves by using different replacement rules. However, the key (and common) property of such curves is that at “infinite” order they will look the same on *all* length scales. That is, no matter what scale or magnification we use to examine such a curve, it will always have the same appearance. This is a property of fractal curves that is not found in regular, nonfractal objects. You may not think this to be very impressive; after all, it can only be a property of a regular fractal. A DLA cluster is a random fractal and will not have precisely the same appearance on all length scales. However, a DLA fractal will have significant structure on all length scales, and we have argued above that it is just this property that leads to a fractal dimensionality that is less than 2. The important point is that a DLA fractal will have the same general appearance, the same structure of open spaces and cracks, and hence the same value of d_f , on *all* length scales.

Let us now consider the effective dimensionality of a Koch curve. As in the previous section, we are after an operational definition. A procedure involving $m(r)$, such as the one we employed to compute d_f for the Eden and DLA clusters, could be used here, but we will instead introduce another approach that is a bit more natural for objects that are close to being one dimensional. Imagine that the Koch curves are walking paths and that you are moving in steps of length L_s along one of the high order Koch curves in Figure 7.21. We define the effective length of the curve to be the number of steps, N_s , required to walk from one end to the other, multiplied by the length of each step, that is, $L_{\text{eff}} = N_s L_s$. For an ordinary curve this product is a constant,²⁹ which means that the length of the curve does not depend on the size of the steps you use to measure it. However, the results for a high-order Koch curve are a bit different. For a given L_s , a step along a Koch curve will pass over the fine structure present at scales smaller than L_s . As the step length is made smaller, more and more of this structure becomes apparent, causing an increase in the number of steps required. For this reason the effective length $N_s L_s$ now depends on the size of the step.

Another way to think of this is to imagine that you are examining a Koch curve through a microscope. At low magnification much of the fine structure is blurred and the curve appears as one of the low order curves in Figure 7.21. As the magnification is increased, more structure is apparent and the effective order of the curve increases. This extra visible structure will make the curve appear longer than it does at lower magnification. If the magnification is increased further, even more structure becomes visible, etc. For a fractal curve, increasing the magnification always reveals more structure and thus a longer curve.

This operational definition of length can be used to define the effective or fractal dimensionality of a curve. For an ordinary nonfractal curve the number of steps would vary as L_s^{-1} , where the factor of 1 is just the dimensionality in this case. We therefore define d_f through

$$L_{\text{eff}} \equiv N_s L_s \sim L_s^{1-d_f}. \quad (7.29)$$

²⁹Strictly speaking, it is a constant only when L_s is smaller than any of the structure in the curve. For an ordinary (nonfractal) curve it always possible to choose a suitably small value of L_s , but for fractal curves there is structure on *all* length scales, so this is not possible.

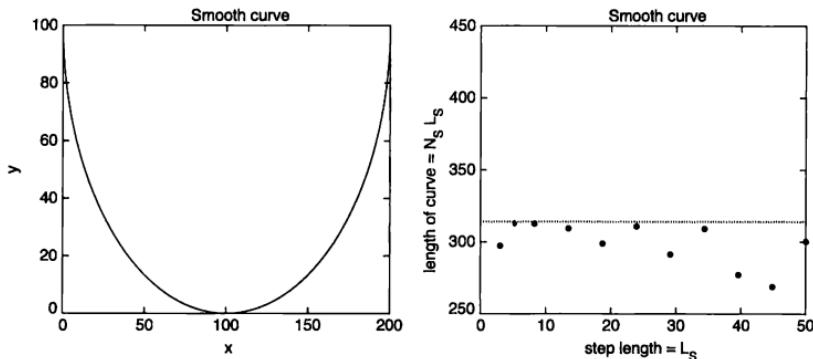


FIGURE 7.22: Left typical smooth curve Right length of the curve at left, calculated by "walking" along the curve, as a function of the length of the walker's steps. The dotted line shows the exact value of the length. The calculated values fall slightly below the exact result due to the manner in which the finite step size was handled (this issue is explored in the exercises)

For the regular Koch curves in Figure 7.21 this expression can be evaluated analytically. If the length of the first-order curve (the bottom one) is unity, then for a step of length $L_s = 1$, $N_s = 1$ steps will be required. For $L_s = 1/3$, the number of steps will be $N_s = 4$, etc., for step lengths of $1/9, 1/27, \dots$. Inserting this into (7.29) yields $d_f = \ln 4 / \ln 3 \approx 1.262$. Since $d_f > 1$, this confirms that the Koch curve is indeed a fractal.

For random fractals or cases where only the coordinates of the curve are known, (7.29) can be used as the basis of a numerical approach. Here we consider two cases. The first is the ordinary nonfractal curve shown in Figure 7.22; it is just a semicircle. For this example we calculated the coordinates of the curve at 1000 points and used these coordinates in a calculation of the number of steps required to traverse the entire curve. We will leave a discussion of the programming to the exercises. The number of steps required, N_s , was calculated as a function of the step length, and the value of N_s then used to obtain the effective length of the curve. The results are shown in Figure 7.22. The effective length is seen to be essentially a constant, independent of L_s , as expected for a nonfractal curve. The scatter in the results for L are due to the finite step lengths (this is also why the values fall a few percent below the exact value), the effects of which are explored in the exercises.

We have repeated this calculation for the random fractal shown in Figure 7.23. This was generated using the Koch algorithm, but instead of replacing each straight section by four simple (but shorter) segments, we have chosen the lengths and angles of these four segments at random. (A program for generating such random Koch curves is given at this book's WWW site.) The effective length of this curve as

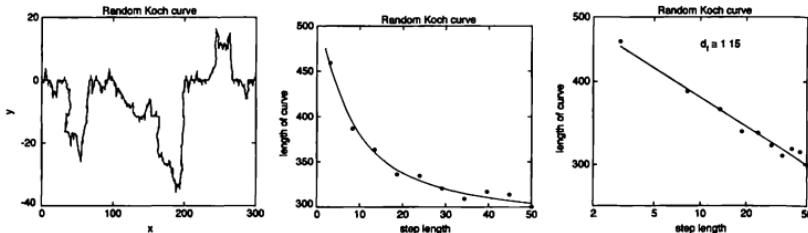


FIGURE 7.23: Left: random Koch curve generated using 5 levels of recursion. Middle: length of the curve at left, calculated by walking along the curve, as a function of the length of the walker's steps. Right: same plot as the middle graph, but on log-log scales. A least-squares fit to (7.29) gives the solid line, whose slope corresponds to $d_f \approx 1.15$.

a function of step size, L_s , is also shown. It is seen that L_{eff} increases as L_s is made smaller, as expected for a fractal curve. Smaller steps allow more of the curve's structure to be discerned, yielding a longer measured length. The slope of the log-log plot of L_{eff} as a function of L_s yields the fractal dimensionality from (7.29), and we find $d_f \approx 1.15$. This is not far from the value we found above for the regular Koch curve (the difference is probably due to the effects of finite step size combined with the uncertainty associated with the least-squares fit in Figure 7.23), suggesting that randomizing the curve in this simple manner does not significantly affect d_f .

Nature contains many examples of random fractal curves. Empirically it has been found that the coastlines of most countries are fractal (see Mandelbrot [1977] for more on this), as are the shapes of many rivers. Much effort has been (and is still being) devoted to trying to understand why nature seems to prefer such fractal shapes.

EXERCISES

- 7.24.** Use (7.29) to calculate the fractal dimensionality of various curves, including smooth curves such as the one shown in Figure 7.22, and fractals such as the one shown in Figure 7.23. A program that generates such fractals is given in Appendix 4. Programming suggestions: To implement this algorithm you must first obtain the coordinates of the curve at a large number of points; 1000 will typically be enough. To calculate d_f , start at the initial point, (x_1, y_1) , and then walk down the list of points until you reach one, call it (x_2, y_2) , which is a distance L_s or greater from the starting point. In general the distance from (x_1, y_1) to (x_2, y_2) will not be precisely L_s . That is, the end of the step will fall between coordinates. You can deal with this in three different ways. One way is to effectively back up the step to the previous point and thereby use a step length slightly smaller than L_s . Another is to move on to the next point and thus use a step that is slightly larger than L_s . The third option is to let the step land in the middle and use a landing spot that interpolates between the two adjacent

points. Try all three in your program, and show that they lead to similar values of d_f , provided that the spacing between points is small compared to L_s . This finite step effect is the cause of the scatter and the deviations from the expected values in Figure 7.22.

- 7.25.** Use the algorithm associated with Figure 7.19 to calculate the fractal dimensionality of the collection of lattice sites visited by a three-dimensional random walk. Such a walk will often revisit sites, so be sure that you do not count a visited site more than once when calculating the mass of the cluster. Also, you should repeat the calculation for a number of different walks so that you can estimate the average fractal dimensionality.
- *7.26.** Write a program to generate fractal curves recursively. In the Koch example discussed in this section we used an equilateral triangle as the basic "unit." Generate fractals with other basic units; interesting possibilities include squares, pentagons, and random multidisted objects. *Hint:* One approach to recursive programming is to structure the algorithm schematically as follows.

EXAMPLE 7.3 Subroutine fractal_curve, a recursive algorithm to plot a fractal curve

- Input: (x_i, y_i) and (x_f, y_f) , starting and ending points of an interval, and *level*, number of levels of recursion desired.
 - If *level* = 1, this is the last level, and no further recursion is needed. In this case, simply plot the line segment and return to the calling program. (The calling program may be `fractal_curve` itself if used recursively.)
 - If *level* > 1:
 - ▷ Calculate the intervals required for the next level using (x_i, y_i) and (x_f, y_f) .
 - Example: Suppose that two intervals are needed at the next level, one with end points $(x_i, y_i), (x_c, y_c)$, and the other with end points $(x_c, y_c), (x_f, y_f)$ where $x_c = x_i + (x_f - x_i)/2 - (y_f - y_i)/2$, $y_c = y_i + (x_f - x_i)/2 + (y_f - y_i)/2$.
 - ▷ Recursively call `fractal_curve` for each of the new intervals with the required number levels decremented by 1.
-

The routine `fractal_curve` is called with the coordinates of the beginning and ending points of the curve and the desired number of recursive "levels"; this routine serves two intertwined functions. If the level number is 1, then `fractal_curve` simply draws a line between the starting and ending points. If the level number is greater than 1, `fractal_curve` calls itself but with different arguments. These arguments are new starting and ending points, which will generally be more closely spaced than the previous ones, and the way these new points are chosen will determine the pattern of the fractal. Note also that each time the `fractal_curve` routine is called, the level number is reduced by one so that the recursive scheme will eventually terminate. This particular version of `fractal_curve` will plot a simple fractal. We suggest that you try to predict what the pattern will look like for n of 2–5 before you run the program. Random fractals can be generated by choosing the new starting and ending points randomly.



FIGURE 7.24: Left: 40×40 square lattice of sites occupied with probability $p = 0.2$. Some typical clusters are circled. Right: same, but with $p = 0.4$.

7.8 PERCOLATION

The motion of groundwater through soil, the strength of a porous network, and the flow of oil through porous rock, are all problems that fall under the general heading known as percolation. The most basic percolation problem is illustrated in Figures 7.24 and 7.25, where we have a square lattice of sites with each site occupied at random according to a certain probability, p . Here the occupied sites are plotted as filled squares while the unoccupied sites are left uncolored. It is useful to group the occupied sites into clusters according to the following rule. If two neighboring sites are both occupied, that is, if two filled squares have a common edge, they are considered to be part of the same cluster. Membership in a particular cluster extends to all sites that share an edge with at least one other member of the cluster. Hence, a cluster is simply a collection of interconnected sites.

For small occupation probability, such as $p = 0.2$ in Figure 7.24, nearly all of the occupied sites are isolated, so the majority of clusters are of size 1, just single sites. When p is increased to 0.4, most sites are connected to several others and the typical cluster contains 5–10 sites. At the opposite extreme, illustrated by $p = 0.8$ in Figure 7.25, it is rare to find occupied sites that are not part of a large cluster. In fact, for large p nearly all sites belong to the *same* cluster, which extends throughout the lattice. The most interesting structure is found for $p = 0.6$. Here many sites are members of fairly large clusters, but these clusters are often barely connected. That is, the removal of a single site would change the size of the cluster drastically. Nevertheless, these barely connected clusters seem to be the rule. If you look carefully you will find that in Figure 7.25, with $p = 0.6$, one such cluster spans the entire lattice, as it touches all four edges of the lattice. Such a cluster is said to be a *spanning cluster*,³⁰ and a lattice that possesses such a cluster is said to “percolate.”

³⁰While this is a good way to define spanning, in many situations (particularly for large lattices), requiring that a cluster touch only two opposite edges may be equivalent



FIGURE 7.25: Left. 40×40 square lattice of sites occupied with probability $p = 0.6$, right same but with $p = 0.8$

These clusters, particularly the spanning cluster, will turn out to be central to many aspects of the behavior of a percolating system.³¹ It is interesting to consider the size of typical clusters as a function of p . We have already seen that for large p we are essentially guaranteed to have a spanning cluster, while for small p the odds are that such a cluster will not occur. It turns out that the transition from one regime to the other is a sharp one. For an infinitely large lattice this transition occurs at a critical concentration, p_c , whose value depends on the lattice structure. For the two-dimensional square lattice considered here, $p_c \approx 0.593$, so the example with $p = 0.6$ is essentially right at the percolation threshold. The problem of calculating p_c is a bit different from anything we have encountered so far. Estimating the percolation threshold requires that we determine whether or not a spanning cluster exists, and the answer to this question involves a global examination of the system. The nature of the problem can be appreciated if we imagine that we start with an empty lattice and then occupy sites (chosen randomly) one at a time. We terminate the process when a spanning cluster first appears and the concentration of occupied sites at that instant is p_c . The last site to be occupied will provide a connection between two or more clusters that then combine to make up the final spanning cluster. However, simply examining this last site and its immediate surroundings will not tell us that it was the missing link in the spanning cluster. The information required to determine if a cluster spans the lattice or not is effectively distributed throughout the lattice.

The spanning question is really a sort of pattern recognition problem, and any computer scientist will tell you that such problems can be very difficult to solve. One way to handle this problem is to simply examine the percolating system by eye, as we did in discussing the results in Figures 7.24 and 7.25; in effect, using

³¹For example, if the occupied sites are voids in an otherwise solid rock, then the flow of a fluid (such as oil) through the rock will be controlled by the properties of the spanning cluster.

your brain as the pattern analyzer. While this method works well in practice for lattices up to about 50×50 in size, it can become tedious. One efficient numerical algorithm for detecting spanning clusters involves labeling the sites in each cluster with a unique cluster number. Then, to check for the existence of a percolating cluster we determine if the cluster number of any of the edge sites is found on all four edges; if so, that number corresponds to the spanning cluster.

In a particular implementation, we may choose to occupy sites sequentially but at a random location, checking for (and assigning, if necessary) the site's unique cluster number each time a site is occupied, and stopping when a spanning cluster is found. A straightforward implementation of the labeling scheme may proceed as follows:

EXAMPLE 7.4 A first algorithm for cluster labeling

- Begin with an empty lattice and initialize all sites to a cluster number of 0 indicating *unoccupied*.
- Select and occupy a random site, labeling it with a cluster number of 1, since it is the first cluster.
- A second site is chosen at random and its neighboring sites are checked to see if any of them happens to be the site we just labelled with the cluster number 1.
 - ▷ If not, the new site is labelled as cluster number 2.
 - ▷ If yes, the new site is also labelled as cluster number 1.
- A next site is chosen at random and its neighboring sites are checked to see if any of them is already occupied.
 - ▷ If not, the new site is labelled with the next cluster number.
 - ▷ If yes:
 - If there is only one cluster number among all the neighboring occupied sites, then give this same number to the new site.
 - If there is more than one distinct cluster number among the occupied neighbors, then this new site *bridges* multiple existing clusters. In this case, we must choose a common, unique cluster number³² for the resulting bridged cluster, give this number to the new site, and *relabel all the sites in the merged cluster with the same number*.
- Continue with more sites until there appears a common cluster number on sites of all 4 edges of the lattice. The fraction of the occupied sites among all the lattice sites at this point is the estimate of p_c for this particular percolation sequence.

³²It is convenient, but certainly not required, to use the smallest of the cluster numbers for the new larger cluster

While this algorithm is straightforward, it is extremely CPU intensive due to the frequent relabeling of existing clusters which occur near p_c . As the percolation threshold approached, many small clusters merge to eventually form a spanning cluster. Each time a merger occurs, this algorithm must travel through the new merged cluster and change many of the labels. If S is the number of lattice sites, then this is roughly an order S^2 algorithm, which means the CPU time requirement varies approximately as S^2 as the lattice size S ($= L^2$ in two dimensions) grows. This inefficiency is a crippling fault, which led Hammersley (one of the researchers credited with defining percolation as an important physical and mathematical problem), to state (in 1964) that the solution of this problem by direct simulation is "out of the question." Of course computing hardware has progressed greatly since then, but the real breakthrough came with the development of a better algorithm. Hoshen and Kopelman (1976) proposed what they called a "cluster multiple labeling technique," which avoids the expensive relabeling entirely, and improves the CPU requirements to something proportional to S^1 . In this technique, each time a cluster merger occurs, the cluster picks up an additional label so that multiple labels remain associated with each cluster. A correspondence table is then set up. Since this is a table of cluster numbers and not of the individual sites, searching through them is much quicker than relabeling every site of a cluster, and this process can be done efficiently by setting it up as a directed tree. Thus, the relabeling portion of the Hoshen-Kopelman algorithm (the part of the algorithm contained in the box in Example 7.8) can be transformed to something like the following.

EXAMPLE 7.5 Improved cluster labeling scheme

- Label the new bridging site by the chosen cluster number, say, n . This label is called a *proper label*.
- Let the other previously assigned cluster numbers, say, m_1, m_2 , etc. (of those parts which merged to form the current cluster) to point to n . This can be accomplished, say, by using an array of cluster labels $Clus()$ and assigning

$$Clus(m_1) = -n, \quad Clus(m_2) = -n, \quad Clus(n) = s_n,$$

where the negative sign is used here to indicate that m_1 and m_2 are *not* proper labels, but point to another label n , while n is a proper label because $Clus(n) > 0$. Its value s_n is a (positive) number which can be used to store some property of interest of the cluster such as the cluster size.

- When checking for merging of clusters, we first get the proper cluster label of the neighboring occupied sites and use these. That is, search the array $Clus(m)$ recursively (starting with m being the cluster number assigned to the neighboring site in question) until its value is positive. E.g., if the label is 151, we look at $Clus(151)$. If it is negative, say, -35, then we further look at $Clus(35)$, etc. until we find the proper label n where $Clus(n) > 0$.

We now consider some of the results. Let us first return to the problem posed above, the calculation of p_c . Strictly speaking, p_c is the smallest concentration at

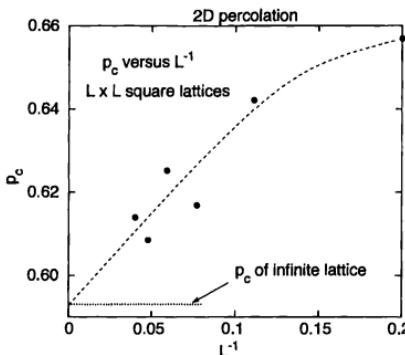


FIGURE 7.26: Results for the critical concentration as a function of lattice size for $L \times L$ square lattices. For each lattice size we have averaged the results for 50 different simulations. The dotted line shows the value of p_c for an infinitely large lattice, while the dashed curve suggests a smooth extrapolation to the infinite lattice limit.

which a percolating cluster is first found for an *infinitely* large lattice. For the finite lattices used in a simulation, the concentration at which a spanning cluster first appears will fluctuate statistically from one simulation to the next.³³ To obtain an estimate of p_c we must average the values for many different lattice realizations, and the results of such a calculation are shown in Figure 7.26. Here we plot the average values of p_c obtained for a square lattice as a function of L^{-1} for a series of $L \times L$ lattices. There is a small but noticeable variation of p_c with L (note that the vertical scale is greatly expanded), and by plotting our results in this manner we can more easily extrapolate to the case of an infinite lattice. The dotted line shows the result for an infinitely large lattice, $p_c = 0.593$, and while there are some statistical fluctuations in our values, they do appear to be heading to the expected target.³⁴

The behavior in the vicinity of the percolation threshold, i.e., near p_c , is an example of a second-order *phase transition*.³⁵ The percolation transition involves a change from a macroscopically *connected* phase, in which a single cluster spans the system, to a *disconnected* phase containing only small, non-spanning clusters. Near this transition, various properties exhibit singular behavior, and these singularities are often described mathematically by power laws.

³³The value derived for p_c for a finite lattice will also depend on the definition of spanning, e.g., whether we require that the cluster touch all sides of the system, or just two opposite sides. We expect that this difference will disappear when we extrapolate to an infinite lattice.

³⁴The manner in which the effective $p_c(L)$ for a finite lattice converges to the true p_c can be analyzed in terms of a theory called *finite-size scaling*. According to this theory, the deviation $\Delta p_c(L) \equiv |p_c(L) - p_c|$ converges toward 0 as $L^{-1/\nu}$ where ν is a critical exponent (see below).

³⁵We will see that it has much in common with other types of phase transitions, such as the ferromagnetic-paramagnetic and melting transitions which we will discuss in Chapters 8 and 9.

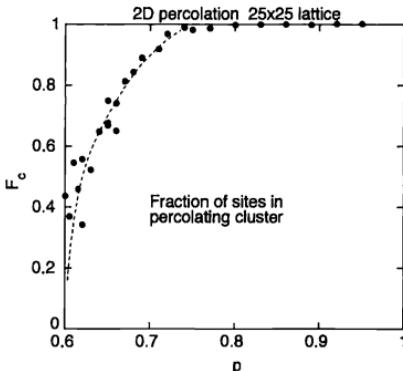


FIGURE 7.27: Fraction of sites in the percolating cluster as a function of concentration for a 25×25 lattice. The dashed curve is simply a guide to the eye and is intended to illustrate the singularity at p_c .

As an example we consider the spanning cluster at concentrations above p_c . We saw in Figure 7.25 that for large p nearly all of the occupied sites were part of the infinite cluster, but as p_c was approached, many small clusters appeared. Let us, therefore, calculate the fraction of sites F , which are in the spanning cluster as a function of p . This can be accomplished using the cluster-numbering algorithm by first generating a lattice with a given probability of occupied sites, and then using the labeling scheme to determine which sites are in the spanning cluster. Some results for F are shown in Figure 7.27. Note that F is the number of sites that are in the spanning cluster divided by the number of occupied sites, *not* the total number of lattice sites. We see that F drops precipitously and seems to be heading to zero at p_c . It turns out that the variation of F near p_c is given by a power law

$$F = F_0 (p - p_c)^\beta, \quad (7.31)$$

where β is another critical exponent. The results in Figure 7.27 are not accurate enough to provide an accurate estimate for β , although they certainly suggest that F goes to zero in a singular manner; that is, $dF/dp \rightarrow \infty$ as $p \rightarrow p_c$. Careful analytic calculations (see the references) have shown that the behavior is indeed singular, and that $\beta = 5/36$ for all two dimensional lattices, such as square, triangular, and honeycomb.³⁶ Singularities of this form are very common near second-order phase transitions.

³⁶Such values are independent of the details of the lattice, and depend only on the dimensionality (and some other fundamental characteristics). They are said to be *universal*. The critical exponents at a second-order phase transition, such as the percolation transition, are examples of such universal quantities.

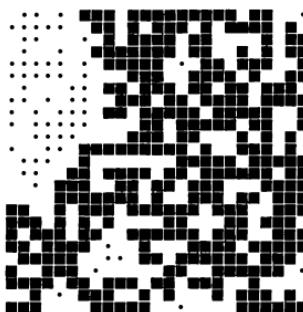


FIGURE 7.28: Plot of the spanning cluster (solid squares) and occupied sites that are not part of the spanning cluster (dots). The concentration is 0.60 and the lattice size is 25×25 . Sites that contain neither squares or dots are unoccupied.

While the singular behavior of F is very important, we should note another striking feature of Figure 7.27. At p_c the spanning cluster has an infinite size so as to span the system, yet, since $F \rightarrow 0$, it contains a *vanishing* fraction of the occupied sites! An object that has an infinitely large extent but has no volume, is certainly very unusual and should make you suspicious that a fractal is involved. To investigate this we have used the cluster-labeling algorithm to separate the spanning cluster from the other sites for a lattice with $p = 0.6 \approx p_c$, as shown in Figure 7.28. This spanning cluster is seen to have a very open structure, and in many places it is just barely connected. That is, the removal of only one or a very few occupied sites would destroy the connectedness of the cluster. Hence, the visual evidence also suggests that the spanning cluster is a fractal.

To get a better feeling for the structure of the spanning cluster at p_c it is useful to consider much larger lattices and to compare results for different concentrations. While this can be accomplished with the cluster labeling algorithm we have employed to this point, there are better methods for directly generating a single cluster with a specified value of the concentration.

One such method illustrates a general search algorithm called a *breadth first search*, which we now describe. We begin by placing a seed (site 1) at the origin, and place this site number in a list of occupied sites. We then go through each adjacent site, checking to see if we should occupy it (with probability p), or mark it as empty. If it is chosen to be occupied, this site is labeled 2 and the number 2 is added to the list of occupied sites. When all the adjacent sites of the seed have been checked in this way, we proceed to check the adjacent sites of the site 2 (the first of the added sites around the seed), and continue in this manner, deciding on site occupation in successive generations, somewhat like the tree rings. The process continues until either all adjacent sites of all the occupied sites have been determined to be empty (the cluster growth stops) or a predetermined exit condition is satisfied (such as

reaching a desired cluster size) In this process, it is important to remember that once a site is determined to be empty, we cannot later reconsider it for occupancy without changing the effective value of p . An implementation of this algorithm is sketched below:

EXAMPLE 7.6 Breadth-first search algorithm for percolation cluster generation

- Decide on a fixed *local* search order around a site. Initialize all variables.
 - Occupy the starting point and put this site on the *stack*. Set both the lower pointer m_1 (for the site whose neighbors are being checked) and the upper pointer m_2 (for the top site on the *stack*) to 1.
 - Search all neighbors of site m_1 by first testing if the neighbor is already on the *stack*.
 - ▷ If yes, go to the next neighbor and continue searching.
 - ▷ If not, check if any of the limits has been reached.
 - If limits have been reached, then growth must be stopped. Exit.
 - If not, check if this neighbor has been marked empty already.
 - ▷ If yes, go to the next neighbor.
 - ▷ If not, get a random number to decide whether to occupy this site. Mark it as occupied, put it on the *stack*, and advance $m_2 \Rightarrow m_2 + 1$, if appropriate. Otherwise mark it empty and go to the next neighbor.
 - ▷ Test if the two *stack* pointers coincide ($m_1 = m_2$).
 - If yes, the cluster is finished. Exit either with or without success depending on the success criteria.
 - If no, increment $m_1 \Rightarrow m_1 + 1$ and repeat the search.
-

This *breadth-first algorithm* differs from the depth-first approach that we described in Example 7.3. Here, for each value of the “depth” (the size of the cluster) the breadth-first algorithm fully explores the “tree” of all possible sites that might be added to the cluster, before expanding the search to include larger clusters.

The breadth-first and depth-first approaches are two powerful search methods, and are useful in many different circumstances, not just for simulating percolation clusters or enumerating SAWs. It is also possible to mix these approaches; e.g., instead of checking *all* neighbors of a given generation of occupied sites before going on to the next generation, and instead of checking all sites connected in a preferred direction before going on to other directions, we may pick one neighbor at *random* and update the neighbor list correspondingly. Such a method will work well for the generation of a percolation cluster, though it can be more difficult to make it exhaustive.



FIGURE 7.29: Clusters grown with $p = 0.55$ (left), $p = 0.60 \approx p_c$ (center), and $p = 0.70$ (right). Note that the plot on the left is greatly expanded as compared to the others, as this cluster contained far fewer sites.

Some typical percolation clusters are shown in Figure 7.29, where we consider several different values of p . These plots show particular clusters for a given p , while we are really interested in typical (that is, average) clusters. To estimate the properties of a typical cluster it is necessary to perform an average over many clusters grown with the same value of p , and we will get to that in a moment. Our point is that when growing a single such cluster it is always statistically possible that we may obtain a cluster containing only a single site or a cluster that contains an infinite number of sites (assuming that p is neither 0 or 1). These possibilities, and all cluster sizes in between, are all conceivable although their probabilities will vary strongly with p .

With this proviso, the plots in Figure 7.29 show “representative” clusters. At $p = 0.55$ the cluster-growth algorithm often terminated with clusters containing 10–20 sites, or even fewer. This is what we should have expected, since we know that below p_c there will not be an infinite (spanning) cluster. For $p = 0.7$ we are above the percolation threshold, so we expect a spanning cluster to be found. This was indeed the case, as the cluster in Figure 7.29 continued to grow without bound; we stopped the simulation to obtain this snapshot. The cluster for $p = 0.7$ has a qualitative appearance very similar to that of Eden clusters, with a fairly smooth perimeter and relatively few interior holes. The cluster at $p = 0.6 \approx p_c$ has a rather different appearance. Its perimeter is much more irregular and there are many large cracks. This is reminiscent of the DLA clusters, and it should not be surprising to learn that the infinite cluster at p_c is a fractal. We will leave a study of its fractal dimensionality to the exercises, although we note that d_f for the spanning cluster at p_c is different from that of a DLA cluster.

The fractal nature of the percolating cluster at p_c has many interesting consequences. For example, if we are modeling the mechanical properties of a porous material and the unoccupied sites are voids in the system, the strength will be strongly dependent on the connectivity of the spanning cluster.³⁷ The flow of a

³⁷A consideration of mechanical properties usually involves more than just the question of spanning, since they will generally depend the kinds of connecting paths that are present, and the

fluid like water or oil through such a system would also be a sensitive function of the connectivity. Another interpretation is to suppose that the occupied sites are trees in a forest. If a fire is somehow set at one edge of the forest, we can use a percolating network to study how the fire will propagate into the forest. This can be modeled by assuming that all of the trees (that is, occupied sites) on one edge of the lattice are set to burn at time $t = 0$. At the next step all of the trees adjacent to a burning tree will themselves start to burn, while the trees burning at $t = 0$ will burn out. This process is then repeated; at step t_n , the previously unburned trees that are adjacent to a burning tree will start to burn, and the trees burning at t_{n-1} will burn out.³⁸ It is instructive to calculate the time it takes for a fire to burn out completely. For small p the fires burn out quickly since the clusters of connected trees are on average very small. The only trees to burn will be those near the edge where the fire is lit, and most of the forest will be spared. For large p the fire burns across the system rapidly. Since essentially all of the trees are connected in this case, the number of time steps for a fire to burn-out is $\sim L$ where L is the size of the lattice, and nearly all of the trees will be engulfed. However, for p near p_c the fractal structure of the spanning cluster forces the fire to follow a tortuous path through the system, and the fire takes many time steps to burn out completely. Some results for the fire burn-out time are shown in Figure 7.30, which shows precisely this behavior. There is a large peak in the fire lifetime at p_c , which directly reflects the fractal connectivity of the critical cluster at p_c . If we were to study this peak as a function of lattice size (a task we will leave for the exercises), we would find that the burn-out time *diverges* for an infinitely large lattice. The nature of this divergence contains important information about the fractal nature of the critical cluster, a topic we will leave for the references. Our results also imply that a forest whose concentration of trees places it below p_c has a better chance of surviving a fire than a more concentrated forest.

EXERCISES

- 7.27. Calculate the critical concentration for percolation for a two-dimensional triangular lattice using one of the cluster labelling schemes described in this section. (A triangular lattice is one for which each site has 6 connected (nearest) neighbors.) Consider various lattice sizes up to 50×50 or 100×100 (whatever your computer can handle comfortably). You should find $p_c \approx 0.500$ in this case. (The exactly known p_c is $1/2$.) Comparing this with the value of p_c for a square lattice, we notice that for a given dimension p_c becomes smaller as the number of nearest neighbors increases. Can you explain this trend qualitatively?
- *7.28. Repeat the previous problem but use a three-dimensional simple cubic lattice. You should find $p_c \approx 0.312$. Comparing this with the value of p_c for a square lattice, we notice that for hyper-cubic lattices p_c becomes smaller as the dimensionality is increased. Give a qualitative argument to explain this trend.

mechanical properties of the individual connections.

³⁸An astute reader will note that this process closely resembles the description of the breadth-first search. Indeed, if we first create a percolative lattice of trees as occupied sites and then start a breadth-first search initially listing all the trees on the edge where the fire started, then the search will efficiently tag the trees by the generation number n , corresponding to the burning time

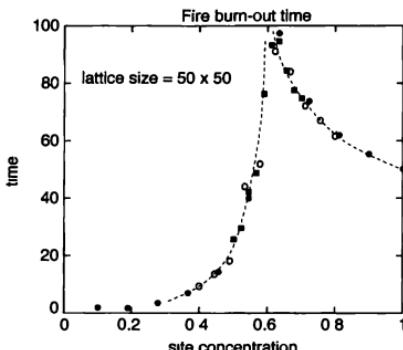


FIGURE 7.30: Forest fire burn-out times for a 50×50 square lattice. The different symbols were obtained from independent simulations. All three simulations computed the average burn-out time for several forests of each size. The dashed curves are guides to the eye that suggest a singularity near $p_c \sim 0.593$.

- 7.29. In this section we discussed only the problem of *site* percolation, for which the lattice sites were occupied with a specified probability. We can also consider the problem of *bond* percolation, in which the links between sites are present with a probability p . Calculate the critical bond probability for percolation in a two-dimensional square lattice. Consider various lattice sizes up to 50×50 or 100×100 (whatever your computer can handle). You should find $p_c \approx 0.5$; can you give an argument to explain this value? The exact value of p_c is $1/2$.
- 7.30. Generate a spanning cluster for a two dimensional square lattice at $p = p_c$ using any of the search methods discussed in connection with Figure 7.29. Estimate the fractal dimensionality of the cluster. You should find a value of d_f , which is slightly smaller than 2 (the expected value is $91/48 \approx 1.90$)
- 7.31. Grow percolation clusters for $p \leq p_c$ and calculate the average linear dimension of the cluster, ξ . One way to estimate ξ is by calculating the average distance of particles in the cluster from the seed (the first occupied site). Show that ξ becomes very large at p_c (it actually diverges there), reflecting the infinite size of the spanning cluster. Try to fit a power law $\xi \sim |p - p_c|^{-\nu}$ to your results and estimate the critical exponent ν . For a two dimensional lattice you should get a value close to $4/3$.
- *7.32. Generate an $L \times L$ percolation grid on a square lattice (L can be 50, 100, 1000 or whatever your computer can handle) for a series of $p \leq p_c$. A suggested method is to go through all sites of the grid in a typewriter fashion (say, top to bottom and left to right), deciding whether to occupy the site comparing a pseudorandom number with p , and assigning a cluster number using the cluster multiple labeling algorithm discussed in the text. While you do this, you can also keep track of the number of sites s present in each cluster, and the number N_s of clusters of size s .

Now, consider the following quantity. if you pick a site at random, what is the size of the cluster to which the site belongs? The average of this quantity is often referred to as the mean size χ of a cluster. This value can be estimated by calculating $\sum_s s^2 N_s / L^2$ (why?). Do this calculation, averaging over many realizations of such a grid at each value of p and plot your result. You should see a sharp increasing trend as p approaches p_c . Try to fit a power law $\chi \sim |p - p_c|^{-\gamma}$ to your results and estimate the critical exponent γ . You should get a value close to 2.4 (for two dimensional lattices).

- *7.33. Calculate the burn-out time for $L \times L$ forests with different values of L . Show that the peak at p_c becomes larger as the forest is made bigger and try to extrapolate your results to estimate the burn-out time as a function of p for an infinite lattice.

Hint: In making an extrapolation to the case $L \rightarrow \infty$ it is useful to plot things as a function of L^{-1} so that $L = \infty$ is on the graph.

7.9 DIFFUSION ON FRACTALS

So far we have discussed diffusion and fractals individually but have not connected the two. However, problems like the diffusion of fluid particles within a porous rock, or electrical conduction through a granular metal-insulator composite may be better modeled by diffusion which is confined to an irregularly shaped spatial region. If the irregularity extends over a range of length scales (from short to very long), such a region may be modeled by a fractal, perhaps a DLA cluster or a percolation cluster at p_c , depending on the physics of the problem. This idea gives us motivation to study diffusion on a fractal. Moreover, this is an example of a physical process (diffusion in this case) which takes place in a non-Euclidean space (fractal), and this will give us a glimpse of how the connectivity of space affects the physics.

We will use a random walk to model diffusion, and consider a randomly generated fractal cluster as the confining space. The first effect of the confinement will be that the walker will not travel as far as it might if it weren't so confined. That is, $\langle r(t)^2 \rangle$ is smaller on a fractal cluster. What might surprise you is that $\langle r(t)^2 \rangle$ is not only smaller but also scales differently with time (step number). Thus, if we define an exponent d_w by

$$\langle r(t)^2 \rangle \sim t^{2/d_w}, \quad (7.32)$$

we have $d_w > 2$ and the random walk is *sub-diffusive*.³⁹ Such a process is also called *anomalous diffusion*. The mean-square displacement grows less rapidly with time because the walker is not only blocked by the holes in the cluster but also because those holes occur at many length scales. Their sizes range from small to very large, and they are ubiquitous in a fractal.

Another consequence effect of being confined to a fractal is that the probability of recurrence is much enhanced. If we denote by $P_0(t)$ the probability for a random walk to return to the starting point after time t , we generally have

$$P_0(t) \sim t^{-d_w/2}, \quad (7.33)$$

³⁹Note that, for a normal random walk in a Euclidean space, $2/d_w = 1$ or $d_w = 2$. This exponent is defined in a suggestive way; if we take r as a linear dimension of the walk trajectory and t is identified with its *mass*, then d_w would be its fractal dimension. For this reason, d_w is often called the walk dimension, or the fractal dimension of the walk

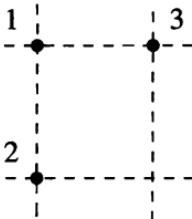


FIGURE 7.31: A 3-site cluster on the square lattice

which defines a new exponent d_s .⁴⁰ According to (7.32), during time t , the random walker will have traveled through the region of linear dimension of the order of $\ell \sim t^{1/d_w}$. Within this linear dimension, there must be of order $\ell^{d_f} \sim t^{d_f/d_w}$ sites of the cluster, where d_f is the fractal dimension of the cluster. If the location of the random walker at time t can be any one of these sites, then the probability that it happens to be the starting point must be proportional to $1/\ell^{d_f} \sim t^{-d_f/d_w}$. Since this probability should be equal to $P_0(t)$, we expect that

$$d_s = 2d_f/d_w, \quad (7.34)$$

a relation that could be checked numerically if we calculate d_f , d_w , and d_s separately. In an ordinary, Euclidean space, $d_s = d$, the spatial dimension. However, on a fractal, d_s is generally less than 2; physically, this means that a random walk will *always* return to its starting point, if one waits long enough.

We can certainly study this problem by first simulating the fractal cluster, and then simulating random walks on it. However, this would require averaging over many random walks for each cluster and then averaging over many clusters. A more efficient way is to do the averaging over one cluster *exactly* and only simulate the random clusters. Consider a cluster of S sites, numbered sequentially from 1 to S . We define the transition probability matrix \mathbf{W} as an $S \times S$ matrix whose ij -th element is the probability that a random walker at site j hops to site i at the next time step. To illustrate this, consider a 3 site cluster on the square lattice shown below. We suppose that the walker “attempts” to make steps in all four orthogonal lattice directions, but is only successful if the cluster actually has a site in that particular direction. If a step then “fails,” the walker stays at his initial location. This is called a *blind ant* random walk, and the matrix \mathbf{W} for such a walk on this small cluster is given by

$$\mathbf{W} = \begin{pmatrix} 1/2 & 1/4 & 1/4 \\ 1/4 & 3/4 & 0 \\ 1/4 & 0 & 3/4 \end{pmatrix}. \quad (7.35)$$

⁴⁰The exponent d_s is called the spectral dimension since it is also related to a *spectral* property of the diffusion problem (see below).

The diagonal elements give the probabilities to stay at the same site, corresponding to the failed attempts to step to a non-existing neighbor.⁴¹

The matrix \mathbf{W} can be used in many different ways to study the time evolution of the random walk problem. For example, suppose that the walker is initially located at site k ; in the matrix notation of (7.35), this state can be described by a column vector of unit length \vec{u}_k , in which only the k -th element is non-zero. After one time step, the state of the walker is given by $\mathbf{W} \cdot \vec{u}_k$. This is a new column vector, whose elements are now the probability for finding the walker at that particular location in the cluster. The probability distribution after n time steps is then $\mathbf{W}^n \cdot \vec{u}_k$. You should now suspect that we might be leading towards an eigenvalue problem, and indeed we are.⁴² We denote the eigenvalues of \mathbf{W} as $\lambda_1, \lambda_2, \dots$ and suppose that they are ordered from largest to smallest. \mathbf{W} is an example of a *Markov matrix* whose elements are all non-negative, and for which each column (or row) adds up to 1. Such a matrix has the mathematical property that it has a nonnegative-definite eigenvector with eigenvalue 1 and all of the eigenvalues satisfy $|\lambda| \leq 1$. For most random walk models of interest, including the blind ant, all of the eigenvalues are real. Since \mathbf{W} describes time evolution, the eigenvector with eigenvalue $\lambda = 1$ gives the *stationary* probability distribution of the random walker. All initial distributions of the random walkers eventually converge to it.⁴³ The other eigenvalues λ_i ($i > 1$) describe the rates of convergence, i.e., each component orthogonal to the stationary state decays in time exponentially as $|\lambda_i|^t = \exp(-|\ln |\lambda_i||t)$. This identifies the characteristic time scale of such relaxation as $|\ln |\lambda_i||^{-1}$.

Figure 7.32 shows the distribution of eigenvalues for a blind ant \mathbf{W} on a percolation cluster of size $S = 500$ at $p = 0.593$ (this is essentially right at p_c for the square lattice), averaged over 10 independent clusters.⁴⁴ Although we could also have studied other kinds of random fractals (e.g., DLA clusters), a critical percolation cluster serves the purpose well; of course, the numerical values of the exponents quoted here are specific to the percolation problem.

Though many things can be done with such an eigenspectrum; for simplicity we focus here on two aspects.⁴⁵ The first is the sharp increase in the density toward $\lambda = 1$. This means that many many eigenvectors accumulate near the stationary state at $\lambda = 1$. These are called “transient states,” since they decay only slowly with time; i.e., on repeated multiplication by \mathbf{W} . Though each of these transient states describes an exponential convergence toward the stationary state, the multitudes of them in fact conspire to produce a power-law convergence. The power-law increase of the density can be shown to scale for λ close to 1 as

$$n(\lambda) \sim |\ln \lambda|^{d_s/2 - 1}, \quad (7.36)$$

⁴¹It is possible to consider other types of rules of what to do when a random walk hits a boundary (see the exercises)

⁴²See Appendix H for a discussion of eigenvalue problems and how to deal with them

⁴³We neglect the possible effect of an eigenstate with eigenvalue -1 , which does not actually arise for the blind ant model. The existence of the stationary state with eigenvalue 1 is the basis of the assured convergence of the Jacobi method we discussed in Chapter 5 for the relaxational solution of the Laplace's equation

⁴⁴These results were obtained using the numerical package *Lapack*. We give a general discussion on the eigenvalue and other linear problems in Appendix H

⁴⁵The interested reader can find more in Nakanishi (1994) and references therein

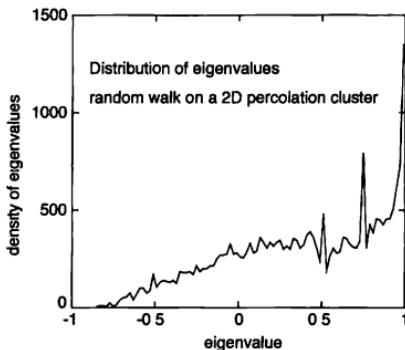


FIGURE 7.32: Distribution of eigenvalues for the random walk on a two-dimensional critical percolation cluster. The values shown were obtained by averaging the results from 10 separate 500-site clusters and binning them into eigenvalue intervals of width 0.02

where the spectral dimension $d_s \approx 1.3$, and d_s is the same exponent as appeared in (7.33).⁴⁶

Our second focus will be on the subdominant, or the second largest eigenvalue λ_2 . This eigenvalue reflects the longest, finite time-scale relaxation,⁴⁷ and thus must reflect the geometry and size of the whole cluster. Thus, the time scale $|\ln \lambda_2|^{-1}$ should correspond to the time required for the random walk to just spread through the entire cluster. Since (7.32) relates the time and length scales for a random walk, the length associated with this time scale must be $|\ln \lambda_2|^{-1/d_w}$. Comparing this to the linear dimension L of the cluster, we finally conclude that

$$|\ln \lambda_2| \sim L^{-d_w}, \quad (7.37)$$

a finite-size scaling relation that can be used with the numerical data for λ_2 to determine the value of d_w . Figure 7.33 shows an example of such a fit.

Taking this result for d_w and the previously given value of $d_s \approx 1.3$, together with the fractal dimension $d_f \approx 1.9$ for two dimensional percolation, we see that the relation (7.34) is satisfied within the statistical error for this particular fractal.⁴⁸ A value of d_s for two-dimensional percolation is typical for a subdiffusive, recurrent

⁴⁶This follows from the mathematical connection between $P_0(t)$ and $n(\lambda)$ through a Laplace transformation. The details of such connections can be found in Nakanishi (1994) and references therein.

⁴⁷We have already shown the correspondence between the Jacobi method introduced in Chapter 5 and the random walk problem. You may be interested to know that how close to 1 the subdominant eigenvalue λ_2 is actually determines how many iterations must be performed in the Jacobi method to achieve a certain level of accuracy.

⁴⁸However, (7.34) may not be universally true for all fractals

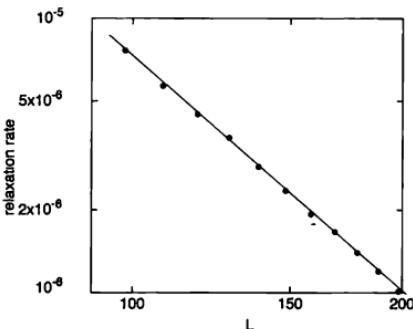


FIGURE 7.33: The dependence of the relaxation rate, or the inverse time scale $|\ln \lambda_2|$ is shown in a log-log plot as a function of the lattice size L . The solid line is a linear least squares fit with a slope of about -2.85 . Shown are the average over many spanning percolation clusters of the square grid of $L \times L$ at $p \approx 0.593$.

random walk on fractals. There is much more that could be deduced from the eigenvalues and eigenvectors, but we do not have space to dwell more on them here.

EXERCISES

- 7.34. Generate a spanning cluster for a two dimensional square lattice at $p = p_c$ using any of the search methods discussed earlier, and simulate many random walks constrained to the cluster. From these simulations, estimate the exponent d_w . Do the same for a triangular lattice. Do you get compatible estimates?
- 7.35. Consider a random walk model called the *myopic ant*. In this model, the random walker can see which nearest neighbor sites are occupied and available, it only chooses the next step from among the available directions. Thus, e.g., for the 3-site cluster shown in Figure 7.31, the matrix \mathbf{W} will be given by

$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 1 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \end{pmatrix}. \quad (7.38)$$

Apply this model to a spanning, near-critical percolation cluster on the square lattice. Start the random walk from one chosen site (say, the origin), and calculate $\langle r(t)^2 \rangle$ exactly on this cluster by multiplying \mathbf{W} at each time step and evaluating the average of $r(t)^2$ from the resulting probability distribution. Fit the power-law of (7.32) to your results and calculate d_w .

- *7.36. Generate a DLA cluster for a two dimensional square lattice. Obtain the transition probability matrices \mathbf{W} for the blind ant model (or the myopic ant model of the previous problem, or both) and solve for all of their eigenvalues. (You will

have to limit the cluster size to a few hundred or less if your computer is of modest capability.) From these data, try to obtain plots analogous to Figures 7.32 and 7.33, and estimate d_s and d_w for random walks on DLA.

REFERENCES

- [1] J. M. Hammersley and D. C. Handscomb, 1964, *Monte Carlo Methods*, Methuen, London. An early introduction to Monte Carlo methods. See p. 135 for the “famous” quote about the limitation of Monte Carlo as applied to percolation.
- [2] J. Hoshen and R. Kopelman, “Percolation and cluster distribution. I Cluster multiple labeling technique and critical concentration algorithm,” *Phys. Rev. B* **14**, 3438 (1976).
- [3] B. B. Mandelbrot, 1977, *Fractals: Form, Chance, and Dimension*, Freeman, New York. An early and very readable book on fractals by the person who “invented” them.
- [4] P. Meakin, 1988, “The Growth of Fractal Aggregates and their Fractal Measure,” in *Phase Transitions and Critical Phenomena*, Vol. 12, C. Domb and J. L. Lebowitz, eds., Academic Press, Orlando. An extensive discussion of research on the growth of fractal clusters and their properties.
- [5] H. Nakanishi, 1994, “Random and Self-avoiding Walks in Disordered Media,” in *Annual Reviews of Computational Physics*, Vol. 1, D. Stauffer, ed., World Scientific, Singapore.
- [6] F. Reif, 1965, *Fundamentals of Statistical and Thermal Physics*, McGraw-Hill, New York.
- [7] D. Stauffer and A. Aharony, 1991, *Introduction to Percolation Theory, 2nd Edition*, Taylor and Francis, Briston. A very entertaining book and an excellent introduction to the subject.
- [8] T. Vicsek, 1989, *Fractal growth phenomena*, World Scientific, Singapore.
- [9] R. F. Voss, 1984. “On 2D Percolation Clusters and on Multi-particle Fractal Aggregation.” In *Kinetics of Aggregation and Gelation*, F. Family and D. P. Landau, eds., Elsevier, New York. A nice paper (and some nice pictures) illustrating how the fractal structure can depend on the growth rules.
- [10] F. T. Wall and J. J. Erpenbeck, “New method for the statistical computation of polymer dimensions,” *J. Chem. Phys.* **30**, 634 (1959).

Statistical Mechanics, Phase Transitions, and the Ising Model

In the early chapters of this book, all of our problems involved systems containing a small number of particles, often only one or two. In the past few chapters have we have begun to focus on more complex systems involving a large number of degrees of freedom, typically with many particles, leading to our work on stochastic systems in Chapter 7. There we concentrated our attention on *random* processes and systems such as diffusion, percolation, and fractals. In all of those cases, the interactions between particles (or degrees of freedom) did not play an explicit role in our computations. In this chapter we extend our studies to multi-particle and many-state systems where the *interactions* between particles play an essential role. We will see that systems of many interacting particles can exhibit a very important phenomenon known as a *phase transition*. Examples include the condensation of a gas into a liquid and the appearance of ferromagnetism in materials such as iron, so these transitions are quite common in nature. You should notice that both of these examples involve the concept of *temperature*, and this will lead us to consider issues in the realm of thermal and statistical physics.

In this chapter we will rely heavily on a stochastic approach in which the interaction of a system with its environment is simulated with the aid of a random number generator. This approach is known as the Monte Carlo method,¹ and we will use it to study the Ising model of magnetism. Along the way we will review and illustrate some statistical mechanical ideas relating to phase transitions and the canonical ensemble, and observe a connection with the percolation phase transition that we studied in Chapter 7.

8.1 THE ISING MODEL AND STATISTICAL MECHANICS

Magnetism is an inherently quantum phenomena. It is interesting that Niels Bohr, one of the creators of quantum mechanics, made a seminal contribution to the field of magnetism. He showed that a classical system could never exhibit ferromagnetism. Quantum mechanics had not yet been invented when he proved this theorem, so in a sense the existence of ferromagnets such as iron is clear and compelling evidence that the world cannot be described in full by classical physics. A key ingredient in the theory of magnetism is the electron's spin and the associated

¹Simulation techniques that rely on stochastic sampling of various states are generally called "Monte Carlo" methods. Thus, many of the simulations in Chapter 7 and the stochastic integration described in Appendix E also fall into this category



FIGURE 8.1: Schematic spin model for a ferromagnet

magnetic moment.² Ferromagnetism arises when a collection of such spins conspire so that all of their magnetic moments point in the same direction, yielding a total moment that is macroscopic in size.³ A central issue in the study of ferromagnetism is to understand how the interactions between spins gives rise to this overall alignment. Since we also know that systems generally lose their magnetism at high temperatures,⁴ we would also like to understand why and how the magnetic properties depend on temperature.

The model of a ferromagnet that we will consider is shown schematically in Figure 8.1. It consists of a collection of magnetic moments, which we denote by arrows and which we can think of as being atoms with spin = $\frac{1}{2}$ magnetic moments. For simplicity we assume that these spins are situated on a regular lattice. Since spin is a quantum mechanical phenomena, this is in a sense a quantum model. A fully quantum mechanical treatment of the model would require that we include all of the quantum rules for dealing with spin angular momentum, etc., in our simulations. It turns out that such a calculation would be extremely difficult, so we will not attempt it here.⁵ We will instead make a few simplifications. Much work in this field has shown that despite the simplifications we are about to make, our model does capture the essential physics of magnetism.

We will assume that each spin is able to point along either $+z$ or $-z$; that is, either up or down. *No* other orientation is permitted. Hence, the i th spin in the system can have one of only two possible values, which for convenience⁶ we take to be $s_i = \pm 1$. Each of these so-called Ising spins interacts with other spins in the lattice; in a ferromagnet this interaction will favor parallel alignment of pairs of spins. In a real magnetic material the interaction will be largest between spins that are nearest neighbors and fall off rapidly with increasing separation between the

²There can also be a contribution to the total magnetic moment from the orbital angular momentum, but for simplicity we will refer to the moment as if it were due solely to spin. This is not a crucial or limiting assumption, but will make our treatment (mainly the terminology), which is aimed at so-called local moment systems, a bit simpler. The more challenging problem of magnetism in a metal is too much for us to tackle here.

³Assuming, of course, that there are a macroscopic number of electrons, as in a typical solid.

⁴For example, iron is no longer a ferromagnet above about 1000 K.

⁵The study of quantum spin models is an active area of current research.

⁶We could instead let the values be $\pm \frac{1}{2}$, but that would only amount to a rescaling of the exchange constant in (8.1).

two spins. With this motivation, the simplest Ising model assumes an interaction only between nearest neighbors so that the energy of the system is

$$E = -J \sum_{\langle ij \rangle} s_i s_j , \quad (8.1)$$

where the sum is over all pairs of nearest neighbor spins $\langle ij \rangle$, and J is known as the exchange constant, which we will assume to be positive. It is useful to imagine that the spin system is in a particular state, corresponding to a particular arrangement of the individual spins (particular values of the spin variables s_i). Equation (8.1) is then the energy of this particular state of the entire system. The energy function (8.1) together with the description of an Ising spin given above, serves to define the Ising model.⁷ It was first conceived by Wilhelm Lenz, who suggested it as the Ph.D. topic for his graduate student Ernst Ising in the 1920s.

Before proceeding to a quantitative treatment of the Ising model, it is useful to first anticipate the qualitative behavior we will find, and in the process give a quick review of statistical mechanics. According to the energy function (8.1), two neighboring spins will have an energy of interaction $-J$ if they point in the same direction and $+J$ if they are antiparallel. Since J is assumed positive, the interactions favor *parallel* alignment of neighboring spins. If each spin is parallel to its neighbors, then *every* spin in the lattice will be parallel to every other spin. Such alignment of all of the magnetic moments would lead to a nonzero magnetic moment for the system and thus yield a ferromagnet. A system that has a magnetic moment in the absence of a magnetic field is said to have a spontaneous magnetization

While the energy of the spin system is lowest if all of the spins are parallel to one another, the disordering effect of temperature must also be considered. We will assume that our spin system is in equilibrium with a heat bath at temperature T , so that the behavior is described by the canonical ensemble. For an introductory discussion of the canonical ensemble see Reif (1965). One way to view this is that over time different spins flip back and forth, and the system will thereby "move" into different spin configurations. The behavior observed in an experimental measurement will depend on how much time the system spends in the different possible spin configurations. It is a fundamental result of statistical mechanics that for a system in equilibrium with a heat bath, the probability of finding the system in any particular state is proportional to the Boltzmann factor

$$P_\alpha \sim e^{-E_\alpha/k_B T} , \quad (8.2)$$

where E_α is the energy of state α (not to be confused with spin i) as calculated from (8.1), k_B is Boltzmann's constant, and P_α is the probability of finding the system in state α . Each of these states is a particular configuration of spins, which we will refer to as a *microstate* of the system. If we have a lattice containing N Ising spins, each spin can be in either of two states, so there are 2^N different possible

⁷This is the simplest example of the Ising model. Nowadays, the idea has been greatly extended and the designation "Ising model" encompasses a wide variety of models in which the interactions involve spin degrees of freedom that have either two discrete values, or can be described by a single scalar variable (often referred to as "single component" spins)

microstates of the system as a whole. We will be most interested in systems for which N is large, so the number of microstates will be very large. This will make the problem difficult to solve, but will also lead to some interesting behavior.

From a microscopic point of view, it is the interaction of the spin system with a heat bath that causes the system to undergo transitions from one microstate to another. Individual spins flip from +1 to -1 or vice versa as they gain energy from, or lose energy to, the heat bath. A macroscopic measurement of a quantity such as the total magnetic moment (which we will also refer to as the magnetization⁸) effectively averages over the many microstates that the system visits during the course of a measurement. In order to calculate the macroscopic behavior, we therefore need to calculate the probabilities P_α of finding the system in its various microstates. For example, the magnetic moment of a microstate M_α is the sum of the values of s_j for all of the spins in that particular state. The *measured* magnetization of the system will then be

$$M = \sum_{\alpha} M_{\alpha} P_{\alpha}, \quad (8.3)$$

where $M_\alpha = \sum s_j$, with the values of the spin variables in this sum corresponding to the spin directions in microstate α . Similarly, other properties can be expressed in terms of the probabilities P_α . This should all be familiar to you from statistical mechanics and is reviewed in the references at the end of this chapter.

In the next few sections we will consider how to calculate quantities such as the magnetization and other properties of our spin system. This calculation is made difficult by the large number of microstates. We noted already that for an Ising system with N spins there will be 2^N states, and we will be interested in systems for which $N \rightarrow \infty$, so the number of states will be very large indeed.⁹ Analytic approaches have proved very formidable, and only a relatively few exact results are known for these systems. This makes simulations very attractive for this problem.¹⁰

Before we move on to calculate the properties of our spin system, it is useful to put the model itself into perspective. In the past few paragraphs we have introduced an *extremely* simple spin model. Many other spin models have been introduced and studied. For example, we can let the spins be fixed-length vectors that are free to rotate either in a plane (the so-called XY model), or in three dimensions (the Heisenberg model).¹¹ It is also interesting to consider the effect of increasing the range of the interactions between spins. That is, we can let spins that are second, third, or more distant nearest neighbors interact. In addition, there is the

⁸Strictly speaking, the magnetization is the magnetic moment per unit volume, but we can assume that our system has unit volume so that these two quantities are equal

⁹When dealing with a sum involving such a large number of terms, it is sometimes possible to ignore many of the terms because they are small. However, it turns out that near a phase transition, such a simplification does not occur. A very large (infinite) number of terms are important in that case.

¹⁰A number of very powerful approximate analytic methods have also been developed, as described in the references. We certainly don't want to give the impression that numerical simulations are the only useful way to attack this problem.

¹¹Physicists have also been led to consider spin vectors with more than three and less than one components!

possibility that a real spin system might have to be treated quantum mechanically, and this makes our job even more challenging. All of these models of magnetism exhibit interesting properties and have been studied extensively over the past 50 years or so. Among other things, this work has shown that the simple Ising model captures many of the essential features of the phase transition to the ferromagnetic phase. This makes it ideal for studying phase transitions. Readers who want stiffer challenges can find them in the references.

8.2 MEAN FIELD THEORY

In this section we consider a very useful *approximate* approach for calculating the properties of a spin system. We will use it to introduce and illustrate several interesting features of the phase transition. While the calculational method explored here is a useful qualitative tool for the study of phase transitions, its results are not quantitatively accurate, so we will revisit some of the same issues and problems when we consider another approach later in this chapter.

The magnetization is closely related to the average spin alignment $\langle s_i \rangle$, where the angular brackets denote a thermal average. It is useful to think of this as a time average for a system in thermal equilibrium with a heat bath.¹² As described in Section 8.1, the interaction with the heat bath causes spins to flip from +1 to -1 and vice versa. A thermal average is an average with respect to the different microstates that are generated by these spin flips. For an infinitely large system, the spins will all have the same average alignment. This can be seen by noticing that the spins are all equivalent in the sense that each one interacts with four nearest neighbors and (in this idealized case) each is infinitely far from any boundaries. Hence all spins must have the same *average* properties. The total magnetization at temperature T for a system of N spins will then be

$$M = \sum_i \langle s_i \rangle = N \langle s_i \rangle , \quad (8.4)$$

where in the last term we can use any value of i that is convenient, since we have argued that all spins are equivalent. Thus, if we can calculate $\langle s_i \rangle$, we immediately have M as well. An exact computation of $\langle s_i \rangle$ would require the probabilities of all possible microstates, the P_α terms in (8.2). This is a formidable task (which we will take up in the next section), so we consider here an approximate alternative known as *mean field theory*.

If we add a magnetic field to the problem, the energy function becomes

$$E = -J \sum_{\langle ij \rangle} s_i s_j - \mu H \sum_i s_i , \quad (8.5)$$

where H is the magnetic field and μ is the magnetic moment associated with each spin [compare with (8.1)]. This field will tend to make the spins orient themselves parallel to H , since this lowers the energy. Let us now assume, for the moment,

¹²The problem of time averages is at the heart of the ergodic hypothesis, which we discussed in Chapter 7.

that our system contains just a single spin, s_i , so that the only energy involved is the field energy. A single spin has two possible states, $s_i = \pm 1$, whose energies are $E_{\pm} = \mp \mu H$. The probabilities of finding the “system” in these two states P_{\pm} are given by (8.2) as

$$\begin{aligned} P_+ &= C e^{+\mu H/k_B T}, \\ P_- &= C e^{-\mu H/k_B T}, \end{aligned} \quad (8.6)$$

where C is a coefficient that can be determined by requiring that the two probabilities add up to unity. This yields

$$C = \frac{1}{e^{+\mu H/k_B T} + e^{-\mu H/k_B T}}. \quad (8.7)$$

The thermal average of s_i can then be calculated as

$$\langle s_i \rangle = \sum_{s_i=\pm 1} s_i P_{\pm} = P_+ - P_- = \tanh(\mu H/k_B T). \quad (8.8)$$

This is the exact result for the behavior of a single spin in a magnetic field. We now use it to obtain an *approximate* solution for a system of N interacting spins. The mean field approximation is based on the *assumption* that the interaction of a spin s_i with its neighboring spins, which is what yields the first term on the right-hand side of (8.5), is equivalent to an *effective* magnetic field acting on s_i . The result for $\langle s_i \rangle$ can then be calculated using (8.8), with H replaced by the effective field, H_{eff} . It remains for us to estimate H_{eff} .

The energy function (8.5) can be rewritten in the suggestive form

$$E = - \left(J \sum_{\langle ij \rangle} s_j \right) s_i - \mu H s_i, \quad (8.9)$$

which shows that the term involving J (which describes the interaction of s_i with its neighbors) has the form of a magnetic field with $\mu H_{\text{eff}} = J \sum s_j$. Now comes the approximation. We assume that the spin variables s_j in this expression for H_{eff} can be replaced by their thermal averages. Since all of the spins have the same average alignment, their thermal average values will all be the same. Denoting this by $\langle s \rangle$ (since we can now drop the subscripts) and assuming that the “true” externally applied field $H = 0$, we have

$$H_{\text{eff}} = \frac{J}{\mu} \sum \langle s \rangle = \frac{z J}{\mu} \langle s \rangle, \quad (8.10)$$

where z is the number of nearest neighbors. Combining this with (8.8) leads to the result

$$\langle s \rangle = \tanh(z J \langle s \rangle / k_B T). \quad (8.11)$$

This is an *implicit* relation for $\langle s \rangle$, which cannot be solved analytically except in certain limits, such as when $\langle s \rangle$ is small, as we will discuss in a moment. We

therefore consider a numerical approach. To illustrate the nature of the problem, Figure 8.2 shows a schematic plot of the two sides of (8.11), both as functions of $\langle s \rangle$. The value(s) of $\langle s \rangle$ at which the two curves intersect are the solutions we are after, since intersection is just a graphical way of expressing the equality (8.11). There is always a solution at $\langle s \rangle = 0$ corresponding to the thermal average with zero magnetization. This is usually referred to as the *paramagnetic phase*. At low temperatures (on the left in Fig. 8.2), the tanh function in (8.11) has a large slope near the origin and this leads to two additional solutions, $\langle s \rangle = +s_0$ and $-s_0$. Since these solutions have a nonzero value of $\langle s \rangle$, they correspond to phases with a nonzero magnetization; they are the *ferromagnetic phases*, with an "up" and a "down" magnetization, respectively.

At low temperatures, where there are multiple solutions of (8.11), it is useful to consider the free energies of different solutions.¹³ It turns out that the solutions with $\langle s \rangle \neq 0$ have a lower free energy than the paramagnetic solution, as illustrated schematically in Fig. 8.3. So, mean field theory predicts that the system is indeed ferromagnetic at low temperatures. The two solutions $\langle s \rangle = \pm s_0$ have equal free energies, and thus they are equally probable. This is due to the symmetry of (8.5) with respect to the sign change (reversal) of all of the spins when $H = 0$.¹⁴ This symmetry will be lost for any non-zero H , resulting in only one global minimum for the free energy for $H \neq 0$.

Our next job is to calculate the solution(s) of (8.11), that is, to find $\langle s \rangle$ as a function of temperature. Aside from locating the intersections graphically (as we have done in Fig. 8.2), there are many ways to solve (8.11) numerically. Once we express this equation as

$$f(\langle s \rangle) \equiv \langle s \rangle - \tanh(zJ\langle s \rangle/k_B T) = 0, \quad (8.12)$$

any of the root finding methods discussed in Appendix B can be used. For situations like this where the function involved is smooth and easy to differentiate analytically, the best approach is usually the Newton-Raphson method.¹⁵

The positive branch of the mean field solution for $\langle s \rangle$ as a function of temperature is shown in Figure 8.4. Since the magnetization is proportional to $\langle s \rangle$ there is a spontaneous magnetization $M > 0$ at low temperatures, that is, the system is ferromagnetic. At high temperatures the disordering effect of temperature dominates, and the system is paramagnetic with $M = 0$. Though M is continuous, its slope changes discontinuously as $M \rightarrow 0$. In this sense, the transition between these two phases is abrupt, occurring at what is known as the critical temperature, T_c . Mean field theory predicts that $T_c = 4$ in this case (where temperature is measured in units of J/k_B).

¹³We appeal here to your statistical physics background on the meaning and role of the free energy in such systems. One knows from statistical mechanics that a system will always assume the state with the lowest possible free energy. For details on how to calculate these free energy curves and a full discussion of how to interpret them, see Stanley (1971).

¹⁴As we will see later, which of the two phases (or even a mixture of the two) actually occurs is usually determined by things like the boundary conditions, and the way that the system is prepared.

¹⁵If the function is difficult to differentiate analytically, one can always use the secant method (see Appendix B).

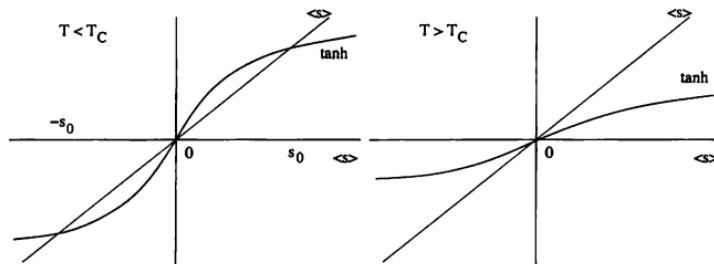


FIGURE 8.2: The solutions of (8.11) correspond to the intersections of the function $\tanh(zJ(s)/k_B T)$ and the function $\langle s \rangle$. Left, at low temperatures (below the critical temperature, T_c), the tanh function has a large initial slope, resulting in three intersections, i.e., three solutions. Two of the solutions occur at nonzero values $\langle s \rangle = \pm s_0$, while the third solution is at the origin ($\langle s \rangle = 0$). Right, at high temperatures (above T_c) the tanh function has a small initial slope, and there is only one intersection, at $\langle s \rangle = 0$.

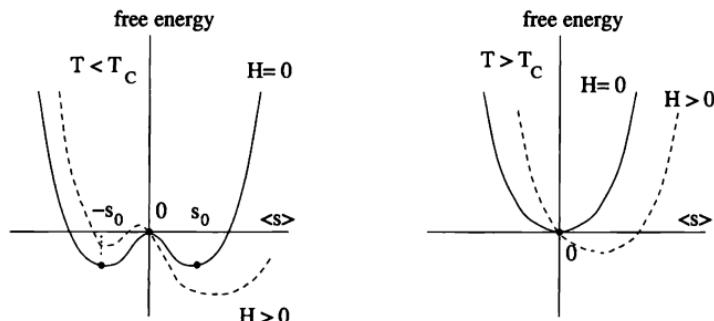


FIGURE 8.3: Schematic plots of the free energy of a spin system as a function of $\langle s \rangle$ at different temperatures. The solid lines are for $H = 0$, while the dashed lines are for $H > 0$. Left, at low temperatures, $T < T_c$. For $H = 0$ there are three solutions to the mean field equation, (8.11); these are shown as the heavy dots, and correspond to the three solutions shown on the left in Fig. 8.2. The solutions at $\langle s \rangle = \pm s_0$ are global minima of the free energy; they are the stable states of the system. The solution at the origin corresponds to a local maximum of the free energy. When the field is nonzero (the dashed curve), the minimum at $\langle s \rangle > 0$ becomes the global minimum and the one at $\langle s \rangle < 0$ is at best a local minimum (for sufficiently large H , it disappears entirely). Right, mean field free energy at high temperatures, $T > T_c$. For $H = 0$, the origin corresponds to the unique (and only) minimum of the free energy. For $H > 0$, there is still a unique minimum, but it is at a positive value of $\langle s \rangle$.

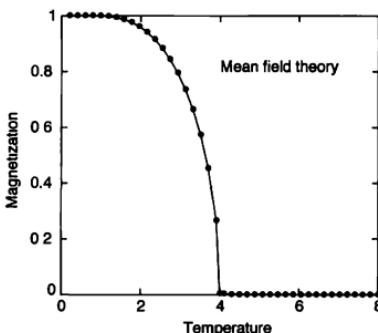


FIGURE 8.4: Numerical solution of the mean field equation (8.11). Here we have taken $J/k_B = 1$ and $z = 4$ corresponding to the number of nearest neighbors in a square lattice.

This is an example of a second-order phase transition and has some important features in common with the percolation transition studied in Chapter 7. The spontaneous magnetization is what is known as the *order parameter* for this transition; roughly speaking, it tells what phase the system is in. Here a nonzero value of the order parameter M is found when the system is in the ferromagnetic phase, while $M = 0$ means that it is paramagnetic. The results in Figure 8.4 suggest that M vanishes in a singular manner at T_c , as the slope dM/dT becomes extremely large as $T \rightarrow T_c$. The form of this singularity can be obtained analytically using the fact that $\tanh(x) \approx x - x^3/3$ for small x . Thus, when $\langle s \rangle$ is small (8.11) becomes

$$\langle s \rangle \approx \frac{zJ\langle s \rangle}{k_B T} - \frac{1}{3} \left(\frac{zJ\langle s \rangle}{k_B T} \right)^3, \quad (8.13)$$

which has the solutions $\langle s \rangle = 0$ and

$$\langle s \rangle = \sqrt{\frac{3}{T} \left(\frac{k_B T}{zJ} \right)^3} \left(\frac{zJ}{k_B} - T \right)^{1/2} \sim (T_c - T)^\beta, \quad (8.14)$$

where in the last step we have made the identification $T_c = zJ/k_B$ ($= 4$ for our choice of units for T), in agreement with the numerical results in Figure 8.4), and $\beta = 1/2$. The parameter β is a *critical exponent*. This behavior is analogous to what we observed near the percolation transition.¹⁶ A second-order phase transition is characterized by power law behavior of many properties. We will describe the power law singularities associated with several other quantities in the next section, where we will introduce the Monte Carlo method for calculating the behavior. It will turn

¹⁶In that case the fraction of sites in the spanning cluster played the role of the order parameter

out that while mean field theory does lead to the correct power law form for $\langle s \rangle$, the value of the critical exponent β predicted by mean field theory is not correct.

EXERCISES

- 8.1. Solve the mean field equation (8.11) numerically using any of the root-finding methods discussed in Appendix B (or elsewhere). Compare the results near T_c with the analytic result (8.14).
- 8.2. At low temperatures the tanh function in (8.11) can be expanded to yield an analytic result for $\langle s \rangle$ in the limit of small T . Do this, and compare it with the numerical solution.

8.3 THE MONTE CARLO METHOD

We have already remarked that mean field theory predicts there will be an abrupt transition between the ferromagnetic and the paramagnetic phases of the Ising model. We also noted that while the mean field approximation is correct in predicting singular behavior near T_c [usually power laws such as (8.14)], it generally does not yield correct values of the critical exponents. A more powerful approach is required to get a quantitatively accurate picture of the behavior, especially near a phase transition. One such approach is the *Monte Carlo method*. This method is similar in spirit to the simulations used in Chapter 7, but with extensions to take into account the crucial role played by the energy and temperature of the system.¹⁷ Rather than beginning with an abstract definition of the method, we will instead describe how it works in practice. After having gained some perspective on its operation, we will then discuss why it works.

Our goal is to simulate how a spin system interacts with its environment; in the language of statistical physics the environment is often termed a *heat bath*. We will consider the particular case of a collection of Ising spins, but the method can readily be extended to deal with more complicated situations.¹⁸ According to statistical mechanics, the role of a heat bath is to exchange energy with the spin system and thereby bring it into equilibrium at some temperature T . As the system gains energy from, or loses energy to, the bath, spins are flipped, causing the system to move to new microscopic states. Each of these microstates corresponds to a *particular* arrangement of the spins. The measured values of quantities such as the magnetization then depend on the probabilities of finding the system in its different microstates.

The Monte Carlo method uses a stochastic approach to simulate the exchange of energy between the spin system and the heat bath. We begin with the system in a particular microstate as illustrated schematically in Figure 8.1. The interaction with the bath is then modeled as follows. A spin is chosen¹⁹ and the energy required

¹⁷Other useful approaches include *high temperature* and *low temperature expansions*. These methods employ enumeration (somewhat similar to the enumeration of the SAWs of Chapter 7) to calculate systematic approximations to various thermodynamic functions.

¹⁸While our treatment here makes use of the canonical ensemble, it can also be modified to apply to the microcanonical ensemble. If you are not familiar with these statistical ensembles, see Reif (1965).

¹⁹The spin can be chosen either at random or by stepping systematically through the lattice.

to make it flip, E_{flip} , is calculated. For our Ising model this energy is calculated using (8.1). If E_{flip} is negative (so that the energy would be lowered by reversing the spin), the spin is flipped and the system moves into a different microstate. If E_{flip} is positive (so that the energy of the system would be increased), a decision must be made. A random number that is distributed uniformly in the range between 0 and 1 is generated, and compared to the Boltzmann factor $\exp(-E_{\text{flip}}/k_B T)$. If this Boltzmann factor is larger than the random number, the spin is flipped, otherwise the spin is left undisturbed. Hence, in this case the system may or may not move to a different microstate, depending on the value of the random number. This completes one Monte Carlo time step. Another spin is then chosen, E_{flip} is calculated, and the spin is either flipped or left unchanged according to the algorithm just described. This is called the *Metropolis algorithm*.²⁰ This procedure is repeated a large number of times, so that every spin is given many chances to flip. We can view each Monte Carlo time step as one interaction with the heat bath. The effect of this interaction varies according to the temperature, since T enters through the Boltzmann factor probability for flipping a spin.

Let us now consider why this Monte Carlo algorithm correctly mimics the interaction with a heat bath. As we noted above, a chosen spin is *always* flipped when $E_{\text{flip}} < 0$, that is, when it would *lower* the energy of the system. If this were the only flipping criteria, the system would rapidly move to the state with lowest energy,²¹ which for our model is a microstate with all spins parallel to each other, the completely aligned, ferromagnetic state. However, we must also allow for transitions to states of *higher* energy, whose probabilities should be governed by the Boltzmann factor $\exp(-E_{\text{flip}}/k_B T)$ with $E_{\text{flip}} > 0$. At low temperatures this factor is small, so the probability of flipping a spin to a higher energy state is very low. Hence, at low temperatures the system will usually be found in a microstate that is very close to the fully aligned state (the state with the lowest energy). That is, the system will be ferromagnetic, although M will not have the value corresponding to complete alignment. At high temperatures the Boltzmann factors will not be as close to zero, so the probability for flipping to a higher energy state will be quite significant. Indeed, as T becomes very large the Boltzmann exponential factor will approach 1, and this will make the spin arrangement tend toward complete randomness. This is the extreme paramagnetic state where every spin configuration is equally likely, regardless of its energy.

These arguments show that the Monte Carlo flipping procedure gives the correct qualitative picture in both the high- and low-temperature limits. We now consider things quantitatively. A Monte Carlo spin flip connects two microstates; let us call their energies E_1 and E_2 and assume that $E_1 > E_2$. If the system is in state 1, the flipping rules specify the probability for flipping the selected spin during a given time step, hence the *rate* of transitions from state 1 to state 2. We will call this rate

²⁰In honor of its inventor, Nicholas Metropolis. It is the most common algorithm used in Monte Carlo simulations, but there are many other ways to accomplish the same goal of simulating an equilibrium ensemble.

²¹Assuming that this ground state can be reached by flipping one spin at a time. We will not worry here about possible complications associated with metastable states, although they may be important in practice, as we will see later in this chapter.

$W(1 \rightarrow 2)$ and since by assumption $E_1 > E_2$, the rules tell us that $W(1 \rightarrow 2) = 1$. Likewise, if the system is in state 2, the probability of a Monte Carlo transition to state 1 during the next time step is $W(2 \rightarrow 1) = \exp[-(E_1 - E_2)/k_B T]$, since $E_{\text{flip}} = E_1 - E_2 > 0$. When our system reaches thermal equilibrium the probability of finding it in any particular state will, on average, be independent of time, so we expect that the number of transitions from state 1 to state 2 must be equal to the number of transitions in the reverse direction. The number of transitions of a particular kind is proportional to the product of the transition rate W and the probability of the system being found in the appropriate initial state. Equating the number of transitions $1 \rightarrow 2$ and $2 \rightarrow 1$ then gives

$$P_1 W(1 \rightarrow 2) = P_2 W(2 \rightarrow 1), \quad (8.15)$$

where P_1 and P_2 are the probabilities of the system being found in these two microstates.²² Inserting our results for the W factors we find

$$\frac{P_1}{P_2} = \exp[-(E_1 - E_2)/k_B T]. \quad (8.16)$$

Comparing this behavior with (8.2) we see that this is precisely what is expected for a system in thermal equilibrium. The Metropolis Monte Carlo algorithm thus leads to a situation in which the relative probabilities of being found in different microstates are given by the correct Boltzmann factors. Assuming that all of the microstates are accessible via the flipping rules, this procedure will indeed simulate a system in contact with a heat bath.

This concludes our brief introduction to the Monte Carlo method. In the next several sections we will use it to study the behavior of the Ising model and investigate phase transitions in that system. However, before moving on we should make a few further comments about the method. First, while we have described how it can be used to simulate a spin system in equilibrium with a heat bath, the method is not restricted to spin systems. Any system can be studied in this way. All that is needed is some way of enumerating the microstates and an expression for the energy of each of these states, so that the appropriate flipping rules, that is, transition rules, can be formulated. Second, a system in equilibrium with a heat bath is described by what is known as the canonical ensemble. It is also possible to modify the flipping rules so that the Monte Carlo method simulates either the microcanonical or the grand canonical ensemble, but this is more than we want to tackle here.

8.4 THE ISING MODEL AND SECOND-ORDER PHASE TRANSITIONS

We are now ready to apply the Monte Carlo method to the study of phase transitions in the Ising model. We will encounter two types of transitions, which are classified as first-order and second-order. Rather than trying to give a completely general

²²Such a relation is called the *detailed balance* condition. It is known to be a sufficient condition to achieve thermal equilibrium, unless the system is rather pathological (e.g., if the system is not ergodic, etc.)

definition of these classifications and all that they entail, we will illustrate them by example in this and the following section.

Our model is again a collection of Ising spins arranged on a square lattice as illustrated in Figure 8.1. The spins interact with each other and with a magnetic field according to the energy function (8.5). We begin by considering only the case with $H = 0$, leaving the behavior as a function of field to the next section. For convenience we measure energy in units of J , so that T is measured in units of J/k_B .²³

We have already considered the mean field solution for the spontaneous magnetization (Figure 8.4), so our first goal will be to compare the Monte Carlo results with that prediction. Since the construction of a Monte Carlo program for the Ising model presents no new programming challenges, we only show a brief sketch of such a program below.

EXAMPLE 8.1 Monte Carlo algorithm for the Ising model on an $L \times L$ square lattice

- Set the desired T and H .
- Initialize all spins s_i ($i = 1, 2, \dots, L^2$). (We can take $s_i = 1$ for all i , for example.)
- Perform the desired number of Monte Carlo sweeps through the lattice.
 - ▷ For a given sweep, loop through the L rows (or columns) of the lattice. On each row (or column), consider each spin for updating.
 - Calculate E_{flip} , the energy required to flip the selected spin. For nearest-neighbor interactions, E_{flip} only depends on the nearest neighbors. For spins on an edge, apply the chosen boundary condition (see below) to determine which spins are the neighbors.
 - If $E_{\text{flip}} \leq 0$, flip the spin.
 - If $E_{\text{flip}} > 0$, generate a uniform random number r between 0 and 1, and compare it with $\exp(-E_{\text{flip}}/k_B T)$.
 - ▷ If $r \leq \exp(-E_{\text{flip}}/k_B T)$, flip the spin.
 - ▷ If $r > \exp(-E_{\text{flip}}/k_B T)$, leave it undisturbed.
 - ▷ After each sweep is completed, record the new energy, magnetization, and any other quantities of interest.
- Store (and/or plot) the recorded thermodynamic quantities.

The calculation consists of using the Monte Carlo rules described in Section 8.3 to simulate the spin system over the course of many Monte Carlo time steps. During each step we choose a spin, calculate the energy that would be required to flip it

²³Hence, T and H are treated as effectively unitless, for convenience. If you wish, you can think of the values of T given here as being in units of J/k_B , with H given in units of J/μ

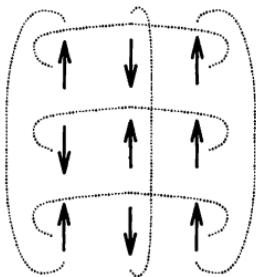


FIGURE 8.5: Ising model with periodic boundary conditions. Spins that are connected by the dotted lines that wrap around the lattice are considered to be nearest neighbors and thus to interact with exchange energy J .

according to (8.5), and then either flip it or leave it unchanged, depending on how the Boltzmann factor compares to a random number that is uniformly distributed in the range 0–1. It is convenient to choose spins systematically by moving along successive rows of the lattice, but one can also choose spins at random. So long as many time steps are considered so that each spin has many opportunities to flip, the results do not depend on how the spins are chosen. After each complete sweep through the lattice one computes quantities such as the total magnetic moment $M = \sum s_i$ and total energy [according to (8.5)]. The averages of M and E over many sweeps through the lattice then give the values for thermal equilibrium.

Ideally we would like to calculate the behavior of very large systems. A real magnet, such as a piece of iron, would usually contain a large number of spins (of order 10^{23}), and one of our goals is to understand the behavior of a real system. Moreover, it will turn out that the phase transitions described above occur only in the limit of very large systems. However, the computer time required for an accurate simulation will limit us to lattices that contain relatively modest numbers of spins. (We show the results from no more than a few hundred spins in this book, although your computer will likely be able to handle much larger numbers.) For such a finite system, the behavior of spins at the edges can have a pronounced effect.

With a square lattice these edge spins have only three nearest-neighbor spins, or two if they are at a corner, as compared to four neighbors for the interior spins. Since it is the exchange interaction (8.1) that causes neighboring spins to tend to point in the same direction, and since the edge spins have fewer neighbors, the spins at the edges of the lattice will have less of a tendency to align with the other spins. While real systems will also have edge spins, the fraction of spins at the edges, as compared to the number in the interior, will be much greater for a small lattice. Hence, it is important in our simulations to minimize the effects of the edges as much as possible. One way to accomplish this is to use *periodic boundary*

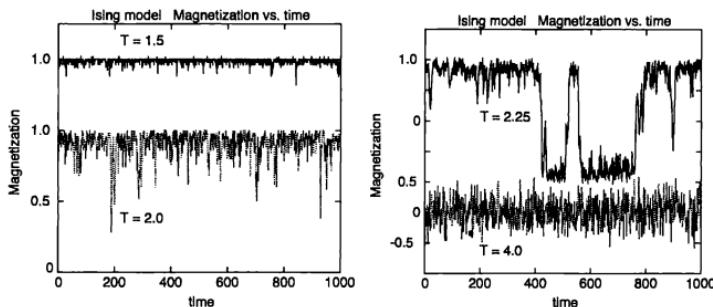


FIGURE 8.6: Magnetization versus time for the Ising model on a 10×10 square lattice at several different temperatures. Note that we have normalized the magnetization so that $M = 1$ corresponds to complete alignment of all of the spins. Left: at relatively low temperatures compared to T_c , with the results for $T = 1.5$ offset upwards for clarity; right: at temperatures near and above T_c (the two results are again offset for clarity).

conditions as illustrated in Figure 8.5. In our original description of the Ising model we specified that spins interact only with their nearest neighbors. Now we assume that a spin on the edge interacts also with the spin on the *opposite* edge of the lattice, as indicated by the dotted lines that wrap around the lattice. Alternatively we could imagine that our lattice is situated on a torus so that these “edge” spins are really neighbors of each other. In either case we have, in a sense, eliminated all of the edge spins. Every spin now has four nearest neighbors so that all spins are in equivalent locations. The use of periodic boundary conditions thus allows us to largely circumvent boundary effects. However, this does *not* mean that a 5×5 lattice with periodic boundary conditions will behave in the same way as a much larger lattice. For an $L \times L$ lattice with periodic boundary conditions the distance between two spins cannot be larger than $L/\sqrt{2}$ lattice spacings, the maximum separation along a diagonal. Precisely why this is important will be discussed below.

Periodic boundary conditions are often used in Monte Carlo simulations and in many other types of calculations, but they are certainly not the only boundary conditions that we can imagine. We could instead choose *free boundary conditions* in which the spin system simply terminates at the edges, or *fixed boundary conditions* in which the spins at one edge are all constrained to point in a certain direction. For our work in this and the next section we will employ periodic boundary conditions, as these will tend to minimize the effects of finite lattice size. However, as mentioned above, this will not completely eliminate such effects. The best way to calculate the properties of a very large system is to perform simulations on a number of lattices of different sizes, and to then extrapolate to the infinite lattice limit. We will not have occasion to do this in any detail in our calculations here, but this procedure will be explored in the exercises and is described in the references.

For our simulations we begin at low temperatures where we know that the system will be in the ferromagnetic phase. We therefore choose an initial spin configuration with all spins along the positive direction, that is, the fully aligned state, and use the Monte Carlo procedure to calculate the magnetization as a function of Monte Carlo time as the simulation proceeds. Some results for a 10×10 lattice at several temperatures are shown in Figure 8.6. Here one unit of time²⁴ corresponds to one complete sweep through the lattice, so that every spin had the opportunity to flip once during each time step. This time unit is often referred to as a "Monte Carlo step per spin." At the lowest temperature the magnetization stays very close to the saturation value corresponding to all of the spins being parallel. While the Monte Carlo rules do lead to the occasional flipping of a spin to the negative direction, the fluctuations in M are small. When the temperature is raised to $T = 2.0$, the average value of M decreases to a value corresponding to about 90 percent of the fully aligned value, since the Boltzmann factor at this temperature favors a spin flip to the higher energy state of order 10 percent of the time. Our system is still ferromagnetic, but the degree of order is reduced from its value at lower temperatures. In addition, the magnitude of the *fluctuations* increases significantly. These fluctuations are important for several reasons. First, we will see shortly how they can be used together with the *fluctuation-dissipation* relation of statistical mechanics to calculate several other thermodynamic quantities. Second and more importantly, these enhanced fluctuations signal that we are approaching a second-order phase transition, also known as a *critical point*. A system at its critical point is extremely sensitive to small perturbations, as its properties change very rapidly in response to changes in temperature, magnetic field, etc. We will see that these fluctuations are intimately related to the singularities at T_c .

When we warm further to $T = 2.25$ the fluctuations become larger, as the system fluctuates between values as large as $M \sim \pm 0.8$. Hence, there are fluctuations in which the magnetic moment of the *entire* system changes direction.²⁵ For the Ising model on a square lattice, exact analytic calculations yield the result $T_c = 2/\ln(1 + \sqrt{2}) \approx 2.27$, so we are very close to the critical point.²⁶ Proceeding to higher temperatures, we find that at $T = 4$ the fluctuations decrease in magnitude and are centered around $M \sim 0$. We are now well above T_c , in the paramagnetic phase. Comparing the behavior at different temperatures we see that the fluctuations are largest near T_c . We will analyze them in more detail shortly.

From the simulations at different temperatures we can calculate the average values of M over time to obtain the magnetization as a function of T , and some results are shown in Figure 8.7. When computing these averages, and in most other applications of the Monte Carlo method, one must be careful to allow enough time (i.e., enough Monte Carlo time) for the system to reach equilibrium. The

²⁴This unit of time is a product of the Monte Carlo method, since the Monte Carlo algorithm does not employ the microscopic equations of motion. It is possible (with some assumptions) to relate Monte Carlo time to "real" time, but that problem is more than we wish to tackle here.

²⁵The probability for such complete flips decreases as the system is made larger.

²⁶This exact value is much lower than the mean field estimate that we derived earlier.

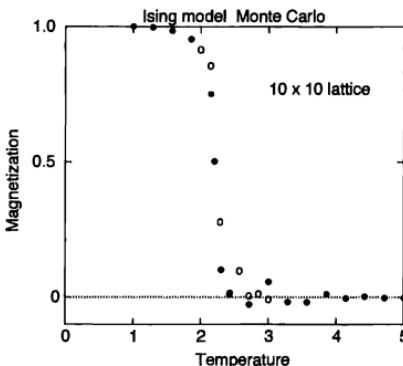


FIGURE 8.7: Spontaneous magnetization as a function of temperature for the Ising model on a 10×10 square lattice. The different symbols show results for two separate simulations. At each temperature M was obtained by averaging over 1000 sweeps through the lattice.

equilibration time will depend on the temperature, and on how far the initial state is from the final equilibrium state. In Figure 8.6 the equilibration time was very short (on the scale of these figures). However, as one approaches a phase transition the equilibration time can become very long.²⁷

While there is some scatter in the results due to the statistical errors associated with averaging data like that in Figure 8.6, M drops precipitously to zero at a value of T which is consistent with the exactly known value of the critical temperature as just mentioned ($T_c \approx 2.27$). The overall quality of these results could be improved in several ways. First, we could simply let the simulations run longer so that more Monte Carlo steps are employed in the averaging at each temperature. This would reduce the statistical errors at a rate roughly proportional to $N_{\text{steps}}^{-1/2}$, where N_{steps} is the number of Monte Carlo time steps used in the calculation.²⁸ Second, we could study the behavior of larger lattices and extrapolate to the infinite lattice limit. We will leave these tasks to the exercises.

It is interesting to compare the Monte Carlo results for M with the prediction of mean field theory, Figure 8.4. While the two results have the same qualitative form, mean field theory overestimates T_c by nearly a factor of 2. In addition, the mean field prediction for M goes to zero at T_c somewhat more gradually than we

²⁷In fact, the equilibration time will usually diverge at a critical point, an effect known as *critical slowing down*. This is an important consideration when analyzing Monte Carlo results near a critical point.

²⁸This should remind you of our discussion of averaging and the central-limit theorem in Appendix G. While this estimate of the Monte Carlo uncertainty is qualitatively correct, there are some subtleties associated with long correlation times near T_c , which we will leave to the references (Binder and Heermann [1992] and Stanley [1971]).

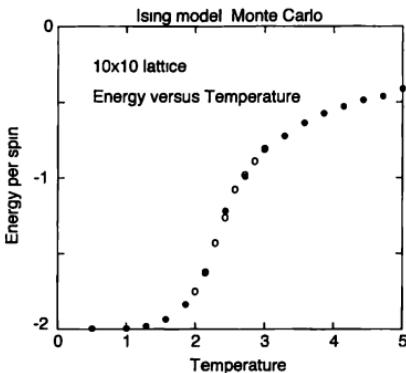


FIGURE 8.8: Thermal average of the energy versus temperature for the Ising model on a 10×10 square lattice. These results were obtained in the simulations used to calculate M in Figure 8.7. Note that $J = 1$ for this simulation and that we normalized $\langle E \rangle$ to obtain the energy per spin, $\langle E \rangle/N$. In the low-temperature (fully aligned) limit, we argued that $\langle E \rangle/N$ should approach $-2J = -2$, which is in agreement with these Monte Carlo results.

find from the Monte Carlo results. As we have already mentioned, the variation of M near T_c is determined by the critical exponent β according to

$$M \sim (T_c - T)^\beta \quad (8.17)$$

Mean-field theory predicts $\beta = 1/2$, while the exact (analytic) result in two dimensions is $\beta = 1/8$. The Monte Carlo results in Figure 8.7 would not yield a terribly precise value of β (we'll discuss how to do a reasonable job of this in the exercises), but they are in much better agreement with the exact value than with mean field theory. Large-scale Monte Carlo simulations (see the references) yield a value of β , which has an uncertainty of less than 1 percent, and which agrees well with the analytic result. Hence, while mean field theory yields a reasonable qualitative picture for the behavior, the quantitative details are not correct. We will consider where mean field theory goes wrong in a moment.

So far, the only property we have considered is the magnetization, but much can be learned from the examination of other quantities. In Figure 8.8 we show the variation of the energy with temperature. This was obtained from the same simulations used to calculate M ; after each Monte Carlo sweep through the lattice our program recorded E [calculated using (8.5)]. At low temperatures, with the spins fully aligned, every spin has an interaction energy of $-J$ with each of its four nearest neighbors. The total energy at $T = 0$ should thus be $-4NJ/2$, where N is the number of spins and the factor of 2 is inserted since we have counted each pair of spins twice. The Monte Carlo results are in good agreement with this result in

the limit $T \rightarrow 0$. On the other hand, at very high temperatures the spins will be randomly oriented, so on average each spin will have two neighbors that are aligned parallel and two that are antiparallel. The thermal average of the total energy in this limit will thus be zero. While the results in Figure 8.8 are tending toward $\langle E \rangle = 0$ at high temperatures,²⁹ they are substantially below this value even at $T = 5$, which is more than twice the critical temperature. This implies that the neighbors of any particular spin are *not* randomly oriented above T_c , even though we know through the results for M that the average alignment of the entire lattice is zero. This indicates that the orientations of neighboring spins are *correlated*, which will turn out to be very important.

These arguments allow us to understand the behavior of $\langle E \rangle$ at both high and low temperatures, but the behavior near T_c is not as simple. We see from Figure 8.8 that $\langle E \rangle$ exhibits an inflection point with a very large slope near T_c . In fact, for an infinitely large system the derivative $d\langle E \rangle/dT$ is infinite at T_c . From thermodynamics we know that the specific heat is related to the energy by $C = d\langle E \rangle/dT$, so this means that the specific heat diverges at T_c . This divergence is another of the singularities associated with the critical point. As with the magnetization, this singularity is described in general by a power law

$$C \sim \frac{1}{|T - T_c|^\alpha}, \quad (8.18)$$

where α is yet another critical exponent.³⁰ There are two ways to study the behavior of the specific heat using the Monte Carlo method. One is to numerically differentiate³¹ the results for E . The other is to study the fluctuations of E as a function of time. When we first considered the calculation of M we noted that its value fluctuates with time and that the magnitude of these fluctuations becomes large near T_c . A convenient measure of the size of such fluctuations is the variance. For example, if we consider the energy as a function of time, we could generate time-dependent plots that are qualitatively similar to those shown for the magnetization in Figure 8.6 (we will leave this task for the exercises). From such results we can compute the average energy as shown in Figure 8.8

$$\langle E \rangle = \frac{1}{N_m} \sum_{\alpha} E_{\alpha}, \quad (8.19)$$

²⁹The angular brackets again denote a thermal average

³⁰For simplicity, and in accordance with the *scaling* theory (see Section 8.6), we assume that divergences above and below T_c are described by the same exponent; in the language of critical phenomena and critical exponents, this means that for the specific heat singularity (8.18), $\alpha = \alpha'$. We will make a similar assumption for other exponents. See Stanley (1971) for more on this point. For the Ising model in two and three dimensions, it turns out that C is divergent at T_c . However, for some spin models $\alpha < 0$, so C is singular but does not diverge. We should also add that for the two-dimensional Ising model $\alpha = 0$. An exponent of zero in (8.18) might seem a bit odd, but it turns out that when (8.18) is suitably generalized, an exponent value of zero corresponds to a logarithmic singularity. This is also discussed in the texts by Goldenfeld, Plischke and Bergersen, and Stanley.

³¹Differentiating such noisy "data" numerically generally leads to large uncertainties. This is studied in the exercises.

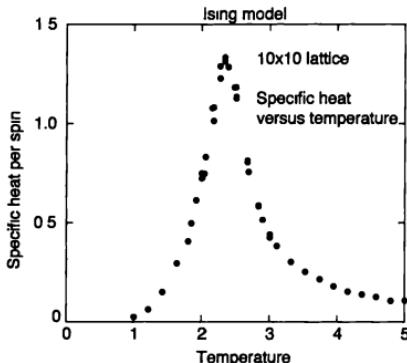


FIGURE 8.9: Heat capacity calculated using the fluctuation-dissipation relation for a 10×10 lattice. Here we plot the heat capacity per spin, C/N , and take $J = 1$.

where the sum is over N_m microstates α generated by the Monte Carlo simulation. Likewise we can consider the variance

$$(\Delta E)^2 \equiv \langle E^2 \rangle - \langle E \rangle^2 , \quad (8.20)$$

where

$$\langle E^2 \rangle = \frac{1}{N_m} \sum_{\alpha} E_{\alpha}^2 . \quad (8.21)$$

According to the fluctuation-dissipation theorem of statistical mechanics, the variance of the energy is related to the specific heat by

$$C = \frac{(\Delta E)^2}{k_B T^2} . \quad (8.22)$$

This shows that the singularity in the specific heat at the transition is *directly* connected with the extremely large fluctuations found near T_c . Equation (8.22) can also be used to calculate C using the Monte Carlo results for E as a function of (Monte Carlo) time.

The behavior of C estimated using (8.22) is shown in Figure 8.9. As we had anticipated, C exhibits a large peak in the vicinity of T_c . The value observed here does not diverge because the lattice has a finite size (recall that the singularities associated with a phase transition are, strictly speaking, only found in an infinite system). By studying the behavior as a function of lattice size, as we will explore in the exercises, one can show that the peak in the specific heat per spin does indeed increase in magnitude as the system is made larger.

The fluctuation dissipation relation applies to a number of other quantities, one of which is the susceptibility $\chi \equiv dM/dH$. This is a measure of how much

magnetization is induced by the application of a magnetic field. In this case the fluctuation-dissipation theorem yields

$$\chi = \frac{(\Delta M)^2}{k_B T}, \quad (8.23)$$

where $(\Delta M)^2$ is the variance of M , which can be calculated from results such as those in Figure 8.6. It turns out that χ also diverges at the critical point and is described by a power law form similar to (8.18), but with a different critical exponent known as γ . We will explore this more in the exercises. It is interesting that the fluctuations can be used to estimate how the system would respond to a magnetic field, even though the simulation is performed in zero field. This is an intriguing aspect of the fluctuation-dissipation theorem.

When we considered the variation of the energy with temperature in Figure 8.8, we noted that the behavior of $\langle E \rangle$ above T_c implies that there must be significant correlations in the relative alignment of neighboring spins. This can be pictured as follows. Above T_c the magnetization is zero so that on average, half the spins point up, while the other half point down. The energy of interaction between neighboring spins is not strong enough to produce a common alignment of all of the spins, so if you hadn't already seen our results for the energy above T_c , you might think that the spin arrangement above T_c would be random from one spin to the next. We will now show why this is not the case.

Consider a particular spin, call it s_0 , and assume that it points up. Let us examine the alignment of the four neighbors of s_0 . Since $s_0 = +1$, the neighbors will have a lower energy if they also point up. Even though the temperature may be above T_c so that the average alignment over the entire system is zero, these four neighbors will still have a higher probability of being aligned parallel to s_0 , as opposed to being antiparallel. The degree to which they are parallel will depend on temperature, but it will *not* be zero even above T_c . This argument can also be applied to any one of the neighbors of s_0 , call it s_1 . The near neighbors of s_1 will tend to be aligned with it, and since s_1 is correlated with s_0 , this tendency to be aligned will "propagate" from spin to spin through their common neighbors.

This tendency to be correlated can be measured using the *correlation function*

$$f(i) = \langle s_0 s_i \rangle, \quad (8.24)$$

where s_i is a spin that is located i lattice sites away from s_0 , and the angular brackets again denote a thermal average that can be computed by averaging over the microstates generated by the Monte Carlo algorithm. The correlation function $f(i)$ is our first encounter with a length-dependent property. It can be calculated by first choosing a spin to be the "central" spin s_0 , then computing the product $s_0 s_i$ for all spins a distance i from the central spin. The average of this product is best evaluated by letting each spin be the central spin and considering many Monte Carlo time steps. For a square lattice it is convenient to let i be an integer, thereby measuring distance in terms of the lattice spacing, and to calculate the correlations for spins separated along rows and columns. Note that for an $L \times L$ lattice with periodic boundary conditions, the maximum distance between two spins in the same

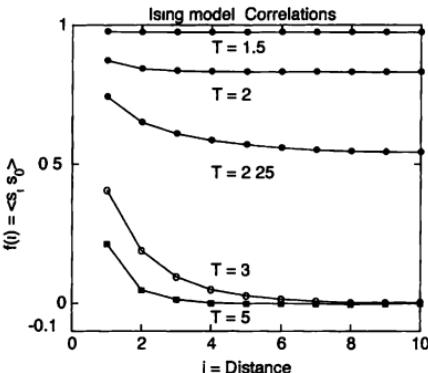


FIGURE 8.10: Correlation functions calculated for a 20×20 square lattice at several temperatures. The distance between spins is measured in units of the lattice spacing, and the two spins lie in the same row or column.

row or column is $L/2$, since this distance must be measured in terms of the smallest number of nearest neighbor bonds that connect the two spins.

Some results for the correlation function are shown in Figure 8.10, which shows $f(i)$ at several different temperatures. At $T = 1.5$ we find that $f(i)$ is nonzero at large distances. This is because the system has a nonzero value of M , so even spins that are very far apart will tend, on average, to point in the same direction. The important feature here is not the average value of $f(i)$, but rather the amount that $f(i)$ increases above this average value as i becomes small. At this low temperature $f(i)$ is nearly independent of separation. While the relative alignment does increase slightly at short distances, the enhancement is very small. Hence, in this case the magnitude of the correlations, which is measured by the *extra* alignment due to being in close proximity, is weak and limited to a very short range. At $T = 2$ the behavior is qualitatively similar, with only a small enhancement of the relative alignment at short distances. While this enhancement is larger than at $T = 1.5$, the spin correlations are still weak and short-ranged.

The picture changes near T_c . The correlation function at $T = 2.25 \sim T_c$ differs from the low temperature behavior in two ways. First, the relative alignment at short distances is *much* larger than the value at large i . Second, the correlations are now long range as $f(i)$ approaches the $i \rightarrow \infty$ limit *very* slowly as i is increased. As the temperature is increased further to temperatures above T_c (for example, $T = 5$), the correlations become smaller in magnitude and again extend over only a few lattice spacings.

The behavior of the correlation function sheds light on the singularities found at the critical point. It turns out that away from T_c the correlations fall off expo-

nentially with distance

$$f(i) \sim C_1 + C_2 \exp(-r_i/\xi), \quad (8.25)$$

where $C_1 (= \lim_{r_i \rightarrow \infty} f(i))$ is a constant that is zero above T_c , r_i is the distance between the spins, and ξ is known as the *correlation length*. As $r_i \rightarrow \infty$, the two spins s_0 and s_i become uncorrelated and

$$f(i) \rightarrow \langle s_0 \rangle \langle s_i \rangle = \langle s \rangle^2. \quad (8.26)$$

We can identify $C_1 = \langle s \rangle^2$, and rewrite (8.25) as

$$f(i) - C_1 = \langle (s_0 - \langle s \rangle)(s_i - \langle s \rangle) \rangle \sim C_2 \exp(-r_i/\xi). \quad (8.27)$$

Thus, ξ gives a measure of the *range* over which the correlations of the spin *fluctuations*, $s - \langle s \rangle$ (i.e., the deviation of s from its average $\langle s \rangle$), approach zero. As we noted from Figure 8.10, this range increases as T_c is approached. A careful calculation on a larger lattice would reveal that ξ diverges at T_c according to the power law

$$\xi \sim \frac{1}{|T - T_c|^\nu}, \quad (8.28)$$

where ν is another critical exponent. Right at the critical point, where $\xi = \infty$, the correlations fall off as a power law function of r . A large value of ξ means that the fluctuating orientation of a single spin influences the corresponding fluctuations of the spins that are very far away. Since ξ diverges at T_c this range becomes effectively *infinite*, so that the fluctuations of every spin is sensitive to those of *every* other spin. It is this extreme sensitivity of the system at T_c that leads to the singular behavior of quantities such as the magnetization and specific heat at the critical point. These enhanced fluctuations are ignored by the mean field approximation, which assumes that the relative alignment of the near neighbors is the same as the average alignment of the entire system. That is, mean field theory completely ignores local fluctuations.

EXERCISES

- 8.3.** Calculate M for the Ising model on a square lattice and try to estimate β . You should find a value close to $1/8$. Repeat this calculation for a triangular lattice. It turns out that β is the same for all regular two dimensional lattices. However, its value does depend on the dimensionality, as studied in the next problem.

Hint: There are several different ways to estimate a critical exponent from data such as Monte Carlo results. One is to perform a least-squares fit of the results to the power law expression (8.17). The slope of a plot of $\log(M)$ versus $\log(T_c - T)$ is equal to β , so you might think that a linear least-squares fit of $\log(M)$ as a function of $\log(T_c - T)$ (using the procedures described in Appendix D) would be suitable. However, there are two problems with this approach. One is that the value of T_c is usually not known ahead of time, but must be estimated at the same time as β is determined. Second, the power law (8.17) is obeyed only near the critical point; there will be deviations as the temperature moves away from T_c , but you don't know ahead of time at what value of $(T_c - T)$ these deviations

will become important. There are two ways to overcome these problems. One is to make a plot of M^{1/β^*} as a function of T . Here β^* is a *trial* value of β . By constructing such a plot with different values of β^* you can determine the value that yields a straight line as $T \rightarrow T_c$. This is the “best” estimate for β . A virtue of this approach is that it does not require that T_c be known, since it uses only the linearity of M^{1/β^*} with T and does not depend on where this line intercepts the temperature axis. A second approach is to construct plots of $\log(M)$ versus $\log(T_c - T)$ for a series of trial values of T_c . The preferred value of T_c is the one that gives a straight line as $T \rightarrow T_c$; the slope of this line then gives β . It is instructive to employ both methods to estimate β . You should find that the power law (8.17) with $\beta \approx 1/8$ is obeyed reasonably well for $2.0 < T < T_c \approx 2.27$.

- 8.4. Calculate M for the Ising model on a cubic lattice and try to estimate β (see the previous problem for some helpful hints). You should find a value close to 0.31. It is interesting that, as the dimensionality is made larger, the value of β approaches the prediction of mean field theory. In fact, β reaches 1/2 in four dimensions for the Ising model with nearest-neighbor interactions, and stays at this value for all higher dimensions.³² This calculation is not much more difficult (conceptually) than the previous problem, but will take more computer time.
- 8.5. Derive the fluctuation-dissipation relations (8.22) and (8.23).
- *8.6. Calculate χ for the Ising model on a two-dimensional square lattice using the fluctuation-dissipation relation, and show that it becomes very large near T_c .
- 8.7. Obtain the specific heat as a function of temperature for a 10×10 square lattice by differentiating the energy and through the fluctuation-dissipation theorem. Show that the two methods give the same result. Which approach is more accurate (for a given amount of computer time)?
- 8.8. The proper extrapolation to the infinite lattice limit is extremely important in a quantitative analysis of Monte Carlo results. It turns out that the manner in which this limit is approached also contains valuable information. Calculate the specific heat per spin, C/N , for $L \times L$ square lattices (pick several values of L in the range 5–40) and investigate how the maximum value (which is found at $T \approx T_c$, where T_c is the critical temperature of the infinite lattice) varies with lattice size. Show that $C_{\max}/N \sim \log L$. This behavior falls under the heading known as *finite size scaling* and is discussed by Fisher (1973).³³ Warning: The peak in the heat capacity becomes much sharper as L is increased, so you will need to use smaller temperature steps in locating the peak when L is large.
- *8.9. Study the time dependence of the fluctuations of either M or E at a fixed temperature. You should find that the fluctuations not only become larger in magnitude near T_c , but also become *slower*. This happens because the regions of correlated spins have a size of $\sim \xi$, which diverges at the critical point, and larger blocks of correlated spins take longer to fluctuate (that is, reorient). Try to obtain a quantitative measure of the time scale associated with the fluctuations. This effect is known as critical slowing down.

³²However, exactly at four dimensions, the power law (8.17) is multiplied by other singular factors. These “correction” terms are similar to what was mentioned in Chapter 7 for SAWs in four dimensions.

³³This logarithmic variation with L depends on the fact that the specific heat of the two-dimensional Ising model diverges logarithmically at T_c (this also leads to a value of the specific heat exponent $\alpha = 0$, as we have already mentioned). Quantities that are characterized by nonzero exponents display a different dependence on L , as discussed by Fisher (1973).

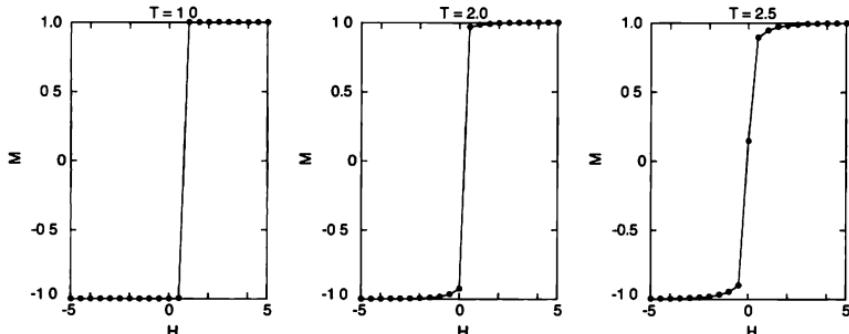


FIGURE 8.11: Field dependence of the magnetization for an Ising model on a 10×10 lattice at several temperatures. As in Figure 8.6, we have the normalized M so that the fully aligned state corresponds to $M = \pm 1$. These results were obtained by sweeping the field up from large negative values. Note the hysteresis near $H = 0$ when $T = 1.0$, as evidenced by the fact that M does not switch from negative to positive when the field first changes sign.

- 8.10. Compare the behavior of M for a system with periodic boundary conditions with the results for free boundary conditions; that is, the system simply ends at the edges and the spins at the boundaries have fewer neighbors than those in the interior. The difference in behavior is largest near T_c .

8.5 FIRST-ORDER PHASE TRANSITIONS

In Section 8.4 we studied the behavior of an Ising model near its critical point. This transition from the ferromagnetic phase with $M \neq 0$, to the paramagnetic phase, where $M = 0$, is an example of a second-order phase transition.³⁴ This raises the obvious question: What is a first-order phase transition? First-order transitions are actually very common in nature, the freezing of water being a typical example. We can observe a first-order transition with our Ising model if we include the effect of a magnetic field. The Monte Carlo method can be used as described above, the only difference being that the energy for flipping a spin must include the energy gained or lost to the field as given in (8.5).

We now have two independent variables in the problem, T and H , so there is a larger phase diagram to explore. Let us first consider the behavior as a function of field at fixed temperature as shown in Figures 8.11 and 8.12. At $T = 1.0$ and with $H < 0$, the magnetization is large and negative. We already know that at this temperature the spins will be nearly fully aligned even without the field. Here the field serves only to determine the direction of M (since the field energy tends to align the spins with H), and this is why M changes sign abruptly when H is increased through zero. This *discontinuous* change of M is an indicator of a

³⁴The terms *continuous* and *higher order* are also commonly used

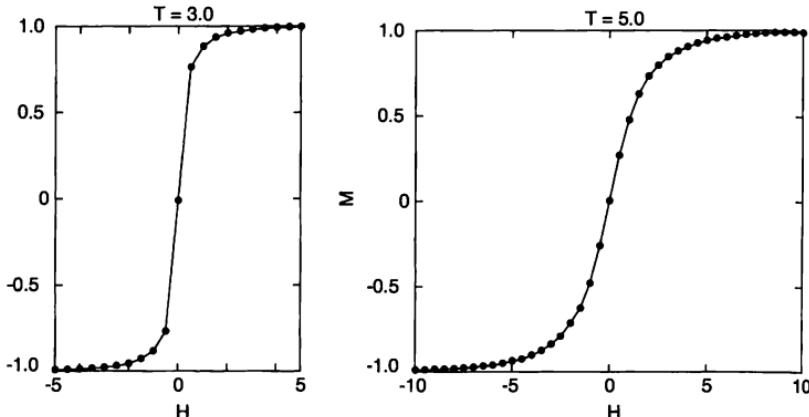


FIGURE 8.12: Continuation of Figure 8.11

*first-order transition.*³⁵ This discontinuity in the magnetization involves the two states of the system that are related by a simultaneous reversal of all of the spins, $M \rightarrow -M$. When $H = 0$ these two states are equally probable, but the application of a small field will make one more likely than the other. A discontinuous jump in M as a function of field is found at all temperatures below T_c . From energy and symmetry considerations we would expect this jump to occur at $H = 0$, but it sometimes happens that there is *hysteresis* associated with the transition. This can be seen from the results at $T = 1.0$, where the jump occurred at a value of H which was slightly but distinctly greater than zero. Here the system was trapped in a *metastable* state ($M < 0$ with $H > 0$) over the time period of the simulation. We will have more to say about this in a moment.³⁶

Above T_c the spontaneous magnetization vanishes, so there can be no discontinuity in M as the field is swept through zero, and hence no first-order phase transition. At temperatures above T_c you can go smoothly from the state with a negative magnetization which is found when $H < 0$, to the state with a positive

³⁵This nomenclature was introduced by Ehrenfest who proposed the name *first-order* for a transition in which a first derivative of the free energy (such as M) is discontinuous. Likewise, according to the Ehrenfest scheme a second-order transition is one in which a second derivative of the free energy, such as the specific heat, is discontinuous. Originally, Ehrenfest had in mind *finite* discontinuities, as found for the magnetization in Fig. 8.11. However, today, this notion is extended to all forms of discontinuities (and singularities), and the term *second-order* is used for transitions at which a second derivative of the free energy is divergent, as we found in Section 8.4.

³⁶This is our first encounter with a metastable state. Such behavior is commonly found in connection with first-order transitions. We will also observe it in a rather different system in Chapter 12.

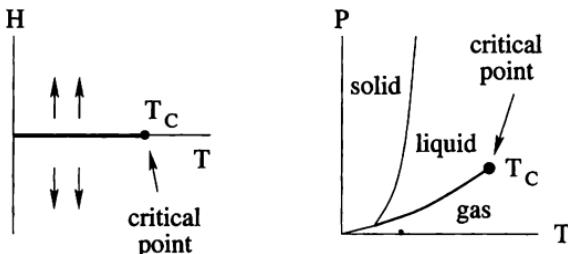


FIGURE 8.13: Left: phase diagram of a ferromagnet in the H - T plane. The heavy line that lies along the temperature axis denotes a first-order transition between states $\pm M$. This line ends at the critical point, $T = T_c$. Right: schematic pressure-temperature phase diagram of a typical pure substance which can be in a liquid, gas, or solid phase. The lines are first-order transitions that separate the various phases. The first-order line that separates the gas and liquid phases is called the vapor-pressure curve, and it terminates at a critical point. The point where all three lines of first-order transitions meet is a triple point.

magnetization with $H > 0$. The magnitude of the discontinuity found below T_c is just twice the spontaneous magnetization we calculated in the previous section. We found that this magnetization vanishes at T_c , so there is a very close connection between the first-order transition observed as a function of field and the second-order transition observed as a function of temperature. This connection can be appreciated by considering the *phase diagram* as viewed in the H - T plane.

At low temperatures the system possesses two distinct phases corresponding to $\pm M$, as denoted by the arrows in Figure 8.13. We can pass from one phase to the other by crossing the temperature axis, and as we have seen in Figure 8.11, M varies discontinuously when we do this at temperatures below T_c . This is the location of the first-order transition and yields a line³⁷ in the H - T plane as shown in Figure 8.13. This line of first-order transitions ends at the critical temperature, where the spontaneous magnetization vanishes. At this temperature the difference between the two phases disappears. At and above T_c we can pass from positive to negative fields without any discontinuity in M .

From the phase diagram we see that the line of first-order transitions terminates at a critical point. This geometry is a general feature of first-order transitions. For example, in a liquid-gas system there is a very similar situation. In that case the relevant variables are pressure (replacing H), temperature, and density (replacing M), as shown schematically in Figure 8.13. The transition from a liquid to a gas is first-order over a range of pressures, with a discontinuity in the density. As the pressure is increased, the magnitude of this discontinuity becomes smaller, and it vanishes at the critical point. As in a spin system, there are singularities in

³⁷This line is an example of a *coexistence curve* where two different phases (here one phase has $M > 0$, while the other has $M < 0$) coexist

various properties of a liquid-gas system at T_c , and these singularities are described by power laws with critical exponents. Interestingly, the values of the critical exponents of a liquid-gas system are believed to be the *same* as those found for the Ising model.³⁸ This universality of the behavior suggests that the essential features of the critical behavior transcend the specific model or system. This is a very interesting subject that we encourage you to pursue through the references.

Returning to Figure 8.13, an interesting feature of the Ising model phase diagram is that you can move from the “up” phase to the “down” phase in two very different ways. One is to cross the temperature axis and thereby experience a first-order transition. The other is to go *around* the critical point at high temperatures and thus avoid the first-order line altogether. Of course, these two options are also available in a liquid-gas system, where you can either pass through the first-order transition line, or go from liquid to gas without a transition by navigating around the critical point.

Finally, we should emphasize an important difference between first and second-order transitions. Near the critical point the fluctuations become very large in anticipation of the singular behavior that is found there. That is, the system “knows” that something important is about to happen. However, a first-order transition occurs abruptly. There are no enhanced fluctuations or any other sort of “warning” that discontinuities are imminent.

This lack of warning is connected with the fact that the spin configurations before and after the transition are very different. At the low temperatures considered here, these two states are ones in which the spins are nearly all parallel to each other, with M either “up” or “down.” If the system is initially in one of these states, then in order for it to undergo a transition to the other state requires that essentially *all* of the spins be flipped. The Monte Carlo flipping rules involve one spin at a time, and since at low temperatures even a single spin flip that raises the energy occurs only very rarely, the probability that a large number of spins will be able to conspire to flip together is extremely small.

This is illustrated in Figure 8.14 which shows results for M as a function of H at low temperatures. At each temperature we have started at a large negative field so that the spins were essentially all aligned in the negative direction. The field was then increased in steps. We would expect that this spin configuration would be the stable one until the field becomes positive, at which point the state with all spins pointing in the positive direction should be the thermodynamically stable state. However, we see in Figure 8.14 that at $T = 0.5$ the system remains in the negative state until $H \sim 1.5$ before it switches to the positive state. Similarly, if we start at a large positive field, and then decrease the field in steps, we see that the system does not switch from the positive to the negative state until $H \sim -1.5$. Hence, the state of the system depends on the past history of the system, an effect known as hysteresis. This delay in switching into the thermodynamically stable state is a result of the extremely low probability for the system to make the transition. Put another way, we have here a case in which the Monte Carlo procedure did

³⁸Note, however, that the exponent values do depend on dimensionality, so the exponents for two- and three-dimensional Ising models are different.

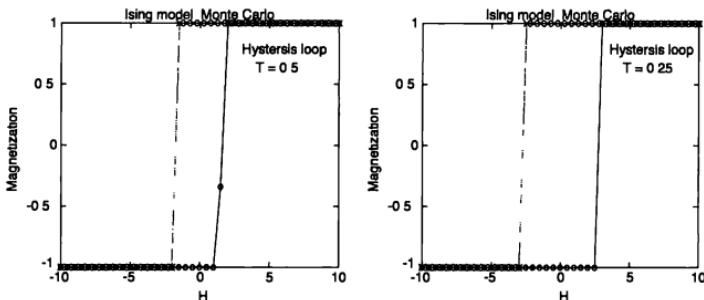


FIGURE 8.14: Hysteresis loops for a 10×10 Ising model calculated at temperatures well below T_c . The open circles (and solid lines) show results obtained by starting at a large negative field and increasing the field in steps, while the crosses (and dotted lines) were obtained by starting at a large positive field and decreasing the field in steps. One thousand Monte Carlo time steps (that is, one thousand complete sweeps throughout the lattice) were spent at each field

not reach equilibrium during the time scale of the simulation. Rather, the system became stuck in a nonequilibrium state. If we had waited longer at each field, this effect would be reduced and the amount of hysteresis would be smaller. However, such hysteretic behavior can often be found even if we wait for macroscopically long times. A good example of this is the behavior of a liquid near its freezing transition. In general, it is possible to cool a liquid well below the (thermodynamic) freezing temperature before it undergoes the transition to the solid phase.

It is instructive to compare the hysteresis at different temperatures. We see from Figure 8.14 that the amount of hysteresis becomes larger as the temperature is reduced. This is because the probability for a Monte Carlo spin flip varies as $\exp(-E_{\text{flip}}/k_B T)$, and hence becomes smaller as T is made smaller. This increases the likelihood that the system will become stuck in a metastable state. We will discuss metastability and related matters further in Chapter 12 when we consider the problems of protein folding and neural networks.

EXERCISES

- 8.11. Above T_c there is no net magnetization when $H = 0$, but applying a field induces alignment of the spins as can be seen for example at $T = 5$ in Figure 8.12. At very high temperatures the interactions between spins have very little effect, and the field dependence is then described by approximately (8.8). Confirm this by calculating $M(H)$ at $T = 100$ (use the value $J = 1$, as we have employed throughout this chapter) and compare your results with (8.8). You should find good, though not exact, agreement. The deviations are due to the interactions between spins. Repeat your calculation at lower temperatures ($T = 30$ and 10 are good choices), and show that the deviations from (8.8), that is, the effects of the interactions, become larger as T is reduced
- *8.12. Calculate M as a function of field at T_c . The behavior should be described by

the power law

$$M \sim H^{1/\delta}, \quad (8.29)$$

where δ is another critical exponent. Try to estimate the value of δ . In two dimensions $\delta = 15$, while in three dimensions its value is close to 5. Hint: It is difficult to get a good estimate for δ because its relatively large value (especially in two dimensions) means that M increases extremely rapidly in small fields. To overcome this it is useful to employ lattices that are larger than our standard 10×10 size. A 20×20 lattice and a field range of 0.02–0.2 are good choices, but you should try lattices of other sizes and explore the behavior for a wider field range. You may also find it necessary to average over 3000 or more Monte Carlo time steps at each field. Once you have obtained reliable results for M , you can then construct plots of M^{δ^*} as function of H for various trial values of δ^* . The value that gives the best straight line as $H \rightarrow 0$ then provides an estimate for δ . Alternatively, the slope of a plot of $\log M$ as a function of $\log H$ will also yield an estimate for δ .

- 8.13.** Study how the hysteresis in M as a function of H at low temperatures varies as a function of the amount of time the system is given to come into equilibrium. Use a 10×10 lattice at $T = 0.25$ and calculate the hysteresis loops (as in Figure 8.14) by stepping the field in small increments ($\Delta H = 0.5$ is a good choice), and averaging over a given number of Monte Carlo time steps at each field. Observe how the amount of hysteresis depends on the amount of Monte Carlo time spent at each field.
- *8.14.** Investigate how the magnitude of the hysteresis depends on how the boundaries of the lattice are treated. Consider a 10×10 lattice at low temperatures and compare the behavior with periodic and free boundary conditions (by “free” we mean that the system simply terminates, and spins at the edges have only three near neighbors, while those at the corners have only two). You should find that there is less hysteresis in the case of free-boundary conditions. Give a qualitative argument to explain why this is so.

8.6 SCALING

In discussing critical exponents in the previous sections (as well as in Chapter 7), we made reference to the universality of their values and to the concept of *scaling*. In this section we consider these ideas a little further. We begin with the mean field equation (8.11). This result was obtained for $H = 0$ using the mean field approximation. In order to do the same but for $H \neq 0$, we add the actual external field H to the effective field (H_{eff}) in (8.10) and get

$$\langle s \rangle = \tanh [(zJ\langle s \rangle + \mu H)/k_B T]. \quad (8.30)$$

For small values of $\langle s \rangle$ and H in the neighborhood of the critical point, we can expand the righthand side of (8.30) and obtain an extension of (8.13) for $H \neq 0$,

$$\langle s \rangle \approx \frac{zJ\langle s \rangle}{k_B T} - \frac{1}{3} \left(\frac{zJ\langle s \rangle}{k_B T} \right)^3 + \frac{\mu H}{k_B T}, \quad (8.31)$$

where we keep terms up to $O(\langle s \rangle^3)$ and $O(H)$.³⁹ Rearranging this equation, we can write it as

$$h = b t m + u m^3, \quad (8.32)$$

where the dimensionless variables are given by $h \equiv \mu H/J$, $m \equiv \langle s \rangle$, and $t \equiv 1 - zJ/k_B T = (T - T_c)/T_c$, while b and u are positive constants.⁴⁰ Equation (8.32) is an *equation of state* as it gives the order parameter m in terms of the two independent thermodynamic variables t and h (or T and H), although this relation is implicit.

Focusing on the various powers with which the quantities m , t , and h appear in (8.32), we notice that⁴¹

$$m(\lambda^{1/2}t, \lambda^{3/4}h) = \lambda^{1/4}m(t, h), \quad (8.33)$$

for any $\lambda > 0$. To understand the meaning of (8.33), consider how the different factors change if we choose $\lambda = 16$. If t is multiplied by $\lambda^{1/2} = \sqrt{16} = 4$ and h is multiplied by $\lambda^{3/4} = 16^{3/4} = 8$, then (8.32) is still valid as long as m is also multiplied by $\lambda^{1/4} = 16^{1/4} = 2$. Functions of two variables which satisfy an equation of this form have the remarkable property that they can be expressed as a function of a *single* variable. To see this, we choose $\lambda = |t|^{-2}$ in (8.33) and rewrite it as

$$m(t, h) = |t|^{1/2}m\left(\pm 1, \frac{h}{|t|^{3/2}}\right) = |t|^{1/2}f_{\pm}\left(\frac{h}{|t|^{3/2}}\right), \quad (8.34)$$

where the \pm signs correspond to $t > 0$ and $t < 0$, respectively. Equation (8.34) states that when *scaled* by the factor $|t|^{1/2}$, $m(t, h)$ is a function of a single, composite variable $h/|t|^{3/2}$.

We have derived the *scaling form* (8.34) for the magnetization using mean field theory. However, we have seen that mean field theory does not provide an accurate quantitative description of behavior near a critical point. In particular, the critical exponents predicted by mean field theory are not correct. Fortunately, it turns out that the general form of scaling expressed in (8.34) is correct.⁴² For the magnetization m this general scaling form is

$$m(t, h) = |t|^{\beta}f_{\pm}\left(\frac{h}{|t|^{\beta\delta}}\right) \quad (8.35)$$

As we see from (8.34), $\beta = 1/2$ and $\delta = 3$ for mean field theory, but to describe a real ferromagnet we must use the correct values of the corresponding critical exponents. The scaling property of thermodynamic quantities near the critical point such as (8.35) provides a connection among the various power law behaviors.

³⁹This is justified since $H \sim \langle s \rangle^3$ near the critical point in mean field theory. That is, the exponent δ in (8.29) is equal to 3 in mean field theory.

⁴⁰We remind the reader that T_c here refers to the mean field critical temperature.

⁴¹This is an example of a property called *generalized homogeneity*. It is a generalization of the ordinary homogeneity of a function where the powers of λ in front of t and h are identical. In *generalized homogeneity*, these powers can be different.

⁴²In fact, other thermodynamic functions such as specific heat, susceptibility, and the free energy exhibit similar scaling properties. Moreover, the scaling functions (e.g., f_{\pm} for $m(t, h)$ in (8.34)) have *universal* functional forms, independent of many details of the system.

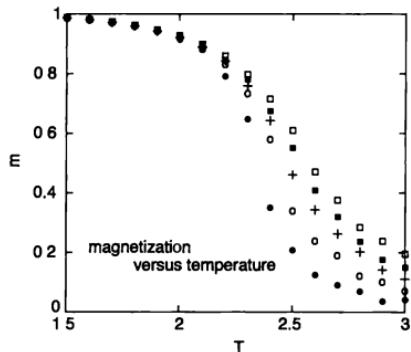


FIGURE 8.15: Magnetization $m(T, H)$ for a 20×20 square lattice as a function of temperature T , for different values of the magnetic field. Various symbols correspond to different values of H , from $H = 0.01$ for the filled circles to $H = 0.05$ for the open squares

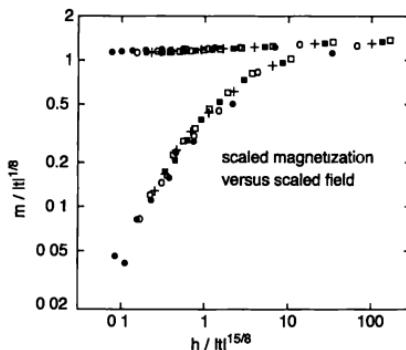


FIGURE 8.16: Scaling of the magnetization $m(t, h)$ is shown by plotting $m / |t|^{\beta/\delta}$ against $h / |t|^{15/8}$ using the same data as in Fig. 8.15. The upper branch consists of data from $t < 0$ (i.e., $T < T_c$), while the lower branch consists of those from $t > 0$ (i.e., $T > T_c$). They correspond to the '-' and '+' branches in (8.35)

This fact is now understood in terms of the *renormalization group* theory of critical phenomena.⁴³

We can test how scaling works for our Ising model on the square lattice, where the correct exponent values are $\beta = 1/8$ and $\delta = 15$. To do this, we simulate the Ising model on a 20×20 square lattice with the temperature T and field H in the vicinity of the critical point, where $T = T_c \approx 2.27$ and $H = H_c = 0$. Here we use a slightly larger lattice than employed in previous sections, since scaling is only valid in a near-critical system and the critical behavior is approximated better in larger systems.⁴⁴ Results for the magnetization as a function of T at various values of H are shown Fig. 8.15. The magnetization is a smooth function of T for any fixed $H \neq 0$, but falls more rapidly near T_c as H is decreased. For each value of H , the values of $m(T, H)$ produce a different curve only approaching a common limit at low temperatures. However, when we plot $m/|t|^\beta$ against the *scaling variable* $h/|t|^{1/\delta}$ as in Fig. 8.16, all of these different curves collapse into the two branches of the scaling function, just as predicted by (8.35). Such behavior is often referred to as *data collapsing*. It occurs around all second-order phase transitions, including the percolation transition studied in Chapter 7.

EXERCISES

- 8.15.** Scaling behavior is found for thermodynamic quantities other than the magnetization. Calculate the susceptibility χ at various values of T and H around the critical point of the Ising model on a square lattice, and study data collapsing using your results. The scaling form for χ is

$$\chi(t, h) = |t|^{-\gamma} g_{\pm} \left(\frac{h}{|t|^{1/\delta}} \right), \quad (8.36)$$

where the critical exponent $\gamma = 7/4$.

- 8.16.** Figure 8.16 shows data collapsing for the magnetization of an Ising model on a square lattice. Try the same calculation on the triangular lattice where $T_c = 4/\ln 3 \approx 3.64$. Plot the scaling function on a log-log scale and compare your results with those shown in Figure 8.16. Are the functions the same or different for these two lattices? Comment on your results in terms of the notion of *universality*.
- *8.17.** In an exercise involving percolation in Chapter 7, we introduced a quantity χ that is a measure of the average cluster size below p_c . It was defined by

$$\chi = \sum_s s^2 N_s / L^2, \quad (8.37)$$

where N_s is the number of clusters of s connected sites. We mentioned that $\chi \sim |p - p_c|^{-\gamma}$ as p approaches p_c from below, and used the same symbol γ to

⁴³You can read more about the renormalization group theory in, e.g., Fisher (1988).

⁴⁴We remind you that the critical behavior, as well as any other type of phase transition, is strictly speaking, only possible in an infinite system. We are merely approximating it when we simulate a finite-size system. Also, the exact value of the critical temperature ($T_c \approx 2.27$ for a square lattice) may not necessarily be a good value to use to fit critical power laws such as $M(T, H) \sim |T - T_c|^\beta$ with data from a finite lattice, though it becomes better and better for larger lattices. Using as large a lattice as possible helps to avoid these complications.

describe the power law as we used to describe the critical power law of the Ising susceptibility. We use the same symbol because the percolation mean cluster size is the analog of the Ising susceptibility. In this problem we consider data collapsing of the mean cluster size using the scaling form of the Ising model susceptibility (8.36).

In percolation, the site occupation probability p plays the role of temperature T in the Ising model, but what plays the role of the external magnetic field H ? The field H in the Ising model couples uniformly to every spin and tries to align them, making the order parameter M larger. Accordingly, we can imagine a *ghost site* in percolation which is connected to every site of the lattice and is occupied with probability h . This probability h turns out to be the analog of H ; $h > 0$ increases the connectivity through every occupied site uniformly, thus increasing the spanning probability, which is the order parameter for percolation. With this addition, the mean cluster size becomes

$$\chi(p, h) = \sum_s (1 - h)^s s^2 N_s / L^2, \quad (8.38)$$

where the factor $(1 - h)^s$ is the probability that a particular cluster of s sites is *not* connected to the ghost site (and thus is *not* a part of an infinite cluster, i.e., it is a finite cluster) and N_s is obtained from standard simulations *without* h . This quantity should then scale as

$$\chi(p, h) = |p - p_c|^{-\gamma} g_{\pm} \left(\frac{h}{|t|^{\beta\delta}} \right) \quad (8.39)$$

The values of the exponents for two-dimensional percolation are $\gamma \approx 2.4$, $\beta \approx 0.14$, and $\delta \approx 18$. Investigate this form of scaling using Monte Carlo data for percolation for $p < p_c$ [The same can also be done for $p > p_c$ if the spanning cluster is excluded from the sum in (8.38).]

REFERENCES

- [1] K. Binder and D. W. Heermann, 1992, *Monte Carlo Simulation in Statistical Physics*, Springer-Verlag, New York. A careful theoretical discussion of the Monte Carlo method.
- [2] M. E. Fisher, "Critical Phenomena in Films and Surfaces," J. Vac. Sci. Technol. **10**, 665 (1973). A very readable discussion of finite size effects near critical points.
- [3] M. E. Fisher, "Renormalization group theory: Its basis and formulation in statistical physics," Rev. Mod. Phys. **70**, 653 (1988). A mostly qualitative and very nice review of renormalization group theory.
- [4] N. Goldenfeld, 1992, *Lectures on Phase Transitions and the Renormalization Group*, in Frontiers in Physics series #85, Addison Wesley.
- [5] D. W. Heermann, 1990. *Computer Simulation Methods in Theoretical Physics*, 2d ed. New York: Springer-Verlag. Discusses how the microcanonical ensemble can be simulated using a Monte Carlo approach

- [6] D. P. Landau and R. Alben, "Monte Carlo Calculations as an Aid in Teaching Statistical Mechanics." Am. J. Phys. **41**, 394 (1973). A nice tutorial introduction to the Monte Carlo method.
- [7] M. Plischke and B. Bergersen, 1994, *Equilibrium Statistical Physics*, 2nd Ed., Prentice Hall, Upper Saddle River.
- [8] W. H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. An excellent all-purpose reference on numerical methods and why they work
- [9] F. Reif, 1965, *Fundamentals of Statistical and Thermal Physics*, McGraw-Hill, New York. Reviews statistical mechanics, including a discussion of the Maxwell-Boltzmann distributions.
- [10] H. E. Stanley, 1971, *Introduction to Phase Transitions and Critical Phenomena*, Clarendon Press, Oxford. A standard text on phase transitions and critical phenomena. Includes treatments of mean field theory and the Ising model.

Molecular Dynamics

9.1 INTRODUCTION TO THE METHOD: PROPERTIES OF A DILUTE GAS

In this chapter we consider a mechanical approach for studying multiparticle systems. The method we describe is known as molecular dynamics, and it is in many respects complementary to the Monte Carlo method employed in Chapter 8. The Monte Carlo algorithm is very convenient for studying the *equilibrium* properties of a system that is in contact with a heat bath. However, there are many questions that cannot be addressed with such an approach. For example, suppose we are interested in how fast a system will come into equilibrium after a sudden change in the temperature. With the Monte Carlo method the dynamics is governed entirely by the Monte Carlo transition rules (for an Ising model these are the spin-flipping rules), together with the appropriate Boltzmann factors. While these rules are designed to simulate a system in thermal equilibrium, they cannot tell us how fast in real time that a system will move from one particular microstate to another, or if such a transition is even possible.

One way to deal with such questions is to directly simulate the dynamics using the microscopic equations of motion, and this is the philosophy of the molecular dynamics technique. The basic idea is similar to the cream-in-your-coffee problem of Chapter 7, but here we are going to use the appropriate equations of motion to treat the full many-body problem,¹ rather than simulate the motion of an “average” particle. We imagine a box containing a collection of molecules. These molecules move throughout the box as they collide with each other and with the walls of the box. To simulate this process we employ Newton’s second law to calculate the positions and velocities of all of the molecules as functions of time. The kinds of questions that can be addressed with this approach include the nature of the melting transition, the rate of equilibration after a sudden addition or loss of energy, and the rate at which molecules diffuse.² You may recall that we spent the first part of Chapter 7 arguing that such a microscopic approach provides results that are much too detailed for many of the questions we might like to address, and that claim still stands. However, some questions *do* require such a full-blown microscopic treatment, we will now describe how to carry out such a treatment, along with some of the problems for which it is needed.

While the general simulation scheme we are going to consider in this chapter could apply to any system containing a large number of particles, including droplets in an aerosol, particles in a flame, or stars in a galaxy, molecular dynamics is, as its name implies, usually employed to study the motion of molecules. We might

¹Albeit for far fewer than 10^{23} particles!

²We will now have a way to calculate the value of the diffusion constant that we have encountered in several previous problems

therefore worry that a *classical* approach involving Newton's second law would not be appropriate, and that we should instead aim for a fully quantum mechanical simulation. A quantum simulation would be *much* more time consuming and computationally difficult than a classical simulation, so it is fortunate that a classical treatment is justifiable for many situations of interest.³ That this is, in fact, the case can be seen from the following arguments. Let us consider the simulation of a collection of atoms, such as argon. All of the electrons associated with each argon atom are bound fairly tightly to their respective nuclei; the energy required⁴ to promote an electron to an excited state, or to remove it entirely from an argon atom, is of order 10 electron volts (eV). This energy is much larger than the typical kinetic energy associated with the center of mass motion of an atom, which is of order 0.1 eV at room temperature. This large energy difference means that collisions between argon atoms will not involve enough energy to have an effect on the electron configuration of either atom. In particular, there is not nearly enough energy available to strip away an electron. For this reason it is a very good approximation to treat each atom as a simple structureless particle. In addition, the DeBroglie wavelength of an argon atom at room temperature is of order 10^{-7} Å. The average spacing between atoms in a solid is of order 1 Å, and we will see shortly that in typical liquids and gasses the atoms never get closer than ~ 1 Å during a collision. Hence, the atomic wavelength is much smaller than the particle separation, which again justifies a classical approach. We can therefore use Newton's second law to calculate the positions of the atoms as a function of time.

It is useful to begin by considering order of magnitude estimates of some of the quantities involved in molecular dynamics. First, the mean number of collisions that a given particle (i.e., molecule) suffers per unit time is related to the particle density n , the mean relative velocity of the particles v_{rel} , and the so-called scattering cross section σ_0 (essentially the target area). The approximate relation is

$$f \approx nv_{rel}\sigma_0. \quad (9.1)$$

It is not surprising that the collision rate (f) is proportional to these quantities as it should increase when n , v_{rel} , or σ_0 increases. The mean length that a particle travels between collisions is called the mean free path ℓ , and this is of order v_{rel}/f , since $1/f$ is the mean time between collisions. Now, if we replace the density n by $1/L^d$ where L is the mean particle separation in d dimensions, and σ_0 by a^{d-1} where a is the linear dimension of a particle, we get

$$\ell \approx \left(\frac{L}{a}\right)^d a = \left(\frac{L}{a}\right)^{d-1} L. \quad (9.2)$$

For gasses and liquids we have $L/a \gg 1$, giving $\ell \gg L \gg a$. It also means that the mean free path is much larger in three dimensions than in two dimensions, for

³There are, however, situations where quantum mechanical treatment is necessary and appropriate. An example would be where the chemical bonds between atoms undergo changes. Quantum mechanical molecular dynamics approaches are usually called *ab initio* calculations since the molecular dynamics time evolution is blended with on-the-fly solutions of the Schrödinger equation. You can read more about it in, e.g., Marx and Hutter in the references.

⁴You may recall that the binding energy of an electron in the ground state of a hydrogen atom is ~ 13.6 eV.

comparable L and a , hence requiring more molecular dynamics time steps to simulate a scattering event in higher dimensions. In addition, the number of particles within a given distance from a given particle is much larger in three dimensions than in two if L is comparable. Therefore, on both counts, molecular simulations in three dimensions will be much more costly in computing resources than in two dimensions. We can estimate crudely how much more. Assume that $L/a = 10$, and we take the mean molecular displacement during a time step to be on the order of a itself. In this case, $\ell \approx 10^3$ steps in three dimensions versus 10^2 in two. Suppose further that the calculation involves 10^3 particles in the volume of side length $10L$ in three dimensions, as compared to 10^2 particles in an area of the same side length in two dimensions. Then, taken together, the three dimensional computation will take 100 times longer in this particular scenario than in the two dimensional case. For these reasons, as well as for simpler presentation and coding, we will consider atoms moving in a two-dimensional plane in this chapter.⁵

For each atom i we then have the equations of motion

$$\begin{aligned}\frac{dv_{i,x}}{dt} &= a_{i,x}, \\ \frac{dx_i}{dt} &= v_{i,x}, \\ \frac{dv_{i,y}}{dt} &= a_{i,y}, \\ \frac{dy_i}{dt} &= v_{i,y},\end{aligned}\tag{9.3}$$

where $v_{i,x}$ and $v_{i,y}$ are the components of the velocity of the i th atom, which is located at position (x_i, y_i) . The components of the acceleration of each particle, $a_{i,x}$ and $a_{i,y}$, are determined by the forces from all of the other particles in the system. To solve these equations numerically we have several choices. In nearly all of the problems we have encountered so far in this book, either the Euler or Euler-Cromer method has been adequate. However, in molecular dynamics we will be interested in computing the motion over a very large number of time steps, and it turns out that the numerical errors associated with Euler type methods are too big to tolerate. It is therefore necessary to use a slightly more complicated scheme for solving the differential equations arising from Newton's second law. The scheme we will employ here is known as the Verlet method and is introduced in Appendix A. As usual, we discretize time in steps Δt . Letting $x_i(n)$, $v_{i,x}(n)$, and $a_{i,x}(n)$ be the x components of position, velocity, and acceleration of particle i at time-step n , their values at the next time step are given according to the Verlet method as

$$\begin{aligned}x_i(n+1) &\approx 2x_i(n) - x_i(n-1) + a_{i,x}(n)(\Delta t)^2, \\ v_{i,x}(n) &\approx \frac{x_i(n+1) - x_i(n-1)}{2\Delta t},\end{aligned}\tag{9.4}$$

⁵Although an extension of the algorithm to three dimensions is straightforward.

with similar equations for y_i and $v_{i,y}$. These equations are derived in Appendix A, where it is also shown that the numerical errors associated with the Verlet method are much smaller than with the Euler method (for a comparable amount of computing time).

At first glance, (9.4) doesn't look much like our usual Euler or Euler-Cromer expressions. However, the origin of this relation for $x_i(n+1)$ can be appreciated if we recall the finite-difference approximation for a second derivative

$$\frac{d^2x_i}{dt^2} \approx \frac{x_i(n+1) + x_i(n-1) - 2x_i(n)}{(\Delta t)^2}, \quad (9.5)$$

which we have encountered on several previous occasions. If we use the fact that d^2x_i/dt^2 is the acceleration and rearrange (9.5) to solve for $x_i(n+1)$, we obtain precisely (9.4) (a more careful derivation is given in Appendix A). An important feature of the Verlet method is that it conserves energy very well over the course of many time steps, which is crucial if we want to perform an accurate molecular dynamics simulation. It is interesting to note that the position can be calculated directly from the acceleration; we don't really have to calculate the velocity as an intermediate step, as is necessary with the Euler method. However, we will see that the velocity contains some very useful information, so we will always compute it along with the position.⁶

A molecular-dynamics simulation consists of solving the Verlet equations for every particle in the system. A key quantity required in this calculation is the acceleration. Each particle experiences a force from all of the other particles; this is what gives rise to collisions between the particles. To estimate the force between any two particles requires knowledge of the interaction potential, $V(r)$, where r is the separation of the particles. The calculation of $V(r)$ does involve quantum mechanics; it depends on what kinds of atoms are involved and the nature of the forces between them. For elements such as argon the situation is relatively simple. For large separations the interaction is due to the Van der Waals force, which is a weak attraction arising from the transient electric dipole moments of the two atoms. We don't have space to give a derivation here;⁷ all we really need to know is that this potential varies as r^{-6} and is attractive. When the atoms get close together there is also a repulsive force due to the overlap of their electron clouds. The precise form of this force is hard to calculate, since it involves many electrons. Common practice is to approximate it by a term that varies as r^{-12} and is repulsive. Adding this to the Van der Waals component yields what is known as the *Lennard-Jones* potential

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^6 \right], \quad (9.6)$$

where ϵ and σ are constants that set the energy and distance scales associated with the interaction. This function is plotted in Figure 9.1; the associated force is the derivative $F = -\partial V/\partial r$, and is directed along the line connecting the atoms. We

⁶However, the errors involved in the Verlet's method calculation of the velocity are of the same order as in the Euler method, see Appendix A

⁷See any introductory quantum mechanics text, such as Schiff (1963)

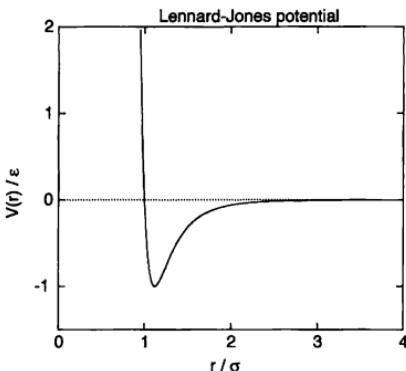


FIGURE 9.1: Lennard-Jones potential

see from Figure 9.1 that the two atoms experience a significant attractive force when the separation is in the range $\sim (1.1\text{--}2.0)\sigma$. For separations larger than about 3σ the force is essentially zero, while for $r \leq 1.1\sigma$ the force is very strongly repulsive.

We will follow standard molecular-dynamics practice and use the Lennard-Jones potential in our calculations.⁸ Its magnitude is set by ϵ , so it is convenient to work in units for which $\epsilon = 1$. Thus, we effectively measure all energies in terms of ϵ , which for argon is $\epsilon/k_B = 120. Argon is a popular choice for molecular-dynamics simulations, since the Ar–Ar interaction is very well described by the Lennard-Jones potential. Likewise, it is convenient to set the length scale $\sigma = 1$, so that all lengths are measured in units of σ , which has a value of 3.4 \AA for argon. These are often referred to in the literature as *reduced units* and are denoted by E^* and r^* , etc.; we will usually just use the symbols E and r for convenience. Finally, standard practice also sets the mass of an atom to unity, so that all masses are measured in terms of the mass of one argon atom. Since energy has units of mass times velocity squared, this leads us to measure time in units of $\tau \equiv \sqrt{m\sigma^2/\epsilon}$ and velocity in units of $\sqrt{\epsilon/m}$, where m is the mass of an argon atom.⁹ One (reduced) unit of time for argon is $\approx 1.8 \times 10^{-12}\text{ s}$, about 2 picoseconds.$

Before we discuss the programming there is one more issue that must be addressed, namely the boundary conditions. Perhaps the most obvious way to enclose

⁸Other potentials are sometimes used, for example, to study systems such as metals in which conduction electrons contribute significantly to the interaction

⁹Thus, a dimensionless velocity of the order of 1 (i.e., of order $\sim (\sqrt{\epsilon/m})$) corresponds to the kinetic energy of order of ϵ . This makes these units very convenient and reasonable to use in simulations

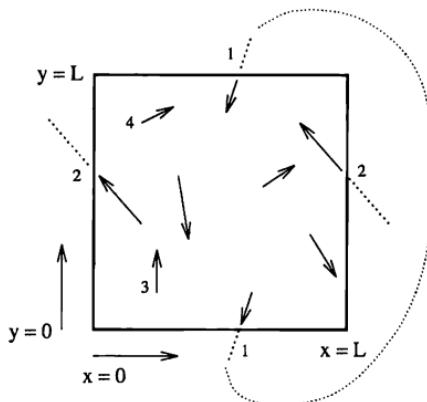


FIGURE 9.2: Periodic boundary conditions for a molecular dynamics simulation using an $L \times L$ box. The arrows denote atoms and their velocities.

the particles is in a box¹⁰ with hard, perfectly reflecting walls. The problem with this approach is that for presently available computers we are limited to systems containing a relatively small number of atoms. In small systems, the collisions with the walls can be a significant fraction of the total number of collisions, in contrast to a real system where the behavior would be dominated by collisions with other particles. Moreover, the *shape* of a small container can greatly affect how the particles are spatially arranged, which can be a serious problem if we want to study a condensed phase (a liquid or a solid). For these reasons it is common practice to use *periodic boundary conditions*. We have already encountered such boundary conditions in our Monte Carlo simulations of the Ising model, but the situation there was a little simpler since the spins did not move. Here periodic boundary conditions work as illustrated schematically in Figure 9.2, which shows a collection of atoms in a square, two-dimensional box with walls at $x = 0, L$ and $y = 0, L$. Whenever an atom encounters a wall it is transported instantly to the *opposite* side of the system. For example, the atom marked 1, which reached the wall at $y = 0$, was transported to the opposite side at $y \approx L$. Note that *only* the y coordinate was affected by this move. The x coordinate and both components of the velocity were unaffected.

One way to view periodic boundary conditions is to imagine that the box is situated on a torus. Two particles that are near opposite faces in a plot such as Figure 9.2 are then actually very close to each other. This avoids all collisions with walls (since there are no walls!), but poses a slight complication when we calculate

¹⁰We need some sort of enclosure to keep the particle from traveling away and never coming back.

the force between two particles, since this force depends on their separation. For example, particles 3 and 4 in Figure 9.2 are separated by a distance of $\approx 3L/4$ along the y direction. However, if we take advantage of the teleportation of periodic boundary conditions, their separation is only $\approx L/4$. In order for our equations of motion to be consistent, particles should only be allowed to interact once, so we must use the smaller of these two separations to calculate both the magnitude and the *direction* of the force.

We have now described the basic conceptual details associated with molecular dynamics and next consider how to construct a program. Here we will describe only its main components. After setting the size of the box, the number of particles, and the time step, we must pick the initial positions and velocities. These will depend on the particular problem we want to study. As an example, let us consider a fairly dilute gas. Your intuitive picture of a gas probably has the atoms arranged randomly within the box, so it is natural to choose the initial positions randomly. However, this intuitive picture does not take into account the hard-core repulsion of the potential. Even in a gas, the atoms are *not* arranged completely at random. If there is an atom at a particular location (x_1, y_1) , the probability of finding a second atom within a distance σ of this position is essentially zero. To take this into account in our initialization it is convenient to give the atoms an approximately regular arrangement so as to guarantee that no two atoms are too close to each other. In the following code example, we do this by first placing the atoms on a square lattice in which the spacing between nearest neighbors is greater than 2σ and then displacing the atoms from these locations at random by a distance $\leq \sigma/2$. This gives a somewhat random initial arrangement and keeps the atoms well separated from each other. The choice of initial velocities is not as complicated. One possibility is to give each particle a speed v_0 in a randomly chosen direction and another is to allow both the direction and the magnitude of the velocity to be random within certain limits. We used the latter approach in the example sketched below. We will consider other possibilities shortly.

EXAMPLE 9.1 Initializing a molecular dynamics simulation

- Choose the number of particles (N), and the size of the system (L).
- Set computational parameters, including Δt (the time step).
- Set initial positions and velocities of the particles:
 - ▷ Set max position component deviation δr from the vertices of a regular array.
 - ▷ Set max initial velocity component magnitude v_0 .
 - ▷ Iterate through N particles ($i = 1, 2, \dots, N$):
 - Calculate the equidistant grid point coordinates $[g_x(i), g_y(i)]$ for particle i .
 - Displace the particle randomly a bit by setting $x_{curr}(i) = g_x(i) + 2(\text{rnd} - 0.5)\delta r$, $y_{curr}(i) = g_y(i) + 2(\text{rnd} - 0.5)\delta r$.
 - Calculate a randomized initial velocity by $v_{x0}(i) = 2(\text{rnd} - 0.5)v_0$,

$$v_{y0}(i) = 2(\text{rnd} - 0.5)v_0.$$

- To set the stage for Verlet method calculations, define the (fictitious) position *previous* to this initial time by $x_{\text{prev}}(i) = x_0(i) - v_{x0}(i)\Delta t$, $y_{\text{prev}}(i) = y_0(i) - v_{y0}(i)\Delta t$.
-

After initializing the particle positions and velocities, the Verlet method can be used to calculate the subsequent motion of each atom. The acceleration components of particle j are $a_{j,x}$ and $a_{j,y}$, and these are obtained from the Lennard-Jones potential. The components of the acceleration for particle j are obtained by simply summing the individual forces from all of the other particles in the system

$$\begin{aligned} a_{j,x} &= \frac{1}{m} \sum_{k \neq j} f_{k,j} \cos \theta_{k,j}, \\ a_{j,y} &= \frac{1}{m} \sum_{k \neq j} f_{k,j} \sin \theta_{k,j}, \end{aligned} \quad (9.7)$$

where $f_{k,j}$ is the force of particle k on particle j (note that in our reduced units $m = 1$), and $\theta_{k,j}$ is the angle that the line between them makes with the x axis. These sums exclude terms like $f_{j,j}$, since a particle does not interact with itself. The pair forces are given by

$$f_{k,j} = -\frac{\partial V}{\partial r_{k,j}} = 24 \left(\frac{2}{r_{k,j}^{13}} - \frac{1}{r_{k,j}^7} \right),$$

where $r_{k,j}$ is the separation between particles k and j , and we have assumed $\sigma = \epsilon = 1$ so that we are using reduced units as mentioned above. Note that $r_{k,j}$ and the associated angle must be measured with the “minimum” separation rule of periodic boundary conditions. For example, particles 3 and 4 in Figure 9.2 are a distance $\approx L/4$ apart (not $3L/4$). The angular factors in (9.7) can be estimated from the relative positions of the two particles. If their separation along x is Δx , then $\cos \theta_{k,j} = \Delta x / r_{k,j}$, etc. for $\sin \theta_{k,j}$.

At each time step it is necessary to compute the acceleration of every particle. Since each particle interacts with every other particle, this involves the calculation of *many* pair forces $f_{k,j}$. In practice this is the most time-consuming part of the calculation, so it is worth making this part of the program efficient. To this end we first recall from Figure 9.1 that the interaction potential is essentially zero for $r > 3$ (remember that this distance is measured in reduced units, so $r = 1$ corresponds to a real separation of σ). We will therefore take the force to be exactly zero when $r > 3$. This is known as “cutting off” the potential, and speeds things up since we can avoid calculating $f_{k,j}$ for most pairs of particles.¹¹ Another way to speed

¹¹However, our program does calculate $r_{k,j}$ for every pair so that it can decide whether or not to evaluate $f_{k,j}$. We could go a step further and not even bother to calculate $r_{k,j}$ for particles that were very widely separated at the previous time step. This approach will speed things up even more, but the programming is a bit more complicated since we must decide how often to check on the value of $r_{k,j}$.

things up is to note that $|f_{k,j}| = |f_{j,k}|$, which is required from Newton's third law. It is most efficient to calculate all of the $f_{k,j}$ at one time (with two nested loops), being careful that each pair of particles is considered only once.

After calculating the $f_{k,j}$ as just described, our program estimates the new position of particle j using (9.4). At the same time the new velocities are also calculated. Note that with (9.4) we use a symmetric form of the derivative to find the velocities, so when we calculate the positions at step $n+1$ we then obtain the velocities at step n . This will be important¹² when we consider the total energy, since we will want to calculate the kinetic and potential energies at the *same* value of t . After obtaining the new positions we must then check to see if any of the particles have "left" the box. If so, our program uses the periodic boundary condition rules to teleport the particle to the opposite side of the system. To be consistent, we also teleport the previous value of the position since this will be needed in the calculation of the velocity at the next time step. The velocity does not need any adjustment.

We sketch below a subroutine update that updates the positions and velocities of the particles implementing these ideas.

EXAMPLE 9.2 Subroutine update

- Iterate through N particles ($i = 1, 2, \dots, N$) and calculate the new position and velocity for each one.
 - ▷ For particle i , iterate through all other particles, performing the following tasks:
 - Calculate the distance r_{ij} to particle j (taking into account the periodic boundary condition).
 - If $r_{ij} > r_{cut}$ (the cutoff length), skip j
 - Otherwise, calculate the force f_{ij} on particle i due to j . Add this force vector to the net force $f(i)$ on i from all the other particles.
 - ▷ Use the net force $f(i)$ to update the position and velocity of particle i :
 - $x_{\text{new}}(i) = 2x_{\text{curr}}(i) - x_{\text{prev}}(i) + f(i)_x(\Delta t)^2$,
 $y_{\text{new}}(i) = 2y_{\text{curr}}(i) - y_{\text{prev}}(i) + f(i)_y(\Delta t)^2$ where the subscripts curr and prev refer to the *current* and *previous* time steps.
 - If the new position falls outside of the system, apply periodic boundary condition and bring it back inside
 - $v_x(i) = (x_{\text{new}}(i) - x_{\text{prev}}(i))/(2\Delta t)$, $v_y(i) = (y_{\text{new}}(i) - y_{\text{prev}}(i))/(2\Delta t)$
- Again iterate through N particles and this time:
 - ▷ Relabel the current positions as *previous*.
 - ▷ Relabel the newly calculated positions to *current*

¹²There are several different variations on the Verlet algorithm, as discussed in Heermann (1990), and some of them avoid this problem. Watch out for this when comparing our program with those of other authors.

- ▷ Calculate and store any quantity you wish to later use, such as the total kinetic and potential energies.
-

Note that we must calculate the forces acting on all the particles in their current positions first, and then update all of positions at one time; we cannot update a particle's position before calculating the forces on other particles.

This completes one time step of the calculation. The above procedure is then repeated for as many time steps as desired. The results can be monitored and analyzed in several ways, which we will now describe as we consider specific simulations. Figure 9.3 shows the results for 20 particles in a 10×10 box. It is usually not necessary to (permanently) record or display the positions after each time step, since the particles should move little during an interval Δt ; otherwise Δt was not chosen small enough in the first place! Here we have plotted the positions sufficiently often that the individual trajectories can be followed without confusing one particle for another. Many pair "collisions" can be noted, although the dots plotted here never actually "touch" each other, since the hard-core repulsion of the Lennard-Jones potential effectively prevents the particles from getting closer than about σ , which is unity in our reduced units. We also see several cases of particles exiting one side of the box and reappearing at the opposite side. A much better feeling for the behavior is obtained from observing the motion in "real time" as the calculation proceeds, but unfortunately we cannot include a movie here. This is also a very good way to find programming errors.

The results in Figure 9.3 show that we can indeed compute the motion of a collection of particles. It remains for us to demonstrate that our simulations have anything in common with a real system. Perhaps the most fundamental issue is whether our system reaches a proper equilibrium state, and if so, how fast it comes into equilibrium. This also raises the question of how such an equilibrium state is described in terms of statistical mechanics. Since the only forces in our problem are those between particles, the total energy must be conserved, corresponding to the microcanonical ensemble of statistical mechanics. There is no external heat bath as we had in our Monte Carlo simulations. Even so, the concept of temperature can still be useful. One way to view this is to imagine a restricted "system" consisting of a small fraction of the particles, with the remaining particles then acting as a heat bath. Thus, we can still use the concept of "temperature," but unlike the case with the Monte Carlo method developed in the previous chapter, the value of T is not an explicitly given parameter. Rather, we will have to calculate T from the behavior of the system. This can be accomplished using the equipartition theorem, which states that for a classical system the average energy of each "quadratic" degree of freedom¹³ is $k_B T/2$. This theorem can be used in association with the velocity components of each particle, since the kinetic energy associated with v_x is $mv_x^2/2$, etc., for v_y . The assertion is that our molecular dynamics simulation will describe a system whose temperature can be computed using the equipartition theorem.

To justify this claim we next consider if and how the system in Figure 9.3

¹³That is, each coordinate or velocity that contributes a quadratic term to the energy.

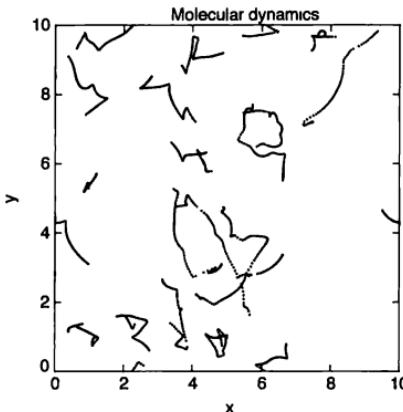


FIGURE 9.3: Trajectories of 20 particles in a 10×10 box with periodic boundary conditions, with initial speeds $v = 1$ in randomly chosen directions. The time step was 0.02 and the positions of the particles were plotted after every third time step.

reaches an equilibrium state. We will focus on the behavior of the particle velocities, since we know that in equilibrium a classical gas is described by the Maxwell distribution. For a two-dimensional gas we have the *speed* distribution

$$P(v) = C \frac{v^2}{k_B T} \exp(-mv^2/2k_B T), \quad (9.9)$$

where $P(v)$ is the probability per unit v of finding a particle with speed v , and C is a constant that depends on the mass of the particle. The corresponding *velocity* distribution is

$$P(v_x) = \frac{C_x}{(k_B T)^{1/2}} \exp(-mv_x^2/2k_B T), \quad (9.10)$$

with a similar expression for $P(v_y)$. If the molecular dynamics method describes the behavior of a real gas, it should yield velocity and speed distributions that have the Maxwell forms.

In Figure 9.4 we show the speed distribution of the gas in Figure 9.3 calculated by averaging over several different time intervals as the simulation proceeded. Initially all of the particles were given a velocity whose magnitude was unity ($v = 1$ in reduced units) and whose direction was random, yielding a speed distribution as shown by the vertical bar at $v = 1$ in Figure 9.4. We then let the simulation run, and after every 10 time steps the speed distribution was recorded by dividing the v range into bins and tabulating the number of atoms whose speed was in the

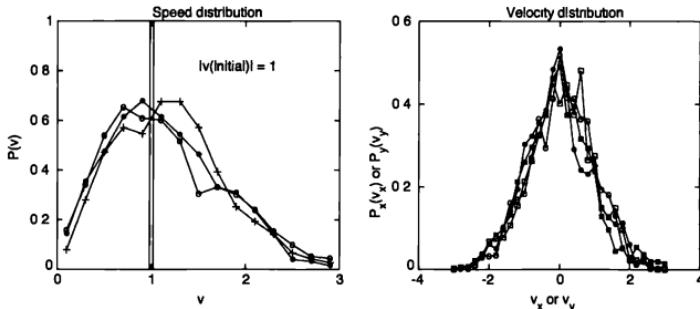


FIGURE 9.4: Left speed distribution of the gas of particles in Figure 9.3 at various times. At $t = 0$ the distribution was the vertical bar at $v = 1$ (which is not drawn to scale). The other symbols correspond to averages over the reduced time intervals $t = 0\text{--}20$ (solid circles), $t = 20\text{--}40$ (open circles), and $t = 40\text{--}60$ (pluses). Right velocity distributions obtained in a similar simulation, but using a different initialization of v_x and v_y . These velocity distributions were calculated by averaging over the intervals $t = 0\text{--}20$ and $20\text{--}40$. Aside from the statistical uncertainties, these results agree with the Maxwell distribution (9.10).

range corresponding to each bin. These histograms¹⁴ were averaged over the time intervals $t = 0\text{--}20$, $20\text{--}40$, and $40\text{--}60$, with the results shown in Figure 9.4. We will leave it to the exercises to demonstrate in detail that in all three cases the speed distribution has a form that (to within the statistical fluctuations) is well described by (9.9). The conclusion is that our system did indeed come into thermal equilibrium. Moreover, once reached, this equilibrium distribution was *Maintained* at future times, since apart from the statistical fluctuations due to the relatively small number of particles in the system, the distribution remained unchanged as the simulation continued.

The results in Figure 9.4 show that our system reaches the expected equilibrium distribution when started from a particular initial state. It is also interesting to consider the behavior starting from *different* initial states. We know that a real system can be started in different initial states but still reach the *same* final equilibrium state, assuming of course that the temperature, pressure, etc., are kept the same.¹⁵ The same is true for our molecular dynamics model. This can be demonstrated using the speed distribution, as we will explore in the exercises. Here we consider this question using the *velocity* distributions $P_x(v_x)$ and $P_y(v_y)$. While these are closely related to the speed distributions just considered, they contain additional information relating to the direction of the motion. While our results for the speed distribution indicated that the magnitudes of the velocities were distributed properly for a system in thermal equilibrium, the velocity distributions can tell us if the directions are also given correctly.

¹⁴Note that these distributions have been normalized so that $\int P(v)dv = 1$, etc.

¹⁵Indeed, this is one of the features that makes the concept of equilibrium so useful in the first place.

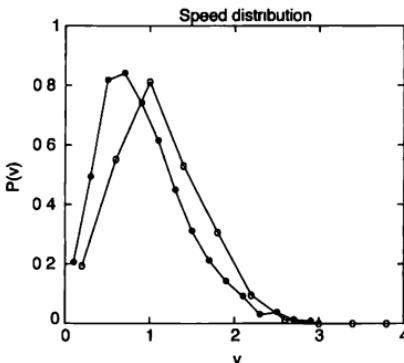


FIGURE 9.5: Speed distributions Filled circles obtained after equilibration from a state with initial velocities v_x and v_y chosen randomly in the range ± 1 . Open circles, same, but with v_x and v_y chosen randomly in the range ± 2 . The system thus had more energy in this case, and the peak in the speed distribution comes at a correspondingly larger value of v . These speed distributions were obtained by averaging over the interval $t = 0 - 20$

We chose an initial state in which half the particles were given an initial x component of (reduced) velocity chosen randomly in the range $-1 \leq v_x \leq +1$ with $v_y = 0$, and the others were given $v_x = 0$ with $-1 \leq v_y \leq +1$. This is certainly a very peculiar situation, which is nowhere near a Maxwell distribution (9.10), so if the system manages to come into equilibrium starting from this state, we would expect that the same would be found for (most) other initial states (we'll consider some exceptional situations in the exercises). We see from the results on the right in Figure 9.4 that our gas quickly reached the expected distributions for both $P_x(v_x)$ and $P_y(v_y)$; we will leave it to the reader to verify that these distributions do indeed have the Maxwell form (9.10). This provides us with more evidence (and greater confidence) that molecular dynamics provides a way to simulate a mechanical system in thermal equilibrium.

An important quantity for a system in thermal equilibrium is the temperature. As we have already mentioned, T does not enter the simulations as an input parameter. It must instead be “measured,” and the results for the speed and velocity distributions give us a way to make this measurement. To demonstrate this we show in Figure 9.5 the results for the speed distributions obtained from two separate simulations. In one case the initial velocity components were both chosen randomly in the range ± 1 , while in the second this range was ± 2 . Hence, on average, the atoms in the latter case had greater speeds and were, therefore, “hotter.” This can be seen from the fact that the speed at which the peak occurs in $P(v)$ is shifted to higher values of v in the system that had the larger initial velocities. Comparing this with the form of the Maxwell distribution, we can see that the

location of this peak is determined directly by the temperature. We could thus use the peak location in conjunction with (9.9) to measure T .

Another approach is to make use of the equipartition theorem. As mentioned above, this theorem states that for a classical system an average energy of $k_B T/2$ is associated with any degree of freedom that enters the energy quadratically. The kinetic energy of an atom in our two-dimensional system is $m(v_x^2 + v_y^2)/2$, so the equipartition theorem tells us that for a system in thermal equilibrium¹⁶

$$k_B T = \left\langle \frac{m}{2} (v_x^2 + v_y^2) \right\rangle, \quad (9.11)$$

where the (reduced) temperature is measured in units of ϵ/k_B , which is ≈ 120 K for argon. Here the angular brackets can be interpreted in two ways. According to one point of view, (9.11) applies to each atom in the system, so we can obtain T by computing the *time* average of the kinetic energy of one particular atom. However, since the atoms are all equivalent, the same result can be obtained by averaging (at a particular instant in time) over the different atoms in the system.¹⁷ In practice, the best computational accuracy will be obtained by combining the two points of view and computing the time average of the kinetic energy per atom averaged over all of the atoms. We will leave it to the exercises to compare the temperature calculated from (9.11) with the value obtained by fitting the Maxwell distribution functions directly to the results in Figures 9.4 and 9.5.

EXERCISES

- 9.1. Calculate the speed distributions for a dilute gas as in Figure 9.4 and compare the results quantitatively with the Maxwell distribution. (For example, perform the χ^2 analysis described in Appendix G.) This analysis also yields the temperature; compare the value you find with the result calculated directly from the equipartition theorem (9.11).
- 9.2. Calculate the speed distributions starting from different initial states and show that if the initial kinetic energy is the same and the particles are widely separated so that the potential energy is also the same, the equilibrium distributions are the same. *Hint:* Choose the initial velocities so that the average values of v_x and v_y are both zero. See the next problem for a discussion of why this is important.
- *9.3. Repeat the previous problem, but now consider how (and if) certain “peculiar” initial states approach equilibrium. Consider a dilute gas in a 10×10 box containing 20 particles. Give all of the particles an initial v_x which is positive; you might, for example, choose v_x randomly in the range 0–1. Show that the distribution $P(v_x)$ never assumes the form (9.10), as the average value of v_x will always be positive rather than zero. Explain why this follows from conservation laws. *Hint:* Consider the conservation of momentum.

¹⁶Note that this assumes that the overall translational kinetic energy of the system as a whole is negligible. For a molecular-dynamics simulation this simply means that the center of mass velocity must be much smaller than that of a typical atom. Otherwise a system in which the atoms were all at rest relative to each other would, if the center of mass velocity were large, have a high temperature, at least according to (9.11). However, we know that this cannot be the case.

¹⁷This should remind you of the ergodic hypothesis.

- 9.4.** Study the diffusion of particles in a dilute system. For example, take 16 particles in a 10×10 box and calculate the mean-square displacement of an atom as a function of time. Show that the motion is indeed diffusive [that $(\Delta r)^2 \approx D t$], and find the value of the diffusion constant. You can also study how D varies with density. The diffusion of an atom in a system containing a large number of like atoms is known as self-diffusion. *Hint:* Be sure to allow for the teleportation associated with the periodic boundary conditions when you calculate Δr .
- 9.5.** The first molecular-dynamics simulations (see Alder and Wainwright [1959]) treated a system of hard disks. In this case the potential is zero when the separation exceeds the disk diameter and infinite for smaller separations, and collisions are assumed to be elastic. Perform a simulation for this case and compare the velocity distribution with the Maxwell form. This may remind you of the billiard problem of Chapter 3.
- *9.6.** The Lennard-Jones potential (9.6) describes a pairwise interaction between atoms and is a function only of their separation. Explain why this potential would *not* be useful for modeling a system in which the bonding between atoms is covalent. Discuss what new features the potential must have in this case. *Hint:* Consider the role of bond angles.
- *9.7.** Investigate the approach to equilibrium of a system containing only a few particles. We have already seen in connection with Figure 9.4 that a gas containing 20 particles will come into equilibrium, as observed using the velocity distribution. However, as an extreme case we know that this will not happen for a gas containing only a single particle, since in this case v_x and v_y will never change (assuming periodic boundary conditions). Perform simulations like those in Figure 9.4 for a system containing 2 particles. Does this system come into equilibrium in the sense of (9.9) and (9.10)? Explain why it does not. *Hint:* Consider how momentum is exchanged in each collision. Repeat the calculation with 3, 4, ... particles and explain your results.
- 9.8.** Write a program that uses hard-wall boundary conditions. That is, when a particle hits a wall it should be specularly reflected, as in the billiard problem of Chapter 3. Show that a gas of 20 particles in a 10×10 box reaches an equilibrium state such as that shown in Figure 9.4.
- *9.9.** The behavior of the total energy of a gas as a function of temperature can reveal the importance of interactions between the atoms. This can be appreciated by calculating the energy as a function of T for several different densities (a time step of 0.01 is a good choice for all of the simulations below).
 - (a) Begin by calculating E , the sum of the kinetic and potential energies of all of the particles, for a system containing 16 particles in a box of size 20×20 . Show that to a reasonably good approximation E varies linearly with T , with $E \rightarrow 0$ as $T \rightarrow 0$. Explain why this should be expected for a very dilute gas. *Hint:* When the particles are far apart the potential energy will be negligible, so E is then approximately equal to the kinetic energy. What does the equipartition theorem then tell you about $E(T)$?
 - (b) Increase the density by confining the same number of particles to a 5×5 box. You should now observe that E does not vanish as $T \rightarrow 0$. Explain how this is a result of the interactions between particles. This is an example of a “nonideal” gas.
- 9.10.** In order to fully characterize a gas it is useful to measure (or calculate) the equation of state. For this you need to know the pressure. This can be calculated in a molecular-dynamics simulation in the following way. If the simulation employs

hard-wall boundary conditions, the pressure on a wall of the container will be the force per unit area exerted by the particles that are reflected by the wall. This force can be calculated from the momentum change that the wall imparts on each particle it reflects. A simulation that uses periodic boundary conditions will not have any walls or reflections, but the same quantity can be obtained by considering the particles that “pass through” a particular boundary as part of the teleportation process associated with periodic boundary conditions. Every time a particle tries to pass through the surface at $x = +L$ (we assume an $L \times L$ box), it is transported via the periodic boundary conditions to $x = -L$. This particle carries an amount of momentum mv_x in the x direction. If there had been a hard wall at this location, the particle would have been reflected ($v_x \rightarrow -v_x$), which would have imparted momentum $2mv_x$ to the wall (since momentum is conserved in a collision with the wall). The force on the wall is the momentum per unit time that is transferred to it by all collisions, and the pressure is the force per unit area.¹⁸

Use this approach to calculate the pressure of a dilute gas as a function of temperature. Good parameter choices are 16 particles in a 10×10 box with $\Delta t = 0.02$. Calculate P and show that it varies linearly with T . Explain why this is so. Hint: Consider the equation of state for an ideal gas.

9.2 THE MELTING TRANSITION

In Section 9.1 we introduced the technique of molecular dynamics and used it to investigate several properties of a dilute gas and how it approaches equilibrium. In this section we will use the same method to investigate the melting transition. Melting is a phenomenon in which the interactions between particles play a crucial role. The phases involved in melting, the liquid and solid phases, are direct results of these interactions. Hence, in order to provide a quantitative description of melting, a method that treats interparticle interactions in a realistic manner is essential. Molecular dynamics is an ideal choice.

We have already mentioned that melting is a first-order phase transition, so we expect to find an abrupt change in the system when it melts. One of our tasks will be to devise useful measures of “liquidness” and “solidness.” This will turn out to be a little more difficult than you might suspect and will force us to think carefully about what is meant by the terms *liquid* and *solid*.

Let us first establish that our molecular-dynamics approach yields a system that is a solid under the appropriate conditions, namely low temperatures and high

¹⁸Another common way to calculate the pressure and the equation of state involves the application of so-called virial theorem. The virial theorem equation of state for pair-wise interacting particles is discussed in standard statistical mechanics texts such as Chandler and Pathria in the references. In the context of molecular dynamics, it is useful to write the virial theorem in the form of

$$PV = Nk_B T + (1/6) \left\langle \sum_{i \neq j}^N \mathbf{r}_{ij} \cdot \mathbf{F}_{ij} \right\rangle,$$

where the summation is over all pairs of particles and \mathbf{r}_{ij} is the displacement vector and \mathbf{F}_{ij} is the force respectively between particles i and j . This method of calculating the pressure P is sometimes preferred since it involves all the particles in the system rather than relying on the few particles that strike a surface as in the method described above.

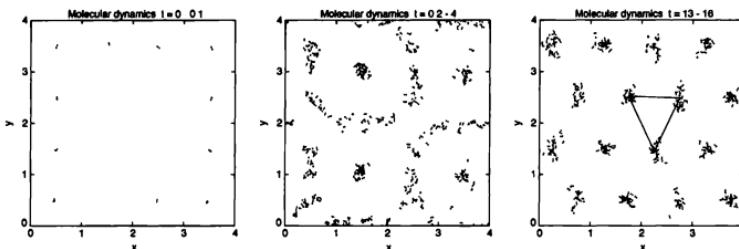


FIGURE 9.6: Snapshots of a system of 16 particles in a 4×4 box at a low temperature. The time step in the simulation was 0.005 in reduced units. These are “time-lapsed” pictures, taken over the time intervals indicated in each figure. The position was recorded after every 10 time steps. On the right we have drawn lines connecting several nearest neighbor atoms to make the triangular structure clear.

density. In order to make T as low as possible, we will start with all of the particles at rest. This does not mean that they will remain at rest, since each will move in response to forces from neighboring particles. Nevertheless, this will give us a low initial temperature. The density is also important, since we expect that for low densities, that is, for a large average particle separation, the system will be a gas. Examining the Lennard-Jones potential in Figure 9.1 we note that the maximum attraction occurs for an interparticle spacing of approximately $1.2\sigma = 1.2$ in reduced units. Hence, we are led to choose a density of approximately 1 particle for each (reduced) unit of area.

Some results for such a dense system are shown in Figure 9.6. Here we consider 16 particles in a 4×4 box, with the particles initially arranged on a square lattice. However, while the particles were at first all at rest, this arrangement was not stable and they immediately began to move; Figure 9.6 shows “time-lapsed” snapshots of the system during various time intervals. The plot on the left shows snapshots taken over the first few time steps, and it is seen that the particles have moved only a little from their initial positions, as the square lattice is still apparent. However, when given some time the particles move substantial amounts, as shown in the plot in the center, which shows a superposition of snapshots taken during the period $t = 0.2-4$. Eventually an equilibrium state is reached, which is shown in the picture on the far right in Figure 9.6. While the particles are still in motion here, each moves in a region that is only $\approx \pm 0.2$ units in size. This is a crystalline solid.¹⁹

¹⁹We need to choose our words carefully here, since the stability of a solid in two dimensions is a rather tricky issue. It turns out that in two dimensions, an infinitely large solid would actually be unstable. That is, an infinitely large two-dimensional crystal would not be the thermodynamically stable phase at any nonzero temperature. This has been established through exact analytic arguments, as described by Mermin (1968) (see also Nelson [1983]). Nevertheless, it appears that finite systems, such as those we study in this section, can be in a state that for nearly all practical purposes is a crystalline solid. The melting transition of an infinitely large two-dimensional system is believed to differ in some subtle ways from that of a three-dimensional solid, as discussed in

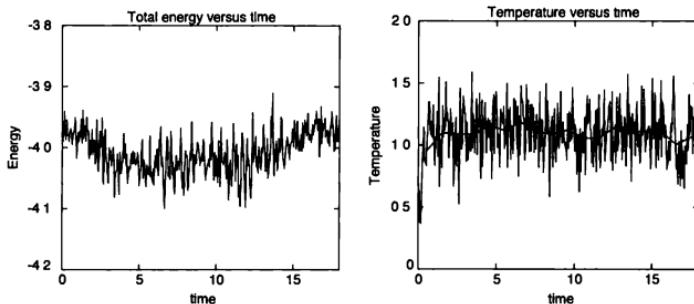


FIGURE 9.7: Energy and temperature as functions of time for the simulation in Figure 9.6. The solid slowly varying line on the right shows the temperature as averaged over intervals of $\Delta t = 1$.

However, we can see that the arrangement of the atoms now is *not* a square lattice (as set up originally). Rather, we have a *triangular* lattice. It turns out that this structure minimizes the energy²⁰ for particles interacting through the Lennard-Jones potential.

Given the rather small size of our system we might worry that boundary effects could influence the crystal structure. Even though we have employed periodic boundary conditions, this does *not* eliminate the effects of the boundaries. These boundary conditions still impose a “square” shape on the system, since the periodicity conditions have that geometry. The fact that we have obtained a structure that is far from square implies that a triangular structure has a significantly lower energy than a square lattice and that the free energy barrier that separates the two phases is also not too large (this should remind you of our discussion of hysteresis in connection with the Ising model in Chapter 8). Thus, it is probably safe to conclude that in this case a triangular lattice is the most stable solid structure for the Lennard-Jones potential. It is possible to treat the periodic boundary conditions in a more general way, in which the *shape* of the effective box is allowed to vary. Such an approach is more complicated, but is required if we want to determine the most stable solid without any bias imposed by the simulation.

We mentioned earlier that the Verlet method was chosen because it conserves the energy fairly well over the course of many time steps. This claim is justified in Figure 9.7, which shows both the total energy (kinetic plus potential) and the temperature for the simulation that yielded the snapshots in Figure 9.6. Since

the references. We will ignore these complications here, as they do not affect the points we wish to make regarding the qualitative aspects of melting. To within the accuracy and resolution of our simulations it is safe to assume, as we will below, that our two dimensional system is an ordinary solid at low temperatures and that the melting transition in two dimensions is an ordinary first-order transition.

²⁰Actually it minimizes the free energy, but at the low temperature considered here this is nearly equal to the energy.

there are no external sources of energy, E should be conserved exactly. We see that the energy remained approximately constant (the vertical scale here is greatly expanded), with fluctuations of about ± 2 percent due to numerical errors associated with the Verlet method. These errors could be made smaller by using a smaller time step, but this would require more computer time. The time step used here is suitable for our purposes, but a research calculation would probably strive to keep variations in E below 1 percent.

The plot on the right in Figure 9.7 shows the variation of the temperature for the same simulation. As in the previous section, we have calculated T using the equipartition theorem, (9.11). It is important to realize that this relation holds for any classical system, even one in which the interactions between particles are strong, as they are here. While the fluctuations in the energy are fairly small, the corresponding fluctuations in the temperature are much larger. They are so large, in fact, that we might worry that the concept of temperature may not be useful (or appropriate). However, the difficulty here is not with the concept of temperature, but in our calculation of the averages needed to evaluate (9.11). To estimate T we need to compute the averages of v_x^2 and v_y^2 for all of the particles. In the simulation here the number of particles was fairly small (only 16), so we shouldn't expect these averages to have a high precision. However, this problem can be circumvented by performing an additional time average of the values of T in Figure 9.7, an example of which is shown by the solid line on the right in Figure 9.7. Here we have averaged the values of T over time intervals of size $\Delta t = 1$, and the fluctuations are now much reduced. The message here is that when evaluating the averages that arise in statistical physics, the fact that there is a small number of particles must be kept in mind. Actually, while this is a concern for small-scale simulations like those we have used here to illustrate the molecular-dynamics algorithm, it would usually not be a serious problem in a research calculation, as these typically use many thousands of particles or more.

Now that we have obtained a system that is a solid, we must devise a method to heat it in order to observe melting. This is usually accomplished by increasing the kinetic energy "by hand." That is, we increase the velocities of all of the particles by a factor that is greater than unity. This gives them all some additional kinetic energy and through (9.11) will increase the temperature. After increasing the velocities we must then give the system a chance to come into equilibrium. The additional energy is injected as purely kinetic energy, but we know that as the system comes back into equilibrium this energy will be redistributed among the kinetic and potential energies of the particles.

As a programming note, we must be careful when rescaling the velocities since it is the positions that enter²¹ the equations of motion (9.4). With the Verlet algorithm we have the position at both the current and the previous time step. A convenient way to rescale the kinetic energies is to adjust the location at the previous time step in the following way. Let \vec{r}_c and \vec{r}_p be the current and previous positions [$\vec{r}_c = (x_c, y_c)$, etc.]. If we want to increase the velocity by a factor of 2,

²¹While the velocities can be calculated at each time step, for the form of the Verlet method that we have described they do not enter into determining the position at the next time step

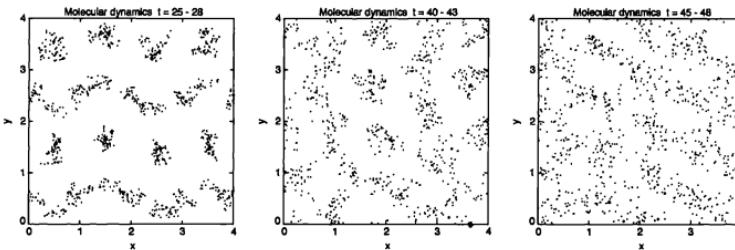


FIGURE 9.8: Time-lapse snapshots at several temperatures as the system considered in Figure 9.6 was heated in stages. The time intervals are given at the top of each plot, and the corresponding energy and temperature are shown in Figure 9.9.

we adjust \vec{r}_p so as to make it twice as far from \vec{r}_c . In general, to rescale the velocity by an amount R we take

$$\vec{r}_p \rightarrow \vec{r}_c - R(\vec{r}_c - \vec{r}_p) . \quad (9.13)$$

This is not the only way that we could rescale the velocities, but it does have one important property: it does not alter the current positions. If we had, instead, chosen to adjust \vec{r}_c , we would affect the current potential energy, as well. However, as long as the time step is small, either approach would be suitable. While we will usually use this rescaling procedure to increase the temperature, it can also be used to decrease the temperature, as would be needed to study freezing or condensation.

In Figure 9.8 we show snapshots of our system as it was heated in stages. The corresponding energy and temperature as functions of time are shown in Figure 9.9, where the abrupt steplike increases in E show the times when the kinetic energy was increased. At each of these points the velocities were increased by a factor of 1.5, and as expected the temperature increases along with E . The fluctuations in T also increased dramatically as the temperature was raised. In our discussion of the Monte Carlo method in Chapter 8, we saw that such fluctuations are related to quantities such as the specific heat (we will leave the exploration of this issue to the industrious reader).

Our goal in this section is to observe the melting transition, and it is time for us to finally tackle that problem. By examining the snapshots of the system at various temperatures in Figure 9.8, we can clearly see by eye that the system became increasingly disordered as the temperature was raised. However, it is not obvious (at least to the authors) how to locate the melting transition from these snapshots alone. In a sense, the information in the snapshots is too “microscopic.” Most of the properties that we associate with a solid are based not on the positions of individual atoms, but on thermodynamic types of variables. Unfortunately, thermodynamic variables are often the ones that are most difficult to determine accurately with molecular dynamics.

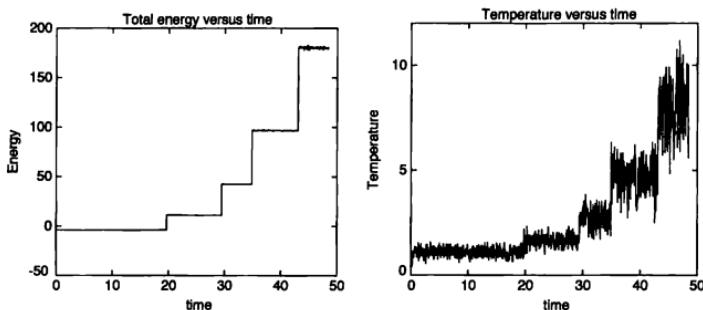


FIGURE 9.9: Energy and temperature as functions of time for the simulation in Figures 9.6 and 9.8. Each abrupt increase in E was produced by rescaling the velocities by a factor of 1.5.

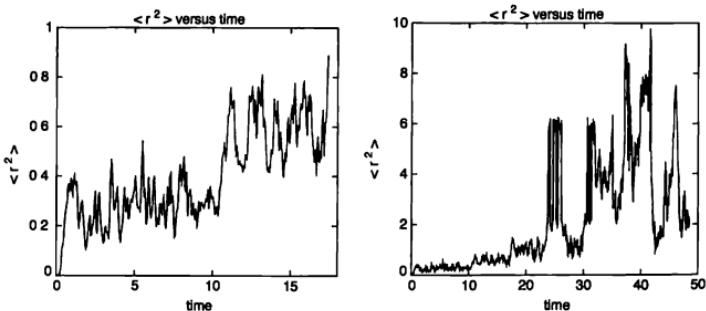


FIGURE 9.10: Motion of a "test" particle in the simulation of Figure 9.8. Left: behavior at early times corresponding to a temperature $T \sim 1.1$; right: variation of $\langle r^2 \rangle$ as the temperature was increased in stages as shown on the right in Figure 9.9. Note that the vertical scale is very different in the two plots.

What we really need is a quantitative measure of solidness and liquidness; preferably this measure should be closely connected with the motion of individual atoms, since that is the type of information we have at our disposal. It turns out that there are several such measures that could be employed. One is to consider in detail the motion of a particular particle, which might be termed a "test particle" even though it is equivalent to all of the others in the system. To illustrate the approach, we have chosen one particle in Figure 9.8 and recorded its position as a function of time. The square of the displacement from its initial position is shown in Figure 9.10. These values were obtained from the simulation considered in Figure 9.9, so the corresponding temperatures can be read from those results. At the lowest temperature (corresponding to the earliest times) the particle quickly settled into a position a small distance away from its initial position, as the structure changed from the initial square form to a triangular lattice. Our test particle then remained nearly stationary up to $t \approx 11$. While the particle did move a little during this interval, the mean-square displacement was a small fraction of the spacing between atoms (which was ≈ 1). There was an abrupt shift in the particle's position at $t \approx 11$, but the overall displacement was still much less than the average spacing between particles and was probably due to motion of the entire lattice.²² Overall, the behavior for $t < 11$ is that expected for a particle in a solid.

As the temperature was increased, the fluctuations of the particle's position became larger. During the period $t \approx 23\text{--}26$ it shifted back and forth between two fairly well-defined locations, before returning (approximately) to its original location at $t \approx 28$. It thus appears that the particle was moving between several different lattice sites, or perhaps the entire lattice was shifting in space. With either interpretation, while our test particle was certainly more mobile than it was at lower temperatures, the system was probably still a solid, since there appear to be fairly discrete "lattice positions" available to the particle. However, after the temperature increase at $t \approx 30$, the behavior changed qualitatively, as the displacement fluctuated rapidly over distances much greater than a lattice spacing and did not tend to prefer any discrete values. This suggests that the system had become a liquid.²³ These fluctuations increased further in magnitude as the system was heated again at $t \approx 35$ and 42. From the corresponding results for the temperature as a function of time in Figure 9.9 we conclude that the melting transition took place at a temperature somewhere in the range $T \sim 1.5\text{--}2.0$.

There are several other measures that we can use to discern whether the system is a solid or a liquid, several of which will be explored in the exercises. We will next consider one that involves the *relative* separation of two atoms. We monitor the square of the separation of two adjacent atoms, $(\Delta r)^2$, as a function of time, the idea being that in a solid this separation will remain (fairly) constant with time, while in a liquid it will not, since in the latter case the atoms will undergo diffusion. Some results of this kind are given in Figure 9.11, which shows $(\Delta r)^2$ as

²²Precisely what happened at $t \approx 11$ could be determined by examining snapshots before and after the displacement. The beauty (and power) of molecular dynamics is that we have such information in great detail

²³Strictly speaking, from these results we really can't distinguish between a liquid and a gas. That would require a study of the phase diagram

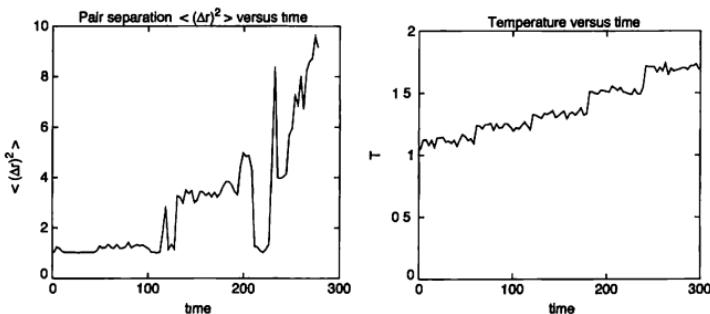


FIGURE 9.11: Left: mean-square separation of a pair of atoms as a function of time while a system similar to that in Figure 9.8 was slowly heated; right: corresponding variation of the temperature. The temperature was increased by rescaling the velocities by a factor of 1.1 at $t = 60, 120, 180, \dots$.

the system was slowly heated, starting again from the solid phase. The separation was initially ~ 1 (in reduced units) as the atoms were in adjacent lattice sites. This separation remained relatively constant up to about $t \approx 120$, when it increased to $\langle (\Delta r)^2 \rangle \sim 3.5$ (recall that here we are plotting the square of the separation), but then it stayed at that value. This indicates that one of the atoms in the pair jumped to a different lattice site, but since the separation remained at a (relatively) "quantized" value, the system was still a solid. Comparing this with the triangular lattice in Figure 9.6, we see that this separation corresponds to the separation of second nearest neighbors in our triangular lattice. [$\langle (\Delta r)^2 \rangle \sim 3.5$ in Figure 9.11 implies a spacing of about 1.9 in reduced units.] However, at $t \approx 200$ the separation began to change rapidly and erratically with time, signaling a much enhanced motion of one or both of the atoms. This is precisely what we expect for a liquid, and we conclude that melting occurred at $T \sim 1.5$. This value is a little more accurate than that obtained from Figure 9.10, as in this simulation the system was heated more slowly.

A key feature of these results is that the melting transition was, to within the resolution of these simulations, abrupt. There were no warnings or enhanced fluctuations to indicate that the transition was imminent. This is in accord with the claim we made earlier that the melting transition is *first order*. While this has admittedly been a very rough analysis of melting, our results do show that the behavior changes dramatically as the temperature is increased. To do a better job of locating the transition would require a careful examination of other properties. We will leave this to the exercises (and also to the readers' exploration of references).

EXERCISES

- 9.11. Study the melting transition at different densities. You should find that the melting temperature drops as the density is reduced. For 16 particles in a box of

size 4.3×4.3 , you should find melting near $T = 1.0$. Toxvaerd (1978) describes a careful study of the melting curve in two dimensions.

- 9.12.** In our determination of the lattice structure in Figure 9.6, we began the simulation with the particles in an ordered array. How do we know that the triangular lattice we found is really the stable lattice type? One way to address this question is to repeat the simulation with the atoms in different initial arrangements. For example, we could begin with the atoms in a honeycomb lattice (in which all of the atoms have three nearest neighbors), or you could start with a disordered arrangement. Perform this simulation at several temperatures below the melting temperature determined from the results in Figure 9.11 (keeping the same density as in that simulation), and determine the final stable structure of the atoms.
- 9.13.** Investigate melting in a three-dimensional system. First determine the structure of the solid (it should be face-centered cubic), then try to locate the melting temperature.
- *9.14.** A useful quantity for studies of structure is the pair correlation function $g(r)$. This is the density of particles per unit distance, at a separation r from any given particle. Calculate this from a simulation like that in Figure 9.8, with 16 particles in a 4×4 box and a time step of 0.001. Take one particle as the "origin" and let r be the distance as measured from this point. Divide the r axis into bins (try ~ 40 or so, from $r = 1$ to 3 for this rather small system), and after every 10 time steps record the number of particles in each bin. After many time steps this will yield a histogram that is proportional to $g(r)$. Compare the results for $g(r)$ in the solid and liquid phases. You should find that at low temperatures (i.e., in the solid phase) the correlation function has a large peak at $r \approx 1.1$, with a slightly smaller peak at $r \approx 1.9$ and a much smaller one at $r \approx 2.8$. These correspond to the spacing of first-, second-, and third-nearest neighbors in the (triangular) lattice (Figure 9.6). The (relative) size of the first peak should decrease and its width should increase when the system melts. Give a qualitative argument to explain this. Why do you expect to find peaks in $g(r)$ even in the liquid state? Rahman (1964) gives results for $g(r)$ for a three-dimensional Lennard-Jones system.
- *9.15.** Investigate condensation from a gas. Begin with the system in a fairly dilute high-temperature phase; a system of 16 particles in a 30×30 box is a good choice. Then gradually cool the system and see if you can observe condensation. *Hint:* It is worth thinking carefully about how you can distinguish a liquid from a gas.
- *9.16.** Consider again the process of self-diffusion discussed in the exercises in Section 9.1. Here we are interested in using the associated diffusion constant to study melting and, in particular, to locate the melting transition. Calculate the diffusion constant for self-diffusion for a system of 16 particles in a 4×4 box as a function of temperature. Compare its value in the solid and liquid phases. Can it be used to determine when the system melts? *Hint:* To improve your statistical accuracy, average the diffusion constants for all of the particles; this makes sense since they are all identical and should, therefore, diffuse in the same manner. However, be sure to account for the teleportation associated with the periodic boundary conditions when calculating the mean-square displacement of an atom.

9.3 EQUIPARTITION AND THE FERMI-PASTA-ULAM PROBLEM

Molecular dynamics lies at the interface between two of the foundational pillars of physics: classical dynamics and statistical mechanics. On the one hand, from the viewpoint of classical dynamics, the world contains a collection of masses that move according to Newton's laws. These laws can be expressed as differential equations; these equations have deterministic solutions, and are the basis of molecular dynamics. On the other hand, a fundamental basis of statistical mechanics is the ergodic hypothesis, which implies that systems with many degrees of freedom will move to a state of thermodynamic equilibrium. This leads to results such as equipartition, which we saw plays a central role in molecular dynamics. The determinism of classical dynamics may, at first glance, seem to be incompatible with the tendency towards disorder inherent in statistical mechanics. However, there is now a fairly happy marriage of these two areas of physics. The groundwork for this marriage was prepared by Poincaré in his work on the three body problem, which led, eventually, to the understanding of chaos in dynamical systems that we discussed in Chapter 3. The basic idea is that systems with many degrees of freedom can exhibit chaotic behavior, with an extreme sensitivity to initial conditions. This in turn can lead to effectively ergodic behavior. However, this simple picture of the interface between classical dynamics and statistical mechanics is actually a little too simple. This lesson is illustrated nicely by a problem that was introduced 50 years ago.

When general purpose computers first became available, several notable physicists, including von Neumann and Fermi, suggested that computational physics should be used to do "experiments." The idea was to use computational techniques to solve problems that had resisted theoretical attack. The insights gained from the computational results could then be used to guide subsequent analytic work, and thereby move a field forward. This approach has been used very effectively in recent years.

One of the very first problems that was attacked in this way is the Fermi-Pasta-Ulam (FPU) problem. The problem is sketched in Fig. 9.12. A collection of masses are connected by springs to form a one dimensional vibrating system. For simplicity, all of the masses are identical, all of the springs are the same, and the masses at the ends are connected (via springs) to rigid walls. This is a simple vibrating system, and is very similar to several problems that we have studied in other chapters, including vibrating strings and membranes. From that work we know what will happen if the masses in Fig. 9.12 are connected by Hooke's law (linear) springs. In this case the system can be described by normal modes of vibration. Each of these modes acts as an independent oscillator, with its own frequency and displacement pattern. The normal modes of the one dimensional FPU system with linear springs are, in fact, identical to the standing waves that one finds for a vibrating string (Chapter 6).²⁴ In this case we know that if the system is set up to vibrate in one normal mode, that vibration will continue forever, with the energy staying in the initial mode. In a linear system (Hooke's law springs) each normal mode behaves as a completely independent oscillator, uncoupled from all of the other modes.

²⁴These frequencies and displacement vectors can also be obtained by solving the corresponding

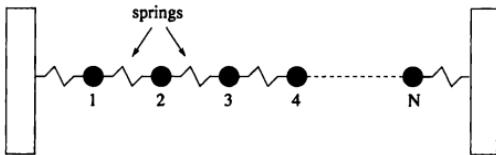


FIGURE 9.12: Fermi-Pasta-Ulam system of masses connected by nonlinear springs. The black dots are each of mass m , and are connected by nonlinear springs described, e.g., by (9.15) or (9.16).

However, the situation is quite different when the springs are nonlinear. Such a nonlinear system does not possess “pure” normal modes. When the nonlinearity is weak, one can think of the system as consisting of weakly coupled modes, and this coupling will cause energy to be exchanged and shared among the modes. Roughly speaking, this sharing is the basis for equipartition of energy and hence ergodicity. Fermi, Pasta, and Ulam wanted to study this energy sharing process in detail. They expected that adding a small nonlinearity to the springs would lead to equipartition of energy, and they wanted to calculate how fast this happens. This calculation can then form the basis for understanding phenomena such as thermal conductivity, a problem that has been studied by subsequent workers. However, we will see that FPU had their hands full with the equipartition problem, as the system in Fig. 9.12 with nonlinear springs did *not* behave in the simple manner expected by FPU.

A simulation of the FPU system is very similar to a molecular dynamics calculation, as we can again use the Verlet method (9.4). The FPU problem is a bit simpler, since (according to the original formulation by FPU) we only need to worry about motion along one direction, which we will call x . If x_i is the displacement of mass i from its equilibrium position in Fig. 9.12, the equation of motion for this mass (just Newton’s second law, $F_i = m_i a_i$) has the form

$$m_i \frac{d^2 x_i}{dt^2} = f_{\text{spring}}(x_i - x_{i+1}) + f_{\text{spring}}(x_i - x_{i-1}), \quad (9.14)$$

which is essentially the same as (9.3). The force is now given by the spring functions $f_{\text{spring}}(\Delta x)$, where Δx is the amount that a spring is stretched or compressed. FPU considered several different types of nonlinear functions for f_{spring} . Two convenient and much studied choices are a quadratic spring

$$f_{\text{spring}}^\alpha(\Delta x) = -K(\Delta x) - \alpha(\Delta x)^2, \quad (9.15)$$

and a cubic spring

$$f_{\text{spring}}^\beta(\Delta x) = -K(\Delta x) - \beta(\Delta x)^3, \quad (9.16)$$

and we will refer to these as alpha and beta-type springs. The two types of springs lead to very similar behavior, so we will only show simulation results for one of them, the beta-type; we’ll leave it to you to explore the other type in the exercises

eigenvalue problem, as we discuss in Chapter 11

We can derive the equation of motion for an FPU system with beta-type springs by inserting the force law (9.16) into the equation of motion (9.14). For simplicity (and following the work of FPU) we will take $m = 1$ for all of the masses, and the Hooke's law term $K = 1$ for all of the springs, so that β measures the strength of the nonlinearity. If we have N masses, then the equation of motion (9.14) is actually a system of N equations. For each one we write the second derivative term in finite difference form (9.5), with a time step Δt . Using our usual notation we let $x_i(n)$ be the displacement of mass i at time step n . We can rearrange our finite difference equations to solve for the displacement of each mass at time step $n + 1$ in terms of the displacements at previous time steps, with the result

$$\begin{aligned} x_i(n+1) &= 2x_i(n) - x_i(n-1) + \frac{1}{(\Delta t)^2}[x_{i+1}(n) + x_{i-1}(n) - 2x_i(n)] \quad (9.17) \\ &\quad + \frac{\beta}{(\Delta t)^2}([x_{i+1}(n) - x_i(n)]^3 + [x_{i-1}(n) - x_i(n)]^3). \end{aligned}$$

From a numerical point of view, (9.17) is equivalent to both the Verlet method *and* to the approach we used to deal with waves on a string in Chapter 6. We thus expect (9.17) to be stable and efficient.

We now follow the work of FPU and consider the behavior when we begin in one of the normal modes of the corresponding linear system. As we already mentioned, these normal modes are simply standing waves, and they can be described by

$$x_i = A\sqrt{\frac{2}{N+1}} \sin\left(\frac{ik\pi}{N+1}\right), \quad (9.18)$$

where N is the number of masses in the system, and k is the mode number ($k = 1, 2, 3, \dots, N$).²⁵ The wavelength λ_k of this mode is equal to $2(N+1)/k$. The first three of these normal modes are sketched in Fig. 9.13.

We are now ready to consider the behavior of a FPU system. We consider a system with $N = 32$ masses, with beta-type springs. The system is given an initial displacement corresponding to the lowest mode (mode 1 in Fig. 9.13) and then released. Since there are 32 vibrating masses, there are a number of different ways that we could plot the behavior. One useful way to display the results is to examine the energy in the different modes. An FPU system is analogous to a vibrating string, and the normal modes correspond to standing waves. At any particular instant in time the FPU displacement profile can be written as a sum of these normal mode displacements; i.e., just as the profile of a vibrating string can be written as a collection of standing wave displacements. Since the FPU normal modes are sine waves (9.18), this is also equivalent to writing the FPU displacement as a Fourier series. Hence, the displacements of the FPU masses can be written as

$$x_i = \sqrt{\frac{2}{N+1}} \sum_{k=1}^N a_k \sin\left(\frac{ik\pi}{N+1}\right) \quad (9.19)$$

²⁵The factor in front of the sin term in (9.18) follows the standard terminology in the FPU literature. Including this factor will make it simpler to compare your results with those of others. A is an arbitrary amplitude factor.

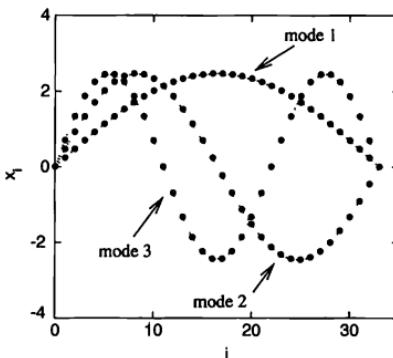


FIGURE 9.13: Lowest three normal modes of a linear FPU system with $N = 32$ masses with the amplitude $A = 10$. These are simply standing waves, with zero displacement (nodes) at each end. The black dots represent the masses, and the dashed lines can be thought of as the connecting springs

Here the sum is over the N normal modes, and a_k is the amplitude (or contribution) of mode k . Equation (9.19) is just a Fourier transform, so there is also an inverse transform

$$a_k = \sqrt{\frac{2}{N+1}} \sum_{i=1}^N x_i \sin\left(\frac{ik\pi}{N+1}\right). \quad (9.20)$$

There will be kinetic and potential energies associated with each mode, so the energy of mode k will be given by

$$E_k = \frac{1}{2} \left[\left(\frac{da_k}{dt} \right)^2 + \omega_k^2 a_k^2 \right]. \quad (9.21)$$

The first term is just the kinetic energy of the mode (proportional to the square of the mode velocity), while the second term is the potential energy. The frequency of mode k is²⁶

$$\omega_k = 2 \sin\left(\frac{k\pi}{2(N+1)}\right) \quad (9.22)$$

Our FPU simulation yields the displacements of all of the masses, x_i , as functions of time. We can then use (9.20) to calculate the amplitude of each mode a_k as a function of time, and obtain the energy of each mode using (9.21). Some results are shown in Fig. 9.14, which shows very striking and unexpected behavior. The energy starts in mode 1, and a small amount rapidly goes into the higher

²⁶Note that this corresponds to $\omega_k = 2 \sin(\pi/\lambda_k)$. The similar problem of finding the frequencies of string vibration modes is discussed in Chapters 6 and 11.

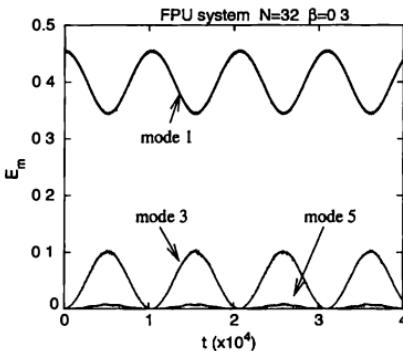


FIGURE 9.14: Energy in modes 1, 3, and 5 as a function of time for an FPU system with $N = 32$ and $\beta = 0.3$. The system was initially given a displacement (9.18) with $m = 1$ (mode 1), with amplitude $A = 10$. For this simulation we used $\Delta t = 0.05$

modes²⁷ 3 and 5. However, this energy quickly *returns* to mode 1, and the modal energies appear to oscillate forever. Hence, energy is *not* shared equally between the modes; there is no equipartition of energy. Indeed, the system seems to oscillate in a very regular manner, with no hint of the chaotic behavior we had expected to see for an ergodic system

This result was very surprising to Fermi, Pasta, and Ulam, and they spent a good deal of time investigating the behavior in many other cases. Figure 9.15 shows results for larger values of β . Increasing β makes the system more nonlinear, which should produce stronger mixing of the energy between modes, and that is what we seem to see with $\beta = 1$. The oscillations are not quite as regular as at smaller β (Fig. 9.14), and more energy finds its way to the other modes, but the energy is still not distributed in any sort of even way across the modes. However, when the nonlinearity is increased further to $\beta = 3$ the behavior changes dramatically. The system now appears to be chaotic and the energy is shared much more evenly between the two modes that are considered in Fig. 9.15. The behavior here certainly seems consistent with being ergodic, although a more thorough analysis (some of which is left for the exercises) is needed to confirm this.

One way to investigate the possibility of chaos at large values of β is to study the trajectories in the appropriate phase space. In the FPU problem, the normal mode amplitudes in (9.20) form a very useful phase space. However, since there are $N = 32$ masses, we have a $2N = 64$ dimensional phase space corresponding to the

²⁷Note that for an FPU system with beta-type springs, the symmetry of the spring force does not allow energy to move from an odd to an even mode. That is, parity is conserved. So, in this example no energy is transferred to the even modes.

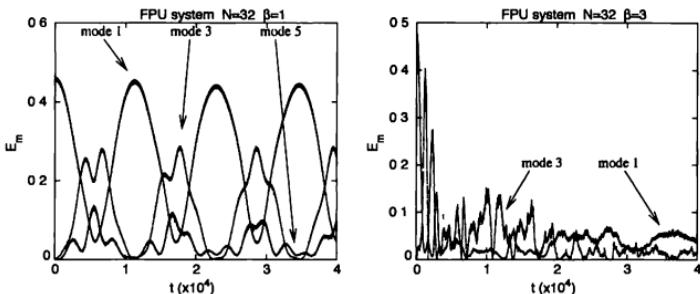


FIGURE 9.15: Energy as a function of time for a few of the lowest modes of an FPU system with $\beta = 1$ (left) and $\beta = 3$ (right). Each system was initially in mode 1 at $t = 0$, with $A = 10$

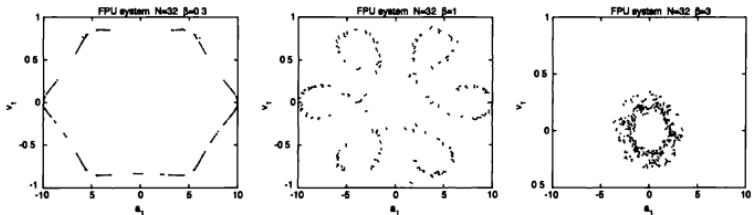


FIGURE 9.16: Phase space plots of the trajectory of an FPU system with beta-type springs. Here we have plotted the “velocity” of mode 1 (da_1/dt) as a function of a_1 whenever the system crosses the $a_3 = 0$ plane in phase space

amplitudes and velocities of all of the normal modes. We must therefore consider a restricted portion of this space. It is useful to again think of our FPU system as moving on a trajectory in this phase space. We can then ask where the system is located when it crosses a particular plane in phase space. You may recall that this is precisely the approach we took in constructing phase space plots in our study of the Lorenz model in Chapter 3. In Fig. 9.16 we have plotted the “velocity” of mode 1 (just da_1/dt) as a function of the amplitude of mode 1 (a_1), at those times when the trajectory crosses the plane $a_3 = 0$. When β is small the phase space plot is very regular. For $\beta = 0.3$ the phase plot approximates a hexagon, with only a few points lying off this curve. If one studies the trajectory carefully, one finds that these few points occurred at the very beginning of the simulation. Hence, over time the system settled down and the trajectory was “attracted” to the hexagon. This should remind you of the behavior we encountered in Chapter 3 in our studies of chaos; here the hexagon is acting very much like an attractor in phase space.

Somewhat similar behavior is found with $\beta = 1$. Recall that for this value of β we found very little sharing of energy between modes in Fig. 9.15. The corresponding phase space plot in Fig. 9.16 shows another regular trajectory, although it is more complex than the hexagon found when $\beta = 0.3$. However, when β is increased further things change dramatically. The phase plot with $\beta = 3$ has no trace of regularity, and confirms our suspicion that chaos is lurking very nearby.

We do not have space in this book to fully explore the FPU problem. Indeed, entire books have been written on the subject (see the end of chapter references) Even so, we can appreciate the central insight of the FPU problem – that the interface between classical dynamics and statistical mechanics is much more complex than had been expected by Fermi, Pasta, and Ulam. In fact, this interface is *very* complex, and we will leave it to the references to take you deeper into this subject. Our results in this section do show a close connection to the behavior we encountered in our studies of chaos in Chapter 3. When the nonlinearity in the FPU problem is large (i.e., β is large) we have found chaotic behavior, while we know that the corresponding linear system is completely regular and non-chaotic. The surprise has come in the interface region of small nonlinearity. Here we found hints of both regular and chaotic behavior. The complications found in this interface region are, in a sense, associated with the route to chaos in this system.

EXERCISES

- 9.17. Repeat the FPU calculation in Fig. 9.14. Compute the total energy of the system, and confirm that it stays constant throughout the simulation. Thus in the FPU problem, chaotic-type states can be obtained even in the absence of external power supplied, in contrast to the case of the driven pendulum of Chapter 3.
- 9.18. Investigate an FPU system with alpha-type springs. You should find that the behavior is similar to that found with beta-type springs, with very little sharing of energy between modes when α is small. Here, “small” values are in the neighborhood of $\alpha = 0.25$ or less. Also, show that with alpha springs the energy can move between even and odd modes.
- 9.19. Carry out an FPU simulation similar to that in Fig. 9.14, but investigate the behavior when the system is initially placed in other modes, such as mode 2 and mode 3.
- *9.20. Investigate equipartition in an FPU system with large β , as in Fig. 9.15. Examine carefully how much energy actually flows to the higher modes, and try to develop a criteria for determining if the system is really ergodic.
- 9.21. Study other types of phase space plots for the FPU problem by constructing other projections of the trajectory in phase space. For example, you might make projections on the $a_1 = 0$ or $v_3 = 0$ planes.
- *9.22. Study the sensitivity to initial conditions in the FPU problem. Use this to determine if the system is really chaotic

REFERENCES

- [1] B. J. Alder, and T. E. Wainwright, “Studies in Molecular Dynamics I. General Method,” *J. Chem. Phys.* **31**, 459 (1959). Pioneering application of molecular dynamics to a system of hard disks.

- [2] D. Chandler, 1987, *Introduction to Modern Statistical Mechanics*, Oxford, New York, Section 7.4.
- [3] E. Fermi, E., J. Pasta, and S. Ulam, 1955, "Studies of Nonlinear Problems I," Los Alamos Technical Report LA-1940. This is the paper that started the FPU problem. It never actually appeared in a scientific journal, as Fermi died before it could be fully written up and published. However, this report can be found in E. Fermi, 1965, *Collected Papers, Vol. II*, University of Chicago Press, Chicago.
- [4] J. Ford, "The Fermi-Pasta-Ulam Problem: Paradox Turns Discovery," *Phys. Reports* **213**, 271 (1992). An advanced, but very readable account of the work that led to our current understanding of the FPU problem, and related issues in nonlinear dynamics.
- [5] D. W. Heermann, 1990, *Computer Simulation Methods in Theoretical Physics, 2d ed.*, Springer-Verlag, New York. Contains a nice introduction to molecular dynamics and discusses how the method can be used to simulate a system that evolves according to the canonical ensemble.
- [6] D. Marx and J. Hutter, 2000, "Ab initio molecular dynamics: Theory and Implementation," in *Modern Methods and Algorithms of Quantum Chemistry*, J. Grotendorst, ed., NIC Series, vol. 1, John von Neumann Institute for Computing.
- [7] N. D. Mermin, "Crystalline Order in Two Dimensions," *Phys. Rev.* **176**, 250 (1968). Some intriguing exact results concerning the (non)stability of a hypothetical *infinitely* large solid.
- [8] D. R. Nelson, 1983, "Defect-mediated Phase Transitions," in *Phase Transitions and Critical Phenomena, Vol. 7*, C. Domb and J. L. Lebowitz, eds., Academic Press, Orlando. Contains a detailed discussion of the theory of melting in two dimensions.
- [9] R. K. Pathria, 1996, *Statistical Mechanics*, 2nd Ed., Butterworth Heineman, Oxford, Section 3.7.
- [10] A. Rahman, "Correlations in the Motion of Atoms in Liquid Argon," *Phys. Rev.* **136**, A405 (1964). A very readable paper describing a molecular dynamics study of pair correlations in a Lennard-Jones liquid.
- [11] F. Reif, 1965, *Fundamentals of Statistical and Thermal Physics*, McGraw-Hill, New York. Reviews statistical mechanics, including a discussion of the Maxwell-Boltzmann distributions.
- [12] L. I. Schiff, 1968, *Quantum Mechanics, 3d ed.*, McGraw-Hill, New York. Contains a nice discussion of the origin of the Van der Waals interaction, which is a key ingredient of the Lennard-Jones potential.

- [13] S. Toxvaerd, "Melting in a Two-Dimensional Lennard-Jones System," J. Chem. Phys. **69**, 4750 (1978). A careful study of melting in two dimensions using molecular dynamics.
- [14] L. Verlet, "Computer 'Experiments' on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules," Phys. Rev. **159**, 98 (1967). Description of the Verlet method and a nice description of some molecular dynamics simulations.
- [15] T. P. Weissert, 1997, *The Genesis of Simulation in Dynamics: Pursuing the Fermi-Pasta-Ulam Problem*, Springer-Verlag, New York. A review of the Fermi-Pasta-Ulam problem and its role in the development the field of classical dynamics. Lots of interesting historical anecdotes.

Quantum Mechanics

There are only a few problems in quantum mechanics that can be solved exactly, most notably the harmonic oscillator, a particle in a box, and the hydrogen atom. Nearly all other nontrivial quantum problems either have no known analytic solutions or can be attacked analytically only with extreme difficulty. This is why perturbation methods play such an important role in quantum theory and also why numerical methods are an attractive alternative. Indeed, an extremely wide variety of numerical methods have been developed for dealing with quantum problems. There are far too many such methods for us to even mention all of them in this chapter. Our goal instead will be to describe a few of the algorithms that have been developed with quantum mechanics in mind, and use them to treat several representative problems.¹ We also hope to find some enlightening overlap with problems and techniques we have encountered in earlier chapters.

This chapter opens with a brief review of the Schrödinger equation, and a few of the fundamental ingredients of quantum theory. We then consider several methods for treating time-independent problems. While we have dealt with partial differential equations in a number of previous settings, the problems associated with the Schrödinger equation involve extra complications associated with boundary conditions and require some extensions of the methods we have previously employed. We also describe how Monte Carlo methods can be teamed up with the variational principle to calculate wave functions and eigenvalues. In the second half of this chapter we turn to time-dependent problems and describe several numerical approaches. The goal, as usual, is to learn how to deal with problems that are difficult or impossible to treat analytically. However, we will find it useful to apply our numerical approaches to several exactly soluble problems, as this will allow us to test the methods and also illustrate some of the key themes of quantum theory.

10.1 TIME-INDEPENDENT SCHRÖDINGER EQUATION: SOME PRELIMINARIES

The time-independent Schrödinger equation for a particle in three dimensions is

$$-\frac{\hbar^2}{2m} \nabla^2 \psi + V(\vec{r}) \psi = E \psi, \quad (10.1)$$

where \hbar is Planck's constant, m is the mass of the particle, V is the potential energy, E is the energy of the particle, and ψ is the wave function. As is well known with quantum theory, it is much easier to write this equation down than to really understand it. The key quantity here is the wave function, which has no direct classical counterpart. In the most general case, ψ is a complex function

¹It is not our intention that this chapter be a substitute for a first course in quantum mechanics, although we will review a few aspects of the theory in the next several pages

of position and, when properly normalized, $\psi(\vec{r})^*\psi(\vec{r})$ is the probability per unit volume that the particle will be found at \vec{r} . Here we use the usual notation in which ψ^* is the complex conjugate of ψ .

The fact that ψ can be complex will complicate our numerical treatments.² However, another aspect of (10.1) will turn out to be much more important in numerical attacks on this equation. This is a partial differential equation whose form is not very different from the wave equation encountered in Chapter 6, with one important exception. The energy E is *also* an unknown. A complete solution involves determining both ψ and the corresponding energy. These are also known as the eigenfunction and eigenvalue of the equation.³ An interesting property of eigenvalue problems is that in many cases solutions exist only for certain special values of the eigenvalue, in this case E . Mathematically, this is the origin of the discrete energy levels of quantum theory. It is useful to illustrate these points with an example that can be solved analytically; we will revisit this problem shortly using a numerical approach. Consider a particle moving in free space, that is, a region in which the potential is constant. Since we are free to shift the origin of the potential energy scale, we will take $V = 0$ everywhere. For simplicity we also assume that space is one-dimensional so that $\psi = \psi(x)$ and the ∇^2 operator in (10.1) reduces to the second derivative d^2/dx^2 . The Schrödinger (time-independent) equation then becomes

$$-\frac{\hbar^2}{2m} \frac{d^2\psi}{dx^2} = E\psi, \quad (10.2)$$

which has the general solution

$$\psi = A \exp(ikx), \quad (10.3)$$

where $i \equiv \sqrt{-1}$, and A and k are constants. That this is indeed a solution to (10.2) can be verified by substitution, which yields

$$E = \frac{\hbar^2 k^2}{2m}. \quad (10.4)$$

The wave function (10.3) has the form of a plane wave with wave vector $k = 2\pi/\lambda$, where λ is the wavelength of the particle. k is also closely related to the particle's momentum, p , with $p = \hbar k$.

In this particular problem there is a solution to the Schrödinger equation for any value of k , and thus all nonnegative values of E are allowed. We do not have quantized energy levels in this example (at least not yet). The constant A is not determined directly by the Schrödinger equation. Our solution satisfies the wave equation (10.2) for any value of A , a fact we should have expected since the equation is linear. Constants such as A , which set the overall magnitude of the wave

²While the wave function may be complex, there are many cases in which its phase can be chosen so as to make ψ real. In fact, in our numerical work we will not encounter any situations that absolutely demand that ψ be complex until we deal with time-dependent problems.

³Equations in which an operator, such as the ∇^2 operator in (10.1), acting on a function is equal to a constant times the function are known as eigenvalue equations and arise in many other contexts, a few of which are described in Chapter 11. See also Appendix H. Griffiths (1995) and Schiff (1968) contain nice introductions to this topic in the context of quantum mechanics.

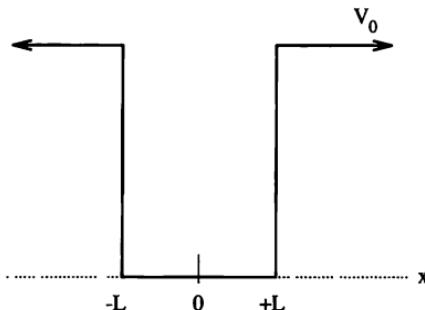


FIGURE 10.1: Potential energy for a particle in a box. The potential is $V = 0$ inside the box (i.e., for $|x| \leq L$) and V_0 outside. If $V_0 = \infty$, then the box is said to have "hard" walls, while if V_0 is finite, the walls are "soft."

function, are determined by the physical interpretation of ψ and its relation to the probability density. We already noted that the probability of finding the particle in a particular region of space is proportional to $\psi^* \psi$. For our one-dimensional case $\psi(x)^* \psi(x) dx$ is the probability of finding the particle within a region of length dx in the neighborhood of x . Since we are considering the behavior of a single particle, this probability must satisfy the normalization condition

$$\int \psi^* \psi dx = 1. \quad (10.5)$$

In words, the total probability for finding the particle someplace must be unity. This condition can generally be used to set the overall magnitude of the wave function. However, applying it to our plane wave yields a minor difficulty. For the wave function (10.3) we have $\psi^* \psi = A^2$, so that $\int \psi^* \psi dx = A^2 \int dx$. If our particle is allowed to roam freely over an infinitely large region, we must make A infinitely small in order to preserve normalization.

It is convenient to deal with this situation by assuming that the particle is confined to a box that extends from $x = -L$ to $x = +L$. The plane wave (10.3) must then fit in this box. The corresponding potential is now $V = 0$ inside the box and very large outside, as sketched in Figure 10.1. Since $V(x)$ is a discontinuous function of position, it is simplest to solve for ψ separately in the regions $|x| \leq L$ (inside the box) and $|x| > L$ (outside the box). Inside we have $V = 0$, so ψ in this region is still given by (10.3). If we assume that the potential outside is $V = \infty$ and that the energy of the particle is not infinite, then the only way we can satisfy the Schrödinger equation in the outside region is if $\psi_{\text{outside}} = 0$. This implies that $d^2 \psi_{\text{outside}} / dx^2 = 0$ also, so both sides of (10.2) are zero. Physically this result should not be surprising. A particle with a finite energy will have no chance of being found in a region where $V = \infty$, so the probability amplitude ψ must vanish.

To complete our solution we make use of one more constraint on ψ ; it must be a continuous function⁴ of x . Since ψ vanishes for $x > \pm L$, the solution for ψ inside the box must vanish at $x = \pm L$. This constraint can be satisfied by noting again that the Schrödinger equation is linear, so the sum of two solutions is also a solution. We can therefore construct wave functions of the form

$$\begin{aligned}\psi_+ &= \frac{A}{2} [\exp(i k_+ x) + \exp(-i k_+ x)] = A \cos(k_+ x), \\ \psi_- &= \frac{A}{2} [(\exp(i k_- x) - \exp(-i k_- x)] = A \sin(k_- x).\end{aligned}\quad (10.6)$$

These are just standing waves, similar to what is found with a vibrating string. The condition $\psi(\pm L) = 0$ requires that

$$k_+ = \frac{\pi}{2L}, \frac{3\pi}{2L}, \dots = \frac{(2n-1)\pi}{2L} \quad (10.7)$$

$$k_- = \frac{\pi}{L}, \frac{2\pi}{L}, \dots = \frac{n\pi}{L}. \quad (10.8)$$

This constraint on k means that we now have discrete energy levels. Only certain special values of the wave vector are allowed. These values are those yielding standing waves that satisfy the boundary conditions on ψ at the walls of the box, as shown in Figure 10.2. Quantization of k also implies quantization of the energy since $E = \hbar^2 k^2 / 2m$. Constraints such as these boundary conditions generally lead to quantized levels. We will find that they play a major role in constructing numerical solutions to the Schrödinger equation.

It is interesting to notice that the wave functions (10.6) all have a definite parity; they are either even $\psi(+x) = \psi(-x)$, or odd $\psi(+x) = -\psi(-x)$, with respect to x . This property can be traced to the potential energy. Whenever the potential is an even function of x , the wave functions can always be written so as to have a definite parity. This will prove useful in developing a numerical approach in the next section. Note also that the wave functions (10.6) can be properly normalized without coefficients that are infinitely small. The normalization condition (10.5) can be used to determine the value of A for these functions, yielding $A = L^{-1/2}$.

Two other features of the wave functions (10.6) will turn out to be very useful later in this chapter. The forms of these functions may remind you of the sines and cosines encountered in connection with Fourier transforms in Chapter 6 and Appendix C. It is thus not surprising that these wave functions form a complete set of orthogonal functions over the interval $|x| \leq L$. The term "orthogonal" means that $\int \psi_1^* \psi_2 dx = 0$ unless ψ_1 and ψ_2 refer to the same eigenstate. The term "complete set" means that any function⁵ in this interval that vanishes at $x = \pm L$ can be written as a sum of these wave functions with suitably chosen coefficients. The orthogonality property makes this expansion, that is, the set of coefficients, unique. Such a collection of functions is usually referred to as a *basis set*. The similarity to Fourier transforms, in which the basis set is also composed of sines

⁴This condition is required to make quantities such as the momentum physically well behaved.

⁵We will not worry about pathological functions here.

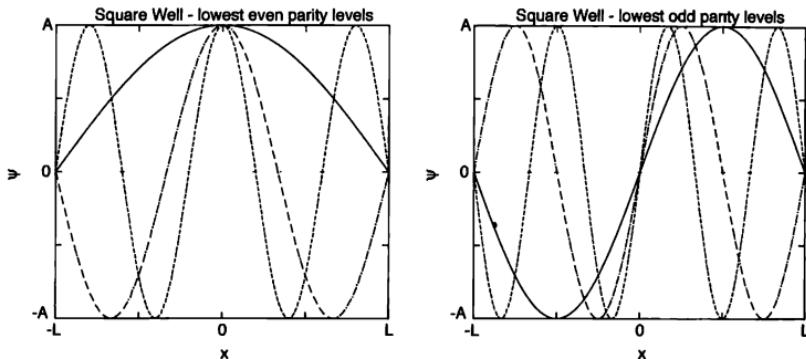


FIGURE 10.2: Wave functions for the lowest six eigenstates for a particle in a box. These are the standing waves in (10.6). The figure on the left shows the cosine-like (even parity) standing waves, while the sine-like (odd parity) wave functions are shown at right. In each case the solid curve is the lowest energy state (of that parity). Note that $A = L^{-1/2}$ is required for the wave functions to be properly normalized.

and cosines, should be evident. However, a basis set need not be this simple, as we will see in an example later in this chapter.

10.2 ONE DIMENSION: SHOOTING AND MATCHING METHODS

We next consider how we can solve numerically the time-independent Schrödinger equation in one dimension, using methods similar to those we have previously developed in connection with ordinary differential equations. We begin with the particle-in-a-box problem, as this will allow us to compare with the analytic solution obtained above. We will then consider several problems for which analytic solutions are not available.

The Schrödinger equation in one dimension (10.2) is very similar in form to the equations of motion we encountered in connection with projectiles in Chapter 2. In those cases we had a differential equation involving the position x , as a function of time. Starting with initial values for x and dx/dt we were able to numerically integrate forward in time to obtain $x(t)$. Now we have a differential equation for ψ as a function of x and we want to use a similar approach. To do this we need the values of ψ and $d\psi/dx$ at some location so that we can begin the integration. These “initial conditions” for the wave function and its derivative are analogous to the initial conditions in a projectile problem. For a projectile these conditions are determined by how the object is struck or thrown, etc. The initial conditions for a wave function are obtained in a somewhat different manner.

When the potential is symmetric, as it is for our particle in a box, we can use the symmetry to our advantage. We noted earlier that if $V(x)$ is an even function of x , the wave functions can be written as purely even or purely odd functions of x . That is, we can choose the wave function to satisfy either⁶ $\psi(+x) = \psi(-x)$, or $\psi(+x) = -\psi(-x)$. Indeed, we have already observed this symmetry in the analytic solutions for the infinite square well, Figure 10.2. An even parity solution has $d\psi/dx = 0$ at $x = 0$, and it is safe to assume (as we will see shortly) that $\psi(0) \neq 0$. Since the Schrödinger equation is linear, we also know that a solution can be multiplied by a constant factor and still be a solution. We are therefore free to *provisionally* choose $\psi(0) = 1$ as an initial value for integration of the Schrödinger equation. When we are finished and have a numerical solution of the equation in hand, we can then normalize that result by a constant factor so as to satisfy the probability constraint $\int \psi^* \psi dx = 1$ and thereby obtain a physically acceptable wave function.

The same approach can be used to obtain an odd-parity solution. Since in this case $\psi(x) = -\psi(-x)$, we must have $\psi(0) = 0$ and $d\psi/dx \neq 0$ at $x = 0$. We are again free to pick the scale of ψ , as long as we normalize it at the completion of the calculation, so we can use $\psi(0) = 0$ and $d\psi/dx = 1$ as initial conditions.

By exploiting the symmetry we have thus obtained values for ψ and $d\psi/dx$ at $x = 0$. These are the initial conditions for our integration. The next step is to rewrite (10.2) in finite-difference form. In most of our previous examples we have written such second-order differential equations as two separate first-order equations. Here, since we aren't interested in the derivative $d\psi/dx$, we will take a slightly different approach (which may remind you of the Verlet method). As usual, we discretize space in steps of size Δx and write $\psi_n \equiv \psi(n\Delta x)$. The second derivative in (10.2) can be written in the usual finite-difference form

$$\frac{d^2\psi}{dx^2} \approx \frac{\psi_{n+1} + \psi_{n-1} - 2\psi_n}{(\Delta x)^2}, \quad (10.9)$$

which is similar to the expression we encountered in connection with the wave equation in Chapter 6. Inserting this into (10.2) we have

$$-\frac{\hbar^2}{2m} \frac{d^2\psi}{dx^2} \approx -\frac{\hbar^2}{2m} \left[\frac{\psi_{n+1} + \psi_{n-1} - 2\psi_n}{(\Delta x)^2} \right] \approx (E - V_n) \psi_n, \quad (10.10)$$

where we have written $V_n = V(n\Delta x)$ and have used the \approx signs to emphasize that this is only an approximation to the original differential equation. This can be rearranged to obtain ψ_{n+1} in terms of ψ_n and ψ_{n-1} . For convenience we will now follow the usual convention for such problems and employ units in which $\hbar = 1$ and $m = 1$, which leads to

$$\psi_{n+1} = 2\psi_n - \psi_{n-1} - 2(\Delta x)^2(E - V_n)\psi_n. \quad (10.11)$$

Let us first consider the even-parity solutions. We have already argued that we can take $\psi(0) = 1$, so we will first apply (10.11) at $n = 0$ (i.e., $x = 0$). Since

⁶If, in addition, the eigenvalue E is non-degenerate (i.e., if there is only one linearly independent solution with this value of E), then the eigenfunction $\psi(x)$ for E must possess a definite parity.

we also have $d\psi/dx = 0$ at $x = 0$, we take $\psi_{-1} = 1$ as well, so we can use (10.11) to calculate $\psi_{n+1} = \psi_1$. We then move on and use ψ_0 and ψ_1 to obtain ψ_2 . The process can be repeated to find ψ_n for all values of n that are of interest.

This procedure is similar to the way we dealt with projectile motion. However, we now have a complication that we didn't have with baseballs. From the analytic solution for a particle in a box we saw that with hard walls ($V_0 = +\infty$) the wave function must vanish at the walls. This means that we have a boundary condition that ψ must satisfy at $x = \pm L$. In terms of projectiles this is analogous to requiring that the baseball land at a precise location. This will only happen if the initial velocity of the ball has a particular value, or perhaps a value chosen from a discrete set of possibilities. Likewise, we will see that for our wave function the conditions at $x = \pm L$ will be satisfied only for certain values of the energy.

It is useful at this point to consider, in a qualitative sense, the behavior of ψ_n obtained by iterating (10.11) with different values of E . Schematic results for three different values of the energy near the ground-state value E_G (you should recall that the ground state is an even-parity solution) are shown in Figure 10.3. We already know E_G from the analytic results. For the units we have chosen here ($\hbar = 1$, $m = 1$, and a box with $L = 1$), the ground state has an energy⁷ of $E_G = \pi^2/8 = 1.2337\dots$. If we were to use (10.11) to calculate ψ for a value of E that is smaller than this, we would find that ψ is greater than zero when we reach $x = L$ and then diverges to $+\infty$ at larger x , so the boundary condition is *not* satisfied. This is illustrated by the top curve in Figure 10.3. While this function is a solution of (10.2), it cannot be normalized and thus cannot be made to yield a physically acceptable probability density. It is therefore not an allowed wave function for our particle. On the other hand, if we were to perform this calculation with a value of E that is slightly larger than E_G , we would find that ψ drops too quickly with x , so that it is negative when we reach $x = L$ and diverges to $-\infty$ at larger x , as illustrated in Figure 10.3. Only if E has *precisely* the value E_G will the wave function reach zero at precisely $x = L$ and remain zero at larger values of x . This wave function is normalizable and is the ground state ψ for our particle in a box.

This approach for solving a differential equation that has boundary conditions that must be satisfied at *both* ends of an interval is known as the *shooting method*. The method is analogous to trying to throw a baseball or shoot a cannon shell so as to hit a specified target. Only for certain special conditions, in this case the value of E , will the function satisfy the required boundary conditions (that is, will the ball land at the desired spot). In the present case the boundary conditions⁸ are such that the wave function must vanish at two particular locations ($x = \pm L$). This was a result of choosing the potential outside the box to be infinitely large. We will see in a moment how to handle cases in which the potential energy varies more slowly with position, such as a box with “soft” walls or a harmonic oscillator.

Let us next consider how to construct a program that implements this approach. The basic plan is to use (10.11) to calculate $\psi(x)$ for different values of the energy, and choose the result that comes closest to satisfying the boundary

⁷This can be verified from the results for k_+ (10.7) in the previous section

⁸Because of the symmetry, we in effect have boundary conditions at $x = 0$ and $x = L$.

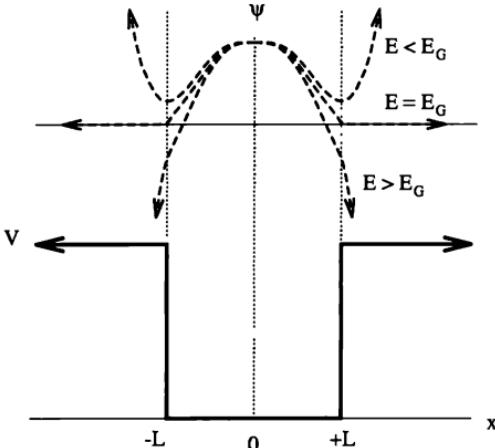


FIGURE 10.3: Schematic results for ψ calculated from (10.11) for several values of E . Top: behavior for the even-parity solution with E near the ground-state energy, E_G . Bottom: corresponding potential energy. The potential outside the box ($|x| > L$) is assumed to be very large.

conditions at the walls of the box. This procedure will yield *both* the wave function and the corresponding energy. We will actually design our program to search systematically for the proper value of E and thereby approach the correct value very closely. The routine sketched below uses the shooting method to integrate starting from $x = 0$ (we again assume $L = 1$), beginning with $\psi_{-1} = \psi_0 = 1$ corresponding to an even-parity solution.⁹ After initializing variables such as the spatial step size Δx and the initial estimate for the energy E , the subroutine calculate is called. This routine integrates from $i=0$ (which corresponds to $x = 0$) to larger values of i , stopping when the wave function ψ begins to diverge. At this point the loop is exited and the results for ψ can be displayed. From Figure 10.3 we know that if E is not precisely equal to one of the eigenvalues, the value of the energy should now be adjusted either up or down depending on which direction ψ last diverged. Our routine adds an amount ΔE to E , giving a new estimate for the energy, and the integration is repeated. Let us suppose that E has just been increased ($\Delta E > 0$). If in the next trial ψ diverges in the *same* direction as in the previous case, this means that we did not change the energy enough.¹⁰ The

⁹We will leave it to the exercises to modify this routine to deal with the odd-parity solutions

¹⁰This assumes that the last variation in E was made in the *correct* direction. Once the procedure starts to home in on a particular eigenvalue, having reversed the sign of ΔE once or more, this assumption is always true. However, initially, before the sign of ΔE is reversed the first time, it may not be true and the procedure may run away looking for an eigenvalue lower than the ground state energy E , or else home in on an eigenvalue different from the one we are after.

routine would then increase E again and repeat the integration. However, if ψ now diverges in the opposite direction, that means we have changed E by too much and have *overshot* the correct value of the energy. The `calculate` routine then cuts the value of ΔE in half, changes its *sign*, and adds it to E , so that we now proceed back toward the solution. We thus hunt down the correct value of the energy with ever-decreasing values of $|\Delta E|$.

EXAMPLE 10.1 Interactive shooting method routine `calculate`

- Input are the number N and the size Δx of the spatial grids, the initial guess for the energy E and increment ΔE , and the cutoff parameter b (see below).
 - Implement $\psi'(0) = 0$ for an even parity solution by setting $\psi_0 = \psi_{-1} = 0$.
 - Initialize variable `last_diverge` (which keeps track of the direction of the diverging trend) to zero (since we do not know this direction *a priori*). If the subsequent calculations to evaluate ψ for a given value of E exits by detecting a diverging trend, the sign of the last value of ψ evaluated indicates the direction.
 - Main loop in which we iterate and home-in to a valid eigenvalue E :
 - ▷ For the current value of E , loop through the index $i = 0, 1, \dots, N - 1$ to successively compute the wave function $\{\psi_i\}$.
 - Calculate ψ_{i+1} from ψ_i and ψ_{i-1} using
$$\psi_{i+1} = 2\psi_i - \psi_{i-1} - 2(E - V(x_i))(\Delta x)^2\psi_i.$$
 - Check if $|\psi_{i+1}| > b$ for a suitable cutoff value b .
 - ▷ If yes, assume that ψ is diverging. Exit the loop over i and display the computed ψ .
 - ▷ Otherwise, continue the loop with the next value i .
 - ▷ Display the computed ψ and current value of ΔE , and wait for key input to see if we should continue further.
 - If ΔE is small enough, we may conclude that the current E is acceptable (if the displayed ψ looks good), or reject this calculation (if it is not). Exit the subroutine.
 - If further iteration is indicated, continue.
 - ▷ If the last value of ψ evaluated ($\psi(i+1)$) and `last_diverge` are of different signs, turn around the direction of varying E and halve its magnitude by assigning $\Delta E = -\Delta E/2$.
 - ▷ Update the trial value of E to $E + \Delta E$.
 - ▷ Replace `last_diverge` by the sign of $\psi(i + 1)$.
 - ▷ Go back to the beginning of the homing-in process and iterate.
-

In the `calculate` routine it is convenient to employ a user-defined function¹¹ for $V(x)$. Using a custom function in this way makes `calculate` easier to read and understand and also makes it simpler to modify the program to deal with other potential functions without modifying any other parts of the program. The *cutoff* parameter b is used to judge whether $\psi(x)$ is diverging (with no hope of return); the appropriate value of b depends on the eigenvalue E as well as the type of potential $V(x)$. For the the square-well problem treated here and the choice of $\psi(0)$ (or $\psi'(0)$ for the odd solutions), a value of $b \approx 2$ works well, but you need to experiment with it for other situations.

Also to be noted is the interactive nature of this routine. When $\psi(x)$ for some trial value of E has been computed (or when $|\psi_{i+1}| > b$ for some i), it is useful to display the current $\psi(x)$ and asks the user whether to continue. This is an optimization problem (similar to the one we discussed in connection with estimating the launch angle that maximizes a cannon range in Chapter 2, for example), and a visual evaluation of how good (or bad) the current wave function is can be valuable, particularly at initial stages of exploring possible solutions. Most programming languages allow some kind of interactive polling from the keyboard. On the other hand, it is not hard to convert this program into an automatic mode, in which case one needs to set a stopping criterion; setting a limit on the magnitude of ΔE is usually a good approach.

Some results from this program are shown in Figure 10.4 where we plot ψ calculated for several values of E near the exact value for the lowest energy level, $E_G = 1.2337\dots$. Here we have taken $V = 0$ inside the box and $V = V_0 = 1000$ outside. This behavior is just like the schematic illustrations in Figure 10.3, with ψ diverging to either $+\infty$ or $-\infty$, depending on whether E is above or below the energy that best satisfies the boundary conditions. As the search routine approached this value, the boundary condition $\psi(L) = 0$ was satisfied with increasing precision and the divergence of ψ was pushed to larger and larger values of x . The last value of E considered here was 1.1790; decreasing it by only 0.0001 caused ψ to diverge to $-\infty$. Hence, these two values of E closely bracket the “best” numerical value. You may notice that this is a few percent lower than the exact value for a box with infinitely hard walls. We will see in a moment that this difference is due (largely) to the value used for V outside the box. The exact value for E_G quoted above assumes $V_0 = +\infty$, while the calculation here used $V_0 = 1000$.

This procedure can be employed to find the other even-parity levels by beginning the search at higher values of E . Finally, we should point out that the wave functions given in Figure 10.4 are not properly normalized. To do this would require that ψ be rescaled by a constant factor so as to satisfy (10.5). Note that this rescaling should be done using the wave function for $|x| \leq L$ from Figure 10.4 (that is, the diverging part outside the box should not be included). We will leave this task to the reader.

Next we consider the calculation of the odd-parity levels. We will leave this job mainly to the exercises, although we will show a few results here. As we

¹¹By this we mean a function that is constructed by the user as opposed to being built into the language (or its libraries)

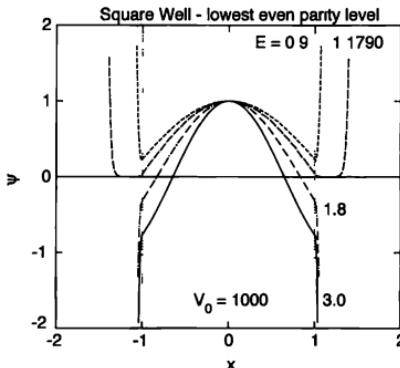


FIGURE 10.4: Trial ground-state wave functions calculated for a box, that is, a square well, with the shooting method for several values of the energy. The potential was $V = 0$ inside the box ($|x| \leq 1$) and $V = 1000$ outside. The dotted lines show the positions of the walls of the box. The spatial step size was $\Delta x = 0.01$. The calculation assumed $\psi(0) = 1$, so the wave functions are not normalized.

have already mentioned, the odd-parity wave functions are the solutions to (10.2) for which $\psi(+x) = -\psi(-x)$. Applying this condition at $x = 0$ requires that $\psi(0) = 0$. Since, as we have argued above, the overall magnitude of ψ at this stage is arbitrary,¹² we take $d\psi/dx = 1$ at $x = 0$. Hence, the calculate routine can again be used to obtain $\psi(x)$ by integration, the only difference being that we must employ the initial values $\psi_{-1} = -\Delta x$ and $\psi_0 = 0$. The rest of the calculation proceeds as before, as our routine hunts down the value of E that best satisfies the boundary conditions at $x = \pm L$. Some results for several different values of E are shown in Figure 10.5; the value we find for the energy of the lowest odd-parity level is ≈ 4.670 , which is about 5 percent below the exact value for a box with infinitely high walls ($\pi^2/2 = 4.934\dots$). As with the ground-state energy, this difference is due mainly to the fact that the potential outside the box is not infinitely large.

If you were to examine Figures 10.4 and 10.5 with very high resolution, you would notice that while the wave function at the walls of the box is small, it is not quite zero, even for the best values of E . You may recall that we obtained the boundary condition $\psi = 0$ at the walls of the box under the assumption that the potential is infinite outside the box. Actually, in our computations we have used a slightly different criteria for choosing the best wave function. This criteria involved the divergence of the wave function at large x and is a more general approach since it can be applied to a wide variety of potentials (as we will see in a moment). For a box with infinitely hard walls, that is, if the potential outside the box is truly infinite,

¹²We can once again adjust the magnitude of $\psi(x)$ at the end of the calculation so as to satisfy the normalization condition $\int \psi^* \psi dx = 1$

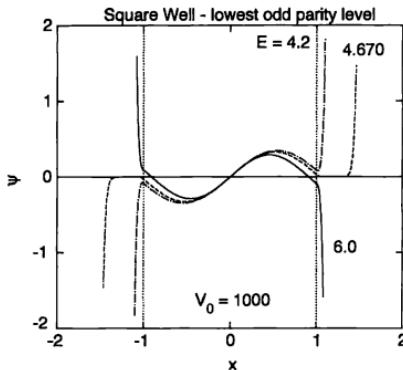


FIGURE 10.5: Trial wave functions for a square well with $V_0 = 1000$. All of the parameters are the same as in Figure 10.4, except that here we consider the lowest odd-parity level.

this divergence criteria is equivalent to the boundary condition that ψ vanish at the edges of the box. However, for a box with soft walls the wave function will be small, but nonzero, in the region beyond the walls. This is a well-known result of quantum theory, which is not found in classical physics.

According to classical theory, a particle cannot venture into a region where V is larger than its total energy. This would require that the classical kinetic energy be negative, which is not allowed. However, quantum theory places no such restriction on the particle. There is a small, but nonzero, chance of finding the particle *anywhere* in space so long as the potential energy is not infinite. This leads to a variety of interesting and important effects such as tunneling, which we will explore shortly. For the particle in a box problem it means that ψ need not go completely to zero at the walls of the box. It turns out that when $V > E$ the wave function decays exponentially with distance as we go farther and farther outside the box. This decay becomes more rapid as $V_{\text{outside}} \equiv V_0$ is made larger, as illustrated in Figure 10.6. Here we show the wave function for the lowest energy level with different values of V_0 . In all three cases E is within ± 0.0001 of the ground-state energy for that particular value of V_0 . We see that the magnitude of ψ at the wall increases as V_0 is reduced. In addition, the rate at which ψ diverges as $|x|$ is increased beyond the walls (at $x = \pm 1$) also increases greatly as V_0 is made larger. We will explore this behavior further in the exercises and in some other examples below.

The shooting method is a convenient way to deal with boundary-value problems in one dimension. However, it is limited to situations in which values of ψ and $d\psi/dx$ at some initial location can somehow be determined. In our particle-in-a-box problem the symmetry of the potential was essential for obtaining ψ and its deriva-

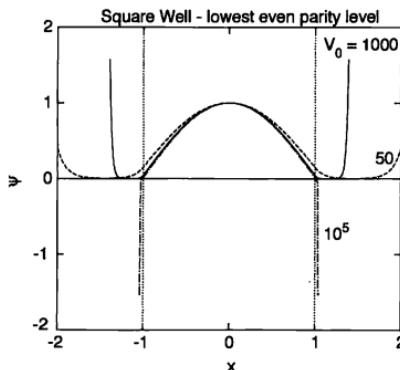


FIGURE 10.6: Ground-state wave function (unnormalized) calculated using the shooting method for a square well with different values of the potential outside the box, V_0 . The walls of the box are at $x = \pm 1$. The ground-state energies here are 1 019, 1 179, and 1 221 for $V_0 = 50$, 10^3 , and 10^5 . The spatial step size was $\Delta x = 0.01$.

tive at the center of the box. In situations involving a nonsymmetric potential, the wave functions will not have a definite parity; thus we cannot obtain initial values of ψ and $d\psi/dx$ from symmetry arguments. In such cases a different approach that relies on the continuity of ψ can be employed. Consider the nonsymmetric potential shown schematically in Figure 10.7. This has a form much like the Lennard-Jones potential we encountered in our molecular dynamics work in Chapter 9 and is similar to what we might find for a system involving two atoms, with one located¹³ at $x = 0$. Note, however, that in this example we restrict things to one dimension, so we are not yet treating the full three-dimensional Schrödinger equation. We expect on physical grounds to find a bound state in which the atom has a large probability of being found near the minimum of the potential. The wave function for this state will vanish at both large and small x and be significantly different from zero only in the region where V is large in magnitude and negative.

Let us therefore consider calculating ψ using (10.11), with integrations beginning in the regions where ψ is very small (i.e., at large x and small x) and proceeding toward the middle. That is, we now perform two integrations. One starts from a point on the far left in Figure 10.7, at the point $x = x_L$. At this location we set $\psi = 0$ and give the derivative $d\psi/dx$ a very small value. We then use (10.11) to obtain ψ at $x_L + \Delta x$, $x_L + 2\Delta x$, etc., and thus determine ψ at ever-increasing values of x . We call this function ψ_L , since it is obtained by integrating in from the

¹³For simplicity we can imagine that the atom at $x = 0$ is much more massive than the other so that it can be taken as fixed in space, and we need worry only about the wave function of the light atom.

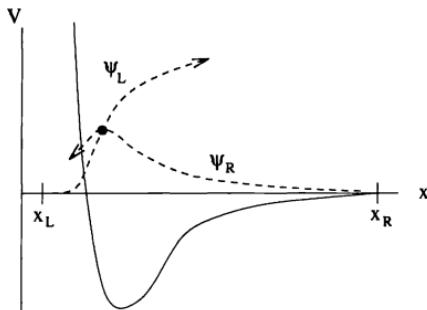


FIGURE 10.7: Schematic illustration of the matching method for solving the Schrödinger equation. The solid curve shows the potential, while the dashed curves show (hypothetical) wave functions calculated using (10.11), starting the integration from the left (ψ_L) and from the right (ψ_R). In this case the two trial functions do not match smoothly, and thus cannot be combined to give an acceptable solution of the Schrödinger equation.

left-most region. This procedure is then repeated starting from a point $x = x_R$, far to the right in Figure 10.7. At this location we set $\psi = 0$ and give $d\psi/dx$ a small value, in preparation for the application of (10.11). We then integrate toward smaller values of x , and obtain a function we call ψ_R . The plan is to combine the functions ψ_L and ψ_R to obtain the wave function over the entire range.

The correct wave function must be a smooth, continuous function. That is, ψ and its derivative must both be continuous.¹⁴ This means that if a combination of our two functions ψ_L and ψ_R is to form an acceptable wave function, they must intersect somewhere (so that the wave function is continuous), and they must have the same slope at the intersection point (so that $d\psi/dx$ is continuous). We can always arrange for ψ_L and ψ_R to intersect since, as has already been noted several times, we can scale the calculated ψ by a constant factor and still obtain a solution to the Schrödinger equation. It is usually convenient and numerically advisable, to pick a location somewhere near the minimum of the potential as the matching point since the wave function will be large there, and then scale either ψ_L or ψ_R or both so that their values are equal at that point.¹⁵ This still leaves us with the requirement that the derivatives be equal at the matching point. The only remaining adjustable parameter in the computation is the energy. For most values of E we find the situation depicted in Figure 10.7, where the derivatives $d\psi_L/dx$ and $d\psi_R/dx$ do not match. However, it is usually possible to adjust the value of E so as to make these derivatives match. When this occurs, the combination of ψ_L and ψ_R gives the wave function, and the associated value of E is the energy of the level.

This approach is often called the *matching method*, for obvious reasons. It

¹⁴This assumes that the potential is a smooth function.

¹⁵This is equivalent to scaling the derivatives at x_L and x_R so as to make $\psi_L = \psi_R$ at the matching point.

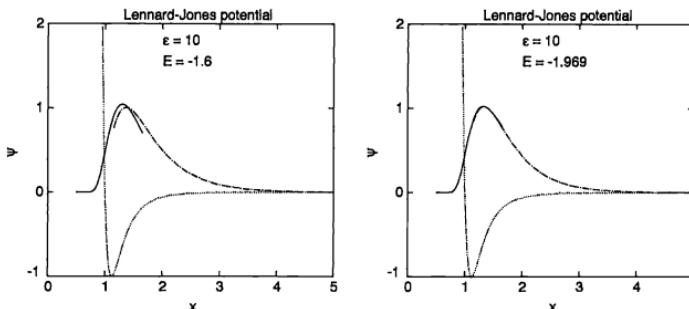


FIGURE 10.8: Solution for the ground state of the Lennard-Jones potential [(10.12) with $\epsilon = 10$ and $\sigma = 1$] constructed using the matching method. The integrations were started at $x_L = 0.5$ (on the left) and $x_R = 5$ (on the right) with $\Delta x = 0.01$. The two trial functions are shown by the solid (ψ_L) and dot-dashed (ψ_R) curves, and the potential is shown by the dotted curve (the scales for ψ and $V(x)$ are not the same). Left for $E = -1.6$ the derivatives of ψ_L and ψ_R are not equal at the matching point ($x = 1.4$). Right for $E = -1.969$ the derivatives match fairly well, so this is an acceptable approximation for the wave function. Note that this wave function is not normalized.

can be implemented numerically by modifying the `calculate` routine we developed for the shooting algorithm. A search routine can again be used to obtain an ever more accurate match of the derivatives $d\psi_L/dx$ and $d\psi_R/dx$, and thus better and better estimates for the wave function and energy. While we can use essentially the same structure as the `calculate` routine of the shooting method, the details differ considerably. First, since we are starting to compute $\psi(x)$ from both ends of the system, we must have initial values of the wave functions and their derivatives at both ends. That is, we need to start with $\psi_L(x_L - \Delta x)$, $\psi_L(x_L)$, $\psi_L(x_R + \Delta x)$, and $\psi_L(x_R)$, where the two values outside the interval (x_L, x_R) are needed to start the iteration of (10.11). These values are usually chosen so that the initial slopes are very small, e.g., $\psi_L(x_L - \Delta x) = -0.0001\Delta x$. For convenience we can choose the same value at both ends as they will be scaled differently to match the values of ψ_L and ψ_R at a central point anyway. Second and more importantly, we must have an efficient strategy to vary E so that the derivatives $d\psi_L/dx$ and $d\psi_R/dx$ approach each other. A valid approach here is to: (1) assign a value of ± 1 to a variable, say `match`, depending on whether $d\psi_L/dx > d\psi_R/dx$ or vice-versa, and (2) if the sign of `match` is different from the corresponding quantity from the previous trial value of E , then (3) assign the new value $\Delta E = -\Delta E/2$ to the increment to use next. We leave it to the reader to confirm that this method works regardless of the signs of the derivatives and to construct a working program based on this strategy! (of course, you are also welcome to design your own strategy!).

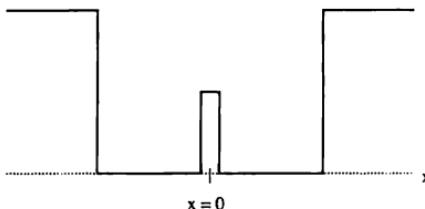


FIGURE 10.9: Two potential wells separated by a narrow "tunnel" barrier

Figure 10.8 shows results for the one-dimensional Lennard-Jones problem¹⁶

$$V(x) = 4 \epsilon \left[\left(\frac{\sigma}{x} \right)^{12} - \left(\frac{\sigma}{x} \right)^6 \right] \quad (10.12)$$

Here we plot results for two values of E . In each case we have normalized both ψ_L and ψ_R to unity at the matching point $x_{\text{match}} = 1.4$. For the case on the left there was a sizable discontinuity in $d\psi/dx$ at the matching point, so this solution is not satisfactory. After letting the search routine iterate several times, we obtain the result on the right, where ψ_L and ψ_R join much more smoothly. By starting the search at other values of E we can obtain the wave functions for other energy levels. We will, as usual, leave this for the exercises.

While we introduced the matching strategy for dealing with nonsymmetric potentials, it is not limited to such cases. An interesting example is a square well containing a small potential barrier inside the well, shown schematically in Figure 10.9. This particular potential is actually symmetric, so we could use the shooting method here, starting from $x = 0$. However, there are many interesting variations on this problem; for example, we could move the small barrier off center or make the potential of the barrier vary with x . Such asymmetric potentials require the matching approach.

Before we construct the wave functions for the potential in Figure 10.9, it is worthwhile to anticipate some of the physics of the problem. This potential consists of two square wells, one to the left and one to the right of the barrier. If the barrier is large (very high or very wide or both), the two wells are effectively uncoupled, since the wave function in one will not penetrate far enough into the barrier to reach into the other well (compare with Figure 10.6). The wave functions in each well will therefore be unaffected by the presence of the other, and the solutions will be those we have already encountered in our work on the particle-in-a-box problem. The ground-state wave function for this case is shown schematically on the left side of Figure 10.10.

If, on the other hand, the barrier separating the two wells is not large, the unperturbed wave function from one side will be able to penetrate through to the

¹⁶You may recall that this is the potential we used in our molecular-dynamics calculations in Chapter 9

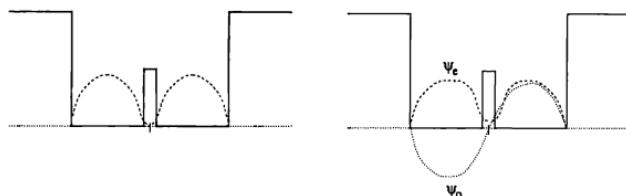


FIGURE 10.10: Left the dashed curves showing the approximate unperturbed wave functions in the two nearby potential wells, right wave functions (very schematic!) after we allow for mixing of the unperturbed wave functions

other before it decays to a negligibly small value. Actually, the unperturbed wave functions from both wells will decay as they penetrate the barrier from opposite sides. These two wave functions must meet continuously in the center, just as we saw with ψ_L and ψ_R in the Lennard-Jones problem (Figure 10.8). This leads to what is usually referred to as *mixing* of the two, initial, unperturbed wave functions. If we denote the unperturbed wave functions as ψ_1 and ψ_2 , then to a first approximation they will combine to form wave functions for the entire two-well system as either $\psi_e \sim \psi_1 + \psi_2$, or $\psi_o \sim \psi_1 - \psi_2$. That is, they can form either an even (ψ_e) or odd (ψ_o) combination.¹⁷ These are sketched on the right side of Figure 10.10. Note that the wave functions we have sketched are derived (approximately) from the initial unperturbed wave functions in the two separate wells. Similar combinations are formed from the unperturbed excited-state wave functions.

The two wave functions ψ_e and ψ_o will have energies close to that of the initial unperturbed states. According to perturbation theory, the mixing of two such states leads to a small splitting of the energy levels, with the even level (ψ_e) being pushed to a lower energy and the odd level pushed higher. This can be illustrated nicely using the solutions obtained from the matching algorithm. The ground state has even parity and is considered in Figure 10.11. On the left we show the behavior for a value of E that is too low, and we see that ψ_L and ψ_R do not match smoothly. In the middle case E is slightly too large, although the derivatives of ψ_L and ψ_R match fairly well on the scale shown here. On the right we show results for a value of E that yields a slightly better match. It is interesting that even here the wave function is not quite symmetric; you can see this from the fact that the amplitudes of the two maxima are not quite the same. This is a result of the numerical errors that arise when integrating through a classically forbidden region (the tunnel barrier), as the solutions tend to be divergent in such regions (compare again with Figure 10.6). In this particular problem we could have obtained a better result by employing the shooting method, as this would preserve the symmetry inherent in the problem. However, using the matching method in this case gives a good impression of the strengths and weaknesses of that algorithm.

¹⁷The precise parity of the solutions in this case can be traced to the symmetry of the potential. When the potential is not symmetric the solutions will not have a definite parity, but there will still be two wave functions constructed (approximately) as the sum and difference of the two unperturbed wave functions

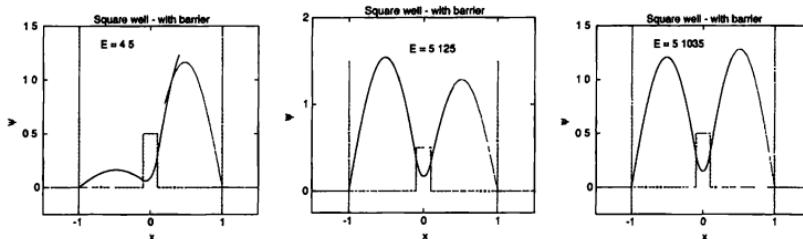


FIGURE 10.11: Results for ψ_L and ψ_R calculated with the matching method for the potential in Figure 10.9, for several values of E . The integrations were started at $x = \pm 1.3$, the step size was $\Delta x = 5 \times 10^{-4}$, and the matching was done at $x_{\text{match}} = 0.3$. The parameters associated with the potential were $V = 0$ inside each well, $V = 10^5$ for $|x| > 1$, and $V = 100$ for $|x| \leq 0.1$. The vertical dashed lines indicate the edges of the potential wells. Note that the scale for ψ varies from plot to plot.

The numerical difficulties associated with integrating through a classically forbidden region (where $V > E$) bring up a related issue. In all of our applications of the matching method we have started the integrations in such forbidden regions and integrated toward the regions where $V < E$, that is, the regions where we expect the wave function to be large. In principle we could have started at a location where $V < E$, and integrated toward the classically forbidden regions. However, the numerical problems and divergences are usually more severe in the latter case. As a general rule it is best to integrate away from classically forbidden regions, toward regions where ψ is large.

Figure 10.12 shows results for slightly higher values of E . The left and middle plots show behavior for values of E that are too low and too high and thus provide poor matches. The plot on the right shows a good match, and the wave function is seen to have odd parity, as expected from the discussion given above.

The shooting and matching methods described in this section are well suited for the time-independent Schrödinger equation in one dimension and similar types of boundary-value problems. We have only been able to consider a few examples here; the exercises explore these and several other cases in more detail.

EXERCISES

- 10.1. Use the shooting method to obtain the lowest six energy levels for a square well. This will require that you write a program that can deal with both even- and odd-parity solutions. Examine how the energy varies with level number and test your values against the exact results (10.7) and (10.8). Compare the wave functions (make sure that they are properly normalized!) with the exact solution (10.6). It is also instructive to examine the behavior as a function of the spatial step size Δx . Study how the value of the ground-state energy varies with the choice of step size; try $\Delta x = 0.05, 0.01$, and 0.005 , for a box with walls at $x = \pm 1$.
- 10.2. Use the matching method to calculate the wave functions and energies of the

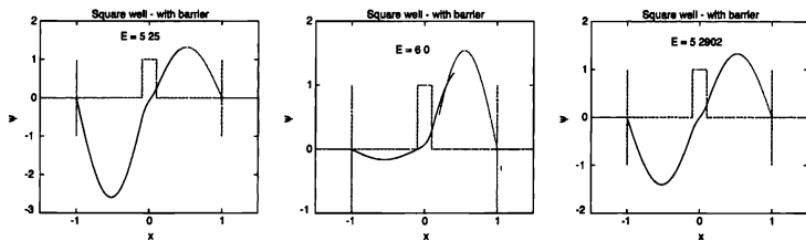


FIGURE 10.12: Results for ψ_L and ψ_R calculated with the matching method for the tunnel potential in Figure 10.9. Here we have used several values of E that are near the energy of the lowest odd-parity level. The other parameters were the same as in Figure 10.11.

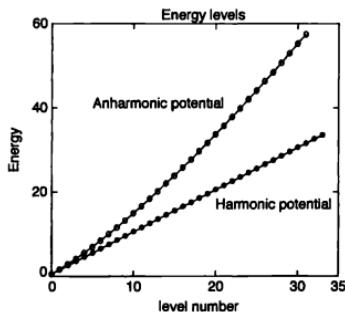


FIGURE 10.13: Lowest few energy levels for a harmonic oscillator ($V = Kx^2/2$ with $K = 1$) and an anharmonic oscillator [$V = (k_1x^2 + k_2x^4)/2$, with $k_1 = 1$ and $k_2 = 0.1$]. In both cases we have used our usual choice of units, for which $\hbar = m = 1$. Note that for the harmonic case the energy varies linearly with level number (which agrees with the well-known, analytic solution). For the anharmonic case the corresponding energies are shifted to progressively higher values. For the higher levels the wave function has more of its “weight” at larger values of x (classically speaking, the vibration amplitude is higher) where the potential is larger due to the anharmonic term, and this pushes the energy higher than in the harmonic case.

- first few energy levels of the harmonic oscillator potential $V = Kx^2/2$. Show that the levels are evenly spaced in energy and compare with the exact result $E = (n + \frac{1}{2})\hbar\omega$, where n is an integer (0, 1, 2, ...) and $\omega = \sqrt{K/m}$ (see Figure 10.13). It is also worthwhile to compare the wave functions with the exact results.
- 10.3. Use either the shooting or matching method to calculate the energy levels for the potential $V = x^n$ for different values of n . As n becomes very large this approximates an infinite square well.
- 10.4. Calculate the first few energy levels and the associated wave functions for a potential of the form $V = (k_1x^2 + k_2x^4)/2$. For small x the first (k_1) term dominates, and the behavior is close to that found for the harmonic oscillator, while for large x the second (k_2) term dominates and the behavior is *anharmonic*. Some typical results for the energy levels in this case are given in Figure 10.13.
- 10.5. Use the shooting method to study how the wave function for a particle-in-a-box depends on the magnitude of the potential outside the box V_0 . Examine the variation of ψ beyond the walls of the box and show that it decays exponentially with distance in this region. Study the decay length as a function of V_0 and compare the results for different energy levels. As the energy of the level approaches V_0 the decay length should become larger.
- *10.6. Use the matching method to obtain the wave functions for the first few states above the ground level for the one-dimensional Lennard-Jones potential. Compare the variation of the energy with level number with the results shown for the harmonic and anharmonic potentials in Figure 10.13. Depending on the number of levels you wish to consider, you may have to make the value of ϵ in (10.12) larger than the value used in Figure 10.8. A deeper potential well possesses more bound states.
- *10.7. Many three-dimensional quantum mechanics problems can be reduced effectively to one- or two-dimensional problems. For example, the hydrogen atom is three dimensional as it involves the wave function of an electron as a function of position relative to a (fixed) proton. However, the spherical symmetry makes it possible to write the wave function in the form

$$\psi(r, \theta, \phi) = f(r) Y(\theta, \phi), \quad (10.13)$$

where r is the (radial) distance between the proton and the electron, and θ and ϕ are the usual spherical coordinates [see Griffiths (1995) and Schiff (1968) for discussions of (10.13) and the derivation of (10.14)]. Here $f(r)$ is a function of r only, and all of the angular dependence is contained in the function $Y(\theta, \phi)$. As you may know, these angular functions turn out to be the spherical harmonics. In this problem we will be concerned with the radial function, $f(r)$. When the form (10.13) is inserted into the three-dimensional Schrödinger equation we obtain a differential equation for $f(r)$. It is common practice at this point to deal instead with the related function $g(r) \equiv rf(r)$, which satisfies the equation

$$\frac{d^2g}{dr^2} = \left(2[V(r) - E] + \frac{\ell(\ell+1)}{r^2} \right) g, \quad (10.14)$$

where $V(r) = -1/r$ is the Coulomb potential¹⁸ and ℓ is an integer associated with the angular momentum of the electron that arises from the solution for

¹⁸More precisely, this is the electric potential multiplied by the charge of the electron. We use units in which this charge is -1 .

$Y(\theta, \phi)$. Employ the shooting method to solve (10.14) for $g(r)$ starting your integration at $r = 0$, with the initial value $g(0) = 0$. Consider the cases $\ell = 0$ and $\ell = 1$ and compare your results with the analytic solution for the hydrogen atom. Hint: In the units used here the energies of the bound states are given by $-1/2n^2$, where n is an integer.

10.3 A MATRIX APPROACH

The methods developed in Section 10.2 can handle most one-dimensional quantum mechanics problems. However, life is more difficult in two or three dimensions for the following reason. As we have already seen, ψ must, in general, satisfy certain boundary conditions. In one dimension these usually concern the value of ψ at two locations; typically at the two extremes of a region. In higher dimensions we must worry about the value of the wave function on a *surface*, which could be a circle in two dimensions or a sphere in three dimensions. In some cases the potential may have enough symmetry that the problem can be reduced to a differential equation involving only one variable. For example, we have seen in the exercises in Section 10.2 that with the hydrogen atom we can separate the radial and angular dependences of ψ and thereby reduce the radial dependence to a one-dimensional problem. In such cases the matching or shooting method can then be used. Many textbook problems are like this, for the simple reason that more complicated cases can almost never be solved analytically!

Shooting and matching methods are not well suited for satisfying boundary conditions at more than one point and therefore cannot (easily) be used in two or three dimensions. A very general method for dealing with these cases is to write the Schrödinger equation in the form of a matrix eigenvalue problem. While such an approach can be computationally very difficult to carry through, it is useful for illustrating several important points.

Consider a two-dimensional problem and assume that, as usual, we break up space into a lattice of points, with $\psi(m, n)$ being the value of the wave function at location (m, n) on the lattice. The boundary conditions then specify the value of ψ on the boundary. For example, ψ might vanish on the edges of the system; if the system is rectangular, we would have $\psi(0, n) = \psi(M, n) = \psi(m, 0) = \psi(m, N) = 0$, where the lattice runs from $m = 0$ to M and $n = 0$ to N . The Schrödinger equation (10.1) then has the form

$$\begin{aligned} -\frac{\hbar^2}{2m} \nabla^2 \psi + V(\vec{r}) \psi &= -\frac{\hbar^2}{2m} \left[\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} \right] + V(\vec{r}) \psi \\ &\approx -\frac{\hbar^2}{2m} \left[\frac{\psi(m+1, n) + \psi(m-1, n) - 2\psi(m, n)}{(\Delta x)^2} \right. \\ &\quad \left. + \frac{\psi(m, n+1) + \psi(m, n-1) - 2\psi(m, n)}{(\Delta y)^2} \right] + V(m, n) \psi(m, n) \\ &\approx E \psi(m, n), \end{aligned} \quad (10.15)$$

where the last several lines give the wave equation in finite-difference form. This is a system of algebraic equations for $\psi(m, n)$; there are (of order) $M \times N$ unknown values of $\psi(m, n)$ and the same number of equations.

If we define a column vector $\vec{\psi}$ whose elements are $\psi(m, n)$,

$$\vec{\psi} = \begin{pmatrix} \psi(0, 0) \\ \psi(0, 1) \\ \vdots \end{pmatrix}, \quad (10.16)$$

and a square matrix \mathcal{H} representing the Hamiltonian *operator*,

$$\mathcal{H} \equiv -\frac{\hbar^2}{2m} \nabla^2 + V(\vec{r}), \quad (10.17)$$

then these equations can be written as a linear equation

$$\mathcal{H} \vec{\psi} = E \vec{\psi}, \quad (10.18)$$

where the Hamiltonian \mathcal{H} is be represented by a matrix

$$\mathcal{H} = \begin{pmatrix} 4 + V(0, 0) & -1 & 0 & \dots \\ -1 & 4 + V(0, 1) & -1 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}. \quad (10.19)$$

Here we have taken the liberty to set $\Delta x = \Delta y$ and to absorb factors such as these as well as \hbar and mass m into V and E .¹⁹ The points we wish to make here depend on the general *form* of these equations, so we will avoid the notational distractions associated with specifying all of the factors precisely.

Equations (10.18) and (10.19) define an eigenvalue problem for the Hamiltonian matrix \mathcal{H} , which is one of the standard problems in linear algebra.²⁰ Efficient programs for obtaining the eigenvalues E and the associated eigenvectors $\vec{\psi}$ can be found in all serious packages of mathematical routines. Moreover, since \mathcal{H} is usually *sparse*, that is, most of its entries are zeroes, a number of techniques are available to simplify such work²¹

Focusing on the eigenvalues, these equations are sometimes put in the form

$$\begin{pmatrix} 4 + V(0, 0) - E & -1 & 0 & \dots \\ -1 & 4 + V(0, 1) - E & -1 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \psi(0, 0) \\ \psi(0, 1) \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \end{pmatrix}. \quad (10.20)$$

If we define a square matrix $\mathcal{M}(E)$ whose elements are shown as the left-most matrix in (10.20), this can be written as

$$\mathcal{M}(E) \vec{\psi} = \vec{0}, \quad (10.21)$$

¹⁹This procedure is equivalent to setting $\Delta x = \Delta y = 1$, $\hbar = 1$, and $m = 1/2$.

²⁰The reader may note the similarities between this problem and the one encountered in Chapter 7 where we discussed the problem of diffusion on a disordered network as well as the one we will treat in Chapter 11 concerning the vibrations of a membrane. More on the numerical aspects of the linear problems are given in Appendix H

²¹Nevertheless, computing the eigenvalues and eigenvectors of large matrices is usually a formidable task, which can require large amounts of computer time as we will detail later.

where $\vec{0}$ is a column vector whose elements are all zero. Here we have indicated explicitly the fact that the matrix $\mathcal{M}(E)$ is a function of E . In general there will be MN values of E for which this equation has a nontrivial solution.²² Formally, these eigenvalues are the roots of the algebraic equation

$$\det[\mathcal{M}(E)] = 0, \quad (10.22)$$

called the *secular equation*, where the function $\det[]$ denotes the determinant of a matrix. Each eigenvalue has an associated eigenvector, which is the wave function for that particular energy level.

This matrix approach is completely general and can be very useful, especially in formal arguments. However, it is often a very large computational task unless the Hamiltonian matrix has a special form that reduces the computational requirements. Since the column or row reduction of the determinant would require of the order of $(MN)!$ operations, a naive estimate of the number of required operations might appear to be an exponential function of the matrix size. Fortunately, there are efficient methods available to reduce it to $O([MN]^3)$ (or even lower if the matrix is of a special form such as *tridiagonal*). Sometimes, an approximate solution is sufficient, allowing us to substitute a matrix of a special form (and often a much smaller one as well), saving a large amount of computing resources. However, in general we are still faced with large computational requirements. For example, if there are 50 grid elements along both m and n in a two-dimensional problem, one must deal with a 2500×2500 matrix. In three dimensions the matrix would be $125,000 \times 125,000$. While such eigenvalue problems arise in many applications, and efficient programs for carrying out these computations do exist, computing the eigenvalues and eigenvectors of such large matrices is still a formidable task.²³

We end this section by describing one particularly effective use of the matrix method. This is sometimes called the *power method*, one in which we do *not* explicitly solve for the eigenvalue or the eigenvector, but rely on the repeated application of the $n \times n$ matrix (say, \mathbf{W}) on an arbitrary starting trial vector \vec{u} to weed out the contributions to \vec{u} from all but the eigenvector \vec{u}_1 with the largest (i.e., “dominant”) eigenvalue λ_1 . First, let us assume that the eigenvalues are ordered so that $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots \geq |\lambda_n|$, and assume also that the initial state \vec{u} can be represented as a linear combination²⁴

$$\vec{u} = \sum_{i=1}^n a_i \vec{u}_i, \quad (10.23)$$

where $a_1 \neq 0$ and \vec{u}_i are the eigenvectors with eigenvalues λ_i , so that

$$\mathbf{W} \vec{u}_i = \lambda_i \vec{u}_i. \quad (10.24)$$

²²A trivial solution is one for which all of the $\psi(m, n)$ are zero. Note also that because of the boundary conditions on $\vec{\psi}$, the number of solutions may be less than MN .

²³However, it can be quite useful for quantum problems in which, for some reason, the size of the $\vec{\psi}$ vector (and hence the size of the Hamiltonian matrix) is not too large. Cases of this kind arise, for example, in magnetic problems in which the number of spin (or angular momentum) states is often of order 10 or less.

²⁴This is known to be possible for Hermitian matrices such as those representing a physical Hamiltonian.

Applying \mathbf{W} on \vec{u} then gives

$$\mathbf{W} \vec{u} = \sum_{i=1}^n a_i \lambda_i \vec{u}_i , \quad (10.25)$$

and doing so k consecutive times gives

$$\mathbf{W}^k \vec{u} = \sum_{i=1}^n a_i \lambda_i^k \vec{u}_i = \lambda_1^k \left[a_1 \vec{u}_1 + \sum_{i=2}^n a_i (\lambda_i / \lambda_1)^k \vec{u}_i \right] . \quad (10.26)$$

Since $|\lambda_i / \lambda_1| < 1$ for $i \neq 1$, all but the first term in the square brackets will decrease in magnitude relative to the first term as $k \rightarrow \infty$. Thus, for large k we eventually get

$$\mathbf{W}^k \vec{u} \approx a_1 \lambda_1^k \vec{u}_1 , \quad (10.27)$$

and hence \vec{u}_1 can be obtained by normalizing $\mathbf{W}^k \vec{u}$.

While this method targets the dominant eigenvalue, there are various extensions which are able to find states other than the one with the largest eigenvalue. For example, if we know that there is an upper bound Λ for the magnitudes of the eigenvalues, we can apply the power method to the modified (and shifted) matrix $\Lambda^2 \mathbf{I} - \mathbf{W}^2$ and obtain the eigenstate with the eigenvalue of the smallest magnitude (say, the ground state wave function). Alternatively, we can also consider the application of the power method with \mathbf{W}^{-1} if we are interested in the ground state. It turns out that this can be accomplished without actually inverting the matrix first. Similarly, it is possible to use the combinations of shifting and inversion to look for eigenstates with eigenvalues in a neighborhood of values of our choice. Moreover, once an eigenstate and its eigenvalue are obtained by such a method, it is possible to iteratively project out the linear space associated with such states and look for the next most dominant one, and so forth. We will not dwell any further on this method, but it can be a very effective way to obtain a specific state, such as the one with the largest (or the smallest) eigenvalue.

10.4 A VARIATIONAL APPROACH

The computational difficulties with a “direct” matrix approach to eigenvalue problems have motivated the development of a number of alternative schemes. We do not have space to discuss all of them here; the interested reader can explore some of them through the references. In this section we will introduce one of these schemes to give an example of how the Schrödinger equation can be attacked. The method we now describe involves a Monte Carlo approach, similar to that described in Chapter 8, coupled with the variational principle.

The variational principle of quantum mechanics can be expressed as follows. Let φ be a “trial” wave function. We imagine that we are searching for the ground-state solution to the Schrödinger equation for a particular problem and that φ is a proposed wave function, or perhaps our best numerical guess for the true wave function. We can calculate the “energy” corresponding to this trial function

$$E^* \equiv \frac{\int \varphi^* \mathcal{H} \varphi d\vec{r}}{\int \varphi^* \varphi d\vec{r}} , \quad (10.28)$$

where the integrations are over all space and \mathcal{H} is the Hamiltonian operator. Note that we have given this effective energy the symbol E^* to emphasize the fact that since φ is not necessarily a solution of (10.17) (it was only a “proposed” wave function), E^* is not necessarily the energy of a true eigenstate.

According to the variational principle, the true ground-state energy is *always* less than or equal to E^* . In fact, the only time they are equal is when the trial wave function is precisely the true ground-state wave function. For *all* other functions φ the effective energy E^* will be higher than the ground-state energy. This principle allows us to recast the problem of solving the Schrödinger equation as a minimization problem. The goal is to find the function φ which minimizes E^* . There are several ways to implement this strategy, corresponding to different ways of searching through the space of all possible wave functions. The approach we describe below is very general, yet also somewhat crude. After seeing how it works, we will consider improved ways of constructing φ and performing the search procedure, some of which will be explored in the exercises.

To be specific we consider again the problem of finding the ground state of the one-dimensional Lennard-Jones potential. We dealt with this problem earlier with the matching method, so we already have a solution with which to compare it. To implement a minimization strategy we must first choose an initial trial wave function, which we will call φ^0 . As usual, we discretize space into bins Δx in size and let φ_n^0 be the value of the initial trial function at $x = n\Delta x$. We also restrict the domain²⁵ of our function to the range $x = 0.7 - 5$ (in reduced units), since we know on physical grounds that the ground state will be one in which the particle is localized near the minimum in the potential, $x_{\min} \sim 1.1$. Hence, our bins run from $n = 0$ corresponding to $x = 0.7$, up to $n = N$ corresponding to $x = 5$. We choose an extremely simple initial trial function: $\varphi_n^0 = \text{constant}$ for $1 \leq x \leq 4$ and zero outside this range, as shown in Figure 10.14. The constant is determined by requiring that our function satisfy the normalization condition $\int \varphi^* \varphi \, dx = 1$. This is certainly a poor approximation to the true ground-state wave function, so if the method can succeed with this choice of φ^0 it can probably be successful with any choice!

The only remaining part of the algorithm that must be specified is the procedure for how to use φ^0 (or any other trial function) to obtain an improved trial function. We will employ a Monte Carlo approach, which operates as follows. First, we pick one of the bins along x at random by choosing an integer in the range $0-N$; we will call this bin *ntest*. Second, we construct a provisional function by changing the value of the trial function at that location, φ_{ntest}^0 , by an amount chosen randomly in the range $\pm d\varphi$. Third, we calculate E^* for this provisional function. If it is lower than the corresponding value for φ^0 , the provisional function must be “closer” to the true wave function, and we use it as the new trial function. If E^* for the provisional function is higher than the value for φ^0 , we discard the provisional function and keep φ^0 as the trial function.²⁶

²⁵We will take $\sigma = 1$ and $\epsilon = 10$ in (10.12), so that the potential is the same as that considered in Figure 10.8.

²⁶Unlike the Monte Carlo method employed in Chapter 8, the algorithm described here does not involve a temperature parameter to enable changes that increase the energy. However, it

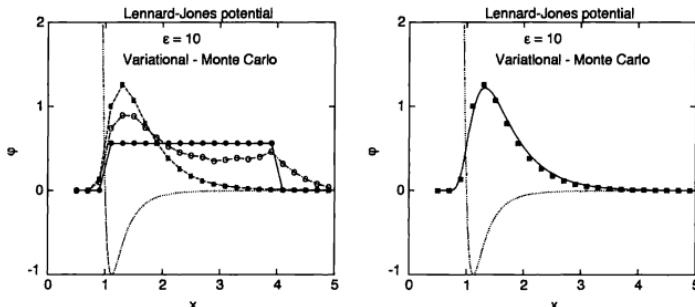


FIGURE 10.14: Left trial wave functions, φ , during the course of a variational-Monte Carlo calculation for the Lennard-Jones problem. The filled circles show the initial trial function, φ^0 , the open circles show the trial function after 10^3 attempted Monte Carlo moves (i.e., changes in the wave function), and the squares show it after 10^4 attempted moves. E^* for these trial wave functions was 0.11, -1.43 , and -2.18 , respectively. Right comparison of the final variational wave function with the results of the matching method (Figure 10.8). The filled squares here are the best trial functions from the plot on the left. Note that both of these wave functions have been properly normalized and that the potential (the dotted curves) is not shown to scale.

We will leave the construction of a variational-Monte Carlo program to the exercises (a full listing of such a program is given at this book's Web site). All of the key ingredients of such a program have been described above. We will mention only one programming subtlety here. You must decide at the outset whether or not the trial functions are to be properly normalized (and if so, when to do it). This arises when evaluating E^* , since if φ is normalized the denominator in (10.28) is unity and thus need not be computed. For this reason it is convenient to always normalize φ . However, if you perform this normalization after each tentative Monte Carlo modification of the trial function, you must make sure that you can return to the previous trial function if the Monte Carlo modification is rejected.

As the Monte Carlo process is repeated many times, both the trial function and the corresponding value of E^* should converge to those of the true ground state. Some results for the trial function for the Lennard-Jones problem at various stages of the calculation are shown on the left in Figure 10.14. After a relatively small number of Monte Carlo iterations (10^3 in Figure 10.14), φ exhibits fluctuations that arise because this function was constructed from a small collection of random changes. However, after a large number of Monte Carlo adjustments φ becomes a smooth function of x , as we would expect for the true wave function. The final trial

is often useful to allow such energy increases with some probability *temporarily*, reducing such probability gradually as the procedure goes forward and the energy approaches the ground state energy. This is called *simulated annealing* and is described in Appendix B. Strictly prohibiting energy increases would correspond to quenching the system instantly to zero temperature, while simulated annealing corresponds to a more gradual decrease in temperature.

function shown here had an energy $E^* = -2.18$, which agrees with the value found with the matching method (Figure 10.8) to within about 10 percent. The trial function itself also agrees well with the wave function calculated with the matching method, as shown on the right in Figure 10.14. There are some slight differences and we will show in the exercises that they are due to the finite grid size used in describing the variational wave function.

The variational-Monte Carlo method can thus deal effectively with one dimensional situations such as the Lennard-Jones problem. However, its real strength is that it can be applied with very little additional effort to treat two- and three-dimensional cases. To illustrate this we now consider a particle confined by a two-dimensional harmonic potential. The Schrödinger equation in this case is (10.1) with the potential

$$V(x, y) = \frac{1}{2} k_x x^2 + \frac{1}{2} k_y y^2, \quad (10.29)$$

where k_x and k_y are related to the curvatures of the potential in the x and y directions. The astute reader will notice that this problem can be separated into two independent one-dimensional harmonic oscillators, one along x and one along y , which can be solved analytically. We will use this exact solution to evaluate the usefulness of the variational-Monte Carlo method. Several problems for which there are no known exact analytic solutions will be considered in the exercises.

To implement the variational approach for the two-dimensional oscillator we discretize space along the x and y directions into segments of length Δx and Δy , and write our trial functions as $\varphi(m, n)$, with $x = m\Delta x$, and $y = n\Delta x$ (for convenience we take $\Delta y = \Delta x$). We also restrict x and y to the range $-N\Delta x \leq x \leq N\Delta x$ and $-N\Delta x \leq y \leq N\Delta x$, so that we have $(2N+1)^2$ bins altogether. For this calculation we again pick an extremely simple initial trial wave function, $\varphi^0 = \text{constant}$ for all points in the region $|x| \leq 1.6$ and $|y| \leq 1.6$, and zero otherwise. This trial function is shown in Figure 10.15.

The Monte Carlo algorithm can then be implemented as in our previous example. During each Monte Carlo step a grid location ($m_{\text{test}}, n_{\text{test}}$) is chosen at random, and a provisional function obtained by changing the initial trial function $\varphi^0(m_{\text{test}}, n_{\text{test}})$ by a small amount chosen randomly from the range $\pm d\varphi$. If E^* for this provisional function is lower than that of φ^0 , then the provisional function becomes the new trial wave function. If E^* is higher, we keep φ^0 as the trial function. Some results are given in Figure 10.15, where on the left we show the trial function as the calculation proceeds. The final trial function has a roughly Gaussian shape, which may be familiar to you from the analytic solution of the harmonic oscillator. Figure 10.15 also shows the behavior of E^* during the calculation. It is a monotonically decreasing function, as required by the Monte Carlo procedure. After many Monte Carlo time steps it approaches a constant, which signals the convergence of the algorithm.

A plot of the final trial wave function as a function of both x and y is shown in Figure 10.16. The wave function falls off more slowly in the x direction than along y , since in this example we have chosen $k_x < k_y$. This means that the "spring constant" for vibrations along x was softer than the one along y , so the effective amplitude of vibration was larger along x . The value of E^* for this trial function

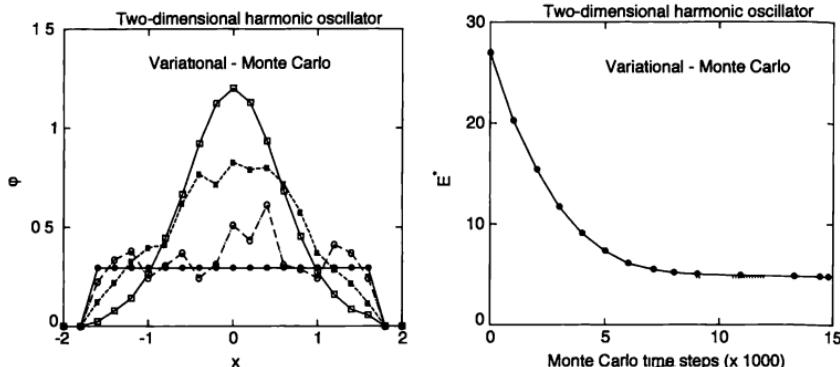


FIGURE 10.15: Left: trial wave functions plotted as $\varphi(x, 0)$ as a function of x , at various stages in the variational-Monte Carlo procedure. The filled circles show the initial trial function, and the open circles, filled squares, and open squares show φ after 2000, 5000, and 15,000 Monte Carlo time steps. Right: variation of E^* as the calculation proceeds. The potential energy was (10.29) with $k_x = 10$ and $k_y = 10$, the maximum values of $|x|$ and $|y|$ were 2.0 (so that $\varphi = 0$ outside this region), and the grid size was $\Delta x = \Delta y = 0.2$.

is ~ 4.79 , which differs from the exact value $4.743\dots$ by about 1 percent. We will see in the exercises that this small difference is due to the finite grid size.

While the specific variational-Monte Carlo approach we have described here is fairly well suited for dealing with two- and three-dimensional problems, it is worth considering how the method can be improved. One difficulty is connected with the number of elements of the trial wave function $\varphi_{m,n}$. For the two-dimensional harmonic oscillator example considered above there were ~ 400 grid elements. In order for the Monte Carlo strategy to be successful, it is necessary for each of these $\varphi(m,n)$ to be adjusted many times, and if the number of grid elements is large this can require a long computation. This difficulty can be dealt with in several ways.

First, it pays to make a good choice of the initial trial function, as then an accurate estimate for the true wave function can be reached with fewer Monte Carlo iterations. Here “good” means that φ^0 should have a form close to the true ground-state wave function. We didn’t attempt to do this in our calculations here, but it is usually possible to make an informed choice for φ^0 . In the exercises we will explore how this can speed up convergence.

Second, in our oscillator example we chose the ranges of x and y to be large enough that the boundaries were far from the region where the wave function was nonzero. This was by design, since ideally the boundary condition $\varphi = 0$ should be applied at $x, y = \pm\infty$. However, this means that there will be many grid elements where φ is extremely small and it would be nice to arrange for the Monte Carlo algorithm to spend less time updating $\varphi(m,n)$ in these regions, since they

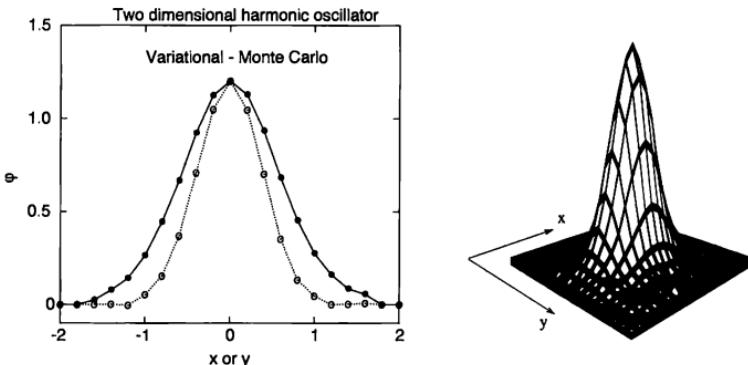


FIGURE 10.16: Left: plot of $\varphi(x, 0)$ as a function of x (filled circles) and $\varphi(0, y)$ as a function of y (open circles) for the final trial wave function from Figure 10.15. Right: three-dimensional plot of the wave function

are not as important as the regions where the wave function is large. This can be accomplished by modifying the way the test locations for the Monte Carlo changes are chosen. In our discussion above we chose the grid element to be considered in the variational procedure, mtest and ntest, at random from the *entire* allowed range of m and n . It would be better to choose the test point with a probability proportional to the value of φ at that point, since then the points where φ is largest would get the most attention from the algorithm. To do this requires that we be able to generate random numbers with a specified distribution. This is a problem that is discussed in Appendix F, where we consider two ways to generate random numbers with a nonuniform distribution. In the problem we are now considering the rejection method is most useful since we do not have an analytic expression for the desired probability function. The method is described in Appendix F, so we will not go into much detail here. A routine that generates mtest and ntest with a probability proportional to $\varphi(m, n)$ is sketched below.

EXAMPLE 10.2 Routine calc_nxy to generate random integers according to a specified weighting function

- Assume $\varphi_{max} \equiv \varphi(0, 0)$ is the largest value of the trial function. This value is used to normalize $\varphi(x, y)$.
- Generate a point (x, y) randomly.
 - ▷ Generate two uniform random numbers r_1 and r_2 between 0 and 1

- ▷ Convert r_1 to a random integer x in the range $(-N, N)$ (use $x = \text{int}(r_1 \times (2N + 1)) - N$ where int means truncation to the integer part).
 - ▷ Do the same for r_2 to get a random integer y
 - Generate a third uniform random number r_3 and compare it with the normalized $\varphi(x, y)$. If $r_3 > \varphi(x, y)/\varphi_{\max}$, reject this choice of (x, y) and try again with a new set of 3 random numbers. Otherwise, accept it and put it into an array of accepted points.
 - Repeat this procedure as many times as we need such points.
-

This routine generates the values of mtest and ntest as follows. A uniform random number generator is first used to generate points that are uniformly distributed over the entire allowed region of x and y . Given these uniformly distributed points we then *accept* each with a probability proportional to $\varphi(x, y)$ (otherwise they are rejected). The routine calc_nxy can readily be used in our variational-Monte Carlo calculations. In fact, we already used it in our work on the two-dimensional harmonic oscillator! In the exercises we will consider how much it speeds up the calculation as compared with simply choosing mtest and ntest uniformly.²⁷

A third useful modification of the variational-Monte Carlo method involves the manner in which we represent the trial functions. So far we have taken a direct, but in some ways crude approach by simply storing the value of φ at each grid site separately. A different strategy, which is in the spirit of a Fourier decomposition, would be to write the trial function as a sum of *basis* functions.²⁸ For example, with the harmonic oscillator we might choose the basis set to be Gaussian-like functions with different widths. The important point is that these functions be orthogonal so that an arbitrary trial function can be written uniquely in a form such as

$$\varphi(x, y) = \sum_k a_k b_k(x, y), \quad (10.30)$$

where $b_k(x, y)$ are the basis functions and the coefficients a_k specify the overlap between the trial wave function φ and the different basis functions. The variational procedure then involves adjusting the values of these coefficients so as to minimize the effective energy E^* . This can again be accomplished using a Monte Carlo approach.

EXERCISES

- 10.8.** Write a program to implement the variational-Monte Carlo approach and apply it to the two-dimensional harmonic oscillator. Study how the choice of initial trial wave function affects the number of Monte Carlo steps required to obtain

²⁷In most cases it is advisable to initially choose mtest and ntest from a uniform distribution. This prevents errors (i.e., zeros) in the initial trial wave function from unduly restricting the form of later trial functions. After the trial function begins to resemble the true wave function, you can then switch to choosing from a biased distribution and thereby speed up the final convergence of the Monte Carlo algorithm.

²⁸We will encounter such basis functions in detail later in this chapter

a "good" wave function. Here the "goodness" of a wave function can be judged from E^* . Compare the results for the very crude φ^0 considered in the text with that found for a Gaussian function. Other things to study include the influence of grid size on φ and E^* and the speed of convergence with and without a biased choice of the test points. *Hint:* Convergence can be determined by requiring that the change in E^* per iteration be less than some specified value. However, since some Monte Carlo iterations will not alter φ (and hence not change E^*), the convergence test cannot be performed after every iteration. Rather, you should make the test after every 10 or 100 iterations. A plot of E^* as a function of Monte Carlo time is also useful in testing for convergence.

- 10.9.** Employ the variational-Monte Carlo approach to obtain the ground-state wave function and energy for the one-dimensional square well. Compare your results with those obtained in Section 10.2 with the shooting method.
- *10.10.** Perform a variational-Monte Carlo calculation using an approach based on expressing the trial wave functions as functions with the form (10.30). Use the potential of your choice (it might be best to start with a simple square well).
- 10.11.** Use the variational-Monte Carlo method to find the ground-state wave function for a particle in a box in three dimensions. Compare the results for boxes with different wall potentials.
- 10.12.** Employ the variational-Monte Carlo method to calculate the ground-state energy and wave function of the anharmonic oscillator whose potential is given by $V(x) = x^4$.
- *10.13.** The variational theorem as stated at the beginning of this section concerns the ground-state energy and wave function. That is why all of the problems in this section have (so far) been concerned with the ground state. However, the method can also be used to deal with the excited states in the following way. The wave functions of the ground and excited states form a complete orthogonal set of basis functions. If the variational function φ is orthogonal to the ground-state wave function, then the final result of a variation calculation will be the wave function of the first excited state and E^* will be the energy of this state. While this may sound simple in principle, a difficulty in practice is that even though φ^0 may be constructed so as to be orthogonal to the ground state, the Monte Carlo changes may spoil this orthogonality.

Investigate this problem by using the variational-Monte Carlo method to estimate the first excited state of a particle in a square well. The challenging part of the problem is to ensure in some manner that after each Monte Carlo modification of φ , this function is still orthogonal to the ground-state wave function.

10.5 TIME-DEPENDENT SCHRÖDINGER EQUATION: DIRECT SOLUTIONS

In the past few sections we have explored several approaches for solving the *time-independent* Schrödinger equation. We have seen how to calculate the wave functions of the ground and excited states for different potentials in one, two, and three dimensions. However, to appreciate some of the most interesting aspects of quantum mechanics it is necessary to consider the time dependence of the wave function. This will be our aim in the remainder of this chapter. Our first goal is to treat problems such as wave-packet propagation in free space, reflection from a potential step, and tunneling through a barrier.

The time-dependent Schrödinger equation has the form

$$-\frac{\hbar^2}{2m} \nabla^2 \psi + V(\vec{r}) \psi = i\hbar \frac{\partial \psi}{\partial t}, \quad (10.31)$$

where t is the time and again $i \equiv \sqrt{-1}$. Dealing with this problem in three dimensions is a formidable computational problem, so we consider the (numerically) simpler case of one dimension. We then have

$$-\frac{\hbar^2}{2m} \frac{\partial^2 \psi}{\partial x^2} + V(x) \psi = i\hbar \frac{\partial \psi}{\partial t}, \quad (10.32)$$

where now ψ is a function of x and t . We will explore a number of ways of dealing with this problem. One is a spectral method, much like the one we employed in connection with waves on a string in Chapter 6; we will describe that approach in Section 10.7. Here and in the next section (10.6) we consider somewhat more direct attacks.

The first and main approach we consider is a rather straightforward extension of the centered difference method used in Chapter 6. It is instructive to rewrite (10.32) in the form

$$i\hbar \frac{\partial \psi}{\partial t} = \mathcal{H} \psi, \quad (10.33)$$

where the Hamiltonian operator is defined by

$$\mathcal{H} \equiv -\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x). \quad (10.34)$$

Constructing finite difference equations from these relations requires that we deal with complex quantities.²⁹ Thus, let us denote the real part of ψ by R and the imaginary part by I ,

$$\psi(x, t) = R(x, t) + iI(x, t). \quad (10.35)$$

Equation (10.33) then breaks into

$$\frac{\partial R}{\partial t} = \left(-\frac{1}{2} \frac{\partial^2}{\partial x^2} + V(x) \right) I \quad (10.36)$$

$$\frac{\partial I}{\partial t} = \left(\frac{1}{2} \frac{\partial^2}{\partial x^2} - V(x) \right) R, \quad (10.37)$$

where the factors involving \hbar and m have (according to common convention for such problems) both been set equal to unity.³⁰ We then approximate both the spatial and the time derivatives as centered differences. For example,

$$\frac{\partial^2 R(x, t)}{\partial x^2} \approx \frac{R(x + \Delta x, t) - 2R(x, t) + R(x - \Delta x, t)}{(\Delta x)^2}, \quad (10.38)$$

²⁹Unlike the time-independent problems considered in the previous sections, for most time-dependent problems we can not avoid a complex wave function.

³⁰This is a slightly different convention than what we used in discussing the time-independent Schrödinger equation in earlier sections.

and

$$\frac{\partial I(x,t)}{\partial t} \approx \frac{I(x,t + \Delta t/2) - I(x,t - \Delta t/2)}{\Delta t}. \quad (10.39)$$

Inserting these into (10.37) yields an expression for $I(x,t + \Delta t/2)$ in terms of $I(x,t - \Delta t/2)$, $R(x \pm \Delta x, t)$, and $R(x, t)$. After a little rearranging we get

$$\begin{aligned} I(x,t + \Delta t/2) &\approx I(x,t - \Delta t/2) \\ &+ \frac{\Delta t}{2(\Delta x)^2} [R(x + \Delta x, t) - 2R(x, t) + R(x - \Delta x, t)] \\ &- (\Delta t)V(x)R(x, t). \end{aligned} \quad (10.40)$$

If we now discretize space by taking $x_m = m\Delta x$, and use the values of R and I at x_m , it is evidently most convenient, according to (10.40), to evaluate $R(x, t)$ at $x = x_m$ and $t = n\Delta t$ while $I(x, t)$ is evaluated at $x = x_m$ and at the *shifted* times $t = (n \pm \frac{1}{2})\Delta t$. The corresponding expression for the real part of the wave function R using the shifted discrete times for R and I is

$$\begin{aligned} R(x,t + \Delta t) &\approx R(x,t) - \frac{\Delta t}{2(\Delta x)^2} [I(x + \Delta x, t + \Delta t/2) - 2I(x, t + \Delta t/2) \\ &+ I(x - \Delta x, t + \Delta t/2)] - (\Delta t)V(x)I(x, t + \Delta t/2). \end{aligned} \quad (10.41)$$

We are thus evaluating the real and imaginary parts of $\psi(x, t)$ in a *leapfrog* fashion at “staggered” times. However, this presents no practical difficulty, and in fact one can show that the numerical errors made in this method are locally of order $(\Delta t)^3$, the same order as with the second-order Runge-Kutta methods. This leapfrog approach also has the very important property of conserving the normalization of the wave function; that is, the integral of $\psi^*\psi$ over all space is conserved.

While the leapfrog method just described is suitable for carrying out the calculations to obtain all of the results that we show later in this section, we will now describe a second direct approach which is related to the general matrix formulation that we discussed earlier in this chapter. This second method provides an important lesson in dealing with linear equations containing Hermitian operators such as \mathcal{H} . Although (10.33) has the appearance of a simple first-order differential equation involving the time dependence of ψ , things are complicated by the fact that \mathcal{H} is an operator rather than a number. Ignoring that fact for a moment, (10.33) has the formal solution

$$\psi(x, t) = \exp(-i t \mathcal{H}/\hbar) \psi(x, 0), \quad (10.42)$$

where $\psi(x, 0)$ is the wave function at $t = 0$. Thus, if we know $\psi(x, 0)$, we can *formally* calculate the behavior at all future times using (10.42). However, this formal solution is not as useful as we might hope because of the operator \mathcal{H} in the exponent. To be clear on what it means mathematically to have an operator expression of this kind, it is useful to rewrite (10.42) using the Taylor expansion of the exponential factor

$$\exp(-i t \mathcal{H}/\hbar) \psi = \left(1 - \frac{i t}{\hbar} \mathcal{H} - \frac{t^2}{2\hbar^2} \mathcal{H}^2 \dots\right) \psi, \quad (10.43)$$

where \mathcal{H}^n means that the operator \mathcal{H} is applied n times in succession. The problem comes when we have to evaluate terms such as $\mathcal{H}^2\psi$. Recalling the definition of \mathcal{H} , we see that such a term expands to give

$$\mathcal{H}^2 \psi = \mathcal{H} \mathcal{H} \psi = \left[-\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x) \right] \left[-\frac{\hbar^2}{2m} \frac{\partial^2}{\partial x^2} + V(x) \right] \psi. \quad (10.44)$$

Difficulties now arise from the fact that the two terms that make up \mathcal{H} do not commute; that is, the order of operation is important. We cannot simply switch the order with which we apply the derivative term in \mathcal{H} (the kinetic energy piece) and the potential energy part (which, in general, depends on position). This makes dealing with the time-dependent Schrödinger equation more complicated than dealing with a simple first-order differential equation such as those discussed in Chapter 1.

While (10.42) is a perfectly fine formal solution, it is not of much practical use since the expansion (10.43) involves a very large (infinite) number of terms. However, this formal solution does suggest a way to proceed numerically. Let us consider applying the formal solution over a very small time interval. Indeed, let us discretize time in steps Δt ; using (10.42) we obtain

$$\psi(x, t + \Delta t) = \exp(-i\Delta t \mathcal{H}/\hbar) \psi(x, t). \quad (10.45)$$

But, you might ask, how can this be a useful approach in view of the difficulties with the expansion (10.43)? The reason we could have some optimism with regard to (10.45) is that a small value of Δt will make the higher-order terms in the expansion (10.43) small. Let us assume for now that these terms can be made small enough that we can stop at the term linear in \mathcal{H} , which yields³¹

$$\psi(x, t + \Delta t) \approx (1 - i\Delta t \mathcal{H}/\hbar) \psi(x, t) = \left[1 + \frac{i\Delta t}{2} \frac{\partial^2}{\partial x^2} - i\Delta t V(x) \right] \psi(x, t), \quad (10.46)$$

where in the last step we have again assumed $\hbar = 1$ and $m = 1$. Given (10.46), a natural way to proceed is to discretize space into units of size Δx and write the wave function as $\psi(m, n) \equiv \psi(m\Delta x, n\Delta t)$. Expressing the second partial derivative in the usual finite-difference form, (10.46) becomes

$$\begin{aligned} \psi(m, n+1) \approx & \psi(m, n) + \frac{i\Delta t}{2} \left[\frac{\psi(m+1, n) + \psi(m-1, n) - 2\psi(m, n)}{(\Delta x)^2} \right] \\ & - i\Delta t V(m) \psi(m, n). \end{aligned} \quad (10.47)$$

Hence, given $\psi(m, n)$ it appears that we can readily calculate $\psi(m, n+1)$, the wave function at the next time step. However, this approach contains two flaws.

The first flaw is this procedure is numerically unstable! The effects of small round-off and other types of errors (due, for example to the higher-order terms we have neglected) grow rapidly with time, essentially because of the exponential factor in (10.45) from which this finite-difference equation was derived. Amazingly,

³¹This could also be obtained by expressing (10.33) in finite-difference form, but we will have further use for (10.45) moment

this instability can be cured by rewriting (10.45) so as to, in a sense, propagate *backward* in time,

$$\psi(x, t) = \exp(+i\Delta t \mathcal{H}/\hbar) \psi(x, t + \Delta t) \approx (1 + i\Delta t \mathcal{H}/\hbar) \psi(x, t + \Delta t). \quad (10.48)$$

We will not go into the details of precisely why the scheme (10.48) leads to numerical stability, but leave that for interested readers to explore using the references. However, this stability does come at a price; when (10.48) is converted into finite difference form one is led to the following set of the finite-difference equations

$$\begin{aligned} \psi(m, n) &\approx \psi(m, n + 1) \\ &- \frac{i\Delta t}{2} \left[\frac{\psi(m + 1, n + 1) + \psi(m - 1, n + 1) - 2\psi(m, n + 1)}{(\Delta x)^2} \right] \\ &+ i\Delta t V(m) \psi(m, n + 1). \end{aligned} \quad (10.49)$$

Comparing this with (10.47) we find that it is an *implicit* relation for $\psi(m, n + 1)$. That is, we cannot use (10.49) to express $\psi(m, n + 1)$ solely in terms of the wave function at time step $n + 1$, but we are stuck with terms of the form $\psi(m', n + 1)$ with $m' \neq m$ in the expression for $\psi(m, n + 1)$. Hence, we cannot just step through the values of m and use (10.49) to calculate the wave function at the next time step. We must instead solve a system of algebraic equations, and this amounts to a matrix formulation as described earlier in this chapter.

However, even this price is not all that we must pay. Equations (10.47) and (10.49) both violate a fundamental property of quantum mechanics called *unitarity*. The Schrödinger equation preserves the proper normalization of a wave function, that is, the total probability for finding a particle somewhere in space must always be unity.³² It turns out that $(1 - i\Delta t \mathcal{H}/\hbar)$, the time evolution operator in (10.46), is *not* unitary, and the same is true of (10.48). We must therefore devise a different strategy.

A numerically stable approach that does satisfy unitarity involves writing the exponential factor in (10.45) in what is known as the Cayley form

$$\exp(-i\Delta t \mathcal{H}/\hbar) \approx \frac{1 - i\Delta t \mathcal{H}/2\hbar}{1 + i\Delta t \mathcal{H}/2\hbar}. \quad (10.50)$$

We will leave it to the reader to show that to lowest order in \mathcal{H} (in a Taylor expansion), this is equal to both $1 - i\Delta t \mathcal{H}$ and $(1 + i\Delta t \mathcal{H})^{-1}$, which are the key factors in (10.46) and (10.48). However, these three ways of approximating the exponential factor $\exp(-i\Delta t \mathcal{H})$ are *not* equivalent with regard to maintaining unitarity, as can be seen in the following way. Using (10.50) to propagate our wave function forward in time we have

$$\psi(x, t + \Delta t) \approx \frac{1 - i\Delta t \mathcal{H}/2\hbar}{1 + i\Delta t \mathcal{H}/2\hbar} \psi(x, t). \quad (10.51)$$

³²The conservation of probability is ensured by the *unitarity* of the operator $\exp(-it\mathcal{H}/\hbar)$ in (10.42), which is the direct result of the fact that \mathcal{H} is Hermitian.

From (10.51) we first obtain

$$\left[1 + \frac{i\Delta t \mathcal{H}}{2\hbar}\right] \psi(x, t + \Delta t) = \left[1 - \frac{i\Delta t \mathcal{H}}{2\hbar}\right] \psi(x, t) \quad (10.52)$$

We now replace \mathcal{H} by (10.34), convert everything to finite-difference form, and discretize space and time so that $\psi(m, n) \equiv \psi(m\Delta x, n\Delta t)$. After rearranging a few terms we get

$$\begin{aligned} & \psi(m+1, n+1) + [2i\lambda - 2(\Delta x)^2 V(m) - 2] \psi(m, n+1) + \psi(m-1, n+1) \\ &= -\psi(m+1, n) + [2i\lambda + 2(\Delta x)^2 V(m) + 2] \psi(m, n) - \psi(m-1, n), \end{aligned} \quad (10.53)$$

where for notational convenience we define $\lambda \equiv 2(\Delta x)^2/\Delta t$. The system of equations (10.53) has a form that is different from anything we have encountered in previous chapters. Most importantly, they cannot be rewritten so as to express $\psi(m, n+1)$ solely in terms of $\psi(m, n)$. Hence, we cannot simply loop through the values of m and compute each $\psi(m, n+1)$ in a straightforward manner. Instead, we have an *implicit* numerical problem, which means that the value of $\psi(m, n+1)$ for a particular value of m depends on $\psi(m', n+1)$, where $m' \neq m$. That is, we have a set of algebraic equations (which can again be cast as a matrix problem) that must be solved to obtain the wave function at the next time step.

As noted earlier in this chapter, many eigenvalue problems can be expressed in matrix form, with the solution then requiring that a matrix be diagonalized or inverted. In the present case we must invert the matrix whose coefficients can be read from (10.53). The bad news is that this will be a *large* matrix. In the particular problems we will consider below, the number of grid elements along x is typically 1000, so the associated matrix is 1000×1000 in size. This is a formidable computational problem, especially when we realize that such a matrix must be inverted for *each* time step! Fortunately, there is some good news: most of the elements of this matrix are zero. If we examine (10.53) we see that the equation for $\psi(m, n+1)$ involves just $\psi(m \pm 1, n+1)$, and no other values of ψ [except for itself, $\psi(m, n+1)$]. Thus, each row of our matrix has only three nonzero elements. Since one of these is on the diagonal and the other two are just one space away, this matrix is *tridiagonal*. Such matrices arise often in numerical work, since as in the present case the effectively “local” nature of the underlying differential equation gives rise naturally to this matrix structure. As you might expect, the large number of the zeros in the matrix can be used to our advantage. There are several algorithms which take advantage of the special form and give good results. A popular approach is the *Crank-Nicholson* method, which is described in detail in the paper by Goldberg, Schey, and Schwartz in the references (this method was also discussed in detail in the first edition of this book). The Crank-Nicholson approach and the leapfrog method described earlier in this section can both be used for the calculations we describe in the remainder of this section.

We now consider a few examples of the time-dependent behavior of ψ as calculated using one of the algorithms described above. The primary initial condition

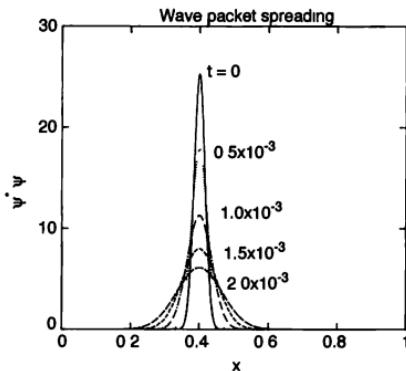


FIGURE 10.17: Time-dependent behavior of a Gaussian wave packet. The initial wave packet ($t = 0$) is given by (10.54) with $x_0 = 0.4$ and $\sigma^2 = 1 \times 10^{-3}$. The wave function at later times (the time values are given in the figure) is seen to spread out in space. For this calculation we used $\Delta x = 0.0005$ with the boundary condition $\psi = 0$ applied at $x = 0$ and $x = 1$. We also took $\Delta t = 5 \times 10^{-7}$ so that $\lambda = 2(\Delta x)^2/\Delta t = 1$.

in these calculations is the initial wave function. In all of the examples considered below we assume a Gaussian form for ψ at $t = 0$, so that our initial wave function is given by

$$\psi(x, t = 0) = C \exp[-(x - x_0)^2/\sigma^2], \quad (10.54)$$

where the center of the packet is at x_0 , its width is σ , and the factor C is chosen to satisfy the normalization condition on ψ . A plot of such a wave packet is shown by the solid curve in Figure 10.17. This wave function is not stable with time; we see from Figure 10.17 that it broadens as t increases. Since our algorithm maintains unitarity³³ a broader wave function must have a lower value at the peak, as also seen in Figure 10.17.

The physics responsible for this wave-packet spreading can be understood by recalling that we are dealing with an equation much like the wave equation considered in Chapter 6. The Gaussian wave packet (10.54) can be decomposed into (i.e., written as) a sum of plane waves of the form e^{ikx} . These plane waves are essentially just Fourier components³⁴ whose amplitudes are chosen so that they add up to give the Gaussian packet (10.54) at $t = 0$. However, each component is a wave that travels along the system. These waves have different wave vectors, k ,

³³It is a good idea to periodically check that $\int \psi^* \psi dx = 1$, as this provides a strong test that your program does not contain any subtle bugs. The wave functions shown in Figure 10.17 and in later figures all satisfy this test. With the leapfrog algorithm, we need to sum the squares of the real and imaginary parts evaluated half a time step apart to obtain ψ^2 , but this usually presents no practical problem.

³⁴Just as the sines and cosine standing waves are the Fourier components for waves on a string

and thus different velocities. In addition, half of them travel in the $+x$ direction and half along $-x$. The motion of the components causes the wave packet to spread with time. This can also be interpreted using the Heisenberg uncertainty principle, according to which $\delta x \delta p \geq \hbar/2$, where δx and δp are the widths of the wave packet in terms of space and momentum. Since the momentum $p = \hbar k$, the uncertainty relation tells us that confining the particle to an initial wave packet of width δx (which in Figure 10.17 is ~ 0.05) requires an uncertainty in k , that is, a corresponding spread in the values of k , and this makes the packet spread out with time.

The simple packet (10.54) includes components with both positive and negative wave vectors in equal amounts. This makes it a stationary packet in the sense that the average position $\bar{x} \equiv \int \psi^* x \psi \, dx$ does not change with time. This is not the case for the packet

$$\psi(x, t=0) = C \exp[-(x - x_0)^2/\sigma^2] \exp[i k_0 x], \quad (10.55)$$

which is just our original Gaussian function multiplied by a plane wave with wave vector k_0 . This wave function, which is complex, can also be written as the sum of components with a range of k values, but with the range now centered around k_0 . As a result, this packet travels with an average velocity $v_0 = \hbar k_0/m$. It is interesting to compare it with the stationary wave functions considered above. While we only show $\psi^* \psi$ in Figure 10.17, you can see from (10.54) that at $t=0$ this is also just the square of the real part of ψ and that the imaginary part is zero. Our propagating packet is a little more complicated, as can be seen from Figure 10.18. While $\psi^* \psi$ has the same simple Gaussian shape, the real and imaginary parts of ψ oscillate rapidly. This oscillation is due to the factor $\exp(i k_0 x)$ in (10.55). In fact, for these oscillations to be prominent, we must choose k_0 and σ appropriately; i.e., we need $\sigma \gg 1/k_0$. This choice also makes $\hbar k_0$ a well-defined mean momentum of the packet, as the width of the wave packet in k -space is proportional to $1/\sigma$.

In carrying out the calculations, we should also not forget the stability and accuracy requirements. As in any numerical solution of initial-value problems, the calculated wave function can propagate at a speed that is no faster than $\Delta x/\Delta t$, where Δx and Δt are the discrete steps in space and time, respectively. Since the wave packet moves on average with a speed $v_0 = \hbar k_0/m$, we must have $\Delta x/\Delta t \geq \hbar k_0/m$ for the calculation to keep up with the packet. Also an accurate description of the oscillating real and imaginary parts in the packet requires $\Delta x \ll \lambda_0$ where $\lambda_0 = 2\pi/k_0$ is the average wavelength.

The results of our propagation program when applied to this packet are shown in Figure 10.19. Here the packet in Figure 10.18 was allowed to move in a region with $V=0$. That is, we had a freely moving particle. We will leave it to the exercises to show that this motion has the expected velocity. You should notice that the height of the propagating packet decreased as it moved. This is due to the wave-packet spreading we noted in connection with the stationary packet in Figure 10.17. The uncertainty relation tells us that all such packets will spread with time. If we had chosen a larger value of k_0 , the velocity of the packet would be larger, and since the rate of spreading would be the same, the packet would better

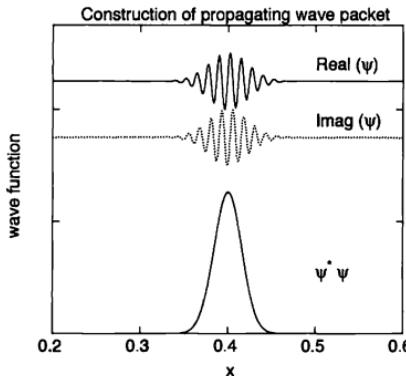


FIGURE 10.18: Composition of a propagating wave packet. For this packet we took $k_0 = 500$, $x_0 = 0.4$, and $\sigma^2 = 0.001$.

maintain its amplitude for a given distance traveled. We will investigate this in the exercises.

We can use this propagating wave packet to study a wide variety of problems involving reflection, transmission, and tunneling. The programming is essentially identical to that required to treat motion in a flat ($V = 0$) potential region. The only difference is that we must employ the appropriate potential function $V(x)$ in the calculation. We first consider an encounter with a potential wall; that is, we let our wave packet begin in a region with $V = 0$ and propagate toward a region where V is very large. Figure 10.20 shows the behavior with a wall located at $x = 0.6$. The potential for $x \geq 0.6$ was 1×10^6 . This was much greater than the average energy of the wave packet, $E_{\text{packet}} = \hbar^2 k_0^2 / 2m$, which in this case was 1.25×10^5 . Hence, the particle did not have enough energy to classically propagate into the region $x \geq 0.6$ and was instead completely reflected.

In some respects the reflection process resembles the behavior of a wave pulse on a string when it encounters the end of the string. However, the details in the quantum case are somewhat different as can be seen from the detailed plots on the right in Figure 10.20. The rapid oscillations of $\psi^* \psi$ are especially striking and are often not appreciated from the usual analytic treatments of the reflection process.

Another interesting case is a potential cliff. We again consider a wave packet that, as in Figure 10.20, was incident from a region in which $V = 0$. Now, however, the potential *dropped* to $V = -10^6$ when $x \geq 0.6$. From Figure 10.21 we see that while some of the packet was transmitted, a comparable amount was also reflected. That is, there was a nonzero probability for the particle to be reflected from the cliff! This is a nice illustration of the wave nature of quantum mechanics, since such behavior would never be found in classical mechanics. In a sense we could

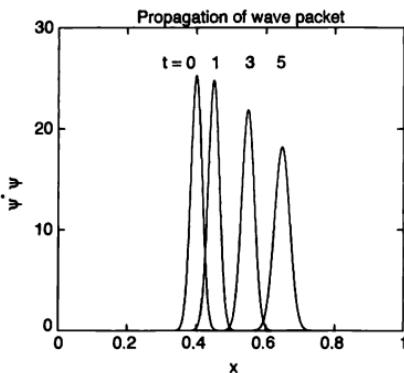


FIGURE 10.19: Wave packet (10.55) from Figure 10.18 propagating in a region with $V = 0$. The time values given in the figure are to be multiplied by $\times 10^{-4}$. We used $\Delta x = 0.0005$ and $\Delta t = 5 \times 10^{-7}$ in the calculation.

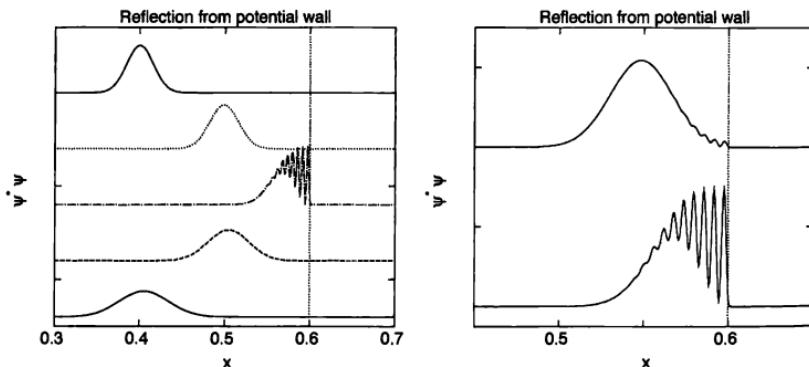


FIGURE 10.20: Wave packet reflecting from a wall, which is indicated by the vertical dotted lines. The wave-packet parameters were the same as those used in Figure 10.19. The potential was $V = 0$ on the left ($x < 0.6$) and $V = 1 \times 10^6$ on the right ($x \geq 0.6$). Left: plots of ψ at various times, with time increasing as we move down the sequence. The particle initially traveled from left to right, was reflected from the wall, and then traveled toward the left. Note that the wave packet at the bottom was significantly broader than the initial packet, due to the spreading effects discussed above. Right: expanded plot of two views recorded as the wave was "striking" the wall.

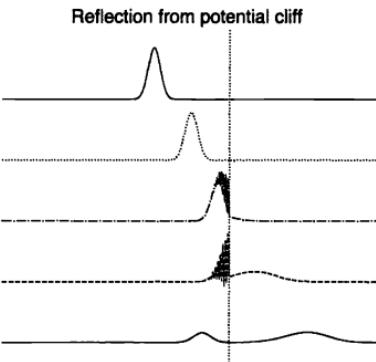


FIGURE 10.21: Wave packet incident on a cliff whose position is indicated by the vertical dotted line. The wave-packet parameters were the same as those used in Figure 10.20. The potential was $V = 0$ on the left ($x < 0.6$) and $V = -1 \times 10^6$ on the right ($x \geq 0.6$). These are plots of $\psi^* \psi$ at various times, with time increasing as we move down the graph. The particle initially traveled from left to right. After the encounter with the cliff, a small packet continued to the right, while another packet was reflected and traveled to the left. Note that the transmitted packet has a higher speed than the reflected one, since in this region the kinetic energy is higher (due to the lower potential energy).

conclude that a quantum world would be a safer place to live, since it would be harder to fall off a quantum cliff than a classical one.

As a final example in this section we consider a wave packet incident on a barrier. Here the potential was $V_0 = 1.25 \times 10^5$ in the region $0.6 \leq x \leq 0.7$, with $V = 0$ to the left and right of the barrier. We chose this value of V_0 since it was equal to the “average” kinetic energy for this wave packet, $\hbar^2 k_0^2 / 2m$. We see from Figure 10.22 that the incident packet was mainly reflected. If we had plotted the results on a greatly expanded scale, you would observe that a small amount was also transmitted. In addition, on the time scale considered here a small amount of the packet “remained behind” in the barrier region. The velocity in this region is essentially zero, since the average kinetic energy is $\hbar^2 k_0^2 / 2m - V = 0$, so it takes a long time for a wave packet in this region to escape.

The general approach to the time-dependent Schrödinger equation, which we have described in this section, can be applied to a very wide variety of problems. Further examples will be explored in the exercises.

EXERCISES

- 10.14.** Study the motion of a propagating wave packet in free space as in Figure 10.19 and show quantitatively that it does indeed move with a velocity of $\hbar k_0 / m$. Compare the behavior with different values of k_0 . You should find that as k_0 is made larger, our algorithm requires smaller spatial and time steps to avoid

Reflection from potential barrier

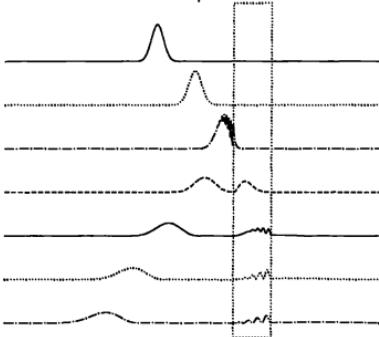


FIGURE 10.22: Wave packet partially reflecting from a potential barrier (indicated by the dotted lines) with $V = 1.25 \times 10^5$. The other parameters were the same as used in Figure 10.20. The packet initially traveled from left to right (top). Part of the packet was reflected and traveled away to the left, while a very small part (which is barely visible on the scale used here) was transmitted and proceeded to the right.

substantial numerical errors. These errors are not significant for the values of Δx and Δt we have used in the calculations in this section, but this would not have been the case if we had employed somewhat larger values of k_0 .

- 10.15.** Study how the rate of wave-packet spreading as observed in Figure 10.17 varies with the width of the initial packet. Compare this spreading quantitatively with the Heisenberg uncertainty relation. Do this by calculating the width of the wave packet as a function of time.
- 10.16.** Observe the propagation of a wave packet in a region in which the potential increases linearly with position. Study the reflection of the wave packet in this case.
- *10.17.** Study reflection and transmission from a barrier for which the potential is greater than the kinetic energy of the incident wave packet. The probability for the particle to tunnel through such a barrier decreases rapidly as the potential is increased. In order to have a significant transmission probability you will have to make the barrier thinner than in Figure 10.22. *Hint:* Good parameter choices are $k_0 = 700$, $\Delta t = 5 \times 10^{-7}$, and $\Delta x = 5 \times 10^{-4}$. Let the height of the potential barrier be $V_0 = 2 \times k_0^2$ with a width of 0.05. You should also explore the behavior for other values of both V_0 and the width of the barrier.
- *10.18.** Compute the scattering of a wave packet from the potential well shown in Figure 10.23. Try to observe effects associated with the bound states in the well. *Hint:* Start with the wave-packet parameters given in the previous problem. Assuming that the particle is incident from the left, take the height of the left-most barrier to be $4k_0^2$ and give it a width of 0.05. Give the right-most barrier the same width and a much larger height, and let the two barriers be separated by $L = 1$ (or less). You should find that the penetration of the wave function into

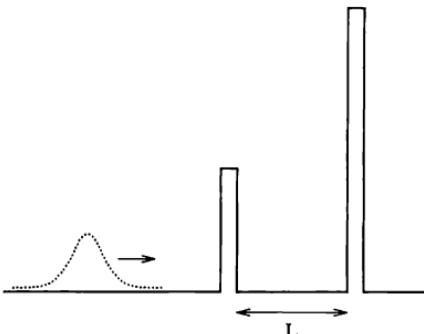


FIGURE 10.23: Schematic illustration of a wave packet (dotted curve) incident on a region containing two potential barriers $V = 0$ on the left, on the right, and between the barriers. The barrier heights are greater than the kinetic energy of the particle.

the potential well depends on the value of k_0 and that this penetration is greatest when the energy of the particle matches the energy of one of the approximate bound states of the well.

10.6 TIME-DEPENDENT SCHRÖDINGER EQUATION IN TWO DIMENSIONS

In this section we extend the leapfrog method introduced in the previous section to two dimensions. The further extension to three dimensions is straightforward, although plotting (and making sense of!) the results becomes more difficult as the spatial dimension grows.

When working in two dimensions, our discrete versions of the potential $V(x, y)$ and the functions $\psi(x, y, t) = R(x, y, t) + iI(x, y, t)$ have two spatial indices, and $\psi(x, y)$ must now be normalized on a two-dimensional grid. However, the modifications required to the numerical algorithm in (10.40) and (10.41) are relatively minor. The replacement of $\partial^2/\partial x^2$ in the one-dimensional Hamiltonian (10.34) by the two-dimensional Laplacian ∇^2 changes the spatial derivative term (10.38) for the real part of ψ (the function R), and for the imaginary part of ψ (the function I). We still use centered differences to approximate these derivatives, and for the real part of the wave function we get

$$\begin{aligned} \nabla^2 R(x, y, t) &\approx \frac{R(x + \Delta x, y, t) - 2R(x, y, t) + R(x - \Delta x, y, t)}{(\Delta x)^2} \\ &\quad + \frac{R(x, y + \Delta y, t) - 2R(x, y, t) + R(x, y - \Delta y, t)}{(\Delta y)^2} \\ &= \frac{R(x + \Delta x, y, t) + R(x - \Delta x, y, t) + R(x, y + \Delta x, t) + R(x, y - \Delta y, t) - 4R(x, y, t)}{(\Delta x)^2}, \end{aligned} \tag{10.56}$$

where we again set $\Delta x = \Delta y$ for simplicity. The spatial derivative term for the imaginary part of the wave function has a similar form. These can then be inserted into the two-dimensional versions of (10.36) and (10.37), and after some straightforward algebra one can obtain leapfrog relations for R and I which are analogous to (10.40) and (10.41). The incoming wave packet must also be two dimensional (by suitable generalization of (10.55)), with an initial wave vector of the form $\mathbf{k}_0 = k_{0x}\hat{x} + k_{0y}\hat{y}$. The constraints for numerical stability and accuracy as well as for the quantum effects are similar to those in one dimension.

The main computational challenge may actually be visualization. Unlike in one dimension, we need both the x and y directions to describe the position and thus we must find a way to represent the probability density $\psi^*\psi$. This is often done by plotting a perspective view of a three-dimensional surface where the z direction represents the probability density. However, another effective way is to use colors or shades of gray to represent $\psi^*\psi$ on a two-dimensional grid. In what follows, we will mostly use the latter method as it tends to require less computing and does not require sophisticated graphics routines.

Some typical results are shown in Figure 10.24, where the wave packet scatters from a cylindrical potential well of radius $a = 0.2$ centered at the origin of a square grid with x and y ranging from -1 to 1 . The well has a depth $V_0 = -1000$, and a Gaussian wave packet of incident energy $k_0 = 800$ is incident from the left. Successive plots describe the time evolution of the wave packet from $t = 0$ through $t = 0.02$ (the time step for the calculation was $\Delta t = 0.0001$). For this case of the kinetic energy of the incoming wave packet on the same order as the depth of the potential well, and we find large oscillations in the probability amplitude near the edges of the well (as we also observed in one-dimension), and most of the packet is eventually transmitted to the right. There is also a small reflected wave, but it is hard to see in this figure because its amplitude is very small. We also find that transmission and reflection occur not just along the incoming axis but also *around* the well, and the main transmitted amplitude spreads laterally as well. In two dimensions, there are simply many more paths the wave packet can take, allowing parts of the packet to avoid going *through* the well.

Another example is shown in Figure 10.25, where the same incoming wave packet impinges on a cylindrical potential barrier instead. The incoming wave packet energy is now a bit smaller than the barrier height, so one would expect large quantum effects. Recall that in one dimension this situation typically results in significant penetration of the packet into the classically forbidden barrier interior, followed by reflection and transmission, with a small third wave portion of the wave function remaining trapped inside the barrier. The result in two dimensions is somewhat different. From Figure 10.25, we see that the bulk of the incident wave packet goes *around* the barrier instead of *through* it. However, similar to one dimension, a small portion of the wave penetrates the barrier and lingers there for quite some time after the main part of the packet passes by, though this is hard to see from these figures. In order to illustrate the last point, we also show in Figure 10.26 the three-dimensional surface view at $t = 0.0250$. We clearly see a small peak right near the center of the barrier (the origin), while most of the wave packet has either gone around the barrier or been reflected.

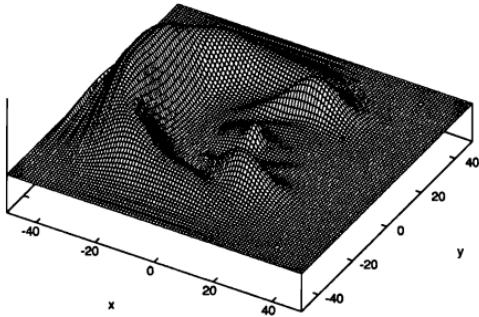


FIGURE 10.26: Wave packet of Figure 10.25 at $t = 250$ is shown in a three-dimensional mesh surface view. Note the small peak near the center of the potential barrier (the origin)

EXERCISES

- 10.19.** Write a program that implements the leapfrog algorithm in two dimensions, and use it to study two-dimensional scattering from cylindrical wells and barriers. Potential wells with a depth of twice the incoming kinetic energy, and barriers with a height of half the kinetic energy are interesting choices. Compare your results with those for the corresponding one-dimensional problem.

10.7 SPECTRAL METHODS

The time-dependent Schrödinger equation can also be treated with spectral methods, much like the approach we developed in our dealings with waves on a string. These methods make use of the formal time dependence of the wave function in (10.42), together with the fact that *any* initial wave function can be written in terms of the eigenstates of the associated time-independent problem. That is, if the functions φ_n are the solutions of the time-independent Schrödinger equation for the particular potential of interest, any wave packet ψ can be written as

$$\psi = \sum_n a_n \varphi_n , \quad (10.57)$$

where the a_n are complex constants. Note that here the label n is used to refer to different eigenstates, that is, energy levels; it does *not* correspond to space or time. This is analogous to the Fourier expansions of arbitrary functions, which are

discussed in Chapter 6 and Appendix C. The eigenfunctions φ_n form a complete set in the sense that *any* function can be written in the form (10.57).

The time dependence of the basis functions follows from (10.42). Since φ_n is an eigenfunction, we know that $\mathcal{H}\varphi_n = E_n\varphi_n$, where E_n is the energy of state n . If our initial wave function happened to be $\psi(t=0) = \varphi_n$, it would evolve in time according to (10.42), which leads to

$$\psi(t) = \exp(-itE_n/\hbar) \varphi_n . \quad (10.58)$$

For a wave packet composed of a sum of basis states such as (10.57) the same arguments give

$$\psi(t) = \sum_n a_n \exp(-itE_n/\hbar) \varphi_n . \quad (10.59)$$

Thus, if we can construct the expansion of the initial wave function in terms of the basis states, that is, if we can determine the coefficients a_n in (10.57), we can use (10.59) to compute the wave function at all future times.

This approach is very much like the spectral method we employed in Chapter 6. There we wrote the displacement of a string in terms of Fourier components. Since the time dependence of the Fourier components for waves on a string has a very simple form (purely sinusoidal), this allowed us to compute the time dependence of a wave packet on a string. Here we take similar advantage of the simple form (10.58) for the time dependence of a basis function. Evaluating (10.59) involves only some arithmetic with complex numbers; there are no algebraic or differential equations to solve.

To illustrate this approach we consider a one-dimensional harmonic oscillator. The first step is to determine the basis functions, φ_n . In this case these are the solutions to the time-independent Schrödinger equation, (10.2), with $V = Kx^2/2$. We considered this problem numerically in the exercises in Section 10.2. The exact solution is, of course, also known (the wave functions can be expressed as the product of a Gaussian and a Hermite polynomial), as discussed in Griffiths (1995) and Schiff (1968). However, since we want to show how the spectral approach works in the general case, we will calculate the basis functions numerically, using the shooting method. In principle, if we want the expansion (10.57) to represent our wave packet exactly, we need an *infinite* number of basis functions in the expansion. Of course, this is not possible in a numerical treatment; in the calculations below we will use 32 basis functions and find that this is sufficient for the particular problems we'll be considering. We will discuss how to estimate the necessary number of basis states in the exercises.

With the basis functions in hand, the next step is to calculate the expansion coefficients in (10.57). To do this we take advantage of the fact that the functions φ_n form an orthogonal set. That is, $\int \varphi_m^* \varphi_n dx$ is zero if $m \neq n$ and unity if $m = n$. Hence we can write

$$\int \varphi_m^* \psi_i dx = \int \left[\varphi_m^* \left(\sum_n a_n \varphi_n \right) \right] dx = a_m , \quad (10.60)$$

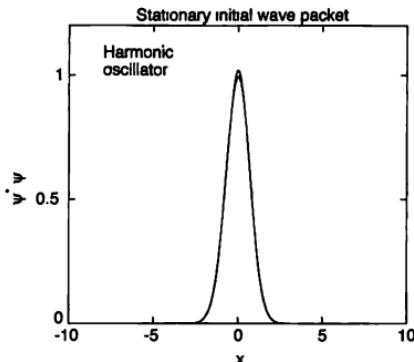


FIGURE 10.27: Time evolution of a stationary wave packet with an initial width $\sigma = 1$ [see (10.54)]. The potential was $V = Kx^2/2$ with $K = 1$. We have plotted the wave packet at $t = 0, 0.5, 1$, and 2 , but the results overlap almost completely and are thus hard to distinguish

where ψ_i is our initial wave packet. Calculation of the expansion coefficients thus requires integration of the product of each basis function with the initial wave function. Since these functions are known only at the grid locations, this integration is actually just a simple sum.

The results of this calculation for a Gaussian initial wave packet (10.54) are shown in Figure 10.27. That is, we first used (10.60) to calculate the coefficients a_k , and then computed the variation of the wave function with time using (10.59). We see from Figure 10.27, that this wave packet changed very little with time. How can this be? It just so happens that we have picked an initial wave packet that is extremely close to the ground-state wave function, of the harmonic oscillator. The ground-state solution corresponds classically to a particle sitting at the bottom of the potential well ($x = 0$), and this is what we find in Figure 10.27.

An example that exhibits more “action” is shown in Figure 10.28, where we consider a narrower initial wave packet. The functional form of ψ_i is again (10.54) but here we have chosen $\sigma = 0.5$. This is considerably narrower than the ground-state wave function, so we are now far from being in an eigenstate, and the result is a substantial time dependence. Our packet was initially centered at $x = 0$, and it remained so as time advanced. However, the width grew considerably from $t = 0$ to 1.5, providing another example of the uncertainty principle. The initial packet effectively contained components with wave vectors different from zero, and these components oscillated (harmonically) in the potential well. The initial symmetry was maintained, so the wave packet was always centered at $x = 0$, but the width varied with time. In fact, the width oscillated with time, as can be seen from the results plotted on the right in Figure 10.28.

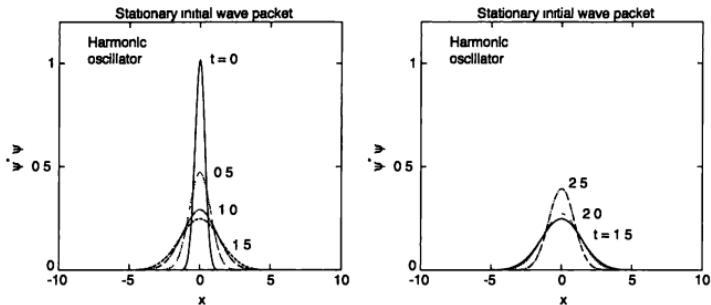


FIGURE 10.28: Time evolution of a wave packet with an initial width $\sigma = 0.5$. Left: $t = 0, 0.5, 1.0$, and 1.5 . Right: $t = 1.5, 2.0$, and 2.5 . The potential was the same as in Figure 10.27. Note that the packet initially broadened, but from $t = 1.5$ to 2.5 it became narrower with time.

It is also interesting to consider the effect of displacing the initial wave packet from $x = 0$, and this is shown in Figure 10.29. The initial wave packet was again given by (10.54), but here we took $x_0 = 2$ so that our packet was displaced from the center of the potential well. Classically this corresponds to giving the oscillator an initial displacement. We see that our particle oscillated back and forth in the well, precisely as expected for a simple harmonic oscillator.

Here the shape and width of the packet change very little with time, as we have chosen the value of σ that we have already shown to give an essentially time-independent width. We can therefore follow the position of the particle by simply plotting the location of the maximum of $\psi^* \psi$ as a function of time, and this is shown on the right in Figure 10.29. We find a simple cosine like oscillation, as expected for a simple harmonic oscillator. We will leave it to the exercises to show that the period of these oscillations has the expected value (it does).

We have illustrated the spectral approach with a problem for which the basis states could have been calculated analytically (although we chose to calculate them numerically). However, this method can be used for problems in which little headway can be made analytically. To demonstrate this we consider the behavior of an anharmonic oscillator. We take the potential to be

$$V(x) = \frac{k_1}{2} x^2 + \frac{k_2}{2} x^4. \quad (10.61)$$

For small x the potential is nearly harmonic, while for large x the quartic part dominates. We have already considered this problem in an earlier exercise, where we used (or suggested that the reader use) the shooting method to calculate the eigenfunctions. Given these basis functions we can use (10.60) to calculate the expansion coefficients for any desired initial wave packet. The time dependence of the basis functions, which follows from the energy eigenvalues, then leads to the time dependence of the complete wave packet.

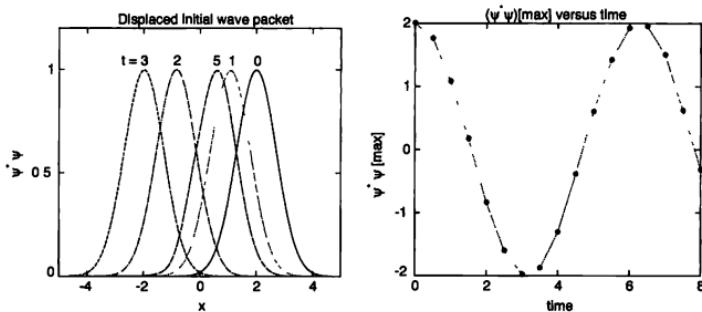


FIGURE 10.29: Left: time evolution of a wave packet with an initial width of $\sigma = 1$, and an initial displacement of $x_0 = 2$ [in (10.54)]. Right: location of the maximum in $\psi^*\psi$ as a function of time. From a “classical” viewpoint this corresponds to plotting the position of the particle as a function of time. The result is simple harmonic motion.

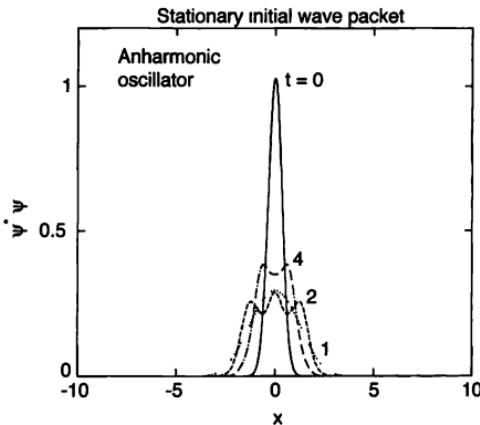


FIGURE 10.30: Time evolution of a wave packet in an anharmonic potential well, with an initial width $\sigma = 0.5$. The potential was given by (10.61) with $k_1 = 1$ and $k_2 = 0.1$. The lowest 32 basis states were used in the calculation.

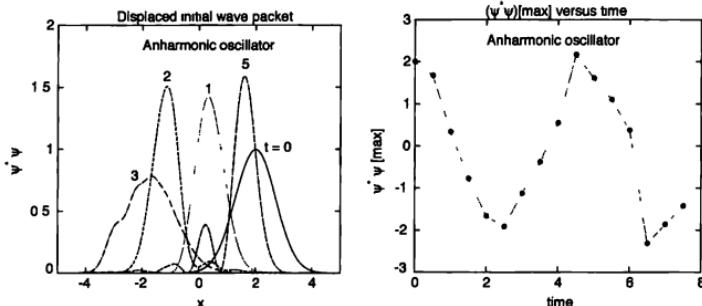


FIGURE 10.31: Time evolution of a wave packet with an initial width $\sigma = 1$ and an initial displacement of $x_0 = 2$, in an anharmonic potential well. The other parameters are the same as in Figure 10.30

Results for a stationary initial wave packet, with $\sigma = 0.5$, are shown in Figure 10.30. This is the same wave packet considered for the harmonic oscillator potential in Figure 10.28. However, the results are very different due to the change in the potential, as we now consider in more detail. While the wave packet remained centered at $x = 0$, its width oscillated with time in a manner reminiscent of the purely harmonic case. However, here the *shape* of the wave packet also changed with time. This was due to the anharmonicity of the potential. For a purely harmonic system the period of oscillation is independent of the amplitude, but this is not true for an anharmonic oscillator. For a potential proportional to x^4 the period *decreases* as the amplitude is increased (see Figure 10.13). This can be understood in terms of the increasing steepness of the potential at large x . As compared with the harmonic case, this confines the particle to a smaller region of space, thereby decreasing the period.

This dependence of the period on amplitude affects the wave packet in the following way. We have already noted that our Gaussian packet, or indeed any other packet, can be thought of as a sum of basis functions with different effective wave vectors; that is, with different velocities, with some of the components moving to the right and others to the left. For the harmonic case the components oscillated with precisely the *same* period, and this preserved the shape of the initial wave packet. However, for the anharmonic case the components have different periods, and this causes them to get out of step with each other as time passes. One result is a distortion of the shape of the wave packet.

This anharmonicity can also be seen in the behavior of a wave packet that is initially displaced from $x = 0$. An example is shown in Figure 10.31 where we plot the time evolution of a packet that was initially centered at $x = 2$. The particle oscillated back and forth in the well but the shape of the packet varied considerably, in contrast to the behavior in the purely harmonic case (Figure 10.29). The time dependence of the location of the maximum in $\psi^* \psi$ also deviated substantially from a purely cosine form.

The relatively simple behavior of a harmonic oscillator is not, in general, found for other potentials. The contrasting results for an anharmonic oscillator illustrate some of these features. These examples also highlight the power of the spectral method and show especially that it can be applied to problems for which analytic results are not possible.

EXERCISES

- 10.20. Consider the harmonic oscillator and calculate the evolution of a wave packet that is initially at $x = 0$ and has a wave vector $k = 0.5$ [see (10.55)] Show that it oscillates, and compare its period with that found when the wave packet is given an initial displacement, but no initial velocity.
- *10.21. Perform a spectral calculation for a harmonic oscillator, and investigate how the results depend on the number of basis states that are used. Contrast the results with 8 and 16 basis functions with the results shown for 32 basis states in Figure 10.29. As a general rule, the basis functions vary more rapidly with position as their energy is increased. These functions must vary at least as rapidly as the initial wave function (if not more so), if they are to be able to accurately represent the time dependence. To see this, perform a calculation with a fixed number of basis states and compare the results for different initial displacements of the wave packet (as in Figure 10.29).
- 10.22. Calculate the (classical) oscillation period for the harmonic oscillator considered in Figure 10.29 and show that it agrees with the observed behavior.

REFERENCES

- [1] J. S. Boilemon, "Computer Solutions to a Realistic 'One Dimensional' Schrödinger Equation," Am. J. of Phys. **40**, 1511 (1972). Describes the shooting method and gives some examples.
- [2] P. L. DeVries, 1994, *A First Course in Computational Physics*, John Wiley & Sons, New York. Contains a nice discussion of the so-called split-operator approach for solving the time-dependent Schrödinger equation.
- [3] I. Galbraith, Y. S. Ching, and E. Abraham, "Two-Dimensional Time-Dependent Quantum-Mechanical Scattering Event," Am. J. of Phys. **52**, 60 (1984). Describes how to extend the time-dependent calculations to more than one dimension.
- [4] A. Goldberg, H. M. Schey, and J. L. Schwartz, "Computer-Generated Motion Pictures of One-Dimensional Quantum-Mechanical Transmission and Reflection Phenomena," Am. J. of Phys. **35**, 177 (1967) A very nice description of a "direct" method for solving the time-dependent Schrödinger equation in one dimension, with many nice pictures.
- [5] D. J. Griffiths, 1995, *Introduction to Quantum Mechanics*, Prentice Hall, Upper Saddle River. A good introduction to the subject.

- [6] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapters 11 and 17.
- [7] L. I. Schiff, 1968, *Quantum Mechanics*, 3d ed., McGraw-Hill, New York. A very useful and complete reference, which is more advanced than the book by Griffiths. Contains derivations of the analytic results discussed in this chapter, along with other useful discussions of subjects such as the variational theorem.
- [8] P. B. Visscher, "A Fast Explicit Algorithm for the Time-Dependent Schrödinger Equation," *Computers in Physics* **5**, 596 (1991).
There are *many* numerical methods for attacking the Schrödinger equation that we did not have space to describe or even mention in this chapter. The following papers describe a few of these approaches.
- [9] M. A. Lee and K. E. Schmidt, "Green's Function Monte Carlo," *Computers in Physics* **6**, 192 (1992).
- [10] P. J. Reynolds, J. Tobochnik, and H. Gould, "Diffusion Quantum Monte Carlo," *Computers in Physics*, November/December, p. 662 (1990).
- [11] J. Tobochnik, G. Batrouni, and H. Gould, "Quantum Monte Carlo on a Lattice," *Computers in Physics* **6**, 673 (1992).
- [12] J. Tobochnik, H. Gould, and K. Mulder, "An Introduction to Quantum Monte Carlo," *Computers in Physics*, July/August, p. 431 (1990).

Vibrations, Waves, and the Physics of Musical Instruments

In this chapter we return to some problems connected with vibrations and waves. Problems of this type occur throughout physics. We encountered them first in our studies of oscillatory motion in Chapter 3, saw them again in connection with wave motion in Chapter 6, and came across them yet again in our discussion of quantum mechanics in Chapter 10. This attention is well justified, since vibrating systems, and the numerical methods used to deal with them are extremely important in physics.

Musical instruments utilize vibrations and waves in variety of different ways, so this chapter will consider several specific issues that are of relevance to musical instruments. However, the methods we will encounter can be applied to many other problems, including lattice dynamics and the vibrations of elastic bodies. To get an idea of the types of physics involved, consider the sketch of a guitar in Figure 11.1. A guitar player sets a string into motion by plucking or strumming, and the resulting string vibrations produce a time dependent force on the bridge and top plate (also called the soundboard) of the guitar. This top plate acts as a sort of loudspeaker, and produces sound waves in the surrounding air. Many stringed instruments, including the piano and the violin, act in basically this way, and the motion of the top plates in guitars and violins has much in common with the vibrations of drums and other percussive instruments. With this in mind, we will consider three problems in this chapter: (1) the motion of strings, as in guitars, pianos and violins, (2) the vibrations of a two dimensional surface, similar to the guitar top plate, and (3) the production of sound waves in a room.

11.1 PLUCKING A STRING: SIMULATING A GUITAR

There are several different types of stringed instruments, and they can be classified according to how the string is excited. In this and the following sections we will consider three different types of string excitation. The numerical treatment of a guitar string given below is based on our work on waves in Chapter 6, so it would

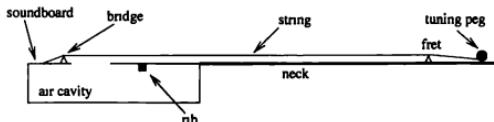


FIGURE 11.1: Basic structure of a guitar

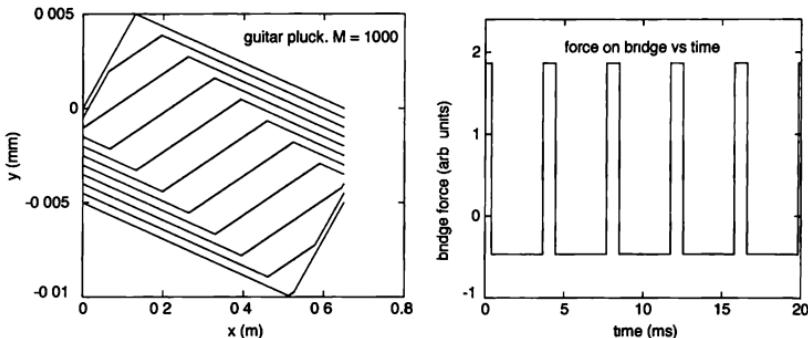


FIGURE 11.2: Left: motion of a plucked guitar string. The initial plucked displacement is plotted at the top. The string profile at successive times is then shown, with these profiles shifted progressively downwards for clarity. For this string the fundamental frequency is $f_0 = 247$ Hz. Here we show results for approximately half of the corresponding period. Right: force on the bridge as a function of time. The parameters used for this calculation were appropriate for the B string of a guitar: $L = 0.65$ m, $T = 149$ N, $c = 320$ m/s, with $\Delta x = 0.65$ mm, and $r = 1$. The plucking point was at $\beta = 1/5$.

be useful for you to review that material before continuing here. Throughout this chapter we will focus on the simple case of an ideal flexible string, with no damping. The wave equation for such a string is

$$\frac{\partial^2 y}{\partial t^2} = c^2 \frac{\partial^2 y}{\partial x^2} \quad (11.1)$$

Here $y(x, t)$ is the (transverse) string displacement, the x direction runs along the string, and c is the wave velocity. For a flexible string, $c = \sqrt{T/\mu}$, where T is the tension and μ is the mass per unit length.

We will use this equation of motion to calculate how the string profile $y(x, t)$ varies with time. However, we also need to know how the string is set into motion. For a guitar string, this is usually a “pluck,” an example of which is shown in the top trace in Figure 11.2. The string is simply “lifted” at one point, so that it has a triangular profile, and then released. This pluck can be described by an amplitude (which will ultimately determine how loud the note sounds), and a plucking location. If the string has a total length L , and it is plucked a distance L_{pluck} from one end (usually the bridge), it is useful to define a “plucking ratio” as $\beta = L_{\text{pluck}}/L$. We will see that the location of the plucking point, i.e., the value of β , affects the sound spectrum in some interesting ways.

To treat the wave equation (11.1) numerically we proceed as we did in Chapter 6, and discretize time and space, with a time step Δt and a spatial step Δx . We then write the string displacement as a function of these discrete variables with $y(x, t) \rightarrow y(i\Delta x, n\Delta t) \rightarrow y(i, n)$. The derivatives in (11.1) can be written in their

usual finite difference forms to get

$$\frac{y(i, n+1) + y(i, n-1) - 2y(i, n)}{(\Delta t)^2} \approx c^2 \left[\frac{y(i+1, n) + y(i-1, n) - 2y(i, n)}{(\Delta x)^2} \right], \quad (11.2)$$

where the \approx sign reminds us that $y(i, n)$ is only an approximation to the true solution. We can now rearrange this to express the string displacement at time step $n+1$ (the “next” time step) in terms of the displacement at previous time steps. The result is (see also Chapter 6)

$$y(i, n+1) = 2[1 - r^2]y(i, n) - y(i, n-1) + r^2 [y(i+1, n) + y(i-1, n)], \quad (11.3)$$

where we define $r \equiv c\Delta t/\Delta x$. We have already discussed the programming associated with (11.3) in Chapter 6, so we refer there for further details on how to implement this method. For all of the calculations we will assume that the ends of the string are rigidly held, so that $y(i=0, n) = y(i=M, n) = 0$, where the ends of the string are at $i=0$ and $i=M$.

Our model of a guitar string as a perfectly flexible and loss-less string is extremely simplified. Nevertheless, we can use this model to understand some of the basic properties of a guitar tone. Let us first consider how a guitar pluck travels back and forth along the string. Figure 11.2 shows the initial plucked displacement, along with snapshots of the string displacement at several later times. For this calculation the string was plucked at a point $L/5$ from one end ($\beta = 1/5$), so there was initially a “kink” at this location. After the string is released, this kink splits into two separate kinks, which propagate towards opposite ends of the string. These two kinks are reflected (with inversion) at the ends, and after one half of a period they reform as an inverted version of the original pluck. This pattern continues, and after one complete period the string is described again by the original plucked waveform. This is what a standing wave really looks like on a guitar string; it is quite different from the simple standing sine-like waves that one normally finds in textbooks.

We now want to understand the sound produced by this vibrating string. In order to give a rigorous description of the sound, we would need to calculate how the soundboard vibrates, and how the soundboard vibrations produce pressure waves in the surrounding air. These are difficult problems that we cannot treat fully in this book (although we will consider some aspects of these problems later in this chapter). Fortunately, there is a relatively simple way to *estimate* the sound. You will recall that one end of the string passes over the bridge (Figure 11.1), and the bridge is firmly attached to the soundboard, so the force of the string on the bridge is transmitted directly to the soundboard. It turns out that the velocity of the soundboard is roughly proportional to this force. Moreover, the sound pressure produced by the vibrating soundboard is approximately proportional to the velocity of the board. We will make use of these two approximations, and thus estimate the sound pressure as a “signal” that is proportional to the force that the string exerts on the bridge. To be precise, we are dealing with the force of the string on the bridge, in the direction perpendicular to the plane of the soundboard. In our calculation, this is the force at the end of the string at $x=i=0$, and is equal to

$F_{\text{bridge}} = T(\partial y / \partial x)$ at this location. In terms of our discrete approximation this is

$$F_{\text{bridge}} = T \frac{\partial y}{\partial x} \Big|_{x=0} = T \frac{[y(1, n) - y(0, n)]}{\Delta x}. \quad (11.4)$$

Given results for the string profile as a function of time, $y(i, n)$, as in Figure 11.2, we can obtain F_{bridge} . Some results are shown in the right in Figure 11.2, and we see that it is simply a square wave. Indeed, F_{bridge} is proportional to the slope of the string at $x = 0$, and we see from the results for $y(i, n)$ that this slope does indeed switch back and forth between only two values.

In order to understand the resulting sound, we next consider the spectrum of the bridge force. We can calculate this spectrum by performing a Fast Fourier Transform analysis of the bridge force signal, and some results are given in Figure 11.3 where we show the power spectrum of a string plucked a fraction $\beta = 1/5$, from one end, corresponding to the bridge force data in Figure 11.2. We see a series of peaks at frequencies $f_1, 2f_1, 3f_1, \dots$. These are just the components of the sound at the fundamental frequency f_1 , and its harmonics nf_1 . Starting from $n = 1$, the magnitude decreases smoothly as n increases, and we find that it seems to vanish at $n = 5$, before increasing again at larger n . This can be understood from the shape of the initial pluck. Here we took $\beta = 1/5$, so the initial pluck had its peak at $x = L/5$. It is straightforward to compute analytically the amplitudes of the various Fourier components (see Fletcher and Rossing, 1998), and one finds that the amplitudes at $n = 1/\beta, 2/\beta$, etc., vanish. These are the harmonics which have nodes at the plucking point. Hence, for our calculation with $\beta = 1/5$, the harmonics at $n = 5, 10, 15$, etc., all vanish.

With this in mind, it is interesting to consider the bridge force spectrum for a different value of β , as shown on the right in Figure 11.3. Here we give results for $\beta = 1/20$, which corresponds to plucking the string much closer to the bridge. When compared with the results for $\beta = 1/5$, there are two important differences. First, the harmonic at which the amplitude first goes to zero is now $n = 20$, as expected for this value of β . Second, the magnitudes of the harmonics decrease much more slowly with n than we found for larger β . Indeed, the second harmonic is now only a few percent smaller than the fundamental. As a result, much more of the sound power is carried by the harmonics. This is why a guitar tone has a pronounced “twang” when when the string is plucked very close to the bridge.

Before we leave this section on guitar strings, we want to consider an aspect of this simulation that has been effectively hidden so far. In all of the results shown above we have used a rather small value of Δx , so that we had a large number of spatial string elements. For the parameters used above we had $M = 1000$ elements. Figure 11.4 shows results for a smaller value, $M = 100$. While the initial string profile was again a “perfect” pluck, the string profile assumes a noticeably jagged appearance after the string is released. This result is an inherent feature of our approximation scheme (11.3). It is straightforward to show (see the exercises) that with a sharp initial pluck, i.e., a pluck whose slope is discontinuous at the plucking point, our numerical scheme will always cause the string profile to evolve in the jagged manner seen in Figure 11.4. This illustrates one of the pitfalls that can

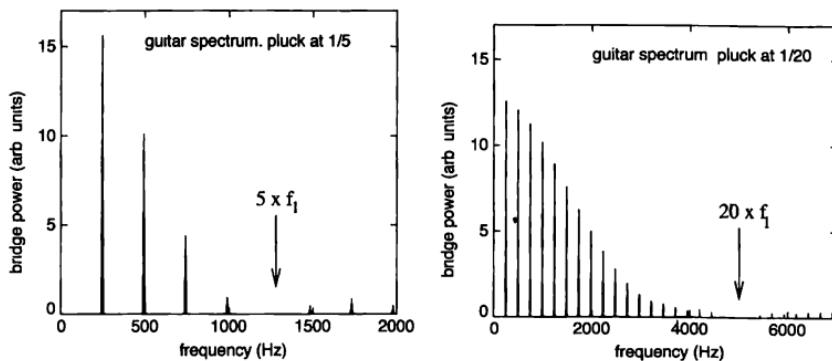


FIGURE 11.3: Spectra of the bridge force for two plucked guitar strings. Left: $\beta = 1/5$. This corresponds to the results in Figure 11.2. Right: the same string, but now plucked much closer to the bridge, with $\beta = 1/20$. Note the difference in the horizontal scales in the two plots.

occur when we try to impose a discontinuous condition (here we have imposed a discontinuous slope at the plucking point) in a numerical calculation.

We should now ask how nature avoids such difficulties, and there are several answers. First, a perfect pluck cannot be imposed on a real string. Real strings always have some stiffness (see Chapter 6), and this will prevent a discontinuity in the slope of $y(x)$. Second, a real player cannot pluck the string at a single point along the string. Instead the corner of the pluck will always be spread over a small range of x , corresponding, e.g., to the width of the guitar pick. These are effects that can be added to the calculation, and when this is done the difficulties seen in Figure 11.4 are eliminated.

EXERCISES

- 11.1. Write a program to simulate a guitar string according to (11.1). Calculate the bridge force spectrum for different values of β , and compare your results to those in Figure 11.3.
- 11.2. Calculate analytically the bridge force spectrum resulting from a plucked waveform, as in Figure 11.3, for different values of β . Show explicitly that the Fourier component at $n = 1/\beta$ is zero.
- 11.3. Consider a string that is plucked a fraction of the string length β from ends. Give a symmetry argument to show why the Fourier amplitude corresponding to the harmonic of order $1/\beta$ must vanish.
- 11.4. Analyze the behavior of $y(i, n)$ near the plucking point for a perfectly sharp pluck, for small n . Show that (11.3) will always produce a “jagged” corner when the string begins with a sharp kink.

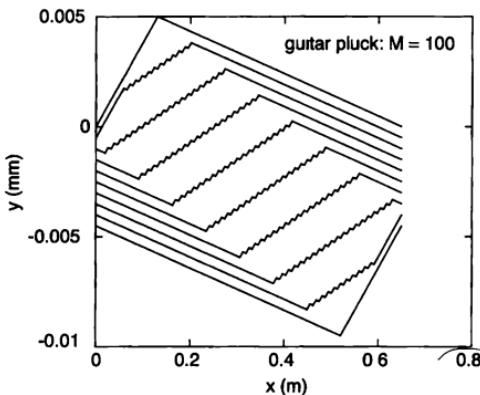


FIGURE 11.4: Simulation of the pluck of a guitar string. The calculation is the same as in Figure 11.2, but $\Delta x = 6.5$ mm, so that there were $M = 100$ string elements. The successive snapshots are again shifted downwards for clarity.

- 11.5. Add damping to our model of a guitar string. Do this by adding a term of the form $-\alpha(\partial y/\partial t)$ to the right hand side of the wave equation (11.1). For a guitar string, $\alpha = 0.5$ is a reasonable value. Calculate the decay time of the string amplitude. *Hint:* You may have to adjust the ratio of the step sizes (the value of the factor r in (11.3)) to maintain numerical stability.
- 11.6. Perform a calculation similar to the one in Figure 11.4, but start with a smoothed initial pluck. That is, smooth out the sharp kink in a “perfect” pluck. Investigate different ways to do this smoothing, and calculate the resulting bridge force spectrum.

11.2 STRIKING A STRING: PIANOS AND HAMMERS

In this section we want to consider a different way of setting a string into motion. We imagine that the string is initially at rest, with $y(x) = 0$ for all x . We then strike the string with a “blunt object;” our goal here is to simulate the operation of a piano. In a real piano, the blunt object is a “hammer,” which is basically a felt covered mallet. The overall layout is shown in Figure 11.5. The strings in a grand piano run horizontally, with one end fastened to a rigid support, and the other end in contact with a soundboard. The motion of the string results in a force on the soundboard, and the vibrating soundboard then acts like a large speaker and produces sound (similar to the case of a guitar). We now want to investigate how to model the hammer-string collision, and how this collision gives rise to some important features of a piano tone.

Experimental studies of piano hammers have shown that they act as nonlinear

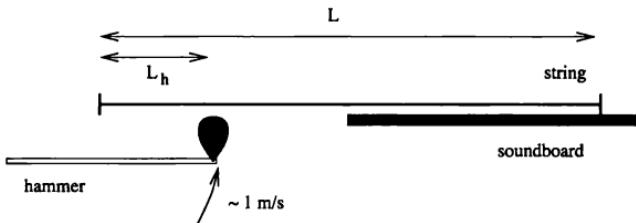


FIGURE 11.5: General layout of a piano string, soundboard, and hammer.

springs. Let z_f be the amount that this spring is compressed; i.e., z_f is the amount that the felt covering of the piano hammer is compressed as it collides with a string. One can then approximate the hammer-string force by

$$F_h(z_f) = K z_f^p. \quad (11.5)$$

Here K is an effective stiffness constant, while the exponent p measures the strength of the nonlinearity. Experiments show that $p \approx 3$ generally gives a reasonable description of real piano hammers.

Our treatment of the motion of a piano string is similar to the approach that we employed for the guitar. We start again with (11.1), and the numerical approach developed in (11.3). However, there is an important difference in the way that a piano string is excited. For the guitar the string is displaced prior to $t = 0$ (i.e., the start of the tone), and then allowed to move freely according to the equation of motion (11.1). Hence, the initial conditions were a given string displacement (a “pluck”), with an initial string velocity of zero. A piano string is also at rest initially, but with $y = 0$ everywhere. The hammer then strikes at $t = 0$, and remains in contact for a short period of time. During this contact time, there is an “external” force (11.5) acting on the string, so we now have consider how to add this force to our equation of motion.

The hammer makes contact with only a short segment of the string, so let us consider the string as a series of discrete elements labeled by i , with displacements $y(i)$. Each of these string elements will have a length Δx and mass $\mu \Delta x$, where μ is the mass per unit length. If we multiply both sides of (11.1) by the factor $(\mu \Delta x)$ we have

$$(\mu \Delta x) \frac{\partial^2 y(i)}{\partial t^2} = (\mu \Delta x) c^2 \frac{\partial^2 y(i)}{\partial x^2}. \quad (11.6)$$

You will now see that the left hand side of (11.6) is just the mass times the acceleration of string element i , and this should make you suspect that the right hand side is the force on this element. That is indeed the case. The right hand side of (11.6) is the transverse component of the tension force on element i from the two adjacent elements of the string.¹

¹See also Chapter 6 and the discussion connected with Figure 6.1

We can now see how include an externally applied force, such as the hammer force; we simply add it to the right hand side of (11.6). Our equation of motion for a piano string is then

$$(\mu\Delta)\frac{\partial^2 y(i)}{\partial t^2} = (\mu\Delta x)c^2\frac{\partial^2 y(i)}{\partial x^2} + F_h , \quad (11.7)$$

where F_h is the hammer force, which acts only at localized region (i.e., only in a particular range of i) along the string (see Figure 11.5).

As a final step we must discretize time, and then [following (11.2)] write both derivatives in (11.7) in finite difference form. We have

$$\begin{aligned} & (\mu\Delta x)\frac{y(i, n+1) + y(i, n-1) - 2y(i, n)}{(\Delta t)^2} \\ &= (\mu\Delta x)\left[\frac{y(i+1, n) + y(i-1, n) - 2y(i, n)}{(\Delta x)^2}\right] + F_h . \end{aligned} \quad (11.8)$$

We now rearrange to get the string displacement at time step $n+1$ as a function of the displacement at previous times. The result is

$$\begin{aligned} y(i, n+1) &= 2[1 - r^2]y(i, n) - y(i, n-1) \\ &\quad + r^2[y(i+1, n) + y(i-1, n)] + \frac{(\Delta t)^2}{\mu\Delta x}F_h , \end{aligned} \quad (11.9)$$

where we have again defined $r \equiv c\Delta t/\Delta x$. This is simply a generalized version of (11.3), that includes a force applied at localized spots along the string

We next need an equation of motion for the hammer. The mechanics of the piano is designed so that the hammer is accelerated rapidly when the key-lever is depressed, and is then released from the lever mechanism just before it strikes the string.² We can thus treat the hammer as a simple mass that is moving freely with a velocity v_h just prior to making contact with the string. Newton's second law for the hammer thus has the form

$$m_h a_h = -F_h , \quad (11.10)$$

where a_h is the hammer acceleration and m_h is its mass, and the force on the hammer is given by (11.5). We can use the Euler method to calculate the position of the hammer z_h , and its velocity v_h , at time step $n+1$ with the result

$$\begin{aligned} z_h(n+1) &= z_h(n) + v_h(n)\Delta t , \\ v_h(n+1) &= v_h(n) + a_h(n)\Delta t . \end{aligned} \quad (11.11)$$

The hammer-string force depends on the compression of the felt z_f . This compression is equal to the difference between the hammer position z_h and the string displacement where the hammer string collision is centered. Real hammers do not meet the string at a single point, and it is customary in such simulations to

²The mechanism is called an "escapement."

spread out the hammer force, and distribute F_h over several string units i . In the results below we have spread this force over three spatial units, but the results are not terribly sensitive to how this spreading is done.³

The programming of the string (11.9) plus hammer (11.11) follow methods that we have developed in previous problems (see Chapters 2 and 6), and the pseudocode for the calculation is shown below.

EXAMPLE 11.1 Pseudocode for subroutine piano_string

- For each time step n :
 - ▷ Find the hammer-string force using (11.5), with $z_f = y_{\text{contact}}(n) - z_h(n)$. Here $y_{\text{contact}}(n)$ is the displacement of the string at the hammer-string contact point.
 - ▷ Note that $F_h = 0$ after the hammer loses contact with the string - this occurs when z_f becomes negative.
 - ▷ Calculate the new string profile using (11.9).
 - ▷ Calculate the new hammer velocity and position using (11.11).

The calculation begins with the string initially at rest, with $y(i, 0) = 0$ everywhere. The hammer is then set into motion when the performer presses the key. The hammer velocity just prior to making contact is typically in the range 0.5–4 m/s, and we take this as the initial velocity of the hammer $v_h(0)$ in our simulation. The hammer then collides with the string at $t = 0$, setting the string into motion and causing the hammer to rebound and eventually lose contact with the string. This collision takes a few ms, after which the string vibrates freely.

Figure 11.6 shows some results for the note middle C on a small size grand piano. Here the initial hammer velocity is $v_h = 0.5$ m/s, which would produce a very soft note. The plot on the left in Figure 11.6 shows how the hammer-string force varies with time. We see that F_h rises fairly smoothly with time, and then falls to zero after about 3 ms. The slight undulations in F_h are not artifacts of the calculation, but are real and can be understood in the following way. When the hammer first hits the string it excites initial waves that travel away from the hammer in both directions. These waves travel to the ends of the string and are reflected back to the hammer before the hammer loses contact with the string. Since these waves are inverted on reflection, they return to the hammer in an inverted form, and cause an extra and sudden compression of the hammer felt. This is the source of the undulations in F_h .

In our work in the previous section with guitar strings, we saw that as a first approximation, the sound produced by instrument is proportional to the force on the bridge. The same is true for the piano. With this in mind, Figure 11.6 also shows the variation of the force on the bridge with time. There is a large initial

³In the results shown below we have applied a force $0.5F_h$ to the central contact point $y(i)$, and $0.25F_h$ to points $i \pm 1$

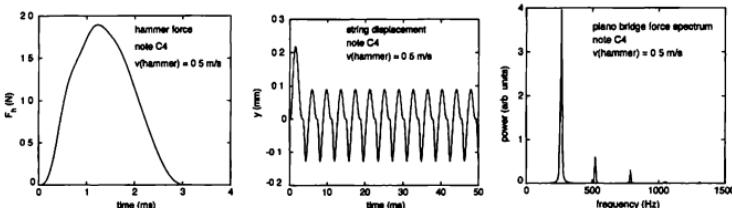


FIGURE 11.6: Results for the note middle C on a medium size grand piano. The plot on the left shows the variation of the hammer-string force with time. In the middle we plot the bridge force as a function of time, while the spectrum of the bridge force is shown on the right. The string parameters used in the calculation were $L = 0.62\text{ m}$, $T = 650\text{ N}$, and $f = 262\text{ Hz}$, while for the hammer we took $K = 1.0 \times 10^{11}\text{ N/m}^{1/3}$, $p = 3$, and $m_h = 3.3\text{ g}$. The hammer struck the string at a distance $L/8$ from the end.

“pulse” on the bridge, but after a few milliseconds the force varies periodically with time. This initial pulse occurs when the hammer is in contact with the string, and produces the characteristic “attack” portion of the tone. After the hammer loses contact, the string moves freely according to the wave equation, so the behavior must then be perfectly periodic. However, while the string motion is periodic, it does have an interesting spectral content. Indeed, it is clear from the plot of the bridge force that this variation is not a simple sinusoid at the fundamental frequency. This is shown also in the plot of the bridge force spectrum on the right in Figure 11.6. While most of the power is at the fundamental frequency, there is substantial power also at the second ($2f_1$) and third ($3f_1$) harmonics. For example, the power at the second harmonic is smaller than at the fundamental by about a factor of 6. This plot shows the power at each frequency, so the *amplitude* of the second harmonic is smaller by about $\sqrt{6} \sim 2.5$. These results for the spectral content are very important, since it is this combination of harmonics that produces the characteristic piano tone.

We next consider how the results change when the note is played more loudly; that is, when we use a larger initial hammer velocity. One might think that if $v_h(0)$ is increased by, e.g., a factor of 2, then the amplitude of the resulting string vibrations would simply be increased by the same factor. If this were true, then the peaks in the resulting power spectrum would be scaled up by a factor of 4 (since power is proportional to the square of the amplitude), but the relative heights of the fundamental and the harmonics would be unchanged. We will now see that this simple expectation does *not* occur.

Figure 11.7 shows results for our note middle C, but now played with a much larger initial hammer velocity. Here we took $v_h(0) = 4\text{ m/s}$, which corresponds to a very loud note. The only difference between these results, and those in Figure 11.6, is that the hammer velocity was larger by a factor of 8. Comparing the results for the hammer force, it is clear that the qualitative *shape* of the curve is quite different. The undulations produced by the reflected waves are much more pronounced, and the hammer-string contact time is reduced to about 2 ms. The bridge force as a

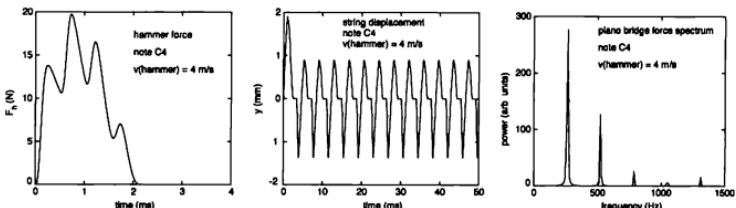


FIGURE 11.7: Results for a loud piano tone. The calculation was similar to the one shown in Figure 11.6, but with $v_h(0) = 4 \text{ m/s}$

function of time has a different appearance, as the harmonics are now much larger in comparison to the fundamental.

These differences between a loud and soft note are due to the *nonlinear* nature of the hammer-string force (11.5). Because of this nonlinearity, doubling the felt compression produces a much larger increase in the force. This larger force causes the hammer to rebound more quickly, reducing the hammer-string contact time. This shorter impulse acting on the string causes the higher frequency harmonics to be excited more strongly, relative to the fundamental. These effects are very important from a musical perspective. They are the reason that a loud tone has a different tone color than a soft one. A listener can readily tell the difference, through the increased (relative) size of the harmonics. The same is true for many other musical instruments. Nonlinearities give rise to important changes in the harmonic content as the overall loudness is varied. Loud musical notes simply sound different than soft ones.

EXERCISES

- 11.7. Show that the right hand side of (11.6) is indeed equal to the transverse component of the tension force on string piece i from adjacent pieces.
- 11.8. Repeat the calculation of a loud and soft piano tone as in Figures 11.6 and 11.7, but now consider the note two octaves below middle C. Use the following string parameters: $L = 1.06 \text{ m}$, $T = 349 \text{ N}$, $K = 6 \times 10^{10} \text{ N/m}^{1/3}$, $p = 3$, $m_h = 4.3 \text{ g}$, $f = 65 \text{ Hz}$.

11.3 EXCITING A VIBRATING SYSTEM WITH FRICTION: VIOLINS AND BOWS

In this section we consider yet another way that a musical instrument string can be set into motion. We now want to treat the case of an instrument such as a violin, in which the string is rubbed by a bow. The general bowing process is shown in Figure 11.8. One end of the violin string passes over a bridge (as with the guitar), while the other is held in place by the player's finger. The bow consists of strands of horsehair (or a similar material) that are held taut by the frame of the

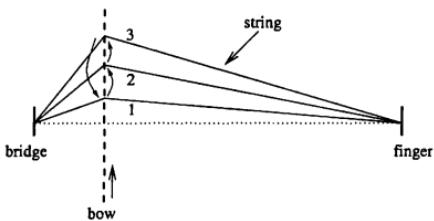


FIGURE 11.8: Stick-slip motion of a violin string as it is bowed. The vertical dashed line is the bow as it moves upwards, it exerts a frictional force that pulls the string upwards, (1 → 2 → 3) Periodically the string slips back down (3 → 1), and the cycle is repeated, as the string moves from 1 → 2 → 3 → 1

bow. The player “rubs” the bow across the string as indicated very schematically in Figure 11.8.

In the simplest case, the player moves the bow at a constant velocity that we will denote as v_{bow} . If the player is sufficiently skilled, the string then moves as follows. Initially the frictional force between bow and the string causes the string to stick to the bow, so the point on the string that is in contact with the bow moves with a velocity v_{bow} . Over time this pulls the contact point farther and farther in the direction of the bowing. Eventually the tension from the string will overcome the frictional force, and the string will then slip back. After a short while the bow recaptures the string and pulls it along until the string slips again, etc.

The motion of a bowed string is thus an example of a “stick-slip” process, and to model this motion we will have to deal with alternations between static and kinetic friction between the bow and the string. However, before we proceed to this simulation it is worthwhile to think a bit about the following question: “How is the frequency of bowed string motion related to the frequency of the same string when it is excited by simply plucking it?” The answer to this question is that these two frequencies are the *same*. Violin players often take this for granted, but a physicist should not, for the following reason. We understand the vibrations of a plucked string in terms of the normal modes of a *freely* moving string; that is, a string that is not subject to any external forces, and is thus described by the wave equation (11.1). The resulting normal modes are just the usual standing waves on a string that is held fixed at the ends, with a fundamental frequency of $f_1 = c/(2L)$, where c is the wave speed and L is the length of the string. However, when a string is bowed, there is a *substantial* external force acting on the string due to the bow at *all* times. It is thus *not* obvious that the fundamental vibration frequency of a bowed string will be the same as that of a free string.

The answer to this puzzle was first given by Helmholtz. He showed theoretically that one of the possible motions of a bowed string is a periodic vibration with frequency f_1 . However, unlike the case with a free string, there are many other possible periodic motions, with frequencies that are not simple multiples of f_1 . Let us now see qualitatively how a bowed string can vibrate at the frequency f_1 . We

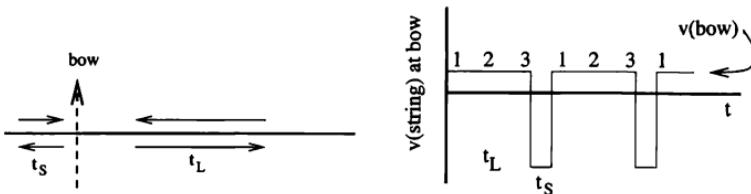


FIGURE 11.9: Left waves excited by the bow are reflected by the ends and cause the string to either begin slipping, or to resume sticking, relative to the bow. Right: string velocity corresponding to the string profiles labeled 1, 2, and 3 in Figure 11.8

assume that it is bowed at a point that is a distance βL from the left end of the string in Figure 11.9, where for simplicity we assume (following Helmholtz) that β is a rational fraction (such as 1/8). When the bowing parameters (i.e., the bow velocity, and the normal force between the bow and the string) are in the proper range, the following steady state motion is possible. Suppose that the bow is “captured” by the string at time $t = 0$; hence, just after this moment the string moves with a velocity v_{bow} , as sketched in Figure 11.9. This abrupt transition to sticking excites waves that travel away from the bow, towards both ends of the string, as indicated schematically on the left in Figure 11.9. These waves are reflected from the ends, and return after a time t_S (from the near end) and t_L (from the far end). These reflected wave pulses result in a sudden additional force between the string and the bow, and if this additional force is large enough, it can cause the string to slip from the bow. In the case we imagine here, we suppose that the reflection at t_L causes the string to slip from the bow. This sets up new wave pulses, and if they have the proper strength the next reflection at t_S will cause the bow to recapture the string, so that we are back in the stick phase of the motion. The slipping time t_S is equal to the length of the short end of the string divided by c , so $t_S = 2\beta L/c$. In the same way $t_L = 2(1 - \beta)L/c$. The period for this stick-slip motion is then $t_S + t_L$, and its frequency

$$f_{\text{bow}} = \frac{1}{t_S + t_L} = \frac{1}{2\beta L/c + 2(1 - \beta)L/c} = \frac{1}{2L/c} = f_1. \quad (11.12)$$

Hence, the frequency of this bowed motion is precisely the *same* as that of the free string! This is essentially the argument that was given by Helmholtz, and this periodic vibration is called Helmholtz motion.

In order to simulate Helmholtz motion we must have some knowledge of the frictional force between the bow and the string. Here we will make the simplest possible assumption about this force. We assume that there are two coefficients of friction, one when the string is sticking, μ_S , and one when the string is slipping, μ_K . Furthermore, we assume that μ_K is independent of the velocity of the string relative to the bow. In order for stick-slip motion to occur, we require that $\mu_S > \mu_K$.

The simulation of Helmholtz motion is similar to our work on the piano string. We can again use the equation of motion (11.8), but with the force F_h replaced by

the frictional bow-string force. We are thus led again to (11.9), which we can use to calculate the time dependence of the string profile $y(i, n)$. The only difference between this simulation and that of the piano string, is that the magnitude of the force on the string depends on whether it is slipping or sticking. When the string is slipping the force has a magnitude

$$F_{\text{slip}} = \mu_k N, \quad (11.13)$$

while if the string is sticking we know only that the force is *no greater than* $\mu_s N$, hence

$$F_{\text{stick}} \leq \mu_s N. \quad (11.14)$$

One of the tasks in the simulation is to determine the precise value of F_{stick} , given the upper limit in (11.14). The pseudocode for this calculation is given below.

EXAMPLE 11.2 Pseudocode for subroutine bowed_string

- Let the velocity of the bow be v_{bow} , and the string displacement at the bowing point be $y(i_{\text{bow}}, n)$ where n is the time step.
 - ▷ Start with the string in the **sticking state**, with the string velocity at the bowing point equal to v_{bow} .
 - ▷ Advance one time step and assume that the string continues to **stick**: $y(i_{\text{bow}}, n+1) = y(i_{\text{bow}}, n) v_{\text{bow}} \Delta t$.
 - ▷ Use (11.9) to calculate the value of the force required to obtain $y(i_{\text{bow}}, n+1)$. If the required force is less than the limit in (11.14), then the string continues in the **sticking mode**.
 - ▷ If the required force is larger than (11.14), then the string goes into the **slipping mode** and the force is given by (11.13). One must then recalculate $y(i_{\text{bow}}, n+1)$ using (11.9) with the force F_{slip} .
 - ▷ The motion of the rest of the string is calculated with (11.9) with an external force of zero.
 - ▷ Repeat this procedure for the desired number of time steps.

Some results for such a simulation are shown in Figure 11.10, where we show the string displacement at the bowing point, $y(i_{\text{bow}})$, as a function of time, starting with the string in the sticking state at $t = 0$. The string is seen to oscillate in a very “noisy” manner; the tone produced by this string would probably not be very pleasing to listen to. While the string does appear to alternate between periods of sticking (when y_{bow} is increasing) and slipping (when y_{bow} is decreasing), this is certainly not the simple Helmholtz motion that we described above.

To understand this result, it is useful to recall what a real violin sounds like when played by an unskilled student. We know that a beginning player will usually produce a collection of very unmusical squeaks and other painful noises. Indeed, it takes much skill to produce a pleasing musical tone. This tells us that the bowing

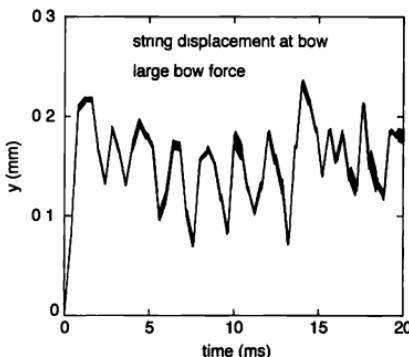


FIGURE 11.10: Results for the stick-slip motion of a hypothetical violin string. The parameters used here are $L = 0.3$ m, $\Delta z = 0.6$ mm, $c = 300$ m/s, $r = 1$, $\mu_S = 1.2$, $\mu_K = 0.35$, $v_{bow} = 0.2$ m/s, and $N = 0.65$ N. These values for the string parameters and bow velocity are roughly appropriate for a real violin string.

parameters must be carefully adjusted to produce the ideal Helmholtz motion. In our qualitative explanation of Helmholtz motion, we argued only that this is a *possible* motion of a bowed string. It turns out that there are *many other* possible periodic motions,⁴ and it is very difficult to know in advance which ones will be most stable. The simulation in Figure 11.10 has involved no adjustment of the bowing parameters, so it should not be surprising that the result is similar to that of a beginning player. In fact, if we change the normal force to a slightly smaller value, we find the results shown in Figure 11.11. Here we find an approximate stick-slip motion, and the frequency is close to the expected value.

We have not performed an exhaustive exploration of the bowing parameters for this simple model of a bowed string; we will let you do that in the exercises. However, it seems likely (although we know of no rigorous proof) that the simple friction model that we have used cannot produce perfect long-term Helmholtz motion. This is the likely origin of the rapid oscillations at the bottom of the stick-slip cycle in Figure 11.11. The best we can probably expect in this case is to observe approximate Helmholtz behavior, as seen in Figure 11.11. To do a better job of simulating a real violin string we would have to use a more realistic model of the frictional force. It is known that in the sliding regime, the frictional force is a function of the velocity of the string relative to the bow, so we would need to use a velocity dependent friction force. That is a job that we will leave for the exercises.

⁴Interestingly, another famous physicist, A. Raman, gave one of the first and most general descriptions of these many possible motions

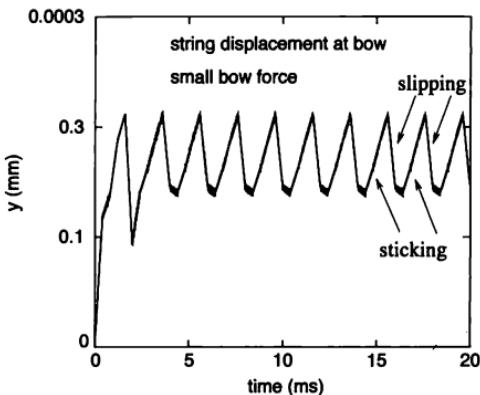


FIGURE 11.11: Simulation of the violin string in Figure 11.10, but with a smaller value of the normal force between the bow and the string. Here we took $N = 0.5 \text{ N}$. Periods of sticking and slipping are indicated

There are several conclusions that can be drawn from our study of bowed strings. First, Helmholtz motion is an extremely intriguing type of periodic motion. Violin string vibrations are much more interesting (in terms of physics) than one might have thought. Second, the simulation of systems with frictional forces poses complications not found in systems described by continuous forces.

EXERCISES

- 11.9. Perform the violin string calculation and study the stick-slip motion, as in Figures 11.10 and 11.11. Explore the behavior for different bowing parameters, including different values of v_{bow} , N , and the coefficients of friction μ_s and μ_K .
- *11.10. Repeat the violin string simulation using a more realistic form for the frictional force. Start by exploring the behavior with a friction function such as

$$\mu_K = \mu_1 \exp(-v/v_1) + \mu_2, \quad (11.15)$$

with $\mu_1 = 0.55$, $\mu_2 = 0.35$, and $v_1 = 0.1 \text{ m/s}$. Note that here $\mu_s = \mu_1 + \mu_2$.

11.4 VIBRATIONS OF A MEMBRANE: NORMAL MODES AND EIGENVALUE PROBLEMS

Consider the one-dimensional system of masses connected by springs in Figure 11.12. We have actually seen this system in Chapter 6, where it was shown (in one of the exercises) that it is closely analogous to a vibrating string. In the limit of a very large

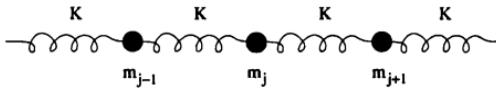


FIGURE 11.12: Model for a vibrating one-dimensional system. The masses interact via springs — i.e., Hooke's law forces.

number of masses with a very small spacing between them (the *continuum limit*), this system of masses and springs actually maps directly to the one-dimensional wave equation and can be solved using the methods introduced⁵ in Chapter 6. However, in this chapter we want to reconsider this system, and use it as a prototypical example of how to deal with more complex types of vibrating systems. You might think that a system connected by simple springs is quite special, but this is not at all the case. In fact, these springs simply represent Hooke's law type forces, and such force laws will almost always be found when the vibration amplitude is small. Hence, simple springs — which represent Hooke's law forces — are a quite general way to model a wide variety of interactions between particles.⁶

We begin by considering compressional vibrations in the mass-spring system in Figure 11.12. We let x_j be the displacement of mass j from its equilibrium position, and assume that this displacement is along the chain direction in Figure 11.12. For simplicity we also assume that all of the particles are of equal mass. The force on particle j from particle $j + 1$ is then $K(x_{j+1} - x_j)$, where K is the spring constant (assumed to be uniform throughout the system). There is also a force $K(x_{j-1} - x_j)$ from particle $j - 1$. These forces apply within the system, away from the ends. If the system contains N masses, we must deal also with the ends at $j = 1$ and $j = N$. The ends are usually dealt with in one of two ways: periodic boundary conditions, which essentially just connect the ends together, or fixed boundary conditions, which connect m_1 and m_N to rigid points via springs. We will use fixed boundary conditions, as these are more appropriate for many real systems, such as musical instruments.

⁵We have also encountered similar methods in connection with random walks in Chapter 7.

⁶A more serious approximation that we are making here is that the force between two adjacent masses is directed along the “bond” direction. Most real systems also have so-called bond-bending forces, and these are important in determining the shear response. Fortunately, a similar method, with a dynamical matrix, etc., can be used to deal bond-bending forces.

Using Newton's second law, we can write an equation of motion for each mass⁷

$$F_1 = m \frac{d^2x_1}{dt^2} = -Kx_1 + K(x_2 - x_1) \quad (11.16)$$

$$F_2 = m \frac{d^2x_2}{dt^2} = K(x_1 - x_2) + K(x_3 - x_2)$$

$$F_j = m \frac{d^2x_j}{dt^2} = K(x_{j-1} - x_j) + K(x_{j+1} - x_j)$$

$$F_N = m \frac{d^2x_N}{dt^2} = K(x_{N-1} - x_N) - Kx_N .$$

If the system is vibrating at a particular angular frequency ω , we can then write the displacement of each mass as

$$x_j = A_j e^{-\omega t}, \quad (11.17)$$

where A_j is the (complex) vibrational amplitude of mass j . Note that (11.17) is a completely general way to write the solution of (11.16). One of our jobs is to find the allowed frequencies of vibration. Since our system has N degrees of freedom, there will be N different solutions and hence N different possible vibrational frequencies. Hence, there will be N different solutions for ω , which we can label ω_m with $m = 1$ to N .

If we insert (11.17) into (11.16), we can write the equations of motion in the very suggestive matrix form

$$\frac{m\omega^2}{K} \begin{pmatrix} A_1 \\ A_2 \\ \dots \\ A_j \\ \dots \\ A_N \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & \dots & 0 \\ \dots & & & & & \\ 0 & & 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} A_1 \\ A_2 \\ \dots \\ A_j \\ \dots \\ A_N \end{pmatrix} \quad (11.18)$$

or

$$\frac{m\omega^2}{K} \vec{A} = \mathcal{D} \cdot \vec{A} \quad (11.19)$$

where \vec{A} is a column vector whose components are the A_j and \mathcal{D} is the $N \times N$ matrix in (11.18). \mathcal{D} is called the *dynamical matrix* of the system. The dynamical matrix for our one-dimensional system has a particularly simple tridiagonal form, with the nonzero entries all within one unit of the diagonal.⁸

We can now see that we are dealing with an *eigenvalue problem*, and can appeal to various theorems of linear algebra to show that (11.19) will in general have N solutions. For each of these solutions there will be a value of ω (the eigenvalue)

⁷Note that this system of equations reduces to the wave equation in the limit $\Delta x \rightarrow 0$.

⁸We saw matrices with a similar structure in Chapter 7

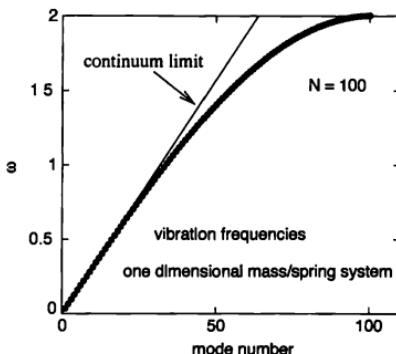


FIGURE 11.13: Eigenfrequencies for the one-dimensional mass-spring system in Figure 11.12. The line shows the exact solution in the continuum limit (as would be appropriate for a string).

and an associated vector \vec{A} (the eigenvector). You should also be able to see that the equations of motion for *any* system of masses and Hooke's law springs can be written in the form (11.19). While the dynamical matrix will usually be more complicated than the simple tridiagonal form in (11.18), we are still left with an eigenvalue problem.

For nearly all of the numerical problems in this book we have encouraged you to write your own programs. However, we do *not* suggest it in this case. Many books have been written on the numerical solution of eigenvalue problems, and much effort has been devoted to writing efficient code for attacking this problem. We strongly recommend that you make use of the matrix capabilities of a program such as Matlab or Mathematica, or a similar numerical package.

Returning to our one-dimensional problem, we are now faced with finding the eigenvalues (vibration frequencies) and eigenvectors (vibration amplitudes) of the dynamical matrix \mathcal{D} . We have carried out this calculation using Matlab, using the values $m = 1$ and $K = 1$, for a system with $N = 100$ masses, and the results for the vibration frequencies are shown in Figure 11.13. This is an easy calculation in Matlab, since this software package (and others like it) are structured to make this type of matrix calculation extremely straightforward. After setting up the appropriate entries in the dynamical matrix, computing the eigenvalues and eigenvectors requires only a single command.

Now that we have the vibrational frequencies, our next problem is to understand the results, and this involves an understanding of the vibrational patterns, i.e., the eigenvectors A_j for the different modes. For the lowest modes, it is simplest to recall the analogy with a vibrating string. We know that the normal modes of a string with fixed ends will be simple standing waves with nodes at each end. These

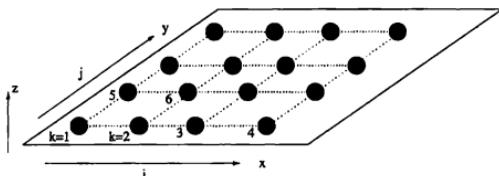


FIGURE 11.14: A discrete model of a thin membrane, involving a system of masses connected by springs (the dotted lines). The numbers show a labeling scheme that can be used to construct the dynamical matrix

standing waves have wavelengths $\lambda_m = 2L/m$, where L is the length of the system, and m is the mode number ($m = 1$ to N). The corresponding (angular) frequencies are then $\omega_m = 2\pi c/\lambda_m$, where c is the speed of a wave on a string. We will leave it as an exercise⁹ to show that our choice of the parameters $m = 1$ and $K = 1$, leads to $c = 1$, $\Delta x = 1$, and $L = (N + 1)\Delta x = N + 1$. We have already noted that in the continuum limit, our mass-spring system maps exactly to a vibrating string. In this limit we know that the wave speed is a constant, independent of the frequency. So, in this limit of low frequencies/long wavelengths in Figure 11.13 we expect

$$\omega_m \approx \frac{2\pi c}{\lambda_m} = \frac{m\pi}{N + 1}. \quad (11.20)$$

Comparing this with Figure 11.13, we find very good agreement. For the highest modes, the analogy with waves on a string breaks down, and the appropriate picture is then one of lattice dynamics. For more on that subject we refer the reader to any introductory book on solid state physics. The highest frequency modes then correspond to vibrations in which adjacent particles vibrate out of phase. The group velocity for such vibrations (which is just the slope of the curve in Figure 11.13) is zero for the highest mode.

We next turn to a problem involving vibrations in two dimensions. In order to keep with our musical instrument theme, we consider the vibrations of a thin membrane, as would be found in a drumhead or a banjo. The equation of motion for a thin membrane that is subject to a tension T is

$$\frac{\partial^2 z}{\partial t^2} = \frac{T}{\sigma} \left(\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2} \right), \quad (11.21)$$

where σ is the mass per unit area of the membrane. Here the undisplaced membrane lies in the $x - y$ plane, and z is the displacement of the membrane with respect to this plane. This is just the wave equation in two dimensions, and the wave speed is $c = \sqrt{T/\sigma}$.

We proceed by discretizing the membrane into two-dimensional elements of size $\Delta x \times \Delta y$, and then use (11.21) to derive the equation of motion of each element.

⁹See Fletcher and Rossing, 1998

It is useful to define a labeling scheme for each of these elements, and it is tempting to label each element using two indices in a row/column arrangement. However, our goal is to construct a dynamical matrix, and each row and column in this matrix will correspond to a single membrane element. For this reason, it is most convenient to use a single index, k , to label each element of the membrane, and a convenient labeling scheme shown in Figure 11.14. According to this scheme, the index k of an element of the membrane is given by

$$k = i + Nj . \quad (11.22)$$

For simplicity we take $\Delta x = \Delta y$, and assume a square $L \times L$ membrane. Since there are N vibrating elements of the membrane with the adjacent edges held fixed, $L = (N+1)\Delta x$.

The equation of motion (11.21) for the displacement of a particular element k can then be written as

$$(\sigma \Delta x \Delta y) \frac{\partial^2 z_k}{\partial t^2} = T(z_{k+1} + z_{k-1} + z_{k+N} + z_{k-N} - 4z_k) . \quad (11.23)$$

This is similar to the equation of motion in the one-dimensional case (11.16), as each z_k is “linked” to the adjacent elements. That is, the motion of each element k is due to forces from neighboring elements; here there are four neighboring elements of the membrane. Following (11.17) the formal solution is $z_k = A_k e^{-\omega t}$. Inserting this into (11.23) leads to

$$\frac{\sigma \Delta x \Delta y}{T} \omega^2 \vec{A} = \mathcal{D} \cdot \vec{A} , \quad (11.24)$$

where \vec{A} is a vector that is composed of elements A_k . The dynamical matrix is now given by

$$\mathcal{D} = \begin{pmatrix} 4 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & \dots & -1 & -1 & 0 & \dots & 0 \\ 0 & -1 & 4 & -1 & \dots & 0 & -1 & -1 & \dots & 0 \\ \dots & & & & & & & & & \\ 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 \end{pmatrix} . \quad (11.25)$$

In general, each row and column of \mathcal{D} will have 5 nonzero entries, with the simple pattern implied by (11.24). The rows and columns of \mathcal{D} that correspond to the edges of the membrane will have fewer nonzero entries.

Now that we have constructed the dynamical matrix, it is straightforward to compute the eigenvalues and eigenvectors using a package such as Matlab. The results for the eigenvalues of \mathcal{D} are shown in Figure 11.15. Note that \mathcal{D} as defined in (11.25) is dimensionless; the eigenfrequencies ω_k of our square membrane are equal to the eigenvalues in Figure 11.15 multiplied by the factor $\sqrt{T/(\sigma \Delta x \Delta y)}$.

The exact solution for a square membrane is well known.¹⁰ The eigenfrequencies are

$$\omega_{mn} = \pi \sqrt{\frac{T}{\sigma}} \sqrt{\frac{m^2}{L^2} + \frac{n^2}{L^2}} , \quad (11.26)$$

¹⁰See, e.g., Fletcher and Rossing, 1998.

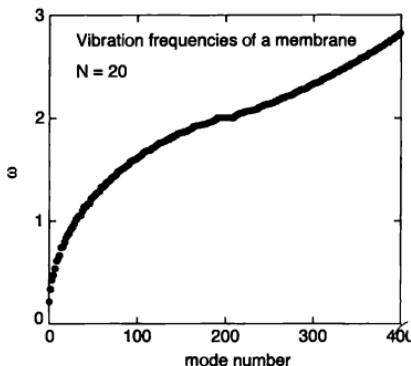


FIGURE 11.15: Eigenvalues of the dynamical matrix for a thin membrane (11.25). Here we have taken the number of discrete elements to be $N \times N$, with $N = 20$, so there are $N^2 = 400$ modes

where m and n are integers, with $m = 1, 2, \dots$, and $n = 1, 2, \dots$. These modes corresponding to standing waves that are similar in form to the standing waves we encountered for the one-dimensional case (11.20). The only difference is that the normal modes are now a product of a standing wave along x with a standing wave along y . In Figure 11.15 we have plotted the eigenfrequencies as a function of “mode number.” In fact, the modes have merely been ordered according to the value of ω_{mn} . The simplest way to check that our calculated results agree with the exact solution is to simply compare the individual values. This is a job that we will leave for the exercises.¹¹ It is also interesting to examine the eigenvectors; i.e., the standing wave patterns that we have mentioned above. Matlab also produces illustrative plots of the eigenvalues, and we show a few examples in Figure 11.16. These are the product of sine-like standing waves along x and y , as noted above.

The two problems that we have considered so far in this section, the mass-spring system in Figure 11.12 and the membrane in Figure 11.15, are both exactly soluble. However, the numerical approach that we have developed can be readily applied to more complicated situations. As an example, we consider again the vibrations of a membrane, but now we are interested in this as a model of a banjo. A banjo is roughly similar to a guitar (Figure 11.1), but the top plate of the guitar is replaced by a thin mylar sheet. A bridge rests against this sheet, and is held in place by the force of the strings which pass over the bridge and are then fastened to the rim of the banjo. This is all sketched, very qualitatively, in Figure 11.17.

¹¹They do, in fact, agree. Note that this agreement is expected only in the continuum limit; i.e., at low frequencies.

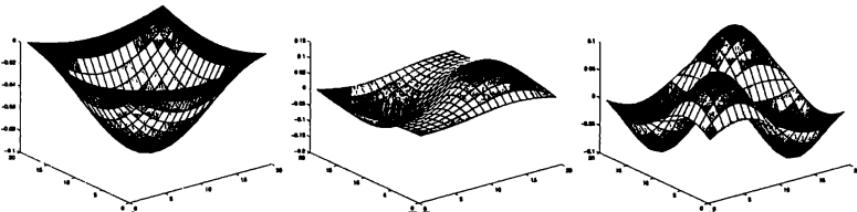


FIGURE 11.16: Eigenvectors for modes $(m, n) = (1, 1)$, $(1, 2)$, and $(2, 2)$ (left to right) of a thin membrane

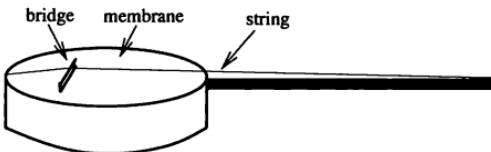


FIGURE 11.17: Schematic of a banjo. Here show only one string, but real banjos typically have 5 strings. A real banjo has a circular membrane, but in our model we assume that the membrane is square

We now want to compute the normal modes of the banjo membrane, with the bridge in place. The presence of the bridge forces us to resort to a numerical approach. We take the approach that we followed for a simple membrane, and use (11.21) to obtain an equation of motion for the banjo membrane. For portions of the membrane that are not under the bridge, we can again use (11.23), while for the elements of the membrane that are under the bridge we have to take into account the added mass of the bridge. For these portions of the membrane, this leads to¹²

$$(\sigma \Delta x \Delta y + \Delta m_{\text{bridge}}) \frac{\partial^2 z_k}{\partial t^2} = T(z_{k+1} + z_{k-1} + z_{k+N} + z_{k-N} - 4z_k). \quad (11.27)$$

where Δm_{bridge} is additional mass due to the piece of the bridge of area $\Delta x \times \Delta y$. These equations of motion then lead to an eigenvalue problem that has the form given in (11.24), with the dynamical matrix

$$\mathcal{D} = \begin{pmatrix} 4 & -1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & \dots & -1 & -1 & 0 & 0 & \dots & 0 \\ -\alpha & 4\alpha & -\alpha & 0 & \dots & -\alpha & -\alpha & 0 & \dots & 0 \\ 0 & -1 & 4 & -1 & \dots & 0 & -1 & -1 & \dots & 0 \\ \vdots & & & & & & & & & \\ 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 \end{pmatrix}. \quad (11.28)$$

¹²We should confess that our model of a banjo membrane with the bridge includes only the added mass of the bridge. We have ignored the extra stiffness associated with the bridge.

This is identical to the dynamical matrix for a simple membrane (11.25), except for the rows involving the membrane elements that support the bridge. These are the rows in (11.28) that contain factors of α (only one such row is shown here). These factors account for the extra mass of the bridge as included in (11.27). Referring to that equation, we see that $\alpha = 1/(1 + \Delta m_{\text{bridge}}/(\sigma \Delta x \Delta y))$.

The dynamical matrix can no longer be cast into a universal form; it now depends on the ratio of the mass of the bridge to the mass density of the membrane. To obtain quantitative results we must therefore choose a value for this ratio; i.e., a value for α . Figure 11.18 shows some results for the eigenfrequencies for a membrane with and without a bridge. For the calculation with a bridge we have chosen a bridge mass that corresponds roughly to a real banjo, with values given in the caption.¹³ In this figure we show the distribution of lowest eigenfrequencies. Each line shows the frequency of a different mode, plotted at the frequency of that mode. The top half of the figure shows results for a membrane without a bridge, so many of the modes are doubly degenerate, and thus plotted with a vertical line of length 2. The bottom half of the figure shows the modes with a bridge. The bridge is located away from the center of the membrane, and this breaks the symmetry so that all modes are now nondegenerate. The important message from this comparison is that the bridge affects the mode frequencies a *lot*, so one must certainly include the effect of the bridge in any quantitative modeling of the banjo.¹⁴ The general frequency shifts are downward, as expected since the bridge adds mass to the system.

It is also interesting to examine the eigenvalues when the bridge is present. One example is shown in Figure 11.19. This eigenvector shows the vibrational pattern for one of the lowest frequency modes (the mode at $\omega \approx 0.17$ in Figure 11.18), and we see that most of the amplitude involves tilting motion of the bridge. This confirms our earlier claim that the bridge has a very large effect on the vibrational behavior.

These results for the vibrational behavior of a banjo membrane give a flavor for how one can construct a fairly realistic computational model of a musical instrument. The same approach can be used to investigate the vibrational modes of a wide variety of vibrating systems.

EXERCISES

- 11.11. Consider the modes of a diatomic chain of masses. That is, assume that the masses in Figure 11.12 alternate in size, with values of m_1 and m_2 . Calculate the eigenvalues and eigenfunctions for this vibrational problem.
- 11.12. Perform a calculation of the modes of drumhead, as described by Figure 11.14.
- *11.13. Perform a calculation of the modes of drumhead, as described by Figure 11.14, but consider a circular membrane and work in polar coordinates.

¹³However, we should remember that this calculation is for a square banjo, so it is not quite a realistic instrument!

¹⁴Interestingly, we know of no other attempts to model the vibrations of a banjo membrane in such a semirealistic manner.

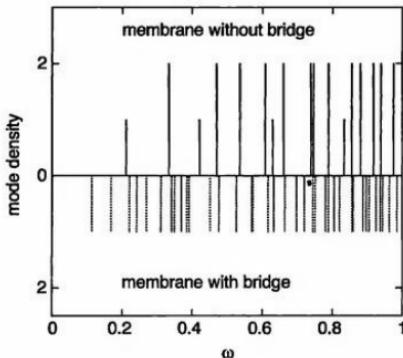


FIGURE 11.18: Distribution of vibration frequencies for a simple membrane (top), and a membrane with a bridge (bottom). For this calculation we took $\alpha = 0.025$. This corresponds to a bridge of total mass 10 g, on a membrane with $\sigma = 0.25 \text{ kg/m}^2$.

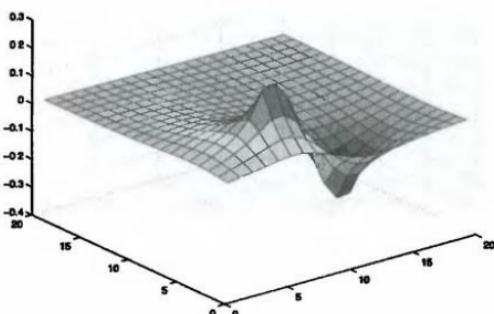


FIGURE 11.19: Vibrational mode (eigenvector) for a banjo membrane with a bridge. This is the mode with frequency $\omega \approx 0.17$ in Figure 11.18. The largest vibrational amplitude occur at the ends of the bridge. For this mode the ends of the bridge vibrate out of phase.

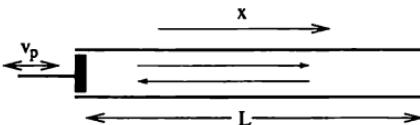


FIGURE 11.20: Narrow pipe that has a vibrating piston with velocity v_p at one end. The piston will excite sound waves that propagate back and forth along the pipe.

11.5 GENERATION OF SOUND

A complete theory of a musical instrument must deal with how the instrument generates sound. So far we have finessed this problem by assuming that the sound pressure is proportional to the force on the bridge. While this is a reasonable approximation, we now want to consider how to construct a real model of sound production. The calculation of the sound produced by a vibrating object is a challenging problem for several reasons. First, most musical instruments have complex shapes; i.e., violins and guitars have complicated shapes that are not described by simple functions.¹⁵ Second, in order to properly model sound production and propagation, we must model the room that contains the instrument, and real rooms are three dimensional. If we employ our usual finite difference approach, the associated spatial grid that we use to describe the room pressure will contain $N \times N \times N$ elements, where (as we will soon see) N must have values of 100 or more.¹⁶ This means we have to compute the motion of a very large number of discrete elements. Third, it is crucial that we take into account the way that sound reflects from a wall or boundary, and we will see that modeling this process is not simple.

In order to avoid at least a few of these complications, we will consider only the very simplified geometry sketched in Figure 11.20. Here we show a simple pipe that is closed at one end (the right), and is fitted with a movable piston at the other (the left). The piston moves with a velocity v_p , and this motion generates sound waves that travel within the air in the pipe. We assume a very narrow pipe, so that this motion is one-dimensional. The sound waves are then described by a velocity v , and an acoustic pressure p , both of which are functions of x and t . Since the pipe is very narrow, the velocity of the air must be along the pipe, so v is the component of the air velocity in the horizontal (x) direction. The acoustic pressure is the pressure associated with the sound wave; that is, p is the difference between the absolute pressure and the normal (constant) atmospheric pressure.

If the sound wave is not too large, so that we are in the regime of simple linear acoustics, the velocity and pressure obey the following equations (see Morse and

¹⁵The inventors of the violin and the guitar did not have the physicist/modeler in mind!

¹⁶There are several other important numerical approaches besides the finite difference method that can be used, but all are computationally quite demanding.

Ingard, 1986)

$$\begin{aligned}\rho \frac{\partial v}{\partial t} &= - \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial t} &= - \rho c^2 \frac{\partial v}{\partial x}\end{aligned}\quad (11.29)$$

where ρ is the density of the air and c is the speed of sound. If our system were three dimensional, we would have additional equations for the other velocity components and additional terms in the expression for $\partial p / \partial t$. These coupled equations are very similar to those encountered in electromagnetism, where the variables are then the electric and magnetic fields. Hence, the numerical method we use here can be applied to electromagnetic problems too.

At this point a natural inclination is to combine the two first order equations for v and p to obtain one second order wave equation that has the form of (11.1), and that involves only v or only p . One can certainly do that, but it is generally not a useful approach for the following reason. We will need to deal carefully with the boundary conditions at the ends of the pipe. These boundary conditions can usually be expressed as a relation involving both v and p , and this will make it simpler to work with the two first order equations in (11.29).

Before we discuss these boundary conditions, we must first introduce discrete grids for x and t . For the purposes of numerical stability, it is actually preferable to introduce two sets of *staggered* grids.¹⁷ Let us first consider the spatial grids sketched in Figure 11.21. We imagine, as usual, that space is divided into units of size Δx , but now we take one set of grid points to be at $x = i\Delta x$, and another separate grid at $x = (i + \frac{1}{2})\Delta x$, where i is an integer. We use the integer grid points for the pressure, and half-integer grid points for the velocity. That is, the pressure will be calculated at locations $x = i\Delta x$, while the velocity will be computed at $x = (i + \frac{1}{2})\Delta x$. The motivation for doing this is that when we write (11.29) in a finite difference form, the space derivatives will now involve symmetric differences. For example, the partial derivatives of v will involve values of the velocity at locations $(i - \frac{1}{2})\Delta x$ and $(i + \frac{1}{2})\Delta x$, and these will be used to compute the pressure at a location that is midway between, at $i\Delta x$. This reduces the associated numerical error by one order in Δx . The use of such staggered grids is useful in many contexts, including electromagnetic problems, where two grids are used for E and H . In our work with the leap-frog method in quantum mechanics separate grids were used for the real and imaginary parts of the wave function.

If a staggered grid is good for treating x , why not use it also for t ? Indeed we will, as this approach also leads to good accuracy for the time derivatives in (11.29). We therefore calculate p at time steps $n\Delta t$, and v at times $(n + \frac{1}{2})\Delta t$.

To translate (11.29) into finite difference form, we write $p(i, n) \equiv p(i\Delta x, n\Delta t)$, etc. for $v(i + \frac{1}{2}, n + \frac{1}{2})$. The first equation in (11.29) then becomes

$$\frac{v(i + \frac{1}{2}, n + \frac{1}{2}) - v(i + \frac{1}{2}, n - \frac{1}{2})}{\Delta t} = - \frac{p(i + 1, n) - p(i, n)}{\rho \Delta x}. \quad (11.30)$$

¹⁷Recall our use of a staggered grid (in time) in our work on quantum mechanics in Chapter 10

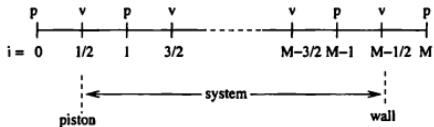


FIGURE 11.21: Staggered grid points p is defined on the integer grid, while v is defined on the half integer grid.

We can then rearrange this to express the velocity at the next time step, $n + \frac{1}{2}$, in terms of the velocity and pressure at previous times. The pressure can be treated in a similar manner, and the results are

$$\begin{aligned} v\left(i + \frac{1}{2}, n + \frac{1}{2}\right) &= v\left(i + \frac{1}{2}, n - \frac{1}{2}\right) - \frac{\Delta t}{\rho \Delta x} [p(i+1, n) - p(i, n)] , \\ p(i, n+1) &= p(i, n-1) - \frac{\rho c^2 \Delta t}{\Delta x} \left[v\left(i + \frac{1}{2}, n + \frac{1}{2}\right) - v\left(i - \frac{1}{2}, n + \frac{1}{2}\right) \right] . \end{aligned} \quad (11.31)$$

To use these relations we first update the velocity, calculating v at time step $n + \frac{1}{2}$. We next update the pressure, obtaining p at step $n + 1$. This process is then repeated for as many time steps as desired.

The only remaining question is how to treat the boundaries at the two ends. At the left end, where we have a vibrating piston, we assume that the air velocity at the surface of the piston is equal to the piston velocity, v_p . We therefore place this boundary of the system at the spatial grid point $i = \frac{1}{2}$, as indicated in Figure 11.21, and the grid point at $i = 0$ is not used (it is “outside” the system). At the right boundary we will assume that the sound waves are reflected with a slight attenuation. This would be typical of a real wall. At a microscopic level, the attenuation process is complicated. One would generally expect that the surface of the wall will deform a small amount in response to the pressure oscillation, so the wall surface will have a small velocity. This deformation of the wall will involve energy loss, causing the reflected wave to be slightly smaller in amplitude than the incoming one. This loss process can be modeled at a macroscopic level in terms of an acoustic impedance,

$$Z \equiv \frac{p_{\text{wall}}}{v_{\text{wall}}} , \quad (11.32)$$

where p_{wall} and v_{wall} are measured at the wall. The precise value of Z will depend on the wall material. An infinite value of Z corresponds to $v_{\text{wall}} = 0$, and hence no loss and a perfect reflection. For real materials, it is necessary to treat Z as a frequency dependent quantity, with p_{wall} and v_{wall} the pressure and velocity amplitudes that would be found for a purely harmonic excitation. In this frequency domain picture, Z can be complex, which means that there is a phase shift on reflection. For simplicity, we will ignore these complications, and assume that Z is real. This is a good approximation for most materials.¹⁸

¹⁸A more general treatment can be found in the papers by Botteldooren that are listed at the end of this chapter

In order to apply (11.32), we need to know both v and p at the wall. However, since the velocity is calculated at half-integer grid points while the pressure is known at integer points, we must do a little interpolation first. We begin by placing our wall at $i = M - \frac{1}{2}$ (see Figure 11.21), and we consider how to apply the relation for updating v from (11.31) at this point. At $i = M - \frac{1}{2}$ (11.31) becomes

$$\begin{aligned} v\left(M - \frac{1}{2}, n + \frac{1}{2}\right) &= \\ v\left(M - \frac{1}{2}, n - \frac{1}{2}\right) - \frac{\Delta t}{\rho \Delta x} [p(M, n) - p(M - 1, n)] , \end{aligned} \quad (11.33)$$

and this contains p at the location $i = M$ which is outside the system (and is hence not defined). However, the pressure terms on the right hand side of (11.33) came from a centered space derivative, so we can rewrite this derivative using an uncentered difference to get

$$\begin{aligned} v\left(M - \frac{1}{2}, n + \frac{1}{2}\right) &= \\ v\left(M - \frac{1}{2}, n - \frac{1}{2}\right) - \frac{2\Delta t}{\rho \Delta x} \left[p(M - \frac{1}{2}, n) - p(M - 1, n) \right] . \end{aligned} \quad (11.34)$$

The pressure at $i = M - \frac{1}{2}$ can be gotten from the acoustic impedance (11.32) which gives

$$p(M - \frac{1}{2}, n) = Zv(M - \frac{1}{2}, n) . \quad (11.35)$$

This result is useful, since it involves the velocity at a half-integer location, which we know. However, it involves an integer time grid point, while the velocity is calculated on half-integer time grid points. So, one more interpolation is needed

$$\begin{aligned} p\left(M - \frac{1}{2}, n\right) &= Zv\left(M - \frac{1}{2}, n\right) \\ &= \frac{Z}{2} \left[v\left(M - \frac{1}{2}, n + \frac{1}{2}\right) + v\left(M - \frac{1}{2}, n - \frac{1}{2}\right) \right] . \end{aligned} \quad (11.36)$$

Inserting this into (11.34) and doing a little rearranging leads to our final result

$$\begin{aligned} v\left(M - \frac{1}{2}, n + \frac{1}{2}\right) &= \\ = \frac{1}{1 + \frac{Z\Delta t}{\rho \Delta x}} \left[\left(1 - \frac{Z\Delta t}{\rho \Delta x}\right) v\left(M - \frac{1}{2}, n - \frac{1}{2}\right) + \frac{2\Delta t}{\rho \Delta x} p(M - 1, n) \right] . \end{aligned} \quad (11.37)$$

Our approach to calculating the sound in the one-dimensional system in Figure 11.20 is then as follows. We use (11.31) to calculate the sound pressure and the air velocity as functions of time, on a grid that is staggered in both space and

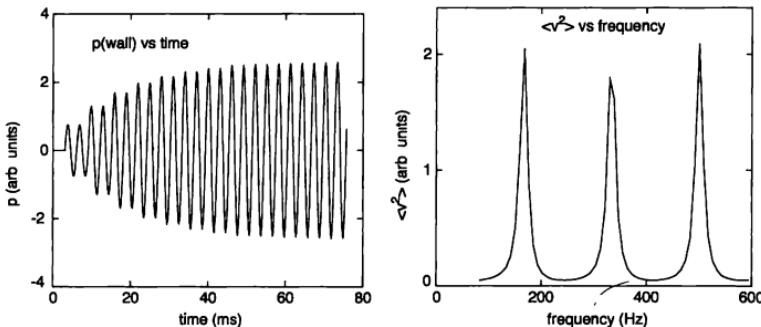


FIGURE 11.22: Left sound pressure at the wall as a function of time when the piston is driven sinusoidally at a frequency of $f = 330$ Hz. The length of the system was $L = 1.0$ m, with $c = 330$ m/s, $\rho = 1.3$ kg/m³, and $Z = 6000$ Pa · s/m. We used grid sizes $\Delta x = 0.01$ m and $\Delta t = \Delta x/c$, so that there were $M = 100$ spatial grid elements.

time. These relations are used to find p and v in the interior of the system. At the walls we use the piston velocity for the wall at $i = \frac{1}{2}$, and (11.37) for the wall at $i = M - \frac{1}{2}$. This is summarized by the following pseudocode.

EXAMPLE 11.3 Pseudocode for subroutine room_sound

- Initialize p and v to zero everywhere.
 - For each time step:
 - ▷ Update v in the interior of the system according to (11.31).
 - ▷ Update v at $i = \frac{1}{2}$ according to the piston velocity.
 - ▷ Update v at $i = M - \frac{1}{2}$ according to (11.37).
 - ▷ Update p using (11.31).
-

Some results from our sound calculation are shown in Figure 11.22. Here we have assumed that the piston moves according to $v_p = v_0 \sin(2\pi ft)$, as would be the case for a loudspeaker playing a pure tone of frequency f . On the left we show how the pressure at the right-hand wall varies with time. We see that the pressure amplitude initially increases, and then approaches a constant. This steady state is reached when the energy input from the piston is balanced by the energy lost at the right-hand wall through the acoustic impedance Z . The time required to reach steady state depends on the value of Z , as we will explore in the exercises.

On the right in Figure 11.22 we show how the steady state sound intensity varies with frequency. Here we have repeated the calculation on the left for many

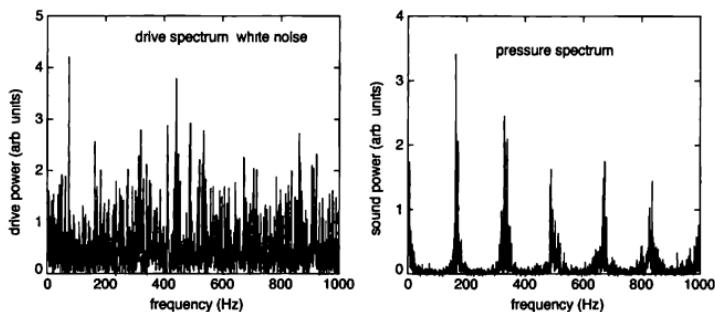


FIGURE 11.23: Sound calculation with a white noise drive signal applied to the piston. Left: spectrum of v_p . Right: spectrum of the right-hand wall pressure. The parameters were the same as those used in Figure 11.22.

different values of the frequency f . For each value of f we have computed $\langle v^2 \rangle$, the mean square of the velocity averaged over the entire system. The peaks in $\langle v^2 \rangle$ are simply resonances. The resonance at $f \approx 170$ Hz occurs when the wavelength $\lambda = c/f$ is equal to $2L$, while the resonance at 330 Hz corresponds to $\lambda = L$, etc. These are just the usual standing wave resonances in a pipe. The widths of these resonances are proportional to the quality factor, Q . The Q depends on the value of the acoustic impedance; larger values of Z lead to higher Q , and vice-versa.

A similar type of resonant behavior is found if we drive the piston with a “noise” signal. We can generate such a signal by setting v_p at each time step to a random number that is uniformly distributed in the interval $-v_0 < v_p < v_0$. The power spectrum of a typical result for v_p generated in this way is shown on the left in Figure 11.23.¹⁹ On the right part of the figure we show the spectrum of the steady state pressure amplitude at the wall, and we see that this spectrum has pronounced peaks at the resonant frequencies of the pipe. These are the same resonant frequencies that we observed with a sinusoidal drive signal in Figure 11.22. In this case, the room has acted as a sort of filter, removing all of the noise components except those at the resonant frequencies.

This approach for calculating the sound pressure and air velocity can be applied to deal with three dimensional geometries. However, the number of spatial grid elements is then proportional to L^3 , which can be a large number. For our pipe with $L = 1$ m, we had $M = 100$, so a realistic sized room will generally require

¹⁹This is an example of “white noise.” To within the statistical fluctuations, the spectrum of v_p is independent of frequency.

more than 10^6 discrete elements. You should also notice that in order for the algorithm to be numerically stable for one-dimensional geometries, the spatial and time steps should satisfy the condition $\Delta x \geq c/\Delta t$. As we explained in Chapter 6, this condition guarantees that our numerical solution is able to produce a wave velocity that is at least as large as c .

EXERCISES

- 11.14. Write a program to calculate the sound pressure in a pipe, as in Figure 11.22. Calculate the Q of the resonances as a function of Z .
- 11.15. Use your program from the previous exercise to study how the sound decays after the piston is turned off. Do this by first driving the piston with $v_p = v_0 \sin(2\pi ft)$ for a time period that is long enough that steady-state is reached. Then turn off the piston (i.e., set $v_p = 0$) and observe the decay of p at the right hand wall. Show that this decay is exponential, and calculate the time constant of the decay as a function of Z .
- *11.16. Extend our room model to treat a three dimensional room. As a sound source you could use a vibrating membrane, or a vibrating piston inserted into one of the walls of the room.

REFERENCES

- [1] D. Botteldooren, "Acoustical finite-difference time-domain simulation in a quasi-Cartesian grid," *J. Acoust. Soc. Am.* **95**, 2313 (1994).
- [2] A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. I. Physical model for a struck string using finite difference methods," *J. Acoust. Soc. Am.* **95**, 1112 (1994); "Numerical simulations of piano strings. II. Comparisons with measurements and systematic exploration of some hammer-string parameters," *J. Acoust. Soc. Am.* **95**, 1631 (1994).
- [3] L. Cremer, 1984, *The Physics of the Violin*, Translated by J. S. Allen, MIT Press, Cambridge.
- [4] N. H. Fletcher and T. D. Rossing, 1998, *Physics of Musical Instruments, 2nd edition*, Springer-Verlag, New York.
- [5] H. L. F. Helmholtz, 1954, *On the Sensations of Tone, 4th ed.*, Dover, New York.
- [6] P. M. Morse and K. U. Ingard, 1986, *Theoretical Acoustics*, Princeton University Press, Princeton.

Interdisciplinary Topics

This is a book about physics. We have considered a wide range of problems in the past 11 chapters, and most readers would probably agree that they are all physics problems. However, the boundary between different disciplines is not always clear-cut. For example, you might claim that the Lorenz model belongs to atmospheric science, or that percolation, since it can be used to discuss the properties of porous materials, is really a geoscience problem. Arguments over such matters are generally not very useful, but they do illustrate that the lines between disciplines are not sharply defined. Indeed, the ideas developed in one area can often lead to important breakthroughs when applied to other fields. In this chapter we will consider five cases in which it appears that such cross-fertilization has (or may) prove to be extremely fruitful. The topics we now discuss are active areas of research, so even the tentative conclusions we will draw may change considerably in the next few years. Nevertheless, the computational methods we employ, which are extensions of approaches we have encountered earlier in this book, have already led to useful new insights.

The first problem in this chapter is from biology and concerns protein structure. Proteins are long flexible molecules that can fold into many different shapes. The problem is to understand the folding process so that we can predict the folded structure of a newly encountered protein and thereby learn how to design molecules with a desired shape. The second example is from geoscience and concerns earthquakes and a possible connection with phase transitions. The third problem deals with the brain and some proposals concerning the way (human) memory might work. The fourth problem discusses first steps toward understanding how the basic elements of the nervous system, i.e., neurons, work. The final section introduces a new technique called cellular automata and illustrates it with applications to an example of self-organizing behavior called the sandpile problem. We will see that all of these problems are closely related in a computational sense to topics we have touched on in this book. These problems will also give us a chance to illustrate further both the process and philosophy of model building in theoretical physics.

12.1 PROTEIN FOLDING

A polymer is constructed by linking together a collection of short molecular segments. Many biological molecules are constructed in this way, including DNA (and its relatives) and proteins. You have probably heard much about the wonders of DNA and the double helix, etc. Here we will focus, instead, on proteins. These are key participants in a very wide variety of biological processes. They are involved in energy storage and conversion, handle communication between cells, are important



FIGURE 12.1: Schematic protein in a folded (left) and unfolded state (right)

structural components in several parts of the body including tendons and bones, and are catalysts in many biochemical reactions.¹

Proteins are built from amino acids, which are relatively short molecules containing 10s of atoms. Nature uses only 20 different amino acids to construct *all* of the known proteins. These 20 monomers (another term we will use for amino acids in this context) are very similar to each other and thus form a fairly homogeneous set of building blocks. A typical protein is made up of a few hundred amino acids linked together end-to-end, although the overall length of a protein can vary greatly. The shortest proteins contain only on the order of 50 monomers, while the longest are comprised of several thousand. A *particular* protein is composed of a *particular* sequence of amino acids. This sequence is known as the primary structure of the protein. It is believed that all of the properties and functions of a protein are uniquely determined by its primary structure.

Proteins can be viewed as chains in which the links are the amino acids. It turns out that the connections between these links are somewhat flexible. In a living cell proteins exist in a solution that is mostly water, and this makes it possible for the chain to assume different shapes. Possible shapes include the two shown schematically in Figure 12.1. In one case the protein is folded into a compact “glob,” while in the other it is unfolded with an end-to-end length that is close to the maximum possible. The shapes of real proteins are generally more complicated than these two very simple illustrations. For example, some proteins form helical sublengths that then fold into sheets or globs. Our point here is only that a long chain molecule with flexible connections between monomers has the possibility of taking on *many* different shapes. Which shape is preferred will depend on temperature and the chemicals present in solution. The structure of a protein when it is in its biologically active state is known as the tertiary structure, and this is generally some sort of folded state. The biological functions of a protein are a result of its tertiary structure, since this determines what other molecules a protein can bind to, the kinds of spaces it can fit into, etc. Hence, if we want to understand how a protein works, we must understand its tertiary structure. In addition, if we want to *design* a new protein with a specific (presumably beneficial) property, we must be able to *predict* the tertiary structure from knowledge of the primary structure.

¹We hope the reader will understand that this section has been written by physicists. Our goal is to emphasize the aspects of proteins that are important for understanding the physics of the protein-folding problem.

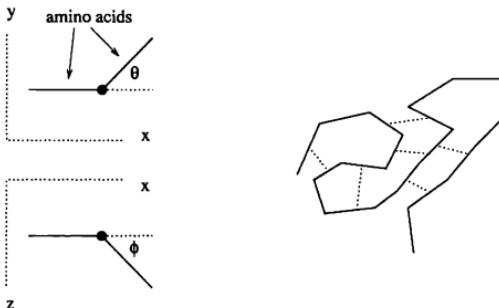


FIGURE 12.2: Left angle between two amino acids when viewed in a projection in the x - y plane (top) and x - z plane (bottom). For simplicity each amino acid is drawn simply as a solid straight line (the side chains are not shown), and the covalent bond between them is represented by the filled circular dot. Right schematic protein in a folded state. The dotted lines indicate interactions between parts of the chain that are not connected by strong covalent bonds.

While the chemical bonds between adjacent amino acids in a protein chain are not completely rigid, they are also not completely flexible. The relative orientation of two adjacent amino acids in a chain are determined by the covalent chemical bond between the two and can typically assume several different values. This is illustrated in Figure 12.2, which shows a hypothetical pair of monomers that are bonded so as to have a relative angle of θ in the x - y plane and ϕ in the x - z plane. In most cases the chemical bond permits either or both of these angles to be reversed (that is, $\theta \rightarrow -\theta$) without affecting the bonding energy, yielding four possible choices for the relative orientation of the two amino acids. When we combine this with four more choices when the next monomer is added to the chain, etc., for a long chain, we can see that there are a *large* number of different tertiary structures that have the same energy as far as the direct, covalent bonds between the amino acids are concerned. While this picture is a bit simplified² it does illustrate (correctly!) the origin of a protein's flexibility and why it can assume many different particular shapes. If we assume four different relative orientations for each link in a chain of N amino acids, there will be $\sim 4^N$ different possible structures (ignoring self-avoidance constraints). For $N = 300$ this yields $\sim 10^{180}$ distinct tertiary structures, all arising from a single primary structure.

Even though a particular protein can take on a large number of possible structures, the molecule "knows" which particular tertiary structure is the proper one. Otherwise it could not carry out its intended biological functions. Under certain conditions a protein can be induced to unfold, that is, be made to assume

²The energetically allowed orientations between all possible pairs of the 20 amino acids have been determined from experiments, and the interested reader can learn about this by reviewing the references. This information could be added to the model we develop here, but would make things more complicated than necessary for our purposes.

the stretched-out shape shown on the right side of Figure 12.1. For example, this can often be accomplished by adding the appropriate chemicals to its solution (e.g., changing the pH). Such an unfolded structure might also be found when a protein is synthesized initially. Experiments have shown that if an unfolded protein is put back into its biologically "natural" environment, it will fold back into its original structure. This is remarkable for two reasons. First, the protein manages to find *precisely* the proper final structure, even though there are an astronomical number of possible alternatives.³ Second, the *time* it takes for the protein to refold is typically the order of seconds. You might have thought that the protein would sample a significant fraction of its $\sim 4^N$ possible states on its way to a final folded structure. It was first pointed out by Levinthal that even if a protein spends only on the order of 10^{-13} s in each of these intermediate states,⁴ the folding process would still take longer than the age of the universe! This conundrum is known as Levinthal's paradox.

Since proteins are in fact able to fold rapidly into their proper tertiary structure, it would appear that a protein is somehow able to locate the correct structure without searching through all of the possibilities. Precisely how it accomplishes this feat and how it "knows" what the proper tertiary structure should be, is not understood. This is the protein-folding problem.

In order to construct a sensible model of the folding process, we need to consider the different forces and energies in the problem. The largest energy scale is that of the direct covalent bond between adjacent amino acids. This is by far the strongest bonding in the protein, but since different tertiary structures have the same collection of covalent bonds, this energy does not play any role in *differentiating* between different folded (or unfolded) structures. This differentiation is the result of several other, much weaker forces. One of these is the Van der Waals force between amino acids that are not covalently bonded. We have encountered this force in our work with molecular dynamics, where we saw that it is attractive at moderate to long distances and falls off rapidly with separation. While it will tend to bring the monomers together, it will only be important when they are not too far apart. There will also be hydrogen bonds between nearby amino acids; these will lead to attractive forces and thus appear to prefer a folded state for the protein. The Van der Waals and hydrogen bond forces are shown schematically by the dotted lines in Figure 12.2. In addition, we must consider the effect of the water molecules and other chemicals in solution. It turns out that water is attracted to some amino acids and repelled from others. Monomers that are strongly attracted to water molecules will prefer an unfolded structure, since this would allow them to be close to more water molecules. Hence, there will be a competition between the various forces, as some prefer a folded state while others favor an unfolded structure.

The relative importance of these forces will be a complicated function of the particular amino acids in the protein chain and how they are arranged. Obtaining

³That the protein has actually assumed its original structure is demonstrated by the fact that its biological properties are the same as before it was unfolded

⁴This corresponds roughly to one period of a molecular vibration and is the time scale on which an atom or molecule is able to move a distance of one atomic spacing.

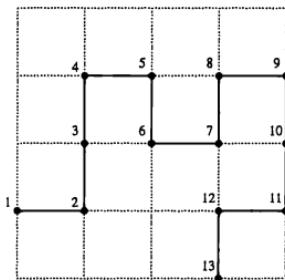


FIGURE 12.3: Lattice model for a protein. The filled circles represent amino acids and the solid lines are covalent bonds between them.

a quantitative understanding of the forces is itself an extremely formidable problem and is certainly more than we want to tackle here. While a great deal is known about these matters, this remains an area of much research. The key point for us is that folding is a *competition* between forces. Moreover, these forces are all relatively weak; they turn out to have a magnitude of order $k_B T$ (where T is room temperature) per protein. Hence, we have a very delicate interplay between these forces and the disordering effect of temperature.

This has been an extremely brief introduction to proteins, and we have obviously omitted a lot of interesting points. Nevertheless, we have described all of the key features that are needed to construct a model for the folding process. This will be an *extremely* simple model, and in some ways it will have only a faint resemblance to a real protein. Our goal is to understand how a protein folds and what factors affect the folding process. We begin with a simple model that we believe contains the essential physics. Only after understanding the behavior of this model could we have any hope in attacking a more realistic situation. We don't offer these words as an apology; rather, this is the philosophy that underlies model building in theoretical physics.

We will consider a model protein that sits on a two-dimensional lattice as shown in Figure 12.3. We assume a particular primary structure, that is, a sequence of N amino acids, which we denote as $A(1), A(2), \dots, A(N)$. These are just numbers in the range 1–20 (with no rule against repetitions) corresponding to the 20 different amino acids. Each link in this chain of amino acids is assumed to be located on a site of the lattice, with adjacent, covalently bonded links situated on nearest-neighbor sites.⁵ In order to mimic the attractive forces between amino acids that are not covalently bonded, we assume that there is an energy $J_{i,j}$ associated with

⁵In comparing Figures 12.2 and 12.3 you may notice a slight change in our graphical notation. In Figure 12.2 the filled circles denote the connections between amino acids, while in our lattice model, Figure 12.3, the lattice sites denote the amino acids themselves, with the connections effectively lying on the bonds between these sites. The two descriptions are equivalent, as both describe a chain with flexible links.

two amino acids that are nearest neighbors, but that are not covalently bonded, that is, *not* adjacent in the chain. $J_{i,j}$ thus represents the combined energies of the Van der Waals and hydrogen bonds, as well as the effects of bonding to water molecules. This energy will depend on the specific amino acids that are involved, so i and j are the indices of the two monomers, $i = A(n)$ and $j = A(m)$, where the n th and m th amino acids are situated on nearest-neighbor lattice sites. For example, in the model protein in Figure 12.3, amino acids 3 and 6 are nearest neighbors and are not adjacent members of the chain (that is, not directly connected by a covalent bond), so in our model they have an energy of attraction $J_{3,6}$. If two amino acids are not nearest neighbors, such as monomers 2 and 6, the energy of attraction is taken to be zero.

The energy of our model protein is then

$$E = \sum_{\langle m,n \rangle} \delta_{m,n} J_{A(m),A(n)}, \quad (12.1)$$

where the sum is over all pairs of proteins $\langle m, n \rangle$ in the chain. $\delta_{m,n} = 1$ if amino acids m and n are nearest neighbors that are *not* connected by direct covalent bonds and is zero otherwise. Which pairs of amino acids are noncovalently bonded nearest neighbors will depend on the tertiary structure of the protein, so this energy will be a function of the structure. The protein is assumed to be in thermal equilibrium with a heat bath, which is just the solution it is dissolved in, so we can use the rules of statistical mechanics to calculate its properties. The problem is thus analogous to the Ising model we studied in Chapter 8, and we will again use the Monte Carlo method in our simulations. We next describe the simulation procedure and make a few comments on the programming.

The first step in the simulation is to choose a primary structure. If there are N links in the protein chain, we need to specify the values of $A(1), A(2), \dots, A(N)$. In the simulations described below we first generated a hypothetical protein by choosing a sequence of N integers at random from the range 1–20, corresponding to the 20 different possible amino acids. This was then the primary structure of our model protein, and it was held fixed during the course of a particular simulation. The interaction energies $J_{i,j}$ must next be specified. Since there are 20 different possible amino acids [values of $A(m)$], $J_{i,j}$ can be thought of as a 20×20 matrix. In principle, the elements of this matrix could be determined from a quantum mechanical calculation of the Van der Waals forces, hydrogen bonding, etc., but this very formidable task is currently too difficult to tackle in a quantitative way.⁶ We will therefore take the convenient approach of assuming that the $J_{i,j}$ vary randomly within some specified range.⁷ We hope that the important features of the simulation will not depend strongly on how the $J_{i,j}$ are chosen, but that is

⁶In a sense, we have already conceded this by constraining the amino acids to lie on a two-dimensional lattice

⁷An even simpler choice would be to take them all to have a constant value, but since proteins have a structure that appears to be fairly “disordered,” it seems reasonable to put some randomness into the model. As described in the references, the approach we use here is taken in most current simulations of protein structure. Another interesting possibility would be to set the $J_{i,j}$ randomly to ± 1 ; we will explore this in the exercises

something we will only know after we have done some work. The final ingredient required for our simulation is the initial tertiary structure. In the calculations shown below we took this to be a completely straight chain. We might instead let it be a self-avoiding walk; we will leave such a study to the interested reader.

After choosing the initial conditions and values, the Monte Carlo method can be employed as follows. A link in the chain is selected at random by choosing a random integer in the range 1 to N . Let the lattice coordinates of this amino acid be (x_0, y_0) . Since we are using a square lattice, this site has four nearest neighbors, and one of these neighbors is next chosen at random; we label it (x_n, y_n) . If this neighboring site is not occupied by another amino acid, we then check to see if the monomer at (x_0, y_0) could move to (x_n, y_n) without breaking a covalent bond. This is illustrated in Figure 12.4, which shows a portion of a hypothetical initial structure on the left. Suppose that the amino acid at site 5 is chosen as (x_0, y_0) . Site 9 is a nearest neighbor, and the amino acid at site 5 could be moved to this location without breaking (stretching or compressing) either of the bonds to adjacent sites, so this would be an "allowed" move. Site 4 is also a nearest neighbor to site 5, but moving the amino acid to this location would stretch the bonds connecting it to sites 6 and 8, so such a move is prevented by the large energy associated with these covalent bonds. After finding that a move of the amino acid from site 5 to site 9 is permitted by the covalent bond constraints, the Monte Carlo approach is then used to determine if such a move is actually made. The energy of the chain in both its original structure (on the left in Figure 12.4), and in the potential new structure (on the right) is calculated from (12.1). This yields the energy required to make the move, ΔE_{move} . If this energy is negative so that the move would lead to a state with lower energy, the monomer is moved and the structure changes. If ΔE_{move} is positive so that the move would cost energy, the monomer is moved only if the Boltzmann factor $\exp(-\Delta E_{\text{move}}/k_B T)$ is greater than a random number in the range 0–1. Hence, moves that increase the energy are made with a probability given by the Boltzmann factor. This algorithm is analogous to the Monte Carlo rules for simulating a spin system as described in Chapter 8.

The simulation consists of repeating this procedure a large number of times. Sometimes it results in a new structure, while other times the structure is left unchanged. After each attempted move the system spends one Monte Carlo time step in the resulting state, whether the move is made or not. This (conceptually) is an essential part of computing the thermal averages of the energy and other properties. After a large number of time steps the protein should reach thermal equilibrium with the heat bath. We can then determine its properties by averaging over the state of the protein during the course of many subsequent Monte Carlo steps. In the language used in Chapter 8, each Monte Carlo step leaves the protein in a particular microstate corresponding to a particular tertiary structure. Properties such as the energy or length of the chain are then obtained by performing an average over many such microstates.

Some snapshots of a protein during a typical simulation are shown in Figure 12.5. The protein was initially in a completely straight state, as shown in Figure 12.5(a). In this state only the ends of the chain can move without violat-

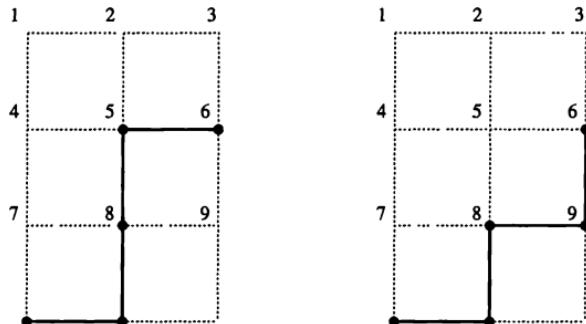


FIGURE 12.4: Left hypothetical initial structure of a portion of a protein chain, right new structure obtained by moving the amino acid initially at lattice site 5 to site 9. Based on the portion of the lattice shown here, these two tertiary structures would have the same energy. However, portions of the chain beyond the right edges of these pictures could cause them to have different energies.

ing the covalent bonding constraints, so most of the randomly generated potential moves are rejected. Here it took 25 Monte Carlo time steps before a move involving an end site was accepted, and this led to the structure in Figure 12.5(b). After 249 time steps the structure in Figure 12.5(c) was obtained, and after 996 Monte Carlo steps the protein had taken on the structure in Figure 12.5(d). In many cases, the moves are between two states with the same energies. For example, the structures shown in parts (a), (b), and (c) of Figure 12.5 all have the same energies, as none are folded in such a way as to have two noncovalently bonded monomers on nearest-neighbor sites. However, the structure in Figure 12.5(d) does have a different energy. The Monte Carlo rules for acceptance or rejection of a potential move have no difficulty in dealing with two states that are degenerate in energy.

The energy as calculated from (12.1) and the end-to-end length of this chain of 15 amino acids are shown in Figure 12.6 as functions of time. Here time is measured in units of Monte Carlo time steps and the temperature was $T = 10$ (for simplicity we measure energy in units of k_B , so that temperature and the $J_{i,j}$ are effectively unitless). The smallest end-to-end length was ~ 1.4 , corresponding to ends located on sites diagonally opposite one another; this is the closest the ends of a chain with an even number of links can get. The largest length was 11, corresponding to nearly the maximum separation. The protein structure thus fluctuated considerably, as it spent a good deal of time in a fairly unfolded state. The energy measured during the same simulation also fluctuated greatly.

As the temperature was lowered, the magnitude of the fluctuations became smaller (Figure 12.8). At $T = 1$ the length fluctuated between about 1.4 and 6, so the protein was now in a more compact state. The fluctuations in the energy were also smaller and occurred over much longer time scales. The protein now spent most, but certainly not all, of its time in states with the lowest energies.

Protein folding 15 amino acids

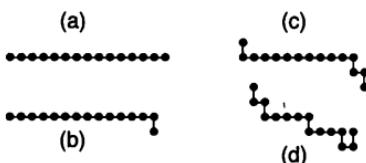


FIGURE 12.5: Snapshots of a model protein at different stages of a simulation. The chain contained 15 amino acids and $T = 10$. The pictures show (a) the initial structure, (b) the structure after 25 Monte Carlo time steps, (c) 249 time steps, and (d) 996 time steps. The interaction energies $J_{i,j}$ were distributed uniformly in the range -4 to -2 .

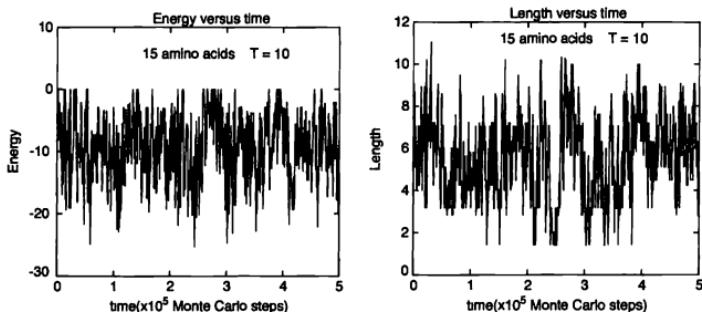


FIGURE 12.6: Energy (left) and end-to-end length (right) as functions of time for a protein chain with 15 amino acids. The temperature was $T = 10$ and the $J_{i,j}$ were chosen as in Figure 12.5.

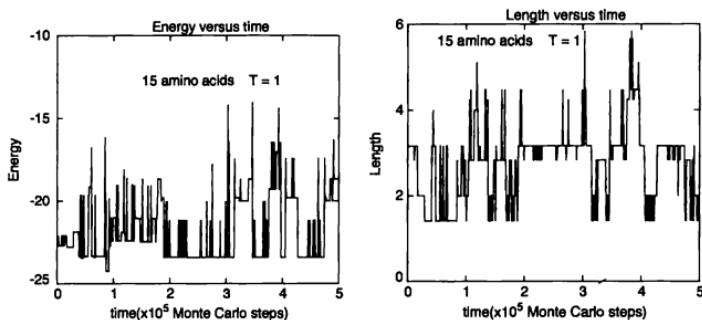


FIGURE 12.7: Energy (left) and end-to-end length (right) as functions of time for a protein chain with 15 amino acids. The temperature was $T = 1$. This is the same chain as considered in Figure 12.6. Note the different vertical scales as compared with Figure 12.6

The thermal averages of the energy and length as functions of temperature are shown in Figure 12.8, and we see that each of these quantities changed continuously from their values in the unfolded state at high temperatures to those of the folded state at low T . While both the energy and the length varied smoothly with temperature, most of the changes occurred between $T = 5$ and $T = 2$, with very little additional change as the temperature was lowered further. These variations should remind you of the behavior of the order parameter for the Ising ferromagnet considered in Chapter 8 in the vicinity of its second-order transition. Thus, in qualitative terms our model protein exhibits a transition in the neighborhood of $T \sim 2$, although this transition is certainly not abrupt. A true, sharp phase transition can only occur in an infinitely large system, and our protein is not infinite in extent. However, since real proteins can be *much* larger than our 15 monomer chain, it would not be surprising to find a much sharper transition in a real chain.

We can investigate this possibility by studying the behavior of longer chains, and some simulations for chains with 30 and 100 amino acids are shown in Figures 12.9 and 12.10. We again see that both E and the length decreased rapidly in the neighborhood of $T = 2$, and the results do seem to suggest that this pseudotransition to a compact structure is sharper in the longer chains. This would be in accord with the behavior of real proteins, as experiments show that they fold or unfold abruptly as conditions such as temperature or the nature of their solution⁸ is changed.

In all of the simulations described so far we have started at a high temperature where fluctuations in the structure were relatively large. The temperature was then reduced in steps, and results at different temperatures obtained. In order to learn more about the specific folded structure obtained at low temperatures and

⁸In our model this would correspond to changing the interaction energies $J_{i,j}$.

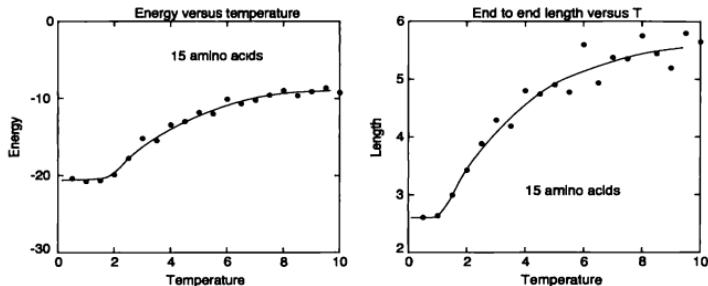


FIGURE 12.8: Average energy (left) and end-to-end length (right) as functions of temperature for a protein chain with 15 amino acids. This is the same model protein as considered in Figure 12.6. The temperature was swept down in stages from $T = 10$. The “scatter” in these results (and in the results shown in the next two figures) arises from the Monte Carlo statistical fluctuations. They could be reduced by averaging over a larger number of Monte Carlo time steps.

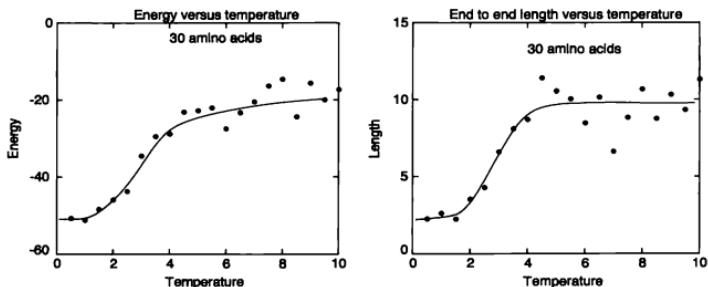


FIGURE 12.9: Average energy (left) and end-to-end length (right) as functions of temperature for a protein chain with 30 amino acids. The energies $J_{i,j}$ were randomly distributed in the range -4 to -2 . The values at each temperature were obtained by averaging over 5×10^5 Monte Carlo time steps. The temperature was swept down in stages from $T = 10$.

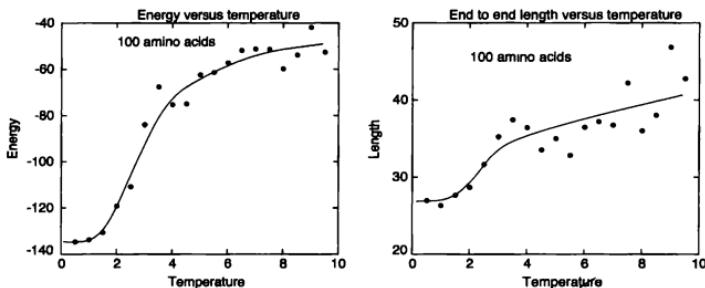


FIGURE 12.10: Average energy (left) and end-to-end length (right) as functions of temperature for a protein chain with 100 amino acids. The energies $J_{i,j}$ were randomly distributed in the range -4 to -2 . The values at each temperature were obtained by averaging over 5×10^5 Monte Carlo time steps. The temperature was swept down in stages from $T = 10$.

to address Levinthal's paradox, it is useful to consider simulations that begin with the protein in a completely unfolded state at a low temperature. Some results of this kind are given in Figure 12.11, which shows two independent simulations for the *same* model protein; that is, the same sequence of amino acids (primary structure) and the same interaction energies, $J_{i,j}$. The temperature used here is low in the sense that it is below the range where the energy and length changed in our previous simulations, Figures 12.8–12.10. In both cases the energy initially dropped with time as the protein folded into a fairly compact state, but the detailed variation of E was very different in the two cases. In one case, the solid curve in Figure 12.11, E dropped quite rapidly until $t \sim 1 \times 10^5$, continued to decrease slowly up to $t \sim 10 \times 10^5$, and was on average constant at longer times. In the other simulation E dropped more slowly at early times, but continued this slow decrease up to $t \sim 10 \times 10^5$. There are two important features of these results. One is that even at long times ($t > 10 \times 10^5$) the energy continued to fluctuate substantially. This implies that even though the chain had found a fairly stable structure, there were still fluctuations involving minor movements of a few amino acids. Second, the energies of the stable structures found in the two cases were significantly different. This protein found two *different* tertiary structures.

Our model protein thus does *not* behave like a real protein, as it does not fold reproducibly into the same tertiary structure. In a sense, our protein has fallen prey to Levinthal's paradox. It was not able to locate the “best” (or even the same) structure during repeated attempts at folding. In at least one of these two cases it was caught in a structure that was not the ideal one. Such nonoptimal structures are known as *metastable states*.

The problem of trapping in a metastable state is actually a very common one in nature. The difficulty can be appreciated from Figure 12.12, which shows a schematic energy landscape for a protein. Here the vertical axis is energy while

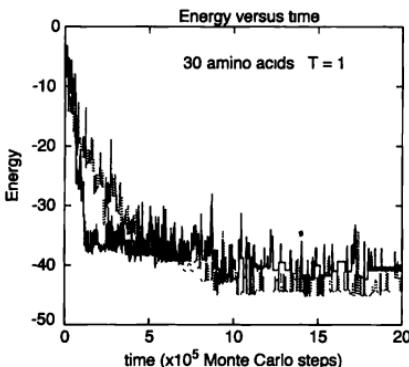


FIGURE 12.11: Energy versus time for a chain with 30 amino acids at $T = 1$. The two curves show results for separate simulations with the same chain. In each case the chain was completely unfolded at $t = 0$. The interaction strength $J_{i,j}$ was distributed uniformly in the range -4 to -2 .

the horizontal axis corresponds to the structure of the protein. Of course, this structure is much too complicated to be described with a single coordinate, so if we wanted to be completely realistic we would have to plot the energy as a function of many variables, such as the positions of all of the amino acids. A high-dimensional plot of this kind would be difficult to project onto two dimensions and probably wouldn't be very useful anyway. The key point is that the energy of a protein is a complicated function of its tertiary structure. E will be lower for some structures than for others, and there will in general be many structures that are metastable, such as the ones labeled 1 and 2 in Figure 12.12. This means that all nearby structures that can be found by small movements of only a few monomers have higher energies.

The optimal tertiary structure of a protein will presumably be the one that minimizes the energy.⁹ However, if it finds itself in a metastable state, the protein may, if the temperature is low, have a very difficult time escaping, since all nearby structures have higher energies. This is a second aspect of Levinthal's paradox. A protein must locate the lowest energy state without becoming trapped in a metastable state.

One way this might be accomplished is suggested by the Monte Carlo algorithm. Recall that the Monte Carlo rules for our protein simulation first generate a potential new state (that is, tertiary structure), and calculate how much energy it would cost, ΔE_{move} , to change to that state. If $\Delta E_{\text{move}} < 0$ so that the new

⁹Since a protein is in thermal equilibrium at some temperature T , there will always be fluctuations away from the equilibrium state. If a protein is to be biologically active, it must spend most of its time in its optimal structure. Ignoring these fluctuations amounts to assuming that the protein is always in the state with the lowest energy.

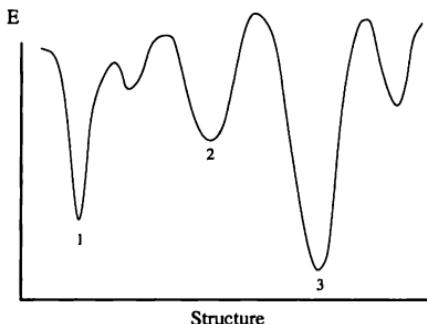


FIGURE 12.12: Schematic energy landscape. Different points along the horizontal axis correspond to different tertiary structures. The structures labeled 1 and 2 are metastable, while structure 3 has the lowest energy and is, therefore, the stable one at low temperatures.

state has a lower energy, this transition is always made. In the energy landscape picture this corresponds to moving downhill. On the other hand, if $\Delta E_{\text{move}} \geq 0$, the transition is made with probability $\exp(-\Delta E_{\text{move}}/k_B T)$. The Monte Carlo procedure thus allows the system to move uphill to higher energy states, although the probability of such a move decreases as ΔE_{move} becomes larger and as T is made smaller. Nevertheless, this makes it possible for a system to escape from a metastable state. The time it takes to escape will depend on both the depth of the metastable potential well and on temperature, but as long as T is not zero there will be some chance of escape.¹⁰

These ideas can be applied to our protein problem by beginning the simulation at a temperature that is high enough that the system does not become trapped in any of the metastable potential wells. The system is thus able to explore a large number of different structures. The temperature is then slowly lowered, and while the protein will continue to fluctuate between different structures, it will spend increasing amounts of time in those states with the lowest energies. When the final, low temperature is reached, the protein will be trapped in some state, but the hope is that it will have sampled many other states and thereby managed to choose the optimal structure.

The usefulness of this approach can be studied by slowly sweeping the temperature downward in a simulation. Some results of such a calculation are shown in Figure 12.13. Here we studied the model protein that became stuck in metastable

¹⁰Here you might object to our use of the Monte Carlo method to address dynamical questions. In Chapter 8 we noted that the Monte Carlo transition rules are designed to lead to a collection of microstates with the correct equilibrium statistics, but that these rules do not correspond to the correct dynamical equations. While the same is true here, the fact that the Monte Carlo algorithm leads to the correct Boltzmann probabilities for finding a system in any particular microstates enables us to use this method for dealing with questions of metastability and the relative amounts of time spent in different states.

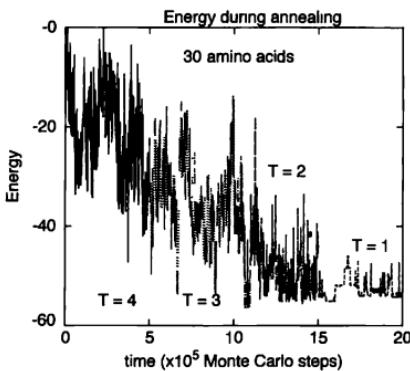


FIGURE 12.13: Annealing simulation. The chain of 30 amino acids considered in Figure 12.11 was simulated starting at $T = 4$ at time $t = 0$. The temperature was then reduced in stages after every 5×10^5 Monte Carlo time steps to $T = 3, 2$, and 1 as indicated.

states in the simulations of Figure 12.11. We started at $T = 4$, which we have already seen to be high enough that the fluctuations are large. The temperature was then reduced in stages to $T = 3$ after 5×10^5 Monte Carlo steps, $T = 2$ after 1×10^6 steps, and finally to $T = 1$ after 1.5×10^6 steps. The final temperature was thus the same as that employed in Figure 12.11. This gradual reduction of temperature, which is known as *annealing*, resulted in a protein structure that was better in terms of the energy than any obtained in Fig 12.11, even though those computations at a fixed low temperature involved much longer simulations. Our annealing procedure was thus successful in avoiding, at least to some extent, problems with metastability in this particular case.

While these results show that annealing can avoid some of the problems with metastability, this procedure is not guaranteed. There is always some chance that the chain will end up in the wrong state; we will consider this and some related issues in the exercises. At this point it is worthwhile to discuss what our simulations have taught us, and what they haven't, about the folding of real proteins. First, our model does exhibit a transition to a folded structure when the temperature is made sufficiently low. This agrees with the behavior observed in experiments. Second, the model does not avoid the trap associated with Levinthal's paradox. In general, our model protein is not able to avoid being trapped in a metastable state. However, this trapping can be greatly reduced by the procedure of annealing. The implications for real proteins is not completely clear,¹¹ but one interesting scenario is the following. It is possible that a real protein is effectively annealed when the environmental conditions are changed. For example, the imposition of changes in

¹¹Don't expect too much; this is not yet a "solved" problem!

temperature or chemical conditions will always take some time, which is modeled in our calculations by the annealing procedure in connection with Figure 12.13. This sort of annealing may allow a protein to, on average, avoid falling into metastable states. Such a solution may only need to be successful on average, as nature may be able to tolerate the small number of proteins that would end up in metastable, and presumably biologically inert, states.

While annealing may help proteins avoid becoming trapped in metastable states, this alone would not resolve Levinthal's paradox. Our simulations imply that relatively short chains are able to explore a sufficiently large number of structures so as to locate the proper final state. But, we have already argued that the number of potential states of a real protein is much larger than for our small models. Even if annealing enables a real chain to avoid metastable states, this will not be a satisfactory solution unless the folding process can occur rapidly. To address this problem we must explore the nature of the energy surface in Figure 12.12, and consider the specific path a protein might follow in locating its proper structure. A solution to Levinthal's paradox would then involve showing that the length of this path does not grow unbearably long for a real protein.

Yet another (related) possibility is that proteins do *not* actually avoid Levinthal's paradox at all. As we have seen, there are a great many possible proteins, that is, primary structures; nature has chosen to make use of only a relatively few of these possibilities. It is conceivable that the ones nature has chosen are those whose energy surfaces allow them to efficiently avoid metastable states on their way to a unique tertiary structure. If so, Levinthal's paradox would be intimately connected with the evolutionary development of proteins. There has been much recent work on these issues, and the interested reader can learn about this work by exploring the references.

The correct solutions to the protein-folding problem and Levinthal's paradox are not yet known. Nevertheless, our simple simulations of protein folding have demonstrated how ideas developed with physics in mind can be very helpful in dealing with a problem that lies in the territory between physics and biology. As is often the case, our model raises as many questions as it answers (or more). Some might say that this is the sign of a good model!

EXERCISES

- 12.1. Simulate the folding transition in three dimensions by allowing the protein to sit on a simple cubic lattice. Compare the results with the behavior in two dimensions. Of particular interest is the width of the pseudotransition from the unfolded to folded state.
- 12.2. Study the importance of the way in which the interaction energies $J_{i,j}$ are distributed. One possibility is to compare the behavior of a chain in which all of the $J_{i,j}$ are the same, with that of a chain in which $J_{i,j}$ has the same magnitude, but with randomly varying signs (i.e., $J_{i,j} = \pm 1$). This would mimic the fact that some amino acids are attracted to water molecules, while others are repelled. The time it takes to reach equilibrium and the nature of the metastable states at low temperature are of special interest.

- 12.3.** In our estimates of the length of a protein, we have used the end-to-end distance. This very simple (to calculate) measure may not tell the whole story. For example, a protein could contain a single fold or have a form like a “ball of string,” and have the same end-to-end length. Investigate the behavior of the mean-square size calculated in the following way. Let \vec{r}_{cm} be the location of the center of mass of the protein. One measure of the size of the chain is the quantity $\Delta \equiv <|\vec{r}_i - \vec{r}_{cm}|^2>$ where \vec{r}_i is the position of the i th amino acid and the angular brackets denote an average over all pieces of the chain. Calculate Δ as a function of temperature and compare its behavior to that of the end-to-end distance.
- *12.4.** Perform a simulation of protein folding and examine the variation of the energy as a function of (Monte Carlo) time. Use this, and any other approaches you can devise, to reconstruct part of the energy landscape, as sketched in Figure 12.12. Can you say anything about the number of wells as a function of their depth? How does this distribution change as the protein chain is made longer?
- *12.5.** Investigate the problem of metastability of protein folding by comparing the structure obtained by two or more separate simulations of the same model protein. Consider a chain with 30 amino acids, and let it find two or more metastable states by letting it fold at $T = 1$, as in Figure 12.11. Then compare the actual structures of the different folded states. Are the structures similar or very different? Can you estimate the size of the energy barriers that separate the different metastable states?

12.2 EARTHQUAKES AND SELF-ORGANIZED CRITICALITY

Earthquakes often have a large and dramatic impact, which makes them a topic of continuing interest. Earth’s crust contains numerous fault lines that separate large pieces of material called *plates*. Each of these plates is fairly sturdy, but the connections across a fault line are relatively weak. Over time the crust deforms, exerting forces on the plates and leading to a gradual build up of potential energy. This energy is released by the sudden movement of one plate relative to an adjacent one. Such an event is an earthquake. While this general picture of earthquakes is well established, there are many questions that are not settled. For example, we would like to know how to predict when earthquakes will occur and how large the next quake associated with a particular fault line will be.

Geoscientists have been involved in modeling earthquakes for many years. In this section we follow their lead and model two adjacent pieces of Earth’s crust as masses that are able to slip past each other in response to a steadily increasing force. Such a mechanical model involves Newton’s second law, which gives a small excuse for considering this to be a physics problem. However, there is another feature of earthquakes that makes them of interest to physicists. It has been proposed that earthquakes may have some important features in common with the second order phase transition we observed in connection with the Ising model in Chapter 8. In order to understand this connection we need to introduce the so-called Gutenberg-Richter law, which can be stated as follows. The size of an earthquake is often measured using the Richter scale, which is commonly referenced by the popular press. This is a logarithmic scale involving the magnitude of an earthquake. The amount that one of Earth’s plates shifts relative to another during an earthquake is

proportional to the moment of the event, M . This quantity is also proportional to the energy released by the event. The magnitude of the quake, \mathcal{M} , is equal to the logarithm of the moment, so an earthquake with a magnitude of 7 on the Richter scale is much more powerful than an event whose magnitude is 6.

A logarithmic scale is convenient because earthquakes come in an extremely wide range of sizes. Fortunately, the number of large quakes is much smaller than the number of little ones. This “preference” for small events is well-documented from observations and also follows a logarithmic form. This is known as the Gutenberg-Richter law and can be stated mathematically as

$$P(\mathcal{M}) = A M^{-b} = A e^{-b\mathcal{M}}. \quad (12.2)$$

Here $\mathcal{M} \equiv \ln M$ is the magnitude of an event,¹² $P(\mathcal{M})$ is the probability (per unit \mathcal{M}) of having a quake of a given magnitude, A is a constant, and b is a factor that lies somewhere in the range 0.8–1.5. The use of the term law in connection with (12.2) is perhaps a bit too strong, as it is really just an empirical rule that has been found to describe the distribution of earthquake magnitudes observed for many different fault lines. Surprisingly, there is no fundamental understanding of why earthquakes (or Earth itself?) follow this rule. In particular, why doesn’t $P(\mathcal{M})$ vary as $e^{+\mathcal{M}}$, or even $\mathcal{M}^{-\pi}$?

The Gutenberg-Richter law is also interesting for what it implies about the amount of energy released in a typical event. Since the energy released is proportional to M , the average energy of an event is just the integral of M over the distribution (12.2)

$$E_{\text{average}} = \int_0^{\infty} E A e^{-b\mathcal{M}} d\mathcal{M} \sim \int_0^{\infty} M e^{-b\mathcal{M}} d\mathcal{M}. \quad (12.3)$$

Since $M \sim e^{\mathcal{M}}$ (and given the observed range of b) this integral *diverges!* Fortunately it appears that such “average” earthquakes don’t happen very often.¹³ More seriously, power law distributions such as (12.2) which have awkward (or infinite) averages, are quite rare in nature. Perhaps the best-documented and understood case in which such distributions occur is near a second-order phase transition. You may recall that in our studies of the Ising model in Chapter 8 we noted that many quantities exhibit power law singularities at a critical point. For example, the correlation length associated with fluctuations of the magnetization is infinitely large when $T = T_c$. This analogy has suggested to some researchers (see Bak and Tang [1989]) that Earth is effectively located at a critical point as far as earthquakes are concerned. The fact that there is a sort of earthquake phase transition for a certain value of Earth’s density and temperature, etc., might seem unlikely, but it is at least plausible. However, it would be even more surprising to find that Earth just happens to have a temperature and density that place it very near this transition. The interesting speculation is that some feature of this phenomena *automatically* causes Earth to be located at the transition. This scheme is known

¹²We could choose to use either natural or base-10 logarithms. Here we follow the convention employed in Carlson and Langer (1989) and Carlson (1991).

¹³This strongly suggests that the Gutenberg-Richter law must break down at large \mathcal{M} .

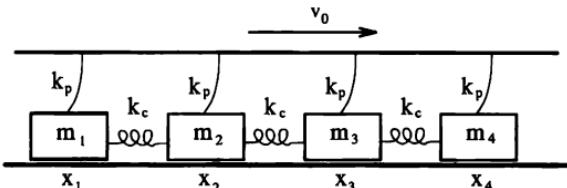


FIGURE 12.14: Model of two plates separated by a fault line at which an earthquake can occur. Imagine this to be a top view of Earth's surface, with the fault line running between the blocks and the bottom plate.

as *self-organized criticality*. The relevance of this concept to any real systems is not universally accepted, but this discussion does emphasize the striking nature of power law distributions such as (12.2). It has also prompted many researchers to try to account for such distributions in terms of (semi)realistic models, and that is the subject of this section.

The model we consider is copied from one proposed by Burridge and Knopoff (1967), and discussed by Carlson and Langer (1989); it is shown in Figure 12.14. We imagine that two of Earth's plates are moving slowly relative to one another. One of the plates is the bottom surface (the lower, thick horizontal line) in Figure 12.14, while the other plate is the top surface. Caught between them is a portion of the crust modeled by a collection of blocks. For simplicity we will assume that the blocks are arranged in a line, but we can also consider a two-dimensional array (we will explore this possibility in the exercises). The blocks are connected to each other by a force that is modeled as springs, with force constants k_c . The blocks are also connected to the top plate via "leaf" springs,¹⁴ k_p . The only other force in the problem is a frictional force between the blocks and the bottom plate, which we will describe in detail shortly.

The top plate in Figure 12.14 is assumed to move to the right with a constant velocity v_0 . Thus, through the leaf springs it exerts an ever-increasing force on the blocks. When this force is small, the frictional force from the bottom plate will prevent the blocks from moving, and energy will build up in the potential energy of the leaf springs. Eventually the force from these springs will overcome the frictional force, and one or more blocks will move suddenly. This is an earthquake. Since the blocks are connected to each other by the springs k_c , the motion of one block can cause other blocks to move as well. If this motion spreads to involve many blocks, one slip will lead to a large quake.

An important ingredient in the model is the frictional force between a block and the bottom surface. We refer to this force as friction because it is assumed to exhibit the general features we all learn in our elementary mechanics courses. When a block is stationary relative to the bottom plate the force is static friction, while if

¹⁴While most springs have a helical form, leaf springs are a single strip of material. They resist bending (that is, "spring" back) much like a stem or leaf would if they were bent and then released.

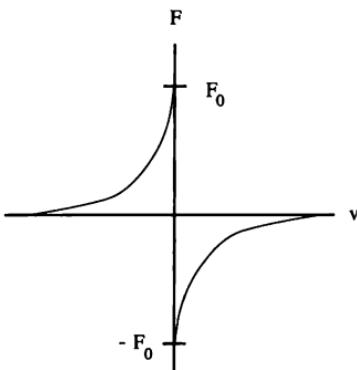


FIGURE 12.15: Schematic of the frictional force between a block and the bottom plate as a function of the velocity of the block

it is moving we are dealing with kinetic friction. Furthermore, we are taught that the maximum force of static friction is greater than the force of kinetic friction. This is sometimes referred to as a stick-slip force, since once an object begins to move, the frictional force becomes smaller, resulting in a sudden increase in the velocity. Figure 12.15 shows the general form we will use for the frictional force. It always opposes the relative motion of the block and the surface, and is largest when the velocity of a block relative to the bottom plate, v , vanishes. We will follow previous work on this model (see Carlson and Langer [1989]), and assume that the magnitude of this frictional force decreases with increasing $|v|$. Unfortunately, there is no fundamental understanding of such frictional forces (in contrast, for example, to the Van der Waals force), so it is hard to put this important feature of the model on a solid basis. We can, however, investigate how the form of this force affects the properties of the model, and we will return to this point below.

Before we consider in detail how to calculate the behavior of the model, it is worthwhile to make a few comments with regard to model building. Our goal in studying this model is *not* to reproduce the detailed behavior of Earth near any particular fault line. Instead, our aim is to determine what properties a system must have to exhibit a power law distribution of earthquake sizes (12.2). We have already encountered systems with simple harmonic forces similar to the spring forces in Figure 12.14.¹⁵ The only unusual feature of the model we are considering here is the frictional force, so we expect that this must be the key for obtaining power law behavior. Confirming this suspicion, or showing it to be false, will be our initial concern. If we do find power law behavior, then we can conclude that we have

¹⁵We saw in Chapter 3 that this force leads to simple harmonic motion. The interesting chaotic behavior was found only when we considered deviations from a purely harmonic force.

(perhaps) captured the essential physics of the problem. If power law behavior is not found, we will be forced to continue our search to identify the key ingredient responsible for (12.2). This could require that we consider other functional forms for the frictional force, or that we generalize the model in other ways.

This is the motivation behind model building in theoretical physics. The goal is not simply to construct a simulation that reproduces nature,¹⁶ but rather to identify the essential physics responsible for the interesting behavior. This process can, of course, be iterated as we attempt to bring a model ever closer to reality.

Now let us return to our earthquake model and consider the forces in a little more detail. The model consists of N blocks whose positions are x_i , where i ranges from 1 to N (see Figure 12.14), and for simplicity we assume that all have the same mass, m . The force between two adjacent blocks is due to the spring that connects them. The force from a spring is given by Hooke's law, and has the form $F = -k\Delta x$, where k is the force constant and Δx is the amount that the spring is stretched or compressed relative to its relaxed state. It is convenient to measure each x_i with respect to the equilibrium position of block i . Hence, $x_i = 0$ if a block is at its equilibrium location. The force on block i from its neighboring blocks is then

$$F_b = -k_c(x_i - x_{i+1}) - k_c(x_i - x_{i-1}). \quad (12.4)$$

Finally, we note that our system will have “free” ends. The blocks at each end will be connected to only one other block.¹⁷

The force of the leaf spring on block i has a similar form, $F = -k_p(x_i - x_{\text{leaf}})$. We assume that at $t = 0$ the leaf springs are all unstretched, so that initially $x_{\text{leaf}} = 0$ for each block. The horizontal bar moves with velocity v_0 , so x_{leaf} increases with time according to $x_{\text{leaf}} = v_0 t$. The force of the leaf spring on block i is then

$$F_l = -k_p(x_i - x_{\text{leaf}}) = -k_p(x_i - v_0 t). \quad (12.5)$$

The only remaining force is that due to friction with the bottom plate. We will assume that it has the form shown in Figure 12.15. When the velocity of a block is zero the frictional force will take on whatever value is necessary to keep the block at rest. That is, the frictional force will oppose the other forces on the block so that the sum of all of the forces (friction included) vanishes. However, the static frictional force is limited to a maximum magnitude of F_0 , so if the sum of the other forces exceeds this level, the block will experience a nonzero force and begin to move. If the block is moving we are then dealing with kinetic friction, which we will assume is given by

$$F_f = -\frac{F_0 \text{sign}(v_i)}{1 + |v_i/v_f|}, \quad (12.6)$$

¹⁶In such a case we could say that the computer understands the problem; we want to understand it, too.

¹⁷We will leave the study of the effects of periodic boundary conditions to the interested reader. In this problem periodic boundary conditions seem unphysical. In particular, we might imagine that some earthquakes start at the end of a fault and propagate inward. Such behavior would not be possible if the model employed periodic boundary conditions.

where v_f is a parameter that determines the velocity dependence of the force. When $v_i = v_f$, the frictional force drops to half of its $v_i = 0$ value. The factor $\text{sign}(v_i)$ ensures that F_f always opposes the motion.

Using springs to model the interactions between blocks and between a block and the opposite side of the fault line may seem a bit contrived, but it is actually on firm mathematical footing for the following reason. The energy of interaction between two blocks will, in general, be a function of the separation between the blocks; let us call this function $U(\Delta x)$, where $\Delta x \equiv x_{i+1} - x_i$. Assuming that $U(\Delta x)$ is a well-behaved function, we can perform a Taylor expansion

$$U(\Delta x) = U(0) + (\Delta x) U' + \frac{(\Delta x)^2}{2} U'' + \dots, \quad (12.7)$$

where U' is the first derivative of U evaluated at $\Delta x = 0$, etc. The corresponding force is $F = -dU/d(\Delta x)$

$$F(\Delta x) = -U' - (\Delta x) U'' - \dots. \quad (12.8)$$

By definition, this force vanishes when the blocks are at their equilibrium spacing, so U' must be zero. For small Δx we thus have $F \approx -(\Delta x) U''$, which is just Hooke's law with $k = U''$. Hence, the form of Hooke's law is a natural result for a force that arises from a well-behaved (Taylor expandable) potential energy function. This is one reason why springs are a popular ingredient in the models devised by physicists. They are in fact a very natural and general way to describe an interaction.

On the other hand, the basis of the frictional force (12.6) is not nearly as firm. As we have already noted, there is no fundamental understanding of friction. The best we can do is assume a simple form such as (12.6) and study the kind of behavior it yields. We will return to this point later.

Putting all of these forces together with Newton's second law yields an equation of motion for each block

$$m_i \frac{d^2 x_i}{dt^2} = k_c (x_{i+1} + x_{i-1} - 2x_i) + k_p (v_0 t - x_i) + F_f. \quad (12.9)$$

This can be written as two first-order differential equations,

$$\frac{dx_i}{dt} = v_i, \quad (12.10)$$

$$m_i \frac{dv_i}{dt} = k_c (x_{i+1} + x_{i-1} - 2x_i) + k_p (v_0 t - x_i) + F_f, \quad (12.11)$$

and this system of equations can be solved using the Euler method. As usual, we discretize time into steps Δt . At every time step we use the velocity of each block to estimate its position at the next step. We also calculate the force on each block and use it to obtain the velocity at the next time step. Note that the forces are functions of the current positions, so to be consistent with respect to the spirit of the Euler method we must calculate the forces on *all* of the blocks before updating the positions and velocities.¹⁸

¹⁸Updating in a different order could easily yield the Euler-Cromer method. For this problem the Euler and Euler-Cromer methods are both acceptable algorithms.

The programming is similar to what we have encountered in several previous cases, including the projectile and pendulum problems. For a system of N blocks we have to keep track of N different positions along with the corresponding velocities. The only really new feature is the frictional force. In order to model this force properly there are several different cases that must be considered.

- The block is not moving at time-step n , and the sum of the forces from the block springs and the leaf spring is smaller (in magnitude) than F_0 . The static frictional force will then adjust itself to precisely cancel the other forces. Since the total force will thus be zero, the velocity at time step $n + 1$ will also be zero.
- The block is not moving at step n , and the sum of the forces from the block springs and the leaf spring is greater than F_0 . The frictional force will have a magnitude of F_0 , and oppose the sum of the other forces. The total force will not be zero and the velocity at the next time step will be nonzero.
- The block is moving at step n . We calculate the velocity for time step $n + 1$ using the (kinetic) frictional force (12.6), along with the forces from the block and leaf springs. If this new velocity has the same sign as the velocity at step n , everything is fine and the calculation proceeds in the usual way. However, if the new velocity would be *opposite* to the previous velocity, this means that the frictional force is sufficiently large that it will "capture" the block; that is, bring it to rest. In this case the velocity at time step $n + 1$ must be set to zero.

This description of the frictional force is actually much longer than the number of lines needed to implement it in a program. However, it does show that a force that is a discontinuous function generally requires some extra care.

We are now ready to consider the behavior of our earthquake model. It contains 5 parameters, k_p , k_c , m , F_0 , and v_0 . Some of these can be effectively removed by the appropriate choice of units, but there will still be a large number of parameter choices to explore. In most of the simulations below we will use the following values: $m = 1$, $k_p = 40$, $k_c = 250$, $F_0 = 50$, and $v_0 = 0.01$. These values appear to give fairly typical behavior, other parameter values will be considered in the exercises (see also Carlson and Langer [1989] and Carlson [1991]).

In addition to these parameters, we must specify the initial conditions. For simplicity we will always assume that the initial velocity of each block is zero, but this still leaves us with the choice of initial positions. One choice is to begin with the blocks all located in their equilibrium positions. In this case the forces on all of the blocks from both types of springs is zero at $t = 0$. Such a perfectly ordered start is not very realistic, but is useful for illustrating a few important points. Some results for this case are shown in Figure 12.16, where we plot the position and velocity of a particular block as functions of time. This simulation involved 25 blocks, but since the initial conditions were uniform, the behavior of every block was the same as that shown here. That is, they all moved together, and the springs k_c were never stretched or compressed. We see from Figure 12.16 that the block remained at $x = 0$ until $t \sim 120$. During this time the opposite side of the fault was moving steadily at speed v_0 and the force from the leaf spring gradually increased. This force was not able to overcome friction until $t = 120$, at which point the *entire* system of blocks began to move. The blocks moved approximately 2 units

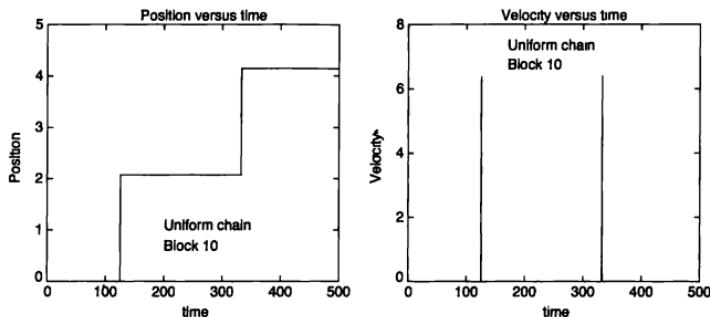


FIGURE 12.16: Behavior of block 10 in a 25-block system, with a completely ordered start. Left position of block 10 as a function of time, right velocity of the same block. The velocity was zero except for the two very narrow spikes at $t \sim 120$ and ~ 330

before the frictional force brought them to a halt. The process then repeated. The force from the leaf springs grew until it overcame friction at $t \sim 330$, and the blocks all moved again, etc. The corresponding velocity is also shown. It was zero until the blocks moved, at which point it exhibited a large but very narrow peak. Of course, this is just the derivative of the position as a function of time.

The two displacements in Figure 12.16 correspond to an abrupt motion of the system of blocks. These are earthquakes. The quakes found with an ordered start are very special, since the blocks all move together and the events occur at regularly spaced intervals.¹⁹ This will change when we start the blocks with random initial positions. However, before we do that it is useful to compare the behavior in Figure 12.16 with some simple, analytic results. Because of the special initial conditions, the block springs k_c were never stretched or compressed. We thus need consider only a single block moving in response to the leaf spring and the force of friction. Initially, the force from the leaf spring was $k_p(v_0 t - x) = k_p v_0 t$, since $x = 0$ was the initial condition. The block will not move until this exceeds F_0 , which occurs at $t = F_0/(k_p v_0)$. For the parameters used in here this yields $t = 120$, in good agreement with Figure 12.16. The block should then move until the frictional force (12.6) is equal to the force from the leaf spring. We will leave it to the exercises to check that the displacements in the two events in Figure 12.16 agree with the expected value. The peaks in the velocity can be estimated in a similar way.

The results in Figure 12.16 are useful since they allow us to check our program against analytic results. This behavior also brings out an important programming issue. The earthquakes occur over a very short time compared to the interval

¹⁹We only showed two quakes here, but if we had shown the behavior for longer times, you would have seen that the quakes do indeed repeat at regular intervals

between quakes. The duration of a quake is too small to resolve on the scale in Figure 12.16, but is of order 0.5 time units. This implies that the time step in our simulation must be a small fraction of this to avoid significant numerical errors. Our usual practice is to use a time step that is 1% of the characteristic time scale of the problem, which would thus be ~ 0.005 . The time between quakes is on the order of 100 time units, so this would lead to $\sim 2 \times 10^4$ time steps between events. A simulation using a large number of blocks would thus take a lot of computer time. Moreover, there would be nothing happening for the vast majority of this time.

There are two ways to deal with this problem: use a fast computer and just be patient, or use *two* different time steps according to the following strategy.²⁰ During the times when no blocks are moving we use a (relatively) large time step. The only motion during these periods involves the top plate, and since it moves with a constant velocity, the use of a large time step does not introduce any errors. A small time step (~ 0.005 in the above example) is used during the times when the blocks are moving. This strategy is straightforward, except that we need to have a systematic way to switch back and forth between time steps. One convenient way to make these switches is as follows. When the blocks are not moving, the larger time step is used to calculate the new velocities. If the velocity of *any* block is nonzero at the next time step, an earthquake is imminent. We then “back up” to the previous time step and continue the calculation with the smaller value of the time step until after the upcoming quake is finished. When the velocities of all of the blocks are again zero, the time step is set back to the larger value and the calculation proceeds. The results in Figure 12.16 were obtained with this algorithm, using a large time step of 0.03 and a small time step of 0.003. We will leave it to the exercises to check that these values are sufficiently small that the numerical errors were negligible.

A simulation with an initially ordered configuration is not very realistic, since we don’t expect Earth’s crust to ever be perfectly uniform. The behavior is quite different when the blocks are given a disordered initial configuration. If we displace them initially from their equilibrium positions by random amounts in the range ± 0.001 , we find the results shown in Figure 12.17. Here we show only four quakes and it is seen that they had different magnitudes, that is, different total displacements. In addition, the time until the next quake varied from event to event.

It is intriguing that such a small initial displacement (only 0.1% of the spacing between blocks) is able to produce such dramatically different behavior. It turns out that the behavior found in Figure 12.16, with a perfectly ordered start, is *unstable* with respect to any initial displacements from equilibrium, no matter how small. That is, the behavior is extremely sensitive to small deviations from a perfectly ordered start. This should remind you of chaotic systems and their extreme sensitivity to initial conditions; this is our first indication that this earthquake model is not a “simple” mechanical system.

²⁰The use of two (or more) time steps is often referred to as an “adaptive” step-size procedure. The example we describe here is a very simple one, but does illustrate the basic idea. Such an approach is useful in simulations involving functions or behaviors that have significant structure limited to small regions of time or space.

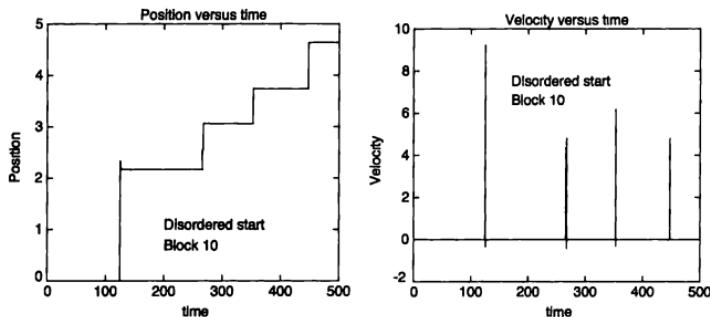


FIGURE 12.17: Behavior of block 10 in a 25-block system. Each block was given a random initial displacement from its equilibrium position. This displacement was in the range -0.001 to $+0.001$. Left: position of block 10 as a function of time; right: velocity of the same block.

So far we have examined the behavior by following the motion of a single block. However, since we expect that there can be earthquakes that do not involve all of the blocks, it is useful to view the same behavior using the perspective plots in Figures 12.18 and 12.19, which show the behavior of the entire system. With an ordered start all blocks move together, as we had anticipated. In contrast, with a disordered start the quakes are much less organized. There are numerous events in a time interval that would contain only one or two quakes for the case of an ordered start. Some of these events involve many blocks, while in others only a few blocks are in motion. We thus have a distribution of earthquake sizes.

One of our primary goals is to try to understand the origin of the Gutenberg-Richter law, and to do this we need to add one more feature to the simulation. As we mentioned earlier, the magnitude of an earthquake is the natural logarithm of the earthquake moment. The moment M is proportional to the total displacement, which can be found by summing (integrating) $v_i \Delta t$ for each block over the course of the event. The moment of an event is thus

$$M = \sum_{n=\text{time}} \left(\sum_{i=\text{blocks}} v_i \Delta t \right), \quad (12.12)$$

where the sums are over all blocks i and over the time steps n for which the velocities are not all zero. The magnitude of the event is then $\mathcal{M} \equiv \ln M$. After accumulating the results for a large number of events we can obtain the distribution $P(\mathcal{M})$ by dividing the \mathcal{M} axis into bins and counting the number of events that fall into each bin. Results for $P(\mathcal{M})$ are shown in Figure 12.20, where the figure on the left shows the distribution for the system of 25 blocks we have considered in all cases to this point. On this semilogarithmic plot the Gutenberg-Richter law (12.2) is a straight line with slope $-b \log_{10}(e) \approx -0.43b$. The results from the simulation are certainly

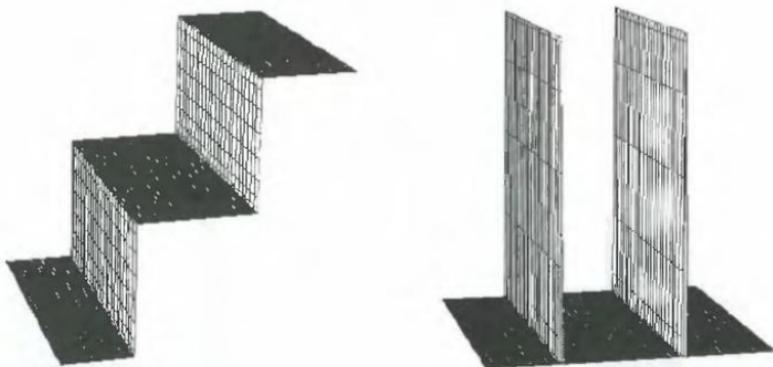


FIGURE 12.18: Behavior of the entire system for the simulation with an ordered initial configuration, Figure 12.16 Time goes from left to right, block number from front to back, and the vertical axis is position (left figure) or velocity (right figure). The time span here is $t = 0$ to 500, the same as that covered in Figure 12.16

not a simple straight line. However, a 25-block system intuitively seems to be a bit too small to be a good model for an entire fault line. We might imagine that such a small number of blocks would artificially restrict the possible event magnitudes and perhaps affect the number of large earthquakes. It is therefore worthwhile to consider larger systems, and results for simulations with 100 and 500 blocks are also shown in Figure 12.20. All three simulations exhibit the same type of deviation from the Gutenberg-Richter law, as all show an excess number of large events.²¹ It is interesting to note that there is actually a regime of intermediate M where $P(M)$ does vary approximately linearly with M . For example, over the range $M \sim -2$ to +2 the 500-block system exhibits an approximate power law, as indicated by the solid line in Figure 12.20. This line is just a plot of the Gutenberg-Richter law (12.2) with $b \approx 0.7$. Hence, over this rather limited range, the system does appear to at least roughly follow the Gutenberg-Richter law, although the value of b is a little lower than that exhibited by nature. While our simulations thus approximate nature to some degree, the excess number of events at large M and the rather low value of b suggest that the model is lacking an important ingredient.

The simulations we have described above have been only partially successful

²¹There is also an excess number of events at very low M in the lowest bin of the distribution, but this is less troubling than the problem at high M , for the following reason. There will be a minimum earthquake size corresponding to the motion of a single block for one time step, and this will result in a pile-up of events of some minimum size. The difficulty is thus with the numerical approach and is not an intrinsic property of the model.

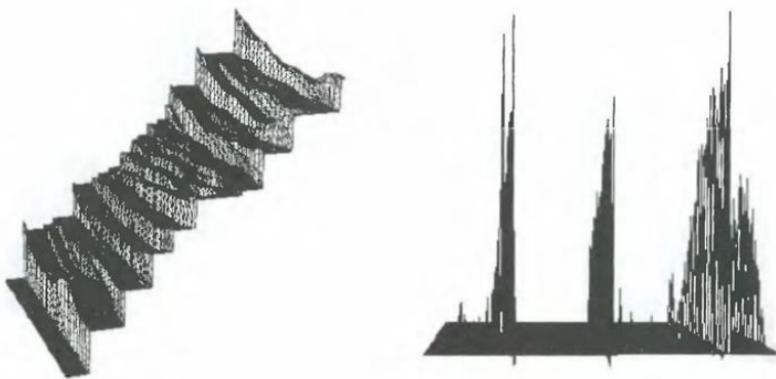


FIGURE 12.19: Results for the simulation of Figure 12.17 in which the blocks were in an initially disordered configuration. Time goes from left to right, block number from front to back, and the vertical axis is position (left figure) or velocity (right figure). The time span covered in the position plot is $t = 0$ to 1000, but for purposes of clarity the velocity plot shows a smaller range, $t = 300$ to ~ 500 .

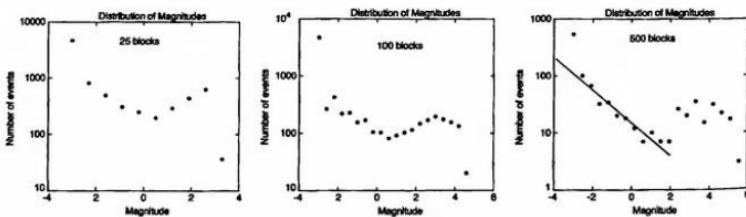


FIGURE 12.20: Earthquake distributions for systems of various sizes. These histograms were obtained from a collection of approximately 7000 events for the 25- and 100-block systems and 3000 events for the 500-block simulation. The vertical axis is the number of events per histogram bin and is thus proportional to the probability density of events.

in explaining the Gutenberg-Richter law. However, we have been able to show that this simple mechanical model is capable of exhibiting power law behavior over at least a limited range. To some small extent this lends support to the proposals concerning self-organized criticality mentioned at the beginning of this section. As for the relevance to real earthquakes, it has been suggested that the Gutenberg-Richter law may actually fail at large M , so perhaps part of our problem is with the law itself, rather than the model.²² Of course, it is also possible that the problem lies with the model. You will recall our philosophy of model building, according to which we strive to construct the simplest model that contains the essential physics of the phenomena of interest. It is certainly conceivable that our simple model has omitted some key element(s). Possibilities include the following: (1) The dimensionality of the fault system; a two-dimensional array of masses might be more appropriate than the one-dimensional arrangement considered above. Here the second spatial dimension would correspond to depth beneath Earth's surface. (2) We have assumed uniform values of m , k_c , and k_p . For a real fault the analogous parameters will not be constants, but vary with position. (3) The frictional law (12.6) has no fundamental basis. We could certainly imagine other plausible possibilities. These are just a few of the ways in which the model could be modified, and we will leave such studies to the exercises. While we have not been able to answer all of the questions concerning earthquakes posed at the beginning of this section, these simulations do shed some light on the problem, and serve to illustrate the model-building process in theoretical physics.

EXERCISES

- 12.6. Consider the simulation in Figure 12.16 in which 25 blocks were given a perfectly ordered arrangement at the start. Continue this simulation to longer times and show that the earthquakes occur at regularly spaced intervals (as we claimed above).
- 12.7. Perform a simulation with 25 blocks, allowing for some randomness in either the masses (let m_i vary from 0.5 to 2.0) or in the spring constants (either k_c or k_p). Compare your results for the distribution of earthquake magnitudes with the results in Figure 12.20. The objective is to see if adding some disorder can lead to better agreement with the Gutenberg-Richter law or reduce the excess number of events at high M , or both.
- 12.8. Assume that the blocks are all initially in their equilibrium positions and obtain analytic estimates for the time between quakes, the displacement of a block during a quake, and the maximum velocity during a quake. Compare these estimates with the results in Figure 12.16.
- *12.9. Explore the properties of a two-dimensional earthquake model. A calculation of this kind is described in the references.
- *12.10. Investigate how the distribution of earthquake magnitudes depends on the form chosen for the frictional force. As an example, consider the case $F_f = F_0$ when $v = 0$ (static friction) and $F_f = -\text{sign}(v)F_0/2$ for $v \neq 0$ (kinetic friction). You should find (see Figure 12.21) that with this friction law there is no longer an excess of events at large M , so the results are more realistic than that obtained

²²This is a difficult issue to resolve, since the number of large quakes is (fortunately) small, making it hard to get a good estimate of $P(M)$ at large M .

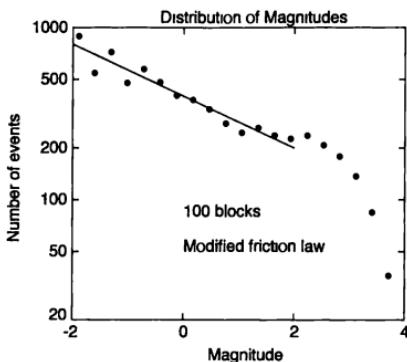


FIGURE 12.21: Earthquake distribution for a system of 100 blocks and the friction law described in Exercise 5. The straight line corresponds to the Gutenberg-Richter law (12.2) with $b \sim 0.35$.

with the frictional force (12.6). However, the slope for small \mathcal{M} now corresponds to a value of b that is much smaller than 1, so the model still seems to lack an important ingredient. Study the behavior with other forms for the frictional force and try to determine one that gives a power law with a larger value of b .

12.3 NEURAL NETWORKS AND THE BRAIN

The Ising model consists of a large number of very simple units, that is, spins, which are connected together in a very simple manner. By “connected” we mean that the orientation of any given spin s_i , is influenced by the direction of other spins through the interaction energy $J s_i s_j$. The behavior of an isolated spin, as outlined in our discussions leading up to mean-field theory, was unremarkable. Things only became really interesting when we considered the behavior of a large number of spins and allowed them to interact. In that case we found that under the appropriate conditions some remarkable things could occur, including the singular behavior associated with a phase transition. In this section we will explore a rather different system, which shares some of these features.

The human brain consists of an extremely large number ($\sim 10^{12}$) of basic units called neurons, each of which is connected to many other neurons in a relatively simple manner. A biologically complete discussion of neurons and how they function is a long story. Here we will give only a brief description of those features that seem to be most relevant to a physicist’s understanding of the brain. A schematic picture of two neurons is given in Figure 12.22. Each neuron has a body (called a soma), along with dendrites and an axon.²³ The size scale depends on the type of neuron,

²³There are other parts as well, but keep in mind that we are giving only a simplified description here.

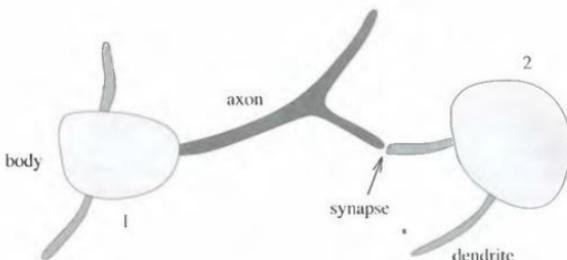


FIGURE 12.22: Schematic of two connected neurons

but the body is typically of order $10 \mu\text{m}$ across. Neurons are electrically active and communicate with other neurons through electrical signals carried by the dendrites and axons. The dendrites serve as the input lines of a neuron, while the axons are output lines. The axon of one neuron is “connected” to the dendrite of another via a synapse, through which is transmitted the electrical output signal of one neuron to the input of the other.

The axon of the hypothetical neuron (number 1) on the left in Figure 12.22 connects to a dendrite of neuron number 2 via the synapse, as indicated. A very important feature is that an axon is generally split into many branches, and thereby connects to many other neurons. Correspondingly, each neuron generally has many dendrites and thus accepts inputs from many other neurons. When we say many, the number we have in mind is typically $\sim 10^4$. The neurons are thus highly interconnected.

Neurons communicate using electrical pulses carried by the axons and dendrites. These pulses are typically of order 10^{-3} s in duration and $5 \times 10^{-2} \text{ V}$ in magnitude. A neuron emits these pulses (this is called firing) in a roughly periodic manner, with the period being a function of the input signals experienced by the neuron at that moment. If the inputs are very active, that is if there are many pulses being received through the dendrites, the neuron will fire often. On the other hand, if its inputs are not active, the neuron will fire at a much lower rate. The picture is actually a bit more complicated than this. The interface where an axon and a dendrite meet (the synapse) may be either excitatory or inhibitory. In the former case a high firing rate of the sending neuron will favor a high firing rate of the receiving neuron. With an inhibitory synapse, a high input firing rate will tend to cause a low firing rate of the receiving neuron.

The firing rate of a particular neuron is a function of all of its inputs. To a very rough approximation, this rate is determined by the sum of all inputs that come into a neuron through excitatory synapses minus the total of the inputs that

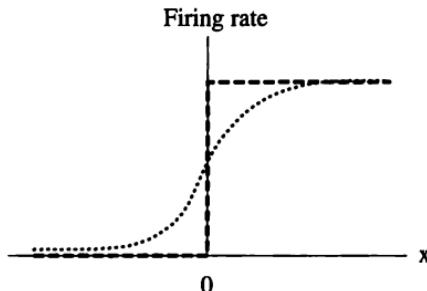


FIGURE 12.23: Firing rate function. The dotted curve is closer to that found in real neurons. The dashed curve is the step function used in our modeling.

enter through inhibitory synapses.²⁴ The firing rate, R , can thus be written as

$$R = f \left(\sum V_i \right) , \quad (12.13)$$

where V_i is the input signal from dendrite i (which may be either positive or negative). Experiments show that the firing function f is significantly nonlinear, as shown schematically in Figure 12.23. Because of this nonlinearity, a neuron is often modeled as a sort of on/off device.

A neural network is composed of a very large number of neurons whose basic properties we have just described. Such a network, that is, a brain, is perhaps the most remarkable object in nature, as it is responsible for the intelligence of living organisms.²⁵ The problem of understanding how a brain functions is of great concern to biologists, psychologists, computer scientists, and (believe it or not) physicists. It is useful to state this problem in the following way. The brain consists of a very large number of neurons that are highly interconnected. The connections are not limited to neighboring neurons, but can extend to neurons in remote areas of the brain. The state of a neuron can be described by its firing rate, which is a function of the firing rates of all of the neurons that have inputs to the neuron in question. The connectivity of the network and the strength of the connections vary from brain to brain. They vary from individual to individual (even within the same species, although there are many common features), and with time

²⁴The synapses also vary in strength, so some signals contribute more strongly to this sum than others. We will ignore this slight complication here, although it will be included in the model we construct below.

²⁵Some people (the authors included) believe that a brain is a neural network (essentially no more and no less), while others believe that a real brain contains some physics that is not contained in such a network. Arguments over this point have filled many books (but not this one). In any case, the majority of people on both sides of the argument would probably agree that neural networks are a major component of biological brains, and this is enough to make them worth studying.

for a given individual. On a very coarse scale, brains seem to possess a significant degree of randomness, yet we know that they are capable of very precise behavior, such as logical reasoning and memory. Understanding how such an arrangement of neurons can actually function as a brain is a key problem.

In this section we will consider how a neural network can function as a memory. Many aspects of biological memories²⁶ are not understood. For example, it is not yet known how information is stored (and forgotten) or how it is recalled. However, some ideas from physics have contributed greatly to recent progress in this area, and the answers to these questions may not be far away. Rather than try to give a logical or historical derivation (or justification) of the model of memory that we will study, it is simpler to just plunge ahead, and that is what we will now do. A little bit of the history of this model will be given at the end of this section.

We will model a neuron as a simple Ising spin. As we saw in Chapter 8, such a spin has two possible states, up and down. We will therefore assume that a neuron also has only two possible states, firing or not firing. This will, among other things, mean that we will approximate the firing function by a step function (the dashed curve in Figure 12.23), which has only two possible values. These two values will correspond to the two possible states of an Ising spin, $s = \pm 1$. The value of s_i corresponds to the current firing rate of neuron i . By convention we take $s_i = +1$ to correspond to a firing rate of 1, and $s_i = -1$ to 0 firing rate.²⁷ By associating the firing rate with the value of the spin we have glossed over the time dependence of the synaptic signals. At first sight this might seem to be a rather drastic approximation, but there are good reasons for believing that the timing of synaptic pulses does not play an important role in real neural networks. First, it is known from experiments that the firing times of different neurons are generally not correlated. That is, the brain does not appear to operate like a conventional computer in which all neuronal firing occurs in synchrony with master clock. Second, the interneuronal signals travel at a speed of ~ 1 m/s, so the propagation delay (which is different for each pair of connected neurons) makes a significant contribution to the pulse arrival times. For these reasons it is generally argued that the *timing* of the pulses does not play an essential role.²⁸ Rather, the average arrival rate seems to be an essential feature, and this can be modeled with simple Ising spin variables.

In the Ising model studied in Chapter 8, the effect of a spin on its neighbors was through the exchange energy, and for our neural network we can model the effect of one neuron on another in a similar way. We will assume that our spins (that is neurons; we will use the two terms interchangeably) experience an exchange interaction, but now we will permit an interaction between *every* pair of spins. This

²⁶As compared to computer memories.

²⁷We could have chosen the pseudospin values to be +1 and 0 so as to correspond directly with the firing rate. For the way we have chosen to write the energy this would contribute a constant term to E and would not affect the behavior of the model.

²⁸While this seems to be the current conventional wisdom, it has recently been suggested that correlated firing times are essential for understanding certain types of computations carried out by the brain. While the verdict has not yet been reached on this issue, it seems safe to conclude that much (if not all) of the brain's operation, including the memory functions we consider in this section, does not depend on precise timing of neuronal firing.

is necessary in order to mimic the highly interconnected nature of a real neural net, which is believed to be crucial for its operation. We will find it very useful to consider the effective energy of our neural network/spin system, which can be written as

$$E = - \sum_{i,j} J_{i,j} s_i s_j , \quad (12.14)$$

where the $J_{i,j}$ are related to the strengths of the synaptic connections, as we will describe shortly. The sum here is over all pairs of spins i and j in the network. The exchange, or more properly the synaptic, energies $J_{i,j}$ describe the influence of neuron i on the firing rate of neuron j . One way to view (12.14) is as follows. The sum of the synaptic inputs to neuron i will be $\sum_j J_{i,j} s_j$, and this will determine the firing rate, that is, the *direction* of spin i . If this input is negative, then according to our model it will be energetically preferred for neuron i to have a firing rate that is also negative, $s_i = -1$. If the synaptic input is positive, $s_i = +1$ will be favored. This is precisely analogous to an Ising spin model since as we saw in Chapter 8, each Ising spin prefers to point in the direction of the effective field established by the exchange energy due to its neighboring spins. Hence, our neural network can be described by the energy function (12.14), in close correspondence with the behavior of an Ising magnet. Individual neurons will prefer to fire at rates determined by the effective fields established by the interactions with other neurons, and this will make the network prefer states that minimize this energy.

As implied by Figure 12.22, the connections in a real neural network are *not* symmetric. The connection of the axon from neuron i to a dendrite of neuron j is completely separate from the connection between the axon of neuron j and a dendrite of neuron i . Indeed, there may be a connection in only one direction. The biological importance of this asymmetry in $J_{i,j}$ is not known. For now we will assume that the connections are symmetric, as this will allow us to make use of some ideas and results from statistical mechanics. The more realistic asymmetric case will be explored in the exercises.

The energy function (12.14) enables us to model our neural net as an Ising model, which can be simulated using the Monte Carlo method. However, we have not yet specified how the exchange energies should be chosen, or even how our lattice of spins can function as a memory. A useful memory must be able to store, recall, and display patterns, so let us consider how these operations can be implemented. The display operation is the easiest and is illustrated in Figure 12.24. On the left we show a lattice of spins with a particular spin configuration. This configuration was chosen so that it stores the letter A , which is seen more clearly on the right, where we show the same lattice, but with the spins for which $s = -1$ replaced by blanks. In this way the spins can be used as pixels to display whatever character or other type of pattern is desired.

In order for our memory to recall a pattern, we require that the spin directions change with time in such a way that the spin configuration eventually ends up in the desired state. For example, if we want to recall the letter A , we want the system to take on the configuration in Figure 12.24. The recall process thus involves first giving the spins a configuration, then allowing the spin arrangement to evolve with time until the system settles into a new, and we hope stable, configuration.

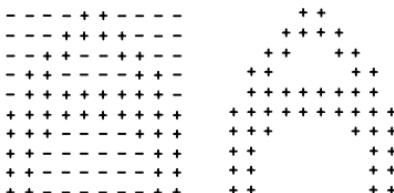


FIGURE 12.24: Left: a 10×10 lattice of spins with a particular configuration of + and - spins; right: the same lattice but with the spins for which $s = -1$ replaced by blanks. This network holds (i.e., displays) the letter *A*.

This time evolution is accomplished using an algorithm similar to the Monte Carlo method employed in Chapter 8. Starting from some particular configuration of the entire network, a spin is chosen and the energy required to flip it, ΔE_{flip} , is calculated using (12.14). If ΔE_{flip} is negative, that is, if flipping the spin would lower the energy, the spin is reversed. If $\Delta E_{\text{flip}} \geq 0$, the spin is left unchanged. This is precisely the Monte Carlo flipping rule employed in our work on the Ising model, with the assumption that the effective temperature is zero. We will have more to say about temperature and what it means in this problem a little later. For now, the important point is that the Monte Carlo rules ensure that the system will always evolve in time to states with the same or lower energy. If conditions are right, the memory will end up in a state in which the spins cease changing with time. This state corresponds to the pattern that is recalled by our memory.

The Monte Carlo rules, together with the energy function, determine how the spin system changes with time, and it is instructive to observe this directly in a simulation. Here, and with all of the other simulations below, we have used a 10×10 lattice of spins. For the moment we will not worry about how the interaction energies $J_{i,j}$ are determined. While their values are central to the behavior, it is simplest to postpone a discussion of how to calculate them until later. We are thus "given" a neural network, which in our specific case is a lattice of 100 spins interacting according to a certain collection of interactions $J_{i,j}$. We have, of course, chosen these interactions to yield some interesting behavior! On the left side of Figure 12.25 we show our lattice in a state that is close to the pattern for the letter *A* that we considered earlier. Even though a few of the spins are in the wrong state, that is, have been flipped with respect to Figure 12.24, it should be clear to your (human) brain that this pattern represents the letter *A*. We started a simulation with the lattice in this configuration, and after one pass through the lattice we obtained the state shown on the right side of Figure 12.25. Subsequent Monte Carlo sweeps through the lattice produced no further changes. The system thus found the ideal letter *A*, and recalled it. The memory worked as it should.

An important feature of human brains is that they are able to generalize in a reasonable manner. For example, you can usually recognize a letter even if it

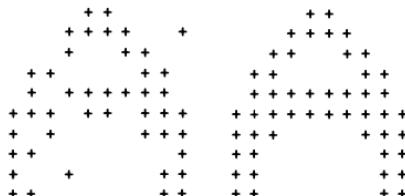


FIGURE 12.25: Left: spin arrangement close to the letter *A*, but with a few spins flipped to the wrong state; right: spin configuration after one Monte Carlo pass through the lattice

is shown to you in a different TYPE or STYLE. There is some property of *A*-ness that you are able to recognize, and all patterns that fall into this class will cause you to recall the same fundamental letter *A*. Our model exhibits similar behavior. The left side of Figure 12.26 shows the letter *A* in outline form. When we initialized our lattice of spins with this pattern, one Monte Carlo sweep through the lattice yielded the ideal letter *A* shown on the right. At least with this simple test of *A*-ness, the model responded correctly.²⁹

A similar test is shown in Figure 12.27, where the left side shows a damaged version of the ideal *A*. Here we have flipped 20% of the spins at random, and the resulting pattern only faintly resembled the ideal one. Nevertheless, after one Monte Carlo sweep through the lattice, the ideal pattern was recovered

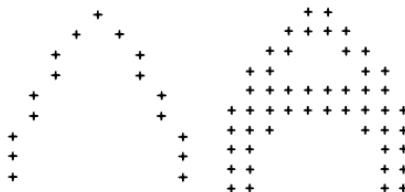


FIGURE 12.26: Left: spin arrangement that shows the letter *A*, but with a different pattern (that is, font) than used previously; right: spin configuration after one Monte Carlo pass through the lattice

At this point you might suspect that we are playing a joke, as our model memory seems to recognize *every* pattern as the letter *A*. To demonstrate that this is not the case, we show in Figures 12.28 and 12.29 the results obtained when the lattice was initially given a configuration that resembled the letters *B* and *C*,

²⁹We will leave the more challenging problems of a letter that is rotated or rescaled in size to the inquisitive reader



FIGURE 12.27: Left: spin arrangement that shows the letter *A*, but with 20% of the spins flipped at random; right: spin configuration after one Monte Carlo pass through the lattice

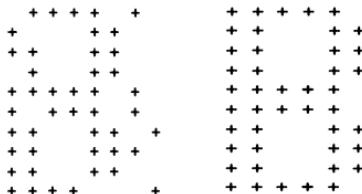


FIGURE 12.28: Left: spin arrangement that shows the letter *B*, but with 10% of the spins flipped at random; right: spin configuration after one Monte Carlo pass through the lattice

respectively. In each case the initial patterns were barely recognizable, but after one sweep through the lattice the ideal patterns were recovered. Our model memory was thus able to recognize several different letters.

Observing the model in operation raises several important points and a few questions. First, the model is seen to operate as a *content addressable* memory. This is in contrast to the type of memory we are familiar with in connection with a conventional computer. In the case of a computer memory, a piece of information, such as the value of a particular variable, is given a *name* or address. In order to recall the value of the variable, the address must be presented to the memory, which is then able to retrieve the desired value. However, with a content addressable memory, information is retrieved by giving a rough description of the information itself. For example, the value of π could be retrieved by giving only the first few digits. This is a very important property of human brains. As another example, we might want to recall a friend's face given only a vague recollection of the shape of her chin and her hair style. In addition, such a memory should be able to recognize her face even if she cuts her hair or dyes it green. Human brains are able to handle such tasks, which are difficult for the more conventional computer-style memories.

+ + + + + +	+ + + + + +
+ + + + + + +	+ + + + + + +
+ + +	+ + +
+ +	+ +
+ +	+ +
+ +	+ +
+ + +	+ + +
+ + + + + +	+ + + + + +
+ + + + + +	+ + + + + +

FIGURE 12.29: Left: spin arrangement that shows the letter C, but with 10% of the spins flipped at random, right: spin configuration after one Monte Carlo pass through the lattice

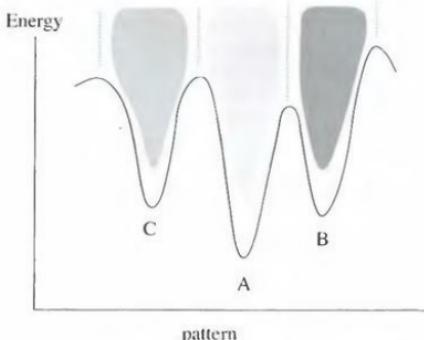


FIGURE 12.30: Schematic energy landscape. The vertical axis is energy as calculated from (12.14), while the horizontal axis corresponds to the spin configuration. The basins of attraction of several stored patterns are indicated by the shaded areas

Second, we have seen that our memory can generalize in a reasonable manner. When presented with a pattern that is similar but not identical to one that it “knows,” the memory is able to choose the correct response in a reasonable way. This behavior can be understood in terms of the energy landscape of the spin system. The energy of the network depends on the specific spin configuration, that is, memory pattern, at that time. Since the interactions $J_{i,j}$ (which we promise to explain shortly) have widely varying magnitudes and signs, this energy function will depend in a complicated way on the spin configuration. Schematically it will resemble the landscape shown in Figure 12.30. Each stored pattern, the letters A, B, and C in the example we have just considered, corresponds to minima in the energy. The Monte Carlo procedure guarantees that given an initial spin configuration, that is, some initial pattern, the system will evolve into a pattern with the same or, most likely, lower energy.

In general, the pattern presented to the network will not be one of the stored patterns, so the system will initially be situated on one of the slopes in the energy landscape. The Monte Carlo flipping rules then take it to the bottom of the nearest valley, and it thus ends up "recalling" the stored pattern associated with that minima. But what do we mean when we say that an initial pattern is close to a stored pattern? There are two ways to answer this question. In terms of the energy landscape there will be a *basin of attraction* in the neighborhood of each minima. If the network is within the basin corresponding to a particular minima, the associated stored pattern is the one that it will eventually locate. Topologically, these basins are the "valleys" surrounding each minima in Figure 12.30.

A quantitative answer to this question requires that we be a little more specific concerning the horizontal axis in Figure 12.30. A useful measure of the distance between two patterns is

$$\Delta_{m,n} = \frac{1}{N} \sum_i [s_i(m) - s_i(n)]^2 , \quad (12.15)$$

where m and n denote two different patterns, $s_i(m)$ is the configuration of spin i in pattern m , and N is the total number of spins. This is usually referred to as the *Hamming distance* between the two states of the system. Two patterns that are separated by a large Hamming distance will usually flow to different final stored patterns, while if the separation is small, there is a much better chance that they will flow to the same final state. For example, the letters *C* and *O* have many features in common, and will thus be closer in pattern space than are the letters *A* and *V*. Our memory will generally do a better job if the stored patterns are as different as possible.

While the Hamming distance is a quantitative measure of how similar two patterns are, it is difficult to be more precise in predicting how a particular network will respond to patterns that differ only slightly. This difficulty can again be traced to the energy landscape. Two patterns might be extremely similar, but if they happen to fall on opposite sides of an energy maxima they will flow to different final states.

The interaction energies $J_{i,j}$ are, as we have noted several times, key elements in our model memory. We are now ready to discuss how they should be chosen so as to yield a useful memory. We again let $s_i(m)$ denote the configuration of spin i in pattern m , and assume that we want our network to have this configuration as one of its stored patterns. That means that the $J_{i,j}$ must be chosen so as to make the energy a minima for this spin configuration. While there is no unique solution for this problem, a popular and very convenient choice is

$$J_{i,j} = s_i(m) s_j(m) , \quad (12.16)$$

which is often associated with the names of Hebb and Cooper, two important researchers in this area.³⁰ We can see that this choice for $J_{i,j}$ will make $s_i(m)$ an

³⁰Their work is described by Hertz, Krogh, and Palmer (1991)

energetically stable pattern as follows. Inserting (12.16) into (12.14) yields

$$E(m) = - \sum_{i,j} J_{i,j} s_i s_j = - \sum_{i,j} s_j(m) s_i(m) s_i s_j . \quad (12.17)$$

If the network is in pattern m , we will have $s_i = s_i(m)$ and $s_j = s_j(m)$, so each term in this sum will be unity, making the energy large and negative. On the other hand, a random pattern (we will be more precise about this in a moment) will, on average, have half of its spins flipped with respect to $s_i(m)$, so roughly half of the terms in (12.17) will be positive and half negative, leading to $E \sim 0$. Thus, our desired pattern will have a much lower energy than a random pattern. Furthermore, the energy of our stored pattern will correspond to a stable minima of the landscape, since flipping any one spin from the pattern value $s_i(m)$ will increase E .

The prescription (12.16) tells us how to store a single pattern, but a useful memory must be able to store many patterns. In that case we choose the $J_{i,j}$ according to

$$J_{i,j} = \frac{1}{M} \sum_m s_i(m) s_j(m) , \quad (12.18)$$

where the index m refers to the stored patterns, and there are a total of M such patterns (we will see in a moment that there is a limit on the total number of patterns that can be stored). The arguments we just gave can be used to show that the energy associated with each stored pattern will be much lower than the energies of a random pattern. In addition, if the stored patterns are sufficiently different from one another, they will each correspond to distinct, well-separated minima in the energy landscape. This is how the values of $J_{i,j}$ were chosen in the calculations described above. We used the letters A , B , and C as our stored patterns, and calculated the energies $J_{i,j}$ using (12.18).

We have now completed the description of our neural network memory model. The key features are:

- An Ising spin is used to model the on/off behavior of a neuron. The firing rate of a neuron is assumed to have only two possible states, corresponding to the spin values $s = \pm 1$.
- The connections between the spins $J_{i,j}$ are not limited to nearest neighbors, but link all pairs of spins in the network
- Given a collection of patterns that we want to store, the $J_{i,j}$ are calculated according to (12.18).
- The network operates as a content addressable memory. The lattice of neurons (that is, spins) is initialized with a configuration that resembles the pattern we want to recall. The Monte Carlo rules for $T = 0$ are then used, and the system evolves to a pattern that is at a minima in the energy landscape. This is the pattern that is “recalled” by the network.

The programming of this procedure is similar to the Monte Carlo routines described in Chapter 8, so we will leave the details to the exercises. However, we will give a few tips later in this section.

The modeling of neural networks is a vast industry, and we have been able to touch on only a very small piece of it here. A brief discussion of several other aspects of this field and some historical notes are given at the end of this section. However, before finishing we want to touch on a few issues associated with the choice of $J_{i,j}$ and the desired stored memories. Our network contains N spins, and since each pair is connected, there are $\sim N^2$ different values of the interaction energies $J_{i,j}$. We have described a procedure for storing a collection of patterns, and it should be clear that this information is stored in the $J_{i,j}$. This brings up several questions. (1) How many different patterns can be stored? (2) Can we add the concept of "learning" to the model, so as to bring it closer to the operation of a biological memory? (3) What happens if the $J_{i,j}$ are damaged in some way? We know that real memories can function even if some of the neurons die. Is our model able to function in the presence of such damage?

Each pattern involves the configuration of N spins and there are $\sim N^2$ different $J_{i,j}$, so the maximum amount of information that can be stored is of order $\sim N^2/N = N$ different patterns.³¹ However, it turns out that the maximum number of stored patterns is much less than this for several reasons. First, if two desired stored patterns happen to be close to each other (in terms of their Hamming distance), they can interfere with one another. This may make one of the patterns unstable, or cause it to be recalled in a distorted (imperfect) form. This problem can be minimized by choosing the stored patterns to be orthogonal to one another to the fullest extent possible. Here orthogonal means that $\sum_i s_i(m)s_i(n) \sim 0$ if $m \neq n$. In our simple example involving letters such orthogonality was not under our control, as it was determined by the shapes of the letters. While we can often preprocess the patterns to make them orthogonal before encoding them in the network, this may not always be convenient in cases such as, for example, the storage of arbitrary optical images.

The use of orthogonal stored patterns can increase the storage capacity somewhat, but it turns out that there is a more fundamental limit. As we use more and more patterns in the calculation of $J_{i,j}$, the nature of the energy landscape changes. Increasing the number of stored patterns means that more and more local minima must be crowded together, and this also affects the depth of each minima. It has been found that if the number of stored patterns exceeds $\sim 0.13N$, the landscape changes dramatically. In this case all of the stored patterns become unstable, and the system ceases to function as a memory. This abrupt change is actually a type of phase transition and has much in common with a system known as a *spin glass*. This is a type of spin system in which the interactions are randomly distributed and which have been studied extensively by physicists over the past 30 years.³² We do not have time here to say anything about spin glasses or how they are related to neural networks, other than to note that this is a very nice example of how

³¹This simple argument ignores the fact that the spins are binary objects, while the $J_{i,j}$ are real numbers, so this may be an underestimate of the capacity of the network.

³²In fact, spin glasses were studied long before it was appreciated that they had anything in common with neural networks. It turns out that spin glasses are typical of random systems with energy landscapes given by functions such as (12.14). The protein-folding problem falls into this class as well.

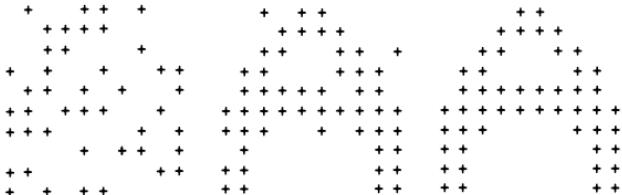


FIGURE 12.31: Operation with 80% of the $J_{i,j}$ set to zero. Left: initial pattern; middle: spin configuration after one Monte Carlo sweep through the lattice, right: configuration after two sweeps

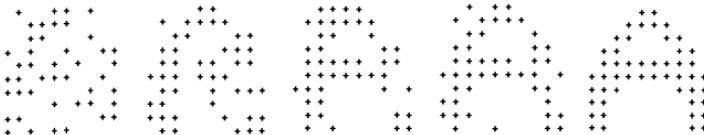


FIGURE 12.32: Operation with 90% of the $J_{i,j}$ set to zero. Left to right: initial pattern and spin configurations after one, two, five, and nine Monte Carlo sweeps through the lattice

ideas developed for one problem (spin glasses) can be profitably applied to a rather different area.

We next consider the effects of damage on the operation of our model memory. Since the information is stored in the interactions $J_{i,j}$, it is these connection values that will be damaged. We start with the $J_{i,j}$ calculated using (12.18) and the three stored patterns, A , B , and C . We then “damage” these values by randomly setting the elements of the $J_{i,j}$ matrix to zero with probability p_{damage} . Some results for $p_{\text{damage}} = 0.8$ are shown in Figure 12.31, where the initial pattern was an A , which itself has been altered from the stored pattern with probability 0.3. We recall that with the ideal $J_{i,j}$, our memory immediately found the correct stored pattern; that is, it took only one Monte Carlo pass through the lattice to obtain the stored pattern, which was then stable for all future times. With $p_{\text{damage}} = 0.8$ and after one Monte Carlo time step (one complete pass through the lattice), the system was closer to the stored pattern, but there were still some spins pointing in the wrong direction. However, after two time steps the system reached the correct, and perfect, final pattern.

The corresponding result with $p_{\text{damage}} = 0.9$ is shown in Figure 12.32. In this case it took nine Monte Carlo time steps to reach a stable pattern, but the system did eventually come to the correct final state. However, if the $J_{i,j}$ are damaged much further, the network fails, as illustrated in Figure 12.33. Here, with $p_{\text{damage}} = 0.95$, the system was unable to find the correct pattern. Instead, it ended up in a state unlike any of the stored patterns.

These simulations show that our model memory continues to function, even when severely damaged. It thus has an important feature in common with human

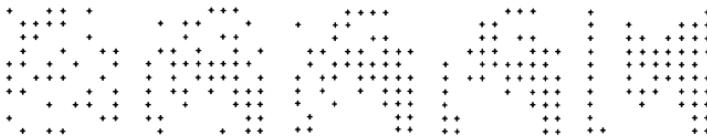


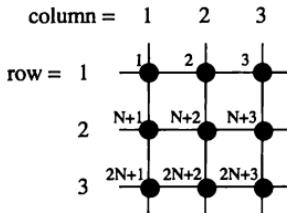
FIGURE 12.33: Operation with 95% of the $J_{i,j}$ set to zero. Left to right: initial pattern and spin configurations after three, four, five, and ten Monte Carlo sweeps through the lattice. In this case our memory did not recall the letter A .

brains. Indeed, it is amazing that the memory works so well, even with 90% of the connections removed. The level of damage that can be tolerated depends on the number of stored patterns. In this example, with only three stored patterns in a system of 100 neurons, we were well below the ideal theoretical limit of ~ 13 patterns. If the number of stored patterns had been closer to this limit, the level of damage that could have been tolerated would have been smaller. Nevertheless, neural network memories can continue to function well even when significantly damaged. Intuitively, this can be understood in terms of their redundancy. With a network that contains the theoretical limit of $\sim 0.13N$ patterns, $\sim 87\%$ of its information content is effectively redundant. A network's operation may not be adversely affected if some of this redundant information is lost.

One aspect of neural networks that we haven't really touched on is learning. The problem is extremely important for two reasons. First, if our goal is to model a real (biological) brain, learning must be accounted for. Second, neural networks of the type described here are being considered for a number of applications, such as image processing and handwriting recognition. These applications require that the network be able to incorporate new patterns, and in some cases forget old ones. The development of learning algorithms is an active area of research, and a number of alternative ways for calculating the $J_{i,j}$ have been proposed. One of the simplest learning procedures is based on (12.18). Consider a working network that contains several stored patterns. A new pattern can be learned by adding a small contribution to the interactions

$$J_{i,j}(\text{new}) = \beta J_{i,j}(\text{old}) + \alpha s_i(p)s_j(p), \quad (12.19)$$

where $s_i(p)$ is the new pattern, α is a parameter that controls how fast the learning should occur, and the value of β can be adjusted to allow for the fading of old memories. This learning algorithm has also been proposed on biological grounds. If a real network is presented with a pattern $s_i(p)$, it is believed that the synapses can adjust their strengths with time so as to make the pattern a stable configuration of the network. That is, the strengths and signs of a synapse adjust so that they are "consistent" with the states of the neurons that are involved. We will explore the behavior of this and several related learning procedures in the exercises.

FIGURE 12.34: Numbering scheme N is the number of spins in one row

PROGRAMMING NOTES

The simulation of a neural network memory involves the following steps.

- The desired stored patterns must be chosen. For the example calculations described above these were just letters laid out on a 10×10 grid, but you could use any other patterns including other characters or pictures.³³ You could also use a larger array of spins. This would allow more detail in the patterns and enable the system to store more patterns, but at the cost of additional memory requirements (memory in your program).
- A natural way to store the spin values is to use a two-dimensional array $\text{spin}(m, n)$ where the indices m and n specify the row and column where the spin is located. This storage scheme is convenient for displaying the stored pattern. However, we also have to store the interaction energies $J_{i,j}$. Here the indices i and j do *not* refer to rows and columns. You will recall that our model allows an interaction between each pair of spins. Hence, $J_{i,j}$ is the strength of the interaction between the i th and j th spins in the lattice, where i and j refer to a particular numbering scheme. One possible scheme is illustrated in Figure 12.34. The spins in the first row ($m=1$) are numbered $i = 1, 2, \dots, N$, those in the second row ($m=2$) have the numbers $i = N + 1, N + 2, \dots, 2N$, and so on up to the final spin, which is number $i = N^2$. Given the row and column numbers, m and n , the spin number is

$$i = N(m - 1) + n. \quad (12.20)$$

Note that this mapping can be inverted, that is, given i we can uniquely determine m and n , although we will not need to do this in our simulations.

The stored patterns can be described using the labeling scheme in terms of spin number i , with $s_i(p)$ denoting the value of the i -th spin for the p -th stored pattern. These in turn are used to calculate the interaction energies $J_{i,j}$ according to (12.18). Storage of the interaction energies requires a two dimensional array of size $N^2 \times N^2$. Thus, a 10×10 array of spins requires $\sim 10^4$ interaction energies³⁴

³³Black-and-white pictures would be simplest. We will leave gray scale or color images to the industrious reader.

³⁴An alternative scheme for storing the $J_{i,j}$ would be to use a four-dimensional array $J(m_1, n_1, m_2, n_2)$ to store the interaction energy for $\text{spin}(m_1, n_1)$ and $\text{spin}(m_2, n_2)$. This makes

- After the interaction energies have been calculated, the memory is ready for operation. The spins are given values corresponding to a particular pattern. In our example this was a character that was similar, though not identical, to a letter in the alphabet, but it could be a picture or other type of image. The Monte Carlo method is then used to calculate the spin directions at future times. The method is very similar to the one used to simulate the Ising model in Chapter 8. A spin is selected and the energy required to make it flip, E_{flip} is calculated from (12.18). If E_{flip} is negative, that is, if the energy would be reduced by flipping the spin, then the spin is reversed. If $E_{\text{flip}} \geq 0$, the spin is left unchanged. These flipping rules correspond to a system in thermal equilibrium with a heat bath at zero temperature. In this case the Monte Carlo procedure takes the system to the nearest energy minima, in the spirit of the energy landscape in Figure 12.30.
- The Monte Carlo procedure is used repeatedly, giving every spin a chance to flip. The spins can be chosen at random, or they can be selected by systematically moving through each row and column of the lattice. The two choices both work well. The most important thing is that *each* spin be given at least one chance to flip, so the systematic approach is often preferred. The Monte Carlo procedure is used until the state of the spin system becomes stable, indicating the pattern that is recalled by the memory. A network will usually find the desired pattern after only one or two Monte Carlo passes through the lattice, as the system will quickly locate the nearest energy minima. However, the behavior can sometimes be more complicated. If the number of stored patterns is close to the theoretical limit, or if two of the stored patterns are similar (have a small Hamming separation), the memory recall may not be ideal. The system may then recall a pattern that is different from any of the stored patterns (often an approximate mixture), or the network might never locate a time independent state. In the latter case it may switch back and forth in time between two or more patterns; such behavior is known as a limit cycle. In human terms we might think of this as corresponding to confusion!

Historical Notes

The field of neural networks is much too vast for us to be able to give a complete description in this section. A good historical discussion of the field is given in Hertz, Krogh, and Palmer (1991) which is listed in the references. In 1943 McCulloch and Pitts recognized that a network of simple neurons was capable of universal computation. This means that such a network could, in principle, perform any calculation that could be carried out with the most general computer imaginable.³⁵ This attracted a good deal of interest from researchers interested in modeling the brain.

³⁵the bookkeeping simpler than the scheme involving (12.20), but some computer languages do not permit four-dimensional arrays. In addition, the method (12.20) for encoding two numbers into one in an invertible manner can be useful in other situations

³⁶More precisely, such a network can calculate any computable function in the sense of a general purpose Turing machine

One of the first specific network models was proposed by Rosenblatt, who introduced a model known as the perceptron. This is a network composed of layers of neurons (i.e., spins), with neurons in each layer receiving connections only from neurons within the same layer and from the preceding layer. These are often referred to as *feedforward networks*, since information flows from one layer to the next and never interacts back with earlier layers. Many related network schemes were devised and studied, both from a physics perspective (in work by Cragg and Temperley, and Little, for example) and from the viewpoint of artificial intelligence. Hebb introduced the learning rule mentioned above, and it was proven that if a particular network was capable of computing a certain function, this learning rule would yield the appropriate interaction energies.

Unfortunately, some of the simplest perceptron networks are not capable of universal computation. This point was emphasized by Minsky and Papert in a very influential book that caused many workers, especially those interested in artificial intelligence, to abandon the field. Nevertheless, work on neural networks continued (albeit at a reduced level), and gradually it came back into fashion. In particular, the work of Hopfield stimulated enormous interest in the physics community. The network model we have studied in this section is often referred to as a Hopfield net and is distinguished by full connectivity (every spin is coupled to every other spin; it does not have a layered structure as in some models). In addition, the connections are often chosen to be symmetric ($J_{i,j} = J_{j,i}$) as this makes the model closely analogous to a spin model for which the theoretical machinery of statistical mechanics can be most readily applied.

Neural networks are now of very great interest to scientists from a number of different fields. They are studied as models of real (biological) brains and are used to gain insights to processes such as learning and memory. Neural networks are also attracting much interest in the computer science community, as they seem capable of massively parallel processing and could provide an efficient means for solving certain computationally very difficult problems, such as those connected with image processing. Artificial intelligence researchers are also studying neural networks, since they seem capable of generalization and association, two phenomena that have proved difficult to capture with conventional computational approaches.

In this section we have been able to give only a very brief description of one particular neural network model and have illustrated its behavior with a few simple examples. Our emphasis has been on the most important and unique features of neural networks and how these are related to the physics that we have encountered earlier in this book.

EXERCISES

- 12.11. Write programs to perform all of the calculations described in this section. Then, first encode a single pattern into memory in a 10×10 neural network. Start with no damage in bonds and see if there is any difference in recalling this pattern in memory starting from an arbitrary initial trial pattern. Try to quantify the relationship between the difference between the stored and initial patterns and the number of sweeps of the grid required to recall the former. Do the same with increasing damage and locate the maximum damage to still be

able to recall the original pattern. Study the relationship between the required sweep numbers and the amount of damage. Second, encode multiple patterns (say, starting from 2 and gradually increasing to 5). Study the accuracy of the neural network, i.e., its ability to recall the *right* pattern in the sense that the random initial pattern may tend to converge to a stored pattern that is *closest* to it. Try the same with varying amounts of damage.

- 12.12.** Investigate what happens when a network is overloaded with too many stored patterns. Do this by adding additional letters of the alphabet to the patterns stored by a 10×10 neural network. You should find that after storing about the first five letters in the alphabet, some of the patterns are no longer stable. However, the theoretical upper limit on the number of stable stored patterns is $\sim 0.13N^2$, which is 13 for a 10×10 network. Explain why you are not able to successfully store this many letters.
- 12.13.** In the previous problem we found that the maximum number of stable stored patterns may be less than the theoretical upper limit of $\sim 0.13N^2$ for an $N \times N$ network. Using a 10×10 network, try to devise 10 patterns that can all be stored simultaneously. Hint: It is best if the patterns are as different from each other as possible (see our discussion of the importance of orthogonality).
- 12.14.** The learning rule (12.18) yields values of $J_{i,j}$ that vary widely in magnitude. Investigate the effect of restricting $J_{i,j}$ to the values ± 1 . Do this by first calculating the Hebb/Cooper interaction strengths $J_{i,j}(\text{Hebb})$ using (12.18), then determine the final value using $J_{i,j} = \text{sign}[J_{i,j}(\text{Hebb})]$. Compare the performance of this network with the one we studied above. Of particular interest is the maximum number of stored patterns and the sensitivity to damage. You should find that this network works nearly as well as one in which the $J_{i,j}$ are continuous variables. This result is important for those who are interested in making neural network integrated circuits. It is much easier to design such circuits with only two different interaction strengths than to arrange for the interactions to vary continuously.
- 12.15.** A two-layer perceptron can be useful for classifying patterns. The structure of such a network is shown in Figure 12.35. The spins/neurons in the input layer are given a configuration corresponding to a pattern to be identified and locked rigidly in place. The Monte Carlo algorithm is then used to enable the spins in the output layer to reach the lowest energy state. The energy has the form (12.14), but the $J_{i,j}$ are now asymmetric; the direction of a spin i in the input layer can affect the direction of spin j in the output layer, but not vice versa. The interactions may be calculated with an expression similar to (12.18)

$$J_{i,j} = \frac{1}{M} \sum_m s_i(m) s_j(n), \quad (12.21)$$

where m refers to the input pattern and n now refers to the desired output pattern. Implement such a network and use it to classify patterns. For example, choose the $J_{i,j}$ so that several different patterns are recognized as the letter *A*, several others as the letter *B*, etc.

- ***12.16.** Throughout this section we have used the Monte Carlo procedure appropriate for a system at zero temperature. As discussed in connection with Figure 12.30, this algorithm takes the system to the nearest accessible energy minima. A drawback with this method is that a deeper minima will not be located unless it is directly "downhill" from the initial pattern. One way to avoid this problem is to

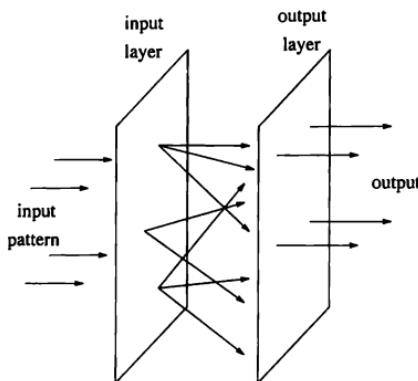


FIGURE 12.35: Schematic two-layer perceptron. Each layer contains a network of neurons, and the arrows between planes denote interactions. There are also interactions within each layer that are not shown.

use the Monte Carlo method with $T > 0$. In this case we accept spin flips with positive flipping energy with a probability $\exp(-E_{\text{flip}}/T)$, as discussed in Chapter 8. If the effective temperature is comparable to the energy barrier separating a metastable pattern from a more stable one, the Monte Carlo algorithm will enable the system to spend more time near the stable pattern. If the temperature is then slowly reduced, the network can sometimes locate the more stable pattern. Implement this procedure and study its performance. You should find that if T is too small, the network will be trapped in undesired states, as before. On the other hand, if T is too large, the system will quickly move far from the initial pattern and the final state will *not* resemble the initial one. This is not the way a memory should function. The best performance is obtained with an intermediate value of T , which is just large enough that the system can overcome the smallest energy barriers. This approach is similar to the annealing procedure we employed in our studies of protein folding.

- *12.17. Investigate the behavior of the learning algorithm (12.19). Use a 20×20 network and begin with the $J_{i,j}$ chosen according to (12.16) so as to store the letter *A*. Then add more patterns to the memory using (12.19) and the stored patterns of your choice. Study the behavior for different values of the parameters α and β .

12.4 REAL NEURONS AND ACTION POTENTIALS

In our work with neural networks we introduced some of the essential properties of neurons, and we mentioned that they communicate with each other via voltage pulses called action potentials. In this section we want to consider, in a fairly realistic way, how these action potentials are generated. An understanding of action potentials will give us insight into some of the basic physics of how neurons and the nervous system function at the molecular level.

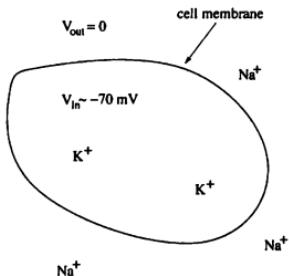


FIGURE 12.36: Schematic of a real neuron. A system of molecular pumps causes the concentration of K^+ ions inside the cell to be larger than the concentration outside. Likewise, the concentration of Na^+ is higher outside the cell. These concentration differences lead to a potential difference, which is typically 70 mV when the cell is "at rest."

Figure 12.36 shows a somewhat realistic picture of a real neuron. Here we show only the central portion of the neuron (the soma) and omit the dendrites and axons that carry signals to and from the soma (compare with Figure 12.22). We want to first understand and model how action potentials are generated in the soma. After that we will add axons and dendrites back into the model, and discuss how they are involved in signal propagation.

Cells are complicated systems (any biologist would tell us that). The inside of the cell is separated from the outside by the cell membrane. One might think that this membrane merely keeps the inside from spilling out, but it does much more than that. A number of different proteins reside inside and near the cell membrane, and these membrane proteins serve several functions. Some act as pumps that transport ions from one side to the other. For example, in typical neurons the concentration of K^+ inside the cell is much higher (by a factor of 10 or more) than the concentration outside, and this concentration difference is maintained by an ion pump protein. Likewise, an ion pump also transports Na^+ ions out of the cell, so that the concentration of Na^+ is about an order of magnitude lower inside the cell than outside. These concentration differences lead to a potential difference between the inside and outside.³⁶ When the neuron is "at rest," i.e., not firing an action potential, this potential difference is typically 70 mV, with the inside being negative with respect to the outside. This is called the resting potential.

In addition to ion pump proteins, the membrane of a neuron also contains proteins that wind their way through the membrane so as to form an open channel through which ions can pass, Figure 12.37. These channels are very narrow, typically only a few Å at their narrowest point, and they have the very interesting property that they are extremely selective. That is, some channels will pass only

³⁶As you might expect, other ions (such as Ca^{++} and Cl^-), in addition to K^+ and Na^+ , are also present and can play important roles. However, to get a basic understanding of action potentials we can get away with including only K^+ and Na^+ .

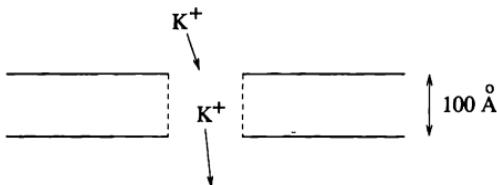


FIGURE 12.37: The cell membrane is a thin ($\sim 100\text{ \AA}$) lipid layer containing ion channels that allow selected ions to pass through

K^+ ions, while others will pass only Na^+ . In addition, these ion channels can open and close in response to the voltage across the membrane.

We have given only a rough, schematic description of a neuron, but even this description became available only about 50 years ago, with the work of Hodgkin and Huxley. While they built on the results of many others, their results and insights truly revolutionized our understanding of the nervous system. We will now discuss the nature of action potentials as developed by the experimental and theoretical work of Hodgkin and Huxley (HH).

Many neurons are relatively small, perhaps only tens of microns in diameter. This makes it hard, even with today's techniques, to probe neurons in great detail. HH understood this problem, and therefore focussed their experiments on neurons found in the squid. In particular, they studied the squid giant axon – here the term “giant” refers to the axon (not the squid). These axons are of order a millimeter in diameter, and HH were able to insert wires³⁷ and other probes inside these neurons and thereby perform a number of very detailed experiments.³⁸

Hodgkin and Huxley realized that the membrane of the squid giant axon is a complicated electrical system, and that current flow across the membrane involves different types of charge carriers. Through some very clever experiments³⁹ they were able to show that most of the current through the membrane is carried by K^+ and Na^+ ions. These currents depend on the voltage across the membrane, but this dependence is much more complex than a simple Ohm's law behavior. By inserting wires inside the axon, Figure 12.38, (and without completely disrupting the function of the cell), HH were able to probe how these two currents behave in detail. Figure 12.38 shows how the current of K^+ across the membrane depends on voltage. Here the system was initially at rest (i.e., at the resting potential), with no current flowing. The voltage was then increased abruptly by an amount V , and we

³⁷In the simplest experiment they inserted a wire along the axis of the axon (see the schematic on the left in Figure 12.38). This allowed them to measure the cross-membrane current very precisely, and it also meant that the entire axon was isopotential so that the action potentials did not propagate along the axon. This simplified an analysis of the time dependence of the current

³⁸It turns out that the properties of other types of neurons, including those found in mammals (us) are very similar to those of the giant squid axon, so the findings of HH apply quite widely

³⁹We do not have space here to give all of the background for, or a complete description of the HH experiments (an interested reader can find much more in the references). What follows should be viewed as an “abridged” version

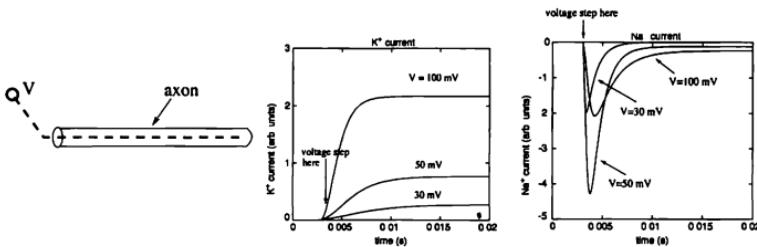


FIGURE 12.38: Left: Schematic of the experimental setup used by Hodgkin and Huxley to study the membrane conductance. A wire inserted along the axis of the axon was used to generate and probe axon potentials generated in a section of axon. Center: K^+ current as a function of time in response to a stepwise increase in the membrane voltage (by an amount V), for three different values of V . Here, by convention, a positive current corresponds to a current flowing out of the neuron. Right: Na^+ current as a function of time in response to a stepwise increase in the membrane voltage

see that the potassium current I_K , then increased with time. These results for I_K are schematic (we'll see below how to describe them mathematically), but HH were actually able to measure I_K in real experiments. There are two important aspects to these results. First, I_K takes several milliseconds to grow to its full value, and the time it takes to do this depends on V (it is shorter for larger V). Second, the magnitude of the I_K is not proportional to V ; the current at $V = 100$ mV is about 8 times larger than at $V = 30$ mV. Hence, (as we have already mentioned) this is not a simple Ohm's law type of system.

Hodgkin and Huxley devised the following model to explain the behavior of I_K . According to this model K^+ flows through ion channels in the membrane (Figure 12.37), and these channels can be in either an open or closed state. The state of a channel is controlled by "gating" particles whose states are described by the variable n . The value of n gives the probability that one of these particles will be in its open state. However, it turns out that there are *four* gating particles associated with each channel, and each particle must be in its open state in order for the channel to be open for K^+ ions. Hence the open probability for the ensemble of potassium channels, and hence the K^+ conductance is proportional to n^4 . To understand the behavior and function of the HH gating particles requires a microscopic picture of an ion channel. Such pictures are only now being developed, but the evidence suggests that the protein that makes up a channel can be in several different structural (i.e., conformational) states. When it is in its closed state, the channel is too small for an ion to pass through. The value of n then corresponds to the structural state of the protein, and there are multiple (in this case four) structural segments that must each be "open" in order for the entire channel to be conducting.

As we have seen from Figure 12.38, the conductance of the K^+ channels, i.e., the value of n , depends on the voltage across the membrane. Hodgkin and Huxley

discovered that this voltage dependence, and also the time dependent behavior seen in Figure 12.38 can be explained if n varies according to the rate equation

$$\frac{dn}{dt} = \alpha_n(1-n) - \beta_n n . \quad (12.22)$$

We can understand (12.22) by assuming, for a moment, that the gating particle state is either open ($n = 1$) or closed ($n = 0$). If it is open ($n = 1$) then there is a probability β_n that it will switch to the closed state, while if it is closed ($n = 0$) it will switch to the open state with probability α_n . In fact, n is a probability and varies continuously between 0 and 1, but this interpretation of (12.22) is still correct. Hence, if the ensemble of K^+ are all closed ($n = 0$) at $t = 0$, the value of n , and hence the open probability, will evolve in time according to (12.22). The voltage dependence comes from the voltage dependences of α_n and β_n , which are given by

$$\begin{aligned}\alpha_n &= \frac{0.01(10-V)}{e^{(10-V)/10}-1} \\ \beta_n &= 0.125e^{-V/20}.\end{aligned}\quad (12.23)$$

Here we have followed convention, with V being the membrane voltage relative to the resting potential, as measured in mV.

The rate equation for n together with (12.23) describe the probability that a potassium channel will be open. The total conductance of an ensemble of these channels is then

$$G_K = \bar{G}_K n^4 , \quad (12.24)$$

where \bar{G}_K is the conductance one would have if all of the channels are open, and n^4 is the open probability, allowing for the fact that four gating particles must be in the open state in order for a channel to be open. In most experiments one actually measures the current carried by a particular ion. For the case here the potassium current is

$$I_K = \bar{G}_K n^4 (V - V_K) , \quad (12.25)$$

which is similar to Ohm's law with the conductance in (12.24). However, here the current does not vanish when $V = 0$; instead there is an offset voltage V_K . This arises due to the different concentrations of K^+ inside and outside the cell. Since this concentration is greater inside the cell, this offset voltage is slightly negative for K^+ .

We now have a complete mathematical description of the potassium current, and how it varies with time. We will leave it as an exercise to show that the rate equation for n (12.22) together with (12.23) can explain the time and voltage dependence of the potassium current seen in Figure 12.38.

Our next job is to consider the sodium current. The picture is similar to that for potassium, although now there are *two* gating variables, called m and h . The idea is that the m gating particle acts just like n , as $m = 1$ corresponds to a Na gating particle in the open state, and $m = 0$ corresponding to a closed state. Three m particles are associated with each channel, so the open probability is proportional

to m^3 . The second sodium gating variable h is needed to explain the variation of the Na^+ current seen in Figure 12.38. There we saw that when the voltage is increased, the sodium current, I_{Na} , first increases in magnitude, but then *decreases* back towards zero after a few milliseconds. Hence, the Na^+ channels first open and then close as time passes. This closing is controlled by the h gating particle. This picture accounts for the behavior in Figure 12.38 as follows. Initially (at $t = 0$) the channels are closed with $m = 0$ and $h = 1$. Increasing V then causes m to grow with time, opening the channels. However, the increase in V also causes h to decrease; this decrease happens a little more slowly than the increase in m , so the channel closes shortly after it opens.

This time dependence of m and h is described by rate equations that are similar to (12.22). They have the form

$$\begin{aligned}\frac{dm}{dt} &= \alpha_m(1-m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h(1-h) - \beta_h h.\end{aligned}\quad (12.26)$$

The corresponding α and β functions are given by

$$\begin{aligned}\alpha_m &= \frac{0.1(25-V)}{e^{(25-V)/10}-1} \\ \beta_m &= 4e^{-V/18} \\ \alpha_h &= 0.07e^{-V/20} \\ \beta_h &= \frac{1}{e^{(30-V)/10}+1}.\end{aligned}\quad (12.27)$$

To complete the picture, we need to relate the sodium current to m and h . The relationship for the conductance is

$$G_{\text{Na}} = \bar{G}_{\text{Na}}m^3h,\quad (12.28)$$

where \bar{G}_{Na} is the total open conductance of all of the Na^+ channels. For the current we have

$$I_{\text{Na}} = \bar{G}_{\text{Na}}m^3h(V - V_{\text{Na}}),\quad (12.29)$$

where V_{Na} is the offset voltage for Na^+ . Since the concentration of sodium is larger outside the cell, V_{Na} is positive.

We are now ready to consider how the cell voltage varies in response to an excitation. There are several ways that a neuron can be excited. Here we will consider what happens when current is simply injected into the cell using an external probe, as sketched in Figure 12.39 (we'll explore excitation via a synaptic current in the exercises). The external probe is typically a glass pipette that has a sharp tip that is stuck into the cell. The pipette contains a water solution with K^+ and Na^+ ions, so it conducts current readily and does not disturb the chemical makeup of the cell. While the cell membrane is a complicated electrical conductor, we know that it must obey current conservation. We can therefore write

$$I_{\text{inj}} = C_m \frac{dV}{dt} + I_K + I_{\text{Na}} + I_{\text{leak}},\quad (12.30)$$

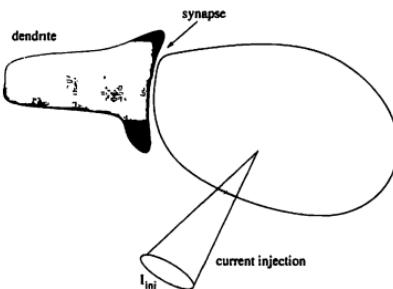


FIGURE 12.39: A neuron can be excited by injecting current via an external probe, or by a synaptic current

where C_m is the membrane capacitance, and the current I_{leak} is due to a voltage independent “leakage” conductance through the membrane. This leak current has the form

$$I_{\text{leak}} = G_{\text{leak}}(V - V_{\text{leak}}) , \quad (12.31)$$

where G_{leak} and the offset voltage V_{leak} are measured quantities.

All of the currents in (12.30) are known, so we can use it to solve for the variation of the membrane current as a function of time. We begin by solving (12.30) for the time derivative of V , and then write this derivative in finite difference form

$$\frac{dV}{dt} = \frac{V(j+1) - V(j)}{\Delta t} = \frac{1}{C_m} (I_{\text{inj}} - I_K - I_{Na} - I_{\text{leak}}) , \quad (12.32)$$

where $V(j)$ is the membrane voltage at time step j . We can then rearrange (12.32) to express the voltage at the next time step ($j+1$) in terms of the voltage and currents at the previous step. We must also solve for the variations of the gating variables n , m , and h with time. This can be done in the same manner, by writing (12.22) and (12.26) in finite difference form, and solving for $n(j+1)$ in terms of $n(j)$, etc. for $m(j+1)$ and $h(j+1)$. This calculation thus yields the membrane voltage, the gating variables, and the ion currents (12.25) and (12.29), all as functions of time.

Some results are shown in Figure 12.40. For this and the remaining calculations in this section, we have used the standard literature values for the various conductances for the squid giant axon system; these are listed in Table 12.1. The conductances, etc., in this Table give values per unit membrane area. We will assume a cell of radius $r_m = 10 \mu\text{m}$, so the membrane area is $4\pi r_m^2/3$. For our first calculation, we have used a relatively small value of the injection current. On the left in Figure 12.40 we see that this excitation leads to potassium and sodium currents that are the same order of magnitude as the injected current. Note also that the Na^+ current is negative, since the offset voltage (V_{Na}) is large and posi-

TABLE 12.1: HH parameters and rate functions for a squid giant axon at (at 6.3°)

C_m^*	membrane capacitance per unit area	$1 \mu\text{F}/\text{cm}^2$
G_K^*	K^+ channel conductance per unit area	$0.036 \text{ S}/\text{cm}^2$
V_K	K^+ offset potential	-12 mV
G_{Na}^*	Na^+ channel conductance per unit area	$0.12 \text{ S}/\text{cm}^2$
V_{Na}	Na^+ offset potential	115 mV
G_L^*	leak conductance	$0.3 \text{ mS}/\text{cm}^2$
V_{leak}	leak offset potential	10.6 mV

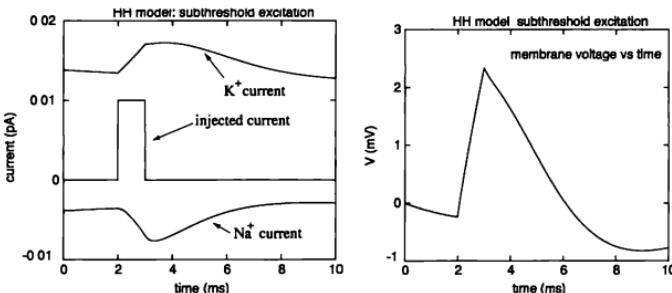


FIGURE 12.40: Behavior of an HH neuron for small excitations. Left: variation of I_K and I_{Na} as functions of time for a subthreshold injection current I_{inj} is also shown, it was just a constant current pulse that was turned on at $t = 2$ ms with a duration of 1 ms. Right: variation of the membrane voltage

tive. The corresponding membrane voltage is fairly small, of order a millivolt; this excitation did not manage to produce an action potential.

Results for a larger excitation are shown in Figure 12.41. The injected current here is only a factor of 3 larger than in Figure 12.40, but the K^+ and Na^+ currents are now several orders of magnitude larger than I_{inj} . Thus, a relatively small excitation was able to produce a large response. The membrane voltage exhibits a large pulse, with a magnitude of approximately 100 mV, and a duration of a few milliseconds. This is an action potential.

A striking feature of an action potential is that the response of the system is very large compared to the size of the excitation. When the excitation is below a certain threshold level (in this example, that threshold is near $I_{\text{inj}} = 0.02 \text{ pA}$), the response is small and proportional to the excitation. However, when this threshold is exceeded there is essentially an “avalanche” effect. Specifically, such an above-threshold excitation increases the membrane voltage, which then causes the sodium current to increase rapidly, which then causes the voltage to increase even more, etc.

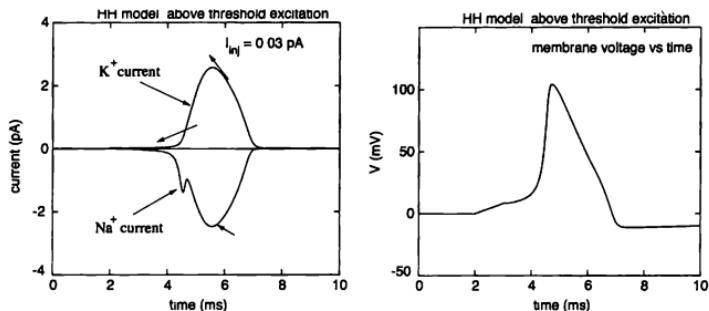


FIGURE 12.41: Left variation of I_K and I_{Na} as functions of time for an above threshold injection current, with $I_{inj} = 0.03$ pA. The injected current was again a constant current pulse that was turned on at $t = 2$ ms, with a duration of 1 ms. Right variation of the membrane voltage, showing an action potential

This avalanche does not grow forever, as the sodium current eventually saturates and then is turned off, and the potassium current turns on to oppose and partially cancel the sodium current. The result of this nonlinear process is an action potential pulse whose shape and size is nearly independent of the excitation conditions. This roughly “quantized” size is reminiscent of a digital system, with pulses of unit size.

We have only considered the generation of a localized action potential in a single, isolated neuron. It also turns out that action potentials can propagate along the axons that connect one neuron to another. These voltage pulses carry signals throughout the nervous system, and their relatively large size (much larger than the background noise) enables these signals to be sent and received reliably. We will leave the details of action potential propagation and several related questions to the exercises.

EXERCISES

- 12.18. Use (12.23) and (12.27) to calculate how the open probabilities for potassium and sodium channels vary with time following a voltage step. These probabilities are closely related (but are not quite equal) to the K^+ and Na^+ currents shown in Figure 12.38.
- 12.19. Calculate how the gating variable m for sodium channels varies in response to a step change in voltage.
- *12.20. An axon is essentially a neuron that is very long compared to its diameter. Consider such an axon, and assume that it has the same membrane conductances as the neuron in Figure 12.41. To model such a long system, we cannot assume that the voltage is the same everywhere inside. Instead, we must let V vary with position along the axon. Do a simulation for a long axon, and show that if an action potential is generated at one end (via current injection), it will propagate as a pulse to the other end of the axon. Hint: Take the length of the axon to be 1 cm, and treat it as a series of “compartments” (that is the conventional term

in the literature), and let the voltage vary from one compartment to the next. In addition to the parameters in Table 12.1, you will need to use the value of the conductivity of the fluid inside the cell. This conductivity is 0.02 S/cm (the siemen=ohm⁻¹ is the unit of conductance)

- 12.21.** In Figure 12.41 we considered a cell that was excited by current injection. However, during normal operation neurons are usually excited by a synaptic current. A synapse is a region where an axon from one neuron comes into very close proximity with another neuron, as sketched in Figure 12.39. An action potential signal travels along the axon, and when it reaches the end, i.e., the synapse, certain biochemical transmitter molecules travel to the receiving neuron and cause ion channels to briefly open. These are non-selective channels, that can be modeled as a simple conductance

$$g_{\text{syn}} = g_0 e^{-t/t_0}, \quad (12.33)$$

and the associated current flow is

$$I_{\text{syn}} = g_{\text{syn}}(V - V_{\text{leak}}). \quad (12.34)$$

Calculate the action potentials produced by this excitation, and find the threshold value of g_0 , i.e., the value of g_0 that is just large enough to produce an action potential.

- 12.22.** When a constant current is injected into a neuron (i.e., not a current pulse as in Figure 12.41, but a constant, time independent current), a neuron can generate a regular sequence of action potentials. Investigate the response of the neuron studied in this section to a constant excitation, and calculate how the firing frequency depends on the magnitude of the excitation current.

12.5 CELLULAR AUTOMATA

In this section we introduce a technique called *cellular automata*. We have discussed many different approaches to the modeling and simulation of various real-life phenomena in this book, including iterative maps, Monte Carlo simulations, molecular dynamics, variational optimization, neural networks and so on. Cellular automata is a general type of approach to dynamic modeling that shares common ideas with many of these other techniques. In the cellular automata approach the time evolution of a system is determined by a rule (usually a deterministic one) involving the current (and perhaps past) local environment at each point in space, called cells. Thus, *cellular* refers to the discrete space and local nature, while *automata* refers to the rule-based automatic time evolution. The local and cellular part is thus similar to most Monte Carlo simulations (e.g., of the Ising model), while the rule-based, deterministic time evolution is similar to molecular dynamics simulations (though the molecular dynamics rules are neither local nor simple in most simulations). These properties make cellular automata ideally suited for computer calculations and it is a great deal simpler than most other methods we have described in this book.⁴⁰

Let us use the most famous example of cellular automata called the “game of life” to introduce the general idea. This game was introduced by Conway in

⁴⁰For this reason, cellular automata is often spoken of as a poor man’s molecular dynamics

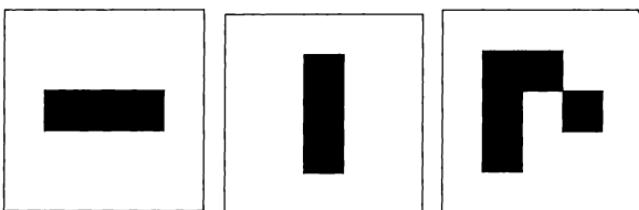


FIGURE 12.42: In the cellular automata *life* the pattern on the left (which contains three living horizontally connected neighboring cells) turns into the pattern in the middle (three vertically connected cells) after one time step. Likewise, the middle pattern turns into the one on the left after one time step. Patterns of this kind are called *blinkers*. The pattern on the right is a *glider* which moves endlessly toward the upper left. There are other gliders which move in different directions as well.

1970,⁴¹ and was inspired by the ideas of Ulam and von Neumann regarding the concept of a universal computer. Conway's cellular automata is a two-dimensional system where the cells are arranged as in the square lattice, with each cell having 8 neighbors (left, right, up, down, and the four diagonal ones). These cells represent organisms which are either alive or dead. At a given time, the 8 neighbors of each cell are examined, and the cell's life status at the next time step is determined by how many of the neighbors are still alive. After this determination is made for all of the cells, they are simultaneously updated to reflect the new status. The precise rule is:

- If exactly two neighbors of a cell are alive, then the life status of that cell is unchanged. (Dead cells thus remain dead and live cells remain alive.)
- If exactly three neighbors are alive, then the cell will be alive at the next time step regardless of its current life status.
- Otherwise, the cell will be dead at the next time step.

The idea of *life* is somewhat analogous to the logistic map studied in Chapter 3; while a certain number of neighbors are needed to keep a cell alive or to give birth, too many living neighbors will compete for resources and end up killing a cell. So while there is a mechanism for growth, there is also a check to prevent too much growth. What sparked interest in this model is that these simple rules can have rather complicated results.⁴² For example, the initial pattern of live cells shown on the left in Figure 12.42 on the left “blinks” into the one in the middle at the next instant and then back again after that. This is an example of a *two-cycle*. On the other hand, the pattern on the right “glides” diagonally towards the upper left.⁴³ Combinations of these and other patterns can, even in a modestly sized lattice, exhibit complicated time evolution that includes chaotic behavior and even

⁴¹Through a Scientific American article by Gardner.

⁴²From our experience with the logistic map, we shouldn't be too surprised about this. What is not obvious is that *life* is capable of performing any finite logic or computation (see references).

⁴³When a glider runs into another live cell they interact, often starting an interesting evolution

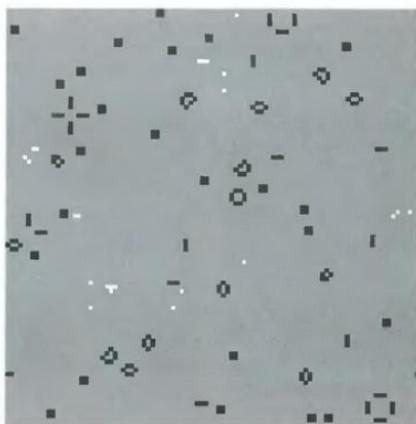


FIGURE 12.43: A 100×100 periodic lattice where an initial pattern was created by randomly making each cell alive by a probability of 0.3. The grey shaded cells were once alive but currently dead. White cells have never been alive and are now also dead. After many sweeps of the lattice according to the rules of Conway's game of life, a typical final pattern emerges with many unchanging shapes (limit points) and blinkers (limit cycles). While such a steady state is most probable, it is known that some initial patterns will evolve into a much more complicated series of patterns, such as chaotic ones and those where the number of live cells continues to grow.

unlimited growth. There are indications that some initially structureless patterns could self-organize into complex patterns with structures on all length scales as well (see the reference by Bak, Chen, and Creutz).

Cellular automata have been used to model a vast array of different phenomena, including urban and emergency evacuation planning, sociology, robotics, and the evolution of languages, as well as more familiar topics in biology, chemistry, and physics. All this is possible because of the huge variety of rules that can be defined, which allows one to devise rules that mimic the basic ingredients of virtually any problem. For example, rules that contain stochastic elements can be used to simulate stochastic systems (similar to Monte Carlo methods), or one can build in conservation laws in order to deal with mechanical or thermodynamic systems. Its simplicity in concept and in implementation made this approach an extremely powerful tool in a wide range of fields.

Let us apply the cellular automata approach to model the dynamics of a sandpile.⁴⁴ If we drop one grain of sand at a time onto an empty table at a random location, piles of sand will grow. Eventually the slope of the pile at some point will be large enough that an avalanche will occur. The build-up then restarts until the

⁴⁴Sandpiles have played an important role in the development of the idea of self-organized criticality, and thus are familiar to many statistical physicists

next avalanche strikes, usually somewhere else. The longer the build-up duration, the larger the scale of the avalanche tends to be. It is well known that avalanches come at all scales and it is impossible to predict the size of the next one. In fact, the frequency of the avalanches and their distribution of magnitudes turn out to involve power laws, similar to those seen near a thermodynamic critical point (and similar to what we found for earthquakes earlier in this chapter). Sandpiles are thus considered to be an example of *self-organized criticality*, a topic also mentioned in connection with earthquakes. In self-organized criticality, a critical state with all length- and time-scales represented develops automatically without requiring any external tuning of parameters.

Here we follow the pioneering work of Bak, Tang, and Weisenfeld (1987, 1988) and treat this problem with a cellular automata on a square lattice. The state of the sandpile in each cell i is given by an integer variable z_i , and if the value of this variable exceeds a threshold z_c , the site i will tumble. When this happens, z_i is reduced by 4 and for each of its 4 nearest neighbors j , the value of z_j is incremented by 1. The variable z_i can be loosely interpreted as the magnitude of the sandpile's local slope at the i -th cell, but this is not an exact correspondence since the true slope, i.e., the gradient, would have a direction whereas z_i is a scalar variable. This variable also has some features of the sand pile's local height, as we represent the addition of a sand grain at cell i by incrementing z_i by 1, and describe a free boundary condition by letting sand spill over it. However, the correspondence is not complete since in a real sandpile it is the slope and not the height that determines if the pile tumbles. While the model gives at best a qualitative description of a real sandpile, we will see that it does capture many features of a sandpile's instability and self-organizing behavior. The dynamics of the system is driven by adding a sand grain one at a time, waiting for the resulting avalanche to settle and die out before adding the next one. Thus each addition of a grain is followed by relaxation that affects varying parts of the system and lasts for varying durations.

Let us now consider the specifics of our cellular automata sandpile model. Since any constant can be added to all of the z_i 's without changing the probability of an avalanche, the actual value of z_c is irrelevant; we therefore take $z_c = 3$ and thus the stable values of z_i are 0, 1, 2, and 3. We must also decide what to do when an avalanche occurs at a boundary cell. If we have a free boundary, then some of the tumbled sand will fall off the system, while if we have a periodic boundary the tumbled sand will reappear at the opposite end. In the latter case, all added grains would stay in the system and all cells would eventually attain values of $z > z_c$, an extremely unstable situation with avalanches everywhere at all times! We expect that a real system should be able to reach a steady state, so there needs to be a mechanism for the system to lose grains. We will thus employ free boundaries on all sides. We then need to decide on the initial conditions. As you might expect from the earlier reference to self-organized behavior, it turns out that the initial state does not make too much difference after a great many grains are added. Both a randomly generated initially stable (i.e., no $z > z_c$) state and one that is minimally stable (i.e., all $z = z_c$) (see Figure 12.44) evolve into states that are virtually



FIGURE 12.44: Two possible initial states of a 100 by 100 sandpile. Left: A minimally stable state with all $z \equiv 3$. Right: A randomly generated state with about half of the cells having $z = 2$ and the other half $z = 3$. Cells with $z = 3$ are shown in black while those with $z = 2$ in gray.

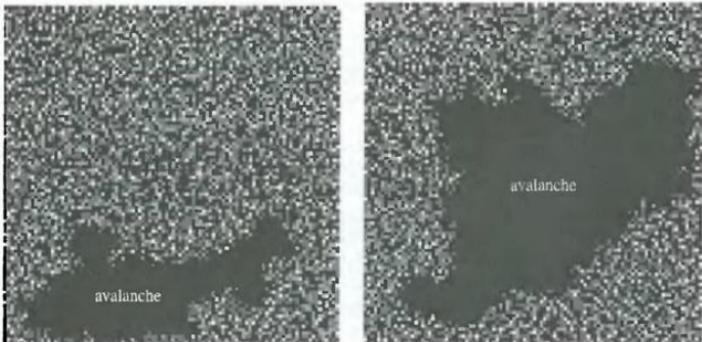


FIGURE 12.45: Left: A state that evolved from the minimally stable (all black) initial state of Figure 12.44 after about 7000 grains have been added at random locations. Right: A state that evolved after a few thousand grains were added to a randomly created state such as the one shown on the right in Figure 12.44. Cells with $z = 3$ are shown in black, those with $z = 2$ are in dark gray, those with $z = 1$ are in light gray, and those with $z = 0$ are in white. The dark inner regions labeled "avalanche" are the regions where an avalanche occurred when the last grain of sand was added.

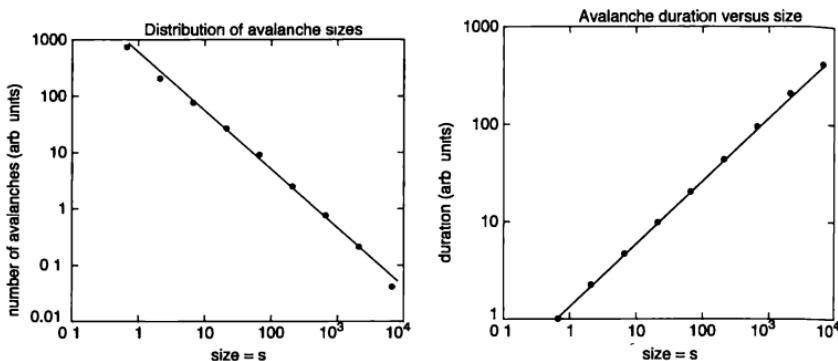


FIGURE 12.46: Left Frequency of avalanches as a function of their size based on just one simulation starting from an initial state with $z = 2$ and the other half $z = 3$, on a 100 by 100 lattice. The line is a least-squares fit to (12.35), which gave a slope of $\tau = 1.03$. Right Dependence of the average duration of the avalanches on their size. The line is a least-squares fit to (12.36) which gave $a = 0.65$.

indistinguishable (though over different time scales).⁴⁵ Shown in Figure 12.45 are states reached from the initial configurations shown in Figure 12.44 after several thousand grains of sand have been added. Also indicated in these pictures is the spatial extent of the latest avalanche, that is, the cells which tumbled at least once during the last avalanche.⁴⁶ Some of the cells in this group actually tumbled many times during this one avalanche.

Bak, Tang, and Weisenfeld showed that the avalanche frequency depends on their size by a power-law,

$$N(s) \sim s^{-\tau}, \quad (12.35)$$

where $N(s)$ is the number of avalanches of size s (per unit size range), s is the number of sites that tumble at least once in a particular avalanche event, and the exponent has a value $\tau \approx 1.0$. We followed their work and recorded the time evolution of an initially random configuration of 100 × 100 cells through the addition of 10,000 grains of sand and show the results in Figure 12.46. With just this single simulation, we confirm the power-law with τ very close to their value. In their

⁴⁵It is possible to be more precise about the comparison of the final states from the two vastly different initial states, but we will leave that to the reader. Also, you may wish to study the times required for the minimally stable state to evolve into a stable state, both in terms of the number of grains added and of the aggregate time counting the relaxation times after each addition of a grain.

⁴⁶These cells can be considered to form a cluster, but unlike those clusters in the percolation problem at criticality studied in Chapter 7, they are evidently not fractals, though they have irregular edges.

calculations, Bak, Tang, and Weisenfeld (1987) averaged over the addition of a single grain with many independent cellular automata arrays, so this seems to be a rather robust result.

On the right in Figure 12.46, we show the dependence of the average duration of an avalanche as a function of its size from the same simulation. Evidently, we also have a power law here,

$$T(s) \sim s^a, \quad (12.36)$$

with $a \approx 0.65$. Furthermore, the frequency of occurrence of avalanches as a function of their duration can also be shown to be a power law,

$$N(t) \sim t^{-b}, \quad (12.37)$$

where $N(t)$ is the number of avalanches of duration t normalized to unit t range, and $b \approx 1.1$. These are consistent with the findings of Bak and coworkers, and testify to the robust (i.e., universal) nature of the sandpile avalanches.⁴⁷

The relaxation dynamics itself is also of interest. Figure 12.47 shows snapshots during the relaxation of an avalanche started by placing a new sand grain near the middle of a randomly generated initial state. The tumbling of cells proceeds in a ring-like manner. The currently supercritical cells and those that tumbled just previously form a double ring; these double rings occur nearly concentrically and move outward toward the edges initially (see the system on the left in Figure 12.47), then, after reaching the edges, the rings turn around and move inward (the image on the right in Figure 12.47). Depending on the state just before adding the grain (and on where it is added), these pictures can become considerably more complicated. However, one thing that is obvious is that the relaxation process has an oscillatory component. Even for the same size avalanches, the relaxation time varies a great deal, depending on how the oscillatory component evolves; for some avalanches, certain cells tumble, get built up, and tumble again many times, while for others, each affected cell tumbles only once.

EXERCISES

12.23. Change the rules for the game of *life* automaton and see if the time evolution retains a similar character to *life*. For example, consider the rules

- If exactly n neighbors are alive, then the life status is unchanged. (Dead cells remain dead and live cells remain alive)
- If exactly $n + 1$ neighbors are alive, then the cell will be alive at the next time step regardless of its current life status.
- Otherwise, the cell will be dead at the next time step.

The game of life corresponds to $n = 2$, but you may change it to other values such as 1, 3, 4, etc [This problem was discussed by Heudin (1994)].

⁴⁷If we simply plug (12.36) into (12.35), we evidently do not obtain (12.37). Think about why they are still consistent with each other. The exponent a corresponds to $(1 + \gamma)^{-1}$ in Bak et al.'s notation.

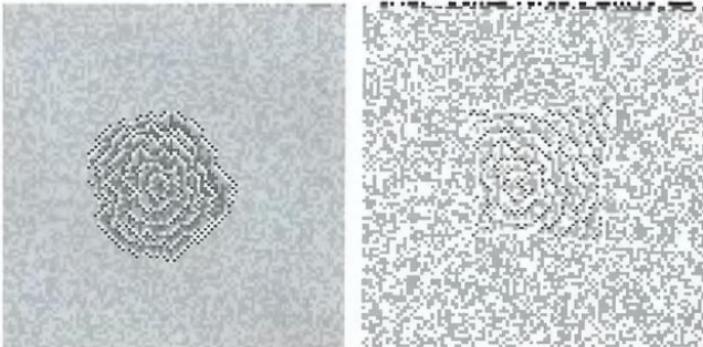


FIGURE 12.47: Snapshots during the relaxation of a sandpile started by placing a new sand grain near the middle of a randomly generated initial state. Left: An outgoing avalanche front is shown, where the dark filled circles denote currently supercritical ($z > z_c$) cells and white cells are those that just tumbled at the previous time step. The double rings of dark and white cells move toward the edges in an expanding avalanche. Within this avalanche, the cells are shaded so that the darker the shade, the higher the value of z of the cell. The generally lighter shaded cells outside of the expanding avalanche indicate that they have not yet tumbled. Right: A receding avalanche front is shown, where the double rings of currently supercritical sites (filled black circles) and those that tumbled just previously (white) now move toward the center. The diagonal cells with lighter shades form due to symmetry imposed by the square lattice. Darker shades indicate larger values of z ; the cells outside the avalanche front are generally darker than in an expanding avalanche, indicating that all of them have tumbled at least once already.

- 12.24.** Investigate the effects of boundary conditions on the game of life. For example, compare the results with free boundaries on all sides to those with periodic boundary conditions.
- 12.25.** Write a cellular automata program to simulate a sandpile and study various statistics such as:
- The overall distribution (i.e., frequency) of the avalanche durations.
 - The distribution of the durations for a given avalanche size range.
 - The relationship between the mean avalanche size that occurs during a time series and the mean avalanche size that a given cell belongs to during the same time series.
- 12.26.** In the sandpile automata, introduce some damage into the lattice. For example, select certain cells to always have $z = 0$ (thus sand can be considered to be always lost there). Or, let certain cells be impenetrable pillars (i.e., a tumbling cell next to it cannot unload a grain on it). How do the results (e.g., the distribution of the avalanche sizes) change?
- 12.27.** Set up a cellular automata of sandpile in three dimensions and perform the simulations following the text. Do you still observe self-organized critical behavior resulting in power-laws? What are the exponent values?
- 12.28.** Alter the dynamics of the sandpile cellular automata by adding new grains immediately, without waiting for an avalanche to relax to a stable state. Observe the differences between this model and the one discussed in the text.

The topics discussed in this chapter are close to the frontiers of current research, which can make it difficult to find introductory references. The history of each topic can be traced back many years, but we don't have space here to list all of the relevant articles. We have instead listed some that we consider particularly readable and important. The interested reader can use these articles to gain access to other papers in each area. We hope that those authors we have not referenced will forgive us for omitting their articles from these lists!

REFERENCES

Protein Folding

- [1] R. Callender, R. Gilmanshin, B. Dyer, and W. Woodruff, "Protein Physics," *Physics World*, August, p. 41 (1994). A general introduction to proteins and the folding problem. Aimed at physicists.
- [2] T. E. Creighton, Ed., 1992, *Protein Folding*, W. H. Freeman & Co, New York. An introduction to proteins and the folding process.
- [3] K. A. Dill, K. M. Fiebig, and H. S. Chan, "Cooperativity in Protein-Folding Kinetics," *Proc. Natl. Sci. USA* **90**, 1942 (1993).
- [4] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science* **220**, 671 (1983). An important paper that pointed out the wide applicability of the Monte Carlo annealing process we employed in our studies of protein folding. This simulated annealing procedure has proved

extremely useful in problems ranging from the behavior of random spin systems to the design of integrated circuits.

- [5] E. M. O'Toole and A. Z. Panagiotopoulos, "Monte Carlo Simulation of Folding Transitions of Simple Model Proteins Using a Chain Growth Algorithm," *J. Chem. Phys.* **97**, 8644 (1992).
- [6] A. Šali, E. Shakhnovich, and M. Karplus, "How Does a Protein Fold?" *Nature* **369**, 248 (1994).
- [7] E. Shakhnovich, G. Farztdinov, A. M. Gutin, and M. Karplus, "Protein Folding Bottlenecks: A Lattice Monte Carlo Simulation," *Phys. Rev. Lett.* **67**, 1665 (1991). Our protein model was inspired by the one studied by these authors.

Earthquakes

- [8] P. Bak and C. Tang, "Earthquakes as a Self-Organized Critical Phenomenon," *J. Geophys. Res.* **94**, 15635 (1989). A discussion of self-organized criticality from the people who invented the term. A rather different model of earthquakes is described in this paper.
- [9] R. Burridge and L. Knopoff, "Model and Theoretical Seismicity," *Bull. Seismol. Soc. Am.* **57**, 341 (1967).
- [10] J. M. Carlson, "Two-Dimensional Model of a Fault," *Phys. Rev. A* **44**, 6226 (1991).
- [11] J. M. Carlson and J. S. Langer, "Properties of Earthquakes Generated by Fault Dynamics," *Phys. Rev. Lett.* **62**, 2632 (1989); "Mechanical Model of an Earthquake Fault," *Phys. Rev. A* **40**, 6470 (1989). Our simulations follow closely the work described in these papers, which also references earlier work on similar models.
- [12] B. Gutenberg and C. F. Richter, "Earthquake Magnitude, Intensity, Energy, and Acceleration," *Bull. Seismol. Soc. Am.* **46**, 105 (1956).
- [13] B. Gutenberg and C. F. Richter, "Magnitude and Energy of Earthquakes," *Ann. Geofis.* **9**, 1 (1956).

Neural Networks

- [14] J. Hertz, A. S. Krogh, and R. G. Palmer, 1991, *Introduction to the Theory of Neural Computation*, Addison-Wesley, New York. A good introduction to neural networks from a physics perspective.
- [15] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Natl. Acad. Sci. USA* **79**, 2554 (1982). A key paper that helped revive interest in neural networks among physicists. Our model is based on the calculations described in this paper.

- [16] W. Kinzel, "Learning and Pattern Recognition in Spin Glass Models." Z. Phys. B **60**, 205 (1985). The calculations in this chapter are similar in many respects to those described in this paper.

Real Neurons and Action Potentials

- [17] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," J. Physiol **117**, 500 (1952).
- [18] C. Koch, 1999, *Biophysics of Computation*, Oxford University Press, New York. An introduction to the physics and biophysics of the nervous system.

Cellular Automata

- [19] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized Criticality: An Explanation of 1/f Noise," Phys. Rev. Lett. **59**, 381 (1987).
- [20] P. Bak, C. Tang, and K. Wiesenfeld, "Self-organized criticality," Phys. Rev. A **38**, 364 (1988).
- [21] P. Bak, K. Chen, and M. Creutz, "Self-organized criticality in the Game of Life," Nature **342**, 780 (1989).
- [22] M. Gardner, "The fantastic combinations of John Conway's new solitaire game life," Scientific American, October, p. 120 (1970).
- [23] M. Gardner, "On cellular automata, self-reproduction, the Garden of Eden and the game life," Scientific American, February, p. 112 (1971).
- [24] J. C. Heudin, 1994, *La Vie Artificielle*, Hermès, Paris.

Ordinary Differential Equations with Initial Values

A.1 FIRST-ORDER, ORDINARY DIFFERENTIAL EQUATIONS

Many of the problems encountered in physics involve ordinary differential equations for which some initial values are specified. For example, the motion of a projectile is described by second-order differential equations that involve the position as a function of time. For these problems, we would like to find the quantity of interest (such as the position of the projectile) as a function of time. In other words, we want to *propagate* the relevant function forward in time starting from the given initial value.

There are several ways to attack such problems numerically, and which ones are appropriate or effective depends on the problem. Thus we begin by considering first order ordinary differential equations of the form

$$\frac{d}{dt}x(t) = f(x, t) \quad (\text{A.1})$$

with the initial condition that $x(0) = x_0$. (Note that our nuclear decay problem in Chapter 1 is of this type.) In this case, the Taylor series expansion of x around t gives

$$x(t + \Delta t) = x(t) + \frac{dx}{dt} \Delta t + \frac{1}{2} \frac{d^2 x}{dt^2} (\Delta t)^2 + \dots \quad (\text{A.2})$$

Thus, assuming a reasonably smooth function $x(t)$ and a small interval Δt , we can propagate $x(t)$ to $x(t + \Delta t)$ to any accuracy desired as long as we know the derivatives of $x(t)$.

However, we often do not know the derivatives for general values of t , and we also want to propagate for much more than an infinitesimally small interval Δt . So we are faced with the problem of estimating derivative(s) of an unknown function as well as the need to repeat the propagation by Δt many times. Different ways of doing this yield different approximations associated with different levels of error.

The Euler method introduced in Chapter 1 corresponds to dropping terms of order $(\Delta t)^2$ and higher in (A.2). If we are somehow given $x(t_i)$, the exact value of x at time $t_i = i\Delta t$, propagation forward with the Euler method gives

$$x(t_{i+1}) \approx x(t_i) + f(x(t_i), t_i)\Delta t. \quad (\text{A.3})$$

The \approx symbol here reminds us that the right-hand side in (A.3) is only an approximation for the exact value $x(t_{i+1})$. The *local* order of a particular approximation is determined by the order in Δt (or whatever the expansion parameter happens to

be) to which the approximation agrees with the exact solution. Comparing (A.3) and (A.2) we see that the Euler method is locally a first order approximation at t_{i+1} , and it is easy to see that it remains so at later values t also. If we want to calculate $x(b)$ for some fixed value of $t = b$, starting from $t = 0$, then the Euler propagation step must be repeated $N = b/\Delta t$ times. Since the error is of the order of $(\Delta t)^2$ at each step, the final estimate for $x(b)$ will have an error of order $N(\Delta t)^2 \propto \Delta t$. According to the conventional terminology, the Euler method is said to be an approximation of zeroth order *globally*. So, even though you could reduce the actual error by reducing Δt , the error at $t = b$ is only reduced linearly as Δt (and the computing requirements go up linearly).

With the Euler method, and with all other approaches, the choice of Δt is very important. Smaller values of Δt will give smaller errors (if one ignores the possibility of round-off error), but reducing Δt leads to greater computational cost (i.e., more steps and a longer run time for your program). On the other hand, too large a value of Δt can make an approximation unstable or meaningless. For example, in our nuclear decay problem in Chapter 1, $f(x, t) = -x/\tau$ where τ is the decay time constant. Thus the first Euler propagation step gives

$$x(\Delta t) \approx x_0 - (\Delta t/\tau)x_0. \quad (\text{A.4})$$

If one takes $\Delta t = \tau$, this approximation becomes $x(t) \approx 0$, and even worse, if $\Delta t > \tau$ is taken, the Euler result for $x(t)$ oscillates in sign.

How can we systematically improve the situation? One obvious answer is to keep to higher orders in the Taylor expansion. For example, we could take an iterative approach

$$x(t_{i+1}) \approx x(t_i) + f_i \Delta t + \frac{1}{2} \left[f_i \left. \frac{\partial f}{\partial x} \right|_i + \left. \frac{\partial f}{\partial t} \right|_i \right] (\Delta t)^2, \quad (\text{A.5})$$

where the subscript i on the derivatives indicates evaluation at x_i and $t = t_i$. While this would produce an approximation that is locally of second order and globally of first order, a drawback of this approach is that the partial derivatives of $f(x, t)$ must be evaluated.

Many other approaches can be constructed which yield the same order of approximation without the explicit need to evaluate the partial derivatives. In this context, it may be useful to think of the problem as the integration of dx/dt from, say, t to $t + \Delta t$. According to the mean value theorem, there is a value t_m in the interval $[t, t + \Delta t]$ such that the *exact* solution can be gotten while stopping at first order in Δt

$$x(t + \Delta t) = x(t) + \left. \frac{dx}{dt} \right|_{t_m} \Delta t. \quad (\text{A.6})$$

In this sense, the slope $dx/dt|_{t_m}$ incorporates the effects of the curvature (the second and higher order terms) present in the Taylor expansion. Of course we do not generally know t_m or the slope, and thus must somehow estimate them approximately. The Euler method replaces t_m by t on the right-hand side of this equation. Graphically, this corresponds to linearly extrapolating up to $(t + \Delta t)$ by using a line tangent to $x(t)$ at t (see Figure A.1).

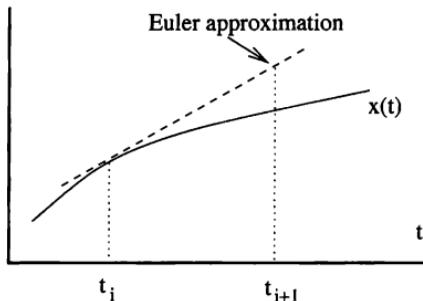


FIGURE A.1: Geometrical interpretation of the Euler method. The solid curve is the actual function $x(t)$. If the Euler approximation is used starting at t_i , the Euler estimate for $x(t_i + \Delta t) = x(t_{i+1})$ is the value obtained by extrapolating linearly from t_i using the slope at $t = t_i$. This extrapolation misses the true value by an amount that (to leading order) is proportional to the curvature (that is, the second derivative) of $x(t)$.

Even though the above expression resembles the Taylor expansion truncated at the first order, they are not the same. By estimating t_m and $dx/dt|_{t_m}$ intelligently, we can effectively construct approximations of higher order than the Euler method. A popular example is the Runge-Kutta methods. The most common second-order Runge-Kutta approximation is constructed by

$$x(t + \Delta t) = x(t) + f(x', t')\Delta t, \quad (\text{A.7})$$

where

$$\begin{aligned} x' &= x(t) + \frac{1}{2}f(x(t), t)\Delta t \\ t' &= t + \frac{1}{2}\Delta t. \end{aligned} \quad (\text{A.8})$$

In other words, the slope $dx/dt|_{t_m}$ is estimated as the value $f(x', t')$ where t' is the midpoint of the interval and x' is the Euler approximated value of x at t' . It is easy to show that this approximation is indeed locally of second order [i.e., agreeing with the Taylor expansion up to order $(\Delta t)^2$], and of first order globally. This means that the error in $x(b)$ for fixed b can be reduced quadratically by reducing Δt .

The cost of the increased accuracy of (A.7) is a larger computational requirement. For each step of propagating by Δt , we must first obtain (x', t') by the Euler method and then substitute the results into (A.7), so we have roughly twice the computing requirement of the Euler method. One way to put the relative computational costs in perspective is to estimate the resources required to obtain comparable accuracy by using different methods. When making such comparisons it is useful to think of the Taylor expansion in terms of a dimensionless expansion parameter. For

our nuclear decay problem we must normalize Δt by the relevant time scale for the problem, τ . So when we speak of the powers of Δt , we are really thinking of $\Delta t/\tau$ as our expansion parameter. Now suppose that we wish to improve on an original Euler method calculation which used $\Delta t/\tau = 0.01$ (which is typical). By keeping the same Δt but going to the second-order Runge-Kutta method, we may expect to improve the accuracy by a factor of about 100. On the other hand, we would have to reduce Δt by the same factor to achieve a comparable improvement using the Euler method. Since the Runge-Kutta approach only doubles the computational requirement while this change in the Euler approach would increase it by 100-fold, it would appear to be a hands-down win for the Runge-Kutta approach. Of course, the values of these numerical errors depend very much on the exact coefficients in our Taylor expansions, and thus on the functional form of $f(x, t)$. In addition, there may be other considerations; e.g., you may wish to use a small value of $\Delta t/\tau$ anyway for smooth interpolation.

There are many other possible second-order approximations, as there are many ways to approximate the slope $dx/dt|_{t_m}$ in (A 6) in such a way as to agree with the Taylor expansion up to second order. In general Runge-Kutta approximations, one estimates this slope by a weighted average of several terms of the form $f(x'_i, t'_i)$ where t'_i ($i = 1, 2, \dots$) are suitably chosen values in the interval $[t, t + \Delta t]$, and x'_i are obtained by using some Euler or Euler-like approximation for $x(t'_i)$. Even limiting ourselves to this type of approximation, there are infinite number of second-order choices. A popular higher order approximation is the fourth-order Runge-Kutta method defined by

$$x(t + \Delta t) \equiv x(t) + \frac{1}{6}[f(x'_1, t'_1) + 2f(x'_2, t'_2) + 2f(x'_3, t'_3) + f(x'_4, t'_4)]\Delta t , \quad (\text{A.9})$$

where

$$\begin{aligned} x'_1 &= x(t) , & t'_1 &= t \\ x'_2 &= x(t) + \frac{1}{2}f(x'_1, t'_1)\Delta t , & t'_2 &= t + \frac{1}{2}\Delta t \\ x'_3 &= x(t) + \frac{1}{2}f(x'_2, t'_2)\Delta t , & t'_3 &= t + \frac{1}{2}\Delta t \\ x'_4 &= x(t) + f(x'_3, t'_3)\Delta t , & t'_4 &= t + \Delta t . \end{aligned} \quad (\text{A.10})$$

It is left as an exercise to confirm that the local error in this approximation is indeed of $O([\Delta t]^5)$. This particular fourth-order Runge-Kutta method requires the evaluation of $f(x', t')$ four times and Euler propagation also four times, and thus consumes roughly four times the computing requirements per step of the Euler method. However, it is possible to take a much larger Δt than with the Euler method at comparable accuracy. (See Figure A.2.) For this reason, the fourth-order Runge-Kutta method is usually the approximation of choice for numerical problems requiring relatively high accuracies, when no specially constructed approximations are available or desired.

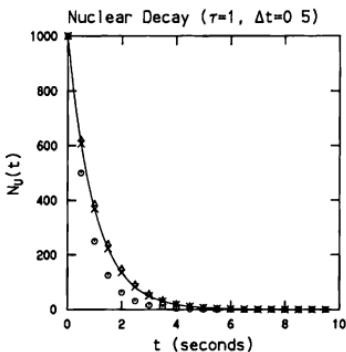


FIGURE A.2: Comparison of the nuclear decay problem of Chapter 1 using the Euler (circles), second-order Runge-Kutta (triangles), and fourth-order Runge-Kutta (crosses) methods for $\Delta t = 1$ and $\tau = 0.5$. The solid curve is the exact solution $N_U(t) = N_U(0) \exp^{-t/\tau}$.

A.2 SECOND-ORDER, ORDINARY DIFFERENTIAL EQUATIONS

In the examples just discussed we assumed that the first derivative was known. However, in many cases encountered in physics it is the second derivative that can be calculated. For example, if y is the position of an object, then Newton's second law tells us that the second derivative d^2y/dt^2 is equal to the force on the object divided by the mass. Initial value problems of this kind are typically solved numerically by starting from the known initial values $y_0 \equiv y(t_0)$ and $v_0 \equiv dy/dt|_{(y_0, t_0)}$ and propagating forward in t with $t_i \equiv t_0 + i\Delta t$. We begin by converting the problem into one involving two first-order differential equations

$$\begin{aligned}\frac{dy}{dt} &= v \\ \frac{dv}{dt} &= f_2(y, t),\end{aligned}\tag{A.11}$$

where $f_2(y, t)$ is our second derivative function.¹

To apply the Euler method we write each of these first derivatives in finite difference form, truncating the Taylor expansion at first order. Proceeding as we did in the first-order case yields

$$\begin{aligned}y(t_{i+1}) &\approx y(t_i) + v(t_i) \Delta t \\ v(t_{i+1}) &\approx v(t_i) + f_2(y(t_i), t_i) \Delta t,\end{aligned}\tag{A.12}$$

¹It is also possible for f_2 to depend on v . For simplicity we will assume that this is not the case, but the results that follow could easily be generalized to handle this situation.

where the \approx symbols remind us again that the right-hand sides here are only approximations to the true solution. Geometrically this has the same interpretation as in Figure A.1. The derivatives evaluated at (y_i, t_i) are used to make linear extrapolations to obtain estimates of v_{i+1} and y_{i+1} .

To see the Euler method in action we now apply it to a simulation of simple harmonic motion. The equation of motion for this problem is (see Chapter 3)

$$\frac{d^2y}{dt^2} = -A y, \quad (\text{A } 13)$$

where A is a constant; hence $f_2 = -Ay$. The main part of a program that implements the Euler algorithm to solve (A.13) is outlined below. The results for y and t are stored in the arrays. We also calculate the total energy of the oscillator, which in this case is given by

$$E = \frac{1}{2} v^2 + \frac{1}{2} A y^2, \quad (\text{A } 14)$$

the first term being the kinetic energy (we take the mass to be unity), while the second term is the potential energy. Note that we do not store $v = dy/dt$ in an array since we don't plan to use it later.

EXAMPLE A.1 Euler method for the harmonic oscillator

- Set y_0 and v_0 to the (given) initial values.
- Loop through each t_i ($i \geq 1$):
 - ▷ $v_{i+1} = v_i - Ay_i \Delta t$ (Euler method for v).
 - ▷ $y_{i+1} = y_i + v_{\text{old}} \Delta t$ (Euler method for y).
 - ▷ Calculate energy with v_{i+1} and y_{i+1} .
 - ▷ Increment t to $t_{i+1} = t_i + \Delta t$.

Some results for the simulation are shown in Figure A.3. Our system does indeed oscillate, but the amplitude of the oscillation appears to grow with time, as does the energy. This is clearly *not* the kind of behavior we expect for a simple harmonic oscillator described by (A.13). Since there is no energy being injected into our system, the total energy *should* remain constant, but this is not what our simulation yields. The problem lies with the Euler algorithm. While the method provides us with a very simple and reasonably accurate solution, it is only an approximation. The error terms we noted above in connection with (A.12) are not zero. It turns out that for this particular problem these error terms effectively add a small amount to the energy each time step. In some cases the errors introduced by the Euler method are negligible. For example, in the problems considered in Chapters 1 and 2 the Euler method was perfectly adequate. However, in oscillatory problems the errors introduced by the Euler method generally tend to accumulate, with results like those shown in Figure A.3. For this reason we did not use this method to treat the oscillatory problems in Chapters 3 and 4.

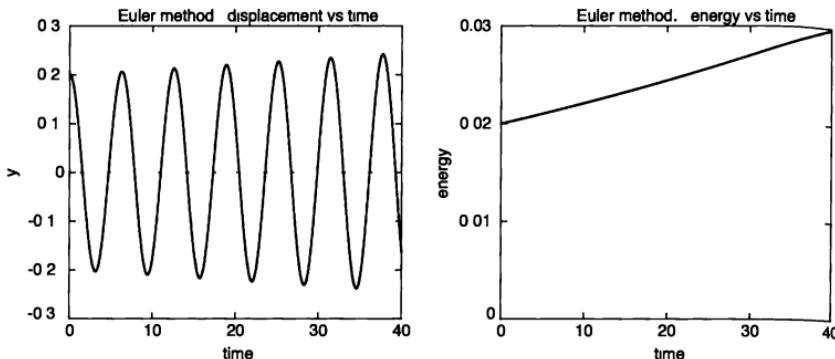


FIGURE A.3: Simulation of a simple harmonic oscillator using the Euler method. Left: displacement, y , versus time; right: total energy as a function of time. The time step was $\Delta t = 0.01$, $A = 1$, and the initial conditions were $y = 0.2$ and $v = 0$. Since the period $\tau = 2\pi/\sqrt{A} \approx 6.28$, we have $\Delta t \ll \tau$ as required.

There are many ways to deal with this instability problem. One obvious method might be to reduce Δt and hope to achieve better accuracy. While this works to some extent, the errors still tend to accumulate and the instability eventually resurfaces. Another way is to go to a higher order approximation such as the Runge-Kutta methods of second or higher order, where the errors in, e.g., energy will be of higher order too, though they may still accumulate over many periods.

There is, however, a simple modification of the Euler method which *sometimes* cures the problem of error accumulation at $O([\Delta t]^2)$. We have seen that the Euler method uses y_i and v_i , the estimates for y and v at time step i , to calculate the estimates at the next time step $i + 1$. A slight change in this procedure yields the Euler-Cromer method, which we described in Chapter 3. With this method we use y_i and v_i to estimate v_{i+1} , but then use y_i and v_{i+1} to estimate y_{i+1} .

We could justify this in part by arguing as follows. Part of the problem with the Euler method is the *one-sided* approximation of the slope $dx/dt|_{t_m}$ appearing in the mean value theorem (A.6). This problem can be reduced by using a more symmetric evaluation of dy/dt and dv/dt , namely, evaluating dv/dt at t while evaluating dy/dt at $t + \Delta t$. In fact, a careful analysis (see Cromer [1981]) shows that for oscillatory problems this algorithm actually *conserves energy* over each complete oscillation at least $O([\Delta t]^2)$. This makes it a better choice for problems such as simple harmonic motion than the Euler method.

The Euler-Cromer method is discussed in Chapter 3 in the context of this and similar problems, so we will not consider it any further here. However, we want to caution that while the Euler-Cromer method is preferable to the Euler algorithm for oscillatory problems, the overall accuracy of the two methods for

other types of problems is usually comparable as the Euler-Cromer method for (A.11) is still a locally first-order approximation just like the Euler method.² In particular, while the Euler-Cromer method conserves the total energy over integer periods in oscillatory problems, this is not necessarily the case in other situations.

As an example of using a higher-order approximation to curb the problems of instability in oscillatory solutions, we again consider the simple harmonic oscillator problem, but this time use the second-order Runge-Kutta method. The second-order Runge-Kutta equations in this case are

$$\begin{aligned} y_{i+1} &= y_i + v' \Delta t \\ v_{i+1} &= v_i + f_2(y', t') \Delta t, \end{aligned} \quad (\text{A.15})$$

where

$$\begin{aligned} y' &= y_i + \frac{1}{2} v_i \Delta t \\ v' &= v_i + \frac{1}{2} f_2(y_i, t_i) \Delta t \\ t' &= t_i + \frac{1}{2} \Delta t, \end{aligned} \quad (\text{A.16})$$

and the function f_2 for the oscillator problem is again just $f_2 = -Ay$.

A subroutine that calculates the displacement and energy of our oscillator using (A.15) is outlined below.

EXAMPLE A.2 Second-order Runge-Kutta method for the harmonic oscillator

- Set y_0 and v_0 and calculate energy for $i = 0$.
- Loop through each t_i ($i \geq 1$):
 - ▷ Calculate $y' = y_i + \frac{1}{2} v_i \Delta t$.
 - ▷ Calculate $v' = v_i - \frac{1}{2} A \cdot y_i \Delta t$.
 - ▷ Let $y_{i+1} = y_i + v' \Delta t$.
 - ▷ Let $v_{i+1} = v_i - A \cdot y' \Delta t$.
 - ▷ Increment t_i to $t_{i+1} = t_i + dt$.
 - ▷ Calculate the energy: $\frac{1}{2}(v_{i+1}^2 + y_{i+1}^2)$.

The results obtained using this algorithm are shown in Figure A.4. The oscillations are seen to be quite stable, that is, their amplitude remains constant with time. This is in sharp contrast to the results obtained with the Euler method (Figure A.3) for the same problem. The energy is also seen to be constant to within the resolution of this plot.

²However, the Euler-Cromer approximation for $y(t)$ [not $v(t)$] has a higher-order accuracy (if f_2 does not depend on v) as it can be seen to be functionally equivalent to the Verlet method and other so-called centered difference methods discussed later.

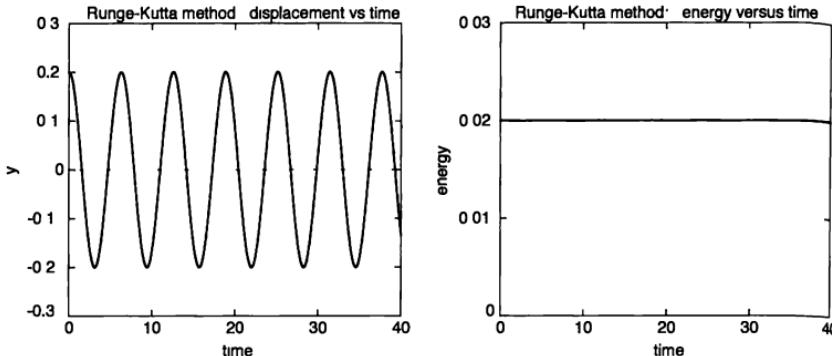


FIGURE A.4: Simulation of a simple harmonic oscillator using the Runge-Kutta method (A.15). Left: displacement, y , versus time; right: total energy as a function of time. The time step was $\Delta t = 0.01$, $A = 1$, and the initial conditions were $y = 0.2$ and $v = 0$.

A.3 CENTERED DIFFERENCE METHODS

You may have noticed that almost all of the methods discussed above (with the exception of the Euler-Cromer method) treats the extrapolation across the interval t to $t + \Delta t$ in a somewhat “unsymmetrical” manner. This can be traced back to the evaluations of the derivative $dx/dt|_{t_m}$ appearing in the mean value theorem (A.6). For example, the Euler method evaluates the derivatives at one end of the t -interval. The usual Runge-Kutta second-order method uses a more complicated way to estimate it as the mid-point derivative, but the way it does this is itself asymmetric. The situation is similar also with other higher order Runge-Kutta methods. All these can be thought of as somewhat “forward”-looking in the estimation of the derivative. While these methods do improve the accuracy by going to a more complicated derivative estimation, simple but more “symmetric” approaches can also lead to smaller numerical errors. This is the basis of centered difference methods.

Let us now describe one such method, called the Verlet method.³ Consider again the second-order ordinary differential equation

$$\frac{d^2y}{dt^2} = f_2(y, t). \quad (\text{A.17})$$

Previously we looked for numerical solutions of this equation by first converting it into two first-order equations (one for y and the other for $v \equiv dy/dt$), and using the first derivatives in the Taylor expansions of y and v . However, if we are only interested in obtaining $y(t)$ and if f_2 does not depend on v , then we can skip dealing

³The introduction of this method to physicists seems to have been the result of a paper by Verlet, which described its application to molecular dynamics.

with the first derivatives entirely. The Taylor expansion of the function $y(t)$ is

$$\begin{aligned}y(t_i + \Delta t) &= y(t_i) + \frac{dy}{dt} \Delta t + \frac{1}{2} \frac{d^2y}{dt^2} (\Delta t)^2 + \frac{1}{6} \frac{d^3y}{dt^3} (\Delta t)^3 + \dots \quad (\text{A.18}) \\y(t_i - \Delta t) &= y(t_i) - \frac{dy}{dt} \Delta t + \frac{1}{2} \frac{d^2y}{dt^2} (\Delta t)^2 - \frac{1}{6} \frac{d^3y}{dt^3} (\Delta t)^3 + \dots\end{aligned}$$

As usual, we anticipate that t will be a discrete variable with time step Δt , so that $y(t_i + \Delta t) = y_{i+1}$ and $y(t_i - \Delta t) = y_{i-1}$. We have also given the Taylor expansion for extrapolating *backward* in time. Of course, the forward and backward expansions are quite similar, the only difference being in the signs of the odd-order terms. Now if we add these two equations, *all* of the odd-order terms cancel exactly. We can thereby derive y_{i+1} in terms of y_i and y_{i-1} as

$$y_{i+1} = 2y_i - y_{i-1} + \frac{d^2y}{dt^2} (\Delta t)^2 + O([\Delta t]^4) \quad (\text{A.19})$$

Thus we define our approximate solution for y by

$$y_{i+1} \approx 2y_i - y_{i-1} + f_2(y_i, t_i)(\Delta t)^2, \quad (\text{A.20})$$

and the lowest-order correction term is of order $(\Delta t)^4$. This is the Verlet method. We have achieved higher accuracy by approximating d^2y/dt^2 symmetrically using centered differences. If we are also interested in the behavior of v (e.g., because f_2 also depends on it), the Verlet method approximates it from the centered difference as

$$v_i \approx (y_{i+1} - y_{i-1})/(2\Delta t), \quad (\text{A.21})$$

where the error is of $O([\Delta t]^2)$, thus larger than that for y .

This approach is interesting for several reasons, especially where f_2 does not depend on v . In this case, the error terms for y are one order of Δt smaller than with the second-order Runge-Kutta method. Second, it avoids calculation of $v = dy/dt$ altogether. A minor drawback is that it requires knowledge of y_{i-1} and is thus not self starting. That is, even if the initial conditions specify y_0 (and v_0), the Verlet algorithm requires that y_1 be calculated with some other method. Usually the Euler or Runge-Kutta method is used to estimate y_1 . However, the accuracy which results from (A.19) can be lost if you are not careful in this first step. Consider for example the simple Euler approach where

$$y_1 = y_0 + v_0 \Delta t + O([\Delta t]^2). \quad (\text{A.22})$$

By inserting this expression into (A.19), we see that an error of $O([\Delta t]^2)$ has been introduced into an otherwise higher-order approximation. This error would then propagate to all later times. However, even with this difficulty, the *stability* of the Verlet method would still be a big advantage. For example, in a mechanics problem (such as molecular dynamics), one would still have energy conserved to higher accuracy than with the Euler or second order Runge-Kutta approaches.

A subroutine that employs the Verlet method for our simple harmonic oscillator is illustrated below. This algorithm can be used in place of the corresponding

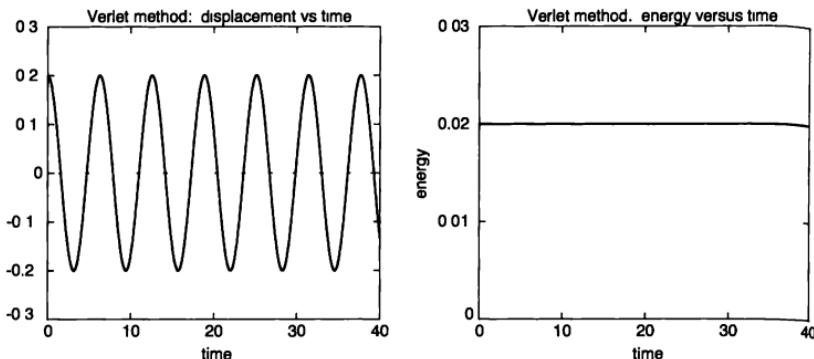


FIGURE A.5: Simulation of a simple harmonic oscillator using the Verlet method. Left: displacement, y , versus time, right: total energy as a function of time. The time step was $\Delta t = 0.01$, $A = 1$, and the initial conditions were $y = 0.2$ and $v = 0$.

Euler or Runge-Kutta approximation given earlier in this appendix. Note that we have used the Euler method to get things started (to calculate y_2), which drops the order of accuracy as discussed above. However, in order to calculate the energy, we require the value of v anyway. Since, with the Verlet method we estimate it via $v_i \approx \frac{1}{2}(y_{i+1} - y_{i-1})/\Delta t$, the error is already $O([\Delta t]^2)$.

EXAMPLE A.3 Verlet method for the harmonic oscillator

- Calculate $y_1 = y_0 + v\Delta t$ (the Euler approximation).
- Loop through each t_i ($i \geq 1$):
 - ▷ Calculate $y_{i+1} = 2y_i - y_{i-1} - Ay_i(\Delta t)^2$ (Verlet method for y).
 - ▷ Calculate $v_i = \frac{1}{2}(y_{i+1} - y_{i-1})/\Delta t$ at t_i .
 - ▷ Estimate energy at t_i using v_i and y_i .
 - ▷ Increment $t_{i+1} = t_i + \Delta t$.

The results from this subroutine are plotted in Figure A.5. The Verlet method does an excellent job for this problem, as the energy remains constant to very high accuracy. Thus, we see that most of the instability inherent in the Euler method for this problem was due to the accumulation of errors at $O([\Delta t]^2)$ which is avoided in the Verlet method.

Other typical methods based on centered differences include the leapfrog method. In this method, dy/dt and dv/dt are approximated as centered differ-

ences evaluated at staggered times. That is,

$$\frac{v_{i+1} - v_{i-1}}{2\Delta t} = f_2(y_i, v_i) \quad (\text{A.23})$$

$$\frac{y_{i+2} - y_i}{2\Delta t} = v_{i+1}. \quad (\text{A.24})$$

Note that i is incremented by 2, so that the derivatives of v and y are written in terms of values at alternative time steps. This leads to the leapfrog approximation

$$v_{i+1} = v_{i-1} + 2f_2\Delta t \quad (\text{A.25})$$

$$y_{i+2} = y_i + 2v_{i+1}\Delta t. \quad (\text{A.26})$$

This method is also *not* self-starting, requiring v_0 in addition to y_1, v_1 . An example of the leapfrog method is used in Chapter 10, where the real and imaginary parts of the wave function play the roles of y and v .

A.4 SUMMARY

In this appendix we have considered several different methods for dealing with ordinary differential equations for which initial conditions are specified and have estimated the numerical errors associated with each. It is tempting to use these errors to classify one method as “superior” or “inferior” to another. However, such a classification is not really appropriate. Each method has its strengths and weaknesses with regard to the types of problems that it can and cannot handle well. For example, the Euler method works well for simple projectile problems, so there is no need to consider the (slightly) more complicated Runge-Kutta and Verlet methods in this case. Our point is only that even the simplest Euler method is adequate in many cases. The optimum algorithm for a problem depends upon the problem. This issue is discussed at length by Press et al. (1986), where you can also find a careful discussion of other algorithms.

EXERCISES

- A.1. Write a program to simulate planetary motion using the Runge-Kutta method and compare its performance with the Euler-Cromer program in Chapter 4.
- A.2. Repeat the previous exercise using the Verlet method.
- A.3. Show that the second-order Runge-Kutta method described in this chapter indeed produces results which are accurate to $O(|\Delta t|^2)$.
- A.4. Show that the fourth-order Runge-Kutta method described in this chapter indeed produces results which are accurate to $O(|\Delta t|^4)$.
- A.5. Come up with a good way to perform the initial step necessary to start the Verlet method (without degrading its accuracy).
- A.6. Show that the Euler-Cromer method is functionally equivalent to the Verlet method as far as the estimation of $y(t)$.

REFERENCES

- [1] A. Cromer, "Stable Solutions using the Euler Approximation," *Am. J. Phys.* **49**, 455 (1981). Discusses the Euler-Cromer method and shows analytically that it conserves energy for oscillatory problems.
- [2] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapter 15 gives a detailed and very readable discussion of methods for dealing with ordinary differential equations.
- [3] L. Verlet, "Computer Experiments on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules," *Phys. Rev.* **159**, 98 (1967). Description of the Verlet method and its application to molecular dynamics simulations.

Root Finding and Optimization

In Chapter 2 we studied the trajectories of realistic projectiles and one of the problems that we considered was finding the launch angle that gives the maximum range. In our work with the variational method applied to quantum mechanics in Chapter 10, we were interested in minimizing the expectation value of the Hamiltonian to obtain the ground state wave function. In these and many other areas we often wish to minimize or maximize a function¹. Such a problem is known as *optimization*. The function to minimize may or may not be available in analytic form, and in fact, may or may not be a continuous function.

A closely related problem is *root finding*. This is the problem of finding the value(s) of variable(s) where a function becomes zero. Optimization and root finding are often treated as independent subjects and a great deal has been written about each of them. However, for a smooth function in a simple domain, local extrema occur when the derivative(s) of the function becomes zero, and thus the two topics share many common features.

B.1 ROOT FINDING

We first address the issue of finding the roots of an equation $f(x) = 0$. As stated above, in some situations this will also serve as the local optimization of its integral $g(x)$, where

$$\frac{dg}{dx} = f(x) \quad (\text{B.1})$$

is either known analytically or can be estimated numerically.

If $f'(x)$ is known analytically, the simplest and most common method to locate the root(s) of $f(x)$ is the *Newton-Raphson* method. This is an iterative method based on truncating the Taylor series expansion of $f(x)$ at first order (analogous to what we did in Euler method approach for dealing with ordinary differential equations). Let the unknown true root be x^* , and x_i be an estimate for x^* . Defining $\Delta x_i = x_i - x^*$, and noting that $f(x^*) = 0$ (since this is the true root), we can write

$$f(x^*) = f(x_i) - f'(x_i)\Delta x_i + O((\Delta x_i)^2) = 0. \quad (\text{B.2})$$

This gives an estimate for x_{i+1}

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \quad (\text{B.3})$$

which (if Δx_i is sufficiently small) is a better estimate for the true root. Graphically, the iteration from x_i to x_{i+1} corresponds to drawing a tangent to $f(x)$ at x_i ; the intersection of this tangent with the x axis is then x_{i+1} , as shown in Figure B.1.

¹Formally, minimization and maximization are equivalent, since minimizing function f is the same as maximizing $-f$

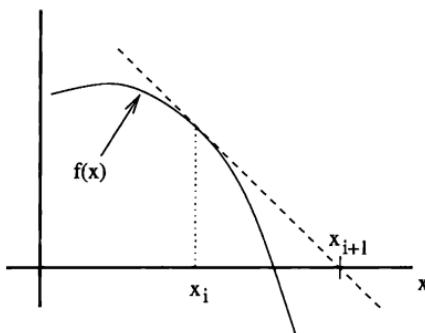


FIGURE B.1: The Newton-Raphson method is illustrated by showing a successive pair of approximants x_i and x_{i+1} for the root of $f(x)$

Although this approximation arises from keeping only up to the linear order in a Taylor expansion, the errors Δx_i can decrease very rapidly. To see this, note that (B.3) gives

$$x^* + \Delta x_{i+1} = x^* + \Delta x_i - \frac{f(x^* + \Delta x_i)}{f'(x^* + \Delta x_i)}, \quad (\text{B.4})$$

which, upon expanding the numerator and denominator on the right-hand side becomes

$$\begin{aligned} \Delta x_{i+1} &= \Delta x_i - \frac{\Delta x_i f' + \frac{1}{2}(\Delta x_i)^2 f'' + O((\Delta x_i)^3)}{f' + \Delta x_i f'' + O((\Delta x_i)^2)} \\ &= \frac{\frac{1}{2}(\Delta x_i)^2 f'' + O((\Delta x_i)^3)}{f' + \Delta x_i f'' + O((\Delta x_i)^2)} \\ &= \left(\frac{f''}{2f'} \right) (\Delta x_i)^2 + O((\Delta x_i)^3), \end{aligned} \quad (\text{B.5})$$

where all of the derivatives are evaluated at x^* . To leading order, the error term (the term containing $(\Delta x_i)^2$ on the right hand side of the last line in (B.5)) decreases quadratically with iteration. The power by which this error decreases is called the order of convergence and thus the order of convergence for the Newton-Raphson method is 2. If $f(x)$ is smooth and the initial guess is close to the desired root, the convergence is typically very fast. However, if your initial guess is far from the desired root, or if $f'(x_i)$ is very small, the convergence will be slow and in some cases the algorithm may never converge at all.²

²Failure to converge can occur when one starts too far from the true root, so that higher order terms in (B.2) cannot be ignored

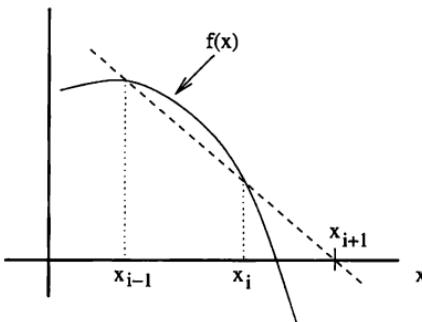


FIGURE B.2: In the secant method, two successive approximants x_{i-1} and x_i for the roots of $f(x)$ are used to obtain the next one x_{i+1}

If $f(x)$ is known analytically but $f'(x)$ is not, a simple modification to the Newton-Raphson method known as the secant method may work well. In the secant method one estimates the derivative $f'(x_i)$ numerically, using

$$f(x_{i-1}) = f(x_i) + (x_{i-1} - x_i)f'(x_i) + O([x_{i-1} - x_i]^2), \quad (\text{B.6})$$

which leads to

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} + O([x_{i-1} - x_i]^2). \quad (\text{B.7})$$

This expression is then substituted in (B.2). Figure B.2 illustrates this method. It is easy to see that successive errors are related as

$$\Delta x_{i+1} = \frac{\frac{1}{2}f''(x^*)\Delta x_i \Delta x_{i-1}}{f'(x^*)} + O([\Delta x_i]^3, [\Delta x_i]^2 \Delta x_{i-1}, [\Delta x_{i+1}]^3). \quad (\text{B.8})$$

Interestingly, the order of convergence is equal to the golden mean $(\sqrt{5} + 1)/2 \approx 1.618\dots$, only slightly worse than the Newton-Raphson method. However, the secant method suffers from all the same shortcomings, and in addition, two initial guesses are needed to start the iterative procedure.

Both the Newton-Raphson and secant methods generalize relatively easily (at least in principle) to multidimensional cases where you need to find a root in an n -dimensional space, given n simultaneous equations each involving n scalar variables. However, the deficiencies of these methods in the one-dimensional case are even more prominent in higher dimensions, and unless you can start the iterative procedure in the vicinity of the root you are after, the methods can fail rather badly.

A method which does not generalize easily to multiple dimensions, but is robust (though less efficient) in one dimension is called *bisection*. If a function f

is continuous in an interval $[x_1, x_2]$ and $f(x_1)$ and $f(x_2)$ are of opposite sign, there must be at least one root in this interval. One can iteratively refine the interval by evaluating f at the midpoint of the interval and selecting the side of the interval for which the end-point values of the function have opposite signs. This method is linear in its order of convergence, but if the first interval brackets a root, it is certain that a root will be found.

B.2 DIRECT OPTIMIZATION

If we wish to optimize $g(x)$ but dg/dx is not readily available (or if we do not wish to use dg/dx even if it is available), what can we do? The simplest and often effective way is to generalize the bisection method discussed above. All such generalizations use the following fact. Suppose that for a continuous function $g(x)$ you have three points $x_1 < x_2 < x_3$ where $g(x_a) < g(x_b)$, $g(x_b) > g(x_c)$, then a local maximum certainly exists in the interval (x_a, x_c) . Thus, once such a situation is found, one can iteratively subdivide the interval to look for a similar situation in a subinterval until the resulting interval is as small as we desire. An example of this approach is illustrated below.

EXAMPLE B.1 A bisection maximization algorithm. Begin with a value x_0 which is known to lie near a local maximum (i.e., between x_a and x_c as defined in the above discussion).

- Choose an initial increment Δx .
 - Evaluate $g(x_0)$ and $g(x_1)$ where $x_1 = x_0 + \Delta x$
 - If $g(x_0) \leq g(x_1)$, interchange x_0 and x_1 and reverse the sign of Δx .
 - Iteration loop (i starts from 2):
 - ▷ Increment x by Δx and evaluate $g(x_i)$.
 - ▷ Test if $g(x_{i-1})$ is greater than both $g(x_{i-2})$ and $g(x_i)$.
 - ▷ If it is, then a maximum is bracketed by $g(x_{i-2})$ and $g(x_i)$.
 - The bracket has the width $2\Delta x$. If this is small enough, exit the loop.
 - Otherwise halve the increment and reverse the sign so that the new Δx is equal to $-(\Delta x)_{\text{old}}/2$.
 - ▷ Increment i and continue with the loop.
-

This bisection method is very simple to implement and quite robust. However, it is clearly not optimal since each iteration is not guaranteed to narrow the interval bracketing a maximum, and the function $g(x)$ might be evaluated at the same point multiple times in search of a narrower bracket. It is possible to avoid such multiple evaluations and minimize the number of iterations needed to achieve the desired interval width by subdividing each interval not equally,³ but by a factor of the

³This will be considered in the exercises

golden mean $(\sqrt{5} + 1)/2$. Such a sectioning scheme can be considerably faster since every step leads to a narrowed interval unlike in the above bisection scheme, but both methods are still linear in the order of convergence, and neither method is quite as efficient as the root-finding bisection method. Also, these types of methods do not generalize easily to higher dimensions.

For one-dimensional problems, even the golden mean approach can be upstaged considerably by using higher order interpolation, if the function to be optimized is smooth. A typical method along these lines is the *Brent method*. If some additional information about the function is available, such as its derivative, the information can be utilized to accelerate the search. In higher dimensional optimization too, a key useful quantity is the gradient of the function being optimized. The basic idea here is to minimize along linearly independent directions, one direction at a time successively. However, naively minimizing simply along the directions of the gradient is often inefficient. A better approach, employed by *conjugate gradient methods*, uses the gradient information efficiently to optimize a function in many variables. There is more written on this topic than we can begin to summarize here. See the references for details.

B.3 STOCHASTIC OPTIMIZATION

So far we have only discussed non-stochastic approaches to optimization. However, algorithms that incorporate a stochastic (i.e., random) process can sometimes be of great value. This is particularly true when the domain of the function to optimized is discrete. Of course, we could discretize even a continuous domain, so this is not an absolute requirement either. We already encountered an example of a stochastic approach to optimization in the variational Monte Carlo method in Chapter 10, which sought to obtain the ground state of a quantum mechanical system by finding the wave function that minimizes the expectation value of the Hamiltonian. In this section we will describe a related but often more powerful method called *simulated annealing*. While there are a number of other equally powerful stochastic approaches (e.g., *genetic algorithms*), we will not have the room to discuss them in this book.

Before we consider the details of simulated annealing, let us first understand why a stochastic approach can be useful in the first place. Suppose that we want to find the minimum of the function $f(x)$ in Figure B.3. We imagine that we are dealing with a multi-dimensional space, and that x represents many different directions in this space. We should expect that this function will contain many local maxima and minima, and that we want to find the *absolute* minimum at x_2 . The basic problem is that we will often have no advanced knowledge of where to find x_2 . If we start far away, and use a method based on low order expansions of $f(x)$ (as in Figs. B.1 and B.2), there is no way that we will find the absolute minimum. Instead, the low order non-stochastic methods will usually find the nearest local minima (e.g., x_1 in Figure B.3). Stochastic optimization methods are designed to overcome this problem.

In simulated annealing, the function to be minimized is interpreted as the energy (Hamiltonian) of a physical system. We apply statistical physics ideas to

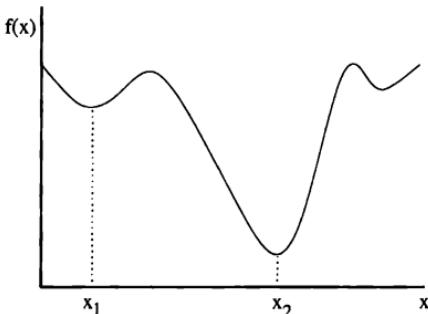


FIGURE B.3: Schematic behavior of a function. We want to find the absolute minimum at x_2 , even if we start very far away (e.g., at x_1)

this system, and imagine that it is in contact with a thermal reservoir at an effective temperature T . We can then use a Monte Carlo simulation method. For any nonzero temperature, our effective system will then fluctuate among different states, according to the Monte Carlo rules (as described in Chapter 8). In Figure B.3 these states correspond to different values of x , so even if we start at x_1 , the algorithm will explore other values of x . If the temperature is large enough, the system will be able to sample states beyond the basin of any local minima, and thereby find the absolute minimum.

We begin with the system in some initial state (i.e., at some initial value of x), and then use the Monte Carlo method to reach equilibrium at a temperature $T > 0$. We then slowly reduce the temperature, a process called *annealing*, as the Monte Carlo simulation is continued. Eventually we will reach a very low temperature, at which point the system will be located near its ground state. Hence, we will reach a value of x very close to the absolute minimum.

While this method can be applied to models of actual physical systems (in which case T is a true temperature), it is often applied to functions which have little to do with a physical system in thermal equilibrium. For functions f unrelated to any real or model Hamiltonian, the concepts of temperature and equilibrium are clearly fictitious, and we could use any reasonable definitions as long as they are useful for our purposes. Typically, we define them so that the states at equilibrium are distributed according to the canonical distribution of statistical mechanics, so that the probability of finding the system in a state s is proportional to the Boltzmann factor $\exp(-E_s/T)$ (where the Boltzmann constant has been absorbed into the definition of T). The principle of ergodicity then allows us to sample these states by successively altering the states using Monte Carlo steps as long as the accepting or rejecting of the new state is decided in such a way to satisfy detailed balance (see Chapter 8). The use of the canonical distribution and Metropolis Monte Carlo algorithm is certainly not the only possibility, but such an implementation has many of the desirable characteristics and is therefore widely used.

An example implementation is sketched below.

EXAMPLE B.2 Simulated annealing minimization of a function of many variables $f(x_1, x_2, \dots, x_N)$

- Pick initial state $(x_1^0, x_2^0, \dots, x_N^0)$ and an initial temperature T_0 . T_0 should be much higher than the changes in f when typical Monte Carlo changes in the state are made.
 - Loop through decreasing temperature T_i ($i = 0, 1, \dots$).
 - ▷ Equilibrate at T_i using Metropolis algorithm with selected, allowed, elementary changes in the states. (Either use a previously estimated value for the number of Monte Carlo steps to equilibrate or judge reaching equilibrium by requiring the rate of change of f averaged over some number of Monte Carlo steps to be small.)
 - ▷ Measure the average value of f (the thermal average). If $i > 1$ and f_{ave} has not decreased sufficiently from $f_{T_{i-1}}$, exit the loop and stop.
 - ▷ Decrease T_i to T_{i+1} . (Typically one reduces the effective temperature gradually, e.g., by 10%.)
-

Implementation of the simulated annealing algorithm requires many decisions that are specific to the function to be minimized. Although no general rules can be given, it is important to start from a sufficiently high temperature, to equilibrate well at each intermediate temperature, and to decrease the temperature gradually. When the function is no longer decreasing much, it sometimes helps to increase T a little and then reduce it again, which can enable the system escape from a local minimum. This procedure can also help confirm that the value of system is indeed near the global minimum.

Since the Monte Carlo algorithm used in simulated annealing takes the system between different states (different values of x), it is particularly efficient for functions f whose domain is discrete and finite, and where there are obvious and effective ways to make small discrete state changes (it is also convenient if these changes are local). The so-called *traveling salesman problem* is a problem that is especially well-suited for attack by simulated annealing. In this problem one wants to minimize the length of the path taken by a salesman who must visit N cities, starting from and returning to one of them, but not visiting any of the other cities more than once. The space of all possible states in this problem consists of the $N!$ permutations of the order of the visits. Removing the obvious N -fold degeneracy by fixing the starting and ending city, we may take

$$f(\Pi) = \sum_{i=1}^N \sqrt{(x_{\Pi(i)} - x_{\Pi(i+1)})^2 + (y_{\Pi(i)} - y_{\Pi(i+1)})^2} \quad (\text{B.9})$$

where Π is a given permutation of the N cities and $\Pi(j)$ is the j -th city in this particular permutation ($\Pi(1) = \Pi(N+1) = 1$). The coordinates of the cities are

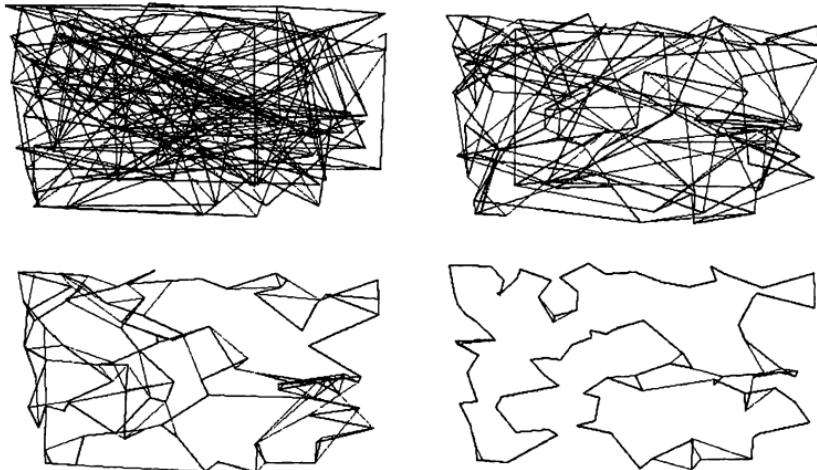


FIGURE B.4: Simulated annealing applied to a traveling salesman problem with 100 cities. From upper left to lower right the temperature was successively lowered from $T = 0.1$ to 0.0168 to 0.00687 and then to 0.000738 , and the corresponding energy E decreased from 5.26 to 0.91 . Typical trajectories, shown as dark lines, gradually settle toward the optimum path. The grey lines show some of the paths also sampled during the Monte Carlo procedure, indicating the kind of fluctuations that are taken into account at each temperature.

given as (x_i, y_i) , where (x_1, y_1) are the coordinates of the chosen terminal city. There are many permutations for any reasonable N (e.g., 15! is greater than 1.3×10^{12}), so it is impractical to find the best path by enumerating all of the possible paths. We therefore attack this problem by simulated annealing. However, only using the obvious elementary permutation operations (such as pairwise reversal of cities) is not very efficient in quickly sampling permutations which are very different from the one we start out with. Much work on this problem (see the references) has shown that a particularly efficient set of elementary operations for this problem involves picking a sequence of cities of any length and either reversing it, or inserting it between two other consecutive cities. Figure B.4 shows an example of using this approach in applying simulated annealing to the solution of a 100-city traveling salesman problem.

There is not room for us to discuss here the many other stochastic optimization schemes. However, we should at least mention *genetic algorithms*. In contrast to simulated annealing where the quantity to be optimized is emulated by a physical system being annealed, genetic algorithms emulate the genetic evolution of living organisms. While simulated annealing moves randomly through different states of the system via a Monte Carlo procedure, a genetic algorithm typically begins with a population of independent samples (i.e., states), and then "crossbreeds" them in a way similar to a living population. From one generation to the next, various possible mutations are allowed and some Darwinian selection is performed to model the survival of the more fit individuals. The reference list contains some further reading for the interested student.

EXERCISES

- B.1. Show that the order of convergence for the secant method is equal to the golden mean.
- B.2. The idea behind improving the bisection optimization method discussed above is to find the constant ratio, say, $1/p$, in which to divide an interval bracketing a maximum (or a minimum) in such a way to surely reduce the interval at each iteration. Suppose that 3 points $x_1 < x_2 < x_3$ bracket a maximum with $f(x_2)$ larger than at the two ends. Assume $(x_3 - x_2)/(x_2 - x_1) = p$. Show that p must be the golden mean if we demand (a) that each subsequent subdivision uses the same ratio, and (b) that each subdivision yields two possible bracketing intervals of equal size.
- B.3. Convert the variational Monte Carlo approach to the Lennard-Jones potential in Chapter 10 into a simulated annealing procedure, and compare the accuracy and efficiency of the two methods.

REFERENCES

- [1] P. R. Bevington and D. K. Robinson, 2003, *Data Reduction and Error Analysis for the Physical Sciences*, 3rd edition, McGraw-Hill, Boston.
- [2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science* **220**, 671 (1983).

- [3] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapter 9 discusses root finding and Chapter 10 discusses optimization.
- [4] M. D. Voss, 1999, *The Simple Genetic Algorithm: Foundations and Theory*, MIT Press, Cambridge.

The Fourier Transform

C.1 THEORETICAL BACKGROUND

The Fourier transform is a subject that is usually reserved for advanced undergraduate mathematics courses and often doesn't sneak into the physics curriculum until somewhat late in the game. This is unfortunate, since the basic ideas are not difficult to grasp, and it is a very useful tool in many problems. Our goal in this appendix is to give a general introduction to the Fourier transform and how it is typically implemented numerically so as to provide a basis for the ideas and techniques we need for the problems discussed in this book.

On the left of Figure C.1 we show a hypothetical signal. It is simply a function that describes how some quantity varies with time. This quantity might be the intensity of a sound wave, the displacement of a particular part of a vibrating string, or the voltage at some point in an electronic circuit. To the eye, this particular signal appears to have an oscillatory character. Thus it should not come as a shock to learn that it was constructed by adding the five individual sine waves shown on the right in Figure C.1. This signal $y(t)$ can thus be written as

$$y(t) = \sum_{j=1}^5 y_j \sin(2\pi f_j t + \phi_j), \quad (\text{C.1})$$

where y_j is the amplitude, f_j the frequency, and ϕ_j the phase of the j th sine wave component.

The total signal in Figure C.1 is a very simple one, so it is probably not surprising that it can be "decomposed" into a collection of sine waves. However, it was shown by Joseph Fourier nearly 200 years ago that virtually *any* signal can be written in this way.¹ That is, any function $y(t)$ can be written as a sum of sine waves. Most signals will be more complicated than the one in Figure C.1, so the sum may involve a large (perhaps infinite) number of sine waves, but just the guarantee that such a sum exists can be extremely helpful. It is convenient to express (C.1) as an integral over frequency f (or equivalently, over angular frequency $\omega = 2\pi f$), which is usually written in the form

$$y(t) = \int_{-\infty}^{\infty} Y(f) e^{-2\pi i f t} df = \frac{1}{2\pi} \int_{-\infty}^{\infty} Y(\omega/2\pi) e^{-i\omega t} d\omega, \quad (\text{C.2})$$

where $i = \sqrt{-1}$. The operation in (C.2) is known as a Fourier transform. The factor $e^{-2\pi i f t}$ in this expression is just the sum $[\cos(2\pi f t) - i \sin(2\pi f t)]$. Hence, in the most general case the Fourier transform function $Y(f)$ will be complex.

¹Strictly speaking, this claim does not apply to extremely pathological signals. We will leave a discussion of such functions to the mathematicians (and sources in the references).

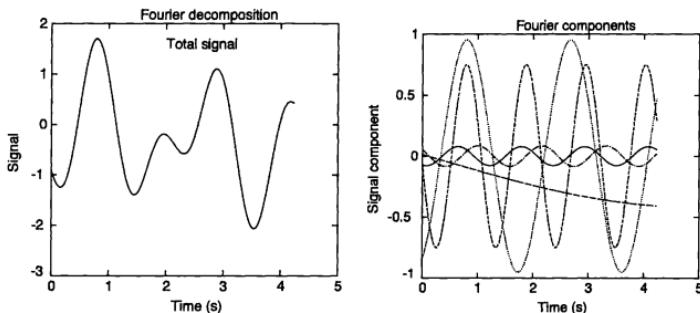


FIGURE C.1. Left Hypothetical signal, right individual sine waves whose sum yields the signal at left

While (C.2) may have a formidable appearance, it is really just a sum of sines or cosines. It is common to refer to $y(t)$ as a function in the time domain and to its transform $Y(f)$ as existing in the frequency domain.² We can think of them as the same object described in two different ways. We move between these parallel descriptions with the Fourier transform. The transform (C.2) takes functions in one direction while its inverse,

$$Y(f) = \int_{-\infty}^{\infty} y(t) e^{2\pi i f t} dt = \int_{-\infty}^{\infty} y(t) e^{i \omega t} dt, \quad (\text{C.3})$$

takes functions in the opposite direction. These are called forward and inverse Fourier transforms (it really doesn't matter which one is considered *forward* and which one *inverse*).³ A forward transform followed by a backward transform returns the original function. This can be verified using the relation

$$\int_{-\infty}^{\infty} e^{i \omega (t-t')} d\omega = 2\pi \delta(t-t'), \quad (\text{C.4})$$

where $\delta(t)$ is the Dirac delta function. Thus, our signal can be described equally well by either function, $y(t)$ or $Y(f)$.

While functions can be written in the form (C.2), it remains to be seen why we would want to do this in the first place. As an example, let us assume that we are dealing with an electronically recorded sound signal. The fact that it can be written

²It can also be useful to consider the Fourier transform of a function that is a function of space. In such cases the variables become space (that is, distance) and wave vector

³Some authors use the factor $e^{-2\pi i f t}$ (rather than $e^{2\pi i f t}$) in the transform from the time domain to the frequency domain and call this the *forward* transform. This is equivalent to the definitions we use in this book since the Fourier cosine transform is the same and the sine transform merely changes sign. We will stick to the conventional usage and call the transform from the time domain to frequency domain the *forward* transform (C 3)

as a Fourier sum means that it can be viewed as (or decomposed into) a sum of “pure” tones (or sinusoids). The frequencies of these tones are the only frequencies present in the original signal, and the Fourier transform $Y(f)$ gives a direct measure of the frequencies that are present. Most sound signals are composed of many such frequencies. In some cases these components may be harmonically related, but things can also be more complicated. In any event, the Fourier transform gives a convenient way to view a signal in the frequency domain, and we will find it useful on many occasions.

Now assume that we have used the Fourier transform to separate the components of a particular signal. What can we do with these besides perform a spectral analysis? Suppose you want to play this sound signal through the speakers in your high-fidelity system. In order to determine how the speakers will respond to this signal, we can first calculate how they would respond to each Fourier component *as if* that component was the *only* signal applied to the speakers. In many physical systems of interest, the total response is just the sum of the individual responses to each component. This approach, which relies on the linearity of the system (in this case the speakers), is intimately linked to the principle of superposition and can be used in a wide variety of problems, including waves on a string (Chapter 6), quantum mechanics (Chapter 10), and heat flow (which is the problem Fourier first treated with this method).

C.2 DISCRETE FOURIER TRANSFORM

The next issue to consider is how to actually compute a Fourier transform. That is, given the function $y(t)$, how *in practice* do we determine $Y(f)$? Suppose that a signal is specified analytically; that is, you are given the functional form of $y(t)$. The Fourier transform $Y(f)$ could then be calculated by simply performing the integral (C.3). Of course, this may not be easy, which is why many mathematics texts have been written on the topic. However, in numerical work we are almost never given the analytic form of the signal, but instead have knowledge of its amplitude at certain discrete values of t . In fact, not only computational but also real experimental data are often recorded at discrete intervals. It is usually the case that the values of $y(t)$ are known at evenly spaced intervals. For example, in our simulations of the pendulum we calculated the angular position at times $t_m = m\Delta t$, where m was an integer and Δt was the time step. In such situations it is useful to define the discrete Fourier transform [compare with (C.2) and (C.3)]

$$y_m = \frac{1}{N} \sum_{n=0}^{N-1} Y_n e^{-2\pi i mn/N} \quad (\text{C.5})$$

$$Y_n = \sum_{m=0}^{N-1} y_m e^{2\pi i mn/N}, \quad (\text{C.6})$$

where the index m on y corresponds to the discrete times $t_m = m\Delta t$, and the index n on Y corresponds to the discrete frequencies $f_n = n/(N\Delta t)$. Here we follow the usual convention and let these indices run from 0 to $N - 1$, where N is the number

of data points. The forward and inverse discrete Fourier transforms are related via

$$\sum_{n=0}^{N-1} e^{2\pi i n(m-m')/N} = N\delta_{m,m'}, \quad (\text{C.7})$$

where $\delta_{m,m'} (= 1 \text{ if } m = m', = 0 \text{ otherwise})$ is the Kronecker delta function. Equation (C.7) means that if we perform a forward transform followed by an inverse transform, we will return to the original function. Again, we can think of our data in the time domain where we have the values y_m , or the frequency domain with Y_n . These are two equivalent ways of describing the *same* collection of data points.

It is important to note that the time step Δt does not enter explicitly into the discrete transforms (C.5) and (C.6). This disappearance is possible because the signal values y_m were obtained at points equally spaced in time. Thus the exponent of the exponential term in the discrete transforms becomes $2\pi i(n/N\Delta t)(m\Delta t) = 2\pi imn/N$, we are also able to drop a factor of $1/\Delta t$ from (C.5), and a factor of Δt from its inverse (C.6) for simplicity. Note that the dropping of these factors from the discrete analogs of the integrals for the continuum transforms gives our discrete Fourier transforms different dimensions from the continuum versions. It also turns out that there are some subtleties associated with the time-frequency correspondence that we will come to in due course.

Returning to (C.5) and (C.6), if we have N data points, that is, N values y_m , then there are N values of Y_n . That is, we have N pieces of information either way. However, this is not quite the whole story. Both the signal y_m and its transform Y_n in (C.5) and (C.6) are, in general, *complex* numbers. Of course, most physical signals can be expressed as real numbers and in such cases the y_m are real. However, even here the Y_n can be complex. This can be understood in the following way. The exponential factors in (C.5) and (C.6) are just a shorthand for the sum $[\cos(2\pi mn/N) \mp i \sin(2\pi mn/N)]$. The imaginary part then causes Y_n to be complex. The real and imaginary parts of Y_n correspond to what are known as the cosine and sine transforms.

Since the Y_n are complex, it appears that we have $2N$ pieces of information in the frequency domain. If the y_m are real, we have only N pieces of information in the time domain. But if these are describing the same function, they must contain the same amount of information. The resolution of this apparent paradox can be understood on two levels. On a fundamental level, the statement that y_m are real implicitly determines N other values, i.e., that its imaginary parts are all zero. Therefore, in fact N values of *real* y_m form $2N$ pieces of information, the same as in the frequency domain. On another level, we note that if the y_m are all real, the Y_n are not all independent. In this case, $Y_{N/2-n}$ can be shown to be the complex conjugate of $Y_{N/2+n}$. Thus we can construct all N points of y_m by Fourier transforming just half of the Y_n , e.g., those with $n = 0, 1, \dots, N/2 - 1$ (if N is even). However, (a) since both the real and imaginary parts of those Y_n are needed, you are still using N pieces of information, and (b) although the real part of the transform obtained this way gives y_m correctly, the imaginary part generally comes out as non-zero and thus incorrect. As you can see, this is because N pieces of information in the frequency domain cannot in general correctly reconstruct $2N$

pieces of information in the time domain. We will have more to say about such issues in a moment. Now we want to return to the question of the frequencies associated with each Y_n . We just noted that for real functions y_m , the Y_n with $n \geq N/2$ are redundant. The highest frequency independent Fourier component is $Y_{N/2-1}$, which corresponds to the frequency $f = (N/2 - 1)/(N\Delta t) \approx 1/(2\Delta t)$ for large N . The special frequency $1/(2\Delta t)$ is known as the Nyquist frequency and plays a very important role. If a signal is measured at time intervals spaced by Δt , then the spectral components that can be recovered with a Fourier transform are those with frequencies below $f_{\text{Nyquist}} \equiv 1/(2\Delta t)$.

If our signal were a simple sine wave at the Nyquist frequency, then we would be sampling it only twice during each period of oscillation, since its oscillation period $2\Delta t$ is twice the sampling time. The amazing thing is that sampling only twice each period is sufficient to capture this Fourier component.⁴ This result is known as the sampling theorem. We will illustrate this and other properties of the discrete Fourier transform below. In particular, we will consider what happens if our signal contains components at frequencies above the Nyquist frequency.

C.3 FAST FOURIER TRANSFORM (FFT)

The discrete Fourier transforms (C.5) and (C.6) are just a sum of exponential terms, so it *appears* to be very amenable to numerical evaluation. However, straightforward evaluation of the sums in (C.5) and (C.6) is computationally very expensive. Each term involves the computation of the exponential factor, which must then be multiplied by y_m and added to the running total. Each sum for a given frequency component has N terms and there are N frequency components, so the total number of operations is of order N^2 . This is bad. It turns out that even with a very fast computer, this brute force approach would take a prohibitively long time for typical values⁵ of N . For this reason a straightforward numerical approach to evaluating discrete Fourier transforms is not practical.

While the simplest approach to evaluating (C.5) and (C.6) would require of order N^2 operations, this does not mean that *all* approaches must involve the same number of operations. The exponential terms in the discrete Fourier transforms are multiples of one another. This makes it possible to intelligently group terms in the sums in such a way that you can “reuse” many of them in evaluating different Fourier components Y_n . In fact, such an approach makes it possible to evaluate the discrete transform with only of order $N \log N$ operations as opposed to N^2 . This is a major time reduction for large transforms, and the savings are substantial even for $N \sim 1000$, a relatively small value we will find useful for the problems in this book. There are several specific algorithms of this kind, and they are collectively known as the *fast Fourier transform* or FFT. The existence of the FFT has made many important calculations feasible; it is used in technologies such as X-ray tomography, and there are many off-the-shelf devices these days that incorporate FFT functionality. The FFT is an excellent example of an improvement in calculational efficiency that makes previously impractical tasks possible.

⁴This is the reason why audio CD's are usually mastered at the sampling rate of 44 kHz, just enough to cover the highest audible frequency of about 20 kHz

⁵ $N \sim 10^6$ is not uncommon in many applications

Most scientific programmers are not likely to program an FFT themselves, and will instead use preprogrammed routines from a standard scientific subroutine library. Even so, we believe that the FFT algorithm is a good example of an algorithmic innovation that every computational scientist should know about and appreciate. FFT algorithms used in practice are sufficiently complicated that we will not give a full explanation (see the references) but opt instead for a demonstration of the principle involved using a simple example.

The basic approach of the FFT can seen by considering the sums in (C.3) for a small set of data points. Thus consider the case with $N = 2^3 = 8$, so that Y_n for a given n involves the sum of 8 terms y_m ($m = 0, 2, \dots, 7$). We can split this sum into two parts, one dealing with even m and the other with odd m , or

$$\begin{aligned} Y_n &= Y_n^e + w^n Y_n^o \\ &= \sum_{m'=0}^3 y_{2m'} w^{2m'n} + w^n \sum_{m'=0}^3 y_{2m'+1} w^{2m'n}, \end{aligned} \quad (\text{C.8})$$

where $w = e^{2\pi i/N}$. Note that the even part Y_n^e involves the terms y_m with original indices m whose binary representation has the least significant bit equal to 0 while the odd part Y_n^o involves those with the least significant bit of m equal to 1.

If we use a binary representation of the index $n = 4n_2 + 2n_1 + n_0$ where n_2 is the most significant bit (0 or 1) and likewise n_0 is the least significant bit, we have

$$w^{2m'n} = e^{2\pi i 2m'(4n_2+2n_1+n_0)/8} = w^{2m'(2n_1+n_0)}, \quad (\text{C.9})$$

since $w^{2m'n_2} = 1$ for both $n_2 = 1$ and 0. The dependence on the bit n_2 thus drops out of both Y_n^e and Y_n^o to give

$$Y_n = Y_{n_1, n_0}^e + w^n Y_{n_1, n_0}^o. \quad (\text{C.10})$$

We can repeat this and split Y_n^e into $Y_{n_1, n_0}^{ee} + w^{2n} Y_{n_1, n_0}^{eo}$, and Y_n^o into $Y_{n_1, n_0}^{oe} + w^{2n} Y_{n_1, n_0}^{oo}$, where the second superscript refers to whether the second significant bit in m is even or odd (0 or 1). However, similarly to (C.9), the dependence on n_1 drops out of all these terms except in the explicit factors of w . Moreover, $w^{2n} = w^{4n_1+2n_0}$ and we have

$$\begin{aligned} Y_{n_1, n_0}^e &= Y_{n_0}^{ee} + w^{4n_1+2n_0} Y_{n_0}^{eo} \\ Y_{n_1, n_0}^o &= Y_{n_0}^{oe} + w^{4n_1+2n_0} Y_{n_0}^{oo}. \end{aligned} \quad (\text{C.11})$$

For our example of $N = 8$, this splitting can be continued one further time depending on the most significant bit of m involved, and one ends up with the individual terms of the original sum. That is,

$$\begin{aligned} Y_{n_0}^{ee} &= Y^{eee} + w^{4n_0} Y^{eoo} = y_0 + w^{4n_0} y_4 \\ Y_{n_0}^{eo} &= Y^{eoe} + w^{4n_0} Y^{eo0} = y_2 + w^{4n_0} y_6 \\ Y_{n_0}^{oe} &= Y^{oee} + w^{4n_0} Y^{ooo} = y_1 + w^{4n_0} y_5 \\ Y_{n_0}^{oo} &= Y^{ooo} + w^{4n_0} Y^{ooo} = y_3 + w^{4n_0} y_7. \end{aligned} \quad (\text{C.12})$$

This may not sound like much of an accomplishment. However, on closer inspection the brilliance of this algorithm can be seen. Let us count how many operations are needed to calculate all $N = 8$ values of the Fourier transform Y_n . At the level of (C.12), there are just 2 different values (corresponding to $n_0 = 0, 1$) which need to be evaluated for each $Y_{n_0}^{xy}$ where the superscript xy is one of the 4 different even-odd combinations. This amounts to 8 multiplications of the powers of w and 8 additions. At the level of (C.11), there are 4 different values (for $n_0, n_1 = 0, 1$) to evaluate for each Y_{n_1, n_0}^x where x is one of e or o . This amounts also to 8 multiplications and 8 additions. Finally, at the (C.10) level, 8 different values need to be evaluated (for $n_0, n_1, n_2 = 0, 1$) again amounting to 8 multiplications and 8 additions. Thus each level requires the number of operations proportional to N (8 in our example) and there are $\log_2 N$ ($= 3$ here) levels. Thus we get the $N = 8$ Fourier transforms in about $N \log_2 N = 24$ steps rather than $N^2 = 64$ steps! As noted earlier, as N increases this difference quickly becomes a decisive factor in determining if a calculation is feasible or not.

Of course, this is just the principle, and in order to make this principle work well in practice we also need an efficient bookkeeping method to store and reuse the common terms. In particular, the FFT is typically performed in place, i.e., returning the transformed output values in the same data arrays as used for the input. Also, a bit reversal⁶ is done at the beginning or at the end of the calculations so that the output values come out in the desired sequence. A schematic illustration of an example of this algorithm is given below. There are several variations on this example which are equally effective (see the references for details).

EXAMPLE C.1 A fast Fourier transform algorithm

- Read in data y_m ($m = 0, 1, \dots, N - 1$).
- Find the power p where $N = 2^p$. (If N is not a power of 2, then pad the input data by zeros up to the next power of 2)
- Bit reverse the indices of the input data array. (This is a bookkeeping trick)
- Outer loop through levels $i = 1, 2, \dots, p$ of even-odd decompositions, beginning with the final decomposition where individual y_m are used (C.12) through the first decomposition where Y_n^e and Y_n^o are involved (C.10).
 - ▷ Nested middle loop where each level is grouped into terms according to the values of k in the factor $e^{2\pi ik/N}$ appearing in each sum. (This middle loop did not appear in the conceptual discussion given above.)
 - Nested inner loop where each group is split into individual sums like those in (C.12) through (C.10).

⁶Bit reversal of a number refers to first writing it in binary representation and then forming another number from the binary form obtained by reversing the bit order in the original binary representation

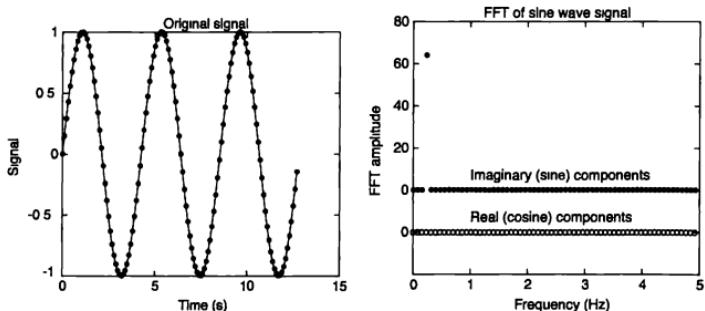


FIGURE C.2: Left: Pure sine wave signal, $y(t) = \sin(2\pi ft)$. These 128 points are the values of the signal that are Fourier analyzed. Right: FFT of this signal. The real (cosine) and imaginary (sine) parts of the transform are shown separately. We have plotted them as discrete points to emphasize that the number of Fourier amplitudes yielded by FFT is equal to the number of original data points. Hence we have 64 cosine and 64 sine components. Only one of the points in the transform is nonzero.

C.4 EXAMPLES: SAMPLING INTERVAL AND NUMBER OF DATA POINTS

Two parameters usually under our control in a discrete Fourier transform are the sampling interval and the number of points to sample. Intelligent choices of these parameters are an important factor in obtaining useful transforms. The sampling interval determines the range of spectral frequencies that can be represented, while the number of data points determines the details that can be obtained. There are also other factors such as whether the sampling duration corresponds to whole periods of the signal, and whether the number of data points match what the particular FFT algorithm is designed for.

To get a feeling for how Fourier analysis works in practice, we now consider a few examples. Perhaps the simplest possible signal is a pure sine wave, such as the one shown in Figure C.2. The signal period here is approximately 4.3 s and a sampling period of 0.1 s is chosen. The sampling period of 12.8 s has been chosen so that the total recorded signal spans precisely three complete periods.⁷ Note also that the 128 signal values used in the analysis⁸ are the points plotted in Figure C.2. The FFT of this signal is shown on the right in Figure C.2. The results are 128 Fourier amplitudes, half of which are the amplitudes of the component sine waves and half of which are the cosine amplitudes. We see that all of these are zero, *except one*, the one corresponding to the sine wave we started with. Hence, the FFT tells us that our signal is composed of a single Fourier component corresponding to a sine wave with a frequency of ≈ 0.23 Hz.

⁷While not absolutely necessary, we will employ the units of seconds (s) and (hertz) Hz in the following discussion, as an aid in appreciating the connection between the time and frequency domains.

⁸As implied above, the algorithm that actually evaluates the FFT works best when the number of data points it receives is a power of 2.

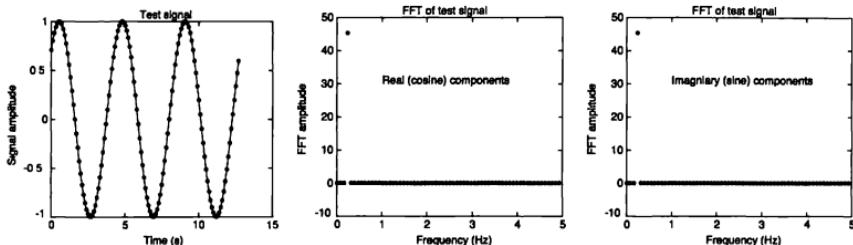


FIGURE C.3: Left: Test signal that is just a sine wave that is phase shifted by $\pi/4$. The dots are the data points that were used in the FFT; center and right: FFT of this test signal

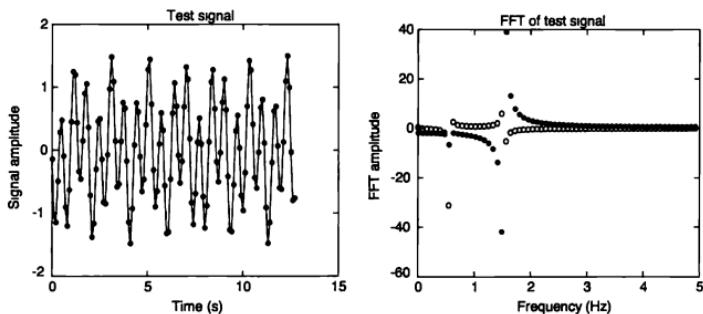


FIGURE C.4: Left: Test signal which is the sum of two sine waves with different frequencies, amplitudes, and phases; right: FFT of this test signal. The filled circles show the real (cosine) components, while the open circles are the imaginary (sine) amplitudes

Next we consider essentially the same signal, but now we shift it along the time axis by adding a phase factor of $\pi/4$. That is, our signal is the function $y(t) = \sin(2\pi ft + \pi/4)$ shown in Figure C.3. It is again sampled 128 times at intervals $\Delta t = 0.1$ s. The FFT is now slightly more complicated, with one nonzero Fourier sine component and one nonzero cosine component. These correspond to writing the signal as the sum of a sine and cosine, that is, $\sin(2\pi ft + \pi/4) = [\sin(2\pi ft) + \cos(2\pi ft)]/\sqrt{2}$. This is precisely what we find in the FFT result and shows why an FFT often yields nonzero sine and cosine components at each frequency. They are both needed if we are to be able to describe a signal with an arbitrary phase [recall the phase factors in (C.1)].

The very simple FFT results we have observed in our first two examples are due in part to the fact that we have chosen the period of the signal to precisely match the total sampling time. That is, we have sampled three *complete* periods.

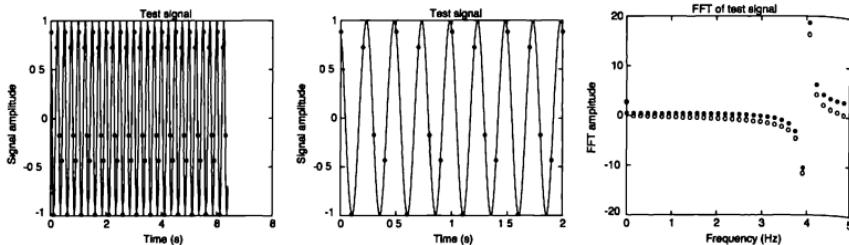


FIGURE C.5: Left: Test signal that is a single sine wave. The curve is what the signal would look like if it were sampled at a very large number of points, while the solid symbols show the 64 data points that were analyzed in the FFT. Center: an expanded view of the signal, right: FFT of this signal. The filled circles are the real (cosine) components, while the open circles are the imaginary (sine) components.

If the sampling time does not match the frequencies of the Fourier components, the FFT has a slightly more complicated appearance. In Figure A2.4 we consider a hypothetical signal that consists of two sine waves of different frequencies, neither of which is commensurate with the sampling time. The FFT shows large components at two frequencies that match the frequencies of the sine waves used to construct the original signal. However, we also find Fourier amplitudes that are small but nonzero over a range of frequencies. This can be understood if we recall that the frequencies of the discrete transform are $f_j = j/N\Delta t$, where j runs from 0 to $N/2 - 1$. If a frequency contained in the signal does not coincide with one of these discrete frequencies, the FFT is forced to represent the signal as a sum of components over a range of f_j . However, it is important to note that such a representation will still give a *perfect* description of the original data values.

C.5 EXAMPLES: ALIASING

We have already mentioned the sampling theorem, which says essentially that the FFT will give us a perfect description of the Fourier components as long as the frequencies of these components are below the Nyquist frequency, $1/2\Delta t$. But what happens if this condition is not satisfied? In Figure C.5 we show a pure sine wave signal that varies rapidly with time. The frequency here was 4 Hz, and the filled circles show the signal values used in the calculation. The FFT obtained using these 64 data points is also shown and exhibits a sizable component at 4 Hz, as expected. Note that the Nyquist frequency in this case was 5 Hz, so the sampling theorem says that we should indeed be able to successfully handle this signal.

Figure C.6 shows what happens when the signal is sampled with a different value of Δt . Here we used $\Delta t = 0.2$ s (N is now 32, so there are half as many Fourier components), which gives a Nyquist frequency of 2.5 Hz. This is lower than our signal frequency so we expect trouble, and we do indeed find it. The FFT now exhibits a peak at 1 Hz, far from the signal frequency of 4 Hz.

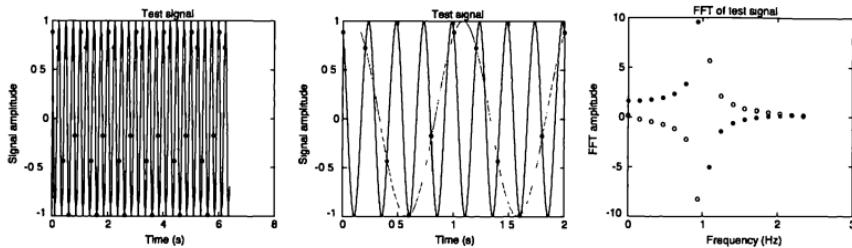


FIGURE C.6: Left test signal that is a single sine wave. The solid curve is what the signal would look like if it were sampled at a very large number of points, while the solid symbols show the 32 data points that were analyzed in the FFT. Center an expanded view of the signal. The dotted curve shows a second sine wave, which has a much lower frequency (1 Hz) and which is seen to also pass through all of the data points. Right FFT. The filled circles are the real (cosine) components, while the open circles are the imaginary (sine) components

The location of this peak comes about in the following way. The frequency of the sine wave was greater than the Nyquist frequency, which means that there were fewer than two sampled points per period. In this situation the sampled points y_k , can be described with equal precision by either of *two* different sine waves, one at the “true” frequency and one at a frequency that is *below* the Nyquist frequency. Here the “true” frequency was 4 Hz. Let us consider how the frequency of the other sine wave, which passes through these points, can be calculated. Since the Fourier transform below and above the f_{Nyquist} are related by the relation $Y_{N/2-n} = Y^*_{N/2+n}$ (when the y_m are real), if there is a peak at 4 Hz, there will also be a corresponding one at the frequency obtained by “reflecting” it about f_{Nyquist} . In this problem $f_{\text{true}} - f_{\text{Nyquist}} = 4 - 2.5 = 1.5$ Hz. The reflected frequency is then 1.5 Hz below the Nyquist frequency, that is, at 1 Hz, and this is where we find a peak in the FFT

We must be careful here when referring to the “true” frequency. When samples are recorded only at intervals of $\Delta t = 0.2$ s, these two sine waves (one at 4 Hz and one at 1 Hz) yield precisely the same signal. This is illustrated in the center part of Figure C 6, which shows that a sine wave with a frequency of 1 Hz passes precisely through all of the data points. Thus, if all we have are these data points, who is to say which is the true frequency? This folding back of frequencies above the Nyquist frequency is known as *aliasing*.

In practice it is preferable to arrange for the Nyquist frequency to be higher than any of the Fourier components that are expected to be present in the signal. In an experiment this can be accomplished by using a low pass filter to remove components with frequencies above the Nyquist frequency, so there is no possibility of aliasing. In numerical work this is generally not a problem, since the sampling interval is usually the time step of a simulation, which should always be small compared to the characteristic time scales of the problem. This is equivalent to saying that all of the Fourier components lie below the Nyquist frequency.

Of course if the frequency components of interest lie much below the other components also present in the signal, then a practical problem arises. We must use a sufficiently small sampling interval to have the Nyquist frequency lie above all present frequencies, but in order to obtain any detail in the spectral region of interest, we need to have a reasonable number of points in that small region compared to the whole spectrum below the Nyquist frequency. This usually requires taking a very large number of data points in the time domain. We saw an example of this in Chapter 3 where we analyzed the spectrum of the nonlinear pendulum. Another typical problem arises when there is a dominant but uninteresting frequency component (such as a dc component, i.e., a component at zero frequency). In such cases, it is best to subtract it out from the signal before Fourier transforming.

C.6 POWER SPECTRUM

To this point we have always displayed the real (cosine) and imaginary (sine) parts of the FFT separately. Such a presentation has the advantage that it contains all of the information in the original signal. This is essential if we want to later use a backward transform to return to the time domain. However, we are often interested only in the frequencies and relative amplitudes of the Fourier components, but don't really care about their phases. In such cases there is another useful way to display the results of an FFT, known as the *power spectrum*. Formally, the power spectrum of a function $y(t)$ can be defined as the Fourier transform of its *autocorrelation*,

$$\text{Corr}[y](\tau) = \int_{-\infty}^{\infty} y(t)^* y(t + \tau) d\tau , \quad (\text{C.13})$$

and the power spectrum is given by

$$PS[y](f) = \int_{-\infty}^{\infty} y(t)^* y(t + \tau) e^{2\pi i f \tau} d\tau = |Y(f)|^2 . \quad (\text{C.14})$$

The autocorrelation function $\text{Corr}[y](\tau)$ measures how well the signal y is correlated across times which are separated by τ , or with a periodicity of τ . Thus if it has structure with peaks at certain periodicities τ , then its Fourier transform, the power spectrum $PS[y](f)$, will also have structure with peaks at *conjugate* or corresponding values of frequencies $f = 1/\tau$.

The power spectrum can be understood in a more practical sense in the following way. Suppose that $y(t)$ is an electrical signal, such as the voltage as a function of time across a resistor. The average power dissipated in the resistor during a long time interval is proportional to

$$P = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T |y(t)|^2 dt . \quad (\text{C.15})$$

Since

$$\int_{-\infty}^{\infty} |Y(f)|^2 df = 2\pi \int_{-\infty}^{\infty} |y(t)|^2 dt , \quad (\text{C.16})$$

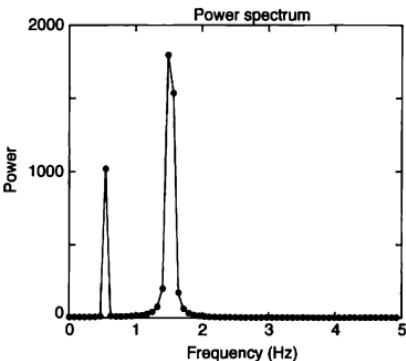


FIGURE C.7: Power spectrum of signal in Figure C.4. The solid symbols are the calculated values of the power

we see that

$$P \propto \int_{-\infty}^{\infty} |Y(f)|^2 df . \quad (\text{C.17})$$

Thus the average power, $P(f)$, dissipated per unit frequency interval around f is proportional to $|Y(f)|^2$, the power spectrum. For the discrete case, the corresponding average power P_j at a frequency f_j is proportional to the sum of the squares of the amplitudes of the cosine (real) and sine (imaginary) components at f_j

$$P_j = Y_j(\text{real})^2 + Y_j(\text{imaginary})^2 . \quad (\text{C.18})$$

FFT results for $Y_j(\text{real})$ and $Y_j(\text{imaginary})$ can thus be used to compute the power at each frequency, f_j .

As an example, Figure C.7 shows results for the power spectrum of the signal in Figure C.4. Here there are just two peaks at the appropriate frequencies, and their relative sizes are proportional to the squares of the corresponding Fourier amplitudes.⁹ Note that in computing the power (C.18) we discard the phase information, since the relative magnitudes of the sine and cosine components cannot be determined solely from P . It is thus not possible to recover the original signal from knowledge of the power spectrum alone.

EXERCISES

- C.1. Write a program that uses the FFT to filter a signal. Take as input an arbitrary signal (such as the one in Figure C.1), and begin by using a forward FFT to calculate the individual components. Components whose frequencies are above a

⁹To be precise we should really compare the areas under each peak in the power spectrum

given cutoff frequency can be removed by simply setting those Fourier amplitudes to zero in Y_k . Then perform a backward FFT to obtain the original signal with the high frequencies filtered out.

REFERENCES

- [1] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapter 12 discusses Fourier transform methods from a practical point of view.
- [2] R. N. Bracewell, 1978, *The Fourier Transform and Its Applications*, McGraw-Hill, New York. Gives a very readable theoretical account of the FFT algorithm.

Fitting Data to a Function

D.1 INTRODUCTION

One is often faced with a collection of data points, perhaps the results of an experiment in which a quantity y has been measured as function of a second variable x . You may then be given the task of devising (or divining) a mathematical relationship between these two quantities. We have seen this in many problems, including our calculations of fractal dimensionalities (Chapter 7), and the precession of the perihelion of Mercury (Chapter 4). Another common situation is where a simplified theory or model predicts the qualitative form of the mathematical relation, but leaves it to experiment or simulation to determine numerical coefficients in this relation. An example was seen in Chapter 2 where the air drag coefficient C would be 1 in a simple theory but, a more accurate value of C can only be found from experiment. In such cases it is desirable to estimate the values of the unknown coefficients themselves by fitting the given (real or numerical) data to the expected theoretical form. The systematic algorithms used to make such estimations are called *regression*, and are at the heart of data fitting.

Regression may also be useful in the absence of a theoretical prediction, if we wish to present the general trend of “noisy” data obtained in an experiment or the results of a numerical calculation. By constructing a “best fit” curve through the data, we can get a synopsis of the results of experiment (real or numerical). This would fall under the subject of interpolation (or extrapolation). Of course we must be cautious in such an attempt, as the chosen fitting function might turn out to have little to do with the true function which should have been used. It is advisable to reflect carefully on the selection of a fitting function and, in particular, to avoid the use of high-order polynomial functions unless there is no other way out.

Regression is typically not computationally intensive, and generally does not require a delicate choice of methods or parameters. One can usually employ a routine from your favorite subroutine library without fear. However, it is still useful to understand what these routines are doing, and what the outputs of the common fitting procedures mean. It is worthwhile to note that fitting data to an expected function is *not* the same as testing the validity of that function. When regression is involved in such a test, the process typically consists of two stages. In the first stage, a particular theoretical model with unknown parameters is fitted to produce the best-fit values of those parameters. Then in the next stage, a suitable measure of the deviations from the best-fit case¹ is used to judge how confident one should be in rejecting or accepting the fitted theory. Sometimes this procedure is

¹This could be the least-squares value or the so-called minimum χ^2 , depending on whether the variances at each value of x are the same or not. See the next section and also Appendix G for more details.

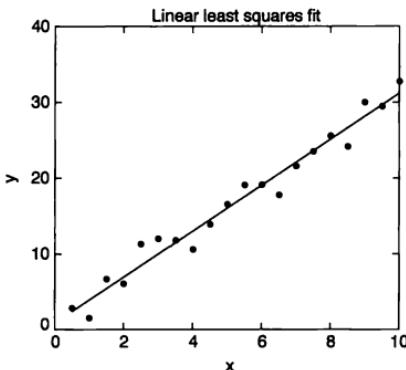


FIGURE D.1: Example of a linear least-squares fit. The points are some hypothetical data and the line is the best fit, least-squares line

repeated for competing theories and the results are compared to see which one is best (or least objectionable).

D.2 METHOD OF LEAST SQUARES: LINEAR REGRESSION FOR TWO VARIABLES

We start the discussion with the problem of fitting data of two variables to a straight line as illustrated in Figure D.1. In this figure, the points cluster around a straight line, and the line drawn in the figure seems like a good choice. However, how do we know if this is the best line for this particular set of data, and if it is, how do we go about constructing it?

We imagine that we have some data that consists of a collection of N equally weighted points $[x(i), y(i)]$, where the index i runs from 1 to N . In physics we most often have situations where one of the two variables, x , is under our control and the other one, y , is the resulting measurement. The points in Figure D.1 are a typical example. To construct a best-fit line that describes these points we must first have a criteria for what it means to be "best." An extremely convenient and nearly universally chosen criteria is the following. The fitted line will be of the form $y_{fit} = ax + b$, where a is the slope and b the intercept. This line will come close to passing through all of the points, but in general will not pass exactly through any of them. The difference between the actual and fitted values is just $\Delta y(i) \equiv y(i) - y_{fit}(i)$, where $y_{fit}(i) = ax(i) + b$.

The *least-squares* criteria for the best fit is that the sum of the squares of these differences be minimized. That is, we want to choose a and b so as to make

$$\Delta_{LS} \equiv \sum_{i=1}^N [\Delta y(i)]^2 \quad (D.1)$$

as small as possible. Since Δ_{LS} is a function of a and b , this means that the values of a and b that satisfy our best fit criteria are the ones for which²

$$\frac{\partial \Delta_{LS}}{\partial a} = 0 \quad (\text{D.2})$$

$$\frac{\partial \Delta_{LS}}{\partial b} = 0. \quad (\text{D.3})$$

If we compute these derivatives using (D.1) we find

$$\begin{aligned} a \sum x(i)^2 + b \sum x(i) &= \sum x(i) y(i) \\ a \sum x(i) + b N &= \sum y(i) \end{aligned} \quad (\text{D.4})$$

where all of the sums run from $i = 1$ to N . The problem thus reduces to solving two algebraic equations for the two unknowns, a and b . The solution is just

$$\begin{aligned} a &= \frac{\sum x(i) \sum y(i) - N \sum x(i)y(i)}{[\sum x(i)]^2 - N \sum x(i)^2} \\ b &= \frac{\sum x(i) \sum x(i)y(i) - \sum x(i)^2 \sum y(i)}{[\sum x(i)]^2 - N \sum x(i)^2}. \end{aligned} \quad (\text{D.5})$$

Since the *sample mean* (*mean* for short) of a variable is obtained by summing over its N values and dividing by N , the means \bar{x} and \bar{y} of x and y are given by

$$\bar{x} = \frac{\sum x(i)}{N}, \quad \bar{y} = \frac{\sum y(i)}{N}, \quad (\text{D.6})$$

and we can rewrite (D.5) as

$$a = \frac{\bar{xy} - \bar{x}\bar{y}}{\bar{x}^2 - \bar{x}^2} \quad (\text{D.7})$$

$$b = \frac{\bar{x}^2\bar{y} - \bar{x}\bar{xy}}{\bar{x}^2 - \bar{x}^2}. \quad (\text{D.8})$$

In addition, (D.4) and (D.6) immediately show that $a\bar{x} + b = \bar{y}$, so that the point (\bar{x}, \bar{y}) lies on the best fit line.

Now that we have obtained the solution to this least-squares problem, we can appreciate the beauty of the process. This criteria for a “best” fit automatically leads to a set of algebraic equations for a and b . The only way the procedure can fail is if the denominator in (D.5) vanishes. This will happen only if all of the $x(i)$ are the same; that is, if all of the data points have the same x value.³ So long as at least two of the points have different x values, we are guaranteed that (D.5) is the unique solution to the least-squares problem.

²You might worry that this procedure could actually locate a maximum in Δ_{LS} . However, for a quadratic form such as (D.1) there is only one extrema, and it is the minimum we are after.

³In this case a line with infinite slope would pass exactly through all of the data.

However, how well does the best fit line really fit the data? If you were to perform an additional measurement $y(N+1)$ at $x(N+1)$, how much would it deviate from $y_{\text{fit}}(x(N+1))$? Or, if there is a *true* value of y at $x(N+1)$, how close would it be to $y_{\text{fit}}(x(N+1))$? All these are relevant questions to which we would like to know the answers.

Many of these questions can be addressed if we can obtain an estimate of the reliability of the fitted coefficients, a and b . To find the statistical uncertainties in these coefficients, we begin by considering the following measure of the average scatter in y at a given x

$$s_{y|x}^2 = \frac{\sum(y(i) - y_{\text{fit}}(x(i))^2}{N-2}. \quad (\text{D.9})$$

Here $N-2$ is used in the denominator because, after determining a and b , there are only $N-2$ degrees of freedom left in the data (in statistics jargon this is the *sample space*). After some algebra, one can show that

$$s_{y|x}^2 = \frac{N-1}{N-2}(s_y^2 - a^2 s_x^2), \quad (\text{D.10})$$

where s_x^2 and s_y^2 are the fluctuations (the *sample variances*) of x and y , respectively, with

$$s_x^2 = \frac{\sum(x(i) - \bar{x})^2}{N-1}, \quad s_y^2 = \frac{\sum(y(i) - \bar{y})^2}{N-1}, \quad (\text{D.11})$$

The denominators here are $N-1$ since this is the number of degrees of freedom left in the sample space after the means \bar{x} and \bar{y} are determined. The quantity $s_{y|x}$ is a measure of the variation of y in the data which cannot be explained by the best fit line y_{fit} , and this variation is the source of uncertainty for the best fit slope a . Since this variation in y occurs over the overall range in x of order $\sqrt{\sum(x(i) - \bar{x})^2}$, our estimate for the uncertainty in a is

$$s_a = \frac{s_{y|x}}{s_x \sqrt{N-1}}. \quad (\text{D.12})$$

To obtain the uncertainty s_b in the intercept b , first note that the best fit line always passes through the point (\bar{x}, \bar{y}) . Thus, a natural measure of s_b is simply

$$s_b = s_a \bar{x}. \quad (\text{D.13})$$

However, this is strictly an uncertainty associated with the intercept of the best fit line. If we want a measure of the scatter in y about the best fit line, the appropriate quantity is $s_{y|x}$ itself. For example, if we ask how far the next measurement $y(N+1)$ at $x(N+1)$ might be from the best fit expected value of $y_{\text{fit}}(x(N+1))$, then $s_{y|x}$ gives you an average measure of that. On the other hand, if we ask how far the *true* value y at a given x might be from the best fit expected value of $y_{\text{fit}}(x)$, then a more appropriate value would be $s_{y|x}/\sqrt{N}$, if the best-fit line was based on N data points

At the same time, we may ask how much of the variations in y in the data are accounted for by the best-fit line. A measure of this is the ratio of the variation in y attributable to the fitted line to the overall variation in y in the data. This ratio,

$$r = \frac{a\sqrt{\sum(x(i) - \bar{x})^2}}{\sqrt{\sum(y(i) - \bar{y})^2}} = a \frac{s_x}{s_y}, \quad (\text{D.14})$$

is the correlation coefficient. It can be shown that $|r| \leq 1$ and $r = \pm 1$ if and only if all points lie on the fitted line.

If the distribution of y measurements for each given x are normally distributed and if the variance is equal at different values of x , then essentially all of the above discussions can be made more precise. That is, *significance tests* and *confidence intervals* can be formulated. If the first condition of normal distribution is true but the variance is not equal at different values of x , similar results still follow if the least squares (D.1) are appropriately normalized (see the next section). However, for the physics discussions in this book, the above bit of statistics should suffice.⁴

D.3 METHOD OF LEAST SQUARES: MORE GENERAL CASES

There is much more to the least-squares story than we have space to cover here. However, we want to give you at least a glimpse of what those gaps contain.

In the problem considered above we assumed that all of the data points are equally reliable. While this is sometimes appropriate, it often happens that some of the data points are more reliable, i.e., more accurate, than others. We would then expect that the “ideal” line might lie closer to these more accurate data values than to the other data points. The fit procedure should then give increased influence to the more accurate points and less influence to the points that are less certain. This can readily be accommodated within the least-squares approach. If each value of $y(i)$ is associated with an estimate for its uncertainty $\sigma(i)$, the importance, or weight, of each point in Δ_{LS} is taken to be proportional to $1/\sigma(i)^2$. This procedure reduces the weighted sample space to the equally weighted one discussed earlier if the measurement of y at a given x is normally distributed and $\sigma(i)^2$ is the variance of the normal distribution at $x(i)$. The function to be minimized in this case is

$$\Delta_{LS} = \sum_{i=1}^N \frac{[\Delta y(i)]^2}{\sigma(i)^2}. \quad (\text{D.15})$$

We then proceed as above to obtain two algebraic equations for a and b , and the results are similar to (D.5). We will leave the details for the references and the curious reader.⁵ The important point is that the uncertainty associated with each data point can be included explicitly in the fitting process.

⁴For more details, see the references

⁵The value of Δ_{LS} after the minimization then constitutes the so-called χ^2 value for the fitted coefficients. For this reason, this procedure is sometimes called χ^2 fitting. One can apply the machinery of the χ^2 test to judge of the validity of the fitting function; i.e., to judge if this is the right function to use to describe the given data. For details, see Appendix G.

Another useful extension involves fitting more than two variables. For example, suppose that a quantity y is a function of several variables x_1, x_2, \dots, x_m with

$$y_{\text{fit}} = a_1 x_1 + a_2 x_2 + \dots + a_m x_m + b, \quad (\text{D.16})$$

We can think of the variables x_i as defining a hyperspace, with (D.16) specifying a plane in this space. The unknown coefficients a_1, a_2, \dots, a_m and b can be determined by minimizing the squared deviation $\Delta_{MLS} \equiv \sum_{i=1}^N [y_{\text{fit}}(\mathbf{X}(i)) - y(i)]^2$, where we use the vector shorthand $\mathbf{X} = (x_1, x_2, \dots)$. This is called *multiple linear regression*. The minimization of Δ_{MLS} results in a set of $m+1$ simultaneous linear equations for the variables a_1, a_2, \dots , and b . One of these equations, the one arising from the condition that $\partial\Delta_{MLS}/\partial b = 0$, immediately yields

$$\bar{y} = a_1 \bar{x}_1 + a_2 \bar{x}_2 + \dots + a_m \bar{x}_m + b. \quad (\text{D.17})$$

In other words, the mean point $(\bar{x}_1, \dots, \bar{x}_m, \bar{y})$ lies on the best-fit hyperplane. Using (D.17), the remaining m linear equations can be cast in the matrix form

$$\begin{pmatrix} \bar{x}_1^2 - \bar{x}_1^2 & \bar{x}_1 \bar{x}_2 - \bar{x}_1 \bar{x}_2 & \dots & \bar{x}_1 \bar{x}_m - \bar{x}_1 \bar{x}_m \\ \bar{x}_2 \bar{x}_1 - \bar{x}_2 \bar{x}_1 & \bar{x}_2^2 - \bar{x}_2 \bar{x}_2 & \dots & \bar{x}_2 \bar{x}_m - \bar{x}_2 \bar{x}_m \\ \dots & \dots & \dots & \dots \\ \bar{x}_m \bar{x}_1 - \bar{x}_m \bar{x}_1 & \bar{x}_m \bar{x}_2 - \bar{x}_m \bar{x}_2 & \dots & \bar{x}_m^2 - \bar{x}_m^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_m \end{pmatrix} = \begin{pmatrix} \bar{x}_1 \bar{y} - \bar{x}_1 \bar{y} \\ \bar{x}_2 \bar{y} - \bar{x}_2 \bar{y} \\ \dots \\ \bar{x}_m \bar{y} - \bar{x}_m \bar{y} \end{pmatrix} \quad (\text{D.18})$$

The values of regression coefficients a_1, \dots, a_m are obtained by solving this matrix equation; i.e., inverting the matrix on the left in (??).⁶ Dealing with such matrix problems is discussed further in Appendix H.

We may consider the same sort of questions in this case as we did for the two-variable case. For example, the quantity

$$s_{y|x_1, x_2, \dots, x_m}^2 = \frac{\sum (y(i) - y_{\text{fit}}(\mathbf{X}(i)))^2}{N - (m+1)} \quad (\text{D.19})$$

measures the variance of data in y from the best fit hyperplane at a given \mathbf{X} , and the multiple correlation coefficient r can be obtained as

$$r = \sqrt{\frac{a_1(\bar{x}_1 \bar{y} - \bar{x}_1 \bar{y}) + a_2(\bar{x}_2 \bar{y} - \bar{x}_2 \bar{y}) + \dots + a_m(\bar{x}_m \bar{y} - \bar{x}_m \bar{y})}{\bar{y}^2 - \bar{y}^2}} \quad (\text{D.20})$$

There are also results available on the uncertainties on the regression coefficients, and under certain conditions on the distributions, on how to formulate significance tests and confidence intervals analogous to those in two-variable regression.

⁶A unique solution exists as long as the matrix is non-singular. The fitted values of one or more of the coefficients a_1, \dots, a_m will be nonzero provided that there is some correlation between x_1, \dots, x_m and y .

There are important extensions of the multiple linear regression. For example, there are many situations in which a polynomial fitting function is desired, such as

$$y_{\text{fit}} = a_1 x + a_2 x^2 + \dots + a_m x^m + b. \quad (\text{D.21})$$

This can easily be transformed into a multiple linear regression problem by letting $x_1 = x, x_2 = x^2, \dots, x_m = x^m$. While the new variables are clearly not independent, independence is not required in multiple linear regression. All the results given above for multiple linear regression apply here.

Similarly, fitting data to a power law (such as in estimating fractal dimensionalities in Chapter 7) can be cast in the form of linear regression. Suppose that we wish to fit data to a function of the form $y = Ax^B$. Taking the logarithm of both sides yields $\log y = B \log x + \log A$. So, the unknown power B can be determined from a linear least-squares fit involving the variables $\log y$ and $\log x$. Another common case may be fitting to an exponential function, say, $y = A \exp(Bx)$. Again, taking the logarithm of both sides yields $\log y = Bx + \log A$, and a linear least-squares fit involving the variables $\log y$ and x will do the job. Of course, in these transformed linear fits, precise statistical meanings of the best fit parameters are not the same as discussed earlier. For example, if measurements of y at a given x are taken from a normal distribution, the transformed values $\log y$ will no longer follow a normal distribution, and thus much of the uncertainties and significance discussion will not apply for the transformed variables.

Finally, there is no reason why we should be restricted to the situations where a transformation to linear regression is possible. There are many cases in which the appropriate fitting function y_{fit} is much more complicated than a polynomial, power-law, or exponential, and a transformation to a linear form is not practical. In such cases, we can still calculate a squared deviation $\Delta_{NLLS} \equiv \sum_{i=1}^N (y_{\text{fit}}(x(i)) - y(i))^2$ and minimize it by varying parameters present in y_{fit} . Although such fits generally lack the precise statistical interpretations available for linear regression, they are extremely useful. There is a well-developed set of algorithms for handling such fitting to a general function. This goes under the heading of *nonlinear least squares* and is a specialized version of the optimization problem. Unfortunately, such a problem is beyond the scope of this book.

REFERENCES

- [1] P. R. Bevington and D. K. Robinson, 2003, *Data Reduction and Error Analysis for the Physical Sciences*, 3rd edition, McGraw-Hill, Boston.
- [2] E. Crow, F. A. Davis, and M. W. Maxfield, 1960, *Statistics Manual*, Dover Publications, New York.
- [3] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapter 14 discusses many aspects of the method of least squares.

Numerical Integration

E.1 MOTIVATION

Many interesting quantities in physics can be expressed as an integral that is difficult or impossible to evaluate in closed form. This may be due to the complexities of the integrand or of the domain of integration or both. For example, the equation of motion of a non-linear pendulum is

$$\frac{d^2\theta}{dt^2} = - \left(\frac{g}{l} \right) \sin \theta . \quad (\text{E.1})$$

The period of this pendulum can be shown to be

$$T = \sqrt{\frac{8l}{g}} \int_0^{\theta_m} \frac{d\theta}{\sqrt{\cos \theta - \cos \theta_m}} , \quad (\text{E.2})$$

where θ_m is the maximum angle the pendulum makes with vertical. This integral cannot be evaluated analytically in terms of elementary functions.¹ Another source of difficult integrals is the calculation of the magnetic field \vec{B} from the Biot-Savart Law. We discussed this problem in Chapter 5, where we encountered integrands of the form

$$d\vec{B} = \frac{\mu_0 I}{4\pi} d\vec{z} \times \frac{\vec{r}}{r^3} . \quad (\text{E.3})$$

These integrals can be performed analytically only when there is a high symmetry in the problem and the region of integration is very simple, otherwise they often lead to elliptic integrals. It is thus desirable to have a systematic way (or ways) to numerically evaluate integrals of known functions over an arbitrary region of integration.

E.2 NEWTON-COTES METHODS: USING DISCRETE PANELS TO APPROXIMATE AN INTEGRAL

In Chapter 5 a numerical integration method was introduced in which rectangular regions (that we will call “panels”) were used to approximate the area under a curve $f(x)$. In this section we treat this and similar methods of integrating a function a little more systematically. According to the lowest order approximation of this kind, the integral of $f(x)$ from $a = x_0$ to $b = x_N$ is calculated using N rectangular

¹The integral in (E 2) can be transformed into a complete elliptic integral of the first kind, and a lot is known about such integrals. However, the values of these integrals must still be found in a table of complete elliptic integrals or evaluated numerically.

panels of width $\Delta x = (b - a)/N$ as

$$\int_a^b f(x)dx \approx \sum_{i=0}^{N-1} f(x_i) \Delta x . \quad (\text{E.4})$$

To find the error made in this approximation, consider the Euler-Maclaurin summation formula²

$$\begin{aligned} \sum_{i=1}^{N-1} F(i) &= \int_0^N F(u)du - \frac{1}{2}[F(0) + F(N)] \\ &+ \sum_{k=1}^{\infty} \frac{B_{2k}}{(2k)!} [F^{(2k-1)}(N) - F^{(2k-1)}(0)] , \end{aligned} \quad (\text{E.5})$$

where F^k is the k -th derivative of F and the constants B_{2k} are the Bernoulli numbers ($B_2 = \frac{1}{6}$, $B_4 = -\frac{1}{30}$, ...). We can convert this to a form useful for us by making a transformation

$$x = u\Delta x + a . \quad (\text{E.6})$$

Thus, $u = i$ corresponds to x_i ($i = 0, 1, \dots, N$), and (E.5) transforms into

$$\begin{aligned} \sum_{i=1}^{N-1} f(x_i) &= \frac{1}{\Delta x} \int_a^b f(x)dx - \frac{1}{2}[f(a) + f(b)] \\ &+ \sum_{k=1}^{\infty} \frac{B_{2k}}{(2k)!} [f^{(2k-1)}(b) - f^{(2k-1)}(a)](\Delta x)^{2k-1} . \end{aligned} \quad (\text{E.7})$$

The error introduced by truncating the sum at a finite order increases by $O(\Delta x)^2$ each time one additional term is retained. Rearranging and expanding this equation we get

$$\int_a^b f(x)dx = \sum_{i=0}^{N-1} f(x_i)\Delta x + \frac{\Delta x}{2}[f(b) - f(a)] - \frac{(\Delta x)^2}{12}[f'(b) - f'(a)] + O([\Delta x]^4) . \quad (\text{E.8})$$

According to (E.8), the approximation (E.4) has an error of order Δx , which is the same as the global error made by the Euler approximation for solving an ordinary differential equation. As with the Euler approximation, we could reduce the error by simply decreasing Δx . However, this is usually not cost-effective. It is preferable to somehow increase the *order* of the error terms.³ We are in luck here since we can reduce the error to order $(\Delta x)^2$ by replacing the sum $\sum_{i=0}^{N-1} f(x_i)$ in (E.4) by $\sum_{i=1}^{N-1} f(x_i) + (f(a) + f(b))/2$. We can construct a trapezoidal panel between x_i and x_{i+1} by connecting the points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$; the

²This assumes that $F(u)$ can be differentiated any number of times. See the references for a derivation.

³We are actually speaking about *reducing* the error, but the higher the order of error the smaller the actual error will be. Thus it is desirable to *increase* the order of the error.

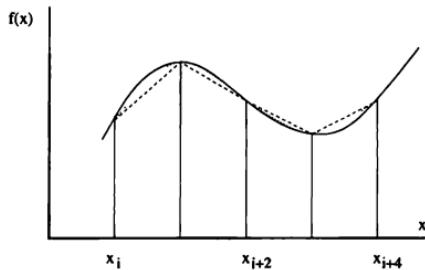


FIGURE E.1: Trapezoidal rule (dashed line) and Simpson's rule (dotted line) are illustrated using 4 neighboring panels for a sample function (solid line). The Simpson's rule parabolas are nearly indistinguishable from the function (the solid line).

area of this panel is given by $(\Delta x)(f(x_i) + f(x_{i+1}))/2$, so the approximation in (E.8) amounts to using trapezoidal panels in place of the rectangular ones in (E.4). This is called the *trapezoidal rule*⁴ and yields

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N-1} f(x_i) + \frac{1}{2}[f(a) + f(b)]. \quad (\text{E.9})$$

The trapezoidal rule yields errors analogous to the second order Runge-Kutta methods introduced in solving ordinary differential equations. The rectangular and trapezoidal rules are examples of Newton-Cotes methods, which use panels of equal width.

We do a better job by using even higher order curves at the upper edges of the panels in Figure E.1. The next higher order approximation, called *Simpson's rule*, is obtained by fitting parabolic curves through the top edges of two neighboring panels (Figure E.1). If A_i is the area of the panel between x_i and x_{i+1} , then the parabolic fit used in Simpson's rule yields the area under two neighboring panels as

$$A_i + A_{i+1} = \frac{\Delta x}{3}[f(x_i) + 4f(x_{i+1}) + f(x_{i+2})]. \quad (\text{E.10})$$

Simpson's rule for the entire integral is thus

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}[f(a) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \dots + 2f(x_{N-2}) + 4f(x_{N-1}) + f(b)], \quad (\text{E.11})$$

and the error is of order $(\Delta x)^4$, analogous to that of fourth order Runge-Kutta methods that we encountered in Appendix A.

⁴The trapezoidal approximation is also equivalent to using rectangular panels whose top edges are centered at $[x_i, f(x_i)]$ for all $i = 1, \dots, N-1$ and using only half panels at $i = 0$ and N . Hence, the approximation suggested by Figure 5.12 is equivalent to the trapezoidal rule.

If we are interested in reducing errors as much as possible, it is to our advantage to use both a small value of Δx (and thus many panels) and a high order approximation. The method called *Richardson extrapolation* does both. Let us denote by $R_{i,j}$ the approximate value of an integral obtained using 2^{i-1} panels with a method of order j (where $j = 1$ is trapezoidal rule and $j = 2$ is Simpson's rule, etc.). If we construct a (lower triangular) table with elements $R_{i,j}$, the farther we move towards the lower right corner, the more accurate an estimate we will find. By populating such a table with the appropriate estimates, we can also estimate the convergence toward the true value of the integral, thus helping us to estimate the errors involved. The calculation of an integral using such extrapolation is called *Romberg integration*. While this may appear to be a computationally costly procedure, it turns out that this calculation is made simpler by a remarkable relation known as *Richardson's formula*,⁵

$$R_{i+1,j+1} = R_{i+1,j} + \frac{R_{i+1,j} - R_{i,j}}{4^j - 1}. \quad (\text{E.12})$$

Thus, knowing the trapezoidal estimates up to, say, 2^{n-1} panels is sufficient to fill the table all the way up to $R_{n,n}$ (with $O([\Delta x]^{2n})$ error). As long as there are no singularities or near singularities (such as places where the derivatives diverge), the convergence is very fast and usually there is no need to go to a large value of n .

Given below is an example of a Romberg table for a complete elliptic integral of the second kind

$$K = \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta} d\theta, \quad (\text{E.13})$$

where we take $k = \sin \pi/2 = 1$:

(i=1)	0.785398			
(i=2)	0.948059	1.002280		
(i=3)	0.987116	1.000135	0.999992	
(i=4)	0.996785	1.000008	1.000000	1.000000
	(j=1)	(j=2)	(j=3)	(j=4)

Thus, convergence to the known exact value of 1 for this example is very rapid, achieving 7-digit precision with just 8 panels. On the other hand, the following integral

$$I = \int_0^2 \sqrt{4 - x^2} dx, \quad (\text{E.15})$$

which yields π turns out to require 64 panels to achieve just 4-digit precision. This relatively slow convergence is mainly due to the infinite slope of the integrand at $x = 2$.

⁵Richardson's formula can be derived by noting that the infinite series sum (E 8) contains only the terms of the even powers of Δx and that these can be successively estimated by using panels of different widths (see the exercises).

E.3 GAUSSIAN QUADRATURE: BEYOND CLASSIC METHODS OF NUMERICAL INTEGRATION

In the above discussion of numerical integration, we always used panels of equal width, i.e., Newton-Cotes methods. We did so because it was the simplest thing to do, as well as being a natural analog of what we did in solving initial-value ordinary differential equations. However, the judicious choice of panels with unequal widths can help in economizing on the number of panels. This is easy to say but it is not obvious how to find the appropriate abscissa points from which to construct the panels. A family of methods called *Gaussian quadrature* provides explicit ways to do this. In fact, many standard handbooks⁶ contain tables of the coefficients needed to make ready use of Gaussian quadrature integration without actually knowing how it works. However, it is such a clever idea that we will very briefly discuss the method here.

The name *Gaussian quadrature* comes from the fact that this idea was first advanced by Gauss. It is based on a mathematical theorem which says that for a suitable, fixed weight function $w(x)$ and the associated integration limits a and b , we can find grid points x_1, x_2, \dots, x_N in the interval (a, b) and also values of the weights w_1, w_2, \dots, w_N such that the following formula holds exactly for any polynomial $f(x)$ of degree less than or equal to $2N - 1$,

$$\int_a^b w(x)f(x) dx = w_1f(x_1) + w_2f(x_2) + \dots + w_Nf(x_N). \quad (\text{E.16})$$

The values of x_i and w_i are independent of $f(x)$, depending only on the weight function $w(x)$, so this theorem provides a powerful way to find the integral of any function that can be expressed (or closely approximated) as the product of $w(x)$ and a polynomial $f(x)$. The limits of integration in the theorem are fixed for a given weight function, but this poses no problem in practice since we can usually transform the integration variable linearly so that any limits will transform to the required interval.

The abscissa points x_i in (E.16) turn out to be the zeros (roots) of the orthogonal polynomial of degree N appropriate for a given $w(x)$. For example, if $w(x) \equiv 1$, they are the zeros of the Legendre polynomial of order N and the integration limits are $(-1, 1)$, while for $w(x) = e^{-x}$ they are the zeros of the Laguerre polynomial of order N with the integration limits $(0, \infty)$. When $w(x) = e^{-x^2}$ they are the zeros of the Hermite polynomial of order N with $(a, b) = (-\infty, \infty)$, and for $w(x) = 1/\sqrt{1-x^2}$, those of the Chebyshev polynomials of order N with the interval of $(-1, 1)$. The abscissa points and the weights for these are commonly given in mathematical tables. Hence, Gaussian quadrature can be used to deal with quite a variety of integrals.

We will now sketch a proof of the theorem. The first step is to realize that given an arbitrary set of distinct abscissa points x_1, x_2, \dots, x_N , any polynomial $f(x)$

⁶For example, in Abramowitz and Stegun

of degree $N - 1$ can be expressed as

$$\begin{aligned} f(x) &= f(x_1) \frac{(x - x_2)(x - x_3) \cdots (x - x_N)}{(x_1 - x_2)(x_1 - x_3) \cdots (x_1 - x_N)} \\ &\quad + f(x_2) \frac{(x - x_1)(x - x_3) \cdots (x - x_N)}{(x_2 - x_1)(x_2 - x_3) \cdots (x_2 - x_N)} \\ &\quad + \dots + f(x_N) \frac{(x - x_1)(x - x_2) \cdots (x - x_{N-1})}{(x_N - x_1)(x_N - x_2) \cdots (x_N - x_{N-1})} \\ &= f(x_1)q_1(\mathbf{X}) + \dots + f(x_N)q_N(\mathbf{X}), \end{aligned} \quad (\text{E.17})$$

where $\mathbf{X} = (x_1, x_2, \dots, x_N)$ and (E.17) defines the $(N - 1)$ -degree polynomials $q_i(\mathbf{X})$. (Note that this observation is also the basis of so-called *Lagrange interpolation*.)

If we now choose the weights w_i by

$$w_i \equiv \int_a^b w(x) q_i(\mathbf{X}) dx, \quad (\text{E.18})$$

then (E.16) is exact for arbitrary \mathbf{X} and arbitrary $(N - 1)$ -degree polynomial $f(x)$. It only remains to consider those polynomials $f(x)$ of degree higher than $(N - 1)$ but still lower than or equal to $(2N - 1)$.

We next choose \mathbf{X} to be the N roots of the orthogonal polynomial $Q_N(x)$ of degree N , where the orthogonality is defined with the particular weight function $w(x)$ and associated integration limits. That is,

$$\int_a^b w(x) Q_M(x) Q_N(x) dx = 0, \quad M \neq N. \quad (\text{E.19})$$

As a corollary to (E.19), for any polynomial $R(x)$ of degree less than N one has

$$\int_a^b w(x) Q_N(x) R(x) dx = 0. \quad (\text{E.20})$$

Now for any polynomial $f(x)$ of order $N - 1 < M < 2N$, define a $(N - 1)$ -degree polynomial $p_{N-1}(x)$ which agrees with $f(x)$ at the chosen abscissa points \mathbf{X} . (This is always possible because there are N coefficients that can be adjusted in $p_{N-1}(x)$.) Then, the remainder $f(x) - p_{N-1}(x)$ must be a polynomial of order M which evaluates to zero at any abscissa point x_1, x_2, \dots, x_N . Thus we can write

$$\begin{aligned} f(x) &= p_{N-1}(x) + (x - x_1)(x - x_2) \cdots (x - x_N) R(x) \\ &= p_{N-1}(x) + Q_N(x) R(x), \end{aligned} \quad (\text{E.21})$$

where $R(x)$ must be a polynomial of order $M - N < N$. Now we integrate $w(x)f(x)$,

$$\begin{aligned} \int_a^b w(x)f(x) dx &= \int_a^b w(x)p_{N-1}(x) dx + \int_a^b w(x)Q_N(x)R(x) dx \\ &= w_1 f(x_1) + w_2 f(x_2) + \dots + w_N f(x_N), \end{aligned} \quad (\text{E.22})$$

where the second integral on the first line vanishes due to orthogonality (E.20), and the first one was already treated in the first two steps. This completes the proof of the theorem.

As an example, let us consider the following integral

$$I = \int_{-1}^1 \frac{x}{(a-x)^{3/2}\sqrt{1-x^2}} dx . \quad (\text{E.23})$$

This integral arises, e.g., in the calculation of the magnetic field due to a current loop at a point P off the axis of the loop. In this context, the value of the constant a , determined by the size of the loop and the location of P , is always greater than 1. For specificity, let us take $a = 5/4$ (corresponding, e.g., to a point in the plane of the loop but half the radius away from its axis). Since a factor $1/\sqrt{1-x^2}$ already appears in the integrand, let us try Gauss-Chebyshev quadrature which uses this weight function and the Chebyshev polynomials. For the 4-point quadrature, the abscissa points and weights are

$$\begin{aligned} x_1 &= -x_4 = \cos \pi/8 = 0.9238\dots, & x_2 &= -x_3 = \cos 3\pi/8 = 0.3826\dots, \\ w_1 &= w_2 = w_3 = w_4 = \pi/4 = 0.7853\dots . \end{aligned} \quad (\text{E.24})$$

The 4-point estimate is thus

$$I_4 = w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3) + w_4 f(x_4) = 3.897816\dots \quad (\text{E.25})$$

Successively increasing the order of calculations, we get $I_6 = 4.04160\dots$, $I_8 = 4.05259\dots$, $I_{10} = 4.05338\dots$, $I_{12} = 4.053435\dots$, $I_{14} = 4.053439\dots$, and I_{16} and higher yield at least 7-digit precision.

E.4 MONTE CARLO INTEGRATION

We round out this appendix by discussing the Monte Carlo integration method mentioned in Chapter 7 as an application of random processes. We assume that we can obtain random (actually *pseudo-random*) numbers uniformly distributed between some limits, say, 0 and 1 at will.⁷

Consider a square dart board that contains a circular region in its interior, as shown in Figure E.2. Assume that you throw darts at the board and are accurate enough that all of the darts hit the board, but with locations that are distributed randomly (and uniformly) over the surface of the board. The probability that a dart lands in any particular region is then proportional to the area of that region. Hence, the fraction of darts that land inside the circular region will be proportional to the ratio of the area of the circle to the area of the entire board.

This observation is the basis of the Monte Carlo method for evaluating an integral. Here the integral of interest is just the area of the circle in Figure E.2. To simulate this dart-throwing process, we use random numbers to select the places where the darts strike the board. Let us consider only the portion of the dart board

⁷The topic of how to generate such numbers has been discussed in Chapter 7 and more will be given in Appendix F

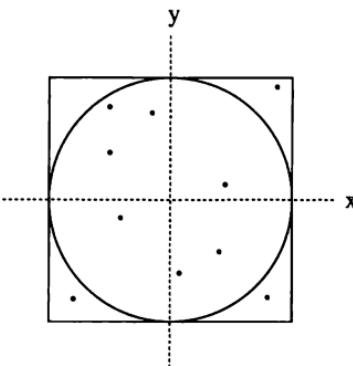


FIGURE E.2: Hypothetical square dart board, with an inscribed circle. The dots show where the first few darts might land.

that lies in the upper-right quadrant in Figure E.2, so that as far as our program is concerned the board lies in the range $0 \leq x \leq 1$, $0 \leq y \leq 1$. For our first dart, we pick values of x and y at random from this range. We then test to see if this point (x, y) satisfies the condition $y \leq \sqrt{1 - x^2}$, that is, if the point lies within the circle. This process is repeated for a large number of such points, N_{total} , and we calculate how many of the darts land within the (quarter) circle, N_{under} . The area under this curve is $N_{\text{under}}/N_{\text{total}}$ times the area of the square, and the area of the square is unity.

A subroutine which carries out this simulation is illustrated in pseudocode below.

EXAMPLE E.1 Monte Carlo integration routine

- Initialize a pseudo-random number generator, say, *rnd*. This often involves setting an initial *seed* integer argument to such a function (see Appendix F).
- Initialize the counter for hits: $N_{\text{under}} = 0$.
- Loop through a predetermined number N_{total} dart throws.
 - ▷ Get the x -component randomly: $x = \text{rnd}$.
 - ▷ Do the same for the y -component: $y = \text{rnd}$.
 - ▷ Check if point (x, y) is under the circular arc:
If $y \leq \sqrt{(1 - x^2)}$, then increment N_{under} .
- Calculate the estimated integral as $\text{Area} = N_{\text{under}}/N_{\text{total}}$.

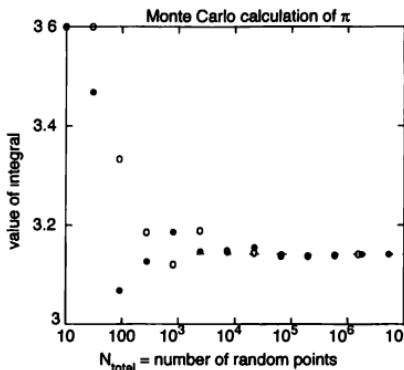


FIGURE E.3: Area of a circle with a radius of unity calculated with the Monte Carlo method, as a function of the number of random points used to compute the integral, N_{total} . The dotted line is the exact result, π . The filled and open circles show the results of two independent simulations

Figure E.3 shows some results for this integral as a function of the number of darts used in the simulation. Here we have multiplied the result by a factor of 4 to obtain the area of the entire circle, for which the exact answer is π . Figure E.3 shows the results for two different runs of our program. In each case we generated a grand total of $\sim 10^7$ random points and plotted the estimate for the area at several values of N_{total} as the calculation proceeded. The numerical result for the integral will, of course, exhibit statistical fluctuations; this is, after all, a random process.⁸ However, as N_{total} increased, the calculated area approached more and more closely to the exact result. The two different runs of the program produced slightly different results due to these statistical fluctuations, but both converged to the exact result as N_{total} became large. We will give more discussions on the statistical errors below.

The method described above can be adapted to calculate the area under any function, and you can also calculate volumes or hyper-volumes of almost any shape as long as there are easily implementable ways of checking if a randomly generated point lies inside or outside of the region of interest. However, we will leave such an exercise for interested readers, and now consider the problem of calculating areas in a little more detail.

The implementation introduced above in Example E.4 is conceptually straightforward, but not as efficient as one might like, for several reasons. First, generating each random point used in the one-dimensional integral requires two random num-

⁸A random process is amenable to various statistical tests, such as the χ^2 test which will be described in Appendix F

bers (one for the x -component and the other for the y -component), and second, we must compare each generated point with the function being integrated to judge whether it lies inside or outside of the area of interest. Both of these issues can be addressed if we realize that in computing the integral of $f(x)$ we are simply finding the average value of the function over the specified interval. This average can be computed by simply evaluating $f(x)$ at a collection of randomly chosen values of x , and then averaging the results. A simpler Monte Carlo integration implementation is thus

$$I \equiv \int_a^b f(x) dx \approx \frac{b-a}{N} \sum_{i=1}^N f(x_i), \quad (\text{E.26})$$

where x_i are the N points chosen randomly from the interval (a, b) with uniform probability. This approach can also be readily extended to higher dimensional integrals.

We now turn to the consideration of the order of errors in Monte Carlo integration. For the purpose of this discussion, let us think in terms of the dartboard method, Figure E.2. Thus, define a random variable f_j ($j = 1, 2, \dots, N$) which is 1 if the dart is under the curve and counted toward the integral, and 0 otherwise. The approximate integral is also be a random variable, say, $A \equiv A_0 \sum_{j=1}^N f_j/N$. That is, the estimate for the integral from a particular run of the Monte Carlo integration routine will generally fluctuate from other independent runs of the routine. As we have already hinted in Chapter 7, random processes often give rise to Gaussian distributions (see Reif [1965] for a nice discussion of this topic). In fact there is a powerful, general mathematical result called the central limit theorem which specifies when we should expect the statistical distribution of a random process to follow a Gaussian distribution. In our example the random variables f_j are independent and of finite mean and variance,⁹ exactly as needed for this theorem to apply to A as $N \rightarrow \infty$.

The values of A from a large number N_{trial} of independent trials will thus be normally (i.e., Gaussian) distributed with a half-width $\sigma = \sigma_0/\sqrt{N}$ where σ_0^2 is the variance of each f_j . The standard error of the mean from *one trial* of N points is $\sigma_0/\sqrt{N} = \sigma$, while the standard error of the mean for the distribution of N_{trial} independent estimates A is given by $\sigma/\sqrt{N_{\text{trial}}} = \sigma_0/\sqrt{N} \sqrt{N_{\text{trial}}}$. So, the uncertainty associated with an overall estimate of A is proportional to $1/\sqrt{N_{\text{Total}}}$, where N_{Total} is the total number of random numbers used and thus also proportional to the number of times the function is evaluated.

In comparison, the uncertainties involved in Newton-Cotes numerical integrations are typically much smaller for the same amount of computational effort. For example, the error of trapezoidal rule integration using a total of N panels in d dimensions scales as $N^{-2/d}$, while that of the Simpson's rule scales as $N^{-4/d}$.¹⁰ Thus rectangular panel, trapezoidal rule, and Simpson's rule integration become of

⁹The variance σ^2 of a random variable f is a measure of the spread of its values around the mean, and is defined by $\sigma^2 = \langle f^2 \rangle - \langle f \rangle^2$, where $\langle \cdot \rangle$ denotes the mean

¹⁰This assumes that the multidimensional integral is performed sequentially over each dimension, which picks up a relative error of order $N^{1/d}$ (number of panels in each dimension) raised to power -4 (Simpson's) and -2 (trapezoidal), respectively, at each stage.

less and less use in higher dimensions. In fact, even Simpson's rule becomes less competitive than Monte Carlo integration above $d = 8$ dimensions. The strength of Monte Carlo integration is in dealing with multidimensional integrals. Note that the above discussion of the Monte Carlo method does not depend on the dimensionality of the integral; in any dimension its error is of order $1/\sqrt{N_{total}}$. In contrast, the errors in Newton-Cotes integration approaches all increase with dimension. For example, the error of trapezoidal rule integration using a total of N panels in d dimensions scales as $N^{1-3/d}$, while that of the Simpson's rule scales as $N^{1-5/d}$. According to this qualitative argument, rectangular panel, trapezoidal rule, and Simpson's rule integration become of no use in two, three, and five dimensions or higher, respectively. In fact, even Simpson's rule becomes less competitive than Monte Carlo integration in $d = 4$ dimensions. This may be critical in some applications, such as those in statistical mechanics or quantum field theory, where integrations over a very high dimensional phase space are commonly required.

EXERCISES

- E.1.** Show that (E.10) does indeed give the area under a parabolic fit to the three points $[x_i, f(x_i)]$, $[x_{i+1}, f(x_{i+1})]$, $[x_{i+2}, f(x_{i+2})]$.
- E.2.** Calculate the period of a non-linear oscillator described by

$$\frac{d^2\theta}{dt^2} = - \sin \theta \quad (\text{E.27})$$

by numerically integrating

$$\sqrt{8} \int_0^{\theta_m} \frac{d\theta}{\sqrt{\cos \theta - \cos \theta_m}} \quad (\text{E.28})$$

for several values of the maximum angle θ_m , using the trapezoidal rule, Simpson's rule, or the Romberg integration method.

- E.3.** From (E.7) - (E.9), let us write

$$I \equiv \int_a^b f(x)dx = R_{4,1} + c_1(\Delta x)^2 + c_2(\Delta x)^4 + \dots, \quad (\text{E.29})$$

where the omitted terms include only even powers of Δx and $R_{4,1}$ is the trapezoidal rule approximation to I with 2^{1-1} panels [or with $\Delta x = (b-a)/2^{1-1}$].

- (a) Write a similar expression involving $R_{4+1,1}$ and calculate c_1 , and thus obtain the next order result $R_{4+1,2}$ in terms of $R_{4,1}$ and $R_{4+1,1}$.
 (b) Continue this one more time and obtain $R_{4+1,3}$ in terms of $R_{4+1,2}$ and $R_{4,2}$. Generalize this to obtain the Richardson's formula

$$R_{4+1,j+1} = R_{4+1,j} + \frac{R_{4+1,j} - R_{4,j}}{4^j - 1}. \quad (\text{E.30})$$

- E.4.** There is a current loop of radius R centered at origin and lying in the $x-y$ plane, carrying current I . Obtain the magnetic field due to this current at a point $P(x, 0, z)$ using (a) a Newton-Cotes method (trapezoidal rule, Simpson's rule, etc.), and (b) a Gaussian quadrature method.

- E.5.** Evaluate the area under a circle using Simpson's rule and compare it with the Monte Carlo method. Show that Simpson's rule is more efficient (in terms of computer time) in this case.
- E.6.** Use the Monte Carlo method to compute the transcendental number e . *Hint:* Consider the integral of $1/x$ from $x = 1$ to 10.
- E.7.** Use the Monte Carlo method to compute the volume of a sphere. Be sure to compare your result with the exact answer.

REFERENCES

- [1] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapter 4 discusses many aspects of numerical integration.
- [2] G. B. Arfken, and H. J. Weber, 2001, *Mathematical Methods for Physicists*, Academic Press, San Diego.
- [3] M. Abramowitz and I. A. Stegun, Editors, 1964, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards, Washington, DC.
- [4] F. Reif, 1965, *Fundamentals of Statistical and Thermal Physics*. McGraw-Hill, New York. See Chapter 1.

Generation of Random Numbers

In Chapters 7 and 8, simulations of random processes and the Ising model were discussed extensively. Now that you believe in, or at least appreciate, the utility of stochastic simulations, we should address the following issue. A computer is, in some ways, the epitome of a deterministic system. How can we possibly hope to get a computer (program) to do anything random? The short answer to this is: with a random number generator. All computer languages¹ contain a mechanism, usually a built-in function, for producing a sequence of random numbers; these should really be termed *pseudorandom* numbers, as we will explain shortly. Typically, once initialized (or *seeded*) in some way, a random number generator returns a different random number each time it is used in a given sequence so that they are distributed uniformly in a range (that is, all values in this range are equally likely). This range is often $(0 - 1)$, though some languages contain functions with different ranges.² In this appendix we examine how the computer language (compiler, interpreter, or whatever) actually generates the random numbers.

F.1 LINEAR CONGRUENTIAL GENERATORS

There are several different numerical schemes for generating sequences of pseudorandom numbers. By far the most common scheme is the *linear congruential generator*, which employs an equation of the form

$$x_{n+1} = (a x_n + b) \bmod m, \quad (\text{F.1})$$

where x_n , a , and b are all integers. The arithmetic here is performed modulo m , with m typically 2^{16} or 2^{32} , so that only the lowest 16 or 32 bits are kept in the calculation of x_{n+1} . Here x_n is the n th random integer in the sequence, and usually the resulting x_{n+1} is normalized so that a number between 0 and 1 is returned by the function for use. The values for the constants a and b must be carefully chosen; otherwise disastrous consequences may follow for the quality of random numbers. We will discuss what we mean by the term “quality” and how to measure it shortly.

Such a generator works as follows. First, an initial random integer x_0 , called the seed, is selected. The value of the seed is usually either chosen explicitly by the user, assigned a default (fixed) value by the function, or automatically selected using some sort of always changing quantity (such as the time-of-day clock). If you (or the function) selects a fixed seed, you will get the same sequence of random numbers

¹At least every one that the authors have encountered

²Occasionally, we need random numbers with a nonuniform distribution. Later in this Appendix we discuss some ways to use uniform random numbers to construct those which follow a desired nonuniform distribution

each time you run your program. It is probably not fair to call this particular sequence random since it is obtained repeatedly, and you would generally not want to use it in your simulations. However, it is very useful in the debugging process since you can use it as you search for programming errors. Such errors are easier to find if they are reproducible! Once you are convinced that your program works correctly, you will usually want to use a different choice for the seed each time you run your program. The sequence of x_n is kept either explicitly as an input/output variable of the function or internally by the function and fed for generating the next random number x_{n+1} implicitly.

Although most readers will likely use an already existing generator rather than programming their own, the issue of the generator's quality is quite relevant for ensuring the accuracy of simulation results. After all, if you wish to perform any serious calculation, would you blindly trust a "black box"? Clearly, we all wish to have a fast generator which produces random numbers of high quality. There are many aspects to the quality of random numbers, but some of the most obvious ones include the length of its period (i.e., how many numbers are generated before the sequence repeats itself), accuracy (the resolution of the final values produced between 0 and 1), the lack of correlation (when generating several numbers sequentially, how "different" they are from each other), and of course, uniformity (how uniform is the distribution of the generated numbers). Some of these criteria are not independent; for a linear congruential generator, each number occurs just once in an entire period, and thus the issue of the period length and resolution are the same. For these and other measures, the values of m , a , and b in (F.1) play crucial roles.³

For computers that use binary arithmetic (i.e., almost all of today's computers), there is a great advantage in using the machine's word size as the number of bits in m (i.e., $m = 2^{32}$ for most machines). This is because integer computations modulo the machine's word size amounts to truncating the high-order bits in integer overflows, which is typically performed automatically by the system in a very efficient manner. Thus, most linear congruential generators used today have $m = 2^{32}$. But this choice alone does not ensure that all 32-bit integers between -2^{31} and $2^{31} - 1$ occur in a full period of a generator. This point is clear since, e.g., if a is even, x_{n+1} is always even or odd depending on whether b is even or odd, respectively, reducing the period by at least a factor of two. Worse yet, if $b = 0$, once x_{n+1} evaluates to zero (modulo m), all subsequent numbers also evaluate to zero. Fortunately, the mathematics of (F.1) is sufficiently simple that many of its properties have been worked out in great detail.⁴ It turns out that choosing a

³For extensive uses of random numbers in, say, a research work using large-scale Monte Carlo simulations, additional considerations may also apply. For example, those generators that are available in a source code form are more easily portable among environments using different compilers and libraries and thus preferable. Also since the period of a linear congruential generator is limited by m , we need a different type of generator if the number of random numbers required is of the order of or greater than this limit. There are a number of algorithms available which produce far larger periods and some of them are described in Knuth and in Press et al. An excellent overall summary of other algorithms can be found in *Handbook on Simulation* (ed. Banks).

⁴This is one of the advantages of the linear congruential generators. Another big advantage is the very low computer resources it requires.

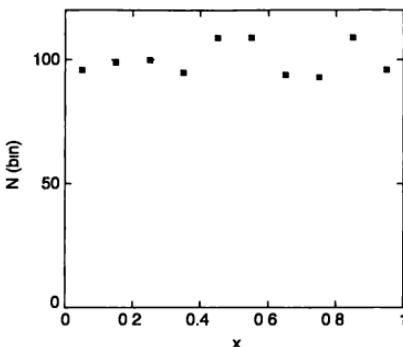


FIGURE F.1: Measured distribution of $N = 1000$ random numbers generated using the *True Basic* `rnd` function. The range 0–1 was divided into 10 bins, and the number of values that fell into each bin was tabulated. Each bin had a width of $\Delta x = 0.1$, and the occupancy was plotted at the value of x corresponding to the center of the bin. That is, the occupancy of the bin that covers $x = 0\text{--}0.1$ was plotted with an abscissa $x = 0.05$, etc. The dotted line shows the ideal (theoretical) value for each bin, $N/10 = 100$.

which satisfies $a \bmod 8 = 5$ together with $b = 1$ ensures that all m integers are produced in a period, independent of the choice of seed x_0 . In addition, a should be large enough to cause an integer overflow condition in (F.1) for most x_n (so that the less random high-order bits are truncated in producing x_{n+1}). The issues of correlations and (subset) uniformity are a bit trickier, and often the only way to ensure good quality is to test them empirically. Some standard tests are introduced and applied to various trial values of a by Knuth (see the references). According to those considerations, Knuth shows that “haphazard” values such as $a = 1566083941$ or 1812433253 pass all of these tests.

It is of course not hard to come up with our own versions of tests for some aspects of these issues. For example, suppose that we generate a sequence of N numbers and want to determine if they are indeed distributed uniformly, as advertised. One way to test for uniformity is to divide the allowed range into bins and compute the number of values that fall into each bin. An example of such a test is shown in Figure F.1. Here we have generated a sequence of $N = 1000$ random numbers using a linear congruential generator `rnd` (built into the *True Basic* language) and calculated how they are distributed among 10 bins that encompass the range 0–1. If the numbers are uniformly distributed, the average number that falls into each bin should be $N/10 = 100$, and we see that the bin occupancies do indeed cluster around this value, the dotted line in Figure F.1. Thus, at a crude level the `rnd` generator seems satisfactory. However, there are fluctuations about the ideal value for each bin and these fluctuations need to be analyzed for deeper understanding of the randomness. This latter topic will be discussed in Appendix G.

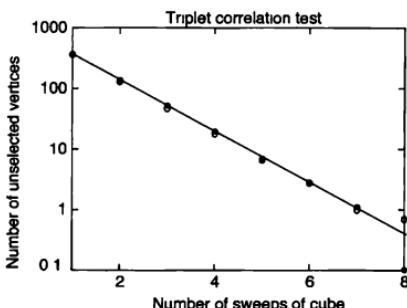


FIGURE F.2: $N = 3000$ random numbers generated using the *True Basic* `rnd` function were used to select vertices of a $10 \times 10 \times 10$ cube in each "sweep". This was repeated 8 times and the results compared with the theoretical expectation (solid line). This procedure was carried out twice, producing thus the open and closed symbols

A useful test for correlations⁵ involves successively generated triplets of random integers⁶ between 1 and L . Each such triplet (y_1, y_2, y_3) is identified as the coordinates of a vertex of a cube of size $N = L^3$, and we can test how quickly vertices of the cube are selected as many such triplets are generated. If there are no correlations among the coordinates of each triplet and they are completely random, then the probability that a given vertex remains unselected after N triplets have been generated is $p = (1 - 1/N)^N \approx e^{-1}$. Thus the theoretically expected number of still unselected sites after n "sweeps" (where each sweep consists of N selections) is equal to $N \times e^{-n}$. On the other hand, if some correlations exist within each triplet, then the selected sites may cluster in some regions of the cube (such as along a diagonal) and the number of unselected sites will deviate from this theoretical expectation. Figure F.2 gives a comparison of the number of unselected vertices with this theoretical value for *True Basic's* `rnd` function. For each of 8 successive sweeps of the $10 \times 10 \times 10$ cube, 10 independent runs are averaged over to produce the points plotted. The open and closed symbols show the results of performing this procedure twice. From Fig. F.2 it seems safe to say that `rnd` passed this triplet correlation test at least up to 7 sweeps. However, a suspicious deviation is apparent at sweep 8. A deeper analysis would require the testing of such deviations.

The reason why the sequence of numbers obtained from (F.1) behaves randomly is also very interesting. In Chapter 3 we discussed a chaotic model called the logistic map. Comparing (F.1) with the logistic equation (3.22), we see that they are very similar in form. It is thus not hard to believe that (F.1) is a chaotic system! The analogy to chaos can also be appreciated from the following argument

⁵Described originally by Dietrich Stauffer

⁶So instead of normalizing x_{n+1} in (F.1) to the range of 0 to 1, it is normalized and truncated to an integer in the range between 1 and L

involving the nonlinear pendulum studied in Chapter 3. Suppose that we perform a simulation of such a pendulum in the chaotic regime. We learned in Chapter 3 that in this case the behavior of $\theta(t)$ is a wildly varying function. If we choose the pendulum parameters to make the behavior as chaotic as possible,⁷ the value of θ at evenly spaced intervals⁸ would be a good candidate for a random sequence of numbers in the range $-\pi$ to π . Choosing the optimum pendulum parameters would not be trivial—this is also why choosing a and b in (F.1) is important—but this illustrates how a chaotic (yet deterministic) system can yield a sequence of pseudorandom numbers. The term “pseudorandom” is often used, since the sequence of numbers obtained from (F.1), or equations like it, is deterministic.

F.2 NONUNIFORM RANDOM NUMBERS

As we mentioned earlier, random number generators that are built into computer languages, such as `rnd` in True Basic or `rand()` in C, usually yield numbers that are uniformly distributed over some range. Nearly all of our stochastic simulations will involve random numbers that are uniformly distributed, so these functions are just what we need in most cases. However, there are instances in which random numbers that are described by a *nonuniform* distribution are required. For example, radioactive decay is characterized by a Poisson distribution, while the distribution of velocities in an ideal gas follows a Maxwell distribution. Let us now consider how we can generate random numbers that follow such a specific nonuniform distribution.

Here we describe two useful methods for accomplishing this. The first approach, which is known as the *transformation method*, makes use of a fundamental property of probabilities. Consider a collection of variables $\{x_1, x_2, \dots\}$ that are distributed according to the function $P_x(x)$. That is, the probability of finding a value that lies between x and $x + dx$ is $P_x(x)dx$. If y is some function of x , then

$$|P_x(x) dx| = |P_y(y) dy|, \quad (\text{F.2})$$

where $P_y(y)$ is the probability distribution that describes the collection $\{y_1, y_2, \dots\}$. In our case the x variables will be generated by `rnd`, or by a similar generator that yields uniformly distributed numbers, so $P_x = \text{constant} \equiv C$. We thus have

$$P_y(y) = P_x(x) \left| \frac{dx}{dy} \right| = C \left| \frac{dx}{dy} \right|. \quad (\text{F.3})$$

In order to obtain a sequence characterized by a distribution P_y , we must be able to evaluate the function $y = f(x)$, whose derivative is $|dy/dx|^{-1} = P_y$.

To see how this works in practice we consider the Poisson distribution, $P_y(y) = \exp(-y)$. The transformation function is then $y = f(x) = -\ln(x)$, as you can verify by evaluating the derivative dx/dy . The procedure is then to take a sequence of random numbers, $\{x_1, x_2, \dots\}$, which follow a uniform distribution (as produced by

⁷The careful reader might complain that based on what we learned about chaotic systems in Chapter 3, the phrase “as chaotic as possible” is not very well defined. However, this notion can be made fairly precise using the concept of entropy introduced in the exercises in Section 7.7. See also the discussion of entropy in Baker and Gollub (1990).

⁸It would probably be best to choose the intervals to be different from the drive period

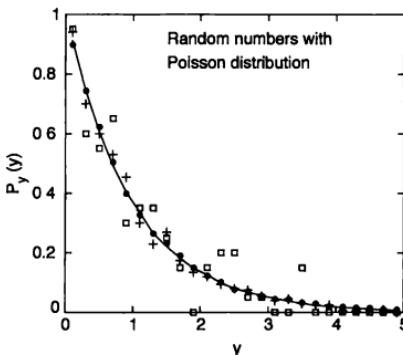


FIGURE F.3: Distributions of random numbers produced with the transformation method. The distribution was designed to have a Poisson form. Open squares distribution calculated with the first 100 numbers, pluses with 1000 numbers, filled circles with 10^4 numbers, smooth curve with 10^6 numbers. On this scale, the smooth curve is indistinguishable from the Poisson distribution $P_y = \exp(-y)$. These distributions were obtained by first generating a sequence of pseudorandom numbers containing the stated number of entries, and then constructing a histogram of the number of entries as a function of y . The histogram bins were of size $\Delta y = 0.2$.

`rnd`), and calculate $y_i = -\ln(x_i)$. The resulting sequence of numbers $\{y_1, y_2, \dots\}$ should obey the Poisson distribution. That this is indeed the case is illustrated in Figure F.3, which shows the distribution $P_y(y)$ for sequences of different lengths. For example, the squares show the distribution calculated for a sequence containing 100 values. The result follows the general form of the Poisson distribution (the solid curve), although there is some statistical scatter, which we will discuss in detail in Appendix G using the so-called χ^2 -test. As the length of the sequence is increased, that is, as more y values are used, the results converge to the Poisson form.

The transformation method is useful when the function $f(x)$ can be easily evaluated. However, there are cases when the desired distribution may not be known in analytic form.⁹ Such problems can be handled with an algorithm known as the *rejection method*. We first use the `rnd` generator (or one like it) to produce a sequence of numbers $\{y_1, y_2, \dots\}$ distributed uniformly in the range of interest, y_{\min} to y_{\max} . Now, suppose that our goal is to produce a sequence of numbers that are distributed according to the function P_y in Figure F.4. We then proceed through the sequence $\{y_1, y_2, \dots\}$ and accept entries with a probability proportional to P_y . That is, we start with y_1 and evaluate $P_y(y_1)$. A new random number p_{test} is then generated (with `rnd`, for example), which is distributed uniformly in the range 0 to $P_y(\max)$ where $P_y(\max)$ is the maximum value of P_y . If $P_y(y_1) < p_{\text{test}}$, we remove

⁹Or the transformation function may not be easily evaluated, or even expressible, in closed form

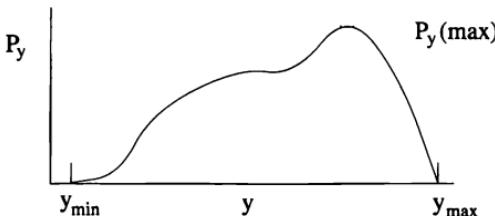


FIGURE F.4: Schematic distribution function for use with the rejection method. The probability of obtaining a particular value of y is proportional to $P_y(y)$.

y_1 from the sequence; otherwise it is kept. This process is repeated with y_2, y_3, \dots . The numbers that remain in the end will be distributed according to P_y . In terms of Figure F.4, the probability that a number y_n passes this test is proportional to the “height” of the function $P_y(y_n)$.

An interesting feature of the rejection method is that it requires only that the function P_y be evaluated. This evaluation may be analytic or numerical, so the distribution function can be very irregular in form, as was intended with the illustration in Figure F.4. In addition, both the generation of the original sequence and the rejection test require random numbers distributed uniformly, which are easily obtained with `rnd` (or a similar generator).

To demonstrate the rejection algorithm we have used it to generate numbers distributed according to the Gaussian distribution

$$P_y = B \exp [-(y - y_c)^2 / \sigma^2], \quad (\text{F.4})$$

where the constant B is chosen to make P_y properly normalized.¹⁰ This distribution is centered at y_c and has a half width $\sigma/\sqrt{2}$. Figure F.5 shows the calculated distributions obtained with sequences of varying lengths. The shorter sequences exhibit the intuitively expected statistical fluctuations, while the longer sequences converge accurately to the desired Gaussian form.

EXERCISES

- F.1. Use the rejection method to generate a sequence of random numbers distributed according to $P_y(y) = 1/y$.
- F.2. Use the transformation method to generate a sequence of random numbers distributed according to $P_y(y) = 1/y^2$.
- F.3. Perform some statistical tests of your random-number generator, such as:
 - Generate a long sequence of random numbers and calculate the average of the squares of these numbers. Compare your result with the ideal value.
 - A bin test as in Figure F.1.

¹⁰Such normalization is actually not required by the rejection algorithm. In fact, it is usually convenient to choose the maximum value of P_y to be unity.

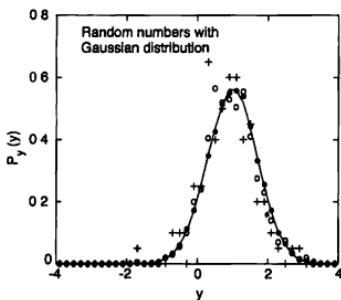


FIGURE F.5: Distributions of random numbers obtained with the rejection method. The distribution was designed to have a Gaussian form, (F.4), centered at $y = 1$ with a width of unity. Pluses distribution calculated with the first 100 numbers; open circles with 1000 numbers; filled circles with 10^4 numbers; smooth curve with 10^5 numbers. On the scale used here this curve is indistinguishable from (F.4). These distributions were constructed in the same manner as those in Figure F.3.

- A triplet (or quadruplet or whatever) correlation test such as in Figure F.2.
- Devise your own tests.

- F.4. Design and test a random-number generator based on the chaotic behavior of the nonlinear pendulum model studied in Chapter 3. Use the value of θ at regular (or irregular) intervals to determine the value of the random number. For convenience, scale the values of θ to obtain numbers n in the range 0–1. Then calculate the averages $\langle n \rangle$ and $\langle n^2 \rangle$. You should also do a bin test like the one in Figure F.1. Study the performance of your generator for different values of the driving force. Be sure to pick values of the drive that place the system in the chaotic regime.
- F.5. Repeat the previous problem, but use the chaotic billiards of Chapter 3 to generate the random numbers.

REFERENCES

- [1] J. Banks, Ed., 1998, *Handbook on Simulation*, Wiley, New York. See especially Chapter 4, written by P. L'Ecuyer, *Random Number Generation*.
- [2] D. Knuth, 1997, *The Art of Computer Programming*, 2, 3rd ed., Addison-Wesley, Redwood City. A volume in a series which constitutes a classic treatise on computer programming. This volume contains an extensive discussion on the measure of goodness of pseudorandom number generators.
- [3] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. This book discusses the generation of random numbers along with numerous statistical tests.

Statistical Tests of Hypotheses

In our work on the generation of pseudorandom numbers in Appendix F we considered two of the tests of the quality of these numbers, and we encountered possible deviations from theoretical expectations based on the assumption of complete randomness. Similarly, in our examples involving the transformation and rejection methods for nonuniform pseudorandom numbers, we noted the statistical fluctuations associated with finite sequences of random numbers. One encounters such fluctuations often in dealing with stochastic processes, including any physical measurements (and not just with random number generators), so it is important to consider them in a little more detail. For example, how can we tell if these fluctuations are too large or too small? That is, how do we know that they are just the expected statistical fluctuations?

This question¹ is a central issue in statistics and is discussed at great length in many textbooks (see, for example, Press et al. [1986]). The standard statistical test for determining if an observed distribution is consistent with a model (that is, theoretical) distribution is based on what is known as the *chi-square statistic*. This is defined by first separating the observed values into suitable bins, and then comparing the number of times the measured value falls into each bin to the statistical expectations. A quantity called χ^2 is defined by

$$\chi^2 = \sum_i \frac{(N_i - n_{\text{ideal}})^2}{n_{\text{ideal}}} . \quad (\text{G.1})$$

Here N_i is the number of events that are *measured* to fall into bin i , and n_{ideal} is the number of events that the theoretical model *predicts* should fall into each bin.² The quantity χ^2 is thus a quantitative measure of how much the observed distribution differs from the theoretical one. We should not expect this difference to be zero, since our intuition tells us that there will always be some fluctuations for such a stochastic problem. On the other hand, if the fluctuations are extremely large, we should be suspicious!

It turns out that statisticians have calculated the probability of finding a particular value of χ^2 , assuming that the process involves a large number of independent random variables. The resulting probability distribution is a generalized form of the so-called normal distribution. Here the term *normal* refers to an underlying Gaussian process which generally arises in connection with random processes. In order to sketch the connection between our χ^2 and the normal distribution, we

¹Note that the present issue of quantifying how consistent the measured data are with a theoretical hypothesis is rather different from fitting the data to a theoretical formula to estimate best numerical values for the unknown parameters in the formula.

²These theoretical values could vary from bin to bin, but we will ignore that (largely notational) complication here.

will now digress with a bit of mathematics; if you are not interested in how the connection comes about, you may wish to skip the following section and go directly to Section G.2.

G.1 CENTRAL LIMIT THEOREM AND THE χ^2 DISTRIBUTION

A normal distribution arises in virtually all processes in which a large number of independent random values are involved. This distribution has a probability density of the form

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2}, \quad (\text{G.2})$$

where $f(x)dx$ is the probability that x lies between x and $x+dx$. It is not difficult to see that μ is the mean $\langle x \rangle$, while σ^2 is the variance or the squared fluctuation. Thus, $\sigma = ((x - \langle x \rangle)^2)^{1/2}$ is the standard deviation.

The fundamental mathematical result we need is the central limit theorem. We referred to this theorem briefly in Chapter 7 in connection with random processes in general. Simply put, the theorem states the following: if we have independent, identically distributed random variables y_j ($j = 1, 2, 3, \dots, N$) with a finite mean μ_0 and variance σ_0^2 , then the distribution of its sum $x[N] \equiv \sum_{j=1}^N y_j$ approaches a normal distribution for large N , with mean $\mu = N\mu_0$ and variance $\sigma^2 = N\sigma_0^2$. Mathematically curious readers can read about the rigorous statement of this theorem as well as its proof in many statistics texts (such as Feller [1968]). Here we accept this “simple” statement without proof. The central limit theorem explains why a normal distribution is found in virtually all processes made up from a large number of constituent random processes that are independent of each other. This applies, e.g., to the random walks, molecules in a gas, and the collection of a large number of random numbers generated by a computer.

The normal distribution (G.2) is for a scalar variable x , i.e., in one dimension. We can generalize it to d dimensions, where the random variable $\vec{r} = (x_1, x_2, \dots, x_d)$ has d independent components x_1, \dots, x_d . Taking $\langle \vec{r} \rangle = (0, 0, \dots)$ for simplicity, the d -dimensional normal distribution would be

$$f_d(\vec{r}) = \frac{1}{(2\pi\sigma^2)^{d/2}} e^{-\vec{r}^2/2\sigma^2}, \quad (\text{G.3})$$

where σ^2 is the variance of each component. It is then not difficult to imagine an extension of the central limit theorem where (G.3) is the limiting probability density (in the d -dimensional space) for $\vec{r}[N]$ when it is the sum of a large number N of independent d -dimensional random vectors \vec{y}_j ($j = 1, 2, \dots, N$):

$$\vec{r}[N] = \sum_{j=1}^N \vec{y}_j. \quad (\text{G.4})$$

If we are interested in the probability density for the squared modulus $r[N]^2$ instead of that for $\vec{r}[N]$, we need to integrate $f_d(\vec{r})$ over all the angles in d -dimensions. This leads to

$$f_d(r^2) = \frac{1}{\sigma^d 2 \Gamma(d/2)} \left(\frac{r^2}{2}\right)^{d/2-1} e^{-r^2/2\sigma^2}, \quad (\text{G.5})$$

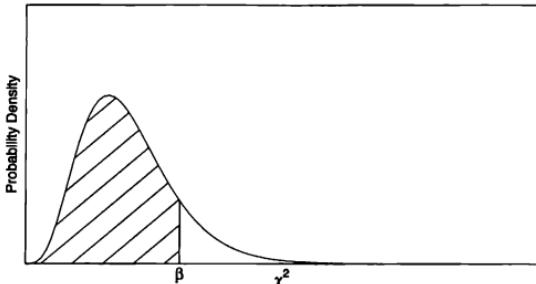
χ^2 Probability Distribution

FIGURE G.1: The probability density of the χ^2 distribution is illustrated above. The shaded area corresponds to the cumulative probability that χ^2 falls somewhere between 0 and β . Sometimes you will find tables of values corresponding to its complement, i.e., the unshaded area which is equal to the probability that χ^2 is larger than β .

in the space of $r^2 \in [0, \infty)$ where $\Gamma(x)$ is the gamma function. If $\sigma^2 = 1$, and $d = \nu$, we can rewrite this distribution in the form of

$$f_\nu(\chi^2) = \frac{1}{2\Gamma(\nu/2)} \left(\frac{\chi^2}{2} \right)^{\nu/2-1} e^{-\chi^2/2}, \quad (\text{G.6})$$

where χ^2 stands for r^2 in (G.5). This is the probability density of the χ^2 distribution of ν degrees of freedom. The χ^2 distribution is often quoted in terms of its cumulative values $\int_0^{\chi^2} f_\nu(x) dx$, or its complement $\int_{\chi^2}^{\infty} f_\nu(x) dx$. By a change of variables, the former can be expressed as

$$\int_0^{\chi^2} f_\nu(u) du = \frac{1}{\Gamma(\nu/2)} \int_0^{\chi^2/2} e^{-u} u^{\nu/2-1} du = P(\nu/2, \chi^2/2), \quad (\text{G.7})$$

where $P(a, x)$ is known as the incomplete gamma function.

If not all the d components of the individual random vectors \vec{y}_j are independent but there is an overall constraint among their components common to all \vec{y}_j , the appropriate limiting distribution is (G.6) with $\nu = d - 1$ degrees of freedom.³ This gives the general connection between independent random processes and the χ^2 distribution. In fact, the χ^2 distribution of ν degrees of freedom is nothing but the normal distribution in $\nu + 1$ dimensions with an overall constraint.

Now that we understand the character of the so-called χ^2 distribution, let us consider how it relates to hypothesis testing as mentioned at the beginning of this appendix. We assume that each independent measurement of a quantity produces a number in some range. We divide this range into d bins and define

³If there are additional constraints, the number of degrees of freedom must be further reduced.

for the j -th measurement a vector random variable \vec{z}_j of d -components so that its i -th component is 1 if the measurement falls in bin i and 0 otherwise. Since these components obey the constraint that their sum is equal to 1 (because any measurement produces a number in the range of allowed values), the sum of many such variables, which constitutes many independent measurements, generally falls within the purview of the $(d - 1)$ -dimensional normal distribution. To be specific, we further define another vector random variable \vec{y}_j whose i -th component is given by

$$(\vec{y}_j)_i \equiv \frac{(\vec{z}_j)_i - \langle (\vec{z}_j)_i \rangle}{\sqrt{N \langle (\vec{z}_j)_i \rangle}}. \quad (\text{G.8})$$

Then the mean of each component of \vec{y}_j is zero ($\langle (\vec{y}_j)_i \rangle = 0$) and its variance is $1/N$. Thus, the premise of the χ^2 distribution of $d - 1$ degrees of freedom applies⁴ and the quantity $\chi^2 = |\sum_{j=1}^N \vec{y}_j|^2$ is distributed according to the χ^2 distribution (G.6) with $\nu = d - 1$. Moreover, we can rewrite χ^2 as

$$\begin{aligned} \chi^2 &= \sum_{i=1}^d \left[\sum_{j=1}^N (\vec{y}_j)_i \right]^2 \\ &= \sum_{i=1}^d \left[\frac{\left(\sum_{j=1}^N (\vec{z}_j)_i - \sum_{j=1}^N \langle (\vec{z}_j)_i \rangle \right)^2}{\sum_{j=1}^N \langle (\vec{z}_j)_i \rangle} \right]. \end{aligned} \quad (\text{G.9})$$

Finally, we can identify $\sum_{j=1}^N (\vec{z}_j)_i$ to be the random variable which counts how many of the N measurements fall in bin i and $\sum_{j=1}^N \langle (\vec{z}_j)_i \rangle$ as its theoretical expectation. This completes our “physicist’s” derivation of how the quantity χ^2 defined in (G.1) obeys the χ^2 distribution (G.6) with $\nu = d - 1$.

G.2 χ^2 TEST OF A HYPOTHESIS

As seen in the previous section, the value of χ^2 for a set of independent measurements divided into $\nu + 1$ bins will be distributed according to the χ^2 distribution of ν degrees of freedom, $f_\nu(\chi^2)$, given in (G.6) and sketched in Figure G.1. Often this probability is expressed and tabulated in terms of the cumulative value between 0 and some limit β

$$P(\nu/2, \beta/2) = \int_0^\beta f_\nu(x^2) d(x^2). \quad (\text{G.10})$$

This corresponds to the probability that the observed $\chi^2 \leq \beta$. There is unfortunately no convenient closed-form analytic expression for the function $P(a, x)$. The behavior of $P(a, x)$ for several values of a is shown in Figure G.2. This probability function involves two parameters; one of them is proportional to the maximum

⁴If the expectation values have to be replaced by their estimates calculated from the sample measurements themselves, then the number of degrees of freedom is reduced further. If r parameters must be estimated from the measurements to obtain the expectation values, then the appropriate number of degrees of freedom will be $d - 1 - r$.

value of χ^2 ($x = \beta/2$), while the other is related to the number of degrees of freedom in the problem, ν ($a = \nu/2$).

A typical use of the χ^2 -distribution is in testing of the validity of a hypothesis. We predetermine a significance level α (0.05 or 5% is typical), and ask whether a given set of measurements reject the hypothesis at this level or not. To do this, we first look for the *critical values* of χ^2 such that the probabilities of χ^2 to fall below the lower critical value or above the upper critical value are each $\alpha/2$. That is, the lower critical value is the value of β where the cumulative probability $P(\nu/2, \beta/2) = \alpha/2$, and the upper critical value is the value of β where $P(\nu/2, \beta/2) = 1 - \alpha/2$. If the actual value of χ^2 calculated from the measurements and the theoretically expected values falls outside these two limits,⁵ then we must reject the hypothesis at the level of α . Thus, if χ^2 were very large or very small, we reject the original hypothesis at a small significance level α . Of course, even if the hypothesis were true, there would still be some probability α that the measured χ^2 would fall outside the critical values. So α can be considered a factor of risk that you are taking by rejecting the hypothesis. For example, for $\alpha = 0.05$ with 10 degrees of freedom, the lower critical value is about 3.247 and the upper critical value is about 20.48 according to common tables.⁶ So, if your 11-bin test gives a value of χ^2 less than 3.247 or greater than 20.48, you will reject the hypothesis which led to this value of χ^2 at the level of $\alpha = 0.05$, taking a 5% chance of rejecting a correct theory, or with 95% confidence. If your actual value of χ^2 is much further outside of the 0.05 critical values, you could push α to smaller values and reject the theory with even greater confidence.

The *bin test* described here is equivalent to the test of the variance often discussed in statistics texts. For such a test, χ^2 is usually defined as

$$\chi^2 = \nu(s/\sigma_0)^2, \quad (\text{G.11})$$

where s is the standard deviation measured from $\nu + 1$ samples and σ_0 is the theoretically expected standard deviation (for the entire population). Here we test the hypothesis that the standard deviation is indeed as the theory predicts by comparing this χ^2 with the critical values determined similarly as before.

In Appendix F we tested random numbers created by the True Basic function `rnd` by counting how many random numbers fall in bins of width 0.1. We can now test the hypothesis that these random numbers are uniformly distributed by using the χ^2 test just described. Suppose that a sample of $N = 1000$ random numbers distributed in 10 bins results in a χ^2 of 11. This value falls just below the upper critical value for the significance level of $\alpha = 0.5$. Thus we reject the hypothesis at this high level of risk, or in other words, we have very little confidence in rejecting the hypothesis, based on this one sample. Actually, with larger samples we will obtain values of χ^2 which are even closer to the critical value for $\alpha = 1$, making it virtually impossible to reject the hypothesis.

An even more stringent test of the significance of a hypothesis is to actually produce the entire measured distribution of χ^2 from many independent set of ex-

⁵This is the so-called equal-tails test. Other tests such as a one-sided one are also used.

⁶Such as in Abramowitz and Stegun in the references.

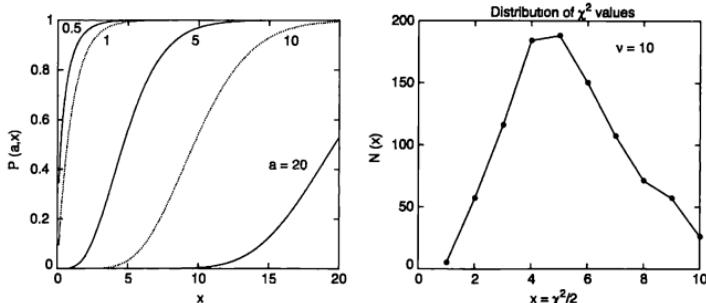


FIGURE G.2: Left: incomplete gamma function, $P(a, x)$. For the χ^2 test, $x = \chi^2/2$ and $a = \nu/2$, where ν is the number of degrees of freedom. Right: distribution of χ^2 values found by performing 1000 bin tests, as in Figure F 1. Here we used 11 bins so that $\nu = 10$ ($a = 5$), and each sequence contained 1100 numbers generated using `rnd`. This distribution was constructed in a manner similar to that employed in Figure F 3.

periments and compare the resulting distribution with the χ^2 -distribution. Though it is harder to come up with a quantitative measure of confidence, it would be a better test since it is based on a whole set of χ^2 measurements. Returning to the test of the `rnd` function, the graph on the right in Figure G.2 shows the *distribution* of χ^2 values found by performing a bin test, such as that seen in Appendix F, many times. Here we have used 11 bins, so the number of degrees of freedom is 10. We see that the most likely value of χ^2 is near $x = \chi^2/2 \sim 4.5$. Since $\nu = 10$ for this case, the curve shown on the left for $P(a, x)$, with $a = \nu/2 = 5$, is appropriate. This theoretical curve has a value of 0.5 for $x = \beta/2 \sim 4.5$. Statistical theory thus predicts that in half of our measurements, that is, for half of the bin tests, we should find a value of χ^2 that is smaller than ~ 9 . This theoretical prediction is in rough agreement with the distribution of χ^2 , which was actually measured, and which is shown on the right in Figure G.2.

One last remark concerns the so-called χ^2 fitting, where fitting is performed with a theoretical function \hat{y} containing unknown parameters (see Appendix D). In general, this is done by varying the parameter values to minimize

$$\Delta_{LS} = \sum_{i=1}^N \frac{[y(i) - y_{fit}(i)]^2}{\sigma(i)^2}, \quad (G.12)$$

where $y(i)$ are the measurements and $\sigma(i)$ are the standard deviations of the y values at $x(i)$. Then the values of $y_{fit}(i)$ with the best-fit parameters become the theoretical predictions of the model. It is easy to see that the minimum value of Δ_{LS} is precisely the quantity χ^2 that we discussed in connection to the variance test, in this case, of the best-fit model. Thus, all of our results concerning the test of the validity of a theoretical model using a χ^2 distribution applies to fitting

problems as well. It is important to note, however, that the appropriate number of degrees of freedom for the χ^2 distribution is $N - n$ where n is the number of parameters whose best values had to be obtained by minimizing Δ_{LS} in the first place.

EXERCISES

- G.1. Perform the χ^2 test of your random-number generator, and compare the results with that for True Basic's `rnd` shown in the text. Is your generator better or worse? How have you made the judgment?

REFERENCES

- [1] M. Abramowitz and I. A. Stegun, Editors, 1964, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. National Bureau of Standards, Washington, DC.
- [2] W. Feller, 1968, *An Introduction to Probability Theory and Its Applications*, Vol. 1, Wiley, New York. A classic introduction to the theory of probability.
- [3] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. This book discusses the generation of random numbers along with numerous statistical tests.

Solving Linear Systems

Systems of simultaneous linear equations lie at the heart of many physical problems. The eigenvalue problems we have encountered in our studies diffusion of on fractals (Chapter 7), in quantum mechanics (Chapter 10), and in the physics of music (Chapter 11) are obvious examples, and the relaxation approach to Laplace's equation (Chapter 5) was also be cast in this form. In this Appendix we focus on two types of linear problems and briefly discuss the main methods used to attack them. For the most part, our discussions will be introductory and emphasize principles rather than give detailed how-to's (although we will, as usual, provide ample references at the end of this appendix). These computations can be quite resource hungry if the size of the problem is large, and for this reason, techniques have been developed to take advantage of almost any special features that may be present in a particular problem.¹ We will not have space to discuss most such special techniques, nor will we discuss the important issues connected with singular (or nearly singular) problems.²

A common type of linear problem is one where we wish to solve for a column vector \mathbf{x} of N components for which

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b}, \quad (\text{H.1})$$

where \mathbf{A} is an $N \times N$ matrix and \mathbf{b} is a non-zero column vector also with N components.³ Another common type of linear problem is the eigenvalue-eigenfunction problem:

$$\mathbf{A} \cdot \mathbf{u}_i = \lambda_i \mathbf{u}_i, \quad (\text{H.2})$$

where the object is to find the eigenvalues λ_i and associated eigenfunctions \mathbf{u}_i for a square ($N \times N$) matrix \mathbf{A} . If N linearly independent eigenvectors can be found, then we can use them to construct the inverse matrix \mathbf{A}^{-1} as well. Of course these two problems are related, and some general matrix techniques are useful in both

¹Common "special" features include tridiagonal systems, which refers to the pattern of nonzero elements in the associated matrix, and sparse systems, in which most of the elements of the matrix are zero. Sparse matrices often arise when a physical system has only local interactions.

²In a matrix formulation $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, a singular problem is where the matrix \mathbf{A} is singular, or its determinant is zero and it projects a set of vectors \mathbf{x} into the null vector ($\mathbf{b} = 0$) when acting on them. A common method for dealing with singular matrices is called *singular value decomposition*.

³More general situations could arise where the matrix \mathbf{A} is not square, i.e., it may have M rows and N columns where $M \neq N$, and \mathbf{x} has N components whereas \mathbf{b} has M components. This situation corresponds to the cases where the number of equations (M) is larger ($M > N$) or smaller ($M < N$) than the number of unknowns. Whether there will be a unique (or *any*) solution for \mathbf{x} then depends on how many of the rows of \mathbf{A} are linearly independent of one another and whether there are any rows that conflict. An example of the latter would be having two identical rows (say, i and j) of \mathbf{A} but with $b_i \neq b_j$.

cases. However, most approaches are designed specifically for one type or the other, so we will consider them separately.

H.1 SOLVING $\mathbf{A} \mathbf{x} = \mathbf{B}$, $\mathbf{B} \neq \mathbf{0}$

The matrix equation (H.1) where \mathbf{A} is $N \times N$ can be written out in terms of the individual elements

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_N \end{pmatrix}, \quad (\text{H.3})$$

or as a set of simultaneous linear equations,

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1N}x_1 = b_1, \quad (\text{H.4})$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2N}x_2 = b_2, \quad (\text{H.5})$$

$\dots = \dots$

$$a_{N1}x_1 + a_{N2}x_2 + \dots + a_{NN}x_N = b_N. \quad (\text{H.6})$$

If some of these equations are linearly dependent on the others, then the set is underdetermined and there are an infinite number of solutions. On the other hand, if some of the equations are in conflict with others, then there is no solution. We now assume that these cases can be excluded, so that there is a unique solution \mathbf{x} .

H.1.1 Gaussian Elimination

The most familiar method of solving (H.1) involves systematic substitution and elimination, and is known as Gaussian elimination. We first use (H.4) to solve for x_1 in terms of (x_2, \dots, x_N) , and substitute this into the remaining $N - 1$ equations (H.5) through (H.6).⁴ This yields $N - 1$ equations in $N - 1$ unknowns. We then repeat this operation until we get down to the last equation (H.6) now containing only x_N , and thereby obtain the solution for x_N . We then work our way back up the chain of equations (H.4)-(H.6) to the equation that only contained x_{N-1} and x_N . Since x_N is now known, this solves for x_{N-1} . We repeat this process until we get x_1 from (H.4), and all of the x_1, \dots, x_N are then determined.

The elimination process requires a total of $O(N^3)$ operations (multiplications and subtractions) since the elimination of x_1 from the $N - 1$ equations require $O([N - 1]^2)$ operations, and the next elimination of x_2 from the $N - 2$ equations require $O([N - 2]^2)$ operations, and so forth. On the other hand, the back substitution process requires $O(N^2)$ operations in total since the first substitution requires $O(1)$ operation, the next one $O(2)$, etc. In the end, the entire process requires $O(N^3)$ operations. This is typical for solving a general $N \times N$ linear system with no special features that can be taken advantage of. Hence, the computational requirements can grow very fast if N is large (e.g., $N = 1000$ or more is not uncommon).

⁴Of course this cannot be done if (H.4) does not contain x_1 (i.e., $a_{11} = 0$). We will come back to this point a bit later.

In terms of the matrix formulation of (H.3), Gaussian elimination can be described as a row reduction of the matrix \mathbf{A} into an upper triangular matrix (i.e., so that all of the entries below the diagonal are zero). That is, Gaussian elimination procedure takes suitable multiples of the first row ($a_{11}, a_{12}, \dots, a_{1N}$) and subtracts them from rows 2 through N to eliminate the first column from those rows. The procedure is then repeated with the now reduced second row (0, $a_{22}, a_{23}, \dots, a_{2N}$) to remove the second column from rows 3 through N , and so forth. As we do this to \mathbf{A} , we also carry out the same multiplications and subtractions to the corresponding component of \mathbf{b} , so that it is transformed in the end to a new vector \mathbf{b}' . In this process the solution vector is unchanged. We then end up with

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ 0 & a'_{22} & a'_{23} & \dots & a'_{2N} \\ 0 & 0 & a'_{33} & \dots & a'_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & a'_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_N \end{pmatrix}. \quad (\text{H.7})$$

We can then write

$$\begin{aligned} x_N &= \frac{1}{a'_{NN}} b'_N, \\ x_{N-1} &= \frac{1}{a'_{N-1,N-1}} (b'_{N-1} - a'_{N-1,N} x_N), \\ \dots &= \dots \\ x_1 &= \frac{1}{a'_{1,1}} \left(b'_1 - \sum_{j=2}^N a'_{1,j} x_j \right). \end{aligned} \quad (\text{H.8})$$

As a bonus, the product of the diagonal elements of \mathbf{A}' is equal to the determinant of the original matrix \mathbf{A} , since these row operations do not change the determinant.

All of the above, however, is contingent upon the diagonal terms of the original matrix \mathbf{A} being non-zero. If, e.g., $a_{11} = 0$, we stumble already in the first step of the Gaussian elimination. Actually, this is not as bad as you may think. Since the order of the simultaneous equations (H.4), (H.5), ..., (H.6) does not matter, we may simply interchange another row i for which $a_{i1} \neq 0$ with row 1 (and correspondingly, the i -th and first components of \mathbf{b}) and proceed from there. The solution \mathbf{x} is unchanged. Here the entry a_{11} is called a pivot (the first pivot in this example) and the procedure of interchanging rows to bring in a non-zero pivot is an example of *pivoting*. Pivoting can be introduced as we proceed with the Gaussian elimination process whenever required and need not be done all at once initially. Row pivoting is often done even when the default pivot is already non-zero if the value of the pivot is very small, since a small pivot can adversely affect the final precision. Sometimes column pivoting may also be done. In this case, \mathbf{b} is unchanged, but you must make a corresponding interchange (or relabelling) of the solution vector components x_i . Thus, much of the computational complexity in implementing Gaussian elimination is in taking care of pivoting; otherwise, the procedure is straightforward (though it is not necessarily the most efficient method).

H.1.2 Gauss-Jordan elimination

A technique that is very closely related to Gaussian elimination is that of Gauss-Jordan elimination. In this method, row reduction is used to transform \mathbf{A} into a diagonal matrix \mathbf{A}' instead of merely an upper triangular one as in Gaussian elimination. Thus the same procedure of subtracting multiples of rows from other rows to eliminate certain elements of the matrix is now applied each time to *all* rows rather than just those below. For example, suitable multiples of the second row ($a_{21}, a_{22}, \dots, a_{2N}$) are subtracted from row 1 as well as rows 3 through N to eliminate the second column entry in all the other rows. Again, as long as the corresponding operations are performed on the vector \mathbf{b} , the solution \mathbf{x} is unchanged. We thus end up with

$$\begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a'_{22} & 0 & \cdots & 0 \\ 0 & 0 & a'_{33} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & a'_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_N \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \cdots \\ b'_N \end{pmatrix}, \quad (\text{H.9})$$

so that the solution is

$$\begin{aligned} x_N &= \frac{b'_N}{a'_{NN}}, \\ x_{N-1} &= \frac{b'_{N-1}}{a'_{N-1,N-1}} \\ \dots &= \dots \\ x_1 &= \frac{b'_1}{a'_{1,1}}. \end{aligned} \quad (\text{H.10})$$

To achieve this, we may use pivoting as needed, just as in Gaussian elimination. Gauss-Jordan elimination also requires $O(N^3)$ operations, though with a different coefficient.

One extension of this method is in the calculation of \mathbf{A}^{-1} , by setting up a matrix equation

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1N} \\ y_{21} & y_{22} & \cdots & y_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ y_{N1} & y_{N2} & \cdots & y_{NN} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \quad (\text{H.11})$$

where the matrix \mathbf{Y} is the desired inverse. As a straightforward extension of the case where the right side of the equation is a column vector \mathbf{b} , we note that the operations of subtracting suitable multiples of rows of \mathbf{A} from other rows of \mathbf{A} to transform it to a diagonal matrix \mathbf{A}' can be applied in this situation as well as long as we also perform the same operations on the corresponding *rows* of the matrix \mathbf{B}

that appears on the right-hand side of (H.11) (initially a unit matrix), transforming it into \mathbf{B}'

$$\begin{pmatrix} a'_{11} & 0 & \dots & 0 \\ 0 & a'_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a'_{NN} \end{pmatrix} \begin{pmatrix} y_{11} & y_{12} & \dots & y_{1N} \\ y_{21} & y_{22} & \dots & y_{2N} \\ \dots & \dots & \dots & \dots \\ y_{N1} & y_{N2} & \dots & y_{NN} \end{pmatrix} = \begin{pmatrix} b'_{11} & b'_{12} & \dots & b'_{1N} \\ b'_{21} & b'_{22} & \dots & b'_{2N} \\ \dots & \dots & \dots & \dots \\ b'_{N1} & b'_{N2} & \dots & b'_{NN} \end{pmatrix}. \quad (\text{H.12})$$

In this process the solution matrix \mathbf{Y} is unchanged. The only twist we add here is to allow an additional type of operation to reduce \mathbf{A}' into a unit matrix. Thus, if we multiply any row of \mathbf{A}' by a constant and also do the same to the corresponding row of \mathbf{B}' on the right, then (H.12) is still valid with the same \mathbf{Y} . But if we do this in such a way as to make $\mathbf{A}' = \mathbf{I}$ (the identity matrix), then the result gives

$$\mathbf{B}' = \mathbf{I} \cdot \mathbf{Y} = \mathbf{Y}, \quad (\text{H.13})$$

and \mathbf{B}' on the right side of (H.12) is equal to the desired inverse $\mathbf{Y} = \mathbf{A}^{-1}$.

H.1.3 LU decomposition

An approach called *LU decomposition* also requires $O(N^3)$ operations, but usually with a smaller coefficient than with Gaussian or Gauss-Jordan elimination. The idea in this technique becomes evident when we realize that once a triangular matrix \mathbf{A}' is constructed in Gaussian elimination, the back substitution process (H.8) only requires $O(N^2)$ operations. Thus, the solution of (H.1) is very fast if \mathbf{A} is triangular to begin with.

Assuming that \mathbf{A} is non-singular and excluding pathological cases, we can write \mathbf{A} as the product of lower triangular (\mathbf{L}) and upper triangular (\mathbf{U}) matrices

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U}. \quad (\text{H.14})$$

In terms of the elements of the matrices we have

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{pmatrix} = \begin{pmatrix} \alpha_{11} & 0 & \dots & 0 \\ \alpha_{21} & \alpha_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \alpha_{N1} & \alpha_{N2} & \dots & \alpha_{NN} \end{pmatrix} \begin{pmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1N} \\ 0 & \beta_{22} & \dots & \beta_{2N} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \beta_{NN} \end{pmatrix}, \quad (\text{H.15})$$

where we can further choose $\alpha_{11} = \alpha_{22} = \dots = \alpha_{NN} = 1$ for simplicity (and without loss of generality). The determination of the α 's and β 's turns out to be straightforward if these calculations are made in a certain order.⁵

Once the matrices L and U are found, we can solve $A \cdot x = L \cdot U \cdot x = b$ in two steps. We first attack

$$L \cdot y = \begin{pmatrix} \alpha_{11} & 0 & \dots & 0 \\ \alpha_{21} & \alpha_{22} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ \alpha_{N1} & \alpha_{N2} & \dots & \alpha_{NN} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}, \quad (\text{H.16})$$

by forward substitution,

$$\begin{aligned} y_1 &= \frac{1}{\alpha_{11}} b_1 = b_1, \\ y_2 &= \frac{1}{\alpha_{22}} (b_2 - \alpha_{21} y_1) = (b_2 - \alpha_{21} y_1), \\ \dots &= \dots \\ y_N &= \frac{1}{\alpha_{NN}} \left(b_N - \sum_{j=1}^{N-1} \alpha_{Nj} y_j \right) = \left(b_N - \sum_{j=1}^{N-1} \alpha_{Nj} y_j \right). \end{aligned} \quad (\text{H.17})$$

Once the vector y is known we can then use backward substitution (as in Gaussian elimination) to solve

$$U \cdot x = \begin{pmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1N} \\ 0 & \beta_{22} & \dots & \beta_{2N} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \beta_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}. \quad (\text{H.18})$$

The results can be written as

$$\begin{aligned} x_N &= \frac{1}{\beta_{NN}} y_N, \\ x_{N-1} &= \frac{1}{\beta_{N-1,N-1}} (y_{N-1} - \beta_{N-1,N} y_N), \\ \dots &= \dots \\ x_1 &= \frac{1}{\beta_{1,1}} \left(y_1 - \sum_{j=2}^N \beta_{1,j} y_j \right). \end{aligned} \quad (\text{H.19})$$

Both (H.17) and (H.19) require $O(N^2)$ operations only. Since calculation of the α 's and β 's costs $O(N^3)$ operations, the total computational requirement is still

⁵See Press et al (Chapter 2) for details. As with the previously discussed methods, pivoting is crucial for the algorithm to work, and that is essentially the sole source of complexity in programming it in practice.

$O(N^3)$. However, one benefit of LU decomposition is that once we find \mathbf{L} and \mathbf{U} (i.e., the α 's and β 's), we can use these same matrices to perform (H.17) and (H.19) to find the solution for any right-hand side vector \mathbf{b} . As a corollary, solving for the N unit vectors \mathbf{b} will give the inverse \mathbf{A}^{-1} as with the other methods.

H.1.4 Relaxational method

Another useful approach to solving (H.1) is the iterative, relaxational method. Such a method is sometimes also used in combination with one of the direct approaches discussed above in order to improve the accuracy of the results. In a relaxational method, we essentially convert the time-independent equation (H.1) into a time-dependent one in the sense that another variable that is analogous to time (typically the iteration count) is introduced. For example, (H.1) is converted to

$$\mathbf{C} \cdot (\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}) = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(n)}, \quad (\text{H.20})$$

where the superscript on \mathbf{x} denotes the iteration count and \mathbf{C} is a suitably chosen matrix that determines the dynamics or time evolution that we are adding to the problem. If (H.20) can be used to construct a sequence $\mathbf{x}^{(n)}$ such that

$$\lim_{n \rightarrow \infty} \mathbf{x}^{(n)} = \mathbf{x}^{(\infty)}, \quad (\text{H.21})$$

then the left-hand side of (H.20) converges to zero, and thus so should the right-hand side. That is,

$$\mathbf{A} \cdot \mathbf{x}^{(\infty)} = \mathbf{b}, \quad (\text{H.22})$$

where $\mathbf{x}^{(\infty)}$ is the desired solution. Of course the iterative sequence defined by (H.20) may not always converge, and thus there is no guarantee that an arbitrarily chosen \mathbf{C} will help us.⁶

Some common examples can be constructed as follows. First, we write the matrix \mathbf{A} as the sum⁷ of a lower triangular matrix \mathbf{L} , a diagonal matrix \mathbf{D} , and an upper triangular matrix \mathbf{U}

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}. \quad (\text{H.23})$$

We then choose $\mathbf{C} = \mathbf{D}$, so that (H.22) becomes

$$\mathbf{D} \cdot (\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}) = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(n)}, \quad (\text{H.24})$$

and by collecting the n -th iteration quantities on the right we find

$$\mathbf{D} \cdot \mathbf{x}^{(n+1)} = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^{(n)} + \mathbf{D} \cdot \mathbf{x}^{(n)} = \mathbf{b} - (\mathbf{L} + \mathbf{U}) \cdot \mathbf{x}^{(n)}. \quad (\text{H.25})$$

⁶In Chapter 5, we encountered examples of the relaxational approach which essentially converted a Laplace's equation into a diffusion equation. Then the intrinsic property of the diffusion equation guaranteed that those iterative methods would lead to the desired solution.

⁷This should not be confused with the LU decomposition discussed earlier. Any matrix can be *additively* decomposed in this way by simply taking all the elements below, at, and above the diagonal into separate matrices. Also, \mathbf{L} and \mathbf{U} in this additive decomposition have nonzero elements only below and above the diagonal respectively

This is the general definition of the Jacobi method.⁸ The hallmark of this choice is that each component $x_i^{(n+1)}$ at iteration $n + 1$ is determined by other components $x_j^{(n)}$ at iteration n , i.e., those with $j \neq i$. Also, since the matrix multiplier on the left-hand side of (H.25) is diagonal, (H.25) gives us an explicit iteration algorithm.

Another common iterative approach is obtained by using $\mathbf{C} = \mathbf{L} + \mathbf{D}$. In this case the corresponding dynamics is

$$(\mathbf{L} + \mathbf{D}) \cdot \mathbf{x}^{(n+1)} = \mathbf{b} - \mathbf{U} \cdot \mathbf{x}^{(n)}, \quad (\text{H.26})$$

which is called the Gauss-Seidel method.⁹ Equation (H.26) is not explicit in the sense that it appears necessary to solve it in some other way to determine $\mathbf{x}^{(n+1)}$ from $\mathbf{x}^{(n)}$ since $\mathbf{L} + \mathbf{D}$ is not diagonal. This seems like a major drawback because we are looking for ways to solve a linear equation just like (H.26) in the first place! However, there is no need to despair; in fact, (H.26) is as good as an explicit relation. We can see this by noting that we can evaluate $\mathbf{x}^{(n+1)}$ sequentially from the first component toward the N -th component. Since the first row of $\mathbf{L} + \mathbf{D}$ only contains the element D_{11} , we can first evaluate $x_1^{(n+1)}$ explicitly from $\mathbf{x}^{(n)}$. Then, since the second row of $\mathbf{L} + \mathbf{D}$ contains only the elements L_{21} and D_{22} , we have $L_{21}x_1^{(n+1)} + D_{22}x_2^{(n+1)}$ as the second component of the left-hand side of (H.26). However, since $x_1^{(n+1)}$ was already evaluated, we can use that value in addition to $\mathbf{x}^{(n)}$ to evaluate $x_2^{(n+1)}$. Likewise, the third row leads to the evaluation of $x_3^{(n+1)}$ in terms of $\mathbf{x}^{(n)}$, $x_1^{(n+1)}$, and $x_2^{(n+1)}$, the latter two already having been evaluated. We can continue in this manner, using previously evaluated components of $\mathbf{x}^{(n+1)}$ immediately in order to evaluate the next component. The end result is that we can solve (H.26) for $\mathbf{x}^{(n+1)}$ without actually inverting $\mathbf{L} + \mathbf{D}$ or resorting to some other linear solvers (such as Gaussian elimination).¹⁰

The important question of convergence needs to be addressed. As already stated, there is no guarantee in general that the iteration $\mathbf{x}^{(n)}$ converges. Let us go back to the general formulation (H.20), and assume for any iterate m

$$\mathbf{x}^{(m)} = \mathbf{x} + \epsilon^{(m)}, \quad (\text{H.27})$$

where \mathbf{x} is the true solution of $\mathbf{A} \cdot \mathbf{x} = \mathbf{f}$. Then (H.20) can be rewritten as

$$\mathbf{C} \cdot \mathbf{x} + \mathbf{C} \cdot \epsilon^{(n+1)} = \mathbf{b} - (\mathbf{A} - \mathbf{C}) \cdot \mathbf{x} - (\mathbf{A} - \mathbf{C}) \cdot \epsilon^{(n)}. \quad (\text{H.28})$$

By using $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, this further simplifies to

$$\mathbf{C} \cdot \epsilon^{(n+1)} = (\mathbf{C} - \mathbf{A}) \cdot \epsilon^{(n)}, \quad (\text{H.29})$$

or

$$\epsilon^{(n+1)} = \mathbf{E} \cdot \epsilon^{(n)}, \quad \mathbf{E} \equiv \mathbf{C}^{-1} \cdot (\mathbf{C} - \mathbf{A}). \quad (\text{H.30})$$

Thus, for the error $\epsilon^{(n)}$ to approach zero as $n \rightarrow \infty$, all the elements of $\mathbf{E}^{(n)}$ must go to zero as $n \rightarrow \infty$. Unfortunately, we usually have to verify this condition for specific matrices \mathbf{C} and \mathbf{A} , as not much can be said for the general case.

⁸The Jacobi approach encountered in Chapter 5 is a special case of this. See the exercises.

⁹Again, the Gauss-Seidel approach we used in Chapter 5 is a special case of this.

¹⁰You should find this familiar from the discussion of the Gauss-Seidel method applied to the Laplace's equation in Chapter 5.

H.2 EIGENVALUES AND EIGENFUNCTIONS

Next, we consider the eigensystem problem (H.2). Although this looks similar to the simultaneous linear equations (H.1) on the surface, it is much more difficult in general. Writing (H.2) as a homogeneous linear equation

$$(\mathbf{A} - \lambda_i \mathbf{I}) \cdot \mathbf{u}_i = \mathbf{0}, \quad (\text{H.31})$$

for each eigenvalue λ_i and its associated eigenvector \mathbf{u}_i , we see that the matrix $\mathbf{A} - \lambda_i \mathbf{I}$ is singular. Thus, even if we know the value of λ_i (which we usually do not!), none of the linear system solvers we discussed can be used to solve for \mathbf{u}_i directly.

If the size of the system N is small, it is tempting to write (H.31) in the form

$$\begin{aligned} (a_{11} - \lambda)x_1 + a_{12}x_2 + \dots + a_{1N}x_1 &= 0, \\ a_{21}x_1 + (a_{22} - \lambda)x_2 + \dots + a_{2N}x_2 &= 0, \\ &\dots = \dots \\ a_{N1}x_1 + a_{N2}x_2 + \dots + (a_{NN} - \lambda)x_N &= 0. \end{aligned} \quad (\text{H.32})$$

Since this is a homogeneous linear equation, the components x_i can be scaled by an arbitrary common factor. Thus, we may try setting a specific scale by choosing¹¹ $x_N = 1$. We could then hope to solve the resulting N simultaneous equations analytically for both λ and the remaining $N - 1$ components of \mathbf{x} . Indeed, for very small matrices, this may be a useful approach.¹² However, already for matrices larger than 3×3 or so, we need to find a way that is amenable to numerical computation.

Most numerical methods for solving an eigensystem attempt to find a similarity transformation¹³ which diagonalizes \mathbf{A} ,

$$\mathbf{P}^{-1} \cdot \mathbf{A} \cdot \mathbf{P} = \mathbf{A}', \quad (\text{H.33})$$

where the matrix \mathbf{A}' is diagonal. If such a matrix \mathbf{P} can be found, we will have found both the eigenvalues and the eigenvectors. This can be seen as follows. Suppose that the N eigenvectors \mathbf{u}_i of \mathbf{A} are linearly independent and span the N -dimensional space, i.e., any vector in this space can be written as a linear combination of those eigenvectors. Now let $\mathbf{v}_i \equiv \mathbf{P}^{-1} \cdot \mathbf{u}_i$, and let \mathbf{A}' act on it. We get

$$\mathbf{A}' \cdot \mathbf{v}_i = (\mathbf{P}^{-1} \cdot \mathbf{A} \cdot \mathbf{P}) \cdot (\mathbf{P}^{-1} \cdot \mathbf{u}_i) = \mathbf{P}^{-1} \cdot \mathbf{A} \cdot \mathbf{u}_i = \lambda_i \mathbf{v}_i, \quad (\text{H.34})$$

where λ_i ($i = 1, 2, \dots, N$) are the eigenvalues of \mathbf{A}' with $\mathbf{v}_i = \mathbf{P}^{-1} \cdot \mathbf{u}_i$ as the corresponding eigenvectors. Since the elements of a diagonal matrix are its eigenvalues, we have

$$\mathbf{A}' = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \lambda_N \end{pmatrix}. \quad (\text{H.35})$$

¹¹This is not always possible since the correct solution may have $x_N = 0$, but as long as it is non-zero, we can do this

¹²This may be tempting, particularly if we already know some of the eigenvalues, e.g., by solving the characteristic equation $\det(\mathbf{A} - \lambda \mathbf{I}) = 0$

¹³This transformation corresponds to a change of basis vectors

Furthermore, since the eigenvectors of a diagonal matrix are simply vectors with all but one of the components equal to zero, we can, without loss of generality, take

$$\mathbf{v}_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (\text{H.36})$$

for $i = 1, 2, \dots, N$ where the only non-zero entry is in the i -th row. Now by construction

$$\mathbf{P} \cdot \mathbf{v}_i = \mathbf{P} \cdot (\mathbf{P}^{-1} \cdot \mathbf{u}_i) = \mathbf{u}_i. \quad (\text{H.37})$$

Since the \mathbf{v}_i 's are unit column vectors, the left-hand side of (H.37) is simply the corresponding i -th column of \mathbf{P} . In other words, the columns of \mathbf{P} are the eigenvectors of the original matrix \mathbf{A} , and the elements of \mathbf{A}' are its eigenvalues.

Finding the proper similarity transformation \mathbf{P} is often long and complicated. For this reason, the task of solving the eigensystems is one that is best left to professionally constructed and well-tested subroutines available in the scientific subroutine libraries such as LAPACK (see the references). Here we only sketch in very general terms how these routines work. First, we note that a series of similarity transformations constitute a similarity transformation as a whole. That is, if

$$\mathbf{P} \equiv \mathbf{P}_1 \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \dots, \quad (\text{H.38})$$

then

$$\mathbf{P}^{-1} \cdot \mathbf{A} \cdot \mathbf{P} = \dots \cdot \mathbf{P}_3^{-1} \cdot \mathbf{P}_2^{-1} \cdot (\mathbf{P}_1^{-1} \cdot \mathbf{A} \cdot \mathbf{P}_1) \cdot \mathbf{P}_2 \cdot \mathbf{P}_3 \dots. \quad (\text{H.39})$$

Thus the typical strategy is to apply many similarity transformations successively, each time reducing the off-diagonal content.

A typical elemental transformation is a plane rotation, i.e., a rotation in a selected two-dimensional subspace of the overall N -dimensional vector space. One such rotation can be written as

$$\mathbf{P} = \begin{pmatrix} 1 & & & & \\ \dots & & & & \\ & \cos\theta & \dots & \sin\theta & \\ & \vdots & 1 & \vdots & \\ & -\sin\theta & \dots & \cos\theta & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}. \quad (\text{H.40})$$

Another class of elemental transformations takes advantage of the fact that we can often factorize a matrix \mathbf{A} into a product of two other matrices where one or

both of those matrices have desirable properties, making the ensuing calculations simpler.¹⁴ The idea here is the observation that if

$$\mathbf{A} = \mathbf{P} \cdot \mathbf{Q}, \quad (\text{H.41})$$

then merely reversing the order of multiplication results in a similarity transformation

$$\mathbf{Q} \cdot \mathbf{P} = (\mathbf{P}^{-1} \cdot \mathbf{A}) \cdot \mathbf{P}. \quad (\text{H.42})$$

The numerical solution of eigensystems typically consists of the judicious combination of these and other similarity transformations, reducing the matrix into simpler forms wherever possible. Once the matrix is reduced to *almost* diagonal form, one then has an approximate set of eigenvalues and eigenvectors. Then there are several approaches to refine the solution to make them more accurate.

H.2.1 Approximate Solution of Eigensystems

So far we have discussed, at least in principle, the exact solutions of the eigenvalue-eigenvector problem. However, there are also approaches that target approximate solutions, often in a certain range of eigenvalues (such as those of the largest or smallest moduli). An example of this called the power method was already discussed in Chapter 10. Here, we mention a related method called *Arnoldi-Saad* method, which allows the very accurate determination of a subset of the eigenvalues near the maximum in the spectrum. This is often convenient, because in many physical problems, the extremal regions of the eigenspectrum is where the action is.¹⁵ In the Arnoldi-Saad algorithm, the original matrix \mathbf{A} is first reduced to a much smaller (and specially formatted) one, and then this latter matrix is solved by one of the standard routines that uses methods discussed earlier.

We start with an arbitrary normalized vector \mathbf{u}_1 of dimension N where \mathbf{A} has the dimension of $N \times N$. Then we choose the subspace dimension $M \leq N$ which is to be the dimension of the reduced upper Hessenberg matrix¹⁶ \mathbf{H} which will then be diagonalized essentially exactly. The reduced matrix \mathbf{H} is obtained recursively and at the same time as a sequence of normalized basis vectors \mathbf{u}_i ($i = 2, \dots, M$) in the following way.

First, the element H_{11} is obtained as $\mathbf{u}_1^T \cdot \mathbf{A} \cdot \mathbf{u}_1$. (The superscript T refers to the transpose, i.e., the matrix obtained by interchanging the row and column indices of all the elements.) Second, the product $\mathbf{v}_2 \equiv H_{21}\mathbf{u}_1$ is calculated as $\mathbf{v}_2 = \mathbf{A} \cdot \mathbf{u}_1 - H_{11}\mathbf{u}_1$. We then choose $H_{21} = \|\mathbf{v}_2\|$ to satisfy the normalization requirement $\|\mathbf{u}_2\|^2 = 1$. At the next iteration, we obtain H_{12} and H_{22} as $\mathbf{u}_1^T \cdot \mathbf{A} \cdot \mathbf{u}_2$ and $\mathbf{u}_2^T \cdot \mathbf{A} \cdot \mathbf{u}_2$, respectively, and then, \mathbf{u}_3 and H_{32} are obtained similarly to the

¹⁴For example, one of the factors may be a triangular matrix. A multiplicative decomposition of the form $\mathbf{A} = \mathbf{Q} \cdot \mathbf{R}$, where \mathbf{Q} is orthogonal and \mathbf{R} is upper triangular is a common example, and is at the heart of the so-called QR algorithm.

¹⁵Such is the case, for example, in the long-time behavior of diffusion on fractals, a topic treated in Chapter 7.

¹⁶An upper Hessenberg matrix is almost like an upper triangular matrix except that it may also have non-zero elements in the first subdiagonal entries (one element below the diagonals).

first iteration. This process continues until all elements of \mathbf{H} and all of the \mathbf{u}_i are determined. In general, we have

$$H_{ij} = \mathbf{u}_i^T \cdot \mathbf{A} \cdot \mathbf{u}_j, \quad (H.43)$$

$$H_{j+1,j} \mathbf{u}_{j+1} = \mathbf{A} \cdot \mathbf{u}_j - \sum_{i=1}^j H_{ij} \mathbf{u}_i,$$

where \mathbf{u}_i ($i = 1, 2, \dots, M$) forms an orthonormal set.

It turns out that this procedure ensures that the eigenvalues of \mathbf{H} are approximate eigenvalues of the original matrix \mathbf{A} , and if \mathbf{y} is the eigenvector of \mathbf{H} with the eigenvalue λ , then the vector \mathbf{z} defined by its components $z_j \equiv \sum_{i=1}^M [u_i]_{j,i}$ satisfies the orthogonality

$$[(\mathbf{A} - \lambda \mathbf{I}) \cdot \mathbf{z}] \cdot \mathbf{u}_i = 0, \quad i = 1, 2, \dots, M. \quad (H.44)$$

This means that \mathbf{z} is an approximate eigenvector of \mathbf{A} to within the subspace spanned by the \mathbf{u}_i 's, with the approximate eigenvalue λ . The approximation is the better the larger M is, and also is more accurate for the eigenvalues for which $|\lambda|$ is largest. The Arnoldi-Saad algorithm is one example of a method that allow us to focus on certain regions of the eigenspectrum, and obtain eigenvalues and eigenvectors accurately (though not exactly) in that region. It should be noted that this region of interest can often be moved around by transforming the original matrix \mathbf{A} suitably (as was discussed when we treated the power method in Chapter 10).

EXERCISES

- H.1. Write a program which implements Gaussian elimination (without pivoting) and demonstrate that it works as intended. Also do the same for Gauss-Jordan elimination.
- H.2. Show that the Jacobi method introduced in Chapter 5 for the Laplace's equation is a special example of the more general definition (H.25) given in this appendix and identify the quantities in that example which correspond to those given in (H.25).
- H.3. Show that the Gauss-Seidel method introduced in Chapter 5 for the Laplace's equation is a special example of the more general definition (H.26) given in this Appendix and identify the quantities in that example which correspond to those given in (H.26).
- H.4. Find an example of a linear problem where the Jacobi method does *not* converge to the correct solution.

REFERENCES

- [1] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, 1979, *LINPACK User's Guide*, SIAM, Philadelphia. Also, B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, 1976, *Matrix Eigensystem Routines - EISPACK Guide*, Springer-Verlag, New York. Linpack and Eispack are some of the earliest libraries of linear system solver subroutines, on which many of the newer generations of numerical subroutine

libraries are based. These user's guides provide not only the detailed how-to's of calling each subroutine in the package, but also give glimpses into how they work with illustrative examples.

- [2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, 1999, *LAPACK User's Guide*, SIAM, Philadelphia. LAPACK is the successor to LINPACK and EISPACK designed for optimized performance on high-performance computers. There are many other packages which supplement LAPACK by further optimizations for certain problems or computing platforms, including LASPACK (for sparse matrices), PARPACK and ScaLAPACK (for distributed memory parallel machines).
- [3] C. E. Pearson, 1986, *Numerical Methods in Engineering and Science*, Van Nostrand Reinhold, New York. There are nice discussions of solvers for linear systems and eigensystems with simple examples in Chapter 2.
- [4] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, *Numerical Recipes*, Cambridge University Press, Cambridge. Chapter 2 discusses various direct solution methods for the linear problem $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, while eigensystems are the main topic of Chapter 11. Relaxation methods are discussed in Chapter 17.
- [5] Y Saad, "Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices," *Linear Algebra Appl.* **34**, 269 (1980). See also W. E. Arnoldi, "The principle of minimized iterations in the solution of the matrix eigenvalue problem," *Quart. Appl. Math.* **9**, 17 (1951).

Index

- χ^2 test, 520, 523
- χ^2 distribution, 521
- acoustic impedance, 384
- action potential, 437
- adaptive time step, 413
- adiabatic approximation, 29
- air drag, 21
 - air drag at high altitudes, 30
 - air drag, approximate theory, 22
- aliasing, 488
- amino acids, 390
- annealing, 402
- approach to equilibrium, 204, 279
- Arnoldi-Saad method, 537
- arrow of time, 204
- autocorrelation function, 490
- avalanche, 447
- banjo, 373
- barrier potential, 318
- baseball, batted, 32
- baseball, effect of wind, 34
- baseball, long home
 - runs, 35
- baseball, spin-dependent force, 38
- baseball, trajectory of a curveball, 36
- baseball, trajectory of a knuckleball, 42
- Bernoulli effect, 37
- bicycles, 18
- bifurcation, 67
- billiard problem, 82
- Biot-Savart law, 148
- bisection, 471
- blind ant, 230
- Boltzmann factor, 237
- brain models, 418
- breadth-first search method, 225
- cannon shell trajectory, 25
- canonical ensemble, 237
- cellular automata, 445
- cellular automata, game of *life*, 445
- centered difference methods, 464
- central limit theorem, 521
- chaos and noise, 89
- chaos, Feigenbaum δ , 69
- chaos, pendulum, 60
- chaos, period doubling, 73
- chaos, sensitivity to initial conditions, 61
- chaos, strange attractors, 65
- chaotic motion of Hyperion, 123
- cluster growth, 206
- cluster labelling method, 220
- cluster, fractal dimensionality, 210
- coarse graining, 195
- correlation function, 255
- Crank-Nicholson method, 338
- cream-in-coffee problem, 182, 197
- critical exponents, α , 253
- critical exponents, β , 243
- critical exponents, γ , 255
- critical phenomena and fluctuations, 257
- critical temperature, 241
- critical temperature, 2-D Ising model, 250
- data collapsing, 267
- data fitting, 493
- debugging and checking, 11
- depth-first search, 191
- derivative, finite difference form, 2
- detailed balance, 246
- diffusion, 131, 181, 185, 291
- diffusion and entropy, 203
- diffusion and fluctuations, 186
- diffusion and probability, 195
- diffusion constant, 184
- diffusion equation, 196
- diffusion limited aggregation, 206
- diffusion on a fractal, 229
- dimensionality of a cluster, 209
- Dirac delta function, 480
- discrete Fourier transform, 481
- dispersion, 171
- dynamical matrix, 374
- earthquakes, 407
- Eden model, 206
- eigenfunctions, 527
- eigenvalue problems, 304, 324, 373, 527

- eigenvalue problems and random walks, 231
 eigenvalue problems, approximate methods, 537
 electric field, 135
 electric potential, 129
 energy landscape, 426
 entropy, 203
 equilibrium, 204
 equipartition, 279
 ergodic hypothesis, 204
 error, global, 457, 458
 error, local, 457, 458
 Euler method, 3, 20, 456
 Euler-Cromer method, 52, 97, 462
 exchange interaction, 237
 fast Fourier transform, 483
 Feigenbaum δ , 69
 Fermi-Pasta-Ulam problem, 294
 ferromagnet, 236
 FFT, 483
 finite difference, first derivative, 2
 finite difference, fourth derivative, 170
 finite difference, second derivative, 157
 first-order phase transitions, 259
 fluctuation-dissipation theorem, 254, 255
 forest fire problem, 227
 Fourier analysis, 89, 165, 360
 Fourier transform, 479
 fractal clusters, 210
 fractal coastlines, 216
 fractal dimension, 210
 fractal dimensionality of a curve, 212
 free energy, 241
 free energy landscape, 401
 friction, 370, 409
 gas, dilute, 280
 Gauss-Jordan elimination, 530
 Gauss-Seidel method, 142, 534
 Gaussain quadrature, 504
 Gaussian elimination, 528
 general relativity, effect on mercury, 108
 generating fractal curves, 212
 genetic algorithms, 473
 golden mean, 471
 golf, 44
 golf ball, effect of dimples, 45
 graphing results, 10
 guitar, 357
 guitar string, 163
 guitar, standing waves, 359
 Gutenberg-Richter law, 406
 half-life, 2
 Halley's comet, 100
 Hamming distance, 427
 Hebbian learning, 427
 Heisenberg model, 238
 Heisenberg uncertainty principle, 340
 Helmholtz motion, 368
 Hodgkin-Huxley model, 439
 homogeneous functions and scaling, 265
 Hoshen-Kopelman algorithm, 221
 hypercubic lattice, 184
 Hyperion, chaotic tumbling, 123
 hysteresis, 260
 initial value problem, 2
 integration, 500
 inverse square law, 101, 103
 ion channels, 438
 Ising model, 236
 Jacobi method, 131
 Kepler's laws, 98
 Kirkwood gaps, 119
 Koch curve, 212
 Laplace's equation, 129
 leapfrog method, 335, 383
 least squares, 494, 497
 least-squares fitting, 112
 Lennard-Jones potential, 273, 318, 327
 Levinthal's paradox, 392
 linear congruential random number generator, 512
 linear regression, 494
 linear systems, 527
 logistic map, 70
 Lorenz butterfly, 81
 Lorenz model, 75
 LU decomposition, 531
 Lyapunov exponent, 62
 magnetic field of a solenoid, 152
 magnetic fields, 148
 magnetic fields and spin systems, 239

- magnetization, 238
- Magnus force, 37
- matrix method in quantum mechanics, 323
- Maxwell velocity distribution, 282
- mean field approximation, 240
- mean free path, 271
- melting, 285
- metastable states, 260, 400
- Metropolis algorithm, 245
- molecular dynamics, 270
- Monte Carlo integration, 506
- Monte Carlo method, 235, 244, 395, 423, 474
- multiple linear regression, 498
- neural networks, 418
- neural networks, Hopfield model, 434
- neuron, 418
- neurons and Ising spins, 421
- Newton-Cotes methods, 502
- Newton-Raphson method, 469
- nonlinear springs, 295
- normal distribution, 521
- normal modes, 296
- Nyquist frequency, 483
- octave stretching, 173
- optimization, 469, 472
- ordinary differential equations, 456
- parity, 308
- pendulum, chaotic, 58
- pendulum, damped and nonlinear, 54
- pendulum, driven, 55
- pendulum, simple, 48
- perceptron, 434
- percolation, 218
- percolation and phase transitions, 222
- percolation clusters, 218
- percolation clusters and fractals, 224
- percolation threshold, 219
- period doubling, 68, 73
- periodic boundary conditions, 249, 275
- phase diagrams, 261
- phase space, 62
- phase transition, 235
- piano, 362
- piano tone spectrum, 168
- planetary data, 97
- planetary orbits, 101
- planetary orbits, stability and the inverse square law, 103
- Poincaré section, 63
- Poisson's equation, 144
- power laws, 243
- power method, 325
- power spectrum, 89, 165, 490
- precession of the perihelion of mercury, 108
- program listings, 4
- programming guidelines, 14
- programming languages, 3
- projectile motion, 18
- protein folding, 390
- proteins, primary structure, 390
- proteins, tertiary structure, 390
- pseudocode, 3
- Q factor, 387
- quantum mechanics, anharmonic oscillator, 322
- quantum mechanics, matching method, 315
- quantum mechanics, shooting method, 307
- quantum mechanics, two-dimensional harmonic oscillator, 329
- quantum mechanics, variational principle, 326
- quantum mechanics, variational Monte Carlo method, 328
- quantum mechanics, wave packet spreading, 340
- radioactive decay, 1
- random number generators, 512
- random numbers, 183
- random numbers, nonuniform distributions, 516
- random numbers, quality tests, 514
- random numbers, rejection method, 517
- random numbers, transformation method, 516
- random systems, 181
- random walk, 182
- random walk in one dimension, 183
- recursion, 212
- regression, linear, 494
- regression, multiple linear, 498
- relaxation method, 131, 533

- relaxation method, diffusion, 131
 resonance, 387
 Richardson's formula, 503
 Romberg integration, 503
 root finding, 469
 round-off error, 12
 routes to chaos, intermittency, 80
 routes to chaos, period doubling, 68
 Runge-Kutta methods, 458
 sandpile, 447
 scaling, 264
 Schrödinger equation, Monte-Carlo
 methods, 328
 Schrödinger equation, time-dependent, 333,
 345, 350
 Schrödinger equation, time-independent,
 303
 Schrödinger equation, two-dimensional
 harmonic oscillator, 329
 Schrödinger equation, variational
 method, 326
 secant method, 471
 second order differential equations, 460
 second order phase transitions, 223
 second-order phase transitions, 243
 second-order transition, 260
 self-avoiding walk, 189
 self-organized criticality, 407, 448
 shooting method, 307, 309
 simple harmonic oscillator, 48
 Simpson's rule, 502
 simulated annealing, 473
 simultaneous over-relaxation, 142
 solar system, 94
 spanning cluster, 218
 specific heat, 253
 spectral methods, 174, 350
 stadium billiard, 82
 staggered grid, 383
 step sizes, 13
 stick-slip motion, 370
 stochastic systems, 181
 Stokes' law, 21
 strange attractor, 65
 susceptibility, 254
 Taylor expansion, 2
 three-body problem, 113
 time-dependent Schrödinger equation, 333
 time-dependent Schrödinger equation,
 spectral method, 350
 time-dependent Schrödinger equation, two
 dimensions, 345
 time-independent Schrödinger
 equation, 303
 trapezoidal rule, 502
 traveling salesman problem, 475
 triangular lattice, 287
 tunneling, 318
 turbulence, 75
 two-dimensional harmonic oscillator, 329
 unitarity, 337
 variational principle, 326
 Verlet method, 272, 463
 vibrating membrane, 373
 violin, 368
 virial theorem, 285
 viscosity, 24
 wave equation, 156, 359, 363
 wave equation and stability, 161
 wave equation for a stiff string, 170
 wave equation for sound, 383
 wave equation in two dimensions, 376
 wave equation, solution, 158
 wave packet, 339
 waves, 156
 waves on a string, 159
 waves, reflection, 162
 waves, spectral methods, 174
 XY model, 238