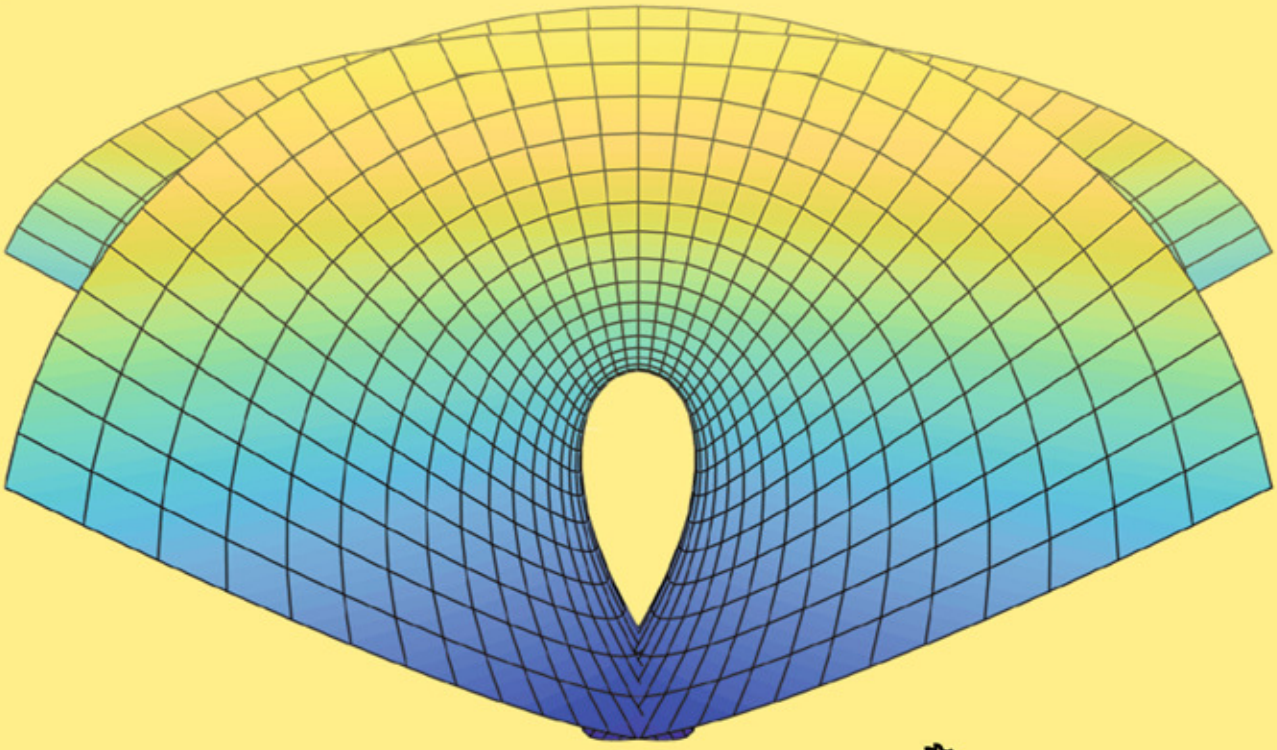
Ronald L. Lipsman · Jonathan M. Rosenberg

Multivariable Calculus with MATLAB®

With Applications to Geometry and Physics



Springer

Multivariable Calculus with MATLAB®

Ronald L. Lipsman · Jonathan M. Rosenberg

Multivariable Calculus with MATLAB®

With Applications to Geometry and Physics

Springer

Ronald L. Lipsman
Department of Mathematics
University of Maryland
College Park, MD, USA

Jonathan M. Rosenberg
Department of Mathematics
University of Maryland
College Park, MD, USA

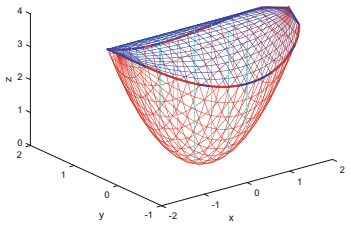
ISBN 978-3-319-65069-2 ISBN 978-3-319-65070-8 (eBook)
DOI 10.1007/978-3-319-65070-8

Library of Congress Control Number: 2017949120

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface



The preface of a book gives the authors their best chance to answer an extremely important question: What makes this book special?

This book is a reworking and updating for MATLAB of our previous book (joint with Kevin R. Coombes) *Multivariable Calculus with Mathematica*®, Springer, 1998. It represents our attempt to enrich and enliven the teaching of multivariable calculus and mathematical methods courses for scientists and engineers. Most books in these subjects are not substantially different from those of fifty years ago. (Well, they may include fancier graphics and omit several topics, but those are minor changes.) This book is different. We do touch on most of the classical topics; however, we have made a particular effort to illustrate each point with a significant example. More importantly, we have tried to bring fundamental physical applications—Kepler's laws, electromagnetism, fluid flow, energy estimation—back to a prominent position in the subject. From one perspective, the subject of multivariable calculus only exists because it can be applied to important problems in science.

In addition, we have included a discussion of the geometric invariants of curves and surfaces, providing, in effect, a brief introduction to differential geometry. This material provides a natural extension to the traditional syllabus.

We believe that we have succeeded in resurrecting material that used to be in the course while introducing new material. A major reason for that success is that we use the computational power of the mathematical software system MATLAB to carry a large share of the load. MATLAB is tightly integrated into every portion of this book. We use its graphical capabilities to draw pictures of curves and surfaces; we use its symbolical capabilities to compute curvature and torsion; we use its numerical capabilities to tackle problems that are well beyond the typical mundane examples of textbooks that treat the subject without using a computer. Finally, and this is something not done in any other books at this level, we give a serious yet elementary explanation of how various numerical algorithms work, and what their advantages and disadvantages are. Again, this is something that could not be accomplished without a software package such as MATLAB.

As an additional benefit from introducing MATLAB, we are able to improve students' understanding of important elements of the traditional syllabus. Our students are better able to visualize regions in the plane and in space. They develop a better feel for the geometric meaning of the gradient; for the method of steepest descent; for the orthogonality of level curves and gradient flows. Because they have tools for visualizing cross sections of solids, they are better able to find the limits of integration in multiple integrals.

To summarize, we think this book is special because, by using it:

- students obtain a better understanding of the traditional material;
- students see the deep connections between mathematics and science;
- students learn more about the intrinsic geometry of curves and surfaces;
- students acquire skill using MATLAB, a powerful piece of modern mathematical software;
- instructors can choose from a more exciting variety of problems than in standard textbooks; and
- both students and instructors are exposed to a more holistic approach to the subject—one that embraces not only algebraic/calculus-based solutions to problems, but also numerical, graphical/geometric and qualitative approaches to the subject and its problems.

Conventions

Throughout the book, MATLAB commands, such as **`solve`**, are printed in typewriter boldface. Theorems and general principles, such as: *derivatives measure change*, are printed in a slanted font. When new terms, such as *torsion*, are introduced, they are printed in an italic font. File names and URLs (web addresses) are printed in `typewriter font`. Everything else is printed in a standard font.

At the start of each chapter, below the title, is a small illustration. Each is a graphic generated by a MATLAB command. Most are taken from the MATLAB solution to one of the problems in the accompanying problem set. A few are taken from the chapter itself. Finally, in this Preface, the graphic represents a more eclectic choice. We leave it to the industrious reader to identify the source of these graphics, as well as to reproduce the figure.

Acknowledgments

We above all want to thank our former collaborators for their contributions to this project. Kevin Coombes (now at the Department of Biomedical Informatics at Ohio State University) was a co-author of *Multivariable Calculus with Mathematica*® and kindly agreed to let us adapt that book for MATLAB. Brian Hunt was a co-author of *A Guide to MATLAB* and taught us many useful MATLAB tricks and

tips. Paul Green helped develop MATLAB exercises for multivariable calculus that eventually worked their way into this book.

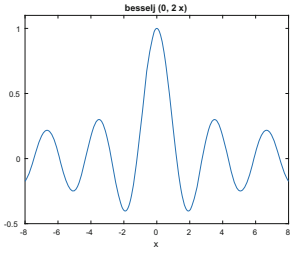
Jonathan Rosenberg thanks the National Science Foundation for its support under grant DMS-1607162. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

College Park, MD, USA
October 1, 2017

Ronald L. Lipsman
Jonathan M. Rosenberg

Contents

Chapter 1
Introduction



We wrote this book with the third semester of a physical science or engineering calculus sequence in mind. The book can be used as a supplement to a traditional calculus book in such a course, or as the sole text in an "honors" course in the subject. It can equally well be used in a postcalculus course or problem seminar on mathematical methods for scientists and engineers. Finally, it can serve as introductory source material for a modern course in differential geometry. The subject is traditionally called *Calculus of Several Variables*, *Vector Calculus*, or *Multivariable Calculus*. The usual content is

- *Preliminary Theory of Vectors*: Dot and Cross Products; Vectors, Lines, and Planes in $\mathbb{R}^3$.
- *Vector-Valued Functions*: Derivatives and Integrals of Vector-Valued Functions of One Variable; Space Curves; Tangents and Normals; Arclength and Curvature.
- *Partial Derivatives*: Directional Derivatives; Gradients; Surfaces; Tangent Planes; Multivariable Max/Min Problems; Lagrange Multipliers.
- *Multiple Integrals*: Double and Triple Integrals; Cylindrical and Spherical Coordinates; Change of Variables.
- *Calculus of Vector Fields*: Line and Surface Integrals; Fundamental Theorem of Line Integrals; Green's, Stokes', and Divergence Theorems.

Our goal is to modernize the course in two important ways. First, we adopt a modern view, which emphasizes geometry as much as analysis. Second, we introduce the mathematical software system MATLAB as a powerful computational and visual tool. We include and emphasize MATLAB in order to

- remove the drudgery from tedious hand calculations that can now be done easily by computer;
- improve students' understanding of fundamental concepts in the traditional syllabus;

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_1

- enhance students' appreciation of the beauty and power of the subject by incorporating dramatic visual evidence; and
- introduce new geometrical and physical topics.

1.1 Benefits of Mathematical Software

To elaborate, we describe some specific benefits that follow from introducing MATLAB. First, the traditional multivariable calculus course has a tremendous geometric component. Students struggle to handle it. Unless they are endowed with artistic gifts or uncanny geometric insight, they may fail to depict and understand the geometric constructs. Often, they rely on illustrations in their text or prepared by their instructor. While the quality of those illustrations may be superior to what they can generate themselves, spoon-fed instruction does not lead to the same depth of understanding as self-discovery. Providing a software system like MATLAB enables all students to draw, manipulate, and analyze the geometric shapes of multivariable calculus.

Second, most of the numbers, formulas, and equations found in standard problems are highly contrived to make the computations tractable. This places an enormous limitation on the faculty member trying to present meaningful applications, and lends an air of untruthfulness to the course. (Think about the limited number of examples for an arclength integral that can be integrated easily in closed form.) With the introduction of MATLAB, this drawback is ameliorated. The numerical and symbolic power of MATLAB greatly expands our ability to present realistic examples and applications.

Third, the instructor can concentrate on non-rote aspects of the course. The student can rely on MATLAB to carry out the mundane algebra and calculus that often absorbed all of the student's attention previously. The instructor can focus on theory and problems that emphasize analysis, interpretation, and creative skills. Students can do more than crank out numbers and pictures; they can learn to explain coherently what the pictures mean. This capability is enhanced by either of the MATLAB environments in which students will work—a published MATLAB script (formerly called a script M-file) or a Live Script (created in the Live Editor). Either will afford the student the capability to integrate MATLAB commands with output, graphics, and textual commentary.

Fourth, the instructor has time to introduce modern, meaningful subject matter into the course. Because we can rely on MATLAB to carry out the computations, we are free to emphasize the ideas. In this book, we concentrate on aspects of geometry and physics that are truly germane to the study of multivariable calculus. With the introduction of MATLAB, this material can, for the first time, be presented effectively at the sophomore level.

1.2 What's in This Book

The bulk of the book consists of nine chapters (numbered 2–10) on multivariable calculus and its applications. Some chapters cover standard material from a non-standard point of view; others discuss topics that are hard to address without using a computer and mathematical software, such as numerical methods.

Each chapter is accompanied by a problem set. The problem sets constitute an integral part of the book. Solving the problems will expose you to the geometric, symbolic, and numerical features of multivariable calculus. Many of the problems (especially in Problem Sets C–J) are not routine.

Each problem set concludes with a Glossary of MATLAB commands, accompanied by a brief description, which are likely to be useful in solving the problems in that set. A more complete Glossary, with examples of how to use the commands, is included in our website at

`http://schol.math.umd.edu/MVCwMATLAB/`

where you can also find

- electronic versions of the sample problem solutions;
- MATLAB scripts for each chapter, containing the MATLAB input lines that recreate all of the output and figures from that chapter;
- the special MATLAB function scripts discussed in the book.

1.2.1 Chapter Descriptions

This Chapter, and Problem Set A, *Review of One-Variable Calculus*, describe the purpose of the book and its prerequisites. The Problem Set reviews both the elementary MATLAB commands and the fundamental concepts of one-variable calculus needed to use MATLAB to study multivariable calculus.

Chapter 2, *Vectors and Graphics*, and Problem Set B, *Vectors and Graphics*, introduce the mathematical idea of vectors in the plane and in space. We explain how to work with vectors in MATLAB and how to graph curves and surfaces in space.

Chapter 3, *Geometry of Curves*, and Problem Set C, *Curves*, examine parametric curves, with an emphasis on geometric invariants like speed, curvature, and torsion, which can be used to study and characterize the nature of different curves.

Chapter 4, *Kinematics*, and Problem Set D, *Kinematics*, apply the theory of curves to the physical problems of moving particles and planets.

Chapter 5, *Directional Derivatives*, and Problem Set E, *Directional Derivatives*, introduce the differential calculus of functions of several variables, including partial derivatives, directional derivatives, and gradients. We also explain how to graph functions and their level curves or surfaces with MATLAB.

Chapter 6, *Geometry of Surfaces*, and Problem Set F, *Surfaces*, study parametric surfaces, with an emphasis on geometric invariants, including several forms of curvature, which can be used to characterize the nature of different surfaces.

Chapter 7, *Optimization in Several Variables*, and Problem Set G, *Optimization*, discuss how calculus can be used to develop numerical algorithms for finding maxima and minima of functions in several variables, or to solve systems of equations in several variables. We also explain how to use MATLAB to test and apply these algorithms in concrete problems.

Chapter 8, *Multiple Integrals*, and Problem Set H, *Multiple Integrals*, develop the integral calculus of functions of several variables. We show how to use MATLAB to set up multiple integrals, as well as how to evaluate them. This chapter contains an introduction to and discussion of numerical methods for multiple integrals, a topic which in standard mathematics textbooks only shows up in advanced courses in numerical analysis.

Chapter 9, *Multidimensional Calculus*, along with Problem Set I, *Multivariable Calculus*, covers the usual topics of "vector analysis" found in most multivariable calculus texts. These include: the *Fundamental Theorem of Line Integrals*, *Green's Theorem*, the *Divergence Theorem*, and *Stokes' Theorem*. The treatment here is specially adapted for use with MATLAB.

Chapter 10, *Physical Applications of Vector Calculus*, and Problem Set J, *Physical Applications*, develop the theories of gravitation, electromagnetism, and fluid flow, and then use them with MATLAB to solve concrete problems of practical interest.

Chapter 11, MATLAB *Tips*, gathers together the answers to many MATLAB questions that have puzzled our students. Read through this chapter at various times as you work through the rest of the book. If necessary, refer back to it when some aspect of MATLAB has you stumped.

The *Sample Solutions* contain sample solutions to one or more problems from each Problem Set. These samples can serve as models when you are working out your own solutions to other problems.

Finally, we have a comprehensive *Index* of MATLAB commands and mathematical concepts that are found in this book.

This book is accompanied by a website

`http://schol.math.umd.edu/MVCwMATLAB/`

where you can find the following:

The *Glossary* includes all the commands from the Problem Set glossaries—together with illustrative examples—plus some additional entries.

The *Sample Solutions* contains the code from the printed sample solutions in the book. You might find this code useful when working out your own solutions to other problems.

The *Scripts* section contains the code from the book chapters, including the function scripts discussed there. These scripts could be useful for working some of the problems.

1.3 What's Not in This Book

This book is not a totally self-contained introduction to multivariable calculus. Specifically, it does not have a bank of "routine" multivariable calculus problems. These are omitted for two reasons: (i) such problems are easily accessible on the Internet; and (ii) there is now little point in excessive drill in routine multivariable calculus problem-solving, as MATLAB can handle such problems quickly and accurately. Moreover, we have included the topics and methods of multivariable calculus that are most important and interesting in mathematics and physics; and de-emphasized, or even omitted, routine matters and topics that are easily obtained elsewhere. We believe that the dedicated instructor can teach a course in multivariable calculus using this book as the sole text, supplemented by material from the Internet. Or, as earlier stated, the book can be used as an intense supplement to a traditional multivariable calculus text.

Neither is this book a self-contained introduction to MATLAB. We assume that you can learn the basics of MATLAB elsewhere. Since we cannot refer to a "standard text" for this purpose, here is a detailed description of prerequisites, along with some suggestions about how to come "up to speed" with the software.

This book requires access to MATLAB along with its accessory, the *Symbolic Math Toolbox*. This toolbox, along with a lot more (such as the simulation software Simulink® and the *Statistics and Machine Learning Toolbox* for statistical applications), is included in the MATLAB Student Suite, currently priced at $99 (in the US). If you are a student and are planning to take other science and engineering courses, this is your most cost-efficient option. Alternatively, you can get just MATLAB and the Symbolic Math Toolbox for the student price of about $80. If you are not a student, you can still get the Home version of MATLAB for non-commercial use for $149. The majority of college and university computer labs should have MATLAB and the Symbolic Math Toolbox already installed and ready to use.

If you have not used MATLAB before, our suggestion is to start by watching the tutorial videos, which you can find on the MathWorks® website at

`http://www.mathworks.com/support/`

and at

`https://matlabacademy.mathworks.com/`.

You might also enjoy reading the free e-books [6, 7] by Cleve Moler, the developer of MATLAB.

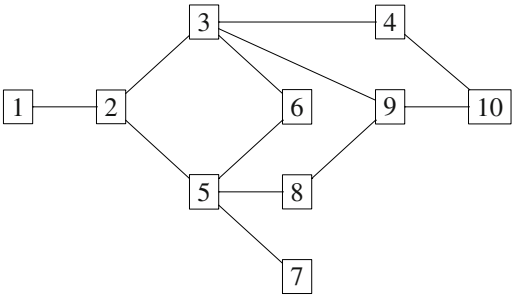
Once you have gone through the "Getting Started with MATLAB" and "MATLAB Overview" videos, plus other tutorials of your choice, try working with the software by yourself to get the hang of it. If you want additional help, there are many good books available. We, of course, are biased and think that the best of these is [5] by Brian Hunt and the two of us. Other options include [1–4, 8, 9], cited in the bibliography below.

Remember that MATLAB comes with very extensive documentation and help. Typing **`help`** followed by the name of a command in the Command Window gives you concise help on that command. For more extensive help, including examples and lists of possible options, use the help browser that comes with the software, which you can call up with the button that looks like (?), or else the website

`http://www.mathworks.com/help/` .

1.4 How to Use This Book

If this book is used as a stand-alone text, then most (but not all) of the material can be covered in a single semester. If the book supplements a traditional text, then it contains more material than can be covered in a single semester. To aid in selecting a coherent subset of the material, here is a diagram showing the dependence among the chapters:



We suggest that you work all the problems in Problem Set A, read Chapter 2, and then work at least a quarter to a half of Problem Set B. After that, various combinations of chapters are possible. Here are a few selections that we have found suitable for a one-semester multivariable calculus course

- *Geometry Emphasis:* Chapters 3, 5, and 6 with Problem Sets C, E, and F. If time permits, you could include portions of Chapter 4 and Problem Set D, or Chapter 8 and Problem Set H.

- *Physical Applications Emphasis:* Chapters 3, 4, 9, and 10 with Problem Sets C, D, I, and J. If time permits, it is desirable to add parts of Chapters 5 and 8 with Problem Sets E and H.
- *Calculus and Numerical Methods Emphasis:* Chapters 5, 7, 8, and 9 with Problem Sets E, G, H, and I.

Note that Chapter 11 does not appear on the above flowchart and does not need to be formally assigned, and instead is intended for help if and when you get stuck doing something with MATLAB.

In a problem seminar, mathematical methods course, or a differential geometry course more flexibility is possible, and one could choose a greater variety of problems from various chapters.

Beginning with Problem Set C, the exercises in the Problem Sets become fairly substantial; it is easy to spend an hour on each problem. To ease the burden, we often allow students to collaborate on the problems in groups of two or three. We ask each collaborating team to turn in a single joint assignment. This system fosters teamwork, builds confidence, and makes the harder problems manageable.

The problems in this book have been classroom-tested according to two different schemes. Problems can be assigned in big chunks, as projects to be worked on three or four times during the term. Alternatively, problems can be assigned one or two at a time on a more regular basis. Both methods work; which works better depends on the backgrounds of instructor and students, and whether or not you want to combine the material from this book with assignments from a standard textbook.

1.5 The MATLAB Interface

You may interact with MATLAB in one of three distinct ways

- **Command Window**. This is where you type (or enter) MATLAB commands (such as `syms x; solve(x^2 - x - 1 == 0, x)`) at the prompt. In the default layout, the Command Window appears in the lower middle of your MATLAB Desktop, and MATLAB responds just below each command with the resultant output.
- **Scripts**. This is where you construct (in the *Editor Window*) a series of commands, and likely comment lines (preceded by the percent symbol), in a file or script, which you execute via the **Run** button. The output appears in the Command Window. Think of scripts as little programs that instruct MATLAB as to what to compute. You may save and edit your scripts and so run the same (or edited) script numerous times. MATLAB has a *Publish* feature, which will run your script, assemble the input and output in a logical fashion, and "publish" the result in an integrated, readable format. You have several choices as to the format; `html` (the default) and `pdf` are the most popular choices.

- **Live Editor**. This is where the output of your script commands are integrated, from the start, with the corresponding input in a window in which you can do live editing. You alter a command (with your mouse or keyboard), evaluate, and the output is automatically updated. You can intersperse comments between different output cells. This is like simultaneous script construction (i.e., programming) and publishing. Either this method or the preceding is extremely handy for transmitting (e.g., to an instructor) the results of your MATLAB investigations.

1.5.1 A Word on Terminology

Until very recently, a MATLAB script was called an M-file. There were two flavors: a script M-file—exactly what we are calling a *script*; and a function M-file—a slightly different format in which a function is defined. The latter are still valid, but they are just called function scripts. Contrast the latter with *anonymous functions*, which are inputted at the command line via the `@` construct.

For more on interface, terminology and MATLAB basics and protocols, see the online help or any of the references listed below.

1.6 Software Versions

MATLAB and its accompanying products (such as the Symbolic Math Toolbox) are constantly being updated. We prepared this book with versions R2016b and R2017a. With each revision, the syntax for some commands may change, and occasionally an old command is replaced by a new one with another name. For example, the MATLAB command **`integral`** for numerical integration was only introduced in version R2012a; before that, the relevant command was **`quadl`**. Live Editor scripts were only introduced in R2016a. In the latest versions of MATLAB, **`fplot3`** has replaced **`ezplot3`** and the new command **`fimplicit3`** was introduced. We mention this because different readers of this book will undoubtedly be using different versions of MATLAB. For more than 90% of what we discuss, this will make no difference. But occasionally you may notice that we refer to a command that has either been superseded or does not exist in the version you are using. In almost all cases, searching the documentation should enable you to find a close equivalent. When major changes occur, we will discuss them on the book website.

Problem Set A. Review of One-Variable Calculus

All problems should be solved in MATLAB, preferably by creating a script combining text and MATLAB commands, and then "publishing" the result. Alternatively, you may use the *Live Editor* to create a polished *Live Script* that contains your commands, output, graphs, and commentary. All of your explanations should be well organized and clearly presented in text cells. You should use the options available with the plotting routines to enhance your plots. See the *Sample Solutions* on the website for examples of what the result should look like.

Problem 1.1. Graph the following transcendental functions using **`fplot`**. Use your judgment and some experimentation to find an appropriate range of values of x so that the "main features" of the graph are visible.

(a) $\sin x$.

(b) $\tan x$.

(c) $\ln x$. (Remember that the natural logarithm is called **`log`** in MATLAB.) Hint: The logarithm is singular for $x = 0$ and undefined for $x < 0$. Allow your range of values to start at $x = 0$ and then at some value $x < 0$. How well does MATLAB cope?

(d) $\sinh x$.

(e) $\tanh x$.

(f) e^{-x^2}.

Problem 1.2. Let $f(x) = x^3 - 4x^2 + 1$.

(a) Graph f. First, let MATLAB choose the interval; then re-graph on $(-\frac{1}{2}, \frac{1}{2})$. (You will see why in part (d) below.)

(b) Use **`solve`** to find (in terms of square and cube roots) the exact values of x where the graph crosses the x-axis. Don't be surprised if the answers are complicated and involve complex numbers. *Hint*: You may find the option **`MaxDegree`** to be helpful.

(c) Use **`double`** to convert your values from (b) to numerical values $x_0 < x_1 < x_2$. Compare with what you get by using **`fzero`** to find the three real zeros of f numerically.

(d) Compute the exact value of the area of the bounded region lying below the graph of f and above the x-axis. (This is the region where $x_0 \le x \le x_1$.) Then convert this exact expression to a numerical value and explain in terms of your picture in (a) why the answer is reasonable.

(e) Determine where f is increasing and where it is decreasing. Use **`solve`** to find the exact values of x where f has a relative maximum or relative minimum point.

(f) Find numerical values of the coordinates of the relative maximum points and/or relative minimum points on the graph.

(g) Determine where the graph of f is concave upward and where it is concave downward.

Problem 1.3. Consider the equation $x\sin x = 1$, for x a positive number.

(a) By graphing the function $x\sin x$, find the approximate location of the first five solutions.

(b) Why is there a solution close to $n\pi$ for every large positive integer n? (Hint: For large x, the reciprocal $1/x$ is very close to 0. Where is $\sin x = 0$?)

(c) Use **`fzero`** to refine your approximate solutions from (a) to get numerical values of the true solutions (good to at least several decimal places).

Problem 1.4. Compute the following limits:

(a) $\lim_{x\to 1} \dfrac{x^2+3x-4}{x-1}$.

(b) $\lim_{x\to 0} \dfrac{\sin x}{x}$.

(c) $\lim_{x\to 0_+} x\ln x$.

Problem 1.5. Compute the following derivatives:

(a) $\dfrac{d}{dx}\left(x^2+5x-1\right)^{100}$.

(b) $\dfrac{d}{dx}\left(\dfrac{x^2e^x-1}{x^2+2}\right)$.

(c) $\dfrac{d}{dx}\left(\sin^5 x\cos^3 x\right)$.

(d) $\dfrac{d}{dx}\arctan(e^x)$.

Problem 1.6. Compute the following integrals. Whenever possible, find an exact expression using **`int`**. If MATLAB cannot compute a definite integral exactly, compute a numerical value using **`integral`**. Alternatively, you can apply **`double`** to any mysterious symbolic antiderivative that you encounter.

(a) $\displaystyle\int (x^2+5x-1)^{10}(2x+5)\,dx$.

(b) $\displaystyle\int \arcsin(x)\,dx$.

(c) $\displaystyle\int_0^1 x^5(1-x^2)^{3/2}\,dx$.

(d) $\int_0^1 x^5(1-x)^{3/2}\,dx.$

(e) $\int_0^\infty x^5 e^{-x^2}\,dx.$

(f) $\int_0^1 e^{e^x}\,dx.$

(g) $\int e^x \cos^3 x \sin^2 x\,dx.$

(h) $\int_0^1 \sin\left(x^3\right)\,dx.$

Problem 1.7. Use the **`taylor`** command to find the Taylor series of the function $\sec(x)$ around the point $x = 0$, up to and including the term in x^{10}. Check the documentation on **`taylor`** to see how to get the right number of terms of the series; the default is to go out only to the term in x^4.

Problem 1.8. The alternating harmonic series

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} + \cdots$$

is known to converge (slowly!!) to $\ln 2$.

(a) Test this by adding the first 100 terms of the series and comparing with the value of $\ln 2$. Do the same with the first 1000 terms.

(b) The alternating series test says that the error in truncating an alternating series (whose terms decrease steadily in absolute value) is less than the absolute value of the last term included. Check this in the situation of (a). In other words, verify that the difference between $\ln 2$ and the sum of the first 100 terms of the series is less than $\frac{1}{100}$ in absolute value, and that the difference between $\ln 2$ and the sum of the first 1000 terms of the series is less than $\frac{1}{1000}$ in absolute value.

Problem 1.9. The harmonic series

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots$$

diverges (slowly!!), by the integral test. This test also implies that the sum of the first n terms of the series is approximately $\ln n$. Test this by adding the first 100 terms of the series and comparing with the value of $\ln 100$, and by adding the first 1000 terms of the series and comparing with the value of $\ln 1000$. Do you see any pattern? If so, test it by replacing 1000 by 10000.

Problem 1.10. The power series

$$\sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(n!)^2}$$

converges for all x to a function $f(x)$.

(a) Let $f_k(x)$ be the sum of the series out to the term $n = k$. Graph $f_k(x)$ for $-8 < x < 8$ with $k = 10, 20, 40$. Superimpose the last two plots. Why is the plot of $f_{40}(x)$ for $-8 < x < 8$ visually indistinguishable from the plot of $f_{20}(x)$ over the same domain?

(b) Apply the **`symsum`** command in MATLAB to the infinite series. You should find that MATLAB recognizes $f(x)$. (However, it may not be a function you saw in your one-variable calculus class.) Plot $f(x)$ for $-8 < x < 8$ and compare with your plot of $f_{40}(x)$ as a means of checking your answer to (a).

Problem 1.11. Use the **`ezpolar`** command in MATLAB to graph the following equations in polar coordinates

(a) $r = \sin\theta$.

(b) $r = \sin(6\theta)$.

(c) $r = 4\sin\theta - 2$.

(d) $r^2 = \sin 2\theta$.

Glossary of MATLAB Commands

`axis` Selects the ranges of x and y to show in a plot

`diff` Computes the derivative

`double` Converts the (possibly symbolic) expression for a number to a numerical (double-precision) value

`ezpolar` Easy plotter in polar coordinates

`figure` Start a new graphic

`fplot` Easy function plotter

`fzero` Finds (numerically) a zero of a function near a given starting value

`hold on` Retain current figure; combine new graph with previous one

`int` Computes the integral (symbolically)

`integral` Integrates numerically

`limit` Computes the limit

`pretty` Displays a symbolic expression in mathematical style

solve Symbolic equation solver

subs Substitute for a variable in an expression

syms Set up one or more symbolic variables

symsum Finds the (symbolic) sum of a series

taylor Computes the Taylor polynomial; use the optional argument **Order** to reset the order

Options to MATLAB Commands

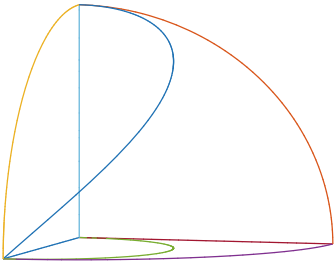
MaxDegree An option to **solve**, giving the maximum degree of polynomials for which MATLAB will try to find explicit solution formulas

Order An option to **taylor**, specifying the degree of the Taylor approximation

References

1. S. Attaway, *MATLAB: A Practical Introduction to Programming and Problem Solving*, 4th edn. (Elsevier, 2016)
2. W. Gander, *Learning MATLAB: A Problem Solving Approach* (Springer, 2015)
3. A. Gilat, *MATLAB: An Introduction with Applications*, 5th edn. (Wiley, 2015)
4. B. Hahn, D. Valentine, *Essential MATLAB for Engineers and Scientists*, 5th edn. (Academic Press, 2013)
5. B. Hunt, R. Lipsman, J. Rosenberg, *A Guide to MATLAB: For Beginners and Experienced Users*, 3rd edn. (Cambridge Univesity Press, 2014)
6. C. Moler, *Numerical Computing with MATLAB* (2004), http://www.mathworks.com/moler/chapters.html
7. C. Moler, *Experiments with MATLAB* (2011), http://www.mathworks.com/moler/exm/
8. H. Moore, *MATLAB for Engineers*, 5th edn. (Pearson, 2018)
9. R. Pratap, *Getting Started with MATLAB*, 7th edn. (Oxford University Press, 2016)

Chapter 2
Vectors and Graphics



We start this chapter by explaining how to use vectors in MATLAB, with an emphasis on practical operations on vectors in the plane and in space. Remember that n-dimensional vectors are simply ordered lists of n real numbers; the set of all such is denoted by $\mathbb{R}^n$. We discuss the standard vector operations, and give several applications to the computations of geometric quantities such as distances, angles, areas, and volumes. The bulk of the chapter is devoted to instructions for graphing curves and surfaces.

2.1 Vectors

In MATLAB, vectors are represented as lists of numbers or variables. You write a list in MATLAB as a sequence of entries encased in square brackets. Thus, you would enter `v = [3, 2, 1]` at the prompt to tell MATLAB to treat `v` as a vector with x, y, z coordinates equal to 3, 2, and 1, respectively.

You can perform the usual vector operations in MATLAB: vector addition, scalar multiplication, and the dot product (also known as the inner product or scalar product). Here are some examples, in which we use semicolons to suppress output:

```
>> a = [1, 2, 3];
>> b = [-5, -3, -1];
>> c = [3, 0, -2];
>> a + b
ans =
    -4    -1     2

>> 5*c
ans =
    15     0   -10
```


R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_2

```
>> dot(a, b)
ans =
   -14
```

The **dot(a, b)** command computes the dot product of the vectors **a** and **b** (the sum of the products of corresponding entries). As usual, you can use the dot product to compute lengths of vectors (also known as vector norms).

```
>> lengthofa = sqrt(dot(a, a))
ans =
    3.7417
```

Actually, MATLAB has an internal command that automates the numerical computation of vector norms.

```
>> [norm(a), norm(b), norm(c)]
ans =
    3.7417    5.9161    3.6056
```

Our attention throughout this book will be directed to vectors in the plane and vectors in (three-dimensional) space. Vectors in the plane have two components; a typical example in MATLAB is **[x, y]**. Vectors in space have three components, like **[x, y, z]**. The following principle will recur: *Vectors with different numbers of components do not mix.* As you will see, certain MATLAB commands will only work with two-component vectors; others will only work with three-component vectors. To convert a vector in the plane into a vector in space, you can add a zero to the end.

```
>> syms x y; p1 = [x, y]
p1 =
[ x, y]

>> s1 = [p1, 0]
s1 =
[ x, y, 0]
```

To project a vector in space into a vector in the *x*-*y* plane, you simply drop the final component.

```
>> syms x y z; s2 = [x, y, z]
s2 =
[ x, y, z]

>> p2 = s2(1:2)
p2 =
[ x, y]
```

The first place we notice a difference in the handling of two- or three-dimensional vectors is with the cross product, which only works on a pair of three-dimensional

vectors. To compute the cross product in MATLAB, you can use the **cross** command. For example, to compute the cross product of the vectors **a** and **b** defined above, simply type

```
>> cross(a, b)

ans =
     7   -14     7
```

Note that the cross product is *anti-symmetric*; reversing the order of the inputs changes the sign of the output.

2.1.1 Applications of Vectors

Since MATLAB allows you to perform all the standard operations on vectors, it is a simple matter to compute lengths of vectors, angles between vectors, distances between points and planes or between points and lines, areas of parallelograms, and volumes of parallelepipeds, or any of the other quantities that can be computed using vector operations. Here are some examples.

2.1.1.1 The Angle Between Two Vectors

The fundamental identity involving the dot product is

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos(\varphi),$$

where φ is the angle between the two vectors. So, we can find the angle between **a** and **b** in MATLAB by typing

```
>> phi = acos(dot(a, b)/(norm(a)*norm(b)))

phi =
   2.2555
```

We can convert this value from radians to degrees by typing

```
>> phi*180/pi

ans =
   129.2315
```

2.1.1.2 The Projection Formula

One of the most useful applications of the dot product is for computing the projection of one vector in the direction of another. Given a nonzero vector **b**, we can always write another vector **a** uniquely in the form $\text{proj}_{\mathbf{b}}(\mathbf{a}) + \mathbf{c}$, where $\mathbf{c} \perp \mathbf{b}$

and $\text{proj}_{\mathbf{b}}(\mathbf{a})$ is a scalar multiple of $\mathbf{b}$. To find the formula for the projection, write $\text{proj}_{\mathbf{b}}(\mathbf{a}) = x\mathbf{b}$ and take the dot product with $\mathbf{b}$. We obtain

$$\mathbf{a} \cdot \mathbf{b} = x\,\mathbf{b} \cdot \mathbf{b} + \mathbf{c} \cdot \mathbf{b} = x\,\|\mathbf{b}\|^2 + 0.$$

Thus $x = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{b}\|^2}$ and $\text{proj}_{\mathbf{b}}(\mathbf{a}) = \frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{b}\|^2}\mathbf{b}$. By the way, even through the formula for $\text{proj}_{\mathbf{b}}(\mathbf{a})$ is nonlinear in $\mathbf{b}$, it is linear in $\mathbf{a}$. That is because the dot product is linear in each variable when the other variable is held fixed. Computing projections is easy in MATLAB. For example, with our given vectors, we obtain

```
>> (dot(a, b)/dot(b, b))*b

ans =
   2.0000    1.2000    0.4000
```

Exercise 2.1. Use MATLAB to compute the perpendicular component $\mathbf{c}$.

2.1.1.3 The Volume of a Parallelepiped

The volume of the parallelepiped spanned by the vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ is computed using the formula $V = |\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})|$. In MATLAB, it is easy to enter this formula.

```
>> abs(dot(a, cross(b, c)))

ans =
   7
```

2.1.1.4 The Area of a Parallelogram

The volume formula becomes an area formula if you take one of the vectors to be a unit vector perpendicular to the plane spanned by the other two vectors. In particular, given vectors $\mathbf{a}$ and $\mathbf{c}$, $\mathbf{a} \times \mathbf{c}$ is perpendicular to both of them, so a unit vector perpendicular to both $\mathbf{a}$ and $\mathbf{c}$ is $\frac{\mathbf{a}\times\mathbf{c}}{\|\mathbf{a}\times\mathbf{c}\|}$ and the area of the parallelogram spanned by $\mathbf{a}$ and $\mathbf{c}$ is just

$$\left| \frac{\mathbf{a} \times \mathbf{c}}{\|\mathbf{a} \times \mathbf{c}\|} \cdot (\mathbf{a} \times \mathbf{c}) \right| = \frac{\|\mathbf{a} \times \mathbf{c}\|^2}{\|\mathbf{a} \times \mathbf{c}\|} = \|\mathbf{a} \times \mathbf{c}\|,$$

which you compute in MATLAB by typing

```
>> norm(cross(a, c))

ans =
   13.1529
```

In a similar fashion, you can use the standard mathematical formulas, expressed in MATLAB syntax, to

(i) project a vector onto a line or a plane;

(ii) compute the distance from a point to a plane;
(iii) compute the distance from a point to a line; and
(iv) check that lines and planes are parallel or perpendicular.

2.2 Parametric Curves

We assume that you already know how to use MATLAB's plotting commands to graph plane curves of the form $y = f(x)$. In fact, you can do so using either MATLAB's symbolic plotting command **fplot** or its numerical plotting command **plot**. The analogs are **fplot3** (symbolic) and **plot3** (numerical) for curves in space. There are also analogs, as we shall see, for surfaces in space, $z = f(x, y)$, namely **fsurf** or **fmesh** (symbolic) and **surf** or **mesh** (numerical). (*Note*: The symbolic **f** commands replaced the **ez** commands in MATLAB as of version R2016a.)

Now it is very common for both curves (in the plane or in space) and surfaces to be specified by parametric equations, rather than explicitly as just described. As we shall see, the same commands as above can be used to graph them—albeit with slightly different syntax. In this section, we will explain how to graph curves defined by parametric equations, both in the plane and in space, and also how to graph surfaces defined by parametric equations. But let us note: Henceforth in this book, and in the spirit of its subject matter, *we shall use symbolic plotting commands whenever feasible*. We shall resort to numerical plotting routines only when absolutely necessary.

For illustrative purposes, consider the parametrized unit circle

$$x = \cos t,\ y = \sin t,\ 0 \le t \le 2\pi.$$

These parametric equations mean that t is a parameter (ranging through the interval $[0, 2\pi]$), and that associated to each value of t is a pair (x, y) of values defined by the given formulas. As t varies, the points $(x(t), y(t))$ trace out a curve in the plane. Now let us draw the curve, first numerically, then symbolically

```
>> T = 0:0.1:2*pi; plot(cos(T), sin(T)); axis square
>> syms t; figure; fplot(cos(t), sin(t), [0, 2*pi]); axis square
```

Both commands result in the graph in Figure 2.1, although the tick marks differ slightly. Note that we inserted the command **figure** in the second instruction. Without it, the second circle would be superimposed on the first instead of it being created in a second graph.

Now let us start in earnest on parametric curves and surfaces by looking first at the spiral plane curve defined parametrically by the equations

$$x = e^{-t/10}(1 + \cos t), \quad y = e^{-t/10} \sin t, \quad t \in \mathbb{R}.$$

```
>> fplot(exp(-t/10)*(1 + cos(t)), exp(-t/10)*(sin(t)), [0 2*pi])
```

Figure 2.2 shows the resulting graph of the portion of the curve that results when the parameter runs from 0 to 2π.

Next we draw a larger portion of the curve.

```
>> fplot(exp(-t/10)*(1 + cos(t)), exp(-t/10)*(sin(t)), [0 20*pi])
```

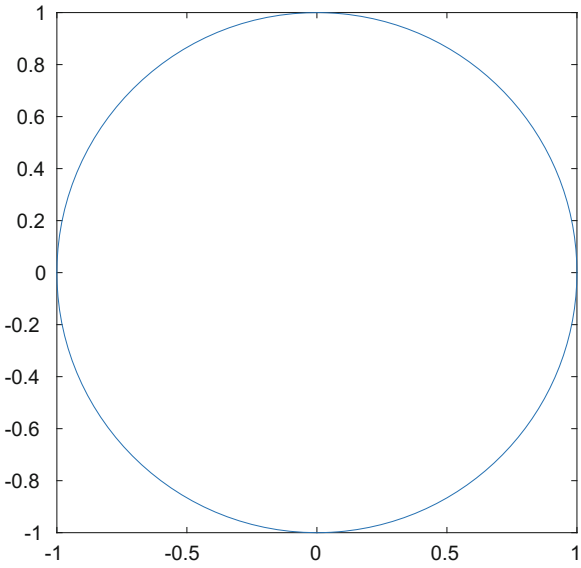


Fig. 2.1 The Unit Circle

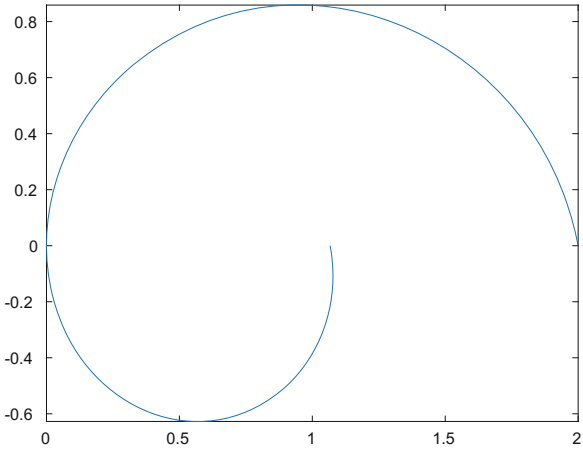


Fig. 2.2 Spiral Curve—One Rotation

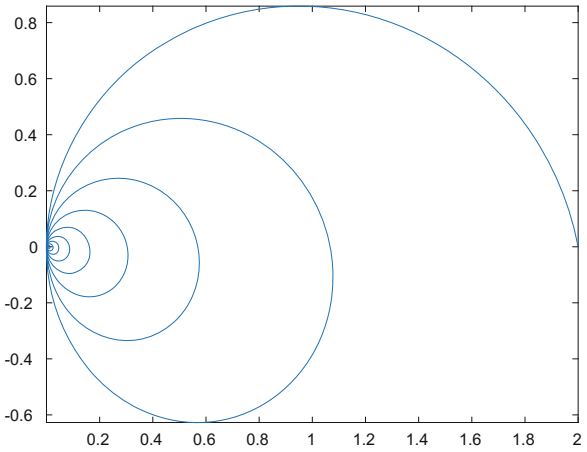


Fig. 2.3 Spiral Curve—Ten Rotations

Figure 2.3 shows the resulting graph of the portion of the curve that results when the parameter runs from 0 to 20π.

You can adjust this graph in various ways by using the **`axis`** command with various options. (See MATLAB online help for suggestions.)

Next let us look at a space curve defined by a set of parametric equations. As our example, we will take Viviani's curve, which is defined as the intersection of the sphere $x^2 + y^2 + z^2 = 4$ with the cylinder $(x-1)^2 + y^2 = 1$. The projection of Viviani's curve into the x-y plane is defined by the same equation that defines the cylinder; thus, it is a circle that has been shifted away from the origin. We can parametrize this circle in the plane by taking

$$x = 1 + \cos t, \quad y = \sin t, \quad -\pi \le t \le \pi.$$

Using this parametrization to solve the sphere's equation for z, we find that

$$\begin{aligned} z &= \pm\sqrt{4 - x^2 - y^2} \\ &= \pm\sqrt{4 - (1+\cos t)^2 - \sin^2 t} \\ &= \pm\sqrt{2 - 2\cos t} \\ &= \pm 2\sin\left(\frac{t}{2}\right). \end{aligned}$$

Now we will define Viviani's curve in MATLAB. After that, we will use **`fplot3`** to graph the part of the curve in the first octant.

```
>> syms t; x = 1 + cos(t); y = sin(t), z = 2*sin(t/2);
>> fplot3(x, y, z, [0 pi])
```

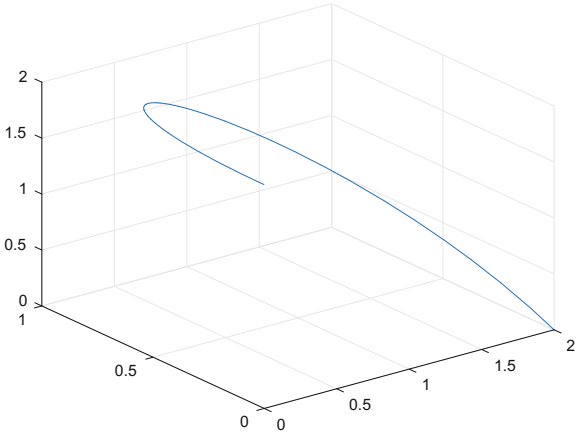


Fig. 2.4 Viviani's Curve—First Drawing

Figure 2.4 is not terribly illuminating; let us see if we can improve it. We shall do so by getting rid of the grid that, by default, surrounds all three-dimensional graphics in MATLAB. We shall also dispense with the labels and tick marks on the axes. It would also be nice to show the curve inside the sphere and cylinder that are used to define it. Therefore, we will show the arcs obtained by intersecting the sphere with the coordinate planes. Finally, we will include the projected circle in the *x*-*y* plane. The reader is encouraged to examine the code carefully to see how we achieved these objectives.

```
>> hold on
>> fplot3(sym(0), 2*cos(t/2), 2*sin(t/2), [0, pi])
>> fplot3(2*cos(t/2), sym(0), 2*sin(t/2), [0, pi])
>> fplot3(2*cos(t/2), 2*sin(t/2), sym(0), [0 pi])
>> fplot3(1+cos(t), sin(t), sym(0), [0, pi])
>> fplot3(sym(0), sym(0), t, [0 2]);
>> fplot3(sym(0), t, sym(0), [0 2]);
>> fplot3(t, sym(0), sym(0), [0 2])
>> title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis off;
```

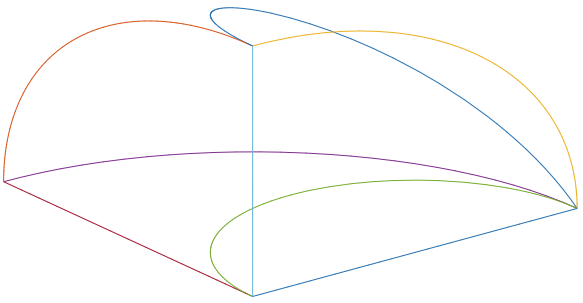


Fig. 2.5 Viviani's Curve—Second Drawing

Note that, since **fplot3** requires symbolic input, we had to specify the numerical input "0" as **sym(0)**. (Alternatively, we could have used function handles format for the input.) The picture still isn't very good; however, a minor change will improve it significantly. The main problem is the viewpoint (Figure 2.5) from which MATLAB has chosen to show us the graph. The default viewpoint is not in the first octant. This viewpoint is chosen generically, to make it unlikely that significant features of a random graph will be obscured. It has the definite disadvantage, however, that it changes the apparent directions of the x- and y-axes when compared with most mathematical textbooks. Choosing all positive values for the viewpoint will put your viewpoint into the first octant with the axes proceeding in the usual directions. In this case, we will make the change **view([10, 3, 1])** to get a better look at the graph. The result is shown in Figure 2.6. (In many cases, **[1, 1, 1]** is a good choice; you may need to experiment to find the best viewpoint.)

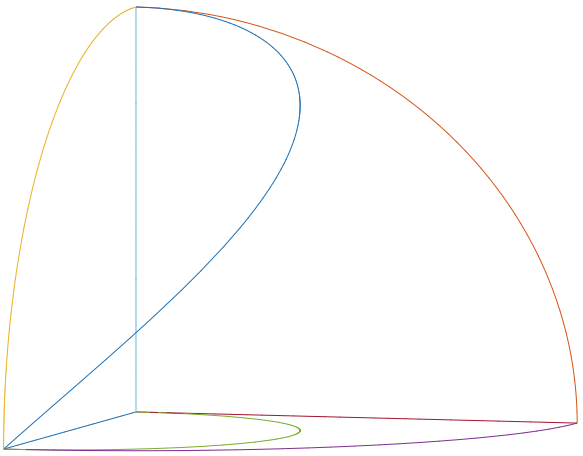


Fig. 2.6 Viviani's Curve—Third Drawing

2.3 Graphing Surfaces

Our next goal is to learn how to graph surfaces that are defined by a single equation $z = f(x, y)$. There are two (symbolic) graphing commands we can use to do that, namely **fmesh** and **fsurf**. The first produces a transparent mesh surface, the latter an opaque shaded one. We will illustrate them both. There are of course numerical analogs **mesh** and **surf**. We'll leave it to the reader to explore those if so desired.

Now, let's look at a simple example. We'll plot the function $f(x, y) = 1 - (x^2 + y^2)$ on the square $-1 \leq x \leq 1$, $-1 \leq y \leq 1$ in Figure 2.7.

```
>> syms x y; figure; fmesh(1 - x^2 - y^2, [-1, 1, -1, 1])
```

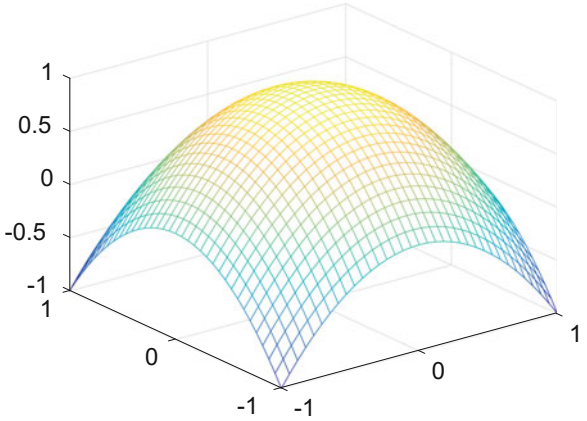


Fig. 2.7 A Paraboloid Above a Square

As in two dimensions, we can combine three-dimensional graphs via the **`hold on`** command. For example, let's draw a second surface over (see Figure 2.8) the same square.

```
>> hold on; fsurf(sin(6*x*y), [-1, 1, -1, 1])
>> title('Interwoven Surfaces')
```

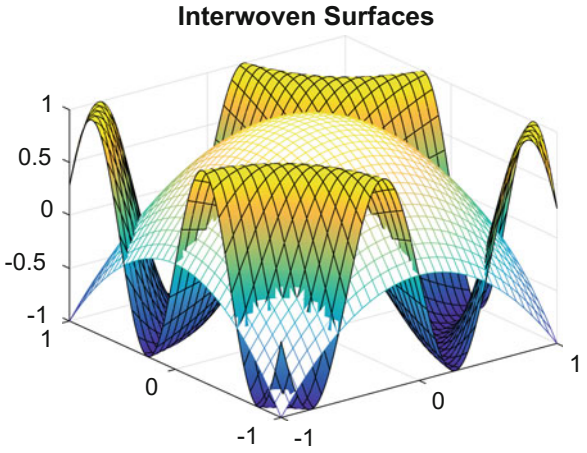


Fig. 2.8 A Paraboloid Interwoven with a Sinusoidal Surface

The **`fsurf`** command requires that the region in the *x*-*y* plane, over which the surface is plotted, must be a rectangle. Later on, we will see that it is helpful to be able to visualize portions of surfaces that lie over curved regions in the *x*-*y* plane.

This will be especially important when we study multiple integrals. In the meantime, here is a simple scheme to implement such a drawing, shown in Figure 2.9.

```
>> fsurf(sqrt(1-x^2-y^2)*heaviside(1-x^2-y^2), [-1, 1, -1, 1])
>> title('Hemisphere with Smooth Edges'), axis equal
```

You can consult the online help for more information on the **heaviside** function, but, essentially, **heaviside**(t) is 1 when $t > 0$ and vanishes otherwise.

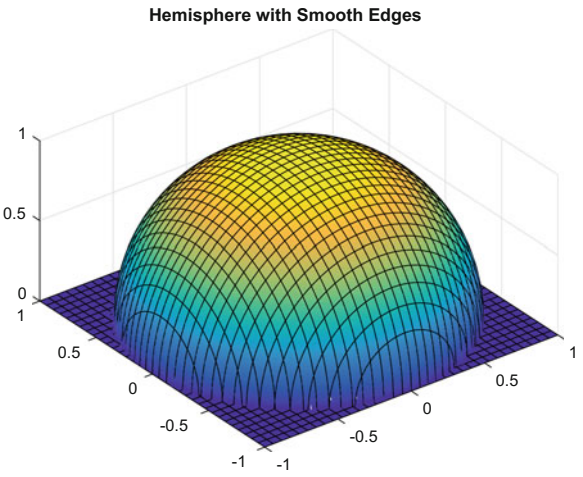


Fig. 2.9 The Hemisphere with no Rough Edges

2.4 Parametric Surfaces

The final topic is the plotting of surfaces that are defined by parametric equations. A parametric curve, being one-dimensional, depends on a single parameter. A parametric surface, being two-dimensional, requires us to use two parameters. Let's denote the parameters by u and v. For each parameter pair (u, v), we need to specify an associated point $(x(u, v), y(u, v), z(u, v))$ in space. As the parameter point (u, v) varies over its domain (which is some region in the plane), the associated point will trace out a surface in three-dimensional space. As an example, we will consider the surface whose parametric equations are

$$x = u^3, \quad y = v^3, \quad z = uv.$$

We often express this notion mathematically by writing

$$(x, y, z) = (u^3, v^3, uv).$$

In MATLAB, we can easily graph this surface. We will use some of the same "tricks" as above to render an uncluttered image, shown in Figure 2.10.

```
>> syms u v, fsurf(u^3, v^3, u*v, [-1, 1, -1, 1])
>> view([1, 1, 1]), title(''); xlabel(''); ylabel(''); zlabel('')
>> grid off; axis off;
```

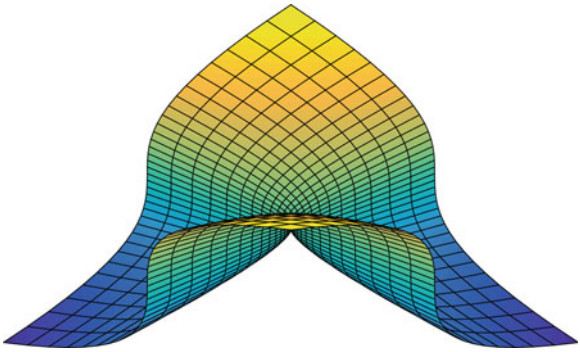


Fig. 2.10 A Cubic Surface

Standard methods for improving two-dimensional plots can also be employed for three-dimensional plots. For example, if your surfaces or curves look jagged or ill-defined, you can open the **Tools** tab in the menu bar of your plot window where you will find a host of tools for adjusting, enhancing and improving your graphs. Depending on your first picture, you may be able to estimate that the range you supplied (for the parameter or the base rectangle) should be adjusted. It is a simple matter to edit the input cell and then reinvoke the command. Your first picture will be superseded with a second picture (provided you do not have **`hold on`**). Labeling the axes, labeling the graph, changing the **`axis`** or **`view`**—these, and other techniques that you will learn to use as you produce more graphs, can greatly improve the quality of your plots.

Problem Set B. Vectors and Graphics

Problem 2.1. Find the distance between the two points $P = (0, 4.516, -5.298)$ and $Q = (-3.33, 0.234, 7.8)$.

Problem 2.2. Show that the point $P = (2, 0, 3)$ is equidistant from the two points $Q1 = (0.12, -1, 5.55)$ and $Q2 = (3.88, 1, 0.45)$.

Problem 2.3. Suppose that the points $P1 = (-2, -3, 5)$ and $P2 = (-6, 3, 1)$ are the endpoints of a diameter in a sphere. Find the equation of the sphere. Then compute the coordinates of any points on the line $z = 10y = -x$ which also lie on the surface of the sphere.

Problem 2.4. Let $\mathbf{a} = (2.9999, 400001, -6)$ and $\mathbf{b} = (0, -3.8765, 592320)$. Find $\mathbf{a} + \mathbf{b}$, $\|\mathbf{b}\|$ and $7\mathbf{a}$. Explain why $\|\mathbf{b}\|$ apparently equals the z-coordinate of $\mathbf{b}$, even though the y-coordinate is not zero.

Problem 2.5. Compute the angle (in degrees) that the vector $\mathbf{a} = -24.56\mathbf{i} + 44.689\mathbf{j}$ makes with the x-axis, measured counterclockwise from the x-axis.

Problem 2.6. Two tugboats are pulling a cruise ship. Tugboat 1 exerts a force of 1000 pounds on the ship and pulls in the direction 30 degrees north of due east. Tugboat 2 pulls in the direction 45 degrees south of due east. What force must Tugboat 2 exert to keep the ship moving due east?

Problem 2.7. Consider the vectors

$$\begin{aligned}\mathbf{a} &= (9, -3, 0.25),\\ \mathbf{b} &= (-3, -4, 60),\\ \mathbf{c} &= (-20.4, -6.2, 155.65).\end{aligned}$$

(a) Verify that $\mathbf{a}$ and $\mathbf{b}$ are perpendicular.

(b) The vector $\mathbf{c}$ lies in the same plane as $\mathbf{a}$ and $\mathbf{b}$. Resolve the vector $\mathbf{c}$ into its $\mathbf{a}$ and $\mathbf{b}$ components. Check your answer.

Problem 2.8. Find the angle (in degrees) between each of the following pairs of vectors:

(a) $\mathbf{a} = (2.467, -4.196, 0.433)$ and $\mathbf{b} = (-10.43, 9.344, 0)$.

(b) $\mathbf{a} = (-3.54, -10.79, 0.991)$ and $\mathbf{b} = (-1.398, 0, 6.443)$.

Problem 2.9. The following four vectors lie in a plane in 3-space. Do their endpoints determine a parallelogram? a rhombus? a square?

$$\mathbf{a} = (1, 1, 1), \qquad \mathbf{b} = (2, 3, 3), \qquad \mathbf{c} = (4, 2, 5), \qquad \mathbf{d} = (3, 0, 3).$$

Problem 2.10. In each of the following cases, find the cross product of the vectors $\mathbf{a}$ and $\mathbf{b}$, and then use it to find the angle between the two vectors.

(a) $\mathbf{a} = (-4.275, -2.549, 9.333), \quad \mathbf{b} = (6.302, -2.043, 0.444).$

(b) $\mathbf{a} = (77, 88, 99), \quad \mathbf{b} = (22, 44, 66).$

Problem 2.11. Prove the identity

$$\|\mathbf{a} \times \mathbf{b}\|^2 = \|\mathbf{a}\|^2 \|\mathbf{b}\|^2 - (\mathbf{a} \cdot \mathbf{b})^2$$

by assigning letter (i.e., variable) coordinates to **a** and **b** and evaluating both sides of the identity using MATLAB.

Problem 2.12. In this problem, we study the volumes of parallelepipeds.

(a) Find the volume of the parallelepiped determined by the three vectors:

$$\mathbf{a} = (8324, 5789, 2098),$$
$$\mathbf{b} = (9265, -246, 8034),$$
$$\mathbf{c} = (4321, -765, 7903).$$

(b) Now consider all parallelepipeds whose base is determined by the vectors $\mathbf{a} = (2, 0, -1)$ and $\mathbf{b} = (0, 2, -1)$, and whose height is variable $\mathbf{c} = (x, y, z)$. Assume that x, y, and z are positive and $\|\mathbf{c}\| = 1$. Use the triple product to compute a formula for the volume of the parallelepiped involving x, y, and z. Compute the maximum value of that volume in terms of x and y as follows. It is clear from the following formula:

$$\mathbf{c} \cdot (\mathbf{a} \times \mathbf{b}) = \|\mathbf{c}\| \, \|\mathbf{a} \times \mathbf{b}\| \cos\theta,$$

where θ is the angle between **c** and the line perpendicular to the plane determined by **a** and **b**, that the maximum occurs when **c** is perpendicular to both **a** and **b**. Use the dot product to determine the vector **c** yielding the maximum value. (We will see in Chapter 7, *Optimization in Several Variables*, how to solve multivariable max-min problems.)

Problem 2.13. Find parametric equations for each of the following lines, then graph the line using **`fplot3`**.

(a) The line containing the points $(5.2, -4.11, 9)$ and $(0.3, 6.33, -2.34)$.

(b) The line passing through the point $(4, 0.35, -3.72)$ and parallel to the vector $\mathbf{v} = (4.66, -2.1, -3.51)$.

Problem 2.14. Find the distance from the point $(4.3, 5.4, 6.5)$ to the line whose parametric equations are $x = -1 + t$, $y = -2 + 2t$, $z = -3 + 3t$.

Problem 2.15. Draw the cylinder whose points lie at a distance 1 from the line $x = t$, $y = 10t$, $z = 0$. (Hint: To use **`fsurf`**, choose two unit vectors perpendicular to the line and use them and a vector along the line to parametrize the cylinder.)

Problem 2.16. For each of the following, find the equation of the plane and graph it:

(a) The plane containing the point $P_0 = (3.4, -2.6, 5)$ and having normal vector $\mathbf{n} = (-3.22, 1.2, 0.3)$.

(b) The plane containing the two lines

$$\begin{aligned} x &= 1 + t, \qquad y = 2 + t, \qquad z = 1 + 2t, \\ x &= 2t, \qquad y = 1 + t, \qquad z = -1 - t. \end{aligned}$$

Problem 2.17. Find the distance from the point $P = (100, 201, 349)$ to the plane $-213x - 438y + 301z = 500$.

Problem 2.18. Find parametric equations for the line formed by the intersection of the following two planes:

$$\begin{aligned} 2x - 3y + z &= 10, \\ -5x - 2y + 3z &= 15. \end{aligned}$$

Graph the two planes and the line of intersection on the same plot.

Problem 2.19. Consider the vector-valued functions

$$\begin{aligned} \mathbf{F}(t) &= (e^t, \sqrt{1+t}, \ln(1+t^2)), \\ \mathbf{G}(t) &= (\sin(t), \sec(t+1), (t-1)/(t+1)). \end{aligned}$$

Compute the functions $\mathbf{F} + \mathbf{G}$, $\mathbf{F} \cdot \mathbf{G}$, and $\mathbf{F} \times \mathbf{G}$.

Problem 2.20. Plot the following curves. In each case indicate the direction of motion. You will have to be careful when selecting the time interval on which to display the curve in order to get a meaningful picture. You may find **`axis`** useful in improving your plots.

(a) $\mathbf{F}(t) = (\cos t, \sin t, t/2)$.

(b) $\mathbf{F}(t) = (e^{-t} \sin t, e^{-t} \cos t, 1)$.

(c) $\mathbf{F}(t) = (t, t^2, t^3)$.

Problem 2.21. Graph the cycloid

$$\mathbf{r}(t) = (2(t - \sin t), 2(1 - \cos t))$$

and the trochoid

$$\mathbf{s}(t) = (2t - \sin t, 2 - \cos t)$$

together on the interval $[0, 4\pi]$. Find the coordinates of the four points of intersection. (Hint: Solve the equation $\mathbf{r}(t) = \mathbf{s}(u)$. Note the different independent variables

for **r** and **s**—the points of intersection need not correspond to the same "time" on each curve. Also, since the coordinate functions are transcendental, you may need to use **vpasolve** rather than **solve**.) Use MATLAB to mark the four points on your graph.

Problem 2.22. Here's a problem to practice simultaneous plotting of curves and surfaces, as well as finding intersection points.

(a) Plot the two curves $2x^2 + 20y = -1$ and $y = x^4 - x^2$ on the same graph. Find the coordinates of all points of intersection.

(b) Plot the two surfaces $x^2 + y^2 + z^2 = 16$ and $z = 4x^2 + y^2$ and superimpose the plots. (The first surface is a sphere, the second is an elliptic paraboloid. The top half of the sphere will suffice here.) Use **fcontour** to plot the projection into the x-y plane of the curve of intersection of the two surfaces. You may find the option **LevelList** useful in identifying the precise curve.

Problem 2.23. This problem is about intersecting surfaces and curves. For helpful models, see the discussions of Viviani's curve, of "Graphing Surfaces," and of "Parametric Surfaces" in Chapter 2, *Vectors and Graphics*. You might wish to adjust the **view** in each of your three-dimensional plots.

(a) Draw three-dimensional plots of the paraboloid $z = x^2 + y^2$ and of the cylinder $(x-1)^2 + y^2 = 1$. Since the cylinder is not given by an equation of the form $z = f(x, y)$, you will need to plot it parametrically. You can use the parametrization

$$(1 + \cos t,\ \sin t,\ z)$$

with parameters t and z. Superimpose the two three-dimensional plots to see the curve where the surfaces intersect. Find a parametrization of the curve of intersection, and then draw an informative three-dimensional plot of this curve.

(b) Do the same for the paraboloid $z = x^2 + y^2$ and the upper hemisphere $z = \sqrt{1 - (x-1)^2 - y^2}$. This time the equation of the curve of intersection is a bit complicated in rectangular coordinates. It becomes simpler if you project the curve into the x-y plane and convert to polar coordinates. Apply **solve** to the polar equation of the projection to find $r(\theta)$, the formula for r in terms of θ. You can then parametrize the curve by

$$(r(\theta)\cos\theta,\ r(\theta)\sin\theta,\ r(\theta)^2),$$

with θ varying. You might find the option **MaxDegree** useful.

Glossary of MATLAB Commands

abs The *absolute value* function

acos The *arc cosine* function

axis Selects the ranges of x and y to show in a 2D-plot; or x, y and z in a 3D-plot

clear Clears values and definitions for variables and functions. If you specify one or more variables, then only those variables are cleared.

cross The cross product of two vectors

dot The dot product of two vectors

double Converts the (possibly symbolic) expression for a number to a numerical (double-precision) value

fcontour Plots the contour curves of a symbolic expression $f(x, y)$

figure Start a new graphic

fminbnd Find minimum of single-variable function on a fixed interval

fplot Easy function plotter

fplot3 Easy 3D function plotter

fsurf Easy 3D surface plotter

fzero Finds (numerically) a zero of a function near a given starting value

linspace Generates a linearly spaced vector

norm Norm of a vector or matrix

polarplot Plots a curve in polar coordinates

real Follows **syms** to insure variables are real

solve Symbolic equation solver

subs Substitute for a variable in an expression

syms Set up one or more symbolic variables

view Specifies a point from which to view a 3D graph

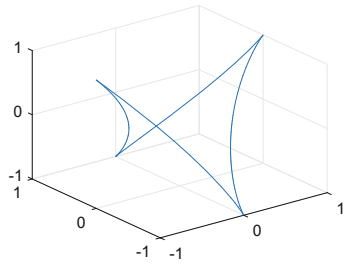
vpasolve Finds numerical solutions to symbolic equations

Options to MATLAB Commands

LevelList Specifies the contour levels for **fcontour**

MaxDegree An option to **solve**, giving the maximum degree of polynomials for which MATLAB will try to find explicit solution formulas

Chapter 3
Geometry of Curves



Curves are the most basic geometric objects. In this chapter, we will study curves in the plane and curves in three-dimensional space. To each curve, we can attach certain natural geometric *invariants*; that is, quantities that remain unchanged when the curve is translated, reflected, or rotated. These invariants include

- arclength;
- the number and nature of singularities (such as cusps or nodes);
- curvature, which measures how much the curve bends; and
- torsion, which measures how much the curve twists.

We will use calculus to define each invariant in terms of derivatives and integrals. Finally, we will show that the geometric invariants characterize the curve. In other words, distinct curves cannot have identical invariants, unless they are congruent.

3.1 Parametric Curves

A curve is the image of a continuous (and usually differentiable) function $\mathbf{r} : I \to \mathbb{R}^n$, where I is an interval. In this book, n is either 2 or 3. We shall be ambiguous about the nature of the interval I, allowing for the possibility of it being open, closed, bounded, or unbounded. Here are some examples of curves.

- The right circular helix of radius 1:

$$\mathbf{r}(t) = (\cos t, \sin t, t), \quad t \in \mathbb{R}.$$

- A three-dimensional astroid:

$$\mathbf{r}(t) = (\cos^3 t, \sin^3 t, \cos 2t), \quad 0 \leq t \leq 2\pi.$$

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_3

- The cycloid is the curve traced out by a point on the circumference of a wheel as it rolls along a straight line at constant speed without slipping. The parametric equations are:

$$\mathbf{r}(t) = (t - \sin t, 1 - \cos t), \quad t \in \mathbb{R}.$$

The cycloid is a plane curve; the helix and astroid are space curves. Throughout the first part of this chapter, we will use the helix as our illustrative example. We will return to the other examples later in the chapter.

We begin by defining the helix in MATLAB and then graphing it (Figure 3.1).

```
>> syms t real; helix = [cos(t), sin(t), t];
>> helplot = fplot3(helix(1), helix(2), helix(3), [0, 4*pi]);
>> view([1,1,1])
```

The function $\mathbf{r}(t), t \in I$, is called a *parametrization* of the curve; the curve itself is the physical set of points (in $\mathbb{R}^2$ or $\mathbb{R}^3$) traced out by the function $\mathbf{r}$ as t varies in the interval I. Every curve has many different possible parametrizations. All of the geometric invariants we associate to a curve will be computed analytically in terms of a parametrization. Clearly, if there is to be any geometric validity for an invariant, the quantity should be independent of the choice of parametrization.

If we think of a particle traversing a curve according to the prescription $\mathbf{r}(t)$, then it is natural to call $\mathbf{r}(t)$ the *position* vector. Continuing with the physical analogy, we call $\mathbf{v}(t) = \mathbf{r}'(t)$ the *velocity* vector and $\mathbf{a}(t) = \mathbf{r}''(t)$ the *acceleration* vector—provided these derivatives exist. The *speed* $v(t)$ of the particle is defined as the magnitude of the velocity vector, $v(t) = \|\mathbf{v}(t)\|$.

As an example, we can compute the velocity, speed, and acceleration of the helix. We start with commands that we will use repeatedly in the future to compute dot products and lengths of vectors. Here we need to explain something about MATLAB. In MATLAB, by default, vectors are assumed to have complex, not real,

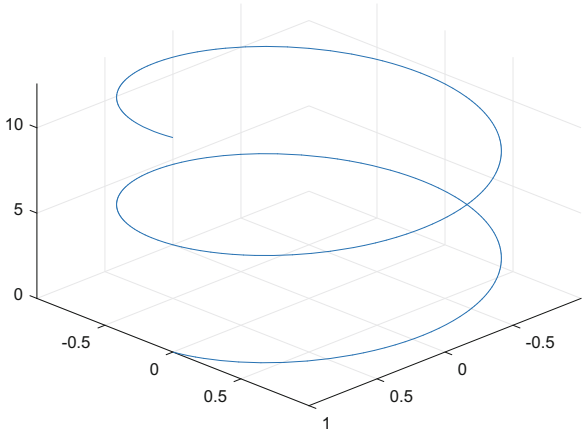


Fig. 3.1 A helix

entries. Accordingly, the MATLAB command **dot** takes the complex conjugate of its first argument before multiplying corresponding entries of the vectors and summing. For numerical vectors with real entries, this has no effect at all. But when the vectors are real-valued symbolic expressions, this has the effect of introducing the command **conj** in ways that sometimes cause trouble. Similarly, **norm** when applied to symbolic vectors introduces the command **abs** in places where we do not want it. That is the reason for our defining **realdot** and **vectorlength** here to replace **dot** and **norm** in the calculations in this chapter.

```
>> realdot = @(x,y) x*transpose(y);
>> vectorlength = @(x) sqrt(simplify(realdot(x,x)));
>> velhelix = diff(helix, t)
[ -sin(t), cos(t), 1]

>> speedhelix = vectorlength(velhelix)
2^(1/2)

>> acchelix = diff(velhelix, t)
[ -cos(t), -sin(t), 0]
```

We say that a curve is *smooth* if its velocity vector is a continuous function and its speed never vanishes. We can see from the formulas just computed that the velocity vector of the helix only involves continuous functions, and that the speed is the nonzero constant $\sqrt{2}$. So, the helix is a smooth curve. More generally, we say that a curve is *piecewise smooth* if in any bounded subinterval of I, the curve is smooth except for finitely many points at which one-sided derivatives of $\mathbf{r}$ are postulated to exist. In the following discussion, we will generally assume our curves are smooth, though piecewise smooth curves with a few singularities will also be considered.

Now suppose we have two parametrizations of the same non-self-intersecting curve:

$$\mathbf{F}(t), t \in [a,b] \quad \text{and} \quad \mathbf{G}(u), u \in [c,d].$$

Then $\mathbf{F}$ and $\mathbf{G}$ are one-to-one functions and for every $t \in [a,b]$, there is a unique $u \in [c,d]$ such that $\mathbf{F}(t) = \mathbf{G}(u)$, and vice versa. Since the two parametrized curves must have the same endpoints, either $\mathbf{F}(a) = \mathbf{G}(c)$ and $\mathbf{F}(b) = \mathbf{G}(d)$, or else $\mathbf{F}(a) = \mathbf{G}(d)$ and $\mathbf{F}(b) = \mathbf{G}(c)$. We assume the former, which means simply that $\mathbf{F}$ and $\mathbf{G}$ trace out the points of the curve *in the same order*, or in other words, they have the *same orientation*. Hence, there is a uniquely defined, monotonically increasing function $u = \varphi(t)$, $\varphi(a) = c$, $\varphi(b) = d$ so that $\mathbf{G}(\varphi(t)) = \mathbf{F}(t), t \in [a,b]$. Moreover, if the parametrizations are smooth, then φ is differentiable, φ' is continuous and never vanishes. The latter follows from the chain rule

$$\mathbf{F}'(t) = \mathbf{G}'(\varphi(t))\varphi'(t),$$

and the fact that neither $\mathbf{F}'$ nor $\mathbf{G}'$ vanishes. Conversely, if $\mathbf{G}(u)$ parametrizes a curve and if $u = \phi(t)$ is a smooth, monotonically increasing function, then setting $\mathbf{F}(t) = \mathbf{G}(\phi(t)) = \mathbf{G}(u)$ defines a *reparametrization* of the original curve.

3.2 Geometric Invariants

We are now ready to start computing the geometric invariants of parametric curves. We will begin with the arclength, which is the integral of the speed. This is a good starting point, because we can use the arclength function to reparametrize a smooth curve with a parametrization that has constant unit speed; doing so will make the rest of the discussion simpler.

3.2.1 Arclength

We define the arclength to be the integral of the speed. More precisely, the *arclength* of the curve $\mathbf{r}(t)$ between $t = a$ and $t = b$ is

$$\int_a^b v(t)\,dt = \int_a^b \|\mathbf{v}(t)\|\,dt.$$

Since the speed is nonnegative, the *arclength function*

$$s(t) = \int_a^t v(u)\,du$$

is a monotonically increasing, smooth function of t. Notice that s depends on t in the manner described by the Fundamental Theorem of Calculus, so we can compute the derivative by the formula

$$\frac{ds}{dt} = v(t) = \|\mathbf{v}(t)\|.$$

Next, let us check that the arclength invariant is independent of parametrization. This follows from a simple change of variable argument:

$$\begin{aligned}\int_c^d \|\mathbf{G}'(u)\|\,du &= \int_a^b \|\mathbf{G}'(\varphi(t))\|\varphi'(t)\,dt \\ &= \int_a^b \|\mathbf{F}'(t)\|\,dt.\end{aligned}$$

We can reparametrize our curve by using arclength itself as the parameter. This may be computationally extremely difficult, but theoretically it will make our ensuing arguments much, much easier. Thus, reparametrizing the curve in terms of arclength has the net effect of making the speed function equal to the constant 1. So, given any (piecewise) smooth curve, it is no loss of generality to assume that it is parametrized by its arclength.

Let us see how this idea works in the case of the helix. Since the speed is equal to $\sqrt{2}$, its integral is the arclength function $s = \sqrt{2}t$. In this case, we can easily solve for t and write $t = s/\sqrt{2}$. Thus, we can reparametrize the helix as follows:

```
>> unithelix = subs(helix, t, s/sqrt(2))

[ cos((2^(1/2)*s)/2), sin((2^(1/2)*s)/2), (2^(1/2)*s)/2]
```

By finding the length of the velocity vector, we can check that the parametrization by arclength gives a curve whose speed always equals 1.

```
>> newspeed = vectorlength(diff(unithelix, s))
1
```

3.2.2 The Frenet Frame

Until further notice, our curves will be parametrized by arclength. So, we will write

$$\mathbf{r} = \mathbf{r}(s), \qquad s \in I,$$

where

$$\left\| \frac{d\mathbf{r}}{ds} \right\| = \|\mathbf{r}'(s)\| \equiv 1, \qquad s \in I.$$

Such curves are called *unit-speed curves*. As we just saw, a smooth curve depending on any parameter t can always be converted into a unit-speed curve by reparametrizing it in terms of its arclength function $s(t)$.

The velocity vector

$$\mathbf{T}(s) = \frac{d\mathbf{r}}{ds} = \frac{d\mathbf{r}}{dt}\frac{dt}{ds} = \frac{1}{\|\mathbf{v}\|}\frac{d\mathbf{r}}{dt} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

is a *unit tangent* vector to the curve at $\mathbf{r}(s)$. In this case, it is perpendicular to the acceleration vector. This fact is a simple consequence of a principle that we shall use repeatedly: *a vector-valued function with constant magnitude is perpendicular to its derivative*. To see this, simply note that differentiating the equation

$$\mathbf{F}(t) \cdot \mathbf{F}(t) \equiv c$$

yields

$$2\,\mathbf{F}(t) \cdot \mathbf{F}'(t) = 0.$$

Now apply the computation to $\mathbf{T}(s)$ itself. The derivative is perpendicular to the unit tangent $\mathbf{T}(s)$, so it must be normal to the curve. Adjusting its length, we get a *unit normal* vector

$$\mathbf{N}(s) = \frac{\mathbf{T}'(s)}{\|\mathbf{T}'(s)\|},$$

which satisfies

$$\mathbf{T}(s) \cdot \mathbf{N}(s) = 0.$$

The vector $\mathbf{N}$ is often called the *principal normal*.

The vectors $\mathbf{T}$ and $\mathbf{N}$ determine a plane, called the *osculating plane*. We shall explain the term below. (It can be shown that if $\mathbf{N} \neq \mathbf{0}$, then it points "into the concavity" when the curve is projected onto the osculating plane.) Of course, we want the osculating plane to exist, so we shall assume tacitly that the derivative $\mathbf{T}'$ vanishes at only finitely many points in any bounded interval. Finally, we set

$$\mathbf{B}(s) = \mathbf{T}(s) \times \mathbf{N}(s)$$

and call it the unit *binormal* vector. It is perpendicular to both $\mathbf{T}$ and $\mathbf{N}$. The three vectors together $\mathbf{T}(s)$, $\mathbf{N}(s)$, $\mathbf{B}(s)$ constitute a *Frenet frame* (named after the mid-nineteenth century French mathematician Frenet).

How does the Frenet frame depend on the parametrization? We saw above that the unit tangent vector $\mathbf{T}$ is the same whether we use the parametrization by t or the parametrization by arclength s. (In fact, the unit tangent only depends on the orientation.) Similarly, we have

$$\frac{d\mathbf{T}}{ds} = \frac{d\mathbf{T}}{dt}\frac{dt}{ds} = \frac{1}{\|\mathbf{v}\|}\frac{d\mathbf{T}}{dt}.$$

In other words, the two vector derivatives differ by a positive scalar function. So, the corresponding unit normal vectors are identical. Clearly, the binormal also turns out to be independent of the parametrization.

Returning to the helix, we will compute its Frenet frame using the parametrization by arclength. We also define another command that we will use repeatedly to compute unit vectors.

```
>> unitvector = @(x) simplify(x/vectorlength(x));
>> UT = diff(unithelix, s)
[ -(2^(1/2)*sin((2^(1/2)*s)/2))/2, (2^(1/2)*cos((2^(1/2)*s)/2))/2,
2^(1/2)/2]

>> UN = unitvector(diff(UT, s))
[ -cos((2^(1/2)*s)/2), -sin((2^(1/2)*s)/2), 0]

>> UB = simplify(cross(UT, UN))
[ (2^(1/2)*sin((2^(1/2)*s)/2))/2, -(2^(1/2)*cos((2^(1/2)*s)/2))/2,
2^(1/2)/2]
```

Once we have computed **UT**, **UN**, and **UB** as functions of **s**, it is possible to plot them at any point on the curve by using the function script `curveframeplot.m` on our accompanying website to draw the three Frenet vectors emanating from that point. Figure 3.2 shows a few Frenet frames for the helix at various points.

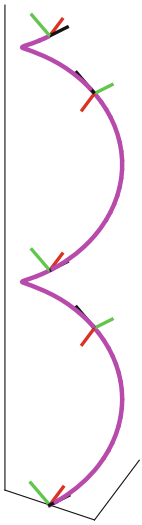


Fig. 3.2 Frenet frames for the helix

Note how the frame rotates as one moves along the curve. It is sometimes useful to think of the Frenet frame as a moving coordinate system specially adapted to the curve. You can see this better by producing a sequence of pictures of Frenet frames using the program on the website, or by producing an animation as we explain there.

3.2.3 Curvature and Torsion

The *curvature* measures, at any point, how much the curve is bending. Said another way, it measures the failure of the curve to be linear. It is defined by the formula

$$\kappa(s) = \|\mathbf{T}'(s)\|, \qquad s \in I,$$

so that curvature is the magnitude of the rate of change of the unit tangent vector. Thus $\mathbf{T}'(s) = \kappa(s)\mathbf{N}(s)$ by definition. The *torsion* measures how much the curve is twisting, or the failure of the curve to be planar. The torsion is uniquely specified by the equation

$$\mathbf{B}'(s) = -\tau(s)\mathbf{N}(s).$$

Indeed, we observe that $\mathbf{B}'(s) \cdot \mathbf{B}(s) = 0$, since $\mathbf{B}(s)$ has constant length. Moreover, $\mathbf{B}'(s) \cdot \mathbf{T}(s) = 0$ (which follows by differentiating the equation $\mathbf{B}(s) \cdot \mathbf{T}(s) = 0$). Hence the vectors $\mathbf{B}'$ and $\mathbf{N}$ must be parallel, so they differ by a scalar. Since the

binormal is the unit normal vector to the plane defined by the curve (to second order), the torsion measures the rate at which the curve twists out of its plane.

We illustrate these ideas by computing the curvature and torsion of the helix.

```
>> helcurv = vectorlength(diff(UT, s))
helcurv =
1/2
```

Thus, the helix has constant curvature, $\kappa = 1/2$.

```
>> diff(UB, s)
ans =
[ cos((2^(1/2)*s)/2)/2, sin((2^(1/2)*s)/2)/2, 0]
```

We observe that the derivative of the binormal is negative one-half of the unit normal vector, so the torsion of the helix is also constant, $\tau = 1/2$.

The three vectors in the Frenet frame form a right-handed frame field of mutually perpendicular unit vectors, traveling along the curve. The definitions of curvature and torsion depend on the fact that the derivatives of $\mathbf{T}$ and $\mathbf{B}$ are parallel to $\mathbf{N}$. Can we say anything about the derivative of $\mathbf{N}$?

Theorem 3.1. (Frenet Formulas). *If* $\mathbf{r}$ *is a smooth unit-speed curve with positive curvature* κ *and torsion* τ*, then we have the following family of first-order differential equations:*

$$\begin{aligned} \mathbf{T}' &= \kappa\mathbf{N}, \\ \mathbf{N}' &= -\kappa\mathbf{T} + \tau\mathbf{B}, \\ \mathbf{B}' &= -\tau\mathbf{N}. \end{aligned}$$

Proof. Only the second differential equation remains to be verified. In fact $\mathbf{N}'$ can be expanded in terms of the frame field $\{\mathbf{T},\mathbf{N},\mathbf{B}\}$:

$$\mathbf{N}' = (\mathbf{N}'\cdot\mathbf{T})\mathbf{T} + (\mathbf{N}'\cdot\mathbf{N})\mathbf{N} + (\mathbf{N}'\cdot\mathbf{B})\mathbf{B}.$$

Differentiating $\mathbf{N}\cdot\mathbf{T} = 0$, we see that

$$\mathbf{N}'\cdot\mathbf{T} = -\mathbf{N}\cdot\mathbf{T}' = -\mathbf{N}\cdot\kappa\mathbf{N} = -\kappa.$$

Also $\mathbf{N}\cdot\mathbf{N}' = 0$ since $\mathbf{N}$ has constant length. Finally, we have

$$\mathbf{N}'\cdot\mathbf{B} = -\mathbf{N}\cdot\mathbf{B}' = -\mathbf{N}\cdot(-\tau\mathbf{N}) = \tau. \quad \square$$

The Frenet formulas in Theorem 3.1 have been simplified slightly, because we are using the parametrization by arclength. In practice, it may be extremely difficult to find a formula for the arclength function, so it may be computationally impossible to reparametrize a given curve into a unit speed curve. We need to have formulas for the geometric invariants expressed in terms of variable-speed parametrizations. We present these formulas in the next paragraph without justification.

If $\mathbf{F}$ is any vector-valued function of the parameter on a curve, then we can think of $\mathbf{F}$ as a function of s or of t. By the chain rule, we have

$$\frac{d\mathbf{F}}{dt} = \frac{d\mathbf{F}}{ds}\frac{ds}{dt} = v(t)\frac{d\mathbf{F}}{ds}.$$

Applying this formula to the vectors in the Frenet frame, we can rewrite the Frenet formulas for any parametric curve in the form

$$\mathbf{T}' = \kappa v\mathbf{N}, \tag{3.1}$$
$$\mathbf{N}' = -\kappa v\mathbf{T} + \tau v\mathbf{B}, \tag{3.2}$$
$$\mathbf{B}' = -\tau v\mathbf{N}. \tag{3.3}$$

The velocity and acceleration vectors are

$$\mathbf{v} = v\mathbf{T}, \qquad \mathbf{a} = \frac{dv}{dt}\mathbf{T} + \kappa v^2\mathbf{N}. \tag{3.4}$$

It is useful to keep in mind that $\mathbf{v}$ and $\mathbf{a}$ both lie in the osculating plane, and that $\mathbf{N}$ and $\mathbf{a}$ are on the same side of $\mathbf{T}$, as shown in Figure 3.3.

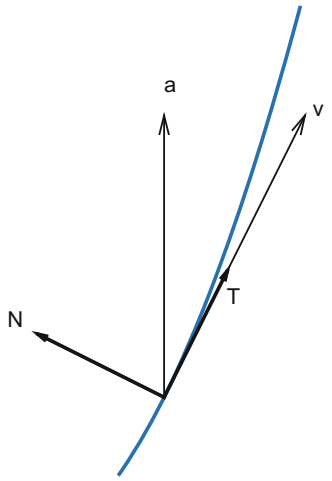


Fig. 3.3 The osculating plane for a parabola, showing $\mathbf{v}$, $\mathbf{a}$, $\mathbf{T}$, $\mathbf{N}$

Here $\mathbf{B}$ would point straight out of the page. If we compute $\mathbf{v}\times\mathbf{a}$, the term involving $\frac{dv}{dt}$ cancels, so the curvature and torsion are given by

$$\kappa = \frac{\|\mathbf{v}\times\mathbf{a}\|}{\|\mathbf{v}\|^3}, \qquad \tau = \frac{(\mathbf{v}\times\mathbf{a})\cdot\mathbf{a}'}{\|\mathbf{v}\times\mathbf{a}\|^2}.$$

It may be easier computationally to compute the Frenet frame by using only the velocity and acceleration vectors, namely

$$\mathbf{T} = \frac{\mathbf{v}}{\|\mathbf{v}\|}, \qquad \mathbf{B} = \frac{\mathbf{v} \times \mathbf{a}}{\|\mathbf{v} \times \mathbf{a}\|}, \qquad \mathbf{N} = \mathbf{B} \times \mathbf{T}.$$

3.3 Differential Geometry of Curves

The Frenet formulas have many uses in the subject called *differential geometry*. Differential geometry involves a sophisticated study of the uses of calculus to penetrate the mysteries of geometric objects. We shall content ourselves here with using the Frenet formulas to solve some elementary problems

1. We know that the best linear approximation to a curve at a point is the tangent vector. What is the best approximation by a circle?
2. Give a condition to determine when a curve is planar.
3. Give a condition to determine when a curve is spherical.
4. Give a condition to determine when a curve is helical. When is it a circular helix?
5. Show that all the geometric invariants we have defined are independent of parametrization.

We shall give solutions to problems 1–4 in terms of the geometric invariants we have developed. To simplify the discussion, we will continue to assume that the curve is parametrized by arclength. Before beginning, we call your attention to the computation of the geometric invariants for the helix. In particular, we note that the speed, curvature, and torsion are all nonzero, but constant. Now we begin to solve the problems.

3.3.1 The Osculating Circle

The circle that best approximates the curve near a point must be one that is tangent to the curve at the point in question. But there are many such circles. We pick out the circle that has the same curvature as the curve and has center along the ray pointing in the direction of the principal normal vector. Since a circle of radius ρ has constant curvature $\kappa = 1/\rho$, that can be done by selecting the circle of radius $1/\kappa(s)$, centered at the point $\mathbf{r}(s) + (1/\kappa(s))\mathbf{N}(s)$. The radius of this circle, $\rho = 1/\kappa$, is called the *radius of curvature* of the curve at the appropriate point. The circle "kisses" the curve accurately to second order, thus is given the name *osculating circle* (from the Latin word for "kissing"). The plane it lies in (i.e., the one determined by $\mathbf{T}(s)$ and $\mathbf{N}(s)$) is therefore referred to as the osculating plane. The curve determined by the centers of all osculating circles is called the *evolute* of the curve. Figure 3.3 shows one of the osculating circles to the helix, along with the evolute of the helix, which

is another helix. You can see how the figure was made by looking at the website accompanying this book. Using the program given there, you can also do an animation to show how the osculating circle moves as you go from point to point on the curve (Figure 3.4).

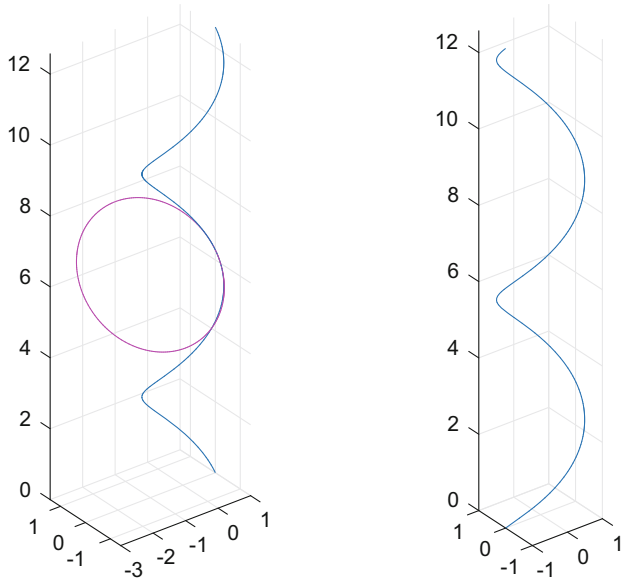


Fig. 3.4 An osculating circle for the helix (left), together with the evolute of the helix (right)

3.3.2 Plane Curves

If we think of plane curves as those which bend but do not twist, then the following theorem is perfectly natural.

Theorem 3.2. *A unit-speed curve is planar if and only if its torsion is zero.*

Proof. If the curve is planar, then the binormal vector is constant and thus has zero derivative. Therefore the torsion, which is, up to scalar, the length of the derivative of the binormal, must vanish. Conversely, suppose $\tau(s) = 0$ for all s. Then $\mathbf{B}'(s) = 0$. Hence $\mathbf{B}(s)$ is a constant vector, say $\mathbf{B}$. Then in fact the curve lies in the plane through $\mathbf{r}(0)$ perpendicular to $\mathbf{B}$. To see that, set $g(s) = (\mathbf{r}(s) - \mathbf{r}(0)) \cdot \mathbf{B}$. Then $g'(s) = \mathbf{T}(s) \cdot \mathbf{B} = 0$. But $g(0) = 0$. Therefore g is identically zero, which proves the assertion. □

3.3.3 Spherical Curves

We say that a curve $\mathbf{r}$ is spherical if it is lies entirely on the surface of a sphere. Then we have the following characterization.

Theorem 3.3. *If a unit-speed curve is spherical, and if $\rho = 1/\kappa$, $\sigma = 1/\tau$, then $\rho^2 + (\rho'\sigma)^2$ is constant, say r^2, and r is the radius of the sphere. Conversely, if κ is not constant, and if $\rho^2 + (\rho'\sigma)^2$ is a constant r^2, then the curve lies on a sphere of radius r. A curve with constant curvature may or may not be spherical.*

Proof. Suppose that $\mathbf{r}$ lies on a sphere of radius r centered at $\mathbf{c}$. Then the vector function $\mathbf{r} - \mathbf{c}$ points from the center of the sphere to the curve and therefore must be perpendicular to the unit tangent vector. Hence we can write $\mathbf{r} = \mathbf{c} + u(s)\mathbf{N}(s) + w(s)\mathbf{B}(s)$, for some as yet undetermined functions u and w. Then we differentiate and use the Frenet formulas

$$\begin{aligned}\mathbf{T}(s) &= \mathbf{r}'(s)\\ &= u'(s)\mathbf{N}(s) + u(s)(-\kappa(s)\mathbf{T}(s) + \tau(s)\mathbf{B}(s))\\ &\quad + w'(s)\mathbf{B}(s) + w(s)(-\tau(s)\mathbf{N}(s)).\end{aligned}$$

Equating coefficients in the Frenet frame, we find that

$$u(s) = -\rho(s) \quad \text{and} \quad w(s) = -\rho'(s)\sigma(s),$$

where by definition $\rho = 1/\kappa$ and $\sigma = 1/\tau$. Hence $\rho^2 + (\rho'\sigma)^2 = r^2$ is constant.

Now for the converse. If the quantity $\rho^2 + (\rho'\sigma)^2$ is constant, say

$$\rho^2 + (\rho'\sigma)^2 = r^2, \tag{3.5}$$

and if $\rho' \neq 0$, then $\mathbf{r}$ lies on a sphere of radius r. To see that, consider the curve $\mathbf{c}(s) = \mathbf{r}(s) + \rho(s)\mathbf{N}(s) + \rho'(s)\sigma(s)\mathbf{B}(s)$. It suffices to show that the curve is constant. To do this, simply compute $\mathbf{c}'$ and employ the Frenet formulas and equation (3.5); we leave the computation to the reader. (At one point in the computation, we must divide by ρ', which explains why we assume $\rho' \neq 0$.)

Finally, a right circular helix is not spherical but has constant curvature; on the other hand, the equator of a sphere is spherical and also has constant curvature. □

3.3.4 Helical Curves

Consider the right circular helix from our first example. It is clear that at every point on the helix, the unit tangent vector makes a constant angle with respect to the z-axis. With that in mind, we make the definition: a curve $\mathbf{r}$ is called a *cylindrical helix* if there is a fixed unit vector $\mathbf{u}$ such that $\mathbf{T}(s) \cdot \mathbf{u}$ is constant. Then we have

Theorem 3.4. *A curve with nonvanishing curvature is a cylindrical helix if and only if the ratio τ/κ is constant.*

Proof. If $\mathbf{r}$ is a cylindrical helix, then $\mathbf{T}\cdot\mathbf{u} = \cos\theta$, where $\mathbf{u}$ is a fixed unit vector and θ is a fixed angle. But then

$$0 = (\mathbf{T}\cdot\mathbf{u})' = \mathbf{T}'\cdot\mathbf{u} = \kappa\mathbf{N}\cdot\mathbf{u}.$$

Since κ is positive (by hypothesis), we find that $\mathbf{N}\cdot\mathbf{u} = 0$. Hence $\mathbf{u}$ lies in the plane determined by the tangent and binormal vectors. Thus

$$\mathbf{u} = \cos\theta\,\mathbf{T}(s) + \sin\theta\,\mathbf{B}(s).$$

We differentiate and make use of the Frenet formulas to obtain

$$(\kappa\cos\theta - \tau\sin\theta)\mathbf{N} = 0.$$

Therefore $\tau/\kappa = \cot\theta$, a constant.

Conversely, suppose τ/κ is constant, say $\cot\theta$ for some choice of θ. Set $\mathbf{U}(s) = \cos\theta\,\mathbf{T}(s) + \sin\theta\,\mathbf{B}(s)$. Differentiating, we find $\mathbf{U}'(s) = (\kappa(s)\cos\theta - \tau(s)\sin\theta)\mathbf{N}(s)$. By assumption, the latter is zero. Hence $\mathbf{U}$ is constant. But then $\mathbf{U}(s) = \mathbf{U}(0) = \cos\theta\mathbf{T}(0) + \sin\theta\mathbf{B}(0)$. Hence $\mathbf{T}(s)\cdot\mathbf{U} = \cos\theta$, for any s. □

Now in the case of a right circular helix, not only is the quotient τ/κ constant, each of the quantities κ and τ is constant. To see that that property characterizes right circular helices (and solves problem 4 completely), we must proceed to the last problem.

3.3.5 Congruence

Here we get a little too heavily into differential geometry to give a complete proof; we'll content ourselves with an informal discussion. A vector-valued function of three variables $\mathbf{F}(x,y,z) = (f(x,y,z), g(x,y,z), h(x,y,z))$ can be thought of as a transformation of space $\mathbf{F}:\mathbb{R}^3 \to \mathbb{R}^3$. Such a transformation is called an *isometry* if it preserves distance, that is, $d(\mathbf{F}(\mathbf{p}),\mathbf{F}(\mathbf{q})) = d(\mathbf{p},\mathbf{q})$, for every pair of points $\mathbf{p}$ and $\mathbf{q}$ in three-space, where d is the Euclidean distance function. Examples of isometries include:

- parallel translations;
- rotations about an axis; and
- reflections in a plane.

It is a fact that any isometry is just a combination of these three types of transformations. The first two are examples of *orientation-preserving* transformations; the

latter is *orientation-reversing*. We will not give precise definitions of these terms; suffice it to say that if we fix a right-handed frame (i.e., a collection $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ consisting of three mutually perpendicular unit vectors satisfying $\mathbf{e}_1 \cdot (\mathbf{e}_2 \times \mathbf{e}_3) = 1$), then an orientation-preserving isometry will convert it into another right-handed frame, while an orientation-reversing one will make it left-handed (i.e., $\mathbf{e}_1 \cdot (\mathbf{e}_2 \times \mathbf{e}_3) = -1$). We say that two curves $\mathbf{r}_1, \mathbf{r}_2$ are *congruent* if there is an isometry of three-space that transforms one curve into the other.

Here is the relationship between the geometric invariants of two congruent curves. Let $\mathbf{r}_1$ be a smooth curve, $\mathbf{F}$ an isometry, and $\mathbf{r}_2 = \mathbf{F} \circ \mathbf{r}_1$. If $\mathbf{F}$ is a translation, then the Frenet frame, speed, curvature, and torsion are exactly the same. (Translations correspond to adding a constant vector to the parametrization; the constant disappears the instant that you take a derivative to compute the velocity.) If $\mathbf{F}$ consists of rotations or reflections, then

$$\mathbf{T}_2 = \mathbf{F}(\mathbf{T}_1), \qquad \kappa_2 = \kappa_1, \tag{3.6}$$

$$\mathbf{N}_2 = \mathbf{F}(\mathbf{N}_1), \qquad \tau_2 = \operatorname{sgn}(\mathbf{F})\tau_1, \tag{3.7}$$

$$\mathbf{B}_2 = \operatorname{sgn}(\mathbf{F})\mathbf{F}(\mathbf{B}_1), \tag{3.8}$$

where $\operatorname{sgn}(\mathbf{F})$ is ± 1 according as $\mathbf{F}$ is orientation-preserving or reversing. Now comes the basic result:

Theorem 3.5. *If $\mathbf{r}_1$ and $\mathbf{r}_2$ have the property that $\kappa_1 = \kappa_2$ and $\tau_1 = \pm\tau_2$, then the two curves are congruent.*

Proof. First assume $\tau_1 = \tau_2$. Consider the two frames $\{\mathbf{T}_1(0), \mathbf{N}_1(0), \mathbf{B}_1(0)\}$ and $\{\mathbf{T}_2(0), \mathbf{N}_2(0), \mathbf{B}_2(0)\}$. There must be an orientation-preserving isometry $\mathbf{F}$ that carries the first frame into the second. Set $\mathbf{r}_* = \mathbf{F} \circ \mathbf{r}_1$. Then by the preceding formulas, the two curves $\mathbf{r}_2$ and $\mathbf{r}_*$ have the same geometric data at the origin. Now consider the scalar-valued function

$$h = \mathbf{T}_* \cdot \mathbf{T}_2 + \mathbf{N}_* \cdot \mathbf{N}_2 + \mathbf{B}_* \cdot \mathbf{B}_2.$$

Since $\mathbf{T}_*$ and $\mathbf{T}_2$ are both unit vectors, we have $\mathbf{T}_* \cdot \mathbf{T}_2 \leq 1$. Furthermore, the dot product equals 1 precisely when $\mathbf{T}_* = \mathbf{T}_2$. Similarly with the other two frame components. Thus it suffices to show that $h(s) \equiv 3$. By the definition of $\mathbf{r}_*$, we have $h(0) = 3$. Now it suffices to show $h' = 0$. But as usual, this follows by differentiating, and then using the Frenet formulas together with the equality of the respective curvatures and torsions.

We have proven, in particular, that $\mathbf{T}_* = \mathbf{T}_2$. Thus, the components of the position vectors have the same derivative. By elementary calculus, they can differ by at most a constant. This says that $\mathbf{r}_* = \mathbf{r}_2 + \mathbf{c}$. But the curves agree at $s = 0$, hence $\mathbf{c} = \mathbf{0}$. Therefore, the fixed isometry $\mathbf{F}$ transforms the curve $\mathbf{r}_1$ into the curve $\mathbf{r}_2$, that is, the two curves are congruent.

The case $\tau_1 = -\tau_2$ is handled similarly, except that an orientation-reversing transformation is employed. □

The dispensation of problem 4 comes swiftly now. Right circular helices have constant curvature and torsion. Since these quantities determine the nature of a curve (up to congruence), the constancy of them characterize helices.

It is also a simple matter to augment Theorem 3.5 to the variable-speed case.

Corollary 3.6. *Two curves have the same speed, curvature, and torsion (the latter up to sign) if and only of they are congruent.*

Proof. Assume the invariants of two curves agree. Then the unit-speed reparametrizations of the two curves have matching curvatures and torsions, therefore are congruent by Theorem 3.5. The original curves are then easily seen to be congruent. We leave the details to the reader. The other direction is even easier. □

3.3.6 Two More Examples

Our last task is to work out the geometric invariants for the remaining examples. For this purpose, we will use the following functions, with arguments **r** and **t**. Here **t** is the name of the independent variable and **r** is the position vector, written as a symbolic expression in **t**.

```
>> velocity = @(r, t) diff(r, t);
>> acceleration = @(r, t) simplify(diff(velocity(r, t), t));
>> thirdder = @(r, t) simplify(diff(acceleration(r, t), t));
>> speed = @(r, t) vectorlength(velocity(r, t));
>> vta = @(r, t) simplify(cross(velocity(r, t), ...
   acceleration(r, t)));
>> UT = @(r, t) unitvector(velocity(r, t));
>> UN = @(r, t) unitvector(diff(UT(r, t), t));
>> UB = @(r, t) simplify(cross(UT(r, t), UN(r, t)));
```

The curvature and torsion functions are slightly more complicated to define. The fastest way to compute them is to use the cross product of **v** with **a**, denoted **vta**, for "velocity times acceleration," since the normal component of **a** is $v^2\kappa\mathbf{N}$ by (3.4).

```
>> curvature = @(r, t) simplify(vectorlength(vta(r, t)) ...
   /speed(r, t)^3);
>> torsion = @(r, t) simplify(realdot( ...
   vta(r, t), thirdder(r, t))/realdot(vta(r, t), vta(r, t)));
```

Now we are ready to look at the examples.

3.3.6.1 The Astroid

The astroid is defined parametrically and plotted as follows:

```
>> astroid = [cos(t)^3, sin(t)^3, cos(2*t)];
>> fplot3(astroid(1), astroid(2), astroid(3), [0, 2*pi])
```

The graph is shown in Figure 3.5. We immediately notice the most significant geometric features of the curve: there are four cusps where the curve fails to be smooth. When we compute the geometric invariants, we should look for evidence of the cusps.

We compute the velocity, speed, and acceleration of the astroid.

```
>> velocity(astroid, t)
ans =
[ -3*cos(t)^2*sin(t), 3*cos(t)*sin(t)^2, -2*sin(2*t)]

>> speed(astroid, t)
ans =
(25/8 - (25*cos(4*t))/8)^(1/2)

>> acceleration(astroid, t)
ans =
[ 6*cos(t) - 9*cos(t)^3, 6*sin(t) - 9*sin(t)^3, -4*cos(2*t)]
```

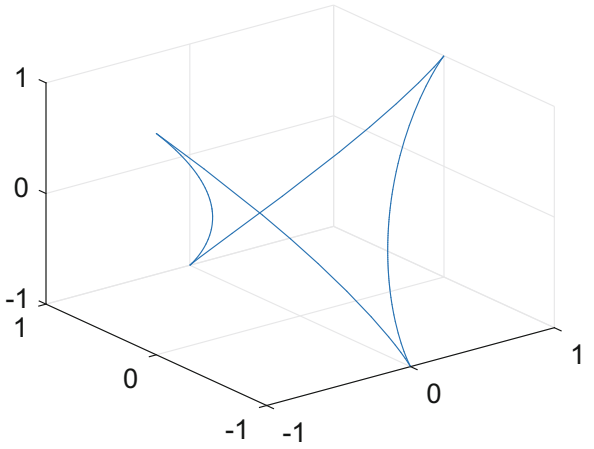


Fig. 3.5 The astroid

From this we see that the speed of the astroid is zero precisely when $1-\cos(4t)=0$, which occurs when $t=0$, $\pi/2$, π, or $3\pi/2$. Thus, we have identified the locations of the four cusps that we saw in the graph.

Our next task is to compute the Frenet frame for the astroid. The commands are as follows:

```
>> UT(astroid, t)

>> UN(astroid, t)

>> UB(astroid, t)
```

The results are quite complicated but the function **dirac** that appears here is only nonzero when its argument is 0, which will happen only at the cusps where we have discontinuities.

Finally, we compute the curvature and the torsion of the astroid.

```
>> curvature(astroid, t)
ans =
-(48*2^(1/2)*sign(sin(t))^2*sin(t)^2*(sin(t)^2 - 1))/(25*(1 - cos(4*t))^(3/2))

>> torsion(astroid, t)
ans =
(4*sin(t))/(25*(cos(t) - cos(t)^3))
```

We can simplify the torsion further since $\cos t - \cos^3 t = \sin^2 t \cos t$. Thus the torsion simplifies to

```
>> simplify(subs(ans, cos(t) - cos(t)^3, cos(t)*sin(t)^2))
ans =
4/(25*cos(t)*sin(t))
```

The curvature and torsion are undefined at the cusps, where $t = 0$, $\pi/2$, π, or $3\pi/2$.

3.3.6.2 The Cycloid

The cycloid is a plane curve, defined parametrically and plotted as follows:

```
>> cyc = [t - sin(t), 1 - cos(t)];
>> fplot(cyc(1), cyc(2), [0, 4*pi]), axis equal tight
```

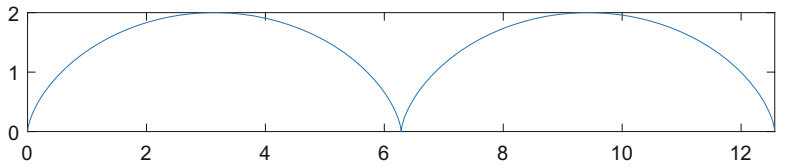


Fig. 3.6 The cycloid

In the graph of the cycloid, we have used **axis equal tight** to cause MATLAB to display the graph using the same scale on both axes with no more of the axes showing than necessary. Since the cycloid is defined as the trace of a point on the edge of a rolling wheel, it is only reasonable to insist that the graph should contain a realistic picture. (Try the graph using the default setting for **axis** to see what happens.)

Before proceeding, we need to discuss a difference between plane curves and space curves. The formulas (and the MATLAB programs) for velocity, speed, acceleration, unit tangent vector, principal normal vector, and curvature work equally well for both kinds of curves. The formulas (and programs) for the binormal vector and for torsion only work for space curves, since they involve a cross product.

Since we know the values of these items for plane curves, we could simply avoid the computation. Another approach is to add a zero component at the end of the plane parametrization.

```
>> cycloid = [cyc, 0];
>> velocity(cycloid, t)
[ 1 - cos(t), sin(t), 0]

>> acceleration(cycloid, t)
[ sin(t), cos(t), 0]

>> speed(cycloid, t)
^(1/2)*(1 - cos(t))^(1/2)
```

The speed of the cycloid is zero when $\cos(t) = 1$, which happens when $t = 0$, 2π, 4π, etc. These are the cusps where the curve is not smooth. We can now proceed to compute the Frenet frame for the cycloid.

```
>> UT(cycloid, t)
ans =
[ (2^(1/2)*(1 - cos(t))^(1/2))/2, (2^(1/2)*sin(t))/(2*(1 - cos(t))^(1/2)), 0]

>> UN(cycloid, t)
ans =
[ (2^(1/2)*sin(t))/(2*(1 - cos(t))^(1/2)), -(2^(1/2)*(1 - cos(t))^(1/2))/2, 0]

>> UB(cycloid, t)
ans =
[ 0, 0, -1]
```

Since the cycloid is a plane curve, we knew that the binormal vector would point in the z-direction. Could we have predicted the negative sign? What does it mean? (Think about the "right-hand rule" for the cross product and about the direction in which the cycloid is curving in Figure 3.6.)

```
>> curvature(cycloid, t)
ans =
2^(1/2)/(4*(1 - cos(t))^(1/2))
```

The curvature of the cycloid blows up at the cusps. Again this is not surprising, since the curve has to turn very sharply at the cusps.

```
>> torsion(cycloid, t)
ans =
0
```

Unsurprisingly, the torsion of a plane curve is zero.

Problem Set C. Curves

When working on this problem set, you may use the programs from this chapter, *Geometry of Curves*. It is not necessary to retype them; they may be found on the website accompanying this book.

Problem 3.1. Each part of this problem contains parametric equations for a curve. You should

- Graph each curve.
- Compute the Frenet frame, speed, curvature, and torsion.
- Describe any points where the curvature or torsion is undefined, and relate those points to geometric properties of the curve as revealed by the graph.

Keep the following points in mind as you work the problems:

- Variable-speed curves may require you to use different formulas than unit-speed curves.
- Plane curves may require different treatment than space curves.
- If you expect to do more than one part of this problem, then you should consider using programs like those presented in this chapter.

(a) The cardioid:

$$\mathbf{r}(t) = 2\cos t(1+\cos t)\,\mathbf{i} + 2\sin t(1+\cos t)\,\mathbf{j}, \quad -\pi \le t \le \pi.$$

(b) The twisted cubic curve:

$$\mathbf{r}(t) = (t, t^2, t^3), \quad -2 \le t \le 2.$$

(c) The tractrix:

$$\mathbf{r}(t) = \begin{cases} e^{-t}\,\mathbf{i} + \int_0^t \sqrt{1-e^{-2u}}\,du\,\mathbf{j}, & t \ge 0, \\ e^{t}\,\mathbf{i} + \int_0^t \sqrt{1-e^{2u}}\,du\,\mathbf{j}, & t \le 0. \end{cases}$$

You may restrict your computations of the geometric data to one of the branches of the curve, but be sure to include both branches when you plot it.

(d) A hyperbolic helix:

$$\mathbf{r}(t) = \left(\sqrt{1+\frac{t^2}{2}}, \frac{t}{\sqrt{2}}, \operatorname{arcsinh}\frac{t}{\sqrt{2}} \right), \quad t \in \mathbb{R}.$$

In this case, make the change of variable $t = \sqrt{2}\sinh u$ in the coordinates of $\mathbf{r}(t)$ and see if you can justify the name of the curve.

(e) A portion of a pseudocircular helix:

$$\mathbf{r}(t) = \left(\frac{1}{3}(1+t)^{3/2}, \frac{1}{3}(1-t)^{3/2}, \frac{t}{\sqrt{2}}\right), \qquad -1 \le t \le 1.$$

In this case, show that when the curve is projected onto the x-y plane, its image is the portion of the curve $x^{2/3}+y^{2/3} = 2/3^{2/3}$ that lies in the first quadrant. We say that a curve $x^a + y^a = c$, for a and c positive, is a *pseudocircle*.

(f) Viviani's curve:

$$\mathbf{r}(t) = (1+\cos t, \sin t, 2\sin(t/2)), \qquad -\pi \le t \le \pi.$$

This curve is the intersection of the sphere $x^2+y^2+z^2 = 4$ with the cylinder $(x-1)^2+y^2 = 1$. You may use the program from Chapter 2 to graph it.

(g) The catenary:

$$\mathbf{r}(t) = (t, \cosh t).$$

(h) The limaçon:

$$\mathbf{r}(t) = (2\cos t + 1)(\cos t, \sin t).$$

The word *limaçon* is old French for "snail." Can you see why this curve is given this name?

Problem 3.2. Find the arclength of the following curves over the indicated interval. Be prepared to use **`double(ans)`** or **`integral`** if **`int`** fails.

(a) $(\cos^3 t, \sin^3 t)$, $t \in [0, 2\pi]$.

(b) $((1/3)(1+t)^{3/2}, (1/3)(1-t)^{3/2}, t/\sqrt{2})$, $t \in [-1, 1]$.

(c) $(\cosh t, \sinh t, t)$ $t \in [0, 1]$.

(d) $(\sin t, \cos 2t, t^3)$, $t \in [0, \pi]$.

(e) (t, t^2, t^3), $t \in [0, 1]$.

(f) $(\arctan t, \cosh t, \ln(1+t))$, $t \in [0, 1]$.

Problem 3.3. Each part of this problem contains a description of a curve as the intersection of two surfaces. Find parametric equations for each curve. Then graph the curve, and find its Frenet frame, speed, curvature, and torsion. Using the criteria presented in this chapter, check each curve to see if it is planar, spherical, or helical.

(a) Intersect the saddle surface $z = xy$ with the cylinder $x^2+y^2 = 1$.

(b) Intersect the paraboloid $z = x^2+y^2$ with the plane $2x+2y+z = 2$.

(c) Intersect the hyperboloid $x^2+y^2-z^2=1$ with the sphere $x^2+y^2+z^2=5$.

(d) Intersect the hyperboloid $x^2+y^2-z^2=1$ with the plane $x+y=1$.

Problem 3.4. Parametrize the sine curve $y=\sin x$ using the parametrization

$$\mathbf{r}(t)=(t,\sin t), \qquad t\in\mathbb{R}.$$

(a) Using the definition in this chapter, show that this curve is smooth.

(b) Compute the curvature, and find all points where the curvature is zero. What geometric property do all those points share?

(c) Without doing any computations, explain why the torsion of this curve must be identically zero.

(d) Rotating the plane 30° is an isometry, which transforms the original sine curve into the curve parametrized by

$$\mathbf{r}(t)=\left(\frac{\sqrt{3}t-\sin t}{2},\frac{t+\sqrt{3}\sin t}{2}\right), \qquad t\in\mathbb{R}.$$

Compute the speed, curvature, and torsion of this curve, and compare them to those of the original curve.

(e) By graphing the curve, decide if it is the graph of some function $y=f(x)$.

(f) In single-variable calculus, you studied the qualitative properties of the graphs of functions $y=f(x)$. In particular, you characterized the maxima, minima, and inflection points in terms of the vanishing of certain derivatives of the function $f(x)$. Using the earlier parts of this problem to supply examples, write a paragraph explaining why the notion of an inflection point is an intrinsic geometric property of the curve, but the notion of a maximum or minimum point is not.

Problem 3.5. Sometimes the formulas for curvature and torsion are so complicated that it is difficult to draw any meaningful conclusion from them. Nevertheless, you can use MATLAB to graph the functions κ and τ, even if their formulas are "messy." For each of the following curves, illustrate this fact by computing κ and τ, and then graphing them together on a single two-dimensional plot. (Use `legend` to identify which is which.) In each case, you should also graph the curve and then reconcile your plots by relating *in detail* the features of the two-dimensional plots of the invariants to the shape of the original curves. (Note: In parts (b) and (c), you must enter the parametrization using fractions and not decimals.)

(a) The figure eight curve, also called the lemniscate of Gerono:

$$\mathbf{r}(t)=(\sin t,\cos t\sin t), \qquad 0\le t\le 2\pi.$$

(b) A logarithmic spiral:

$$\mathbf{r}(t) = (e^{t/10}\cos t, e^{t/10}\sin t), \qquad -10\pi \le t \le 10\pi.$$

(c) The helical logarithmic spiral:

$$\mathbf{r}(t) = (e^{t/10}\cos t, e^{t/10}\sin t, t/10), \qquad -5\pi \le t \le 5\pi.$$

(d) The portion of the intersection of the two cylinders $x^2 + y^2 = 1$ and $y^2 + z^2 = 4$ that lies above the x-y plane.

Problem 3.6. Compute the speed, curvature, and torsion of the curve

$$\mathbf{r}(t) = (t + \sqrt{3}\sin t)\,\mathbf{i} + 2\cos t\,\mathbf{j} + (\sqrt{3}t - \sin t)\,\mathbf{k}.$$

Deduce from those computations that the curve must be a helix. Confirm it by plotting the curve. Then find another helix

$$\mathbf{R}(t) = a\cos t\,\mathbf{i} + a\sin t\,\mathbf{j} + bt\,\mathbf{k}$$

and an isometry $\mathbf{F}$ such that $\mathbf{F}(\mathbf{r}) = \mathbf{R}$. (Hint: It's a rotation. To find it, compute the two 3×3 matrices consisting of the values of the frame fields for $\mathbf{r}$ and $\mathbf{R}$ at zero. Then find the matrix that moves one into the other. You will have to use the values of the speed, curvature, and torsion that you compute to find the correct choices of a and b. That is, find these data for $\mathbf{R}$ in terms of a and b and use `solve`.)

Problem 3.7. As a point (or particle) sweeps over a curve $\mathbf{r}(t)$, the centers of its corresponding osculating circles sweep out another curve, called the *evolute* of $\mathbf{r}$. Recall that the osculating circle is the unique circle in the osculating plane (determined by $\mathbf{T}$ and $\mathbf{N}$) that is tangent to the curve and has the same curvature. Using the fact that the curvature of a circle of radius r is $1/r$, find a formula for the center of the osculating circle in terms of $\mathbf{r}(t)$, $\mathbf{T}(t)$, $\mathbf{N}(t)$, and $\kappa(t)$. (Hint: It may be convenient to compute $\mathbf{N}$ from the formula relating it to $\mathbf{a}$ and $\mathbf{T}$.) Use that formula to obtain a parametric equation for the evolute of the following curves. In each case, plot the original curve and its evolute on the same graph. (Note: In part (c), you must enter the parametrization using fractions and not decimals.)

(a) The ellipse: $2x^2 + y^2 = 1$.

(b) The cycloid: $\mathbf{r}(t) = (t - \sin t, 1 - \cos t)$, $0 \le t \le 8\pi$.

(c) The helical log spiral: $\mathbf{r}(t) = (e^{t/10}\cos t, e^{t/10}\sin t, t/10)$, $0 \le t \le 6\pi$.

(d) The tractrix: Use the following parametrization:

$$\mathbf{r}(t) = (\sin t, \cos t + \log(\tan(t/2))), \qquad 0 \le t \le \pi.$$

Show via the transformation $u = \log(\tan(t/2))$ that the evolute of the tractrix is a catenary. (Hint: When you first compute the parametrization of the evolute, you may end up with a mixture of trig functions of both t and $t/2$. It helps to replace $\sin t$ by $2\sin(t/2)\cos(t/2)$ and then to replace $t/2$ by $\arctan(e^u)$.)

Problem 3.8. In this problem, we will attempt to explain what it means to *deform* one curve into another. Intuitively, we mean something like the following. Suppose we have two smooth plane curves $\mathbf{r}_1(t)$ and $\mathbf{r}_2(t)$, each having the same initial and terminal points $\mathbf{r}_1(0) = \mathbf{r}_2(0)$, $\mathbf{r}_1(1) = \mathbf{r}_2(1)$. (Note that we conveniently assumed both curves were parametrized on the interval $0 \le t \le 1$. Because of our ability to reparametrize curves, that is no loss of generality.) We say that the curves are deformations of each other if we can *continuously* move the first into the second without cutting or tearing the curve. Here is a more precise mathematical definition. There should be a continuous vector-valued function of two variables $\Theta(t,u)$, $0 \le t \le 1$, $0 \le u \le 1$, so that

$$\Theta(t,0) = \mathbf{r}_1(t), \qquad \Theta(t,1) = \mathbf{r}_2(t), \qquad t \in [0,1].$$

It is thus clear that as u progresses from 0 to 1 the curves $t \mapsto \Theta(t,u)$ constitute the continuous deformation. For each of the following pairs of curves, find a continuous deformation between them; i.e., produce Θ, and then draw a family of curves that illustrates the deformation. Label the two curves on your picture. (Look up **`text`** or **`annotation`** in the Help browser to see how to do the labeling in MATLAB.)

(a) The portion of the unit circle in the first quadrant, and the portion of the curve $x^{1/2} + y^{1/2} = 1$ in the first quadrant.

(b) The two lemniscates of Bernoulli given by

$$\mathbf{r}_j(t) = \left(\frac{j\cos t}{1+\sin^2 t}\right)\mathbf{i} + \left(\frac{j\cos t \sin t}{1+\sin^2 t}\right)\mathbf{j},$$

$j = 1$ or 5.

Problem 3.9. We have concentrated most of our attention on curves defined by parametric equations. In single-variable calculus, you studied curves in the form $y = f(x)$ and also in the form $F(x,y) = 0$. The former can be parametrized by x itself (to wit, $\mathbf{r}(x) = (x, f(x))$, but the issue of whether the locus of points (x,y) satisfying an implicit equation like $F(x,y) = 0$ can be given in parametric coordinates is quite subtle. It is bound up with the *Implicit Function Theorem*, about which we will have a little to say in Chapter 6. For the moment, let us simply investigate implicitly defined curves via the command **`fimplicit`**.

(a) Draw each of the following curves. Make sure to try several different ranges to be sure that you display all the interesting features of the curve.

(i) $x^2 + y^2 = 1$.

(ii) $x^2 = y^3$.

(iii) $x^3 - x = y^2$.

(iv) $x^3 + y^3 - 3xy = \varepsilon$, with $\varepsilon = 0.1$, 0, and -0.1.

(b) From the pictures you obtain, explain why even if F is a "very nice" function, its locus of zeros may not be a curve in the sense in which we have been studying it. In particular, all of our parametrized curves are connected—they only have one piece. Use the Intermediate Value Theorem from one-variable calculus to show that a parametrized curve cannot have two pieces which are a fixed horizontal distance apart (e.g., all the points on one piece satisfy $x \leq 0$ and on the other piece $x \geq 1$).

Problem 3.10. Geometers recognize two particularly simple kinds of singularities on curves: *cusps* and *nodes*. In this problem, we will compute the geometric invariants of the simplest example of each kind of singularity.

(a) The simplest curve with a cusps, also called a semicubical parabola, is defined implicitly by the equation $y^2 = x^3$. Show that the formula

$$\mathbf{r}(t) = (t^2, t^3), \qquad t \in \mathbb{R},$$

gives a parametrization of the semicubical parabola.

(b) Graph the semicubical parabola. (You may use the implicit equations or the parametric equations.) Describe any significant geometric features of the graph.

(c) Compute the Frenet frame, speed, curvature, and torsion of the semicubical parabola. Which of these invariants, if any, can be used to detect the singularity?

(d) The simplest nodal curve is defined implicitly by the equation $y^2 = x^3 + x^2$. Show that the formula

$$\mathbf{r}(t) = (t^2 - 1, t^3 - t), \qquad t \in \mathbb{R},$$

gives a parametrization of the nodal curve.

(e) Graph the nodal curve. (You may use the implicit equations or the parametric equations.) Describe any significant geometric features of the graph.

(f) Compute the Frenet frame, speed, curvature, and torsion of the nodal curve. Which of these invariants, if any, can be used to detect the singularity?

(g) Write a paragraph explaining the differences between your results for the semicubical parabola and your results for the nodal curve.

Problem 3.11. It is natural to ask: given a smooth nonnegative function $\kappa(s)$, is there a smooth curve $\mathbf{r}(s)$ having $\kappa(s)$ as its curvature? As the notation suggests, there is no harm in asking that $\mathbf{r}$ be unit speed. The solution is straightforward. We only need to set

$$\theta(s) = \int_0^s \kappa(u)\,du$$

and

$$\mathbf{r}(s) = \left(\int_0^s \cos\theta(u)\,du\right)\mathbf{i} + \left(\int_0^s \sin\theta(u)\,du\right)\mathbf{j}.$$

(a) Show that the curve $\mathbf{r}(s)$ so defined indeed has curvature $\kappa(s)$. (In order to do this problem, you want to use a syntax like **`syms kappa(u)`** to define a symbolic function of u without assigning a formula for it.)

(b) Use the formulas to find, and then graph, a curve whose curvature is

$$\kappa(s) = \frac{1}{\sqrt{1-s^2}}, \qquad -1 < s < 1.$$

(c) Use the formulas to find, and then graph, a curve whose curvature is

$$\kappa(s) = \frac{1}{1+s^2}, \qquad s \in \mathbb{R}.$$

(d) The method works because you (or MATLAB) can do the antidifferentiation. But the theory is correct even when the antidifferentiation is not possible. In that case, we can invoke the MATLAB command **`ode45`** to find a numerical solution to the system of *differential equations*

$$x'(s) = \cos\theta(s), \qquad y'(s) = \sin\theta(s), \qquad \theta'(s) = \kappa(s).$$

Look up the syntax of **`ode45`** and use it to solve the system in the case $\kappa(s) = s + \sin s$. (Choose appropriate initial data.) Plot the solution. Note: an example of the correct syntax appears in Chapter 4. You might need the options to **`ode45`** described in Problem 4.6.

Problem 3.12. Write an essay of about 500 words entitled *How Calculus Helps Us to Understand the Geometric Nature of Curves*. In your essay, you should explain how a curve is an intrinsically geometric notion which we can describe and study analytically by formulas and calculus operations. Your essay should contain only text—no mathematics formulas. It should contain two parts: in the first, explain as clearly as possible your intuitive geometric understanding of curves and their invariants; in the second, describe how those geometric notions are made precise by means of calculus. Your essay should answer, at a minimum, the following questions:

(a) What does it mean to express a geometric invariant of a curve analytically?

(b) How does the very definition of "curve" give analytic content to what is geometrically an intuitive idea?

(c) Why are curves one-dimensional objects?

(d) What does it mean for a geometric property or invariant of a curve to be independent of parametrization?

(e) What does it mean for two curves to be congruent?

(f) What does it mean to say that the three geometric invariants of arclength, curvature, and torsion together characterize the shape, but perhaps not the position, of a curve?

You should include any other points you feel will help convey that you understand the geometric nature of curves and that that nature can be revealed analytically via functions from vector calculus.

Problem 3.13. In this problem we will be concerned with finding curves having prescribed curvature and torsion properties. Consider a curve $\mathbf{r}$ of the form

$$\mathbf{r}(t) = \int f(t)\cos t\,dt\,\mathbf{i} + \int f(t)\sin t\,dt\,\mathbf{j} + \int f(t)g(t)\,dt\,\mathbf{k},$$

where f and g are smooth functions and f is positive.

(a) Compute the curvature and torsion of $\mathbf{r}$ in terms of these functions and their derivatives.

(b) Use those formulas to construct a specific curve $\mathbf{r}_1$ which has constant curvature, but nonconstant torsion.

(c) Use the formulas to construct a specific curve $\mathbf{r}_2$ which has constant torsion, but nonconstant curvature. (Hint: Select $g(t) = \sinh t$ in parts (b) and (c). Then choose f appropriately.)

(d) Have MATLAB attempt to carry out the symbolic antidifferentiations in the components of the curves $\mathbf{r}_1$ and $\mathbf{r}_2$ with the functions you have chosen. If you succeed, proceed to part (e). If you are unsuccessful, then do the integrals numerically (e.g., with **`ode45`** or **`integral`**) to find the antiderivatives.

(e) Use your results from (d) to plot the two curves.

Problem 3.14. Fix a circle of radius a. Let a second circle of radius b roll around the outside of the first circle without slipping. The *epicycloid* is the curve traced by a point on the circumference of the rolling circle.

(a) Find parametric equations for the epicycloid.

(b) Graph the epicycloid for at least three values of a and b.

(c) Compute the speed and curvature of the epicycloid. Relate any points where these invariants are zero or undefined to geometric features of the graph. (Hint: The standard program in this chapter may take a long time to compute the curvature, since it gets hung up trying to **`simplify`** intermediate steps. Try computing from the formulas for variable-speed curves, stripping out all the simpli-

fication commands until you get to the end. You get a simpler result if you first compute the square of the curvature and then take the square root.)

(d) Find the evolute (see Problem 3.7) of the epicycloid. When $a = 3$ and $b = 1$, graph the epicycloid, the fixed circle, and the evolute on the same axes. (The hint for (c) applies here as well.)

Problem 3.15. The *hypocycloid* is the path traced by a point on the circumference of a wheel of radius b rolling on the inside of another circle of radius a. Follow the instructions for Problem 3.14, using the hypocycloid instead of the epicycloid. Use $a = 11$, $b = 7$ when you graph the hypocycloid and its evolute.

Problem 3.16. A reflector is attached to a bicycle wheel of radius a at a distance b from the center. As the wheel rolls without slipping, the reflector sweeps out a curve called a *trochoid*. Follow the instructions for Problem 3.14, using the trochoid instead of the epicycloid. Use $a = 1$, $b = 1/2$ when you graph the trochoid and its evolute, and replace the fixed circle in part (d) by the straight line on which the wheel is rolling.

Problem 3.17. Write a function script called `geometricInvariants.m` that takes two arguments: a parametrization of a curve and the name of the parameter. This program should return a structure array (see the help on the **`struct`** command) containing the components of the Frenet frame along with the speed, curvature, and torsion.

Glossary of MATLAB Commands

`assume` Set assumptions on a symbolic object

`axis` Selects the ranges of x and y to show in a plot

`cross` Cross product in 3-space

`diff` Differentiates an expression

`double` Converts an expression to a double-precision numerical value

`dsolve` Symbolic differential equation solver

`fimplicit` Symbolic plotter for implicitly given curves in 2D

`fplot` Symbolic plotter in 2D

`fplot3` Symbolic plotter for curves in 3D

`int` Integrates an expression

`integral` Numerically integrates a (vectorized) function

`legend` Attaches a legend to a combination of plots

`ode45` Numerical differential equation solver

`simplify` Used to simplify a symbolic expression. Note that sometimes you will need to increase the option **`Steps`** for best results.

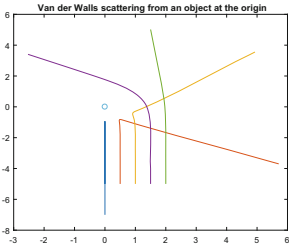
`struct` Creates a structure array

`syms` Defines symbolic variables or functions

`subs` Substitute for a variable in an expression

`transpose` Computes the transpose of a symbolic matrix

Chapter 4
Kinematics



The study of curves in space is of interest not only as a topic in geometry but also for its application to the motion of physical objects. In this chapter, we develop a few topics in mechanics from the point of view of the theory of curves. Additional applications to physics will be considered in Chapter 10.

4.1 Newton's Laws of Motion

In the last chapter, we alluded to the fact that a parametrized curve $t \mapsto \mathbf{r}(t) = (x(t), y(t), z(t))$, with t representing time and $\mathbf{r}$ representing position, may be used to describe the motion of an object in space. This works provided the object is small enough compared to the distances over which it travels that we may represent it as a point. Even if the moving object is large, it is often convenient to replace it by an idealized particle concentrated at its center of mass; it often turns out that the center of mass of the macroscopic object moves in the same way as this idealized particle. We will, therefore, only consider point objects in this chapter.

The basic principles of both motion and the calculus were enunciated by Isaac Newton in his book *Philosophiæ Naturalis Principia Mathematica*, or *Mathematical Principles of Natural Philosophy*, in 1687. The first two of Newton's laws of motion read as follows [1, p. 19]:

> Every body perseveres in its state of rest, or of uniform motion in a right line, unless it is compelled to change that state by forces impressed thereon.
>
> The alteration of motion is ever proportional to the motive force impressed; and is made in the direction of the right line in which that force is impressed.

In deference to Newton, it is traditional in mechanics to use his *dot notation* for derivatives with respect to time. (This is the only place in calculus where Newton's notation has triumphed over that of Leibniz.) For an object represented mathematically by a parametrized curve $t \mapsto \mathbf{r}(t)$, the first law says that the velocity $\mathbf{v}(t) = \dot{\mathbf{r}}(t)$ remains constant in the absence of outside forces. The second law says that the rate of change of the velocity, or in other words the acceleration $\mathbf{a}(t) = \dot{\mathbf{v}}(t) = \ddot{\mathbf{r}}(t)$, is

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_4

proportional to the force exerted. We usually summarize both laws in a single equation: $\mathbf{F} = m\mathbf{a}$, where $\mathbf{F}$ is the total force acting on the object, $\mathbf{a}$ is the acceleration, and m is the mass of the object. (Newton established later in his book that the mass is the correct constant of proportionality.) This law is a second-order differential equation for the vector-valued function $\mathbf{r}(t)$, or a system of three second-order scalar differential equations for the components $x(t)$, $y(t)$, and $z(t)$ of $\mathbf{r}(t)$.

A number of special cases of the equation $\mathbf{F} = m\ddot{\mathbf{r}}$ are of particular interest. In studying them, it is useful to introduce the *kinetic energy*, $E_{\text{kin}} = \frac{1}{2}m\|\mathbf{v}\|^2$. We find that

$$\frac{d}{dt}E_{\text{kin}} = \frac{d}{dt}\left(\tfrac{1}{2}m\|\mathbf{v}\|^2\right) = \tfrac{1}{2}m\frac{d}{dt}(\mathbf{v}\cdot\mathbf{v}) = \mathbf{v}\cdot(m\dot{\mathbf{v}}) = \mathbf{v}\cdot\mathbf{F}. \tag{4.1}$$

Thus, if $\mathbf{F}$ is always perpendicular to the velocity $\mathbf{v} = \dot{\mathbf{r}}$, then (4.1) says that the kinetic energy remains constant, and that the object moves with constant speed. This case arises when a charged particle moves under the influence of a magnetic field $\mathbf{B}$. The magnetic force on a charged particle with charge q and position vector $\mathbf{r}$ is

$$\mathbf{F}_{\text{mag}} = \frac{-q}{c}\mathbf{B}\times\dot{\mathbf{r}}, \tag{4.2}$$

where c is a universal constant, the speed of light in a vacuum. Since the cross product $\mathbf{B}\times\dot{\mathbf{r}}$ is perpendicular to $\dot{\mathbf{r}}$, the charged particle moves at constant speed v. From the theory of constant-speed curves in the last chapter, we can compute the curvature of the motion:

$$\kappa = \frac{1}{v^2}\|\mathbf{a}\| = \frac{q}{mv^2c}\|\mathbf{B}\times\dot{\mathbf{r}}\| = \frac{q}{mvc}\|\mathbf{B}\|\sin\theta,$$

where θ is the angle between the magnetic field $\mathbf{B}$ and the velocity $\mathbf{v}$.

Another case of interest is that of motion in a *central force field*, where the force $\mathbf{F}$ on the object is a *time-independent* function of the position $\mathbf{r}$, and this function is of the very special form

$$\mathbf{F} = f(r)(\text{unit vector pointing toward the origin}),$$

where $r = \|\mathbf{r}\|$ is the distance to the origin, and f is an ordinary scalar function. Since the unit vector pointing toward the origin is $-\mathbf{r}/r$, motion in a central force field is given by the differential equation

$$\ddot{\mathbf{r}} = -\frac{f(r)}{mr}\mathbf{r} = g(r)\mathbf{r}. \tag{4.3}$$

This differential equation is *autonomous*, i.e., the right-hand side of (4.3) does not depend explicitly on t. When $f(r) > 0$, the force on the object points toward the origin with magnitude $f(r)$. This is the case of an *attractive force*. The standard example, as originally studied by Newton, is a planet moving around a sun (located at the origin) under the influence of gravity. In this case, the *inverse square law*

says that the magnitude $f(r)$ of the force is inversely proportional to r^2, the square of the distance to the origin. More precisely, *Newton's law of gravitation* says that $f(r) = GMm/r^2$, where M is the mass of the Sun, m is the mass of the planet, and G is a universal gravitational constant. Electrical forces between point charges of opposite sign are also attractive and also obey the inverse square law, but with $f(r)$ now depending on the product of the charges. When $f(r) < 0$, the force on the object points away from the origin, and this is the case of a *repulsive force*. For example, the electric force between charges of the same sign is repulsive.

To study equation (4.3) for motion in a central force field, it is traditional to introduce the *angular momentum* vector

$$\mathbf{L} = m\mathbf{r} \times \mathbf{v}. \tag{4.4}$$

Differentiating (4.4), and using the product rule, we find that

$$\dot{\mathbf{L}} = m\dot{\mathbf{r}} \times \mathbf{v} + m\mathbf{r} \times \dot{\mathbf{v}} = m\mathbf{v} \times \mathbf{v} + m\mathbf{r} \times g(r)\mathbf{r} = \mathbf{0} + \mathbf{0} = \mathbf{0}.$$

Thus, $\mathbf{L}$ does not change with time; it is a fixed quantity. This is the *law of conservation of angular momentum*. Since $\mathbf{L}$ is perpendicular to both $\mathbf{r}$ and $\mathbf{v}$, if the latter are not collinear, then $\mathbf{L}$ is normal to the unique plane containing them. Constancy of $\mathbf{L}$ therefore implies that the plane containing $\mathbf{r}$ and $\mathbf{v}$ does not change, and thus *the curve traced out by the object lies in a plane.*

We may obtain the same conclusion using the theory of the torsion of curves from the last chapter. There we had the formula

$$\tau = \frac{(\dot{\mathbf{r}} \times \ddot{\mathbf{r}}) \cdot (\ddot{\mathbf{r}})^{\cdot}}{\|\dot{\mathbf{r}} \times \ddot{\mathbf{r}}\|^2}$$

for the torsion of the curve traced out by $\mathbf{r}(t)$. Substituting $g(r)\mathbf{r}$ for $\ddot{\mathbf{r}}$, we obtain the equation

$$\tau = \frac{(\dot{\mathbf{r}} \times g(r)\mathbf{r}) \cdot (g(r)\mathbf{r})^{\cdot}}{\|\dot{\mathbf{r}} \times g(r)\mathbf{r}\|^2}.$$

But

$$(g(r)\mathbf{r})^{\cdot} = (g(r))^{\cdot}\,\mathbf{r} + g(r)\dot{\mathbf{r}},$$

which is a sum of a scalar multiple of $\mathbf{r}$ and a scalar multiple of $\dot{\mathbf{r}}$, hence perpendicular to $\dot{\mathbf{r}} \times g(r)\mathbf{r}$. Thus, the numerator vanishes and $\tau = 0$; i.e., the motion is planar. The experimental evidence for this conclusion is exceptionally good. In our own solar system, it turns out that to a high degree of approximation, the motions of the Sun and of all of the planets lie in a single plane called the *ecliptic*. The motion of Pluto, a dwarf planet, also lies in a plane, though it is slightly tilted with respect to the ecliptic.

4.2 Kepler's Laws of Planetary Motion

By analyzing equation (4.3) using his theory of gravitation, Newton [1, Book III] was able to derive all the laws of planetary motion previously found purely phenomenologically by Kepler. One of these laws is that the motion of a planet stays in a plane; as we have seen, this is a property of motion in any central force field. Another law of a similar nature is that a planet "sweeps out equal areas in equal lengths of time." In other words, as a planet moves from a position P to a position Q around the sun at position S, the area of the region $\sphericalangle SPQ$ bounded by the line segments $\overline{SP}$ and $\overline{SQ}$ and by the path (or orbit) of the planet from P to Q is proportional to the time taken in going from P to Q and otherwise independent of the choices of P and Q. To derive this law, note that for P and Q very close together corresponding to times t_0 and $t_0 + \Delta t$, $\sphericalangle SPQ$ is approximately a triangle, so its area is approximately

$$\frac{1}{2}\,\|\mathbf{r}(t_0) \times \mathbf{r}(t_0 + \Delta t)\| \approx \frac{1}{2}\,\|\mathbf{r}(t_0) \times (\mathbf{r}(t_0) + (\Delta t)\dot{\mathbf{r}}(t_0))\| = \frac{1}{2}\Delta t\,\|(\mathbf{r} \times \dot{\mathbf{r}})(t_0)\|.$$

But $\mathbf{r} \times \dot{\mathbf{r}}$ is just $\mathbf{L}/m$, which is constant, so the area is the constant $\|\mathbf{L}\|/2m$ times the time interval Δt. Figure 4.1 shows the region $\sphericalangle SPQ$ along with a second region of equal area.

Kepler's other two laws depend on the fact that gravitation obeys an inverse square law. They say that planets travel in ellipses or circles, and that more generally, objects moving under the central force field of the sun move in conic sections: circles, ellipses, hyperbolas, or parabolas, with the sun at one focus. Furthermore, for a planet in an elliptical orbit, the period of the orbit (the time it takes for one complete revolution around the sun) is proportional to the $\frac{3}{2}$ power of the major axis of the orbit. These two laws will be studied in Problem Set D.

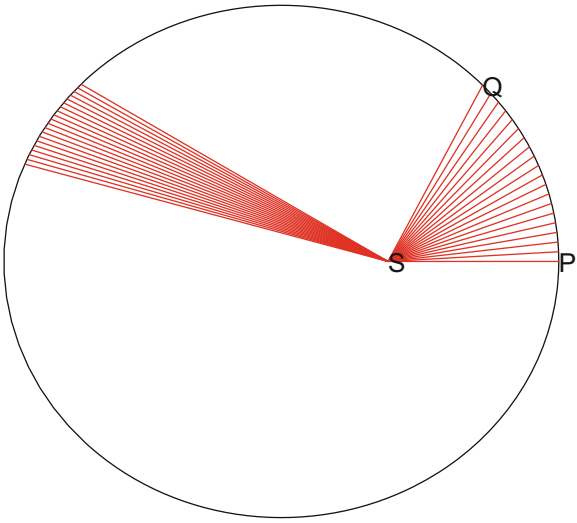


Fig. 4.1 Kepler's Law of Equal Areas

4.3 Studying Equations of Motion with MATLAB

MATLAB provides a convenient framework for studying Newton's equation of motion $\mathbf{F} = m\ddot{\mathbf{r}}$, once we understand how to use the two major MATLAB commands for solving differential equations, **dsolve** and **ode45**. The command **dsolve** works only with the (somewhat limited) class of differential equations that can be solved explicitly in closed form. It gives its output as a symbolic expression for the unknown functions, possibly involving various parameters. To illustrate how it works, we solve the equation of motion for a charged particle with charge q, mass m, and initial velocity $\mathbf{v}(0) = (v_1, v_2, v_3)$, in a uniform magnetic field $\mathbf{B} = B\mathbf{k}$. We take the initial position to be at the origin. The syntax for **dsolve** should be clear from this example.

```
>> syms q c m t v1 v2 v3 B x(t) y(t) z(t)
>> rhs = (-q*B/(c*m))*cross([0,0,1],[diff(x), diff(y), diff(z)])
rhs(t) =
[ (B*q*diff(y(t), t))/(c*m), -(B*q*diff(x(t), t))/(c*m), 0]

>> sol = dsolve([diff(x,2),diff(y,2),diff(z,2)] == rhs, ...
  [x(0),y(0),z(0)]==[0,0,0], [Dx(0),Dy(0),Dz(0)]==[v1,v2,v3]);
```

To use the answer, we need to extract the various components of **sol**. For example, since there is no force parallel to the magnetic field (in the z-direction), we expect constant-speed motion in the z-direction. And indeed:

```
>> sol.z
ans =
t*v3
```

If, for example, we take $v1 = 0$ and set $v2$, B, q, c, and m all to 1, we obtain:

```
>> assume(t, 'real'); assume(v3, 'real'); simplesol = ...
simplify(subs([sol.x,sol.y,sol.z],[c,m,q,B,v1,v2],[1,1,1,1,0,1]))
simplesol =
[ -(exp(-t*1i)*(exp(t*1i) - 1)^2)/2, sin(t), t*v3]
```

MATLAB has given the answer in part in terms of complex exponentials. To check that this expression is indeed real and to write it in terms of trigonometric functions, we can do the following.

```
>> simplify(imag(simplesol))
ans =
[ 0, 0, 0]

>> simplify(real(simplesol))
ans =
[ 1 - cos(t), sin(t), t*v3]
```

We recognize this from the last chapter as the parametrization of a helix.

The **ode45** command works slightly differently. It can deal with essentially any system of ordinary differential equations but only produces a numerical approximation to a solution, in the form of a table of values of the dependent variable(s) for various values of the independent variable (which for present purposes is always time, t). Since **ode45** works numerically, all parameters and initial conditions in the equations must be specified explicitly, and we must specify a domain for the independent variable. Furthermore, **ode45** is set up for first-order differential equations only. For applications to Newton's laws, we get around this by thinking of $\mathbf{F} = m\mathbf{a}$ as a first-order differential equation in a vector variable:

$$\frac{d}{dt}\left(\mathbf{x}(t), \mathbf{v}(t)\right) = \left(\mathbf{v}(t), \frac{1}{m}\mathbf{F}(\mathbf{x}(t))\right).$$

As an example, here is an application of **ode45** to the case we were just discussing (the motion of a charged particle in a uniform magnetic field). For instance, if we take $v1 = 0$, $v2 = v3 = 1$ and set b, q, c, and m all to 1, we proceed as follows. The variable **t** is a column vector of values of t, and **w** is an array with six columns, corresponding to the three components of the particle's position and the three components of the particle's velocity.

```
>> [t,w] = ode45(@(t,w) [w(4); w(5); w(6); w(5); -w(4); 0],...
[0 6*pi],[0; 0; 0; 0; 1; 1]); plot3(w(:,1),w(:,2),w(:,3))
```

We indeed see the characteristic shape of a helix. The output of the MATLAB code is shown in Figure 4.2.

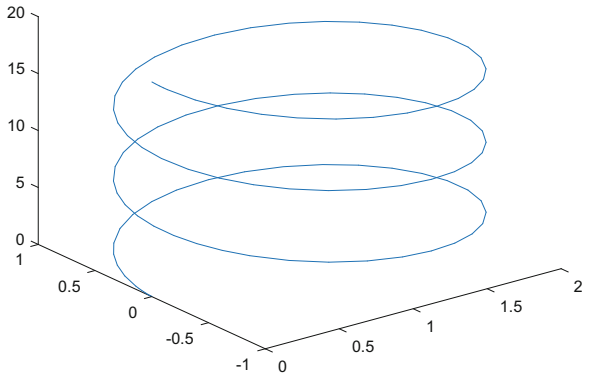


Fig. 4.2 Helical path of a charged particle in a uniform magnetic field

Problem Set D. Kinematics

Problem 4.1. This problem continues the study, begun in Chapter 4, of the motion of a charge q under the influence of a magnetic field $\mathbf{B}(\mathbf{r})$. We assume $\mathbf{B}(\mathbf{r})$ varies with position in space but is *static*, i.e., time-independent, and that there are no other forces acting on the charge. For convenience in the calculations, take $q=m=c=1$ (i.e., ignore constant factors). Recall that the force in this situation is given by equation (4.2).

(a) Using **dsolve**, redo the exact solution of the equations of motion in the case where $\mathbf{B}=\mathbf{k}$ is constant and $\mathbf{v}(0)$, the initial value of $\mathbf{v}$ at time $t=0$, is given but arbitrary. (You can adapt the MATLAB code from Chapter 4.) Show that the charge moves in a circle if $\mathbf{v}(0)$ is a nonzero vector with vanishing $\mathbf{k}$-component, and that it moves in a helix if $\mathbf{v}(0)$ has both a nonzero $\mathbf{i}$- or $\mathbf{j}$-component and a nonzero $\mathbf{k}$-component. Compute the curvature and torsion to verify the condition for a right circular helix. By choosing specific initial data, draw pictures of both a circular and a helical trajectory.

(b) Solve the equations of motion numerically, using **ode45**, in the case where $\mathbf{B}(x,y,z)=(-y\mathbf{i}+x\mathbf{j})/(x^2+y^2)$. This corresponds to the magnetic field due to a current flowing along the z-axis. Take the time interval to be $[-5,5]$ in each case. Using **plot3**, graph the resulting trajectories. Try to interpret the results. Use the following initial conditions:

(i) $\mathbf{r}(0)=(1,0,0)$, $\mathbf{r}'(0)=(1,0,0)$.

(ii) $\mathbf{r}(0)=(1,0,0)$, $\mathbf{r}'(0)=(2,0,0)$.

(iii) $\mathbf{r}(0)=(1,0,0)$, $\mathbf{r}'(0)=(1,0,1)$.

(iv) $\mathbf{r}(0)=(1,0,0)$, $\mathbf{r}'(0)=(2,0,1)$.

Problem 4.2. One of the simplest cases where Newton's law of gravitation can be applied is that of the motion of a ball near the surface of the Earth.

(a) We begin by neglecting air resistance. Let us choose coordinates so that the y-axis points straight up and the x-axis is along the ground. Then the gravitational force on the ball is $\mathbf{F}=(0,-mg)$, where g is the acceleration due to gravity. Suppose a batter hits a baseball from the position $(0,h)$ with initial velocity $\mathbf{v}_0=(u_1,u_2)$ at time $t=0$, and thereafter the only force acting on the ball is that of gravity. Solve the equations of motion (with **dsolve**) for the trajectory $\mathbf{r}(t)$ of the ball, and show that the ball moves in an arc of a parabola (assuming $u_1>0$). Find the distance d traveled by the ball until it hits the ground, and the time t_0 taken to travel that distance. (You need to solve the vector equation $\mathbf{r}(t_0)=(d,0)$, with a *positive* value of t_0. Your answers should involve h, u_1, u_2, and g.)

(b) Suppose that the speed $v_0=\sqrt{u_1^2+u_2^2}$ at which the ball leaves the bat is fixed, but that the batter can adjust the angle θ at which the ball is hit. In other

words, take $\mathbf{v}_0 = v_0(\cos\theta, \sin\theta)$, where $-\pi/2 \le \theta \le \pi/2$. (Here, $\theta = -\pi/2$ corresponds to the batter hitting a foul tip that bounces off home plate, and $\theta = \pi/2$ corresponds to the batter popping the ball straight up.) Find the value of θ that maximizes the distance d traveled by the ball. A good way to do this is to write d as a function of $s = \sin\theta$, where s varies from -1 to 1, and set the derivative with respect to s equal to 0. Also, find that value of θ that maximizes the time t_0 that the ball remains in the air. (Your answers should involve h, v_0, and g.) Do your answers match your expectations? (Keep in mind that h is very small compared to the maximum height of a fly ball.)

(c) Assuming the batter adjusts θ to maximize the distance d, obtain numerical values for d (in feet) and t_0 (in seconds) if $g = 32\,\text{ft/sec}^2$, $h = 4\,\text{ft}$, and $v_0 = 100\,\text{ft/sec}$ (about 68 mi/hr). Do your values for d and t_0 seem reasonable?

(d) Suppose the center field fence is 400 feet from home plate. How much harder does the batter have to hit the ball so that it reaches the fence?

(e) How fast does the batter have to hit the ball so that it *clears* the center field fence, assuming that the fence is 10 feet high? (This problem is harder; it requires redoing the calculations to maximize, as a function of θ, the distance at which the height of the ball is 10 feet instead of 0 feet, and then solving to see for what value of v_0 this distance reaches 400 feet.)

(f) To obtain a more reasonable model, we should include the effect of air resistance. Then the total force on the ball is

$$\mathbf{F} = (0, -mg) - \frac{mg}{v_{\text{term}}}\dot{\mathbf{r}},$$

where the air resistance term involves the quantity v_{term}, the *terminal velocity* of the ball. (This is the speed at which the ball would eventually fall if dropped from a great height.) Redo the solution of the equations of motion. Plot the trajectory of the ball (with the initial conditions from part (c)), both with and without air resistance (assuming, say, that $v_{\text{term}} = 300\,\text{ft/sec}$). Superimpose the two plots. Do you see any significant difference?

Problem 4.3. Consider an object with position vector $\mathbf{r}(t) = (x(t), y(t), z(t))$, of mass $m = 1$, moving in a central force field

$$\mathbf{F} = -\frac{1}{\|\mathbf{r}\|^3}\mathbf{r},$$

which is an idealized model of a planet moving around a massive sun. (For convenience, we have chosen units so as to set the attractive constant GM to 1.) For simplicity, we will assume the object moves in the x-y plane.

(a) If the planet moves in a circular orbit, then its speed v is a constant. Why? (Use equation (4.1) in Chapter 4.)

(b) Continue to assume that the planet moves in a circular orbit. What is the relationship between the radius r of the orbit and the speed v of the planet? Also, find the period T of the orbit as a function of r. Your answer should be a form of one of *Kepler's laws of planetary motion*. (Hint: In this case, r is a constant, so the motion is described by the angular coordinate θ. The speed is $v = r\omega$, where $\omega = \dot{\theta}$ is the angular velocity, a constant of the motion, and $\mathbf{r} = (r\cos\theta, r\sin\theta) = (r\cos\omega t, r\sin\omega t)$. Differentiate twice with respect to t and substitute in Newton's law of gravitation.)

(c) Use **`ode45`** to solve numerically the equations of motion:

$$\ddot{x}(t) = -\frac{x(t)}{(x(t)^2 + y(t)^2)^{3/2}}, \qquad \ddot{y}(t) = -\frac{y(t)}{(x(t)^2 + y(t)^2)^{3/2}},$$

with the initial conditions $x(0) = 1$, $y(0) = 0$, for $0 \le t \le 10$, in the three cases:

(i) $\dot{x}(0) = 0, \dot{y}(0) = 1.$

(ii) $\dot{x}(0) = 0, \dot{y}(0) = 1.2.$

(iii) $\dot{x}(0) = 0, \dot{y}(0) = 2.$

Display the trajectory in each case using **`plot`**. (You may need to adjust the **`axis`** so that the scales on the two axes are the same.) Which conic sections do you observe? In one case, you may want to allow a longer time interval to clearly identify the trajectory. In another case, you may want to augment the time interval to allow "negative time."

Problem 4.4. This problem is similar to Problem 4.3, except that the central force field does not quite obey the inverse square law. Suppose now that a particle of mass 1 with position vector $\mathbf{r}(t) = (x(t), y(t), z(t))$ is moving in a central force field

$$\mathbf{F} = -\frac{1}{\|\mathbf{r}\|^{\alpha}}\mathbf{r},$$

where the parameter α is not necessarily equal to 3.

(a) As before, if the planet moves in a circular orbit, find the relationship between the radius r of the orbit and the speed v of the planet. Your answer should depend on α. Also, find the period T of the orbit as a function of r and α. Explain how you can use your answer to provide an experimental test of the inverse square law from the fact that the radius of the orbit of Mars is 1.524 times that of the Earth, while a year on Mars is 687.0 days long (compared with 365.26 days on Earth). Keep in mind that you cannot expect perfect accu-

racy since we have neglected the eccentricities of the orbits and the fact that the planets exert gravitational forces on one another.

(b) Repeat the calculations of Problem 4.3(c), but with $\alpha = 4$ (an inverse cube law) instead of $\alpha = 3$. How do the trajectories differ from what they would be with an inverse square law, and why?

Problem 4.5. This problem explores two other central force fields that arise in molecular and nuclear physics.

(a) The Yukawa force field, which models short-range attractive force such as those that hold an atomic nucleus together, gives the equation of motion

$$\ddot{\mathbf{r}} = \frac{\mathbf{r}}{mr}\frac{d}{dr}\left(\frac{be^{-ar}}{ar}\right),$$

where a and b are positive constants. (The quantity b measures the strength of the force field, and a measures its "range." More precisely, the force is significant only at distances on the order of $1/a$. Of course, we are using a somewhat inappropriate model, since nuclear forces can only be understood using quantum mechanics, and we are using classical mechanics here.)
Find, for this force field, the relationship between the speed v and the radius r for circular orbits. Compute the period T as a function of r. (The result will depend on a and b.) Compare the results to the situation in Kepler's laws, where v is inversely proportional to $\sqrt{r}$ and T is proportional to $r^{3/2}$. To visualize what happens, it is convenient to use **`loglog`** to draw a log-log plot of T as a function of r. (For convenience, take $m = a = b = 1$.) Recall that the log-log plot of a power function is a straight line, with slope equal to the exponent of the power law. So the deviation of the log-log plot from a straight line measures the deviation from a power law.

(b) The van der Waals force field, which models the interaction between atoms or molecules, is mildly attractive at larger distances, and strongly repulsive at very short distances. (This reflects the fact that molecules resist being pushed into one another. Again, we are using a somewhat inappropriate model, since molecular forces ought to be modeled with quantum mechanics.) Try the force field

$$\ddot{\mathbf{r}} = \frac{\mathbf{r}}{r}\frac{d}{dr}\left(\frac{1}{r^6} - \frac{1}{r^{12}}\right),$$

which is a version of a commonly used model. Plot the force as a function of r (with a suitable **`axis`**) to see that the force is indeed strongly repulsive for $r < 1$ and mildly attractive for some bigger values of r, especially for $1.2 < r < 2$. The correct function to plot is

$$\frac{d}{dr}\left(\frac{1}{r^6}-\frac{1}{r^{12}}\right).$$

Use **ode45** to compute the trajectories for objects starting at $y=-5$ and at $x=0, 0.5, 1.0, 1.5, 2.0$, in each case with initial velocity $(0, 1)$. You should observe what physicists call *scattering* in various directions. How can you explain the results?

Problem 4.6. Celestial mechanics is the study of the motion of celestial objects under the influence of gravity. The study of the motion of two objects under Newtonian gravitation is known as the *two-body problem* in celestial mechanics. The two-body problem can be solved completely. The *three-body problem*, on the other hand, cannot be solved completely and sometimes exhibits strange behavior. We will investigate some simple three-body systems in this and the next problem.

Assume we have a "solar system" with a massive fixed Sun at the origin, containing a planet and a comet that interact with each other as well as with the Sun. For simplicity, we assume that the mass of the planet is negligible with respect to the mass of the sun, and the mass of the comet is negligible with respect to that of the planet. Thus, the planet exerts a force on the comet but not vice versa.

(a) Represent the coordinates of the planet by $(x_1(t), y_1(t))$ and those of the comet by $(x_2(t), y_2(t))$. As in Problem 4.3(c), use **ode45** to solve numerically the equations of motion

$$\ddot{x}_j(t)=-\frac{x_j(t)}{(x_j(t)^2+y_j(t)^2)^{3/2}},\quad \ddot{y}_j(t)=-\frac{y_j(t)}{(x_j(t)^2+y_j(t)^2)^{3/2}},\quad j=1,2,$$

with the initial conditions $x_1(0)=1, x_2(0)=2, \dot{x}_1(0)=0, \dot{x}_2(0)=0, y_1(0)=0, y_2(0)=0, \dot{y}_1(0)=1, \dot{y}_2(0)=0.1$, for $0\le t\le 15$. These equations describe the motion when the planet and the comet move independently (i.e., without interacting), in the Sun's central force field. For this problem, you should decrease the error tolerances in the calculation so that the trajectories are reasonably accurate. To do this, add the line

```
>>  opts = odeset('RelTol',1e-5,'AbsTol',1e-7);
```

to your code *before* calling **ode45** and then add **opts** as the last argument of **ode45** (after the required arguments). Plot the resulting trajectories (as in Problem 4.3(c)) on a single set of axes. You should find that the planet moves in a circular orbit, and that the comet moves in an elongated elliptical orbit that comes very close to the Sun. By adjusting **axis**, graph the comet's orbit near the Sun. Does it crash?

(b) Now add the effect of the planet's gravitation on the comet and redo the calculation (for t running from 0 to 50) and plot as in part (a). Assume for simplicity that the mass of the planet is one-tenth that of the Sun, so that the equation for $\mathbf{r}_2=(x_2, y_2)$ now becomes

$$\ddot{\mathbf{r}}_2 = -\frac{\mathbf{r}_2}{\|\mathbf{r}_2\|^3} - 0.1\frac{\mathbf{r}_2 - \mathbf{r}_1}{\|\mathbf{r}_2 - \mathbf{r}_1\|^3}.$$

Explain why this is the correct equation of motion. What is the effect of the planet on the motion of the comet?

Problem 4.7. In this problem, we will investigate another three-body problem, that of a planet moving around a *double* star.

(a) Suppose that two suns of equal mass M revolve in a circle under the influence of each other's gravity. (This is a reasonable classical model for a double star.) Let $\mathbf{r}_j = (x_j, y_j)$, $j = 1, 2$, denote the position of the jth star. If the stars revolve opposite each other in a circle of radius R at fixed angular velocity ω, then

$$\mathbf{r}_1(t) = (R\cos\omega t, R\sin\omega t), \quad \mathbf{r}_2(t) = -\mathbf{r}_1(t) = (-R\cos\omega t, -R\sin\omega t).$$

Since the distance between the stars is $2R$, the equations of motion for Newtonian gravitation are then given by

$$M\ddot{\mathbf{r}}_j = -GM^2\frac{\mathbf{r}_j - (-\mathbf{r}_j)}{(2R)^3},$$

or

$$\ddot{\mathbf{r}}_j = -GM\frac{\mathbf{r}_j}{4R^3}.$$

Differentiate $\mathbf{r}_1$ twice and find the relationship between M, R, and ω.

(b) Suppose that we have normalized things so that $R = 0.1$ and $\omega = 1$. What does this tell you about GM? What is the equation of motion for a planet moving about this double star?

(c) From the point of view of a planet far away from the double star, things should not be that different from the case of a planet moving around a single star of mass $2M$. As in Problem 4.3(c), solve and plot the trajectory for a planet moving around a single star of mass $2M$ in a circular orbit in the x-y plane, centered at the origin and with radius 1. Then, solve and plot the trajectory for the same planet moving with the same initial conditions around the double star in (b) above. The latter looks circular, but the trajectory is a little ragged. That is because there are *wobbles* in the orbit. Plot separately (using **`plot`**) the differences between the x- and y-coordinates for the two trajectories to see the magnitude of the wobble.

Glossary of MATLAB Commands

`assume` Set assumptions on a symbolic object

`axis` Selects the ranges of x and y to show in a plot

cross The cross product of two vectors

diff Differentiates an expression

dsolve Symbolic differential equation solver

double Converts the (possibly symbolic) expression for a number to a numerical (double-precision) value

fplot Easy function plotter

fplot3 Function or symbolic plotter for curves in 3-space

legend Attaches a legend to a combination of plots

loglog Log–log plotter

logspace Generates a logarithmically spaced vector

ode45 Numerical differential equation solver

odeset Sets options for **ode45**

plot 2D numerical plotter

plot3 Numerical plotter for curves in 3-space

simplify Used to simplify a symbolic expression. Note that sometimes you will need to increase the option **Steps** for best results.

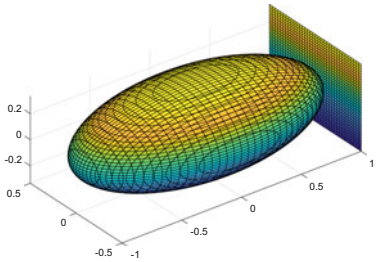
solve Symbolic equation solver

subs Substitute for a variable in an expression

Reference

1. I. Newton, *The Principia*, Great Minds Series (Prometheus Books, New York, 1995). transl. by Andrew Motte

Chapter 5
Directional Derivatives



In the last two chapters, we studied curves in space, that is, *vector-valued functions of a single scalar variable*. We move on now to functions of two variables, or equivalently, *scalar-valued functions of a single vector variable* $\mathbf{x} = (x, y)$. Such a function f is a rule, usually given by a formula, that associates a value $f(\mathbf{x}) = f(x, y)$ to any $\mathbf{x} = (x, y)$ in a subset of $\mathbb{R}^2$ called the *domain* of the function. We are interested primarily in the case where f is continuous and differentiable. (Sometimes this may involve cutting down the domain. For example, if

$$f(x, y) = \sqrt{1 - x^2 - y^2},$$

then the natural domain where f is defined is $\{(x, y) : x^2 + y^2 \leq 1\}$, but it is convenient to restrict to the smaller domain $\{(x, y) : x^2 + y^2 < 1\}$. Otherwise, f is not differentiable at the "boundary points" where $x^2 + y^2 = 1$.) Our goals are to learn how to visualize the function, how to differentiate it, and how to use the visualizations and derivatives to understand algebraic and geometric properties.

5.1 Visualizing Functions of Two Variables

There are two standard ways to visualize a function f of two variables: looking at its *graph*, that is, the set of points

$$\{(x, y, z) : z = f(x, y)\}$$

in $\mathbb{R}^3$; and looking at its *level curves*, which are the curves in $\mathbb{R}^2$ given implicitly by the equations $f(x, y) = c$ with c a constant. One of the main points of this chapter is to discuss what information about f is encoded in the graph and in the level curves. While sketching either the graph or the level curves of f is often hard to do by hand, both are easy to display with MATLAB.

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_5

5.1.1 Three-Dimensional Graphs

fsurf and **fmesh** are the basic MATLAB commands for graphing a function of two variables. We shall illustrate primarily with the former. Let us start with a simple example, the function

$$f(x, y) = y\left(1 - \frac{1}{(x^2 + y^2)}\right), \quad (x,\ y) \neq (0,\ 0).$$

It has an interesting connection with fluid flow, which we will discuss later. We encode the function as a MATLAB function handle and then plot it.

```
>> flowfunction = @(x, y) y*(1 - 1/(x^2 + y^2));
>> figure; fsurf(flowfunction, [-2 2 -2 2])
```

MATLAB spews forth a few warning messages because the square on which the graph is plotted includes the origin—where the function is not defined. But it still produces a reasonable graph (Figure 5.1).

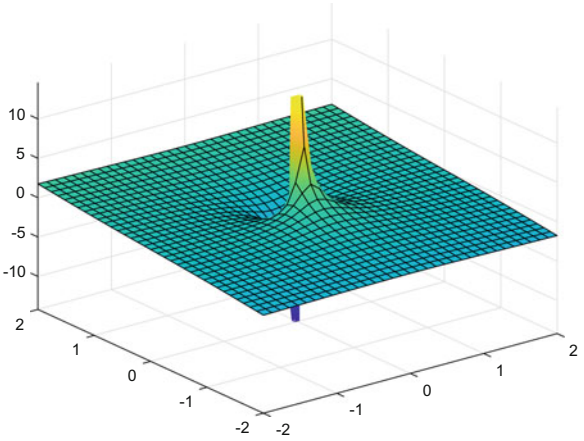


Fig. 5.1 Fluid Flow Surface—First Drawing

Actually, this is not the greatest picture, but it does tell us some things about the function f. First of all, the graph is highest where f is the largest, has peaks where f has values that are higher than in the surrounding regions, and has troughs where f has values that are lower than in the surrounding regions. We can see a small region near the origin along the negative y-axis where f gets large, and a small region near the origin along the positive y-axis where f gets to be quite negative. (Recall from Chapter 2 that MATLAB's default view in three-dimensional graphics is $(-5.3, -6.8, 5)$, which results in the y-axis running from lower right to upper left.) Far away from the origin, the graph resembles a plane which is tilting upward

in the direction of the positive y-axis. We can see these features better by cutting down the domain, moving the view, and labeling the axes (Figure 5.2).

```
>> figure; fsurf(flowfunction, [-1.5 1.5 -1.5 1.5])
>> axis([-1.5 1.5 -1.5 1.5 -5 5])
>> view([2 1 2])
>> xlabel('x'); ylabel('y'); zlabel('z')
```

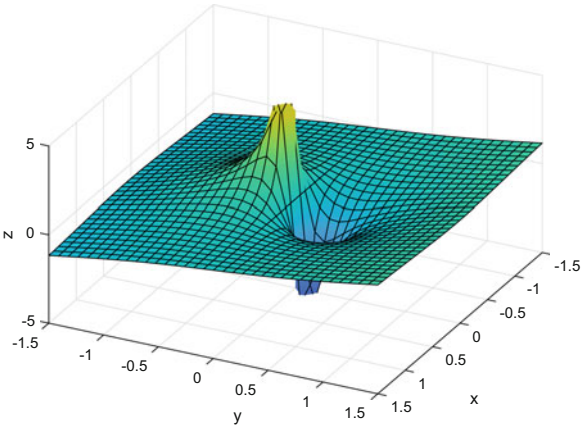


Fig. 5.2 Fluid Flow Surface—Second Drawing

5.1.2 Graphing Level Curves

The second way to visualize a function of two variables is by means of a two-dimensional plot of its *level curves* or *contours*: that is, the curves along which the function takes a constant value. In MATLAB, such a plot is produced with the **`fcontour`** command. Here is an important example to keep in mind. If $g(x, y)$ denotes the elevation of a point (x, y) on a small region of the Earth, then the graph of g is the shape of the Earth's surface, and a plot of the level curves of g is a topographic map (with contour lines). You will often find topographic maps with different elevation ranges shaded in different colors. As we shall see, MATLAB can also be instructed to shade the regions between level curves to indicate different ranges of values of the function. By default, MATLAB presents different level curves in different colors. Now let us try our **`flowfunction`** example.

```
>> list1 = -2:0.5:2; figure; fcontour(flowfunction, [-2 2 -2 2],...
     'LevelList',list1);
```

Once again, MATLAB issues some warning messages because of the singularity at the origin—but again, it does an excellent job of rendering the graphics. Note that by default, MATLAB selects the contours to present. However, we overrode (Figure 5.3) that via the list of values we supplied—that is, we selected the level curves to

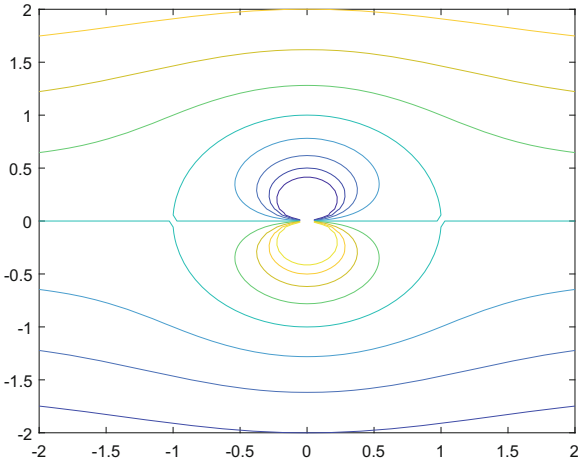


Fig. 5.3 Fluid Flow Contours—First Drawing

display. Next, let us go back and shade the regions between the level curves we have selected.

```
>> figure; fcontour(flowfunction, [-2 2 -2 2],...
     'LevelList',list1, 'Fill', 'on')
```

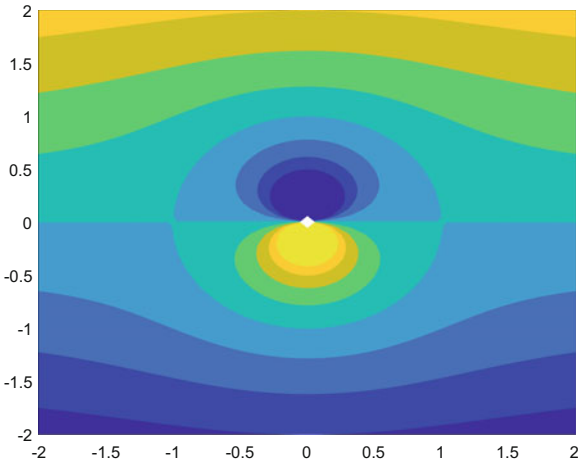


Fig. 5.4 Fluid Flow Contours—Second Drawing

The second picture (Figure 5.4) tells us a bit more about the function f. For example, the lightest regions of the plot correspond to the largest positive values of f, and the darkest to the smallest values (i.e., most negative). So, if we were to draw

a contour plot of $-f$, the level curves would be the same, but the shading would be reversed, with the black and white regions interchanged. We can see from looking carefully at the shading pattern that for a fixed level $c \neq 0$, the equation $f(x, y) = c$ describes a level curve with two components, both of which cross the y-axis. One component is roughly horizontal, except for a somewhat rounder portion in the middle, and extends infinitely to the right and to the left. The other component is a small loop located near the origin, above the origin if c is negative, and below the origin if c is positive. We can verify this observation by having MATLAB do a computation:

```
>> syms y c; eqn = (flowfunction(0, y) == c), solve(eqn, y)

eqn =
-y*(1/y^2 - 1) == c

ans =
 c/2 - (c^2 + 4)^(1/2)/2
 c/2 + (c^2 + 4)^(1/2)/2
```

This shows that each contour crosses the y-axis at two distinct places, with y-coordinates differing by $\sqrt{4 + c^2} \geq 2$.

The picture above is especially nice in color, which you cannot see here, since we have printed this book in black and white. You can see the color on your computer screen if you reevaluate the Chapter 5 script, which you can find on the accompanying web site for the book. Two more points worth making: (i) The function is changing most rapidly where the contours are closest together; and (ii) although the "small loop" components of the contours appear to pass through the origin, they really do not, since the function f is undefined at this point.

Now, we can explain the motivation for our example and the reason for the name **`flowfunction`**. If we restrict to the domain $x^2 + y^2 > 1$, then the level curves of our function are *streamlines*, that is, the trajectories along which objects would drift with the current, for an ideal fluid flowing around a circular obstacle in the shape of the disk $\{x^2 + y^2 \leq 1\}$.

This raises the issue of how to use **`fcontour`** to visualize a function whose domain does not fill up the whole rectangle enclosing it. Either we can redefine the function at the missing points, or else we can "block out" the portion of the region that is not supposed to be part of the domain. We can do the latter for our example by blotting out the unit disk via a MATLAB command that draws (in black) a point at the origin, blown up suitably. Here is some MATLAB code that will do the trick

```
>> hold on; plot(0, 0, '.k', 'MarkerSize',500); axis equal
```

This is a nice picture (Figure 5.5) of streamlines for flow around an obstacle (shown in black).

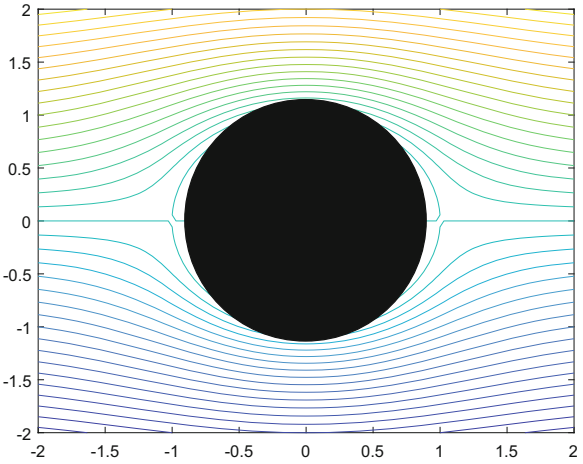


Fig. 5.5 Fluid Flow Contours—Third Drawing

5.2 The Gradient of a Function of Two Variables

So far, we have discussed the level curves of a function, that is, the curves along which the function is not changing at all. Now we want to discuss how functions change from point to point. As in one-variable differential calculus, this will be measured by derivatives.

5.2.1 Partial Derivatives and the Gradient

There are several types of derivatives for a function f of two variables. It is natural to start with the easiest ones to compute: the *partial derivatives*. These are the derivatives of f viewed as a function of one variable while the other variable is held fixed. The partial derivatives of f with respect to x and y are denoted by f_x and f_y, or by $f^{(1,0)}$ and $f^{(0,1)}$, or by the notation $\partial f/\partial x$ and $\partial f/\partial y$. (Note the use of a curly ∂ in place of the ordinary d of one-variable calculus.) In MATLAB, the partial derivatives are computed using the **`diff`** command. The second argument of **`diff`** is the variable with respect to which one differentiates. For instance:

```
>> syms x y; diff(x^3 + sin(x*y), x)

ans=
y*cos(x*y) + 3*x^2
```

It is particularly convenient to assemble the two partial derivatives into a single vector, called the *gradient* of f, and denoted with the special symbol ∇f. Thus

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j}. \tag{5.1}$$

As we will see shortly, *the gradient of a function at any point is perpendicular to the level curve through that point.* In a sense, this is not surprising, for a function is constant along level curves, and the gradient is concocted from derivatives, which measure change in a function. (Keep in mind the motto: "Change is orthogonal to constancy.")

MATLAB has two commands that we will find useful in the study of gradients, **jacobian** and **quiver**. (Actually, there is a third, **gradient**, that we only use later on, but which the reader is encouraged to investigate now in the online help.) Let us illustrate the first by applying it to the **flowfunction** function.

```
>> gradflowfunction = jacobian(flowfunction(x, y))

 gradflowfunction =
[ (2*x*y)/(x^2 + y^2)^2, (2*y^2)/(x^2 + y^2)^2 - 1/(x^2 + y^2) + 1]
```

The above command computes the gradient of the **flowfunction** as a two-dimensional vector, in other words, as a two-element list of functions—the partial derivatives of the **flowfunction** with respect to x and y. In fact, if we were working with a function of three variables instead, the **jacobian** command would compute the vector of the three partial derivatives. More generally, **jacobian** computes the Jacobian matrix of a vector-valued function (of several variables), but we will not have need of that until Chapter 8.

Next, we turn our attention to drawing the vector field of the gradient of a scalar function. This is where the **quiver** command is indispensable. That command can be used to draw more general vector fields (as we will do in Chapter 10). Here, the **quiver** command, paired with the **meshgrid** command, will be used to produce a graphical depiction of the gradient (of a scalar function, like **flowfunction**), with arrows showing the direction and magnitude of the gradient at various points. Here is our main example. We will give the input and then offer some explanation. The picture produced is Figure 5.6.

```
>> [X, Y] = meshgrid(-2:0.2:2, -2:0.2:2);
>> U = (2.*X.*Y)./((X.^2 + Y.^2).^2);
>> V = (2.*Y.^2)./((X.^2 + Y.^2).^2) - 1./(X.^2 + Y.^2) + 1;
>> L = sqrt(U.^2 + V.^2);
>> figure; gradientfield = quiver(X, Y, U./L, V./L, 0.5)
>> axis equal tight
```

Note that **quiver** is a numerical command, not a symbolic command. Thus, it requires the numerical grid set up by the **meshgrid** command as a precursor. (Any use of **meshgrid** should be followed by a semicolon, to avoid spewing forth a long list of coordinates.) **quiver** draws the vector field by placing a representative set of vectors—corresponding to the gradient of the scalar function—at various spots, in the direction indicated and with length proportional to the length of the gradient vector. This can make for an unwieldy diagram as the varying length vectors sometimes can be too short to be legible and at other times tend to have their tails and heads obscure nearby vectors. It is better to have all the vectors depicted be of the same length. That is the reason for the normalizing factor **L**. Note also that **U** and **V** are precisely the x- and y-coordinates computed by **jacobian**. But the formulas for **U**, **V** and **L** have

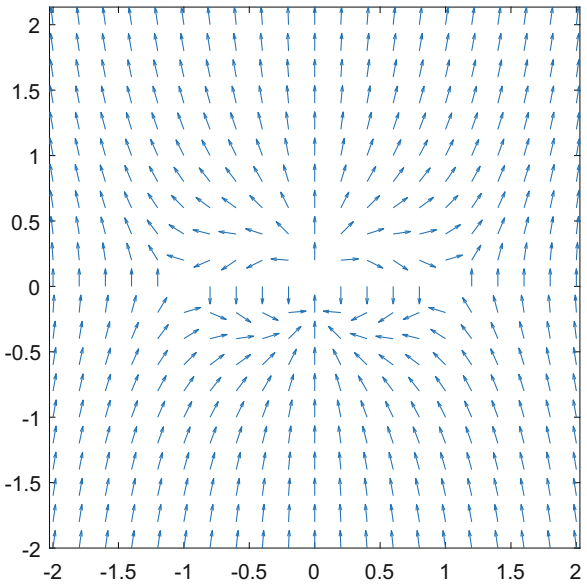


Fig. 5.6 Gradient Field of Flowfunction

to be *vectorized* since **X** and **Y** are arrays. Finally, the last parameter, namely 0.5, in the **quiver** command tells MATLAB to cut all the vectors down to half their size.

Finally, Figure 5.7 is a picture combining the gradient field and the contour plot. As you can see, the arrows representing the gradient are perpendicular to the level curves. This depends on our use of **axis equal tight** so that the arrows are not distorted in shape.

5.2.2 Directional Derivatives

The importance of the gradient comes from the formula for *directional derivatives*. Suppose f is a function of two variables and $t \mapsto \mathbf{r}(t) = (x(t),\ y(t))$ parametrizes a curve in the domain of f through a certain point, say $(x_0,\ y_0) = \mathbf{r}(0)$. Then we can form the composite function

$$g(t) = f \circ \mathbf{r}(t) = f(x(t),\ y(t)).$$

The chain rule for derivatives says that

$$g'(0) = \frac{\partial f}{\partial x}(x_0,\ y_0) \cdot x'(0) + \frac{\partial f}{\partial y}(x_0,\ y_0) \cdot y'(0) = \nabla f(x_0,\ y_0) \cdot \mathbf{r}'(0). \qquad (5.2)$$

Why are there two terms, and not just one? Well, a change in t affects the value of g in two different ways: once by changing $x(t)$ (at the rate $x'(t)$), which then affects f at the rate $\partial f/\partial x$; and once by changing $y(t)$ (at the rate $y'(t)$), which then affects f at the rate $\partial f/\partial y$. But note the vector form of the equation above: the rate of change of g is just the dot product of the gradient of f with the velocity vector of the curve. This suggests defining the *directional derivative* of f (at $(x_0,\ y_0)$) in the direction $\mathbf{u} = (u_0,\ u_1)$ to be

$$D_{\mathbf{u}} f(x_0,\ y_0) = \nabla f(x_0,\ y_0) \cdot \mathbf{u} = \frac{d}{dt}\Big|_{t=0} f\ (x_0 + tu_0,\ y_0 + tu_1). \tag{5.3}$$

(Although some books require $\mathbf{u}$ to be a unit vector here, there is no reason to make this restriction.) Then formula (5.2) becomes

$$\frac{d}{dt}\Big|_{t=0} f(\mathbf{r}(t)) = D_{\mathbf{r}'(0)} f(\mathbf{r}(0)). \tag{5.4}$$

Suppose the curve parametrized by $\mathbf{r}(t)$ is a level curve of f. By definition of the term "level curve", $f(\mathbf{r}(t))$ is a constant, so the left-hand side of equation (5.4) is 0. This tells us that $\nabla f(x_0,\ y_0) \cdot \mathbf{r}'(0) = 0$, that is, the gradient of f at $(x_0,\ y_0)$ is perpendicular to a tangent vector to the level curve through that point, and thus to the level curve itself. This explains the picture depicted in Figure 5.7.

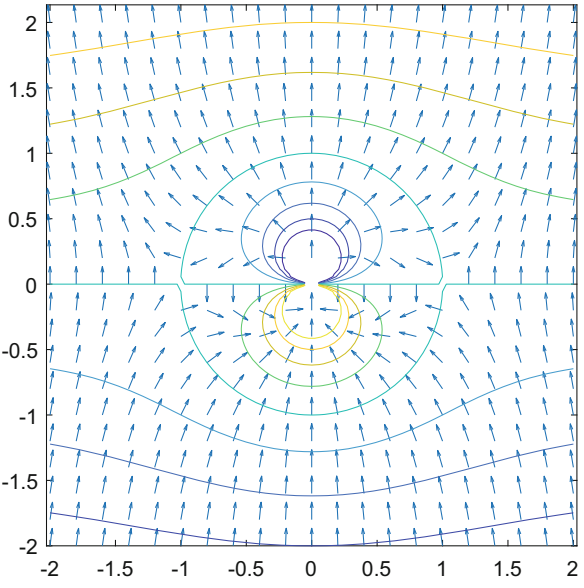


Fig. 5.7 Gradient Field and Contour Plot of Flowfunction

The directional derivative formula has other consequences. For instance, if we fix x_0 and y_0, and maximize $D_{\mathbf{u}} f(x_0,\ y_0)$ over all unit vectors $\mathbf{u}$, then the maximum value is $\|\nabla f(x_0,\ y_0)\|$, and is achieved when $\mathbf{u}$ and ∇f point in the same direction. This fact is often stated informally as follows: *the function f increases the fastest in the direction of the gradient vector.* We will use this fact in Chapter 7, *Optimization in Several Variables*.

We can see a final consequence of the directional derivative formula if we look at a *critical point* of f, that is, at a point $(x_0,\ y_0)$ where $\nabla f(x_0,\ y_0) = (0,\ 0)$. At such a point, all directional derivatives of f are 0, i.e., the rate of change of f is 0 in all directions. The fact that the gradient is perpendicular to the level curve through such a point gives no information; indeed, the level curve through a critical point may not have a well-defined tangent direction at this point. We can see this in the example above. From **`gradflowfunction`**, we see that ∇f vanishes at $(\pm 1,\ 0)$, where the function f takes the value 0. Also, the level curve $f = 0$ is quite unusual in shape.

```
>> list3=0; figure; fcontour(flowfunction, [-2 2 -2 2],...
       'LevelList',list3,'MeshDensity',200); axis equal
```

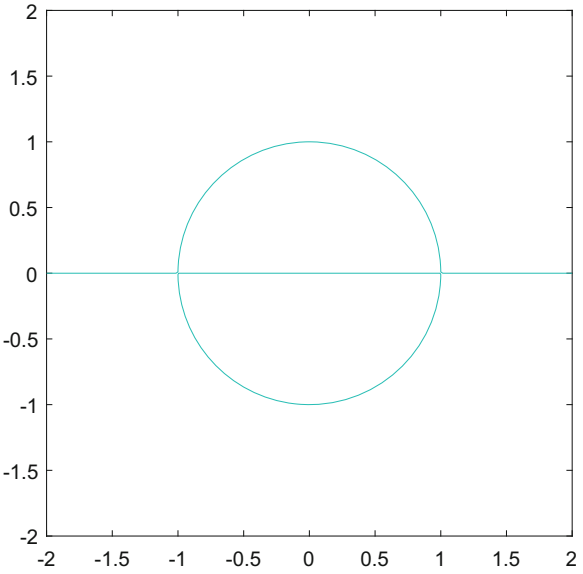


Fig. 5.8 The level curve **`flowfunction`** = 0

We see (as we can also check from the original formula for f) that the curve $f = 0$ is the union of the unit circle and the x-axis (with the origin deleted), and that the two pieces of the level curve cross each other at the critical points of f (Figure 5.8). We will see a more detailed explanation for the shape of this level curve in Chapter 7.

5.3 Functions of Three or More Variables

The theory of gradients and directional derivatives generalizes without difficulty to functions of three of more variables. Visualizing such functions, however, is more difficult. A function of three variables f is a rule that associates a value $f(\mathbf{x}) = f(x, y, z)$ to any $\mathbf{x} = (x, y, z)$ in a subset of $\mathbb{R}^3$ called the *domain* of the function. We will again mostly be interested in the case where f is *smooth*, i.e., has continuous partial derivatives. The set where such a function takes a fixed value c will now usually be a surface, called a *level surface* of the function. Alas, the MATLAB command **`fsurf3`** does not exist. In fact, until version R2016b, MATLAB did not have a symbolic command to plot level surfaces. (Well, that is not quite true. MuPAD did. But in accordance with the paradigm we have set, we will not make use of MuPAD in this book.) However, one of the authors (JR) wrote a (numerical) routine for plotting level surfaces some years ago. It is called **`implicitplot3d`**. (Not completely coincidentally, the symbolic command in MuPAD is called **`plot::implicit3d`**.) If you wish to use **`implicitplot3d`**, you can find the code for the corresponding function script on the companion web site. However, since as of the writing of this book, a symbolic code does now exist in MATLAB, we shall use it. It is called **`fimplicit3`**.

Now let us consider a famous example of a level surface, the *hyperboloid of two sheets*:

$$z^2 - (x^2 + y^2) = 1.$$

This is clearly the level surface of the function $F(x, y, z) = z^2 - (x^2 + y^2)$ determined by the "slice" $F(x, y, z) = 1$. That the surface has two sheets is clear if we solve for z:

$$z = \pm\sqrt{1 + x^2 + y^2}.$$

Here is the code to generate the surface; the resulting graph is Figure 5.9.

```
>> syms x y z; h = z^2 - x^2 - y^2;
>> fimplicit3(h - 1, [-1.1 1.1 -1.1, 1.1 -2 2]); axis equal
```

The '-1' appended to the 'h' is there because **`fimplicit3`** by default plots first argument = 0; the following six numbers are the bounds on x, y, and z.

Other level surfaces of the same function $z^2 - x^2 - y^2$ include the *double cone* $z^2 = x^2 + y^2$ and the *hyperboloid of one sheet* $z^2 - x^2 - y^2 = c$ with $c < 0$. We leave it to the reader to use **`fimplicit3`** to display them. Actually, we will also draw them using **`fsurf`** applied to parametric equations in the next chapter on geometry of surfaces. But let us note the important fact that the double cone is not smooth; it has a "singular point" at the origin. We will explain shortly why it is that a level surface of a smooth function can fail to be smooth at certain points.

The gradient of a function f of three variables is defined, in analogy with (5.1), by

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) = \frac{\partial f}{\partial x}\mathbf{i} + \frac{\partial f}{\partial y}\mathbf{j} + \frac{\partial f}{\partial z}\mathbf{k}.$$

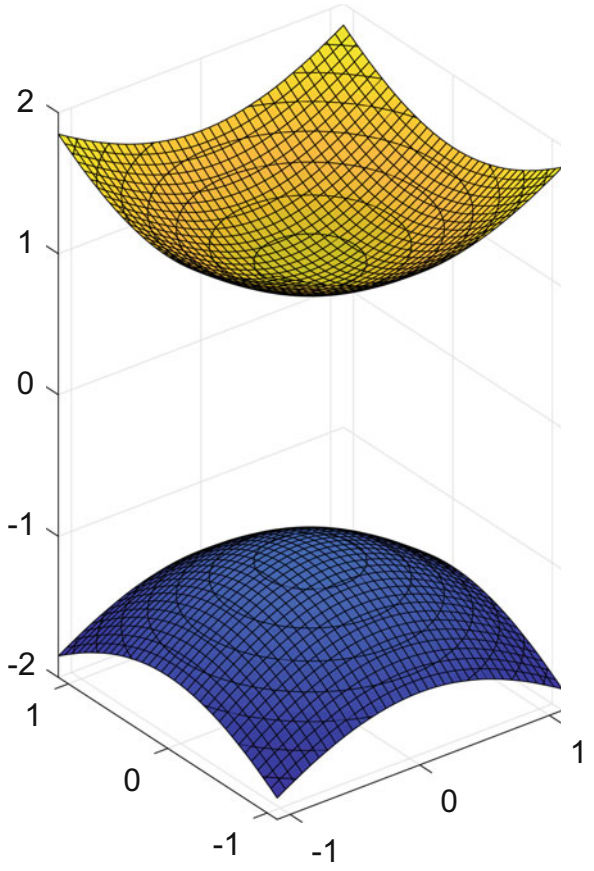


Fig. 5.9 Hyperboloid of Two Sheets

The directional derivative of f in the direction $\mathbf{u} = (u_0,\ u_1,\ u_2)$ is defined by the analog of equation (5.3):

$$D_{\mathbf{u}} f(x_0,\ y_0,\ z_0) = \nabla f(x_0,\ y_0,\ z_0) \cdot \mathbf{u} = \frac{d}{dt}\Big|_{t=0} f\ (x_0 + tu_0,\ y_0 + tu_1,\ z_0 + tu_2)\,.$$

If $t \mapsto \mathbf{r}(t)$ parametrizes a curve in $\mathbb{R}^3$, equation (5.4) above still applies. So, if the curve $\mathbf{r}(t)$ stays in a level surface $f(x, y, z) = c$, we conclude as before that for any point $(x_0,\ y_0,\ z_0)$ in the surface, $\nabla f(x_0,\ y_0,\ z_0)$ is again perpendicular to the curve. This being true for *any* curve in the level surface, we see that $\nabla f(x_0,\ y_0,\ z_0)$ is perpendicular to the level surface through $(x_0,\ y_0,\ z_0)$. We can confirm this visually using the MATLAB command **`quiver3`**, to visualize the gradient vectors. Note the **`hold on`**, which causes the vector field to be superimposed on the hyperboloid

that we just drew. Also, the MATLAB command **surfnorm** computes the normal vectors to a mesh surface. Finally, the resulting figure appears in Figure 5.10.

```
>> [X,Y] = meshgrid(-1:0.5:1.,-1:0.5:1.);
>> Z = sqrt(1 + X.^2 + Y.^2);
>> [U,V,W] = surfnorm(X,Y,Z);
>> hold on; quiver3(X,Y,Z,U,V,W,2)
>> Z1 = - sqrt(1 + X.^2 + Y.^2);
>> [U,V,W] = surfnorm(X,Y,Z1);
>> quiver3(X,Y,Z1,U,V,W,2)
```

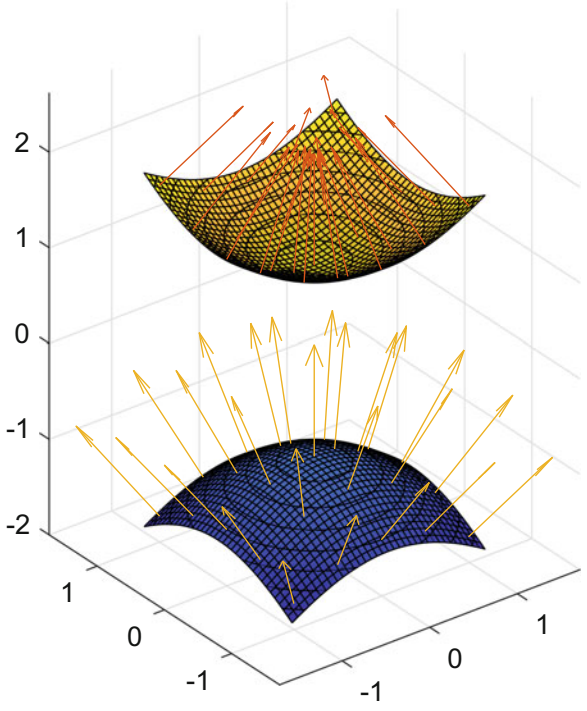


Fig. 5.10 Hyperboloid of Two Sheets with Vector Field

Now we can see why it is that the level surface $z^2 - x^2 - y^2 = 0$ (the double cone) has a singular point at the origin. The gradient of $z^2 - x^2 - y^2$ is $(-2x, -2y, 2z)$, which degenerates to $(0, 0, 0)$ at the origin. Thus, we cannot use the gradient to find a unit vector perpendicular to the cone at the origin, and in fact the cone has *no tangent plane* at this point. On the other hand, if f is a smooth function and if $\nabla f(x_0, y_0, z_0) \neq (0, 0, 0)$, then the level surface $f(x, y, z) = f(x_0, y_0, z_0)$ has a well-defined tangent plane at (x_0, y_0, z_0), namely the unique plane through this

point which is normal to the vector $\nabla f(x_0,\ y_0,\ z_0)$. So a level surface of a smooth function of three variables is *nonsingular* provided it does not pass through any *critical point* of the function (a point where the gradient vector vanishes). This is a higher dimensional analog of the fact that a *level curve* of a smooth function of *two* variables is smooth (without "crossing points") if it does not pass through a critical point.

Problem Set E. Directional Derivatives

Problem 5.1. This problem examines in detail the level curves of the function

$$f(x,\ y) = (x^2 + y^2)^2 - x^2 + y^2.$$

(a) Graph f in the region $-2 \le x \le 2$, $-2 \le y \le 2$. Draw a contour plot in the same domain.

(b) To focus on the level curves in greater detail, draw three plots, each with a different choice in **`LineColor`** in order to vary the color of the contours. Draw the first with **`LevelList`** set to **`[-0.2, -0.1]`**, one with **`LevelList`** set to **`0`** and the third with **`LevelList`** set to **`[0.1, 0.2, 1, 4, 10]`**. Then assemble all the contour plots together. You should see three distinct shapes of level curves: one shape for negative values of f, one shape for positive values of f, and one special shape for $f = 0$. Can you speculate what is responsible for the differences? (If not, just state what the three shapes are and come back to this question after you've done the next parts of the problem.)

(c) Compute the gradient of f, and determine where it vanishes. (Hint: Apply **`solve`** to the equation that sets the gradient equal to the zero vector. You need to use **`[0, 0]`** for 'zero' if you think of the gradient as a two-element list.) What is special about the level curves through the critical points?

(d) Use **`meshgrid`** and **`quiver`** to draw the gradient field of f, and superimpose the gradient plot on your contour plot. What do you observe about the gradient vectors along the three different kinds of level curves? What happens at the origin, and how does this help explain the special shape of the level curve $f = 0$? Complete your answer to (b).

Problem 5.2. Some pairs of functions of two variables have the property that the level curves of one function are orthogonal to the level curves of the other. This problem investigates that phenomenon.

(a) Suppose that u and v are smooth functions of x and y and that

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

Verify that $\nabla u \cdot \nabla v = 0$ and that u and v are both *harmonic functions*; that is, they each satisfy *Laplace's equation*:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = 0.$$

(You will need the fact, found in all multivariable calculus texts, that for a smooth function, say u, its mixed partials are equal, i.e., $\dfrac{\partial^2 u}{\partial x \partial y} = \dfrac{\partial^2 u}{\partial y \partial x}$.) Under these circumstances, u and v are said to be *harmonic conjugates* of one another.

(b) Verify that the following pairs of functions are harmonic conjugates of one another. In each case, plot the level curves, using a different **`LineColor`** setting for each function, and then superimpose the two contour plots to give a picture of the orthogonal families of level curves. (You may need to experiment with the domain of the plot until you get a good picture. You may also need to make use of **`LevelList`** to adjust the number of contours revealed. Finally, use a different **`LineColor`** for each family of curves to distinguish the two families.)

(i) $u(x,\ y) = x,\ v(x,\ y) = y$ (the simplest case).

(ii) $u(x,\ y) = x^2 - y^2,\ v(x,\ y) = 2xy$ (hyperbolas).

(iii) $u(x,\ y) = \sin x \cosh y,\ v(x,\ y) = \cos x \sinh y$.

(iv) $u(x,\ y) = x\left(1 + \dfrac{1}{x^2 + y^2}\right),\ v(x,\ y) = y\left(1 - \dfrac{1}{x^2 + y^2}\right)$.

(v) $u(x,\ y) = \dfrac{1}{2}\ln\left(x^2 + y^2\right),\ v(x,\ y) = \arctan\left(\dfrac{y}{x}\right),\ x > 0$.

Problem 5.3. Consider a conical mountain, rising from a flat plain, in the shape of the graph of the height function $h(x,\ y) = \max(3(1 - \sqrt{x^2 + y^2}),\ 0)$. (The "max" is put in to cut the function off at 0, since away from the mountain the height should be everywhere 0.)

(a) Draw a picture of the mountain using **`fsurf`**, and verify that it has an appropriate "mountain-like" appearance.

(b) Now suppose that a builder wants to build a road up the mountain to the top. The road may be idealized as a curve lying on the mountain, so it has parametrization

$$(\mathbf{r}(t),\ z(t)), \qquad z(t) = h(\mathbf{r}(t)),$$

where $\mathbf{r}(t) = \big(x(t),\ y(t)\big)$ is the parametric equation of the projection of the road into the x-y plane. It is not practical for the road to go straight up the side of the mountain, since it is too steep. Therefore, the functions $x(t)$ and $y(t)$ should be chosen so that the directional derivative of h in the direction of a unit vector pointing along the road, which represents the rate of climb of the road, is a fixed reasonable constant, say $\frac{1}{20}$. (This constant corresponds to a 5% grade, quite a steep road by normal standards.) This condition amounts to a *differential equation* for $x(t)$ and $y(t)$. We may as well take the curve $\mathbf{r}(t)$ to be parametrized by arclength, so that $x'(t)^2 + y'(t)^2 = 1$. Show then that the 5% grade condition is satisfied by the spiral

$$x(t) = at\cos(b\ln t), \quad y(t) = at\sin(b\ln t), \quad z(t) = 3(1-at), \text{ for } 0 \le t \le \frac{1}{a},$$

when $a = \frac{1}{60}$, $b = \sqrt{3599}$.

(c) Use **fplot3** to plot the spiral. You may find that the options **LineWidth** and **Color** come in handy for delineating the curve clearly, especially when you combine it with the graphic of the mountain. In fact, use **hold on** to do exactly that.

Problem 5.4. In this problem, we further explore the notions of directional derivative and of derivative along a curve. Consider the function of two variables

$$f(x,\ y) = x^3 - 2xy^2 + xy - y^2.$$

(a) Begin by applying **fcontour** and **fsurf** to the function in the square domain $\{-3 \le x \le 3,\ -3 \le y \le 3\}$. (Regarding the former, you may wish to adjust the number of contours by using **LevelList**.) Next, compute the gradient of f and locate the critical points. (See Problem 5.1 for a method to do this.) After throwing away the complex critical points, and perhaps using **solve** and **double** to pinpoint the real critical points, plot the critical points with **plot** and **MarkerSize** (so that the dots are big enough to be easily visible) and superimpose them on your contour plot.

(b) Next, let us restrict f to lines in the x-y plane, and investigate the curves in the graph of f lying over these lines. Interesting cases are the lines $2x - 3y = 0$ and $3x - 5y + 9 = 0$. Superimpose these lines on your contour plot (using **fplot**) so you can see roughly how f behaves along them. Also use **fplot3** to draw the curves in the graph of f lying over these lines, and superimpose these on your surface plot. (See part (c) in the previous problem for a method to do this effectively.)

(c) Consider the functions $g(x) = f(x, 2x/3)$ and $h(x) = f(x, (3x + 9)/5)$. These are the restrictions of f to the lines in part (b). Plot g and h and analyze their behavior. Then check that

$$g'(x) = (D_{(1,2/3)}f)(x,\ 2x/3), \qquad h'(x) = (D_{(1,3/5)}f)(x,\ (3x+9)/5),$$

and explain why this should be the case from the directional derivative formula (formula (5.3) in Chapter 5).

(d) Finally, let us study what happens when we lift one of the level curves of f to the graph of f. By definition, f is constant along a level curve, so the lift of the level curve to the graph should be a curve of constant height. One way to visualize such a curve is to intersect the surface with a horizontal plane, by displaying the three-dimensional graph of f and the plane simultaneously. Do this for the horizontal planes $z = 0$ and $z = 10$, and check that the shape of the

intersections agrees with what you get from contour plots with **`LevelList`** set to **`0`** and **`10`**, respectively. The level curve $f = 0$ is special; explain how and why.

Problem 5.5. In this problem, we study the tangent plane to a surface at a point. We know that for a function f of three variables, ∇f points perpendicular to the level surfaces of f. Thus, if $(x_0,\ y_0,\ z_0)$ is not a critical point of f, then the level surface

$$f(x,\ y,\ z) = f(x_0,\ y_0,\ z_0)$$

has a well-defined tangent plane at $(x_0,\ y_0,\ z_0)$; namely, the plane through this point with $\nabla f(x_0,\ y_0,\ z_0)$ as a normal vector. For each of the following functions f, draw a picture of the level surface through the indicated point (using **`fimplicit3`**), compute the equation of the tangent plane at this point, and display the tangent plane and the level surface simultaneously to visualize the tangency. (These are examples of *quadric surfaces*, i.e., surfaces given by quadratic equations.)

(a) elliptic paraboloid: $f(x,\ y,\ z) = x^2 + 4y^2 - z$, $(x_0,\ y_0,\ z_0) = (0,\ 0,\ 0)$.

(b) hyperbolic paraboloid or saddle surface: $f(x,\ y,\ z) = x^2 - 4y^2 - z$, $(x_0,\ y_0,\ z_0) = (0,\ 0,\ 0)$.

(c) ellipsoid: $f(x,\ y,\ z) = x^2 + 4y^2 + 9z^2$, $(x_0,\ y_0,\ z_0) = (1,\ 0,\ 0)$.

(d) $f(x,\ y,\ z) = xy + xz$, $(x_0,\ y_0,\ z_0) = (1,\ 1,\ 0)$. See if you can identify the surface.

Problem 5.6. For each of the following functions f of three variables, plot the gradient field using **`meshgrid`** and **`quiver`**, and then use **`fimplicit3`** to visualize the indicated level surfaces. (You may need to experiment with the **`view`** and with the ranges of x, y, and z values to get informative pictures.) Superimpose the two plots to see that each level surface is perpendicular to the gradient field.

(a) $f(x,\ y,\ z) = x^2 + y^2 + z^2$, $f = 1$ (a sphere).

(b) $f(x,\ y,\ z) = x^3 - 3xy^2 + z$, $f = 0$ (the "monkey saddle").

(c) $f(x,\ y,\ z) = xyz$, $f = 1$.

Problem 5.7. Given a function f of two or three variables, it is interesting to study curves tangent to the gradient vector field of f. Let us make this precise. For simplicity, let us restrict to the two-variable case. A curve C with parametrization $t \mapsto \mathbf{r}(t) = (x(t),\ y(t))$ is everywhere tangent to the gradient vector field of f if for all values of t, $\mathbf{r}'(t)$ is a scalar multiple of $\nabla f(\mathbf{r}(t))$. Since we are free to reparametrize, we may as well take the scalar multiple to be 1, so we obtain the vector-valued differential equation

$$\mathbf{r}'(t) = \nabla f(\mathbf{r}(t)),$$

or the pair of coupled scalar-valued differential equations

$$x'(t) = f_x(x(t),\ y(t)),$$
$$y'(t) = f_y(x(t),\ y(t)).$$

These equations are then said to give the *gradient flow* of the function f, and C is called an *integral curve* of the flow. At each noncritical point $(x(t),\ y(t))$ on such a curve C, the unit tangent vector to C is parallel to $\nabla f(x(t),\ y(t))$, hence orthogonal to the level curve through this point. So the level curves and the curves obtained from the gradient flow form two mutually orthogonal families. (See Problem 5.2.)

For each function below, draw a contour plot of the function. Then solve the differential equations for the gradient flow through the indicated points, by using either **dsolve** or **ode45**, as appropriate. Plot the integral curves of the gradient flow in a different style from the level curves of the contour plot, and then superimpose them to view the two orthogonal families.

(a) $f(x,\ y) = x^2 - y^2$ in the region $-2 \le x \le 2$, $-2 \le y \le 2$. Plot the integral curves through the points

$$\left(\cos\left(\frac{j\pi}{12}\right),\ \sin\left(\frac{j\pi}{12}\right)\right), \qquad 0 \le j \le 24,$$

which are equally spaced around the unit circle. (In this case, **dsolve** can easily solve the differential equation with arbitrary initial conditions, so do this first and then substitute.)

(b) $f(x,\ y) = \cos x + \sin y$ in the region $-\pi \le x \le \pi$, $-\pi \le y \le \pi$. Use **dsolve** to solve the equations for the integral curves starting at a point of the form $(a,\ 0)$. You will see more than one solution, but the solutions only differ in some signs, and it is not hard to figure out what the appropriate solution is, going *forward* in time, for $-\pi < a < \pi$. Plot the integral curves through the points

$$\left(\frac{j\pi}{12},\ 0\right), \qquad -11 \le j \le 11,$$

for $0 \le t \le 4$, and then superimpose them on the part of the contour plot with $0 \le y \le \pi/2$. By adjusting the aspect ratio using **axis**, you should clearly see the orthogonal families.

(c) $f(x,\ y) = x^3 + 3xy + y^3$ in the region $-2 \le x \le 2$, $-2 \le y \le 2$. Locate the critical points; there are two of them. This time, the differential equations are not very well behaved and you need to use **ode45**. The calculations may be problematic when trajectories get close to the critical points, so plot only a few trajectories, some starting on the line $x = -2$, and some starting on $y = -2$. In both cases, you want to go forward in time, say for $0 \le t \le 2$, to get integral curves lying in the desired region.

Glossary of MATLAB Commands and Options

@ Creates a function handle or anonymous function

axis Selects the ranges of x and y to show in a 2D-plot; or x, y, and z in a 3D-plot

diff Computes the derivative

double Converts the (possibly symbolic) expression for a number to a numerical (double-precision) value

dsolve Symbolic ODE solver

eval Evaluates a string as a MATLAB expression; useful in scripts

fcontour Plots the contour curves of a symbolic expression $f(x, y)$

fimplicit3 Plots a contour surface of a symbolic expression $f(x, y, z)$

fplot Easy function plotter

fplot3 Easy 3D function plotter

fsurf Easy 3D surface plotter

jacobian Computes the Jacobian matrix of a function of two or three variables

matlabFunction Converts a symbolic expression to a vectorized numerical function

max Computes the maximum of the entries of a vector

meshgrid Returns grid coordinates

ode45 Numerical differential equation solver

plot Numerical plotting routine

quiver Plots vector field based on grid coordinates supplied by **meshgrid**

realdot Not a built-in MATLAB function. Computes the dot product of two vectors with real coordinates

solve Symbolic equation solver

subs Substitutes for a variable in an expression

view Specifies a point from which to view a 3D graph

Options to MATLAB Commands

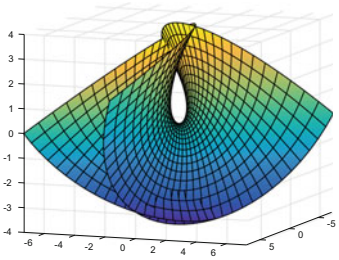
LevelList Controls the contours plotted by **fcontour**

LineColor Specifies color of curves in a graph

LineWidth Controls thickness of contours in a graph

MarkerSize Controls size of points in a graph

Chapter 6
Geometry of Surfaces



Just as a curve is (at least locally) the image of a continuous function from an interval in $\mathbb{R}$ to $\mathbb{R}^3$, a *surface* is (at least locally) the image of a continuous function from a domain in $\mathbb{R}^2$ to $\mathbb{R}^3$. We shall carry out a program for surfaces in space similar to our study of curves in Chapter 3. Not surprisingly, the details are more complicated. Therefore, we will be content on occasion to state results without elaborate justification. Our overall goals for surfaces are the same as they were for curves:

- to associate geometric invariants to surfaces;
- to describe and compute these invariants analytically using calculus; and
- to show that the invariants encode enough information about the geometric object to characterize it.

The main invariants we will investigate are the normal vector, the tangent plane, and various forms of curvature. You may think of the normal vector and tangent plane as analogs of the Frenet frame for a curve, and of the curvatures of a surface as the analogs of the curvature and torsion for a curve. Since surfaces are two-dimensional and curves only one-dimensional, there is a great deal more information to encode—thus accounting for increased complexity of the analytic formulas.

6.1 The Concept of a Surface

We shall define a surface by analogy with our definition of a curve. It is common to give a definition of surfaces that allows us to glue together several *patches*. To keep matters simple, however, we only deal with single patches. So, a surface $\boldsymbol{\Sigma}$ is the image of a continuous function $\sigma : D \to \mathbb{R}^3$, where D is either a rectangle or a disk in $\mathbb{R}^2$. The function σ is called a *parametrization* of the surface. If u and v are coordinates on D, then via σ we may think of them as parameters or *generalized coordinates* on $\boldsymbol{\Sigma}$. As with curves, we shall be ambiguous about whether the boundary of D is included or not.

In order to proceed, we need to impose additional conditions on the function σ. First, we say that σ is *regular* at a point if its partial derivatives, σ_u and σ_v,

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_6

are continuous, nonzero, and noncollinear there. (Points where σ is not regular are called *singular* points.) At a regular point (u_0, v_0), the level curves of u (with parametrization $v \mapsto \sigma(u_0, v)$) and of v (with parametrization $u \mapsto \sigma(u, v_0)$) provide a nice, cross-hatched family on the surface. Since the tangent vectors σ_v and σ_u to these curves do not lie on the same line, they must span a well-determined *tangent plane* at any regular point. The vector $\mathbf{N} = \sigma_u \times \sigma_v$ is normal to this tangent plane, and hence to the surface itself.

Next, we say that σ is a *patch* if it is regular and one-to-one with a continuous inverse function, except perhaps at some exceptional points. When there are no exceptional points, we say that σ is a *smooth patch*. On a smooth patch, the normal vector $\mathbf{N}$ is a nonzero continuous function, and thus the surface has an *orientation* in the sense that there is a preferred normal direction at each point. (This need not be true for surfaces composed of more than one patch.)

If we were to require our surface to be the image of a single smooth patch, then we would have to exclude certain standard (and important) surfaces such as the sphere and the cone. So, we allow our parameterizations to fail to be one-to-one or regular on a one-dimensional subset, just as we allow parametrizations for curves, such as the circle or the cycloid, to fail in the same way on a zero-dimensional subset (i.e., a discrete set of points). We defer a rigorous treatment of these matters to a more advanced course in advanced calculus, differential geometry, topology, or manifolds.

You have already encountered one important example of a patch. Consider a surface $\boldsymbol{\Sigma}$ that is the graph of a function φ of two variables. A *Monge patch* on $\boldsymbol{\Sigma}$ is given by the parametrization $\sigma(u, v) = (u, v, \varphi(u, v))$. In this case, σ is one-to-one with continuous inverse provided φ is continuous. Moreover,

$$\sigma_u = (1, 0, \varphi_u) \quad \text{and} \quad \sigma_v = (0, 1, \varphi_v)$$

are never collinear. So, regularity reduces to the assumption that the partial derivatives φ_u, φ_v are continuous.

6.1.1 Basic Examples

In this section, we discuss some basic examples of surfaces and explain how to parametrize and plot them in MATLAB.

6.1.1.1 The Sphere

The unit sphere (Figure 6.1) centered at the origin is parametrized by spherical coordinates:

$$\sigma(u, v) = (\cos u \sin v, \sin u \sin v, \cos v), \quad u \in [0, 2\pi], \quad v \in [0, \pi].$$

Here, u is called the *longitude*, v the *colatitude*. The coordinate functions,

$$x = \cos u \sin v, \qquad y = \sin u \sin v, \qquad \text{and} \quad z = \cos v,$$

satisfy the equation $x^2 + y^2 + z^2 = 1$. Thus, the sphere is a *quadric surface*—a surface determined by a quadratic equation in x, y, and z. (Other quadric surfaces were plotted in Problem Set E, Problem 5.5.) We can plot the sphere using **fsurf**.

```
>> syms u v real; sphere = [cos(u)*sin(v), sin(u)*sin(v), cos(v)];
>> fsurf(sphere(1), sphere(2), sphere(3),  [0 2*pi 0 pi])
>> view([1,1,1]); title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis equal; axis off;
```

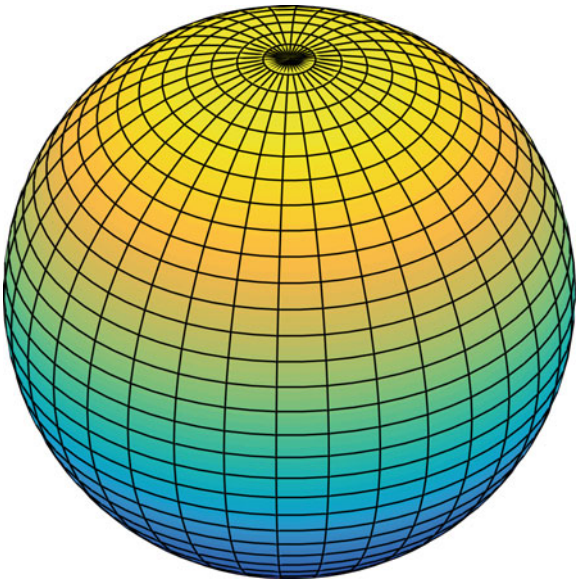


Fig. 6.1 The Unit Sphere

6.1.1.2 A Sinusoidal Cylinder

Any surface parametrized in the form

$$\sigma(u, v) = \mathbf{r}(u) + v\mathbf{q},$$

where $\mathbf{r}(u)$ is a plane curve and $\mathbf{q}$ is a fixed direction, perpendicular to the plane of $\mathbf{r}$, is called a *cylindrical surface*. The sinusoidal cylinder is a cylindrical surface determined by the usual sine curve in the x-y plane and the perpendicular z-direction:

$$\sigma(u, v) = (u, \sin u, v), \quad u \in [-2\pi, 2\pi], \quad v \in [-1, 1].$$

We again plot the surface using **fsurf** (Figure 6.2).

```
>> syms u v real; sincylinder = [u, sin(u), v];
>> fsurf(sincylinder(1), sincylinder(2), sincylinder(3), ...
     [-2*pi 2*pi -1 1])
>> view([1,1,1]); title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis equal; axis off;
```

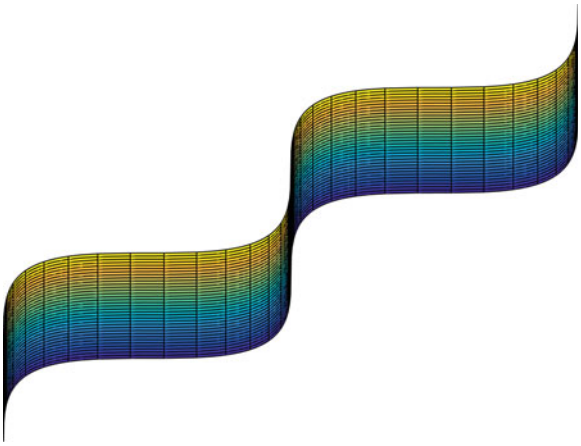


Fig. 6.2 A Sinusoidal Cylinder

6.1.1.3 A Cone

The cone is an example of a *ruled surface*, that is, one swept out by a straight line moving along a base curve. The curve need not be planar and the direction of the line may change. Still, a ruled surface must have a parametrization of the form

$$\sigma(u,v) = \mathbf{r}(u) + v\mathbf{q}(u),$$

where $\mathbf{r}$ is the curve along which the line is traveling and $\mathbf{q}$ is the direction of the line. (The line through the point $\mathbf{r}(u)$ is parametrized by $v \mapsto \mathbf{r}(u) + v\mathbf{q}(u)$.) Clearly, a cylindrical surface is ruled. The right circular cone may be parametrized as follows:

$$\sigma(u,v) = (v\cos u, v\sin u, v), \quad u \in [0, 2\pi], \quad v \in \mathbb{R}.$$

You should recognize this surface as a portion of the quadric surface $x^2 + y^2 - z^2 = 0$. The base curve is $\mathbf{r}(u) = (\cos u, \sin u, 1)$, and the line at the point $\mathbf{r}(u)$ is the one through the origin given by $v \mapsto v\mathbf{r}(u)$. We plot the double cone using this parametrization (Figure 6.3).

```
>> syms u v real; cone = [v*cos(u), v*sin(u), v];
>> fsurf(cone(1), cone(2), cone(3), [0 2*pi -1 1])
>> view([1,1,1]); title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis equal; axis off;
```

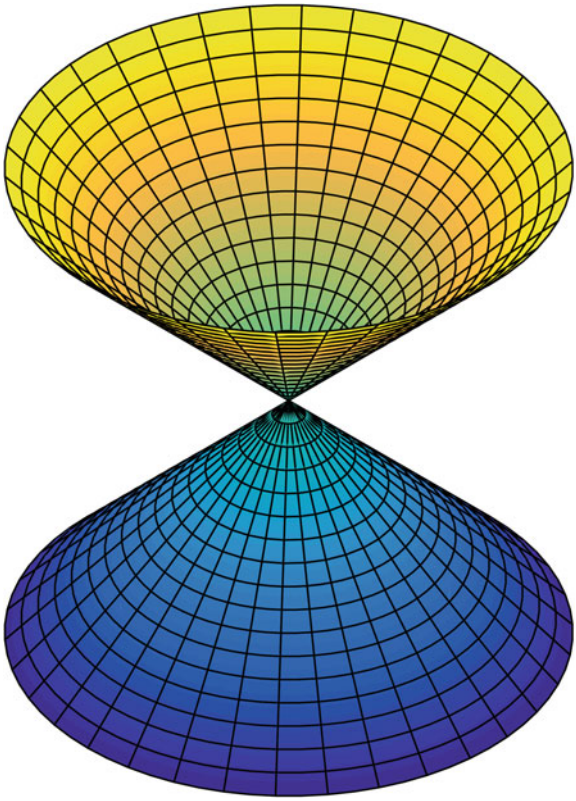


Fig. 6.3 A Double Cone

6.1.1.4 The Monkey Saddle

The graph of the function $z = x^3 - 3xy^2$ provides (Figure 6.4) an example of a Monge patch:

$$\sigma(u,v) = (u,v,u^3 - 3uv^2), \quad u \in [-2,2], \quad v \in [-2,2].$$

Nevertheless, the best way to visualize a Monge patch is still to employ **`fsurf`**:

```
>> syms u v real; figure; fsurf(u^3 - 3*u*v^2, [-2 2 -2 2])
>> view([1,1,1]); title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis off;
```

6.1.1.5 The Torus

The torus (Figure 6.5) is an example of a *surface of revolution*, that is, a surface obtained by revolving a plane curve about an axis that does not meet the curve. To

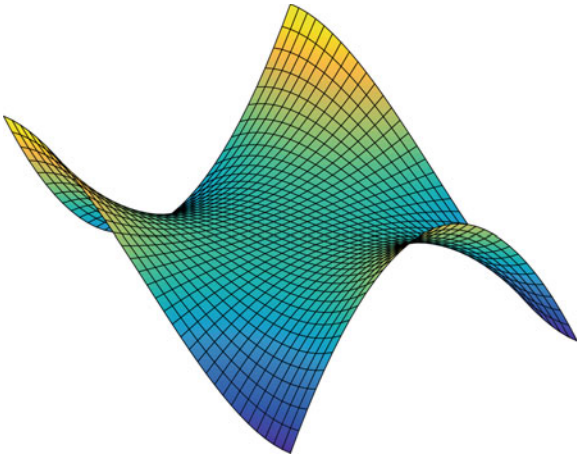


Fig. 6.4 The Monkey Saddle

simplify, assume the plane curve lies in the y-z plane, and that it is revolved around the z-axis. Then, if $(0, g(u), h(u))$ parametrizes the curve, a parametrization of the surface can be given by

$$\sigma(u,v) = (g(u)\cos v, g(u)\sin v, h(u)).$$

If we start with the circle $(y-2)^2 + z^2 = 1$, then

$$\sigma(u,v) = ((2+\cos u)\cos v, (2+\cos u)\sin v, \sin u), \quad u \in [0, 2\pi], \quad v \in [0, 2\pi],$$

and the resulting surface of revolution is a torus. MATLAB gives the picture:

```
>> syms u v real;
>> torus = [(2+cos(u))*cos(v), (2+cos(u))*sin(v), sin(u)];
>> fsurf(torus(1), torus(2), torus(3), [0 2*pi 0 2*pi])
>> view([4,4,2]); title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis equal; axis off;
```

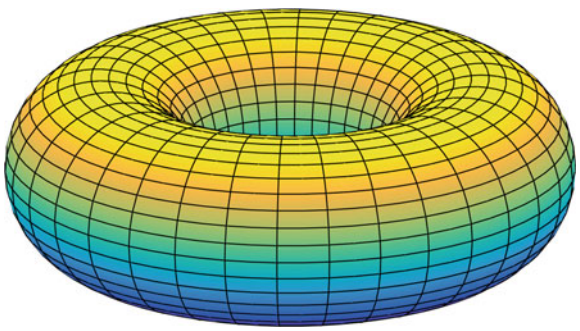


Fig. 6.5 A Torus

6.1.1.6 The Helicoid

The helicoid (Figure 6.6), obtained by connecting the points on the right circular helix horizontally to the z-axis, is another example of a ruled surface.

$$\sigma(u,v) = (v\cos u, v\sin u, u), \quad u \in [0, 3\pi], \quad v \in [0,3].$$

```
>> helicoid = [v*cos(u), v*sin(u), u];
>> fsurf(helicoid(1), helicoid(2), helicoid(3), [0 3*pi 0 3])
```

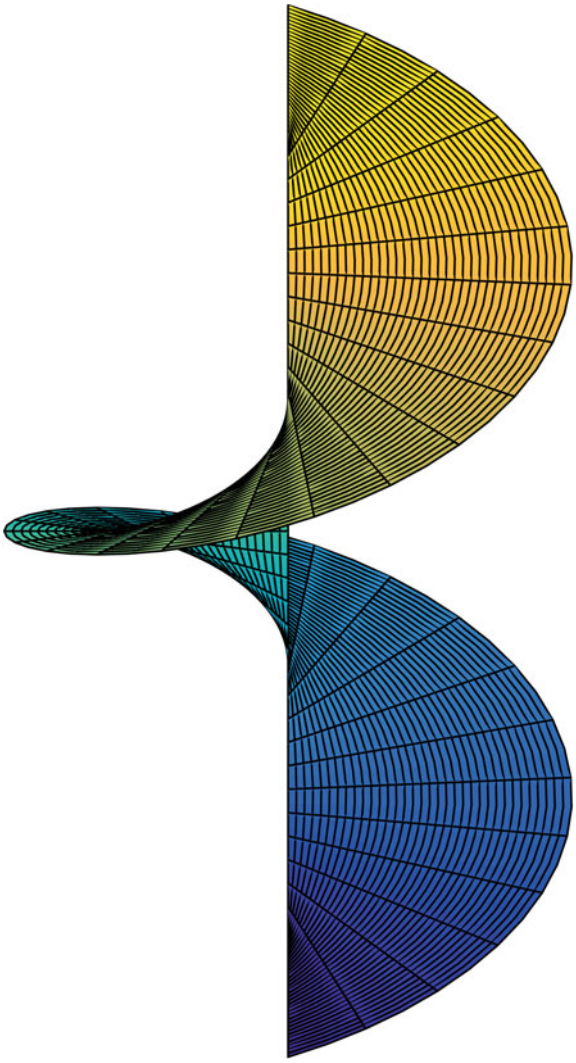


Fig. 6.6 The Helicoid

```
>> view([4,0,2]); title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis equal; axis off;
```

6.1.1.7 The Hyperboloid of One Sheet

In the last chapter, we looked at the hyperboloid of two sheets, given implicitly by the equation $x^2+y^2-z^2=-1$. We deferred to this chapter the hyperboloid of one sheet, given by $x^2+y^2-z^2=1$. The one-sheeted hyperboloid is both a quadric surface and a ruled surface. A ruled parametrization

$$\sigma(u,v)=\mathbf{r}(u)+v\mathbf{q}(u),$$

can be given by taking

$$\mathbf{r}(u)=(\cos u,\sin u,0),\qquad \mathbf{q}(u)=(-\sin u,\cos u,1),$$

or else by pairing the same function $\mathbf{r}(u)$ with $\mathbf{q}(u)=(\sin u,-\cos u,1)$. (The two rulings are mirror images of one another.) We check using MATLAB that the parametrization satisfies the equation of the hyperboloid, and then sketch the surface, which appears in Figure 6.7.

```
>> hyperboloid = [cos(u), sin(u), 0] + v*[-sin(u), cos(u), 1];
>> simplify(hyperboloid(1)^2 + hyperboloid(2)^2 - ...
     hyperboloid(3)^2)
```

```
1
```

```
>> fsurf(hyperboloid(1), hyperboloid(2), hyperboloid(3), ...
      [0 2*pi, -2 2])
>> title(''); xlabel(''); ylabel(''); zlabel('');
>> grid off; axis equal; axis off;
```

6.2 The Implicit Function Theorem

As we saw in Chapter 5, *Directional Derivatives*, surfaces can also be described as the locus of solutions to an equation $f(x,y,z)=c$. In order to understand how, at least locally, the solution set is a surface in the sense of this chapter, we must understand the Implicit Function Theorem. Let us first consider it for curves and then explain the generalization to surfaces.

Suppose we have a function $f(x,y)$ and we consider a level curve $f(x,y)=c$. Typically, such curves look like Figure 6.8.

We would like to know which points (x_0,y_0) on the curve have the property that we can find exactly one smooth function $y=\varphi(x)$, defined in some interval I containing x_0 in its interior, for which $f(x,\varphi(x))=c$, for all $x\in I$. Clearly, for the

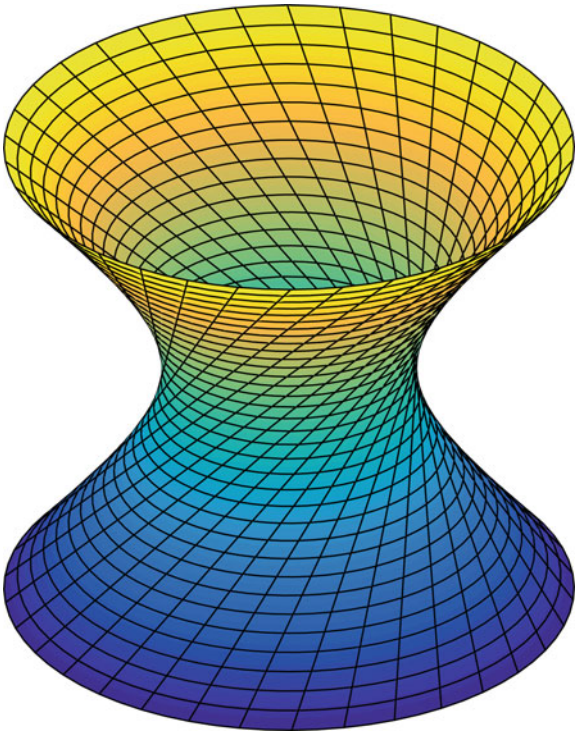


Fig. 6.7 The Hyperboloid of One Sheet

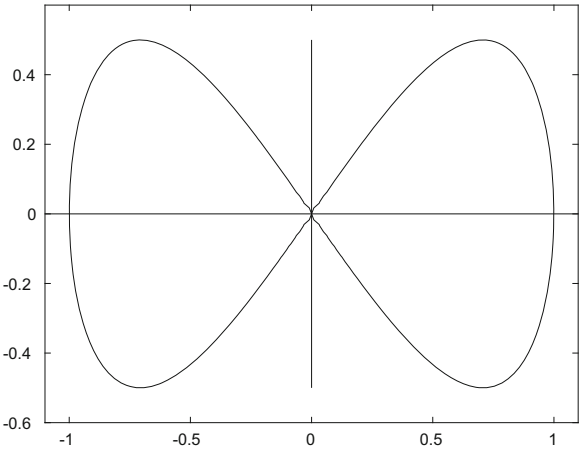


Fig. 6.8 A level curve

curve depicted, this will not be the case at the vertical and the crossing points, but it will be true elsewhere. Here is a basic theorem that answers the question.

Theorem 6.1 (Implicit Function Theorem). *Suppose that f, f_x, and f_y are continuous in a neighborhood of (x_0, y_0). Moreover, suppose that $f(x_0, y_0) = c$ and $f_y(x_0, y_0) \neq 0$. Then there is an open interval I containing x_0 such that for every $x \in I$ there is a unique number y for which $f(x, y) = c$. The function $y = \varphi(x)$ thus defined, and its derivative, are continuous functions on I.*

Proof (Idea of Proof). If $f_y(x_0, y_0) \neq 0$, then it is either positive or negative. Without loss of generality, we may assume it is positive (otherwise replace f by $-f$). Since f_y is continuous, we may assume f_y is positive on some disk containing (x_0, y_0). In particular, if $F(x, y) = f(x, y) - c$, then $y \mapsto F(x_0, y)$ is an increasing function that vanishes at (x_0, y_0). So, there is a small positive number k for which

$$F(x_0, y_0 + k) > 0 \quad \text{and} \quad F(x_0, y_0 - k) < 0.$$

By continuity of F, we can find another small number h so that

$$F(x, y_0 + k) > 0 \quad \text{and} \quad F(x, y_0 - k) < 0 \quad \text{for} \quad |x - x_0| < h.$$

For any x in this small interval, by the Intermediate Value Theorem, there must be a value of y for which $F(x, y) = 0$. Since this value depends on x, we denote it by $y = \varphi(x)$. Moreover, because the function F_y is positive on the original disk, the value $\varphi(x)$ is uniquely determined by x. Thus, we have found the desired function. We omit the proof of continuity for φ and its derivative. (See any Advanced Calculus textbook.) □

Now, it is easy to see how the result generalizes. We state it without proof.

Theorem 6.2 (Implicit Function Theorem (for Three Variables)). *Suppose that $f(x, y, z)$ and f_x, f_y, f_z are continuous on a neighborhood of (x_0, y_0, z_0). Moreover, if $f(x_0, y_0, z_0) = c$ and $f_z(x_0, y_0, z_0) \neq 0$, then there is an open disk D containing (x_0, y_0) such that for every $(x, y) \in D$ there is a unique number $z = \varphi(x, y)$ for which $f(x, y, \varphi(x, y)) = c$. The parametrization $\mathbf{x}(u, v) = (u, v, \varphi(u, v))$, $(u, v) \in D$, thus defined, yields a smooth Monge patch surface on which the function f takes only the value c.*

Similarly, we can show that if $f(x, y, z)$ is a continuously differentiable function, if $f(x_0, y_0, z_0) = c$, and if $\nabla f(x_0, y_0, z_0) \neq 0$, then the level surface $f(x, y, z) = c$ is indeed a surface in the sense of this chapter in the vicinity of (x_0, y_0, z_0), even though it may not have a Monge patch parametrization.

6.3 Geometric Invariants

Now, we begin to develop geometric invariants associated with surfaces. Let $\sigma(u,v)$ be defined on a domain D and parametrize a smooth surface $\mathbf{\Sigma}$. The vectors σ_u, σ_v are, by hypothesis, noncollinear. Each is tangent to a curve on the surface, so they span a plane, called the *tangent plane*. Now any curve on the surface corresponds to a curve in D. Here is where we use the inverse continuity property of the parametrization! Thus, curves on the surface may be parametrized by $t \mapsto \sigma(\mathbf{r}(t))$, where $\mathbf{r}(t) = (u(t), v(t))$ is a plane curve inside D. Then, a tangent vector to the surface is obtained by differentiating with respect to t. By the chain rule, we obtain

$$\frac{d}{dt}\sigma(\mathbf{r}(t)) = \sigma_u u'(t) + \sigma_v v'(t).$$

That is, any tangent vector is a linear combination of σ_u and σ_v, and our notion of the tangent plane is a good one. We shall always write

$$\mathbf{U}(u,v) = \frac{\sigma_u(u,v) \times \sigma_v(u,v))}{\|\sigma_u(u,v) \times \sigma_v(u,v)\|};$$

it is a unit normal vector to the surface at the point $\sigma(u,v)$. It is a *first-order* object since it involves first derivatives. In order to characterize curves, we needed second-order (curvature) and third-order (torsion) objects. Because of the extra degrees of freedom in space, we only need second-order objects to characterize surfaces. But we shall see that there are several different kinds of curvature that we can associate to a surface. More than one of them will be required for characterization.

All of the curvatures arise from differentiating the normal vector. The normal vector is an example of a *vector field*, a vector-valued function defined at each point of the surface. We can differentiate a vector field in any direction on the surface, i.e., in any direction specified by a tangent vector. This amounts to a directional derivative, analogous to the directional derivatives applied to scalar-valued functions.

Differentiating the normal vector gives rise to something called the *shape operator* . It is defined as follows. Let $P_0 = \sigma(u_0, v_0)$ be a point on the surface $\mathbf{\Sigma}$. Let $\mathbf{T}$ be a tangent vector to $\mathbf{\Sigma}$ at P_0. Let $\alpha(t) = \sigma(\mathbf{r}(t))$ be a curve on the surface having $\mathbf{T}$ as tangent vector at P_0; that is,

$$\mathbf{r}(0) = (u_0, v_0), \quad \alpha(0) = \sigma(\mathbf{r}(0)) = P_0, \quad \alpha'(0) = \mathbf{T}.$$

We set

$$S_{P_0}(\mathbf{T}) = -D_{\mathbf{T}}\mathbf{U} = -\frac{d}{dt}\mathbf{U}(\alpha(t))_{t=0}.$$

In fact, the value of $S_{P_0}(\mathbf{T})$ is independent of the choice of the curve α. (This is seen by an argument analogous to the one in Chapter 3.)

We call S_{P_0} an *operator* because it converts one tangent vector into another tangent vector. Indeed, since $\mathbf{U}\cdot\mathbf{U}$ is constant, the usual computation shows that $S_{P_0}(\mathbf{T})\cdot\mathbf{U}(P_0)=0$. So, $S_{P_0}(\mathbf{T})$ is perpendicular to $\mathbf{U}(P_0)$ and again lies in the tangent plane. The shape operator is easily checked to be *linear* on the two-dimensional tangent plane; that is, it satisfies

$$S_P(a\mathbf{T}_1+b\mathbf{T}_2)=aS_P(\mathbf{T}_1)+bS_P(\mathbf{T}_2),\qquad a,b\in\mathbb{R}.$$

Therefore, we can use the language of linear algebra to express it as a 2×2 matrix. If $\mathbf{T}_1$ and $\mathbf{T}_2$ are linearly independent vectors in the tangent plane, then

$$S_P(\mathbf{T}_1)=\alpha\mathbf{T}_1+\beta\mathbf{T}_2,\qquad S_P(\mathbf{T}_2)=\gamma\mathbf{T}_1+\delta\mathbf{T}_2,$$

for some real numbers $\alpha,\beta,\gamma,\delta$. We say that the matrix $\begin{pmatrix}\alpha & \gamma\\ \beta & \delta\end{pmatrix}$ represents the operator S_P. (Of course, the four numbers depend on P, i.e., on the location on $\boldsymbol{\Sigma}$, but we often suppress the subscript P.) The matrix representation is unique up to conjugation. In other words, if $\begin{pmatrix}\alpha' & \gamma'\\ \beta' & \delta'\end{pmatrix}$ results from another choice of basis, then $\begin{pmatrix}\alpha & \gamma\\ \beta & \delta\end{pmatrix}=A\begin{pmatrix}\alpha' & \gamma'\\ \beta' & \delta'\end{pmatrix}A^{-1}$ for some invertible matrix A. In fact, the invariants we will attach to the operator S_P will be unchanged by matrix conjugation (i.e., they are independent of the choice of basis).

Well, the actual computation of the matrix of S is "painful." Differentiating the unit normal and representing the result as a linear combination of σ_u and σ_v is a tedious chore. But it has been done many times by many people, and there is a standard presentation of the results, which we now supply. Define scalar-valued functions on the surface by

$$E=\sigma_u\cdot\sigma_u,\qquad F=\sigma_u\cdot\sigma_v,\qquad G=\sigma_v\cdot\sigma_v.$$

Also define

$$e=\mathbf{U}\cdot\sigma_{uu},\qquad f=\mathbf{U}\cdot\sigma_{uv},\qquad g=\mathbf{U}\cdot\sigma_{vv}.$$

Then we have

$$S(\sigma_u)=\frac{eG-fF}{EG-F^2}\sigma_u+\frac{fE-eF}{EG-F^2}\sigma_v,$$

$$S(\sigma_v)=\frac{fG-gF}{EG-F^2}\sigma_u+\frac{gE-fF}{EG-F^2}\sigma_v.$$

(See [1, p. 275].) Now $\mathbf{U}$, being perpendicular to the tangent plane, satisfies $\mathbf{U}\cdot\sigma_u=0$. If we differentiate that equation with respect to v, we obtain $\mathbf{U}_v\cdot\sigma_u+\mathbf{U}\cdot\sigma_{uv}=0$. But $\mathbf{U}_v=-S(\sigma_v)$ by definition; hence

$$S(\sigma_v)\cdot\sigma_u=\mathbf{U}\cdot\sigma_{uv}=f.$$

Reversing the roles of u and v, we see that $S(\sigma_u)\cdot\sigma_v$ also equals f; thus

$$S(\sigma_u)\cdot\sigma_v = \sigma_u\cdot S(\sigma_v).$$

Since σ_u, σ_v span the tangent plane and since S is linear, we have $S(\mathbf{T}_1)\cdot\mathbf{T}_2 = \mathbf{T}_1\cdot S(\mathbf{T}_2)$ for any two tangent vectors $\mathbf{T}_1, \mathbf{T}_2$. If we choose $\mathbf{T}_1$ and $\mathbf{T}_2$ orthonormal (i.e., of unit length and mutually perpendicular), and if $\begin{pmatrix}\alpha & \gamma\\ \beta & \delta\end{pmatrix}$ denotes the matrix of S with respect to this choice of basis, then $S(\mathbf{T}_1)\cdot\mathbf{T}_2 = \beta$ and $\mathbf{T}_1\cdot S(\mathbf{T}_2) = \gamma$. Hence $\beta = \gamma$, and S is represented by a *symmetric* matrix (one which equals its own transpose). We use the fact from linear algebra that any symmetric matrix can be diagonalized. Thus, if $S = \begin{pmatrix}a & b\\ b & c\end{pmatrix}$, we can choose an orthogonal matrix $A = \begin{pmatrix}\cos\theta & \sin\theta\\ -\sin\theta & \cos\theta\end{pmatrix}$ so that $ASA^{-1} = \begin{pmatrix}k_1 & 0\\ 0 & k_2\end{pmatrix}$. In fact,

$$k_1, k_2 = \frac{(a+c)\pm\sqrt{(a-c)^2+4b^2}}{2}.$$

In linear algebra, the numbers k_1, k_2 are called the *eigenvalues* of the matrix. If we rotate the original basis by an angle θ, the new basis satisfies

$$S(\mathbf{T}_1) = k_1\mathbf{T}_1, \qquad S(\mathbf{T}_2) = k_2\mathbf{T}_2.$$

The perpendicular directions $\mathbf{T}_1, \mathbf{T}_2$ are called *principal directions* and the eigenvalues are called the *principal curvatures*. More generally, for any unit tangent vector $\mathbf{T}$, the number $k(\mathbf{T}) = S(\mathbf{T})\cdot\mathbf{T}$ is called the *normal curvature* in the direction $\mathbf{T}$. Since any $\mathbf{T}$ may be written $\mathbf{T} = a\mathbf{T}_1 + b\mathbf{T}_2$, $a^2+b^2 = 1$, then $k(\mathbf{T}) = a^2k_1 + b^2k_2$. Clearly (if $k_1 \geq k_2$), the normal curvature is largest (resp., smallest) in the principal direction of k_1 (resp., k_2).

Exercise 6.3. Show that the quadric surface $z = \frac{1}{2}(k_1x^2 + k_2y^2)$ has the same shape operator at the origin as $\boldsymbol{\Sigma}$ does at P if k_1 and k_2 are the principal curvatures there.

To clarify Exercise 6.3 and the discussion that precedes it, we present two graphs in Figure 6.9. In both, the normal vector $\mathbf{U}$ and the principal directions $\mathbf{T}_1, \mathbf{T}_2$ are shown at one point on the surface. Curves through the point having those directions as tangent vectors are also displayed. In the left graph, the principal curvatures k_1, k_2 have opposite signs; in the right graph, they have the same sign. In the left graph, one principal direction is partly obscured by the surface.

Let us give some geometric content to the notion of curvature. Let $\alpha(t) = \sigma(\mathbf{r}(t))$ be a curve on $\boldsymbol{\Sigma}$, $\alpha(0) = P$. Then $\alpha'\cdot\mathbf{U} = 0$. Differentiating again, we get $\alpha''\cdot\mathbf{U} + \alpha'\cdot\mathbf{U}' = 0$, that is,

$$\alpha''\cdot\mathbf{U} = \alpha'\cdot S(\alpha').$$

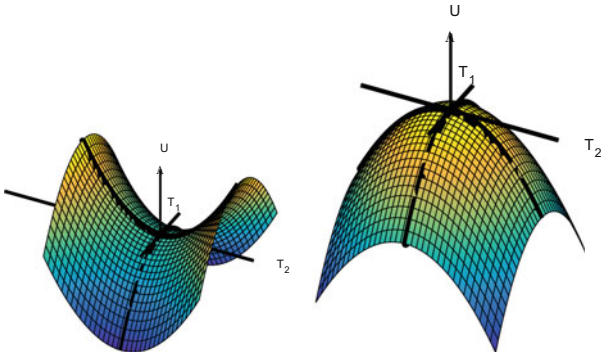


Fig. 6.9 Negative and positive curvature

If α is unit speed and $\alpha'(0) = \mathbf{T}$, we can compute the normal curvature in the direction $\mathbf{T}$ as follows:

$$k(\mathbf{T}) = S(\mathbf{T}) \cdot \mathbf{T} = \alpha''(0) \cdot \mathbf{U}(P) = \kappa(0)\mathbf{N}(0) \cdot \mathbf{U}(P) = \kappa(0)\cos\theta,$$

where $\mathbf{N}(0)$ is the principal normal to the curve α at the point P, $\kappa(0)$ is the curvature of the curve there, and θ is the angle between the two normals $\mathbf{N}(0)$ and $\mathbf{U}(P)$. If we pick the curve to lie in the intersection of the surface with the plane determined by $\mathbf{U}(P)$ and $\mathbf{T}$, then $\mathbf{N}(0)$ must be in that plane; so it must be parallel to $\mathbf{U}(P)$ and $\cos\theta = \pm 1$. In other words, $k(\mathbf{T})$, the normal curvature of the surface in the direction $\mathbf{T}$, is ± 1 times the curvature $\kappa(0)$ of the curve. (The ambiguity in sign is due to the fact that the surface only determines $\mathbf{U}$ up to sign; specifying $\mathbf{U}$ amounts to fixing an orientation.) If $k(\mathbf{T}) = \kappa(0) > 0$, the surface is bending toward $\mathbf{U}(P)$ in the direction $\mathbf{T}$; if $k(\mathbf{T}) < 0$, the surface is bending away in that direction. If $k(\mathbf{T}) = 0$, then $\kappa(0) = 0$ and $\mathbf{N}(0)$ is not defined—the surface is not bending at all in that direction.

We can enrich the discussion by introducing two more forms of curvature:

Definition 6.4. Define the *Gaussian curvature* K to be the product of the principal curvatures $K = k_1 k_2$, and the *mean curvature* H to be their average $H = \frac{1}{2}(k_1 + k_2)$.

Remark 6.5. The Gaussian curvature is the *determinant* of the matrix S; and the mean curvature is one-half of its *trace*.

Experience has shown that the Gaussian curvature is the most useful of these invariants. The reason has to do with the following remarkable fact discovered by Gauss. Take three points P, Q, and R on the surface $\boldsymbol{\Sigma}$ that are close together, so that for all practical purposes

$$K(P) \approx K(Q) \approx K(R).$$

Join these points by a *geodesic triangle* in $\boldsymbol{\Sigma}$. (The term "geodesic" comes from surveying; literally, it means "earth-dividing.") In other words, we join P to Q, P to

R, and Q to R by curves lying in the surface, chosen to be as short as possible. (Line segments will usually not work, since the line segment in $\mathbb{R}^3$ joining two points in $\boldsymbol{\Sigma}$ may not be contained in $\boldsymbol{\Sigma}$.) Since the triangle ΔPQR is "curved," the angles of this triangle need not add up exactly to π. However, Gauss observed that

$$\angle RPQ + \angle PQR + \angle QRP - \pi \approx \text{area}(\Delta PQR) \cdot K(P), \tag{6.1}$$

and that this formula is exact if the Gaussian curvature of $\boldsymbol{\Sigma}$ is constant.

The sphere of radius 1, which has constant Gaussian curvature, provides a good illustration of this phenomenon. Consider the geodesic triangle with vertices $P =$ the north pole and Q and R on the equator. Suppose the longitudes of the points Q and R differ by θ. The sides PQ and PR of the triangle are meridians, curves of constant longitude, which meet the equator at right angles. So

$$\angle RPQ = \theta, \qquad \angle PQR = \frac{\pi}{2}, \qquad \angle QRP = \frac{\pi}{2}.$$

This is illustrated in Figure 6.10. Thus the left-hand side of the equation (6.1) is θ. On the other hand, the area of ΔPQR is $\theta/(2\pi)$ times the area of the northern hemisphere, which is 2π, so

$$\text{area}\,(\Delta PQR) = \frac{\theta}{2\pi} \cdot (2\pi) = \theta,$$

as required by Gauss' formula.

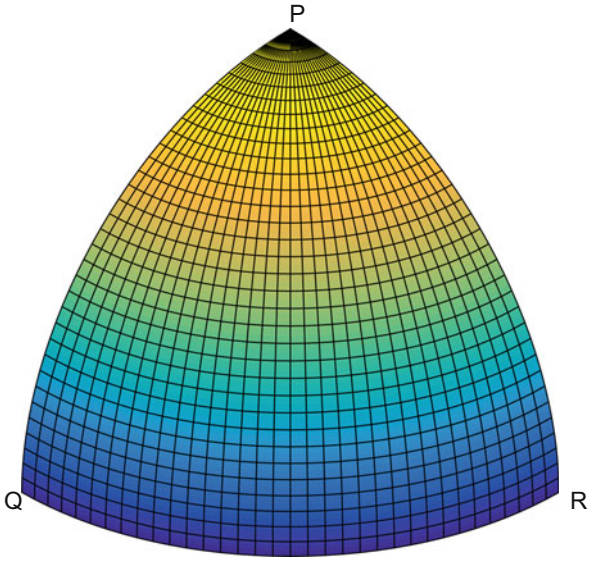


Fig. 6.10 Geodesic Triangle on a Sphere

Now, we return to our general discussion of curvature invariants. If $K(P) > 0$, the principal curvatures have the same sign, and the surface is bending in the same direction as you proceed in all directions from P. Locally, the surface looks like the paraboloid $2z = k_1x^2 + k_2y^2$ near P. If $K(P) < 0$, the principal curvatures have opposite sign. In one direction the surface bends "up," in the perpendicular direction it bends "down," and it could bend up and down in many different ways in between. (P is a *saddle point*, but the surface may resemble a hyperboloid, a monkey saddle, or some other wavy object). If $K(P) = 0$, there are two possibilities. If only one principal curvature vanishes, the surface resembles the trough-shaped paraboloid $2z = k_1x^2$. If both vanish, then locally the surface is flat and we cannot really say much more. Geometers use the following terminology: points where $K > 0$ are called *elliptic*, points where $K < 0$ are called *hyperbolic*, and points where $K = 0$ are called *parabolic*.

Before proceeding, we redraw the two examples above in Figure 6.11, replacing the principal directions by the tangent plane. The tangent plane in the first example appears on both sides of the surface, because the Gaussian curvature is negative. In the second example, the Gaussian curvature is positive and the tangent plane lies completely on one side of the surface.

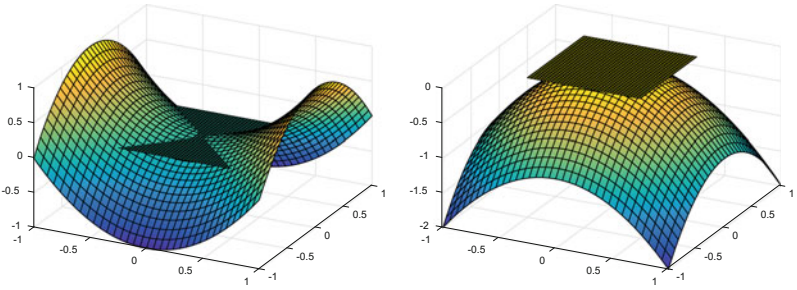


Fig. 6.11 Tangent planes in negative and positive curvature

We shall not examine the mean curvature in detail in this chapter. The mean curvature is related to the question of how the (local) area of a surface varies under small perturbations. Suffice it to say that a surface is called a *minimal surface* if its mean curvature vanishes. The name comes from the fact that a surface that minimizes area (locally) is a minimal surface. The exploration of minimal surfaces has a venerable history, because it has been known for a long time that minimal surfaces have an intuitively appealing application: a soap bubble tends to assume the shape of a minimal surface.[1] We refer the interested reader to a course in differential geometry for more details. Instead, we simply present the formulas for K and H in terms of the shape operator invariants. Here they are, first for a general patch:

[1]This is because surface tension causes the surface to contract until its area is as small as possible subject to the appropriate boundary conditions.

$$K = \frac{eg - f^2}{EG - F^2}, \qquad H = \frac{eG - 2fF + gE}{2(EG - F^2)},$$

and then for a Monge patch:

$$K = \frac{\varphi_{uu}\varphi_{vv} - \varphi_{uv}^2}{(1 + \varphi_u^2 + \varphi_v^2)^2}, \qquad H = \frac{(1 + \varphi_v^2)\varphi_{uu} - 2\varphi_u\varphi_v\varphi_{uv} + (1 + \varphi_u^2)\varphi_{vv}}{(1 + \varphi_u^2 + \varphi_v^2)^{3/2}}.$$

We conclude this chapter with a proposition that is reminiscent of the characterization results in Chapter 3, and with the analog of the congruence results (Theorem 3.5 and Corollary 3.6) in that chapter.

We say that a point P on $\boldsymbol{\Sigma}$ is *umbilic* if the normal curvature is constant there. Thus $k_1(P) = k_2(P)$, whence $K(P) \geq 0$.

Proposition 6.6. *If every point on a surface is umbilic, the surface is a portion of either a plane or a sphere.*

Proof. We show that not only is the normal curvature $k(\mathbf{T})$ independent of $\mathbf{T}$, but the value is independent of the point P. In fact, if we differentiate the following equations with respect to v and u, respectively,

$$\mathbf{U}_u = -S(\sigma_u) = -k\sigma_u,$$
$$\mathbf{U}_v = -S(\sigma_v) = -k\sigma_v,$$

we obtain

$$\mathbf{U}_{uv} = \mathbf{U}_{vu} = -k_v\sigma_u - k\sigma_{uv} = -k_u\sigma_v - k\sigma_{vu}.$$

Therefore, $k_v\sigma_u = k_u\sigma_v$. Since σ_u and σ_v are linearly independent, $k_u = k_v = 0$. Hence, the principal (and normal) curvatures are constant. It follows that the Gaussian curvature is a nonnegative constant. If $K \equiv 0$, then $D_{\mathbf{T}}\mathbf{U} = 0$ for any direction $\mathbf{T}$. So $\mathbf{U}$ is constant and we have a plane. If $K > 0$, we reason as follows. Let P and Q be any two points on $\boldsymbol{\Sigma}$. Select a curve α on $\boldsymbol{\Sigma}$ connecting the points, say $\alpha(0) = P$, $\alpha(1) = Q$. Define a function $\gamma(t) = \alpha(t) + (1/k)\mathbf{U}(\alpha(t))$. Differentiating, we find $\gamma'(t) = \alpha'(t) + (1/k)(\mathbf{U} \circ \alpha)'(t) = \alpha'(t) - (1/k)S(\alpha'(t)) = \alpha'(t) - \alpha'(t) = \mathbf{0}$. Thus, γ is constant, say $\mathbf{c}$. Since the length of $\mathbf{U}$ is constantly equal to 1, we see that every point on the surface is at a constant distance $1/k$ from the point $\mathbf{c}$. □

Finally, here is the congruence theorem (see [2, Ch. 6]):

Theorem 6.7. *Let $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ be two surfaces. Suppose there is a one-to-one map $\mathbf{F}$ from $\boldsymbol{\Sigma}_1$ onto $\boldsymbol{\Sigma}_2$ satisfying*

(i) $\mathbf{F}$ *takes the* shape operator S_1 *at every point P_1 on $\boldsymbol{\Sigma}_1$ to the shape operator S_2 at the corresponding point $\mathbf{F}(P_1)$ on $\boldsymbol{\Sigma}_2$; and*
(ii) $\mathbf{F}$ *preserves distances in the tangent planes*, $\|\mathbf{F}(\mathbf{T})\| = \|\mathbf{T}\|$.

Then there is a congruence of 3-space, composed of a rotation and a translation, that takes $\boldsymbol{\Sigma}_1$ *onto* $\boldsymbol{\Sigma}_2$.

A one-to-one correspondence between two surfaces satisfying property (ii) is the analog of a correspondence between two curves under which the speeds of the curves match up. Such a correspondence is called an *isometry*. Thus, we see that isometric surfaces that have the same shape operator are congruent, giving substance to the assertion that *the shape operator encodes enough information to characterize the shapes of surfaces.*

This leads us to another question. When is there an isometry between surfaces that is *not* necessarily implemented by a congruence? An important aspect of this question was answered by Gauss, and again points to the fundamental role of the Gaussian curvature.

Theorem 6.8 (Gauss). *Let* $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ *be two surfaces. If* $\mathbf{F}$ *is a smooth map from* $\boldsymbol{\Sigma}_1$ *to* $\boldsymbol{\Sigma}_2$ *which is a (local) isometry, i.e., which preserves distances in the tangent planes, then* $\mathbf{F}$ *preserves Gaussian curvature, i.e., for every point* P_1 *on* $\boldsymbol{\Sigma}_1$,

$$K_{\boldsymbol{\Sigma}_1}(P_1) = K_{\boldsymbol{\Sigma}_2}(\mathbf{F}(P_1)).$$

(*However,* $\mathbf{F}$ *need* not *preserve the mean curvature.*)

6.4 Curvature Calculations with MATLAB

Even though the calculations of the principal curvatures of a surface are quite messy to do by hand, it is not hard to automate the calculations using MATLAB. We develop MATLAB functions to do the calculations, and then apply them to some examples. The input for our functions should be a vector-valued function **`surf`** of two parameters **`u`** and **`v`**, giving the parametrization of our surface. We define the functions using handles, since we do not want to do the actual calculations before specifying the function **`surf`**. (See Question 3 in Chapter 11, MATLAB *Tips*, for an explanation.)

We start by computing the normal vector, unit normal vector, and the quantities e, E, etc. For the same reasons as in Chapter 3, we replace MATLAB's **`dot`** with **`realdot`**.

```
>> realdot = @(v, w) v*transpose(w);
>> veclength = @(vec) sqrt(realdot(vec, vec));
>> normalvector =  @(surf,u,v) simplify(cross(diff(surf,u), diff(surf,v)));
>> tangentplane = @(surf, u, v, x, y, z) realdot([x, y, z] - ...
  surf, normalvector(surf, u, v)) == 0;
>> unitnorm = @(surf, u, v) normalvector(surf, u, v)/ ...
  veclength(normalvector(surf, u, v));


>> E = @(surf, u, v) realdot(diff(surf, u), diff(surf, u));
>> F = @(surf, u, v) realdot(diff(surf, u), diff(surf, v));
```

```
>> G = @(surf, u, v) realdot(diff(surf, v), diff(surf, v));
>> e = @(surf,u,v) realdot(unitnorm(surf,u,v), diff(surf,u,2));
>> f = @(surf,u,v) realdot(unitnorm(surf,u,v), diff(surf,u,v));
>> g = @(surf,u,v) realdot(unitnorm(surf,u,v), diff(surf,v,2));


>> shapeoperator = @(surf, u, v) ...
   ((E(surf,u,v)*G(surf,u,v)-F(surf,u,v)^2)^(-1)) ...
      *[[e(surf,u,v)*G(surf,u,v)-f(surf,u,v)*F(surf,u,v), ...
      f(surf,u,v)*E(surf,u,v)  - e(surf,u,v)*F(surf,u,v)]; ...
      [f(surf,u,v)*G(surf,u,v) - g(surf,u,v)*F(surf,u,v), ...
      g(surf,u,v)*E(surf,u,v)  - f(surf,u,v)*F(surf,u,v)]];
```

Now, we can compute the principal curvatures, using MATLAB's routine for computing the eigenvalues of a matrix.

```
>> principalcurves = @(surf, u, v) eig(shapeoperator(surf, u, v))
```

We could compute the Gaussian curvature and the mean curvature from the principal curvatures, but it is easier to compute them directly.

```
>> gausscurv = @(surf, u, v) det(shapeoperator(surf, u, v));
>> meancurv = @(surf, u, v) trace(shapeoperator(surf, u, v))/2;
```

Next, we try our programs on three of the surfaces described at the beginning of the chapter. In these and in many cases for which you will employ these functions, the output that MATLAB supplies requires simplification. Usually it is enough to invoke MATLAB's **simplify** command. However, in some instances, MATLAB does not utilize sufficiently many algorithms in its attempt to simplify. We can ask it to employ more algorithms with the **Steps** option (see below). We start with the sphere:

```
>> principalcurves(sphere, u, v);
>> simplify(ans)

sign(sin(v))
sign(sin(v))
```

Both principal curvatures here are really 1, because $0 \leq v \leq \pi$, hence $\sin v$ is positive. Therefore, the mean curvature and Gaussian curvature are everywhere equal to 1. As a check, we have

```
>> gausscurv(sphere, u, v);
>> simplify(ans)

1

>> meancurv(sphere,u,v);
>> simplify(ans)

sign(sin v)
```

As before, this really equals 1. Next the hyperboloid of one sheet:

```
>> principalcurves(hyperboloid, u, v);
>> simplify(ans, 'Steps', 30)

 1/(v^2 + 1)
-1/(v^2 + 1)

>> gausscurv(hyperboloid, u, v);
>> simplify(ans, 'Steps', 30)

-1/(2*v^2 + 1)^2
```

The hyperboloid of one sheet has negative Gaussian curvature, with the curvature tending to zero at infinity. As a final example, we consider the helicoid.

```
>> principalcurvs(helicoid, u, v)
>> simplify(ans)

 1/(v^2 + 1)
-1/(v^2 + 1)
```

Since the principal curvatures are negatives of one another, the helicoid is an example of a minimal surface. Here's a check:

```
>> meancurv(helicoid, u, v)

0
```

Problem Set F. Surfaces

When working on this problem set, you may use the programs in Chapter 6, *Geometry of Surfaces*. It is not necessary to retype them; they may be found on the companion web site for this book.

Problem 6.1. Use the programs in Chapter 6 to compute the Gaussian curvature and mean curvature for the remaining examples from the chapter: the sinusoidal cylinder, the cone, the monkey saddle, and the torus. In each case, interpret the results. Here are a few special points to look for:

(a) In the case of the sinusoidal cylinder, what is the meaning of the Gaussian curvature calculation? (If you feel you need more information, compute the principal curvatures.)

(b) In the case of the cone, what is the Gaussian curvature away from the vertex, and what does this mean? What happens at the vertex?

(c) In the case of the monkey saddle, the surface does not have rotational symmetry. Nevertheless, what symmetry do you observe in the Gaussian curvature calculation?

(d) In the case of the torus, what kind of symmetry is there and why? The Gaussian curvature varies from negative to zero to positive. Can you see why?

Problem 6.2. Compute formulas for the tangent plane, Gaussian curvature, and mean curvature at each point of the following Monge patch surfaces. Plot each surface. In each case, identify from your formula all points where the tangent plane is either horizontal or vertical, and reconcile this with your plot. Also, make any relevant comments about the Gaussian curvature—such as whether it is constant, where it is positive and where negative, and whether it manifests any symmetry properties (for instance, whether it is unchanged by translations in any direction, or rotations, etc.). Finally, state where the mean curvature vanishes.

(a) The hyperbolic paraboloid: $z = xy$, $-2 \le x \le 2$, $-2 \le y \le 2$.

(b) The paraboloid: $z = x^2 + y^2$, $-2 \le x \le 2$, $-2 \le y \le 2$.

(c) The hemisphere: $z = \sqrt{1 - x^2 - y^2}$, $x^2 + y^2 \le 1$. (Hint: Since the function $\sqrt{1 - x^2 - y^2}$ is not defined on all of the square $-1 \le x, y \le 1$, for purposes of plotting you should replace it by the function **`sqrt(max(1 - x^2 -y^2, 0))`**.indexkfMax

Problem 6.3. Compute a formula for the Gaussian curvature at each point of the following quadric surfaces. First parametrize and view the surface, as in the examples in Chapter 6, *Geometry of Surfaces*. (Select the **`view`** carefully!) Discuss the sign of the Gaussian curvature and what it means. Point out places where the curvature is largest and smallest, any asymptotic behavior (i.e., what happens as you tend to infinity in some direction), and any symmetries.

(a) The ellipsoid: $x^2 + y^2 + \dfrac{z^2}{4} = 1$.

(b) The elliptic hyperboloid of one sheet: $x^2 + 2y^2 - z^2 = 1$.

(c) The hyperboloid of two sheets: $x^2 + y^2 - \frac{z^2}{4} = -1, z > 0$.

Problem 6.4. Sometimes, the formulas for Gaussian and mean curvature are complicated, and it is difficult to draw conclusions from them. In fact, even a MATLAB portrait of the surface may leave you a little puzzled about the way the surface is bending. In such a case, it can be useful to have MATLAB plot the functions K and H, even if their formulas are "messy." This may help you describe the nature of the original surface. Illustrate this fact in the following cases by: plotting the surface, computing the invariants K and H, and then plotting K and H separately in three-dimensional plots. (Choose a consistent **`view`** for the three plots in each case.) In some cases, you might also want to use **`fcontour`** to display the shape of the curves along which K or H vanishes. For each surface, do your best to use the plots of K and H to say something meaningful about the shape of the original surface. Specifically, describe the elliptic and hyperbolic regions of the surface, any curves on which the surface is parabolic, where the Gaussian curvature is "small," and where (in the elliptic regions) the surface is concave-up and where concave-down.

(a) A cubic surface: $z = x^3 - y^3, -1 \le x \le 1, -1 \le y \le 1$.

(b) A quartic surface: $z = (y - x^2)^2, -2 \le x \le 2, -1 \le y \le 2$.

(c) An exponential surface: $z = e^{x^2} - e^{-y^2}, -1 \le x \le 1, -2 \le y \le 2$.

Problem 6.5. The following surfaces have some sharp edges and vertices. Nevertheless, we can still use MATLAB to compute the curvature functions. Compute the Gaussian curvature for each surface, and then plot it as a function of the parameters. Observe where it is singular (i.e., where it is discontinuous, undefined, or blows up). Plot the surfaces and correlate the singular points you found to the plots. Is the curvature singular at *every* place on the surface where there is a sharp edge or vertex? Is the surface ever hyperbolic?
(**Note**: The curvature computations for these surfaces may take a few moments; be patient! Also, you may want to select choices for **`axis`** and **`view`** that are different from the defaults in order to improve your graphs. Finally, here (and in other problems as well), you will likely need to apply the **`simplify`** command to the output of your curvature computations. Moreover, you may need to utilize it in its most robust fashion, to wit **`simplify (ans, 'Steps', x)`**, where **`x`** is a number between 30 and 100.)

(a) The astroidal sphere:

$$\sigma(u,v) = (\cos^3 u \cos^3 v, \sin^3 u \cos^3 v, \sin^3 v).$$

(b) A figure eight surface:

$$\sigma(u,v) = (\cos u \cos v \sin v, \sin u \cos v \sin v, \sin v),$$

where in both examples, the parameter ranges can be selected to be $0 \le u \le 2\pi$, $-\pi/2 \le v \le \pi/2$.

Problem 6.6. You know that the Gaussian and mean curvatures are defined in terms of the two principal curvatures by the formulas

$$K = k_1 k_2, \qquad H = (k_1 + k_2)/2.$$

(a) Solve these formulas for k_1 and k_2 in terms of K and H.

(b) What can you say about the relation between H^2 and K?

(c) Give conditions (in terms of H and K) for exactly one of the principal curvatures to vanish, and for both to vanish.

(d) Here's an application of the result in (c). Consider the surface $z = x^3 + x^2 y - 2x + y^2$. Compute H and K. The results, while complicated, are manageable. For example, show that K vanishes along a curve whose projection into the x-y plane is a parabola. By applying **`solve`** to the simultaneous equations given by the vanishing of the numerators of H and of K, show that all simultaneous solutions of $H = 0$ and $K = 0$ are complex. Thus, there is no point on the surface where both principal curvatures vanish. On the other hand, try to compute the principal curvatures directly. You will see that the formulas are so complicated that they are basically useless.

Problem 6.7. In this problem, we study curvature for surfaces of revolution.

(a) For each of the following surfaces of revolution: plot the surface, compute its Gaussian curvature, and discuss the results. Use the examples in Chapter 6 as models.

(i) The hyperboloid of one sheet:

$$\sigma(u,v) = (\cosh u \cos v, \cosh u \sin v, \sinh u), \quad -1 \le u \le 1, \quad 0 \le v \le 2\pi.$$

(ii) A cubic surface:

$$\sigma(u,v) = (u \cos v, u \sin v, u^3 - 6u^2 + 9u - 2), \ 0.25 \le u \le 3.75, 0 \le v \le 2\pi.$$

(iii) An inverted bell surface:

$$\sigma(u,v) = (u \cos v, u \sin v, 1 - e^{-u^2}), \qquad 0 \le u \le 3, \quad 0 \le v \le 2\pi.$$

(b) Consider the surface of revolution determined by a profile curve $z = h(y)$ that does not intersect the z-axis:

$$\sigma(u,v) = (u\cos v, u \sin v, h(u)), \qquad u \ge \delta > 0, \quad 0 \le v \le 2\pi.$$

Compute its Gaussian curvature. Based on the result and on examples a(ii) and a(iii), explain why there are no points of singularity for K (assuming h has a continuous second derivative). Determine where K is positive, negative, and zero, purely in terms of the geometry of the curve h. (Hint: The max/min and inflection points of h should play a role in your discussion.) You may want to include a two-dimensional plot of the profile curves from a(ii) and a(iii).

Problem 6.8. In this problem, we study the notion of *umbilic* points. Recall that a point is called umbilic if its principal curvatures coincide there.

(a) Find a single equation relating K and H that must be satisfied at an umbilic point. Use it to explain why umbilic points can only occur where $K \geq 0$.

(b) At an umbilic point, the shape operator must be of the form $\begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix}$, where k is the unique principal curvature. Why? (Hint: Recall from the discussion in Chapter 6 that the shape operator can be written in the form

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} k_1 & 0 \\ 0 & k_2 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

for some θ. See what happens to this formula when $k_1 = k_2 = k$.) Deduce that at an umbilic point, the unique principal curvature k satisfies

$$e = kE, \qquad f = kF, \qquad g = kG.$$

(Hint: Use **`solve`**.)

(c) Use these facts to find all the umbilic points on the following surfaces:

(i) The saddle surface: $z = xy$.

(ii) The monkey saddle: $z = x^3 - 3xy^2$.

(iii) The elliptic paraboloid: $z = ax^2 + by^2$, $a, b > 0$.

(iv) The hyperbolic paraboloid: $z = ax^2 - by^2$, $a, b > 0$.

(Hint: You can use part (a) for (i) and (ii), but you will have to use part (b) to do (iii) and (iv). **`solve`** is useful.)

(d) Draw a picture of the elliptic paraboloid $z = 2x^2 + y^2$ with the umbilic points marked on it. You can include the umbilic points on the graph of the surface by numerically plotting them using **`plot3`**, but it is helpful to control the color of the point (make it black) and its size (use **`MarkerSize`**).

Problem 6.9. In this problem, we will study some surfaces with constant Gaussian curvature. We have already met several examples: a plane, a cone, a cylindrical surface, a sphere. In the first three $K \equiv 0$, the plane having both principal curvatures always vanishing, and the next two having exactly one principal curvature identically zero. The sphere has constant positive curvature. In this problem, we will find

more exotic examples of surfaces of revolution with constant Gaussian curvature, both positive and negative, defined by means of *elliptic integrals*.

(a) Consider a surface of revolution:

$$\sigma(u,v) = (g(u)\cos v, g(u)\sin v, h(u)).$$

Compute its curvature K in terms of the functions g and h and their derivatives.

(b) Now assume the profile curve $(g(u),h(u))$ is of unit speed, in which case we have $(g')^2 + (h')^2 \equiv 1$. Solve for h', differentiate, and substitute for $h'(t)$ and $h''(t)$ in the expression for K obtained in (a) to show that $K = -g''/g$.

(c) Now to construct a surface of revolution of constant curvature 1, we solve the differential equation $1 = -g''(u)/g(u)$. Solve it, assuming for simplicity $g(0) = a > 0$, $g'(0) = 0$.

(d) The function h then is given by $h(u) = \int_0^u \sqrt{1-g'(t)^2}\,dt$, which, given that you found the correct function g, is classically called an elliptic integral. Integrate to find the formula for h.

(e) Draw the three surfaces you get by choosing $a = 1$, $a = 0.5$, and $a = 1.5$. You should get a sphere, a football and a barrel, respectively. (You may use the interval $-\pi/2 \le u \le \pi/2$ for the first two plots, but you will have to shrink it for the third plot. Regarding v, the interval $0 \le v \le 2\pi$ works fine for all three. You will likely have to adjust **`axis`**, especially when $a = 0.5$ to see the football.)

(f) We can play a similar game to get a constantly negatively curved surface. Solve the differential equation $1 = g''/g$ with the same initial data, except choose $a = g(0) = 1$ for convenience. After obtaining g, compute h as in part (d). Well, it involves an antiderivative that MATLAB struggles with. You may find the substitution $\sinh(t) = -i\sin(it)$ useful. Now the answer is in terms of elliptic integrals again. Do not be swayed by the appearance of the imaginary number i; the functional values are actually real. One other point: you may have to enter the antiderivative as **`int(g(t))`** rather than **`int(g(t), 0, u)`** at the appropriate juncture. (MATLAB can be temperamental at times.) Draw the resulting surface. It is sometimes called the *bugle surface*. (Let u run from 0 to 1.5 in your graph.)

Problem 6.10. In this problem, we study minimal surfaces.

(a) Draw the helicoid

$$\sigma(u,v) = (u\cos v, u\sin v, v),$$

and show it is a minimal surface.

(b) Draw the catenoid (the surface of revolution spanned by the *catenary* $y = \cosh z$)

$$\sigma(u,v) = (\cosh u \cos v, \cosh u \sin v, u),$$

and show it is a minimal surface.

(c) There is a deformation of the catenoid into the helicoid through minimal surfaces. Consider the collection of surfaces, depending on a parameter t, defined by

$$\begin{aligned}\Theta(u,v,t) = &\cos t(\sinh u \sin v, -\sinh u \cos v, v)\\ &+\sin t(\cosh u \cos v, \cosh u \sin v, u).\end{aligned}$$

Check that for each t, $\Theta(u,v,t)$ parametrizes a minimal surface. Also, check that this surface is a helicoid when $t = 0$ and a catenoid when $t = \pi/2$. Successively draw the surfaces corresponding to the values $t = 0$, $\pi/10$, $\pi/5$, $3\pi/10$, $2\pi/5$, $\pi/2$ and watch the helicoid turn into a catenoid. (Suggestion: Use the ranges $-2 \le u \le 2$, $0 \le v \le 2\pi$.) If you wish, you can animate the result by making a "movie." Consult the online help for instruction on how to use the **`movie`** command.

(d) The catenoid is a surface of revolution; the helicoid is not. In fact, the only surfaces of revolution that are minimal are catenoids. Prove this fact as follows. Consider the surface of revolution given by a curve $y = h(z)$, that is, the surface

$$\sigma(u,v) = (h(u)\cos v, h(u)\sin v, u).$$

Compute the mean curvature of the surface in terms of h and its derivatives. Set it equal to zero and solve the differential equation. You will see that the profile curve $z = h(u)$ is a catenary.

(e) Finally, here are two famous examples of minimal surfaces. Draw them and verify that their mean curvatures are identically zero.

(i) Scherk's minimal surface: $e^z \cos x = \cos y$, or

$$\sigma(u,v) = \left(u, v, \ln \frac{\cos v}{\cos u}\right),$$

where $-\pi/2 + 0.01 \le u \le \pi/2 - 0.01, -\pi/2 + 0.01 \le u \le \pi/2 - 0.01$.

(ii) Enneper's minimal surface:

$$\sigma(u,v) = \left(u - \frac{u^3}{3} + uv^2, v - \frac{v^3}{3} + u^2 v, u^2 - v^2\right),$$

where $-2 \le u \le 2$, $-2 \le v \le 2$.

MATLAB's default **`view`** gives a nice picture in case (i). But try several choices (starting with the default) for your **`view`** in case (ii), until you can clearly see the "hole."

Glossary of MATLAB Commands and Options

`axis` Selects the ranges of x and y to show in a 2D-plot; or x, y and z in a 3D-plot

`cross` The cross product of two vectors

`det` Computes the determinant of a square matrix

`diff` Computes the derivative

`eig` Computes the eigenvalues of a square matrix

`fsurf` Easy 3D surface plotter

`int` Computes the integral

`max` Computes the maximum of the entries of a vector

`plot3` Plots points or lines in 3D

`real` Follows **`syms`** to insure variables are real

`realdot` Not a built-in MATLAB function. Computes the dot product of two vectors with real coordinates

`simplify` Performs algebraic simplification of a symbolic expression

`solve` Symbolic equation solver

`sqrt` Square root function

`subs` Substitute for a variable in an expression

`syms` Set up on or more symbolic variables

`trace` Sum of the diagonal elements of a matrix

`view` Specifies a point from which to view a 3D graph

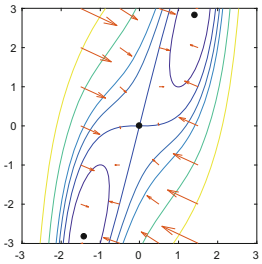
Options to MATLAB Commands

`Steps` Specifies the number of steps that **`simplify`** will use in its attempt to algebraically simplify an expression

References

1. A. Gray, *Modern Differential Geometry of Curves and Surfaces* (CRC Press, 1993)
2. B. O'Neill, *Elementary Differential Geometry*, revised 2nd edn. (Academic Press, Elsevier, 2006)

Chapter 7
Optimization in Several Variables



In this chapter, we will study generalizations to several variables of a few important applications of the derivative for a function f of one variable x. To make clearer the analogy between the one-variable and the several-variable cases, we begin with a quick review of the one-variable case, formulated in a way that will generalize easily.

7.1 The One-Variable Case

Suppose f is a *smooth* function defined on an interval of the real line. For our purposes, smooth will mean that f has a continuous second derivative. (The interval of definition may be finite, or it may be a half-line, or it may be the whole line.) We say f has a *local maximum* or *relative maximum* at $x = x_0$ if $f(x_0) \geq f(x)$ for all x close to x_0, and we say f has a *local minimum* or *relative minimum* at $x = x_0$ if $f(x_0) \leq f(x)$ for all x close to x_0. The word *extremum* means either a maximum or a minimum. Many real-world problems boil down to the mathematical problem of locating extrema. So how do we find the local extrema of a function f? There are two ways to do this:

1. *analytically*, by an exact calculation from the formula for the function f; or
2. *numerically*, by a numerical calculation from data associated with f. Often, numerical methods are augmented by a graphical procedure.

Calculus helps with both methods.

7.1.1 Analytic Methods

In pursuing an analytic method, we usually use one of the basic principles of one-variable calculus: *A (local) extremum always occurs at a critical point*, that is, at a point where the derivative f' of f vanishes. Actually, this is only a necessary

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_7

condition: $f'(x_0)$ necessarily vanishes if $f(x)$ has a local extremum at $x = x_0$, but it can happen that a critical point is not a local extremum. (For example, think of the function $f(x) = x^3$ at $x = 0$.) Nevertheless, for the (smooth) functions we typically encounter, there are usually not many critical points, and it's often easy to tell by inspection which are local maxima, which are local minima, and which are neither. In cases where inspection does not work, there is always the *second derivative test*. If $f'(x_0) = 0$ and $f''(x_0) > 0$, then f has a local minimum at $x = x_0$, and if $f'(x_0) = 0$ and $f''(x_0) < 0$, then f has a local maximum at $x = x_0$. In fact, when $f'(x_0) = 0$ and $f''(x_0) > 0$, then f has a *strict local minimum* at $x = x_0$, in the sense that $f(x)$ is strictly bigger than $f(x_0)$ for x close to, but not equal to, x_0. Similarly, when $f'(x_0) = 0$ and $f''(x_0) < 0$, then f has a *strict local maximum* at $x = x_0$, in the sense that $f(x)$ is strictly smaller than $f(x_0)$ for x close to, but not equal to, x_0. Of course, it can happen that $f'(x_0) = 0$ and $f''(x_0) = 0$, but this behavior is *nongeneric*; i.e., it doesn't happen for typical functions. When $f'(x_0) = 0$ and $f''(x_0) = 0$, the second derivative test fails, but we can get more information from the third derivative, and so on.

7.1.2 Numerical Methods

Sometimes a function may not come with an explicit formula, or the formula may be too complicated to make it possible to solve directly for the critical points. In this case we can still locate local extrema numerically or graphically. Calculus can help us with this task in several ways. If the derivative f' is computable, but the equation $f'(x) = 0$ is too complicated to solve analytically, then we can use numerical root-finding procedures to locate numerically the solutions to $f'(x) = 0$. The most important such method, derived from calculus, is *Newton's method*.[1] It is implemented in MATLAB via the `fzero` command.[2] We will return to this in a moment.

Alternatively, we can use calculus to formulate an algorithm to hunt for local extrema, say local minima. This method is based on the familiar fact that when $f'(x_0) > 0$, then f is *increasing* at $x = x_0$, and when $f'(x_0) < 0$, then f is *decreasing* at $x = x_0$. Thus, suppose we start at some point $x = x_0$ where $f'(x_0) > 0$. (We might do this by explicit calculation of the derivative, by numerical estimation, or by looking at the graph of f.) Then f is *increasing* near $x = x_0$. So, if we want f to *decrease*, we have to move in the direction of *decreasing* x, that is, to the *left*. So we move some *step size* h to the left and look at $x_1 = x_0 - h$. If $f'(x_1) > 0$, then f is still increasing at x_1, so look at $x_2 = x_1 - h$, and so on. This process will not always converge (it certainly will not if f does not have a local minimum), but

[1] What one sees in all calculus books is a slight variant of Newton's original scheme, called the *Newton–Raphson method*.

[2] Actually, this is a white lie. The `fzero` algorithm doesn't assume f is differentiable and, even if f is differentiable, doesn't assume the derivative is known. Instead, the way the algorithm proceeds, it tries first to locate the zero approximately, and then it zooms in on it by an interpolation method that closely approximates Newton's method.

sometimes it will happen that eventually we come to a point x_n where $f'(x_n) > 0$ and $f'(x_{n+1}) < 0$ (with $x_{n+1} = x_n - h$). This means (since we are assuming f has a continuous derivative) that f' changes sign between $x = x_{n+1}$ and $x = x_n$. Hence, there is a local minimum between them. We can get a better approximation to this local minimum by repeating the same process with a smaller step size. Of course, if at some point x_i we have $f'(x_i) < 0$, then f is *decreasing* near $x = x_i$, so in this case we move to the *right*. We have just described a numerical algorithm for searching for a local minimum point. The algorithm has the advantage that it works even when we cannot differentiate the formula for f analytically.

A slightly fancier version of this procedure is implemented in MATLAB via the command **fminsearch**, whose first argument has to be a function. For example, the MATLAB command

```
>> fminsearch(@(x) x^4 - 4*x^2, 1)
```

searches for a minimum of the function $x^4 - 4x^2$, starting at $x = 1$. This command produces the output:

```
ans =
   1.4143
```

This reflects the fact that $x^4 - 4x^2$ has a minimum value of -4 when $x = \sqrt{2} \approx 1.41421$. If instead we wanted to find a local *maximum* for f, we could simply apply the same algorithm to the function $-f$, since f has a local maximum where $-f$ has a local minimum.

7.1.3 Newton's Method

Newton's method is a process for solving nonlinear equations numerically. We might want to do this for reasons that have nothing to do with finding local extrema, but nevertheless the topics of equation-solving and *optimization* (looking for extrema) are closely linked. As we have already seen, looking for a local extremum of f forces us to try to solve the equation $f'(x) = 0$. We can also go in the other direction. If we want to solve an equation $g(x) = 0$, one way to do this is to look at the function $f(x) = g(x)^2$. The function f is nonnegative, so the smallest it could ever be is 0, and f takes the value 0 exactly where the equation $g(x) = 0$ is satisfied. So the solutions of $g(x) = 0$ occur at local minimum points of f.

To explain Newton's method for solving equations, let us take a simple example. Suppose we want to solve an equation such as $e^x = 3x$. We start by formulating an initial guess as to where a solution might be found. Since $e^1 = 2.7\ldots$ and $3 \cdot 1 = 3$, it looks as if a good starting guess might be $x_0 = 1$. Next, we rewrite the equation in the form $g(x) = 0$. In this case we would take $g(x) = e^x - 3x$. Near $x = x_0$, we have, by the *tangent line approximation*, the estimate $g(x) \approx g(x_0) + (x - x_0)g'(x_0)$. Setting this equal to zero, we get a linear equation for a (presumably) better approximation x_1 to a solution. In other words, solving

$$g(x_0) + (x_1 - x_0)g'(x_0) = 0,$$

we get

$$x_1 = x_0 + (x_1 - x_0) = x_0 - \frac{g(x_0)}{g'(x_0)}.$$

Then we take x_1 as our new guess for a solution and compute an even better guess

$$x_2 = x_1 - \frac{g(x_1)}{g'(x_1)},$$

and so on. The process does not always converge, but it usually does if the starting value x_0 is close enough to a solution. In this case, the convergence to a true solution is usually quite fast. For example, in the given example, we have $g'(x) = e^x - 3$, so

$$\begin{aligned}
x_1 &= 1 - \frac{e-3}{e-3} = 1 - 1 = 0,\\
x_2 &= 0 - \frac{1-0}{1-3} = \frac{1}{2},\\
x_3 &= \frac{1}{2} - \frac{e^{.5} - 1.5}{e^{.5} - 3} = 0.61005965,\\
x_4 &= 0.61005965 - \frac{e^{0.61005965} - 3 \cdot 0.61005965}{e^{0.61005965} - 3} = 0.61899678,\\
x_5 &= 0.61899678 - \frac{e^{0.61899678} - 3 \cdot 0.61899678}{e^{0.61899678} - 3} = 0.61906128.
\end{aligned}$$

In fact, a similar process is automated in MATLAB's **fzero** command.

```
>>  format long; fzero(@(x) exp(x) - 3*x, 1)

ans =
   0.619061286735945
```

This tells us that, up to 15-digit accuracy, the number 0.619061286735945 satisfies the equation $e^x = 3x$.

One caution about the method: it tells you nothing about how many solutions there are. To find another solution (if there is one), you need to repeat the process with a different starting guess. For example, in this case, there is another solution, as you can see from Figure 7.1, which is generated by the commands

```
>> fplot(@exp, [0, 2]); hold on
>> fplot(@(x) 3*x, [0, 2]); hold off
```

The graph suggests using $x = 1.5$ as a second starting point. And indeed

```
>>  fzero(@(x) exp(x) - 3*x, 1.5)

ans =
   1.512134551657843
```

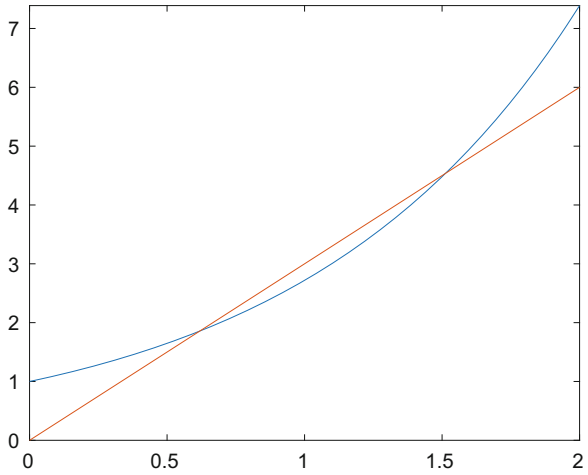


Fig. 7.1 The curves $y = e^x$ and $y = 3x$

Note that the default in MATLAB is to give the answer to about the maximal degree of precision possible using double-precision arithmetic. But you can adjust the degree of precision by using **`optimset`** to adjust the options of **`fzero`**, notably **`TolX`**.

7.2 Functions of Two Variables

Now consider a function f of two variables x, y, defined in a suitable domain of the x-y plane. We assume our function is *smooth*. As with one variable, this means that f has continuous second partial derivatives. As before, we say f has a *local maximum* or *relative maximum* at (x_0, y_0) if $f(x_0, y_0) \geq f(x, y)$ for all (x, y) close to (x_0, y_0), and we say f has a *local minimum* or *relative minimum* at (x_0, y_0) if $f(x_0, y_0) \leq f(x, y)$ for all (x, y) close to (x_0, y_0). We say f has a *strict local maximum* or *strict relative maximum* at (x_0, y_0) if $f(x_0, y_0) > f(x, y)$ for all (x, y) close to, but not equal to, (x_0, y_0), and we say f has a *strict local minimum* or *strict relative minimum* at (x_0, y_0) if $f(x_0, y_0) < f(x, y)$ for all (x, y) close to, but not equal to, (x_0, y_0). Again, the word *extremum* means either a maximum or a minimum.

We still have the basic principle that *a (local) extremum always occurs at a critical point*, but now a critical point is a point where *both* partial derivatives vanish, in other words, where the *gradient vector* ∇f vanishes. (See Chapter 5.) In searching for local extrema, we can still proceed either analytically or numerically. Searching for analytic solutions is more complicated than in the one-variable case, since we need to solve two simultaneous equations in two unknowns. Sometimes the MATLAB routine **`solve`** can help with the algebra. As in the one-variable case,

it remains true that not every critical point is a local extremum. But it may be harder in the two-variable case to tell by inspection whether a critical point is a local maximum, a local minimum, or neither. Again, we can use a *second derivative test*.

7.2.1 Second Derivative Test

Before we get to the second derivative test itself, it is useful to make a slight digression to say a bit about the multivariable version of Taylor's Theorem. Recall that for a smooth function f of one variable, Taylor's Theorem says that

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \cdots + \frac{1}{n!}f^{(n)}(x_0)(x - x_0)^n, \tag{7.1}$$

with an error that can be estimated as

$$\text{error} = \frac{1}{(n+1)!}f^{(n+1)}(x_1)(x - x_0)^{n+1},$$

where x_1 lies between x_0 and x. For now we are not so much worried about the error estimate as about a simple principle: when x_0 is a critical point (so that the linear term in (7.1) drops out), then the first nonzero higher derivative of f at x_0 determines the local behavior close enough to the critical point. Just as an example, suppose $f(x) = x^3 - 3x^2 + 3x$ and $x_0 = 1$. Then $f'(x) = 3x^2 - 6x + 3$, which vanishes at $x_0 = 1$, so x_0 is a critical point. Furthermore, $f''(x) = 6x - 6$, which also vanishes at $x = 1$, so the second derivative test fails at this point. Moreover, $f'''(x) = 6 = 3! > 0$ and the Taylor series for f at $x_0 = 1$ becomes $f(x) = 1 + (x - 1)^3$. Thus behavior near $x = x_0$ is dominated by the cubic term in (7.1), which is increasing.

Now suppose f is a smooth function of two variables. We now have multiple partial derivatives of each order and so (7.1) becomes more complicated. However, what is still true is that f can be approximated by polynomials (in both x and y), and the coefficient of any monomial $(x - x_0)^j(y - y_0)^k$ involves the higher partial derivative $\frac{\partial^{j+k} f}{\partial x^j \partial y^k}(x_0, y_0)$. The best approximation of f by a polynomial of total order n turns out to be

$$\begin{aligned} f(x) &\approx \sum_{j+k \le n} \frac{1}{j!k!} \frac{\partial^{j+k} f}{\partial x^j \partial y^k}(x_0, y_0)\,(x - x_0)^j (y - y_0)^k \\ &= f(x_0, y_0) + \frac{\partial f}{\partial x}(x_0, y_0)\,(x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0)\,(y - y_0) \\ &+ \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(x_0, y_0)\,(x - x_0)^2 + \frac{1}{2}\frac{\partial^2 f}{\partial y^2}(x_0, y_0)\,(y - y_0)^2 \\ &+ \frac{\partial^2 f}{\partial x \partial y}(x_0, y_0)\,(x - x_0)(y - y_0) + \cdots . \end{aligned} \tag{7.2}$$

The simplest way to verify (7.2) is to first observe that for reasons of symmetry, we can assume that $x_0 = y_0 = 0$. In this case, if f is a polynomial containing a term of the form $c\,x^j\,y^k$, then we can recover the c from the fact that $\frac{\partial^{j+k} f}{\partial x^j \partial y^k}$, when evaluated at the origin, will be exactly $c\,j!\,k!$. Thus, after dividing by $j!\,k!$, we recover the coefficient c.

Next, consider in detail the quadratic terms in (7.2). There are three of them, a term which is a constant times $(x - x_0)^2$, one which is a constant times $(y - y_0)^2$, and one which is a multiple of $(x - x_0)(y - y_0)$. We can use matrix multiplication to see that the quadratic terms in (7.2) can be collapsed into the simple form

$$\frac{1}{2}\begin{pmatrix} x - x_0 & y - y_0 \end{pmatrix} A \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix}, \tag{7.3}$$

where A is a 2×2 matrix, obtained by evaluating the *Hessian matrix*

$$\text{Hess}\, f = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial y \partial x} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

at the point $x_0,\, y_0$. The Hessian is a *symmetric matrix*; i.e., the entry in the upper right equals that in the lower left, since the two mixed second partial derivatives of a smooth function are equal to one another. If $\nabla f(x_0,\, y_0) = \mathbf{0}$, so that $(x_0,\, y_0)$ is a critical point for f, then (7.3) is the first term in (7.2) after the constant term $f(x_0,\, y_0)$, and we will see shortly how it determines the behavior of f near the point $(x_0,\, y_0)$.

Now an important theorem from linear algebra says that a symmetric matrix can be *diagonalized*. What this means is that one can make a linear change of variables, in effect introducing new coordinates $x' = ax + by$, $y' = cx + dy$, so that in the new coordinates, the matrix takes the simple form $\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$, for some constants λ_1 and λ_2 called the *eigenvalues* of the matrix. Eigenvalues can be computed very simply in MATLAB using the command **`eig`**. When we diagonalize the Hessian, that tells us that (7.2) near a critical point can be rewritten in the new coordinates as

$$f(x', y') = f(x'_0, y'_0) + \frac{\lambda_1}{2}(x' - x'_0)^2 + \frac{\lambda_2}{2}(y' - y'_0)^2 + \text{higher order terms}, \tag{7.4}$$

where λ_1 and λ_2 are the eigenvalues of the Hessian at the critical point.

A consequence is that if $\lambda_1,\, \lambda_2 > 0$, then f is increasing in all directions away from the critical point, and thus the critical point is a *strict local minimum*. Similarly, if $\lambda_1,\, \lambda_2 < 0$, then f is decreasing is all directions away from the critical point, and thus the critical point is a *strict local maximum*. This is the first version of the second derivative test for functions of two variables.

The usual way of formulating the second derivative test is a bit complicated. It involves the determinant of a 2×2 matrix, that you might have learned about in high school algebra:

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc.$$

This is the product $\lambda_1 \lambda_2$ of the two eigenvalues, so it will be positive exactly when the eigenvalues are nonzero and have the same sign. The determinant of the Hessian matrix is called the *discriminant* of f. The test says:

1. If $\nabla f(x_0, y_0) = \mathbf{0}$ and $\det \operatorname{Hess} f > 0$ at (x_0, y_0), then f has a strict local extremum at (x_0, y_0). (This follows from the above discussion, i.e., from (7.4) together with the fact that $\det \operatorname{Hess} f = \lambda_1 \lambda_2 > 0$.) If $\partial^2 f/\partial x^2$ and $\partial^2 f/\partial y^2$ are both positive at (x_0, y_0), then f has a strict local *minimum* at (x_0, y_0). If $\partial^2 f/\partial x^2$ and $\partial^2 f/\partial y^2$ are both negative at (x_0, y_0), then f has a (strict) local *maximum* at (x_0, y_0). (Positivity of the determinant of a symmetric 2×2 matrix guarantees that the entries in the upper left and lower right corners are nonzero and have the same sign. Can you see why?)
2. If $\nabla f(x_0, y_0) = \mathbf{0}$ and $\det \operatorname{Hess} f < 0$ at (x_0, y_0), then f does *not* have a local extremum at (x_0, y_0). Instead we say that f has a *saddle point*. This can be seen from the fact that λ_1 and λ_2 must be nonzero with opposite sign. Thus in a suitable coordinate system, the graph of f along one axis locally looks like a parabola opening upwards, and along the other axis locally looks like a parabola opening downwards. At a saddle point, f is increasing in some directions, and decreasing in others. The central point of a riding saddle has this property—whence the name. A nice way to visualize this behavior is with **`fcontour`**. At a saddle point, there will be two level curves intersecting as in the following example, shown in Figure 7.2.

```
>> syms x y real; fcontour(x^2 - y^2, [-3,3,-3,3], ...
     'LevelList', -3:3)
```

2. When $\nabla f(x_0, y_0) = \mathbf{0}$ and $\det \operatorname{Hess} f = 0$, the second derivative test is inconclusive. There may or may not be a local extremum at (x_0, y_0), and if there is a local extremum, it may not be strict. The reason is that at least one of the eigenvalues of the Hessian vanishes at the critical point. Thus in the corresponding direction, the quadratic term in (7.4) vanishes and local behavior of the function is determined by terms of third (or higher) degree. Fortunately, this behavior is *nongeneric*; i.e., it doesn't happen for "typical" functions. But this situation does happen for the "monkey saddle," as you will see in Problem 7.8.

7.2.2 Steepest Descent

Consider the algorithm we discussed above to locate a local minimum of a function f. In the one-variable case, the idea was to move in the direction in which the function is decreasing. In the two-variable case, however, there are infinitely many directions in which we can possibly move, not just two. The idea then is to move in the direction in which the function is decreasing the fastest. What direction is that?

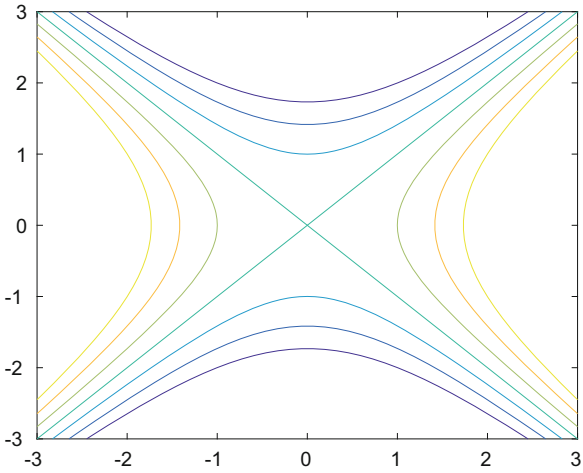


Fig. 7.2 Level curves near a saddle point

Well, recall from Chapter 5 that the directional derivative of f in the direction of a unit vector $\mathbf{u}$ is defined by

$$D_{\mathbf{u}} f(\mathbf{x}_0) = \lim_{t \to 0} \frac{f(\mathbf{x}_0 + t\mathbf{u}) - f(\mathbf{x}_0)}{t},$$

and is computed by the formula

$$D_{\mathbf{u}} f(\mathbf{x}_0) = \nabla f(\mathbf{x}_0) \cdot \mathbf{u}.$$

The directional derivative tells us how fast f is increasing in the direction of the unit vector $\mathbf{u}$. So to find the direction in which the function is decreasing the fastest, we need to make $D_{\mathbf{u}} f(\mathbf{x}_0)$ as negative as possible. If $\mathbf{u}$ is a unit vector, then

$$|D_{\mathbf{u}} f(\mathbf{x}_0)| = |\nabla f(\mathbf{x}_0) \cdot \mathbf{u}| \leq \|\nabla f(\mathbf{x}_0)\| \, \|\mathbf{u}\| = \|\nabla f(\mathbf{x}_0)\|,$$

so the minimum possible value of $D_{\mathbf{u}} f(\mathbf{x}_0)$ is precisely $-\|\nabla f(\mathbf{x}_0)\|$, and this minimum is achieved when $\mathbf{u}$ points in the direction of $-\nabla f(\mathbf{x}_0)$. Thus, we have arrived at the idea of the *method of steepest descent*: to locate a local minimum of a function of two (or more) variables, *move in the direction of the negative of the gradient*. With this modification, the method we outlined for the one-variable case works just fine. A version of it is implemented in the MATLAB routine **`fminsearch`**.[3] For example,

[3] Again, we have oversimplified. Steepest descent, using a formula for the gradient, is in fact implemented in the routine **`fminunc`**, part of the Optimization Toolbox. The algorithm for **`fminsearch`** is an iterative search method that does not use the gradient. The difference between the two methods is explored in Problem 7.9.

```
>> fminsearch(@(x) x(1)^4 - 4*x(1)*x(2) + x(2)^2, [1, 1])
ans =
   1.4142    2.8284
```

Here MATLAB searched for a minimum of the function $f(\mathbf{x}) = x_1^4 - 4x_1x_2 + x_2^2$, starting at $(1.0, 1.0)$, and returned the local minimum at the point $\mathbf{x} = (1.41421, 2.82843)$. For this particular example, it is not so hard to see that there are three critical points, located at $(0, 0)$ and at $\pm(\sqrt{2}, 2\sqrt{2})$. We can better understand how steepest descent works for f by plotting the three critical points, some contour lines for f, and the gradient field of $-f$, using the methods of Chapter 5. (The "minus sign" arises since it is the gradient of $-f$ that tells us in which direction f is decreasing the fastest.)

```
>> a = sqrt(2); fcontour(x^4 - 4*x*y + y^2, [-3,3,-3,3], ...
    'LevelList', [-4:2:4, 10, 20], 'MeshDensity',120), hold on
>> plot([-a, 0, a], [-2*a, 0, 2*a], 'ok', 'MarkerFaceColor', 'k')
>> gradf = -jacobian(x^4 - 4*x*y + y^2, [x, y])
>> [xx, yy] = meshgrid(-2:0.25:2, -3:0.5:3);
>> u = -4*xx.^3 + 4*yy; v = -2*yy + 4*xx;
>> ll = sqrt(u.^2 + v.^2);
>> quiver(xx, yy, u./ll, v./ll, 0.3)
>> hold off, axis([-2.2 2.2 -3.1 3.1])
```

The figure generated is shown in Figure 7.3. The steepest descent algorithm will

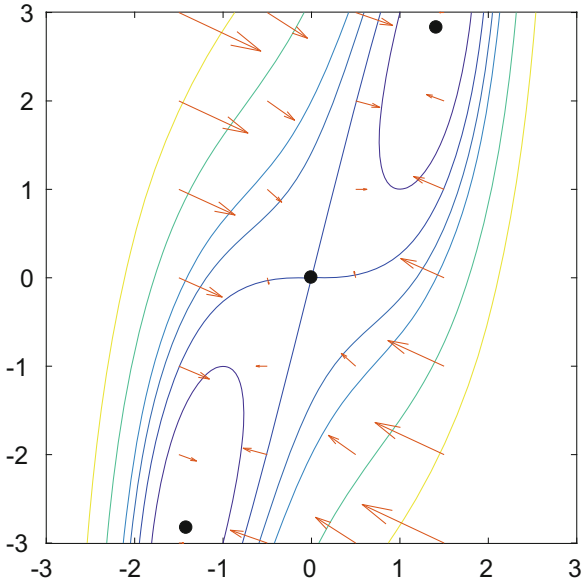


Fig. 7.3 A function with three critical points

follow the arrows, crossing a number of level curves of f, until it terminates at one of the critical points marked by a solid dot. Note that a saddle point such as the one at the origin in Figure 7.3 is an *unstable* critical point for the method, in that roundoff errors will usually prevent landing on the saddle point unless we began there in the first place.

To search for a local maximum of f, apply the steepest descent algorithm to $-f$. Just as in the one-variable case, there is no guarantee that the steepest descent method will converge, and there is no way to be sure we have found *all* the local minima this way.

7.2.3 Multivariable Newton's Method

Since a single equation in two unknowns will usually have a whole curve of solutions (see the discussion of the Implicit Function Theorem in Chapter 6), the correct analog of the one-variable procedure is to solve not one equation but a system of two equations in two unknowns. We also need to do this to locate critical points of a function f, since they occur where both $\partial f/\partial x$ and $\partial f/\partial y$ are equal to 0. We use the language of vectors, suggested by the combination of $\partial f/\partial x$ and $\partial f/\partial y$ into a single vector function, ∇f. Let

$$\mathbf{f}(x, y) = (f_1(x, y), f_2(x, y)),$$

where f_1 and f_2 are both real-valued functions of x and y. Suppose we want to solve the vector equation

$$\mathbf{f}(x, y) = \mathbf{0}.$$

As before, we need an initial guess for a solution; call it $\mathbf{x}_0 = (x_0, y_0)$. In the one-variable Newton's method, we used the tangent line approximation to a function. In the two-variable case, we use the *tangent plane approximation*. Thus for $\mathbf{x}$ close to $\mathbf{x}_0$, we have

$$\mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_0) + \big(\nabla f_1(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0), \nabla f_2(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0)\big).$$

Since we want to solve $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, it is natural to try solving the approximation

$$\mathbf{f}(\mathbf{x}_0) + \big(\nabla f_1(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0), \nabla f_2(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0)\big) = \mathbf{0}.$$

In other words, we must solve the pair of simultaneous linear equations

$$\begin{aligned} f_1(\mathbf{x}_0) + \nabla f_1(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) &= 0, \\ f_2(\mathbf{x}_0) + \nabla f_2(\mathbf{x}_0) \cdot (\mathbf{x} - \mathbf{x}_0) &= 0. \end{aligned}$$

There is always the chance that this system might be *singular*, that is, that it may not have a unique solution. But generically (i.e., if the starting value $\mathbf{x}_0$ and the function $\mathbf{f}$ are "typical"), this won't happen, so we get a unique solution $\mathbf{x}_1$. We use this as our next approximation to a solution and try solving

$$\begin{cases} f_1(\mathbf{x}_1) + \nabla f_1(\mathbf{x}_1) \cdot (\mathbf{x} - \mathbf{x}_1) = 0, \\ f_2(\mathbf{x}_1) + \nabla f_2(\mathbf{x}_1) \cdot (\mathbf{x} - \mathbf{x}_1) = 0, \end{cases}$$

to get $\mathbf{x}_2$, and so on. The process doesn't always converge, but when it does, the convergence is usually quite rapid. The multivariable Newton's method is not implemented as a built-in function in MATLAB, though it is easy enough to write a program for this purpose (see Problem 7.5). If you have access to the Optimization Toolbox, then the MATLAB command **`fsolve`** will accomplish something similar. For example, the following solves for a solution of the system

$$\begin{cases} e^x + e^y = 6, \\ x + y = 2. \end{cases}$$

```
>>  fsolve(@(x) [exp(x(1))+exp(x(2))-6, x(1)+x(2)-2], [0, 1])
ans =
   0.5486    1.4514
```

Without using the Optimization Toolbox, we could have also solved the same system by solving for y in the second equation to get $y = 2 - x$, substituting back into the first equation, and using

```
>>  xsol = fzero(@(x) exp(x) + exp(2-x) - 6, 0)
ans =
   0.5486

>>  ysol = 2 - xsol
ans =
   1.4514
```

7.3 Three or More Variables

Everything we have said also works for functions of three of more variables, with obvious modifications. The only thing that's a bit different is the second derivative test for critical points. If f is a function of three variables x, y, z, the local extrema still occur at critical points, that is, at points where the gradient

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$

vanishes as a vector. To apply the second derivative test at such a critical point $\mathbf{x}_0 = (x_0, y_0, z_0)$, we again look at the symmetric matrix of second partial derivatives

$$\operatorname{Hess} f = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial z \partial x} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial z \partial y} \\ \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial z^2} \end{pmatrix}.$$

This time, the matrix contains not three, but six independent functions. (The three entries below the diagonal duplicate those above.) So the criteria for local maxima and minima are correspondingly more complicated than in the two-variable case. Still, just as in the two-variable case, the Hessian can be diagonalized and this time has *three* eigenvalues $\lambda_1, \lambda_2, \lambda_3$. The product of these is again equal to the determinant of the matrix, and the analog of (7.4) is that after a linear change of coordinates, the function near a critical point is locally given by

$$f(x', y', z') = f(x'_0, y'_0, z'_0) + \frac{\lambda_1}{2}(x' - x'_0)^2 + \frac{\lambda_2}{2}(y' - y'_0)^2 + \frac{\lambda_3}{2}(z' - z'_0)^2 + \text{higher order terms.} \quad (7.5)$$

The best way to formulate the criteria is to say that $\mathbf{x}_0$ gives a strict local minimum if the Hessian matrix is *positive definite* at $\mathbf{x}_0$, and that $\mathbf{x}_0$ gives a strict local maximum if the *negative* of the Hessian matrix is positive definite at $\mathbf{x}_0$. In this formulation, the criteria even work for functions of four or more variables. The term positive definite is defined in linear algebra courses, and is equivalent to positivity of *all* of the eigenvalues. For present purposes, we can use the following working definition: a symmetric $n \times n$ matrix is called *positive definite* if all its *principal minors* are positive. The principal minors are the determinants of the submatrices obtained by deleting the last i rows and i columns, for $i = n-1, n-2, \ldots, 0$. Thus the principal minors of the Hessian matrix of a function of three variables are the determinants

$$\frac{\partial^2 f}{\partial x^2}, \quad \det \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial y \partial x} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}, \quad \det \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial z \partial x} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial z \partial y} \\ \frac{\partial^2 f}{\partial x \partial z} & \frac{\partial^2 f}{\partial y \partial z} & \frac{\partial^2 f}{\partial z^2} \end{pmatrix}.$$

When all three of these quantities are positive, we have a strict local minimum point. If the determinant of the Hessian is positive but either of the first two principal minors is nonpositive, then the critical point is definitely not a local minimum. (It will be a higher-dimensional saddle point.) If the determinant of the Hessian van-

ishes, the test is inconclusive. In general, the second derivative test for a local minimum of a function of n variables at a critical point involves positivity of n different functions of the second partial derivatives.

7.4 Constrained Optimization and Lagrange Multipliers

Sometimes in applications, one needs to find the local extrema of a function not on a region of the plane or 3-space but subject to a *constraint*. For example, one might need to find the maximum or minimum of a function $f(x,y)$ restricted to a curve C in the x-y plane. To solve a problem like this, two methods are available:

1. eliminate the constraint and thereby reduce the number of variables by one, or else
2. use the method of *Lagrange multipliers*.

We will explain how these work with a simple problem that one can do analytically with the first method and then numerically with the second method using MATLAB.

Suppose C is the circle $x^2+y^2=4$ and we want to find the extrema of $f(x,y)=5xy-y^2$ along C. To follow the first approach, parametrize C via

$$(x,y)=(2\cos\theta,2\sin\theta).$$

Substitute this back into f and we get a function

$$h(\theta)=f(x(\theta),y(\theta))=20\cos\theta\sin\theta-4\sin^2\theta=10\sin(2\theta)+2\cos(2\theta)-2.$$

Computing the derivative, we get $h'(\theta)=20\cos(2\theta)-4\sin(2\theta)$ and this vanishes when $\tan(2\theta)=5$, $\theta=\frac{\arctan(5)+n\pi}{2}$. Thus there are four constrained critical points along the circle, two of which are maxima and two of which are minima.

The method of Lagrange multipliers instead depends on a geometric principle. If we are given a function $f(x,y)$ to optimize along a constraint curve C: $g(x,y)=c$, most level curves of f will cross C transversally. At a point where $f(x,y)=a$ cuts across C transversally, ∇f will have a nonzero component pointing along C. Moving in this direction along C causes f to increase, and moving in the opposite direction along C causes f to decrease, Thus we cannot be at either a constrained local maximum or a constrained local minimum. The upshot of this analysis is that a constrained local extremum will only occur when ∇f is perpendicular to C. Since the tangent direction to C is perpendicular to ∇g, that means we need to have $\nabla f=\lambda\nabla g$, where λ is some constant of proportionality, called a Lagrange multiplier. Thus we get the system of equations

$$\begin{cases} \dfrac{\partial f}{\partial x} = \lambda \dfrac{\partial g}{\partial x}, \\ \dfrac{\partial f}{\partial y} = \lambda \dfrac{\partial g}{\partial y}, \\ g(x,y) = c. \end{cases} \tag{7.6}$$

This is a system of three simultaneous equations in the three variables x, y, and λ, the third of which is the constraint equation, which does not involve λ. In fact, one can always eliminate λ from the first two equations to get the system

$$\begin{cases} \dfrac{\partial f}{\partial x}\dfrac{\partial g}{\partial y} = \dfrac{\partial g}{\partial x}\dfrac{\partial f}{\partial y}, \\ g(x,y) = c. \end{cases} \tag{7.7}$$

These equations are rather complicated, but can be solved graphically or numerically using the 2-dimensional Newton's method of Section 7.2.3.

We'll apply this method to our sample problem, and also show that the gradient field of the objective function f is normal to the curve C at the extrema. First we compute the gradients of f and g.

```
>>  f = 5*x*y - y^2; g = x^2 + y^2; [jacobian(f), jacobian(g)]
```

```
ans =
[ 5*y, 5*x - 2*y, 2*x, 2*y]
```

Now we plot on the same axes the constraint curve $g = 4$, the gradient field of f, and the Lagrange curve $f_x g_y - f_y g_x = 0$. The code is given below and the result is Figure 7.4. Note that the intersection of the constraint curve with the Lagrange curve consists of four points spaced around the circle. These are indicated with solid dots in the picture; we know their exact coordinates because of our analytic solution. Also, note that the gradient field of f is normal to the constraint curve at these four points.

```
>> fimplicit(g-4,[-2.1,2.1,-2.1,2.1],'k','LineWidth',2), hold on
>> [xx,yy] = meshgrid(-2:.5:2,-2:.5:2);
>> quiver(xx,yy,5*yy,5*xx - 2*yy), axis equal
>> lagr = (5*y)*(2*y) - (5*x - 2*y)*(2*x);
>> fimplicit(lagr,[-2.1,2.1,-2.1,2.1],'r')
>> x1 = 2*cos(atan(5)/2); y1 = 2*sin(atan(5)/2);
>> plot([x1,-y1,-x1,y1],[y1,x1,-y1,-x1],'o','MarkerFaceColor','k')
```

The Lagrange multiplier method can also be applied to the problem of finding constrained extrema of a function $f(x,y,z)$ of three variables, with a constraint *surface* $g(x,y,z) = c$. In this case, the Lagrange multiplier equations are still $\nabla f = \lambda \nabla g$ and $g = c$, but these amount to *four* scalar equations in the four variables x, y, z, λ. One can again eliminate λ from the first three equations to get the system

$$\begin{cases} f_x g_y - f_y g_x = 0, \\ f_x g_z - f_z g_x = 0, \\ g = c. \end{cases}$$

Nonlinear systems like this are very difficult to solve by hand, but MATLAB can be used effectively to find numerical solutions.

Finally, the Lagrange multiplier method also applies to problems with two constraints, as when one wants to minimize a function f of three variables along a curve C in 3-space given by two equations $g(x, y, z) = c$ and $h(x, y, z) = d$. In this case there are two Lagrange multipliers λ and μ, and in vector form, the Lagrange multiplier equations take the form

$$\nabla f = \lambda \nabla g + \mu \nabla h.$$

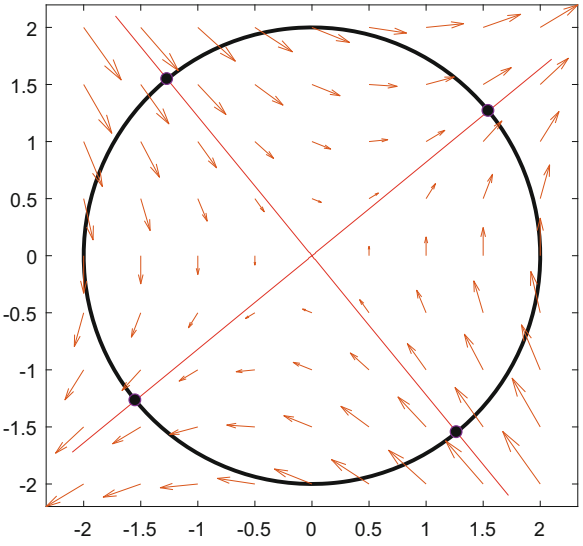


Fig. 7.4 Constrained extrema

These are really three scalar equations in the variables x, y, z, λ, μ, and when we add in the two constraint equations, we have five equations in five unknowns.

Problem Set G. Optimization

Problem 7.1. Let $f(x, y) = x^4 - 3xy + 2y^2$.

(a) Compute the partial derivatives of f as well as its discriminant. Then use **`solve`** to find the critical points and to classify each one as a local maximum, local minimum, or saddle point.

(b) Check your answer to (a) by showing that **`fminsearch`** correctly locates the same local minima when you start at $(0.5, 0.5)$ or at $(-0.5, 0.5)$.

(c) What happens when you apply **`fminsearch`** with a starting value of $(0, 0)$? Explain your answer.

(d) What are the values of f at the extrema? Now, using **`fmesh`**, graph the function on a rectangle that includes all the critical points. Experiment with **`view`** and **`axis`** until you get a picture that shows the behavior near the critical points. Use the graph and all the previous data to justify the assertion: *Sometimes symbolic and/or numerical computations are more revealing than graphical information.*

Problem 7.2. Let $f(x, y) = (2x^2 + 3y^2)e^{-x^2-y^2}$.

(a) Compute the partial derivatives of f as well as its discriminant. We would like to use **`solve`** to find the critical points and to classify each one as a local maximum, local minimum, or saddle point. But the complexity of the algebraic expressions for f_x, f_y, and Hess f might give MATLAB some difficulty. If necessary, recompute these expressions by using **`simplify`**. Notice that there is a common exponential expression. Delete it before solving $f_x = f_y = 0$. Why is that legitimate? Then proceed with the classification.

(b) The point $(0, 0)$ gives a local minimum, even a global minimum, for f. Why is this obvious even without the second derivative test?

(c) Show that when you apply **`fminsearch`** to $-f$ with starting values of $(1, 0)$ and $(1, \pm 0.00001)$, you get very different results. How can you explain this?

(d) Draw a contour plot of f that encompasses the critical points $(0, 0)$ and $(0, 1)$. Draw another one encompassing all five critical points. Explain how the pictures relate to the results of Sections 5.1, 5.2, and 7.2.

Problem 7.3. Let $f(x, y) = x^6 + 3xy + y^2 + y^4$.

(a) Show that f remains unchanged if you replace x by $-x$ and y by $-y$. Hence, if (x, y) is a critical point of f, so is $(-x, -y)$. Thus, critical points other than $(0, 0)$ come in $\pm$ pairs.

(b) Compute the partial derivatives of f. Show that **solve** applied directly to the system $f_x = f_y = 0$ fails to locate any of the critical points except for $(0, 0)$.

(c) Let's compensate by eliminating one of the variables and then using **solve** followed by **double**. First solve for y in terms of x in the equation $f_x = 0$. Substitute back into the formula for f_y and then apply first **solve** and then **double**. You should end up with three critical values of x, giving a total of three critical points. Find the numerical values of their coordinates. (Be sure you have set x and y to be real; otherwise you will also end up with many irrelevant complex critical points.)

(d) Confirm the calculation of the critical points by graphing the equations $f_x = 0$ and $f_y = 0$ on the same set of axes (using **fimplicit** and **hold on**). You should see exactly one additional pair of critical points (in the sense of (a)).

(e) Classify the three critical points using the second derivative test.

(f) Apply **fminsearch** to f with the starting values $(1, 1)$ and $(0, 0)$. Show that in the first case you go to a minimum and that in the second case you stay near the saddle point.

Problem 7.4. As we mentioned in Section 7.1.2, the MATLAB command **fzero** and the Newton–Raphson method described in Section 7.1.3 do not work exactly the same way. This problem explores the difference between them. (You may find an implementation of Newton's Method—a script file called `newton.m`—on the companion web site for this book.) We start with a fairly simple example. Define a function handle in MATLAB by

```
>>  f = @(x) sin(x) - x/2
```

(a) Plot the function f and verify (graphically) that this function has a zero a little to the left of $x = 2$.

(b) Turn on **format long** and do iterations of Newton's method starting from $x = 2$ until the value of f is within an error of 10^{-15}. Make a table where each line gives the iteration number, value of x, and value of $f(x)$. How many iterations are required for convergence, and what is the numerical value of the zero of f?

(c) Look at the help for **fzero** and see how to get the routine to print out a similar table. How many iterations does **fzero** require for convergence?

(d) Do a similar test with the function $(x - 2)^{10}$, which has a much "flatter" graph near the zero at $x = 2$, starting from the point $x = 1$. (Note that the graph of this function never crosses the x-axis.)

(e) Finally, do a similar test with the function $(x-2)^{11}$, which is like the previous example except that it does change sign. Again use $x = 1$ as the starting point.

(f) Based on your tests, explain the advantages and disadvantages of the two algorithms.

Problem 7.5. Write a MATLAB function script `newton2d.m` that takes as inputs two symbolic expressions **`f1`** and **`f2`** in variables **`xvar`** and **`yvar`** as well as (numerical double-precision) starting values **`xstart`** and **`ystart`** and (assuming the 2-dimensional Newton's method converges) outputs a vector that is a numerical approximation to a joint zero of **`f1`** and **`f2`**. Your function should implement the algorithm discussed in Section 7.2.3.

Problem 7.6. Find (numerically) all solutions of the system of nonlinear equations

$$\begin{aligned} x^4 + y^4 &= 2, \\ y &= xe^{-x}. \end{aligned}$$

First determine the number of solutions and their approximate locations graphically; then eliminate y using the second equation and use **`fzero`** to compute the solutions numerically. You may have to adjust certain options to get a sufficiently accurate answer. Now try using your function from Problem 7.5 as another way of finding the solutions.

Problem 7.7. Use the second derivative test to find the conditions on a and b for the function $f(x, y, z) = x^2 + 3y^2 + z^2 + axy + byz + xz$ to have a strict local minimum at $(0, 0, 0)$. When is the second derivative test inconclusive? In fact the values of a and b for which f has a strict local minimum at the origin will be the interior of a solid region in the a-b plane. The second derivative test fails for points (a, b) on the boundary of that region. Your task is to find formulas in terms of a and b to describe the region. Finally, graph the region in the a-b plane where the conditions for a strict local minimum are satisfied.

Problem 7.8. As we indicated in the text, for a function of two variables with an *isolated* critical point, if the determinant of the Hessian is negative at this critical point, then the level curve through this critical point locally looks like the two intersecting lines of the letter "X." (See Figures 5.8, 7.2, or 7.3 for examples.) However, if the determinant of the Hessian vanishes, then the level curve through the critical point can have a higher degree of self-intersection. An interesting example is provided by the "monkey saddle," corresponding to the function $f(x, y) = x^3 - 3xy^2$.

(a) For this function, show that the only critical point is at $(0,0)$ and that the discriminant (the determinant of the Hessian) vanishes there.

(b) Draw a contour plot of the level curve $f(x,y) = 0$, as well as of a few level curves of f for values just above and below 0. What is the shape of the level curve through the critical point? How would you describe this kind of critical point?

Problem 7.9. This problem illustrates some of the issues involved in searching numerically for a minimum of a function of many variables. Let $f(x,y,z) = x^4 + y^4 + z^4 - 3x^2 - 8y^2 - 6xz - 4xy - 10z^2 + 6y$. This function has a global minimum, since the quartic terms grow rapidly compared to the other terms as any of $|x|$, $|y|$, $|z|$ goes to infinity, and thus f becomes large except in a neighborhood of the origin in 3-space. However, it is not easy to solve for the minimum analytically.

(a) Apply **fminsearch** to search for a minimum, starting from the origin. Recall that **fminsearch** requires a function of a single vector variable rather than of multiple variables, so you will have to input f as something like

```
>>  f = @(u) u(1)^4 + u(2)^4 + u(3)^4 + ...
```

Read the help on **fminsearch** and use options to trace how many steps the algorithm takes and how it arrives at the minimum.

(b) If you have access to the Optimization Toolbox do the same with **fminunc**.

(c) Even if you do not have access to **fminunc**, try to use steepest descent to arrive at a minimum, again starting from the origin. If $\mathbf{x}$ is the starting point for a step, you should move to $\mathbf{x} - h\frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$, where you can for example start with $h = 0.1$ and cut h down to 0.01 when the step takes you to a point where f is bigger, not smaller. Stop when even such a small step takes you to a point where f is bigger. How many iterations are needed, and what is your approximation to the minimum point?

(d) You should have found that (a) and (c) seemed to lead to different answers. In order to try to explain this, show that you can solve for x as a function of z using the equation $\frac{\partial f}{\partial z} = 0$. Take this expression for x in terms of z and substitute into the equations $\frac{\partial f}{\partial x} = 0$ and $\frac{\partial f}{\partial y} = 0$ in order to get two equations in the two unknowns y and z for a critical point. Solve these equations graphically (by plotting both on the same set of axes) and show that f has multiple critical points. In fact, what happened in the earlier parts of the problem is that different algorithms converged to different critical points of f.

(e) Still another way to search for a global minimum is the *Monte Carlo method*. Use the MATLAB function **rand** to generate a large number of random vectors in 3-space, say with coordinates between -6 and 6. (Once any of the coordinates is large enough, the quartic terms in f will swamp the other terms, and thus one can't be close to the minimum.) Evaluate f at these points and find the

minimum. Do it a few times. See how your result compares to what you found in earlier parts of the problem.

Problem 7.10. The MATLAB routines **`solve`** and **`fzero`** can also be applied to the system of Lagrange multiplier equations from Section 7.4:

$$\nabla f(x, y) = \lambda \nabla g(x, y),$$
$$g(x, y) = 0$$

for two functions f and g. Here f is the function to be maximized or minimized and g is the constraint function.

(a) Find (exactly, using Lagrange multipliers and **`solve`**) the location and value of the absolute minimum of the function $xy - 3x$ on the circle $x^2 + y^2 = 1$. Give numerical values of your exact answers.

(b) Find (numerically) the location and value of the absolute maximum and minimum of the function $f(x, y) = e^x - \sin y$ on the ellipse $g(x, y) = x^2 + 3y^2 = 1$. Do this two ways.

(i) Parametrize the ellipse by $x = \cos t$, $y = \frac{1}{\sqrt{3}} \sin t$. Substituting into the formula for f, get a function of the single variable t and find its maximum and minimum on the interval $[0, 2\pi]$ using **`fminbnd`**.

(ii) Now do the problem using Lagrange multipliers. Plot equations (7.7) on a single set of axes to see roughly where the solutions are located. Use `newton2d` from Problem 7.5 to find numerical values for the location and value of the absolute maximum and minimum of f on the ellipse. Check your answer by drawing a picture illustrating that the maximum and minimum occur where the level curves of f are tangent to the ellipse.

Problem 7.11. In practical optimization problems, we often have several constraints in the form of inequalities. Suppose for example that we want to find the absolute minimum of a function $f(x, y)$ subject to a constraint $g(x, y) \leq 0$. Geometrically, the constraint represents a region $\mathscr{R}$ in the x-y plane, with boundary defined implicitly by the curve $g(x, y) = 0$. Two things can happen: either the minimum occurs at a critical point of f where $g(x, y) < 0$ (the case of an *interior minimum*) or else it occurs where $g(x, y) = 0$ (the case of a *minimum on the boundary*). We can use MATLAB first to find the interior minima (either by solving for the critical points or by using **`fminsearch`**) and then to find the minimum on the boundary using Lagrange multipliers. The true global minimum is found by comparing the two answers and picking the point where f is the smallest. Carry out the complete procedure to find the location and value of the absolute minimum of the function $f(x, y) = 4 - x^2 - y^2 + x^4 + y^4$ subject to the constraint $g(x, y) = x^2 - 2x + y^2 - 3 \leq 0$. Be patient; there are many interior critical points and several very complicated solutions of the Lagrange multiplier equations.

Problem 7.12. This problem is a follow-up to the "baseball problem" (Problem 4.2) in Problem Set D. For simplicity we will neglect air resistance.

(a) In part (b) of the baseball problem, you determined the angle θ at which a batter should hit the ball to maximize the distance d traveled by the ball before it hits the ground, assuming that the initial speed v_0 at which the ball leaves the bat is held fixed. However, it is probably true that the batter can hit the ball with more power if his swing is level than if he needs to hit under the ball to loft it into the air. Take this into account by assuming that the speed with which the ball leaves the bat is not independent of θ but of the form $v_0/(1+\sin\theta)$. Redo part (a) of the baseball problem with this variable initial velocity function (that depends on the angle); take $g = 32\,\text{ft/sec}^2$, $h = 4\,\text{ft}$, and $v_0 = 100\,\text{ft/sec}$ (about 68 mi/hr). Use **`fminsearch`** to locate the value of θ that maximizes the distance d. Does your answer seem reasonable?

(b) If a batter is trying for a single and not for a home run, the best strategy is probably not to maximize the distance traveled by the ball but rather to minimize the chance that it will be caught, subject to the constraint that the ball should be hit out of the infield. Thus the batter wants the ball to stay in the air as little time as possible. So solve the following constrained optimization problem: find the value of θ that minimizes the time t_0 till the ball hits the ground, subject to the constraint that $d \geq 120\,\text{ft}$. Again take the speed of the ball to be $v_0/(1+\sin\theta)$, but assume now that the batter can also adjust v_0 to be anything from 0 up to 120 ft/sec. (You want to minimize t_0 as a function of v_0 and θ.)

Problem 7.13. Optimization of functions of several variables is exceptionally important in quantum mechanics. To demonstrate this, we consider for simplicity a particle of mass m free to move in one direction (say the x-axis), sitting in the "potential well" described by a function $V(x)$ that tends to $+\infty$ as $x \to \pm\infty$. Then the "ground state" of the particle is given by the *wave function* ϕ which satisfies

$$\int_{-\infty}^{\infty} \phi(x)^2\,dx = 1 \tag{7.8}$$

and for which the energy

$$E(\phi) = \int_{-\infty}^{\infty} \left(\frac{\hbar^2}{2m}\phi'(x)^2 + V(x)\phi(x)^2 \right) dx \tag{7.9}$$

is as small as possible. Here $\hbar$ denotes Planck's constant. (We will not attempt to justify this here, or to explain why such a function exists and is unique for reasonable choices of V.) Now, for any x, the size of $|\phi(x)|^2$ is roughly speaking a measure of how likely the particle is to be located close to x, and equation (7.8) says that

the total probability of finding the particle *somewhere* is 1. In formula (7.9), the first term measures average "kinetic energy" and the second term measures average "potential energy."

A practical method that is often used to estimate the ground state energy and the form of the corresponding wave function is to guess some reasonable form for $\phi(x)$, depending on a few parameters, and then to minimize $E(\phi)$ as a function of these parameters. Here we experiment with this optimization technique in a few examples.

(a) For simplicity, take $m = 1$, $\hbar = 1$, and suppose $V(x) = x^2/2$, the case of a *harmonic oscillator*. In this case, it is known that the ground state wave function $\phi(x)$ is a *Gaussian function* of the form ce^{-ax^2} for some $a > 0$; where

$$c = \left(\int_{-\infty}^{\infty} e^{-2ax^2}\, dx \right)^{-1/2}$$

in order for (7.8) to be satisfied. Find the explicit formula for c in terms of a, and then compute the value of a that minimizes $E(\phi)$. What is the ground state energy? What fraction of this energy corresponds to kinetic energy and what fraction corresponds to potential energy? Plot the wave function ϕ.

(b) Again take $m = 1$ and $\hbar = 1$, but this time suppose $V(x) = x^4/2$, the case of the *anharmonic oscillator*. This potential is flatter than the previous one when $-1 < x < 1$, but it rises much faster for larger x. This time there is no closed-form expression for ϕ, so numerical methods are essential. By symmetry, we can see that ϕ must be an even function of x. Try taking $\phi(x) = ae^{-x^2}(1 + bx^2 + cx^4)$, where a is chosen in terms of b and c to satisfy (7.8). Find the formula for $E(\phi)$ as a function of b and c. Apply **`fminsearch`** with starting values $b = 0$ and $c = 0$ to estimate the ground state energy, and plot the approximate wave function. What fraction of the ground state energy corresponds to kinetic energy and what fraction corresponds to potential energy? Compare with what happened in part (a). Do the results make physical sense?

(c) Again take $m = 1$ and $\hbar = 1$, but this time suppose $V(x) = \sec x$ for $|x| < \pi/2$, $V(x) = +\infty$ for $|x| \geq \pi/2$. Since the potential is infinite for $|x| \geq \pi/2$, the particle in this case is constrained to the interval $|x| < \pi/2$, and in formulas (7.8) and (7.9) you can replace the upper limit of integration by $\pi/2$ and the lower limit of integration by $-\pi/2$. This time try $\phi(x) = a\cos^2 x(1 + bx^2 + cx^4 + dx^6)$, where a is chosen in terms of b, c, and d to satisfy (7.8). Apply **`fminsearch`** with starting values $b = 0$, $c = 0$, $d = 0$ to estimate the ground state energy, and plot your approximate wave function. Compare with what happened in parts (a) and (b).

Glossary of MATLAB Commands

fimplicit Produces a plot of an implicit equation $f(x, y) = 0$

fminbnd Searches numerically for a minimum of a function of a single variable on a bounded interval

fminsearch Searches numerically for a minimum of a function of a single (possibly vector) variable

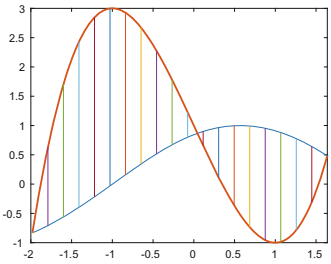
fminunc In the Optimization Toolbox. Searches for an unconstrained minimum of a function of several variables

fzero Searches numerically for a zero of a function of a single variable

jacobian Finds the gradient of a function of several variables. When applied to the symbolic expression for the gradient, can be used to compute the Hessian

rand Generates a matrix of uniformly distributed random numbers

Chapter 8
Multiple Integrals



In this chapter, we explain how to use MATLAB to set up and to compute multiple integrals. We discuss techniques for visualizing regions in the plane and in space. We describe how to use these techniques to find the limits of integration in multiple integrals. We also discuss various numerical algorithms for multiple integration, and finally, we explain how to use these techniques in a variety of coordinate systems.

8.1 Automation and Integration

We believe that it is crucial for students to work problems by hand in order to master the methods of integration that form the backbone of a traditional multivariable calculus course. In addition, students must learn how to use technological tools like MATLAB to solve similar—but harder and more realistic—problems. We believe this dual strategy provides benefits in all of mathematics; indeed, in all of modern science.

It is commonly held that the advent of handheld calculators has freed us from the drudgery of arithmetic. To some extent, this is true. We use calculators to balance checkbooks, to compute loan installments, and to free up time on mathematics and engineering examinations.

It is further held that there is no need to drill students in multiplication tables or similar techniques for arithmetic computation—the calculator will do it! Alas, there is a fatal flaw in this reasoning. If students cannot compute arithmetically, they are easily mislead by arithmetic mistakes and chicanery. If we have to multiply two 3-digit numbers, we can expect the calculator to get it right. If our fingers slip, however, and the resulting answer consisting of a 12-digit number does not cause us instant suspicion, it is because our inability to compute has also left us unable to estimate. Without that ability, we will not know what mistakes—inadvertent or otherwise—are being foisted upon us.

Modern technology has raised the stakes. We now have tools (such as MATLAB) that automate the basic techniques of algebra (finding roots, multiplying matrices,

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_8

simplifying radicals) and calculus (differentiating, integrating, finding limits). Just as we need to have mastered the basic rules of arithmetic in order to use a calculator effectively, so also must we master the basic rules of algebra and calculus in order to use MATLAB skillfully. If we do not understand the basic concepts, or have not mastered the basic methods of applying those concepts, then we cannot truly understand the output of a mathematical software system. Instead, we are reduced to the meaningless manipulation of symbols. In fact, the situation is worse than that. Calculus was invented to give a mathematical formulation to the basic rules of the universe (the laws of gravity, the nature of chemical reactions, the rules of chance). Without understanding the mathematics, we have no hope of understanding the phenomena that the mathematics is designed to represent.

The effective use of MATLAB for computing multiple integrals is an ideal venue in which to illustrate the importance of understanding concepts, as opposed to the mindless manipulation of symbols. In fact, MATLAB has already automated the computation of multiple integrals. If f_1 and f_2 are symbolic functions of x and g is a symbolic function of x and y, then in order to compute (symbolically) the double integral

$$\int_a^b \int_{f_1(x)}^{f_2(x)} g(x, y)\, dy\, dx,$$

you just enter the following command:

```
>>  int(int(g(x, y), y, f1(x), f2(x)), x, a, b)

ans =
int(int(g(x, y), y, f1(x), f2(x)), x, a, b)
```

But that is the easy part. Once you know how to describe a region in the plane in terms of simple boundaries (the vertical line $x = a$, the vertical line $x = b$, and the graphs of two functions $y = f_1(x)$ and $y = f_2(x)$ which never cross), then computing the double integral is just two iterations of the process that you already know for computing an integral in one variable. The hard part is figuring out how to describe a complicated region in such a simple fashion.

8.1.1 Regions in the Plane

We want to look at the following problem, which is an archetype of an entire class of problems found in every textbook ever written about multivariable calculus. Find the area between the two curves

$$y = \sin(x + 1) \quad \text{and} \quad y = x^3 - 3x + 1\,.$$

We start by defining the two functions, and plotting them on the same graph, shown in Figure 8.1.

```
>> f = @(x) sin(x + 1); g = @(x) x.^3 - 3*x + 1;
>> fplot(f, [-3, 3]), hold on
>> fplot(g, [-3, 3], 'LineWidth', 2)
```

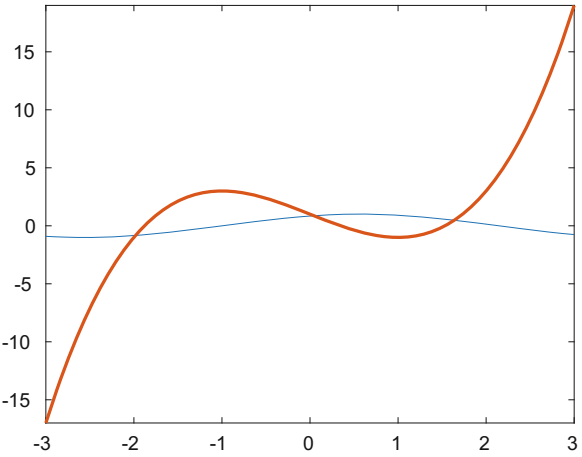


Fig. 8.1 Two intersecting curves

We drew the polynomial with a thicker style so we can distinguish the two curves; we also invoked the command **`hold on`** because we will augment the graph below.

The graph gives us several pieces of information. First of all, there are three points of intersection, whose x-coordinates are close to -2, 0, and 2. In order from left to right, let us call those x-coordinates x_1, x_2, and x_3. The region between the two curves consists of two distinct pieces, one lying above the interval $[x_1, x_2]$ and the other above $[x_2, x_3]$. Moreover, the two curves cross at x_2, exchanging the roles of top and bottom curve. Thus, when we finally get around to computing an integral, we will need to compute the contribution of each of these regions separately.

The **`solve`** command cannot find the points of intersection in this case, because of the transcendental function. Instead, we will use **`fzero`**, discussed in the previous chapter.

```
>>  x1 = fzero(@(x) f(x) - g(x), -2)

x1 =
   -1.9810

>>  x2 = fzero(@(x) f(x) - g(x), 0)

x2 =
   0.0450

>>  x3 = fzero(@(x) f(x) - g(x), 2)

x3 =
   1.6383
```

In order to identify the region visually, we are going to "shade" it with vertical lines. We will need the version of the MATLAB command **plot** in which both arguments are matrices of the same size. In this case, **plot** plots the columns of the first matrix against the columns of the second matrix. So if each matrix is of size 2×10, and both rows of the first matrix are equal, we end up plotting 10 vertical lines. The following commands draw 10 lines (some of which may collapse to a point), each of which connects a point $(x, f(x))$ on one curve via a vertical line to a point $(x, g(x))$ on the other curve.

```
>> xcoord = linspace(x1, x3, 10);
>> ycoord = [f(xcoord); g(xcoord)];
>> plot([xcoord;xcoord], ycoord), hold off
```

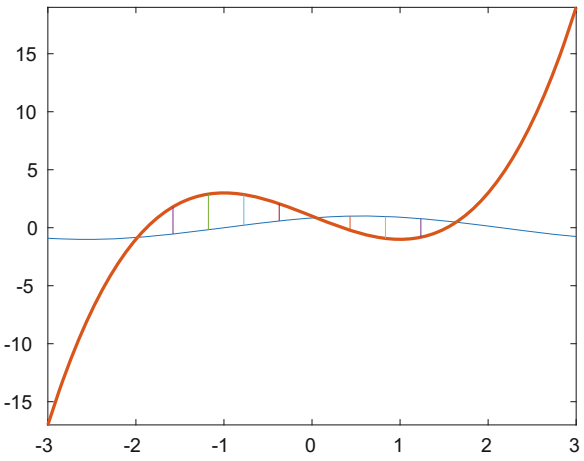


Fig. 8.2 Two intersecting curves with vertical shading

With the graph (Figure 8.2) to guide us, we can now confidently compute the area to a high degree of precision using integrals.

```
>> format long
>> leftarea = integral(@(x) g(x) - f(x), x1, x2)

leftarea =
   4.005062870518554

>>  rightarea = integral(@(x) f(x) - g(x), x2, x3)

rightarea =
   2.006604929298614

>>  area = leftarea + rightarea

area =
   6.011667799817168
```

Exercise 8.1. We were careful to break the region into two pieces; that operation is necessary to solve the more general problem of integrating any function of two variables over a region. When you only want to compute an area, you can sometimes get a quicker estimate by integrating the absolute value of the difference between the bounding functions. Try the command

```
>> integral(@(x) abs(f(x) - g(x)), x1, x3)
```

and compare its result with the answer we just computed. (You will need **format long** to see the difference.) Which method is more accurate, do you think, and why?

Depending on the functions involved, it may be easiest to use numerical integration, as in the above example, or it might be simpler to use symbolic double integration. To do the same problem by the second method, replace the above code with

```
>> leftarea = double(int(int(1, sin(x+1), x^3-3*x+1), x1, x2))

leftarea =
   4.005062870518554
```

and so on.

8.1.2 Viewing Simple Regions

A region is called *vertically simple* if each vertical line intersects the region in at most one connected segment. We can automate the technique we just used to view a vertically simple region by writing our own MATLAB script. Recall that we drew two curves, one with a normal **LineWidth** and the other with a thicker style; then we drew several vertical lines connecting the two curves. We can write a MATLAB function script to carry out the same two steps for any vertically simple region. The arguments that we will need to supply to the script come from those we needed to supply to the integration command: the left boundary constant **a**, the right boundary constant **b**, the bottom bounding function **f**, and the top bounding function **g**.

```
function verticalRegion(f, g, a, b)
fplot(f, [a, b]), hold on
fplot(g, [a, b], 'Linewidth', 2)
xcoord = linspace(a, b, 20);
ycoord = [f(xcoord); g(xcoord)];
plot([xcoord;xcoord], ycoord), hold off
end
```

The definition of **verticalRegion** wraps everything inside a **function** definition. This construction serves two purposes. First, it allows us to group several statements into a single structure. Second, it allows us to use local variable names (in this case, **xcoord** and **ycoord**) without having to worry about whether those names have been used elsewhere in the same MATLAB session.

We could reproduce (an enhanced version of) our previous picture by typing

```
>> verticalRegion(x1, x3, f, g)
```

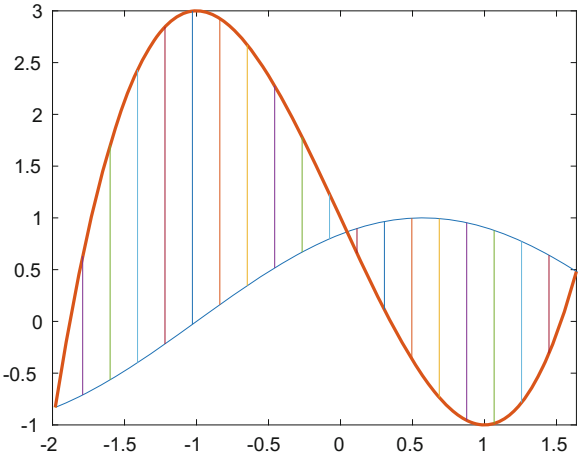


Fig. 8.3 More detailed picture of the region between two curves

See Figure 8.3 for the output. As the picture indicates, we can only refer to **f** as the bottom and **g** as the top if we were careful enough to divide the region up into pieces before trying to graph it. A visual inspection of the graph will tell us if we got it right; the top curve should be drawn with a thicker style than the bottom curve. (See the help on **LineSpec** for other ways to define distinct plot styles.)

A region is called *horizontally simple* if each horizontal line intersects the region in at most one connected segment. In Problem 8.3 of Problem Set H, you will be asked to modify the **verticalRegion** program to draw horizontally simple regions. Figure 8.4 shows a horizontally simple region between the curves $x = \sin(y)$ and $x = \cos(y)$ that was drawn with such a program.

We can find the area of this region by integrating first with respect to x, and then with respect to y.

```
>> int(int(1, x, sin(y), cos(y)), y, -3*pi/4, pi/4)

2*2^(1/2)
```

8.1.3 Polar Regions

Many regions are more naturally described using polar coordinates instead of rectangular coordinates. Using a different coordinate system provides us with different natural ways to chop up regions. As an example, consider the following region R. Start with a circle of radius 2. Now, remove a circle of radius 1 that is tangent to the first circle. What is the area of the region R that remains?

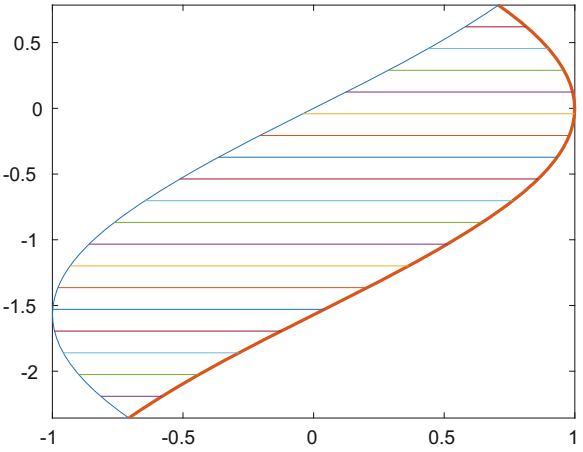


Fig. 8.4 Picture of a horizontally simple region

We do not need calculus to answer this question. The outer circle has area 4π; the deleted circle (regardless of its position inside the bigger circle) has area π, so the remaining area is just 3π. So why would we go to the trouble of carefully graphing the region and evaluating a double integral in polar coordinates just to find out something we already know? Well, consider Figure 8.5.

This graph shows a sphere of radius 2 intersected by a cylinder of radius 1; one edge of the cylinder passes through the center of the sphere, and the other edge is tangent to the sphere at the equator. (Note: The curve along which these surfaces intersect is called Viviani's curve, and was studied in Chapter 2.) Suppose we drill a hole where the cylinder passes through the sphere. What is the volume of the portion of the sphere that remains? Finding the volume of this solid requires us to integrate a function over the plane region R—but this time, we do not know the answer in advance. We will return to this problem later.

We are going to define a MATLAB function script that plots polar regions in the same way **`verticalRegion`** plots vertically simple regions. This command will plot regions corresponding to a polar integral of the form

$$\int_{\alpha}^{\beta} \int_{r_1(\theta)}^{r_2(\theta)} f(r,\theta)\, r\, dr\, d\theta.$$

In other words, we want to be able to display regions defined in polar coordinates that are bounded by the lines $\theta = \alpha$ and $\theta = \beta$, and lying between the curves $r = r_1(\theta)$ and $r = r_2(\theta)$.

Now we can design a MATLAB function script that plots regions in polar coordinates.

```
function polarRegion(R1, R2, a, b)
tt = linspace(a,b);
```

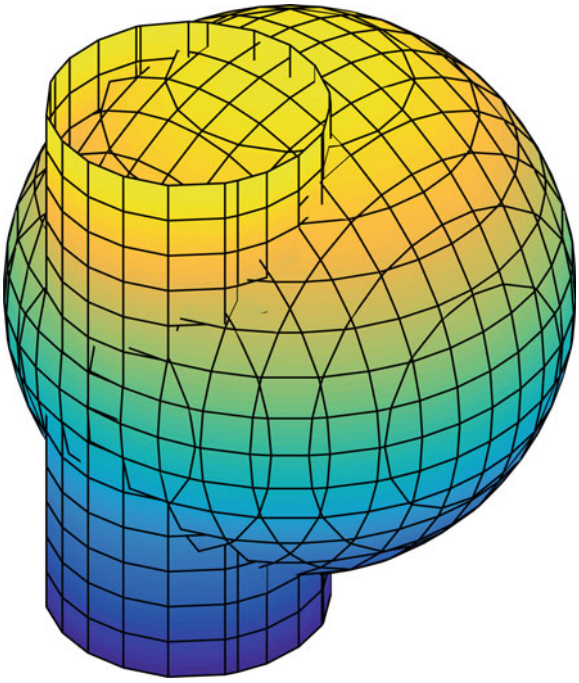


Fig. 8.5 Boring out a cylindrical hole from a sphere

```
polarplot(tt, R1(tt)), hold on
polarplot(tt, R2(tt), 'Linewidth', 2)
ttt = linspace(a, b, 20);
rcoord = [R1(ttt); R2(ttt)];
polarplot([ttt;ttt], rcoord), hold off
end
```

The solid shown in Figure 8.5, obtained by drilling out a cylinder from a sphere, projects onto a region R in the plane bounded on the outside by the projection of the sphere, or in polar coordinates, the circle $r = 2$, and bounded on the inside by the projection of the cylinder, or the circle $(x-1)^2 + y^2 = 1$ or $x^2 + y^2 = 2x$. When we convert the equation of this circle to polar coordinates, $x^2 + y^2$ becomes r^2 and $2x$ becomes $2r\cos\theta$, so after canceling an r, we get the polar equation $r = 2\cos\theta$. Since we want $r \geq 0$, in general we want the maximum of this and 0. So we can sketch the region R between the two circles using the following code. The result is shown in Figure 8.6.

```
>> circ1 = @(t) max(2*cos(t), zeros(size(t)));
>> circ2 = @(t) 2*ones(size(t));
>> polarRegion(circ1, circ2, 0, 2*pi)
```

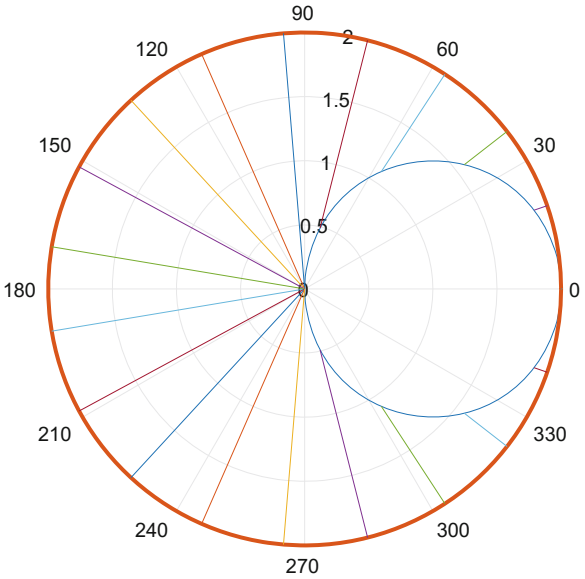


Fig. 8.6 The polar region between two circles

Now let us check that we can compute the area correctly. We integrate separately the regions with $|\theta| \leq \frac{\pi}{2}$ and with $\frac{\pi}{2} \leq \theta \leq \frac{3\pi}{2}$.

```
>> int(int(r, r, 0,  2), theta, pi/2, 3*pi/2) + ...
     int(int(r, r, 2*cos(theta),  2), theta, -pi/2, pi/2)

ans =
3*pi
```

This is correct since we are subtracting the area of a circle of radius 1 from the area of a circle of radius 2.

Next, we return to the problem of computing the volume of the part of the sphere that remains after the intersection with the cylinder is removed. Observe first that the sphere of radius 2 with center at the origin has the equation $x^2 + y^2 + z^2 = 4$ or $r^2 + z^2 = 4$, or $z = \pm\sqrt{4 - r^2}$. The volume is

$$\iint_R (z_{\text{top}} - z_{\text{bot}}) r \, dr \, d\theta,$$

or by symmetry, the integral of $4\sqrt{4 - r^2}\, r$ with θ only ranging from 0 to π. So the volume is:

```
>>  vol = int(int(4*sqrt(4-r^2)*r, r, 0, 2), theta, pi/2, pi) + ...
     int(int(4*sqrt(4-r^2)*r, r, 2*cos(theta), 2), theta, 0, pi/2)

vol =
(16*pi)/3 + 64/9
```

Here the powerful symbolic capabilities of MATLAB have proved quite useful; this would have been hard (though not completely impossible) to compute by hand.

8.2 Algorithms for Numerical Integration

At this point it is time to say a bit about algorithms for numerical integration—how they work and what the drawbacks and advantages are. This information is important for interpreting numerical output produced by MATLAB and knowing when to trust it.

8.2.1 Algorithms for Numerical Integration in a Single Variable

Let us begin by reviewing integration in a single variable. Suppose f is a continuous function on an interval $[a, b]$. If we know an antiderivative F of f, then of course the Fundamental Theorem of Calculus (FTC) tells us that

$$\int_a^b f(x)\,dx = F(b) - F(a).$$

From a theoretical point of view, this is perfect. But from an algorithmic point of view, there is a hidden difficulty. In order to compute the integral accurately, we need to be able to compute the function F accurately. If the function F is very complicated, this may not be so easy to do. So if you are integrating a complicated symbolic expression f using MATLAB's Symbolic Math Toolbox and then use **`double`** to convert to a numerical answer, remember that you are using MATLAB's built-in algorithms for evaluating functions numerically. Most of the time these are pretty accurate, but since they often rely on series approximations, or other approximation techniques, keep in mind that they are never perfect.

In many cases, an explicit formula for F may simply not be available, or it may be so complicated that evaluating it accurately may not be so easy. In this case, you probably want to use numerical integration. Algorithms for this purpose that are commonly taught in first-year calculus include the trapezoidal rule and Simpson's rule. These algorithms, as well as the (usually) more accurate algorithm used by MATLAB's **`integral`** command, have certain features in common. They rely on the fact that definite integration, $I\colon f \mapsto \int_a^b f(x)\,dx$, is a *linear* operation on functions; that is $I(f+g) = I(f) + I(g)$ and $I(cf) = cI(f)$ for c a constant. Furthermore, $I(1) = b - a$ (this helps normalize things) and $I(f) \geq 0$ if $f \geq 0$. Since a computer can only do finitely many evaluations of the input function f, numerical algorithms for approximating $I(f)$ replace integration I by an operation of the form

$$Q(f) = c_1 f(x_1) + c_2 f(x_2) + \cdots + c_n f(x_n), \tag{8.1}$$

called a *quadrature formula*, where $x_1, \ldots, x_n$ are suitably chosen points in the interval $[a, b]$ and the constants c_j are positive (to guarantee that $Q(f) \geq 0$ if $f \geq 0$) and satisfy $\sum_j c_j = b - a$ (to get the normalization right). The art of numerical integration consists of making the right choices of the points x_j and the constants c_j. For example, in the case of Simpson's rule, the x_j's are chosen equally spaced and the c_j's are proportional to $1, 4, 2, 4, 2, \ldots, 4, 1$.

In most commonly used quadrature formulas, the x_j's and the c_j's are chosen so that the formula (8.1) will be *exact* for polynomials up to a certain degree. For example, suppose one wants a formula for $Q(f)$ with $n = 3$ that will correctly integrate polynomials f of degree ≤ 5 over the interval $[-1, 1]$. By symmetry considerations and because we need $Q(1) = 2$, it is clear that we want to take

$$Q(f) = 2(1 - c)\, f(0) + c\, f(a) + c\, f(-a)$$

for some $0 < a \leq 1$ and some $0 < c < 1$. It is clear that with these choices, $Q(f) = 0$ for f an odd function (such as x, x^3, or x^5). So we just need to arrange that $Q(x^2) = \int_{-1}^{1} x^2\, dx = \frac{2}{3}$ and $Q(x^4) = \int_{-1}^{1} x^4\, dx = \frac{2}{5}$. We get the equations $2c\,a^2 = \frac{2}{3}$ and $2c\,a^4 = \frac{2}{5}$. Thus $a^2 = \frac{3}{5}$, $a = \sqrt{\frac{3}{5}}$, and $c = \frac{5}{9}$. This is called the *Gauss quadrature formula* for $n = 3$. See Problem 8.8 for more on this topic and Problems 8.9 and 8.10 for 2-dimensional analogues.

Since arbitrary continuous functions can be approximated by polynomials, using quadrature formulas of the type we have just discussed means we can expect reasonable results for most commonly encountered continuous functions. However, *the more jagged or highly oscillatory the function, the worse its approximation by polynomials may be, and the greater the error in using formula* (8.1) *as a replacement for the integral.* Usually one chooses an *adaptive* method, which means that one increases the number n of points, and sometimes adjusts their positions and the constants c_j, until certain tolerances are met.

8.2.2 Algorithms for Numerical Multiple Integration

Now let us move on to double integrals. If one wants to integrate a continuous function f of two variables over a region R in the plane, similar quadrature formulas can be generated. (See, for example, Problems 8.9 and 8.10.) But a new difficulty arises. Except for rescaling, all intervals in one variable look alike. But in the plane, the geometry of the region R will play a role. Since it is impractical to write new quadrature algorithms for every possible geometry, the standard algorithms are only written for the case of a rectangle. In principle this results in no loss of generality, since one thing we can do is to enclose R in a rectangle and extend the function g that we are integrating to be 0 outside of R. For example, to find the area of a

circular disk of radius 1, which of course is π, we can integrate $g(x, y)$ over the square $-1 \le x,\ y \le 1$, where

$$g(x, y) = \begin{cases} 1, & x^2 + y^2 \le 1, \\ 0, & x^2 + y^2 > 1. \end{cases}$$

The function g can be defined by heaviside$(1-x^2-y^2)$, where the *Heaviside function* is the step function that takes the value $+1$ for positive inputs and the value 0 for negative arguments. So we might try the MATLAB code

```
>> integral2(@(x,y) heaviside(1-x.^2-y.^2), -1, 1, -1, 1)

Warning: Reached the maximum number of function evaluations
(10000). The result fails the global error test.
> In integral2Calc>integral2t (line 129)
  In integral2Calc (line 9)
  In integral2 (line 106)
ans =
   3.1417
```

Note that we have accuracy only to 3 decimal places. The reason why the result is so inaccurate, and also the reason why MATLAB spits out a warning message, is that the Heaviside function is discontinuous, and thus the algorithm does not work so well.

To get better results, we should integrate as in Section 8.1.2. In the case where R is vertically simple, we have

$$\int_a^b \int_{f_1(x)}^{f_2(x)} g(x, y)\, dy\, dx = \int_a^b \int_0^1 g\left(x, f_1(x) + t\left(f_2(x) - f_1(x)\right)\right)\left(f_2(x) - f_1(x)\right) dt\, dx, \tag{8.2}$$

and thus we have again reduced to the case of integration over a rectangle, this time with a continuous integrand. An important point to keep in mind is that in doing this, we have in effect replaced the original function $g(x, y)$ by a new function

$$h(x, t) = g\left(x, f_1(x) + t\left(f_2(x) - f_1(x)\right)\right)\left(f_2(x) - f_1(x)\right).$$

Even if the original function g was very smooth (or even constant!), this new function h can be quite complicated, and depends on the geometry of the region R through the functions $f_1(x)$ and $f_2(x)$. This often results in hidden difficulties with numerical multiple integration. If we were integrating in 3 or more variables, problems of this sort would tend to be even worse. Here is an example. We will turn on **`format long`** to see effects out to many decimal places.

Suppose we want to find the area of a circular disk R of radius 1. Of course we know that the area is π. But notice what happens when we do numerical double integration to find the area. The region R is vertically simple and is bounded above by the graph of $y = \sqrt{1-x^2}$ and below by the graph of $y = -\sqrt{1-x^2}$. Applying (8.2) with $g \equiv 1$, we get the integral

$$\int_{-1}^{1} \int_{0}^{1} 2\sqrt{1-x^2}\, dt\, dx\,.$$

Let's use MATLAB's numerical double integration command **integral2**.

```
>> format long, integral2(@(x,t) 2*(1-x.^2).^(1/2), -1, 1, 0, 1)

ans =
   3.141592653589794

>>  ans - pi

ans =
   8.881784197001252e-16
```

This is quite good; we have accuracy to 15 decimal places. But suppose the region were more jagged. For example, suppose we want to find the area of interior of the astroid $x^{\frac{2}{3}} + y^{\frac{2}{3}} = 1$. It is known that the area is $\frac{3\pi}{8}$, as one can see from

```
>> 4*int((1 - x^(2/3))^(3/2) ,x, 0, 1)

ans =
(3*pi)/8
```

Doing the calculation numerically the same way as in the last example, MATLAB gives

```
>> integral2(@(x,t) 2*(1-(x.^2).^(1/3)).^(3/2), -1, 1, 0, 1)

ans =
   1.178097109965436

>>  ans - 3*pi/8

ans =
   -1.351307366004306e-07
```

This time we only get 7-place accuracy. Incidentally, the reason for writing **(x.^2).^(1/3)** instead of **x.^(2/3)** is that MATLAB misinterprets the latter when x is negative. (Try it and see!)

For the reasons we have already explained, as the number of dimensions goes up, numerical integration over arbitrary regions by reducing to integration over (higher dimensional) rectangular boxes becomes increasingly inaccurate. At some point, it becomes more efficient to use a well-chosen coordinate system or other numerical methods, which will be discussed in Problem Set H. For example, suppose one wants

to compute the volume of a spherical ball R of radius 1 in 3-space. Of course we know that the answer is $\frac{4}{3}\pi$. Since R is the region where $x^2+y^2+z^2 \le 1$, the naive approach with integration in Cartesian coordinates would be to write

$$\begin{aligned} V &= \iiint_R dz\,dy\,dx = \int_{-1}^{1}\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}}\int_{-\sqrt{1-x^2-y^2}}^{\sqrt{1-x^2-y^2}} dz\,dy\,dx \\ &= \int_{-1}^{1}\int_{-\sqrt{1-x^2}}^{\sqrt{1-x^2}} 2\sqrt{1-x^2-y^2}\,dy\,dx \end{aligned}$$

or

$$\begin{aligned} V &= \int_{-1}^{1}\int_{0}^{1} 2\sqrt{1-x^2-\left((2t-1)\sqrt{1-x^2}\right)^2}\cdot 2\sqrt{1-x^2}\,dt\,dx \\ &= 4\int_{-1}^{1}\int_{0}^{1}\sqrt{(1-x^2)\left(1-(2t-1)^2\right)}\,dt\,dx \\ &= 8\int_{-1}^{1}\int_{0}^{1}(1-x^2)\sqrt{t-t^2}\,dt\,dx. \end{aligned}$$

Computing numerically in MATLAB, we get

```
>>  integral2(@(x,t) 8*(1-x.^2).*sqrt(t-t.^2), -1, 1, 0, 1)

ans =
   4.188790204785021

>>  ans - 4*pi/3

ans =
   -1.369571123177593e-12
```

Thus we have accuracy to 12 decimal places. This is good but not perfect. In more complicated problems with triple integrals, one has to be alert to the possibility that the errors might be more substantial, especially if the geometry of the region R is complicated.

8.3 Viewing Solid Regions

Let us turn our attention now to solid regions. Our first task is to devise an analog of the **`verticalRegion`** command that will allow us to visualize three-dimensional solid regions. We will need to supply more arguments, as we need upper and lower limits for each of three variables. Here is a script that meets our requirements:

```
function viewSolid(xvar,a,b, yvar,f,g, zvar,F,G)
% This function script draws a solid region.
% The inputs xvar, yvar, and zvar are strings indicating how the
% axes should be labeled.  In general, they should be 'x', 'y',
% and 'z', but perhaps permuted. Then a, b are limits
% of integration (constants) for the variable called xvar,
% f and g are functions of xvar that are limits of integration
% for the variable called yvar, and F and G are functions of xvar
% and yvar that are limits of integration for the variable called
% zvar.  Note that f, g, F, and G should be handles for
% vectorized functions or anonymous functions.
for counter=0:20
  xx = a + (counter/20)*(b-a);
  YY = f(xx)*ones(1, 21)+((g(xx)-f(xx))/20)*(0:20);
  XX = xx*ones(1, 21);
%% The next lines inserted to make bounding curves thicker.
  widthpar=0.5;
  if counter==0, widthpar=2; end
  if counter==20, widthpar=2; end
%% Plot curves of constant x on surface patches.
  plot3(XX, YY, F(XX, YY).*ones(1,21), 'r', 'LineWidth', widthpar);
  hold on
  plot3(XX, YY, G(XX, YY).*ones(1,21), 'b', 'LineWidth', widthpar);
end;
%% Now do the same thing in the other direction.
XX = a*ones(1, 21)+((b-a)/20)*(0:20);
%% Normalize sizes of vectors.
YY=0:2; ZZ1=0:20; ZZ2=0:20;
for counter=0:20,
%% The next lines inserted to make bounding curves thicker.
  widthpar=0.5;
   if counter==0, widthpar=2; end
   if counter==20, widthpar=2; end
  for i=1:21,
      YY(i)=f(XX(i))+(counter/20)*(g(XX(i))-f(XX(i)));
      ZZ1(i)=F(XX(i),YY(i));
      ZZ2(i)=G(XX(i),YY(i));
  end;
  plot3(XX, YY, ZZ1, 'r', 'LineWidth',widthpar);
  plot3(XX, YY, ZZ2, 'b', 'LineWidth',widthpar);
end;
%% Now plot vertical lines.
for u = 0:0.2:1,
    x=(a + (b-a)*u)*ones(1,6); y = f(a + (b-a)*u)*ones(1,6) ...
        +(g(a + (b-a)*u)-f(a + (b-a)*u))*(0:0.2:1);
    plot3([x; x], [y; y], [F(x,y); G(x, y)], 'c');
end;
xlabel(xvar); ylabel(yvar); zlabel(zvar); hold off
end
```

Since this is a lot of code to digest at one time, let's go through it step by step. The first line simply describes the arguments to **viewSolid**. These are compatible with the arguments that you need to supply to the **int** command to compute the triple integral

$$\int_a^b \int_{f(x)}^{g(x)} \int_{F(x,y)}^{G(x,y)} h(x, y, z)\, dz\, dy\, dx.$$

(Of course, the **int** command also requires the integrand at the beginning of the argument list.)

You should note that **viewSolid** differs from **verticalRegion** by requiring you to specify the variables x, y, and z as arguments to the command. In the plane, we only had two choices for the order of integration, and only two kinds of regions to display. So, it made sense to construct separate commands. For **verticalRegion**, we could assume that y ranged between functions of x while x ranged between constants; **horizontalRegion** reversed the role of the variables. In three dimensions, we have six possible orders of integration, and six kinds of regions to display. In this case, it is easier to create a single command that will work for all six possible orders. We will illustrate the use of **viewSolid** with integration order z-y-x. So, the parameters **a** and **b** should be given constant values a and b; the parameters **f** and **g** should be function handles representing the mathematical functions $f(x)$ and $g(x)$, and the parameters **F** and **G** should be function handles for the mathematical functions $F(x, y)$ and $G(x, y)$. When using a different order, you will have to change the variables in the functions accordingly, and remember that "up" in your picture is not necessarily the z-direction.

The first block of code draws a cross-hatching of the top and bottom surfaces using MATLAB's command **plot3**, in blue and red, respectively. Note that we have an internal variable **widthpar**, "width parameter," that is set to 2 (for rather thick lines) on the bounding curves and is set to 0.5 (for rather thin lines) in all other cases.

The last block of code draws vertical lines joining the bottom surface to the top surface in cyan. Finally, we label the axes according to the variable names given as arguments.

Here is an example. We will view the solid region bounded below by the x-y plane, bounded above by the plane $z = (x + y)/4$, and lying over the plane region bounded by $x = 1$, $x = 2$, $y = x/2$, and $y = x$. The result is shown in Figure 8.7.

```
>> viewSolid('x', 1, 2, 'y', @(x) x/2, @(x) x,...
    'z', @(x,y) zeros(size(x)), @(x,y) (x+y)/4)
```

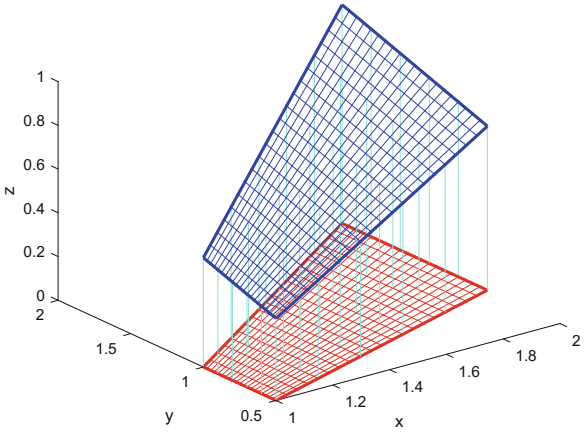


Fig. 8.7 The solid region between two planes

Now let us look at a more interesting example: we want to compute the volume of the solid region R bounded by the parabolic cylinder $z = 4 - y^2$ and the paraboloid $z = x^2 + 3y^2$. The first thing to do is to determine where the surfaces intersect. To do this we have to take the right-hand sides of the two equations and set them equal to one another.

```
>>  cyl = (4 - y^2); par = (x^2 + 3*y^2);

>>  xycurve = simplify(cyl == par)

xycurve =
x^2 + 4*y^2 == 4
```

We recognize this as the equation of an ellipse and can draw it with **`fimplicit`**. The result is shown in Figure 8.8.

```
>>  fimplicit(xycurve, [-2, 2, -1, 1]), axis equal
```

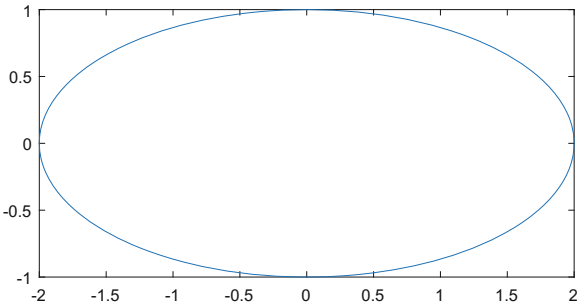


Fig. 8.8 The projection of the curve where a paraboloid and a parabolic cylinder intersect

Now, we can use the **solve** command to get the limits for y as a function of x.

```
>>  ysols = solve(xycurve, y)

ysols =
 -((2 - x)^(1/2)*(x + 2)^(1/2))/2
  ((2 - x)^(1/2)*(x + 2)^(1/2))/2
```

Here, we have some suppressed some warning messages that MATLAB prints out regarding possibly taking the square root of a negative number.

Before viewing the solid, we need to think for a moment about whether the parabolic cylinder $z = 4 - y^2$ or the paraboloid $z = x^2 + 3y^2$ is on the top. Since the point $(0, 0)$ lies inside the ellipse in the x-y plane, we can do a simple hand computation at this point to see that the height of the parabolic cylinder at this point is 4, while the height of the paraboloid is 0. So, the parabolic cylinder is on top, and the appropriate command to view the solid is as follows. Note that we have to convert symbolic expressions to function handles, which we can do with **matlabFunction**. We show the result in Figure 8.9.

```
>> ylim1 = matlabFunction(ysols(1));
>> ylim2 = matlabFunction(ysols(2));
>> viewSolid('x', -2, 2, 'y', ylim1, ylim2, 'z', @(x, y) ...
    x.^2 + 3*y.^2, @(x, y) 4 - y.^2)
```

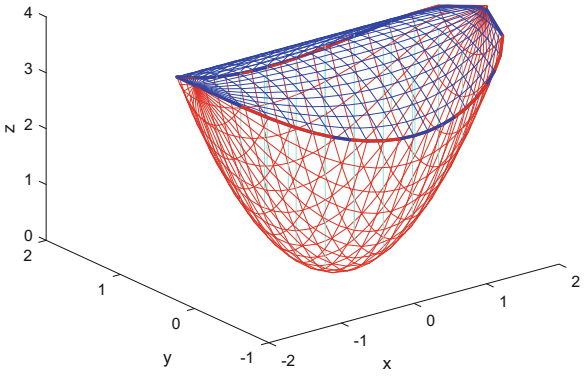


Fig. 8.9 The solid region between a paraboloid and a parabolic cylinder

Now we can integrate to find the volume. To speed up the process, we stop to simplify after the first two integrals.

```
>> firststep = int(int(1, z, par, cyl), y, ysols(1), ysols(2));
>> int(simplify(firststep), x, -2, 2)

ans =
4*pi
```

8.4 A More Complicated Example

In this section, we want to compute the volume of the intersection of two solids, the inside of a torus and the inside of a cylinder. A torus is the result of revolving a circle around a disjoint axis. Here, we take a circle of radius 1 and revolve it around an axis that lies four units from the center of the circle. Taking the axis of revolution to be the z-axis, and parametrizing the circle by $(4 + \cos u, 0, \sin u)$ for $0 \leq u \leq 2\pi$, we find that the torus is defined parametrically in MATLAB by

```
>> syms u v t r x y z real
>> torus = [(4 + cos(u))*cos(v), (4 + cos(u))*sin(v), sin(u)];
```

where $0 \leq u \leq 2\pi$ and $0 \leq v \leq 2\pi$. Writing $r = \sqrt{x^2 + y^2}$, we have

$$r^2 = x^2 + y^2 = (4 + \cos u)^2$$

and $z = \sin u$. Thus, the torus is defined in rectangular coordinates by:

```
>>  toreqn = simplify(expand(subs((r-4)^2+z^2==1,r,sqrt(x^2+y^2))))

toreqn =
x^2 + y^2 + z^2 + 15 == 8*(x^2 + y^2)^(1/2)
```

We intersected the torus with a cylinder of radius 1 that passes through the center of the torus. The cylinder is defined parametrically by

```
>>  cylinder = [u, cos(v), sin(v)];
```

and it is defined in rectangular coordinates by the following equation:

```
>>  cyleqn = (y^2 + z^2 == 1);
```

A picture of the two surfaces is shown in Figure 8.10.

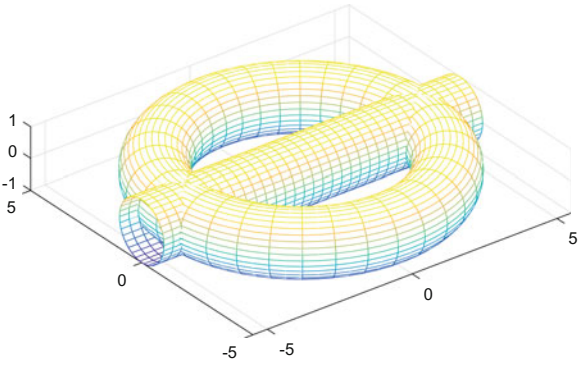


Fig. 8.10 Intersecting a torus and a cylinder

The task before us is to write down a triple integral that computes the volume of the solid obtained by intersecting the torus and the cylinder. The hardest part of this task is choosing the order (and finding the limits) of integration. We plan to create a tool for visualizing the solid that will help us make that choice.

Here is an idea for doing that. Suppose, for example, that we replace x by a constant value in the equations that define the cylinder and the torus. This algebraic substitution corresponds to the geometric operation of intersecting both solids with a plane. In this case, the plane lies at right angles to the cylinder, so it intersects it in a circle. The intersections with the torus will vary. By plotting the intersections for various values of x, we can get a good idea of the different cross sections of the intersection of the two solids. If the cross sections are simple, then x is a good choice for the final variable in the order of integration. If the cross sections are complicated, then we should try again with a different variable instead of x. The following command draws six cross sections for values of x between 3 and 5. The command **`subplot`** distributes them sequentially into "subplots" in a 3×2 grid.

```
>> for j=0:5
  subplot(3,2,j+1)
  fimplicit(cyleqn, 'r', [-1, 1, -1, 1]), hold on
  fimplicit(subs(toreqn, x, 3+2*j/5), 'b', [-1, 1, -1, 1])
  hold off, axis equal
end
```

The output of this command is shown in Figure 8.11.

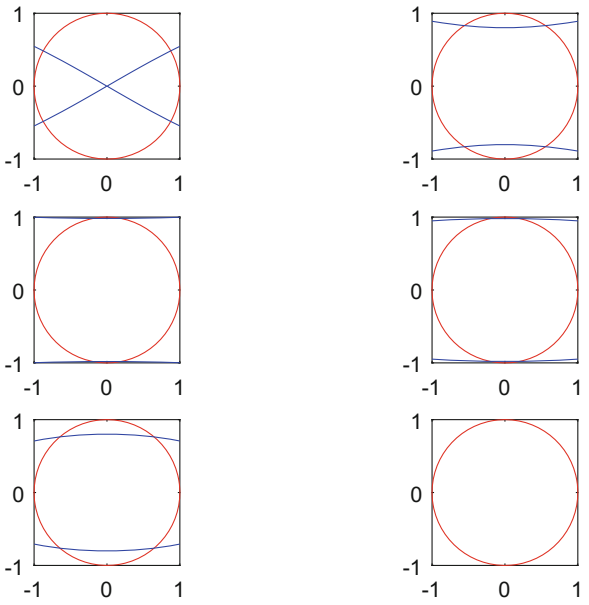


Fig. 8.11 Intersecting a torus, a cylinder, and a plane of constant x

We have taken the equations of the cylinder and the torus in rectangular coordinates and substituted values of x of the form $3 + 0.4j$, where j runs from 0 to 5. These equations are plotted with **`fimplicit`**. The intersection of the plane with the cylinder is drawn in dotted red and is always a circle of radius 1; the intersection of the plane with the torus is drawn in blue and has varying shape. Since the intersections change drastically in appearance, we can make a decision: we do **not** want to save the x variable for last when we integrate. The variation in cross sectional shapes makes it extremely difficult to describe the region simply. We should instead look at the cross sections we get by holding one of the other variables constant. For example, if we slice through the solid with the planes corresponding to $y =$ constant, we instead get the following:

```
>> for j=0:7
  subplot(4,2,j+1)
  zval = sqrt(1-(-1+2*j/7)^2);
  plot([-5 -5; 5 5], [zval -zval; zval -zval], 'r'), hold on
  fimplicit(subs(toreqn,y,-1+2*j/7), 'b', [-5, 5, -1, 1])
  hold off, axis equal
end
```

Note that with y constant, in the equation of the cylinder, x is unconstrained and $z = \pm\sqrt{1 - y^2}$. So the intersection of the cylinder and the plane consists of two lines. That explains the two lines of code beginning with **`zval`** and **`plot`**. The output is shown in Figure 8.12.

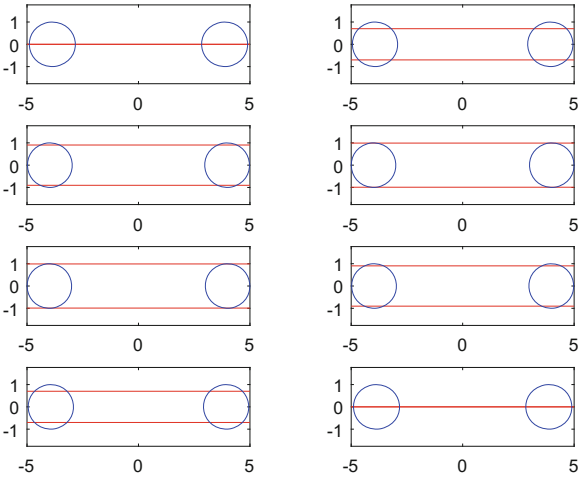


Fig. 8.12 Intersecting a torus, a cylinder, and a plane of constant y

This time all the cross sections look basically the same. The intersection of the torus and the plane is a union of two ovals, shown in blue, and the intersection of the cylinder and the plane consists of two horizontal lines (in the x-z plane), shown in red.

Now that we have figured out the best order of integration, we can go ahead and solve for the limits of integration. We know that we want the y-integral on the outside, with y ranging from -1 to 1. Since the equation of the cylinder doesn't constrain x at all, we want to solve for x in terms of y and z in the equation of the torus to get the limits of integration for the innermost integral (over x). Then the limits of integration for z in terms of y are given by the equation of the cylinder. Below, we suppress various warning messages that come from worry about when one needs to take the square root of a negative number.

```
>>  xsol = solve(toreqn, x)

xsol =
  (17 - z^2 - 8*(1 - z^2)^(1/2) - y^2)^(1/2)
  (8*(1 - z^2)^(1/2) - z^2 - y^2 + 17)^(1/2)
 -(17 - z^2 - 8*(1 - z^2)^(1/2) - y^2)^(1/2)
 -(8*(1 - z^2)^(1/2) - z^2 - y^2 + 17)^(1/2)
```

The two negative solutions correspond to the intersections to the left of the origin; the other two correspond to the intersections to the right of the origin. By symmetry, we only need the piece on the right. By substituting $y = z = 0$, we can check which solution goes on the bottom.

```
>>  subs(xsol, [y, z], [0, 0])

ans =
  3   5  -3  -5
```

```
>> xlo = matlabFunction(xsol(1)); xhi = matlabFunction(xsol(2));
>> zlo = matlabFunction(-sqrt(1-y^2));
>> zhi = matlabFunction(sqrt(1-y^2));
```

Now, we can use **viewSolid** to draw the picture (of the piece of the solid region with $x > 0$). The picture is shown in Figure 8.13

```
>>  viewSolid('y', -1, 1, 'z', zlo, zhi, 'x', xlo, xhi)
```

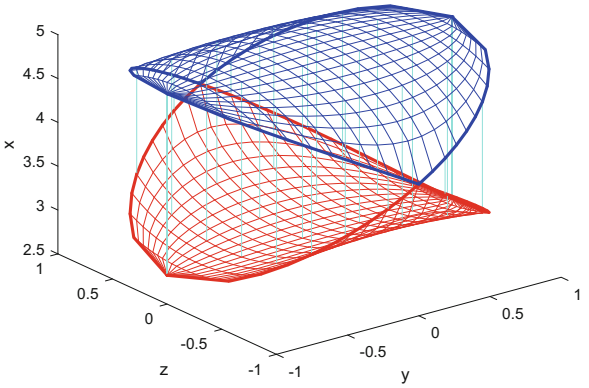


Fig. 8.13 The solid region bounded by a torus and a cylinder

Finally, the volume of the total intersection is given by the integral:

$$2\int_{-1}^{1}\int_{\text{zlo}}^{\text{zhi}}\int_{\text{xlo}}^{\text{xhi}} 1\,dx\,dz\,dy.$$

MATLAB cannot evaluate this integral symbolically, so we will compute it numerically. Note that to use MATLAB's numerical integration routines, the integrand must be a function handle producing a vector output of the same size as the vector inputs. That is the reason for the use of **`ones`** and **`size`** below.

```
>>  vol = 2*integral3(@(x,y,z) ones(size(x)),-1,1,zlo,zhi,xlo,xhi)

vol =
   10.7633
```

Exercise 8.2. Estimate the volume using the picture in Figure 8.13 and explain why this answer is reasonable.

8.5 Cylindrical Coordinates

The problem we have just solved (computing the volume of the intersection of a solid torus with a solid cylinder) can also be done, perhaps more naturally, in cylindrical coordinates. Recall that this means we use polar coordinates in the x-y plane along with the vertical coordinate z.

We briefly indicate how to solve the problem this way. To begin, we need the equations of the torus and cylinder in cylindrical coordinates. The equation of the

torus came to us in the form $(r-4)^2+z^2=1$, but we need to convert the equation of the cylinder.

```
>> newtoreqn = ((r-4)^2 + z^2 == 1);
>> newcyleqn = subs(cyleqn, y, r*sin(theta))

newcyleqn =
r^2*sin(theta)^2 + z^2 == 1
```

Note that the equation of the torus imposes no constraints at all on the angular variable θ, while the equation of the cylinder tells us that in the interior of the cylinder, $|r\sin\theta| \le 1$, or $|\sin\theta| \le (1/r)$. We also know from the equation of the torus that $|r-4| \le 1$, or that $3 \le r \le 5$. By symmetry, the volume we want is 8 times the volume of the portion of the region of intersection in the first octant. In this portion of the region, for fixed r, θ ranges from 0 to $\arcsin(1/r)$. Then for fixed r and θ, z ranges from 0 to the smaller of the two z-values given by the equations of the torus and the cylinder. Recall that in cylindrical coordinates, we need to include a factor of r in the integrand. So, the desired volume is

```
>> 8*integral3(@(r, theta) r, 3, 5, @(r) zeros(size(r)), ...
    @(r) asin(1./r), @(r, theta) zeros(size(r)), @(r, theta) ...
    sqrt(min(1-(r-4).^2, 1-r.^2.*sin(theta).^2)))

ans =
   10.7633
```

This is the same answer that we got before, at least to four decimal places.

8.6 More General Changes of Coordinates

Some multiple integrals can be computed more easily by changing from the standard rectangular coordinates to another coordinate system. The best-known other systems are polar, cylindrical, and spherical coordinates. But in this section, we explain how to use MATLAB to set up and evaluate integrals involving a more general change of coordinates.

For simplicity, we will confine our discussion to two dimensions. We may want to change variables in the two-dimensional integral

$$\iint_R h(x,y)\,dx\,dy$$

for one of two reasons: either to simplify the *integrand*, or to simplify the *limits of integration*. While both are valid reasons for wanting to use a change of coordinates, *simplifying the limits of integration is usually more compelling than simplifying the integrand.* For example, let us try to compute the area of the region R in the first quadrant enclosed by the two hyperbolas

$$x^2 - y^2 = 1, \qquad x^2 - y^2 = 4,$$

and the two parabolas

$$y = 9 - x^2, \qquad y = 16 - x^2.$$

A picture of the region is shown in Figure 8.14.

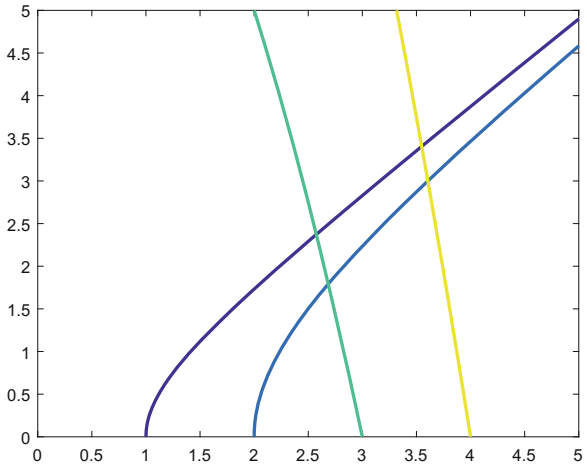


Fig. 8.14 The region bounded by a two hyperbolas and two parabolas

It is clear that solving the problem in rectangular coordinates—either vertically or horizontally—will require dividing the region into three separate pieces. There is an alternative, however. The bounding curves in this example come in pairs. Each pair consists of two level curves of some function of two variables. We can make a change of variables using those auxiliary functions:

$$u = x^2 - y^2,$$
$$v = x^2 + y.$$

When we change variables like this, the formula for the area of the region becomes

$$\text{area}(R) = \iint_R dx\, dy = \iint_S |J(u, v)|\, du\, dv,$$

where S is the region in the u-v plane that R transforms into under the change of variables, and $J(u, v)$ is the Jacobian determinant that can be computed two different ways:

$$J(u, v) = \det \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial y} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{pmatrix} = \det \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{pmatrix}^{-1}.$$

Three steps are required to compute the integral with this change of coordinates. First, we need to describe the region S. In our example, we chose coordinates to make that task particularly easy: S is the rectangle $1 \le u \le 4, 9 \le v \le 16$. Next, we need to compute the determinant of the Jacobian matrix.

```
>> f = x^2 - y^2; g = x^2 + y;
>> jake = det(jacobian([f, g], [x, y]))

jake =
2*x + 4*x*y
```

Finally, we need to rewrite the inverse of the Jacobian in terms of the variables u and v. So, we need to find formulas for x and y. Since $f = u$ and $g = v$, $g - f = v - u$, and this is a function of y only, so we can solve for y.

```
>>  backsoly = solve(g - f == v - u, y)

backsoly =
 - (4*v - 4*u + 1)^(1/2)/2 - 1/2
   (4*v - 4*u + 1)^(1/2)/2 - 1/2
```

Since y is positive, we want the second solution.

```
>>  backsoly = backsoly(2); backsolx = sqrt(v - backsoly)

backsolx =
(v - (4*v - 4*u + 1)^(1/2)/2 + 1/2)^(1/2)
```

Now we substitute this into the formula for the Jacobian and integrate. MATLAB cannot compute the integral symbolically, but it can still get a numerical answer.

```
>> jacobianuv = subs(jake^(-1), [x, y], [backsolx, backsoly]);
>> area = integral2(matlabFunction(jacobianuv), 1, 4, 9, 16)

area =
   0.5442
```

Note by the way that this is not the only change of variables that will work in this problem, although it is the simplest. Another possibility is to take $x = u \cosh v$, $y = u \sinh v$, since then $x^2 - y^2 = u^2(\cosh^2 v - \sinh^2 v) = u^2$ and the two hyperbolas become the vertical lines $u = 1$, $u = 2$. We leave it to the reader to work out the integral in this coordinate system and to show that one gets the same answer as before. In this choice of coordinates, the integrand (which is the Jacobian factor) becomes quite simple but the limits of integration for v are a bit complicated.

Problem Set H. Multiple Integrals

Problem 8.1. Each part of the following problem contains a double integral. In each case, you should first use the **verticalRegion** program from Chapter 8 to plot the (shaded) region over which the integral extends. Then compute the value of the integral.

(a) $\displaystyle\int_1^2 \int_0^x y\sqrt{x^2 - y^2}\, dy\, dx.$

(b) $\displaystyle\int_0^1 \int_{x^2}^x xy\, dy\, dx.$

(c) $\displaystyle\int_1^e \int_{\ln(x)/x}^{1/x} e^{xy}\, dy\, dx.$

(d) $\displaystyle\int_{1.6}^{4.7} \int_{\tan x}^{e^x} \cos e^x\, dy\, dx.$

Problem 8.2. Each part of this problem contains a double integral. In each case, you should first convert to polar coordinates. Next, use the **polarRegion** program from Chapter 8 to plot the (shaded) region over which the integral extends. Finally, compute the value of the integral.

(a) $\iint_R x^2 y^2\, dA$, where R is the inside of the cardioid $r = 2(1 + \sin\theta)$.

(b) $\iint_R \cos(x^2+y^2)\, dA$, where R is defined by $0 \le \theta \le \pi/2$ and $0 \le r \le \sin^2\theta$.

(c) $\displaystyle\int_{3/\sqrt{2}}^3 \int_0^{\sqrt{9-x^2}} \left(x^2 + y^2\right)\, dy\, dx.$

(d) $\displaystyle\int_0^1 \int_{\sqrt{x-x^2}}^{\sqrt{1-x^2}} (x^2 + y^2)^{3/2}\, dy\, dx.$

Problem 8.3. A horizontally simple region R is bounded below by the horizontal line $y = c$, above by the horizontal line $y = d$, on the left by the curve $x = f(y)$, and on the right by the curve $x = g(y)$. It is natural to integrate over horizontally simple regions with respect to x first, and then y.

(a) Write a program called **horizontalRegion** that takes f, g, c, and d as its arguments. The output of this program should be a graph that shades the region R with horizontal lines. (Hint: Modify the **verticalRegion** program from the chapter by using **fimplicit** in place of **fplot**, where you view the curve $x = f(y)$ as given implicitly.)

(b) Use the **horizontalRegion** program to shade the regions over which each of the following integrals is being computed; then evaluate the integrals:

(i) $\displaystyle\int_0^2 \int_{y/2}^{y} e^{x^2}\,dx\,dy.$

(ii) $\displaystyle\int_0^1 \int_{-\sqrt{1-y^2}}^{\sqrt{1-y^2}} yx^4\,dx\,dy.$

(iii) $\displaystyle\int_{-2}^2 \int_{1-\sqrt{2-y}}^{y/2} xe^x\,dx\,dy.$

Problem 8.4. Each of the following double integrals extends over a region that is both vertically simple and horizontally simple. In each case, you should:

- Plot the shaded region corresponding to the integral as given.
- Compute the value of the given integral.
- Reverse the order of integration. Plot the new shaded region.
- Compute the value of the integral with the order of integration reversed.

(a) $\displaystyle\int_0^1 \int_{\sqrt[3]{y}}^1 \sqrt{1+x^4}\,dx\,dy.$

(b) $\displaystyle\int_0^{\pi/2} \int_0^{\cos x} x^2\,dy\,dx.$

(c) $\displaystyle\int_{-5}^4 \int_{2-\sqrt{4-y}}^{(y+2)/3} e^{x^2-4x+y}\,dx\,dy.$

Problem 8.5. Each part of this problem describes a region in the plane. Plot the indicated region. Then find the area. You may need to use **`solve`** or **`fzero`** to find the points of intersection of the curves.

(a) The region bounded by the parabolas $y = 3x^2$ and $y = 5x - 2x^2$.

(b) The region inside the ellipse $4x^2 + y^2 = 4$.

(c) The total region bounded by the curves $y = \cos x$ and $y = 4x^2 - x^4$.

(d) The entire region between the curves $y = e^{-x^2}\cos 10x$ and $y = 1 - x$.

Problem 8.6. Each part of this problem describes a region in the plane described in polar coordinates. Plot (and shade) the indicated region. Then find the area. You may need to use **`solve`** or **`fzero`** to find the points of intersection of the curves.

(a) The region inside the rose $r = 2\sin 3\theta$.

(b) The region inside the lemniscate $r^2 = 2\cos 2\theta$ and outside the circle $r = 1$.

(c) The region inside the cardioid $r = 1 + \cos\theta$ and outside the circle $r = 1/2$.

(d) The region inside the circle of radius 8 and outside the loop of the curve defined implicitly by $8y^2 = x^3 + 8x^2$.

Problem 8.7. Each part of the following problem describes a region R and a function $f(x, y)$. In each case, you should graph the shaded region, and then compute the value of the integral

$$\iint_R f(x, y)\, dA.$$

You must decide: (i) whether to use rectangular or polar coordinates; and (ii) in which order to integrate. Your shaded graph should coincide with the chosen method for evaluating the integral.

(a) $f(x, y) = e^{7y-x}$ and R is the region bounded by $y^2 = x-12$ and $y^2 = 30-x$.

(b) $f(x, y) = 1/\sqrt{x^2 + y^2}$ and R is the region to the right of $x = 0$ which is inside the ellipse $x^2 + 4y^2 = 9$ and outside the circle $(x - 1)^2 + y^2 = 1$.

(c) $f(x, y) = x^2 + y^2$ and R is the bounded region between the graphs of $y = \sin x$ and $3y + x = 3$.

Problem 8.8. This problem investigates the Gauss quadrature formula derived in Section 8.2.1,

$$Q(f) = \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right) + \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right),$$

which approximates $\int_{-1}^{1} f(x)\, dx$ for f defined on $[-1, 1]$. Compute $Q(f)$ for $f(x) = \cos\left(\frac{\pi}{2} x\right)$ and compare with the Simpson's rule approximation

$$Q_{\text{Simp}}(f) = \frac{1}{3}\big(4f(0) + f(1) + f(-1)\big).$$

Which is more accurate?

Problem 8.9. Derive a Gauss quadrature formula of the form

$$Q(f) = 4(1 - c) f(0, 0) + cf(a, a) + cf(a, -a) + cf(-a, a) + cf(-a, -a),$$

which approximates $\int_{-1}^{1} \int_{-1}^{1} f(x, y)\, dy\, dx$ for f defined on the square $[-1, 1] \times [-1, 1]$. Choose the constants $0 < c < 1$ and $0 < a < 1$ so that the quadrature formula is exact for the functions x^2, y^2, x^4, and y^4. How good is it for the function x^2y^2? For the function $\cos\left(\frac{\pi}{2}x\right) \cos\left(\frac{\pi}{2}y\right)$?

Problem 8.10. Derive a Gauss quadrature formula of the form

$$Q(f) = (\pi - 4c) f(0, 0) + cf(a, 0) + cf(a, \tfrac{\pi}{2}) + cf(a, \pi) + cf(a, \tfrac{3\pi}{2}),$$

which approximates $\int_0^{2\pi} \int_0^1 f(r, \theta)\, r\, dr\, d\theta$ for f defined in terms of polar coordinates on the disk $r \le 1, 0 \le \theta \le 2\pi$. Choose the constants $0 < c < \frac{\pi}{4}$ and $0 < a < 1$

so that the quadrature formula is exact for the functions $r^m e^{i n\theta}$ for $(m, n) = (0, 0)$, $(1, \pm 1)$, $(1, 0)$, and $(2, 0)$. How good is it for the function $f(r, \theta) = r^2 - 2r\cos\theta$? For the function $f(r, \theta) = r^3$?

Problem 8.11. Use the **`viewSolid`** program from Chapter 8 to sketch the region over which each of the following triple integrals extends, and then evaluate the integrals.

(a) $\displaystyle\int_0^1 \int_{x^3}^{x^2} \int_0^{(x+y)/4} z\sqrt{x}\, dz\, dy\, dx.$

(b) $\displaystyle\int_0^1 \int_0^{\sqrt{1-y^2}} \int_{x^2+y^2-1}^{1-x^2-y^2} e^y \sqrt{1-y^2}\, dz\, dx\, dy.$

(c) $\displaystyle\int_0^{\pi/6} \int_0^z \int_0^{y+z} \left(1 + y^2 z^3 \sin(xz)\right)\, dx\, dy\, dz.$

(d) $\displaystyle\int_0^1 \int_0^{x^2} \int_0^{x^3-y^3} x^2 y^2 e^z\, dz\, dy\, dx.$

Problem 8.12. In each part of this problem, use the **`viewSolid`** program from Chapter 8 to sketch the region D over which the integral extends, and then evaluate the triple integral. You should be able to determine the limits of integration directly from the description of D. In (c) and (d), it may help to think about planes that contain three of the given points.

(a) $\iiint_D xyz\, dV$, where D is the region in the first octant bounded above by the hemisphere $z = \sqrt{4 - x^2 - y^2}$ and on the sides and the bottom by the coordinate planes.

(b) $\iiint_D xz\, dV$, where D is the region in the first octant bounded above by the sphere $x^2 + y^2 + z^2 = 9$, below by the plane $z = 0$, and on the sides by the planes $x = 0$, $y = 0$, and the cylinder $x^2 + y^2 = 4$.

(c) $\iiint_D yz^2 e^{xy}\, dV$, where D is the prism with vertices $(0, 0, 1)$, $(0, 1, 2)$, $(0, 1, 0)$, $(1, 0, 1)$, $(1, 1, 2)$, and $(1, 1, 0)$.

(d) $\iiint_D \log(z)/z\, dV$, where D is the tetrahedron with vertices at $(1, 0, 2)$, $(1, 1, 2)$, $(1, 1, 1)$, and $(2, 0, 2)$.

Problem 8.13. Each part of this problem describes a solid region. Graph the region. Based on the graph, estimate the volume. (Do not just write down a number; give a brief plausibility argument.) Then compute an integral to find the actual volume.

(a) The solid region between the saddle $z = xy/2$ and the cone $z^2 = x^2 + 2y^2$ that lies above the first quadrant of the disk whose boundary is $x^2 + y^2 = 1$.

(b) The solid region bounded above by the graph of $z = y^2/4$, and whose base in the x-y plane is the bounded region between the curves $x = \sin y$ and $x = 3y - y^2$.

(c) A right pyramid with height h and a square base having sides of length s. (Hint: Place the base of the pyramid in the x-y plane, with center at the origin and vertices on the axes. For purposes of plotting a sample, pick specific values of s and h. Using symmetry, just compute the volume of the part in the first octant.)

(d) The region bounded above by the graph of the function $z = \frac{1}{2} + x^2 - y^2$, on the sides by the cylinder $x^2 + y^2 = x$, and below by the x-y plane.

Problem 8.14. This problem is concerned with regions defined in cylindrical coordinates.

(a) Modify the **`viewSolid`** program from Chapter 8 to construct a function script called **`viewCylindrical`**. The arguments to this function should be (except for the integrand) the same as those needed to compute the triple integral

$$\int_{\alpha}^{\beta} \int_{f(\theta)}^{g(\theta)} \int_{F(r,\theta)}^{G(r,\theta)} h(r,\theta,z) r \, dz \, dr \, d\theta.$$

(b) Use your program to sketch the regions over which each of the following integrals extends. Describe the resulting regions in words, and compute the volumes.

(i) $\displaystyle\int_0^{\pi} \int_0^{1} \int_0^{\sqrt{1-r^2}} r \, dz \, dr \, d\theta.$

(ii) $\displaystyle\int_0^{2\pi} \int_0^{1} \int_0^{r} r \, dz \, dr \, d\theta.$

(iii) $\displaystyle\int_0^{2\pi} \int_0^{3} \int_{1-(r^2/9)}^{\sqrt{r^2+4}} r \, dz \, dr \, d\theta.$

Problem 8.15. Each part of this problem presents a triple integral in cylindrical coordinates. Sketch the region over which the integral extends, and then evaluate the integral.

(a) $\displaystyle\int_0^{2\pi} \int_0^{1} \int_r^{1} z r^3 \cos^2\theta \, dz \, dr \, d\theta.$

(b) $\displaystyle\int_0^{2\pi} \int_0^{1} \int_{-\sqrt{4-r^2}}^{\sqrt{4-r^2}} z^2 r \, dz \, dr \, d\theta.$

(c) Find the volume of the region under the cone $z = 6 - r$ whose base is bounded by the cardioid $r = 3(1 + \cos\theta)$.

Problem 8.16. This problem is concerned with regions defined via spherical coordinates. Note that labeling conventions for spherical coordinates vary from one author to another. Here we shall generally follow the convention that ρ is the distance to the origin, θ is the angle around the z-axis (as in cylindrical coordinates), and ϕ is the

colatitude, that is, the angle down from the positive z-axis. The relations between these coordinates and Cartesian coordinates are

$$\begin{aligned} x &= \rho \cos\theta \sin\phi, \\ y &= \rho \sin\theta \sin\phi, \\ z &= \rho \cos\phi, \end{aligned} \tag{8.3}$$

and the Jacobian factor needed when integrating is $\rho^2 \sin\phi$.

(a) Modify the **`polarRegion`** program from Chapter 8 to construct a program called **`viewSpherical`**. The arguments to **`viewSpherical`** should be (except for the integrand) the same as those needed to compute the triple integral

$$\int_a^b \int_{f(\theta)}^{g(\theta)} \int_{F(\theta,\phi)}^{G(\theta,\phi)} \rho^2 \sin\phi \, d\rho \, d\phi \, d\theta.$$

(Since labeling conventions differ, you should allow the user to choose the names for the coordinates.)

(b) Use your program to sketch the regions over which each of the following integrals extends, and compute the volumes.

(i) $\displaystyle\int_0^{\pi/2} \int_0^{\pi/2} \int_1^3 \rho^2 \sin\phi \, d\rho \, d\phi \, d\theta.$

(ii) $\displaystyle\int_0^{2\pi} \int_0^{\pi/4} \int_1^{2\sec\phi} \rho^2 \sin\phi \, d\rho \, d\phi \, d\theta.$

(iii) $\displaystyle\int_0^{2\pi} \int_0^{\pi/6} \int_0^{\sqrt{\sec(2\phi)}} \rho^2 \sin\phi \, d\rho \, d\phi \, d\theta.$

Problem 8.17. Each of these problems presents a triple integral in spherical coordinates. Sketch the region over which the integral extends, and then evaluate the integral. Here ρ is the distance to the origin, ϕ is the colatitude, and θ is the angular variable in the x-y plane.

(a) $\displaystyle\iiint_D \left(x^2 + y^2\right) dV$, where D is the ball $x^2 + y^2 + z^2 \le 1$.

(b) The volume inside the cone $3(x^2 + y^2) = z^2$ and below the upper sheet of the hyperboloid $z^2 - x^2 - y^2 = 1$.

(c) The volume of a "slice of an apple," that is, the smaller of the two regions bounded by the sphere $\rho = 3$ and the half-planes $\theta = 0$ and $\theta = \pi/6$.

(d) The volume of the solid region bounded below by the x-y plane and above by the surface $\rho = 1 + \cos\phi$.

Problem 8.18. You can use multiple integrals to compute surface areas. Suppose $\mathbf{\Sigma}$ is a surface patch in the sense of Chapter 6, *Geometry of Surfaces*, defined by a parametrization $\sigma : D \to \mathbb{R}^3$. Then the area of $\mathbf{\Sigma}$ is:

$$\text{area}(\mathbf{\Sigma}) = \iint_D J(u,\, v)\, du\, dv,$$

where the Jacobian factor $J(u,\, v)$ measures the local stretching done by σ. The formula for J is derived as follows. As u ranges between u_0 and $u_0 + \Delta u$ and as v ranges between v_0 and $v_0 + \Delta v$, $\sigma(u,\, v)$ approximately traces out a parallelogram, with vertices at $\sigma(u_0,\, v_0)$, $\sigma(u_0,\, v_0) + \sigma_u(u_0,\, v_0)\Delta u$, $\sigma(u_0,\, v_0) + \sigma_v(u_0,\, v_0)\Delta v$, and at $\sigma(u,\, v) + \sigma_u(u_0,\, v_0)\Delta u + \sigma_v(u_0,\, v_0)\Delta v$. The area formula in Section 2.1.1.4 gives the area of this parallelogram as $\|\sigma_u(u_0,\, v_0)\Delta u \times \sigma_v(u_0,\, v_0)\Delta v\|$. So, adding the areas and passing to the limit yields the formula:

$$J(u,\, v) = \|\sigma_u \times \sigma_v\|;$$

in other words, the Jacobian factor equals the length of the normal vector $\mathbf{N}$ that we studied in Chapter 6.

(a) Show that the formula for $J(\phi,\, \theta)$ reduces to $a^2 \sin\phi$ in the case of a piece of a sphere of radius a parametrized by spherical coordinates ϕ and θ:

$$\sigma(\phi,\, \theta) = (a\cos\theta\sin\phi,\, a\sin\theta\sin\phi,\, a\cos\phi).$$

(b) Show that the formula for $J(x,\, y)$ reduces to $\sqrt{f_x^2 + f_y^2 + 1}$ in the case of a Monge patch parametrization $\sigma(x,\, y) = (x,\, y,\, f(x, y))$ with parameters x and y.

(c) Show that the formula for $J(z,\, \theta)$ reduces to $r(z)\sqrt{1 + r'(z)^2}$ in the case of a surface of revolution parametrized by cylindrical coordinates z and θ:

$$\sigma(z,\, \theta) = (r(z)\cos\theta,\, r(z)\sin\theta,\, z).$$

(d) Sketch each of the following surfaces and compute the surface area:

(i) The portion of the paraboloid $z = 9 - x^2 - y^2$ above the x-y plane.

(ii) The portion of the sphere $x^2 + y^2 + z^2 = 4$ that is above the x-y plane and inside the cylinder $x^2 + y^2 = 1$.

(iii) The portion of the sphere $x^2 + y^2 + z^2 = 9$ that is inside the paraboloid $x^2 + y^2 = 8z$.

(iv) The torus whose equation in cylindrical coordinates is $(r-4)^2 + z^2 = 1$.

Problem 8.19. This problem is concerned with the solid region D bounded by the surface

$$x^{2/3} + y^{2/3} + z^{2/3} = 1.$$

Because the solid is symmetric, we will only look at the portion D lying in the first octant.

(a) Graph the projection of D onto the first quadrant of the x-y plane.

(b) Find the area of the projected region.

(c) Graph the solid D.

(d) From the graph, estimate the volume of D.

(e) Finally, compute the volume of the solid D. MATLAB cannot symbolically compute the volume integral that you get from rectangular coordinates. Make a change of variables to find the volume. (Hint: setting $u = x^{1/3}$, $v = y^{1/3}$, $w = z^{1/3}$ converts D to a spherical ball.) Then check your answer using numerical integration in rectangular coordinates.

Problem 8.20. In this problem, we consider the intersection of the two cylinders $x^2 + y^2 = 1$ and $x^2 + z^2 = 1$.

(a) Graph the two cylinders on the same axes.

(b) Use the methods of Chapter 8 to draw cross sections of the region bounded by the two cylinders. Based on your results, choose an order of integration and compute the volume of the region.

Problem 8.21. In this problem, we consider the intersection of the cylinder $y^2 + (z-2)^2 = 1$ with the cone $z^2 = x^2 + y^2$.

(a) Graph both surfaces on the same axes.

(b) Use the methods of Chapter 8 to draw cross sections of the region bounded by the cylinder and the cone. Based on your results, choose an order of integration and compute the volume of the region. You may need to do the integral numerically.

Problem 8.22. In this problem, you are asked to compute some areas that are either difficult or impossible to compute in rectangular coordinates. In each part, first draw a picture of the region being discussed. Then compute the area of the region using integration in an appropriate coordinate system.

(a) The region in the first quadrant of the x-y plane that lies between the hyperbolas $xy = 1$ and $xy = 2$ and between the lines $y = x$ and $y = 4x$.

(b) The region in the first quadrant of the x-y plane that lies between the hyperbolas $x^2 - y^2 = 1$ and $x^2 - y^2 = 2$ and between the circles $x^2 + y^2 = 9$ and $x^2 + y^2 = 16$.

Problem 8.23. Sketch the region in the first quadrant of the x-y plane that lies between the circles $x^2 + y^2 = \frac{1}{9}$ and $x^2 + y^2 = \frac{4}{9}$ and between the two parabolas $y = x^2$ and $y = \sqrt{x}$. Then compute the area of the region using a double integral in an appropriate coordinate system. (Hint: You can use the symmetry of the region in computing the area.)

Problem 8.24. In this problem, we will introduce another numerical algorithm, called the *Monte Carlo method*, that can be used to estimate the values of complicated multiple integrals. The reason for the name is that the algorithm relies on a random process somewhat akin to the spinning of a roulette wheel. To illustrate the method, we will use it to estimate $\operatorname{vol} R = \iiint_R dV$, where R is a possibly complicated region in $\mathbb{R}^3$. The first step is to enclose R in a rectangular box, say

$$B = \{x_0 \le x \le x_1,\ y_0 \le y \le y_1,\ z_0 \le z \le z_1\}.$$

Then the volume of R can be computed as

$$\operatorname{vol} R = p \operatorname{vol} B = p(x_1 - x_0)(y_1 - y_0)(z_1 - z_0),$$

where p is the fraction of the volume of B occupied by the region R.

What makes the method work is that the fraction p can be interpreted as the probability that a point chosen "at random" in B will lie in R. So if we choose a very large number n of points "randomly" in B (by picking x to be a random number in the interval $[x_0, x_1]$, y to be a random number in the interval $[y_0, y_1]$, and z to be a random number in the interval $[z_0, z_1]$), the Law of Large Numbers says that the number of points that land in R, divided by the number of trials n, will be a good approximation to p. In fact, one can also show, using the Central Limit Theorem in probability theory, that the error in the approximation should behave like $1/\sqrt{n}$. This means convergence to the true answer is somewhat slow, as numerical methods go, but even when R is complicated, the method at least gives a reasonable estimate of the value of the triple integral. Here is a MATLAB program for implementing the algorithm:

```
function vol = volMonteCarlo(region,x0,x1,y0,y1,z0,z1,n)
%VOLMONTECARLO estimates the volume of a region in 3-space.
% VOLMONTECARLO(region,x0,x1,y0,y1,z0,z1,n) estimates
% the volume of region between the given limits for x, y, and z
% with a Monte Carlo method with n trials.  The region should
% be specified by a function that takes a matrix with 3 columns
% and outputs a column vector with a 0 if a certain row
% is not in the region and a 1 (or 1 multiplied by an integrand)
% if the row is in the region.  For example one could take
% sphere = @(X) heaviside(ones(size(X,1),1) ...
% - X(:,1).^2 - X(:,2).^2 - X(:,3).^2)

X = rand(n,3);  % random numbers between 0 and 1, now rescale
X(:,1)= x0+(x1-x0)*X(:,1);
X(:,2)= y0+(y1-y0)*X(:,2);
```

```
X(:,3)= z0+(z1-z0)*X(:,3);
% At this point each row of X is in the desired range.
vol = sum(region(X))*(x1-x0)*(y1-y0)*(z1-z0)/n;
end
```

Here, the command **rand(3, n)** produces a $3 \times n$ matrix with random entries in $(0, 1)$. Here is a sample usage of the program. Typing

```
sphere = @(X) heaviside(ones(size(X,1),1) ...
  - X(:,1).^2 - X(:,2).^2 - X(:,3).^2);
volMonteCarlo(sphere,-1,1,-1,1,-1,1,100000)
```

yielded on one try the answer 4.2278, compared with the true value of the volume, which is $\frac{4}{3}\pi = 4.1888$.

(a) Run the program a few more times with different values of n to estimate the volume of (the region bounded by) a sphere of radius 1. Compare with the numerical value of the exact answer, $4\pi/3$, and study how the error behaves with n. (Warning: Do not take n TOO large with this program or it will take an unmanageable amount of time. $n = 10^8$ is about the upper limit on most computers.)

(b) Run the program several times with different values of n to estimate the volume of the region bounded by the astroidal sphere of radius 1:

$$R = \left\{x^{2/3} + y^{2/3} + z^{2/3} \leq 1\right\}.$$

(As you may have noticed in Problem 8.19, this is the sort of problem where numerical algorithms can be useful, because the integral for the volume of R is quite hard to evaluate, even numerically, without a change of variables.) Just one word of caution: fractional exponents can lead to trouble in MATLAB, since they are evaluated as complex numbers. To avoid difficulties, take the absolute value of each fractional power. Check your answer for the volume against the results of the exact calculation with the change of variables $x = u^3$, $y = v^3$, $z = w^3$ from Problem 8.19.

(c) By changing the definition of **region**, you can modify the same program to compute more complicated triple integrals of the form $\iiint_R f(x, y, z)\, dV$. Use the program to estimate the mass of a sphere of radius 1 with mass density $\delta(x, y, z) = x^2y^2z^2$. Check your answer for the mass against the results of an exact calculation in spherical coordinates. (If you do this right you will see a curious connection with part (b).)

(d) Use the Monte Carlo method to estimate the volume of the region inside the sphere $x^2 + y^2 + z^2 = 4$ and between the two sheets of the hyperboloid $z^2 - (x^2 + y^2) = 1$.

Glossary of MATLAB Commands

`heaviside` A step function that is $+1$ for $x > 0$ and 0 for $x < 0$

`int` Computes a symbolic integral

`integral2` Computes a numerical integral in two variables

`integral3` Computes a numerical integral in three variables

`jacobian` Finds the gradient of a function of several variables

`matlabFunction` Converts a symbolic expression to a vectorized numerical function

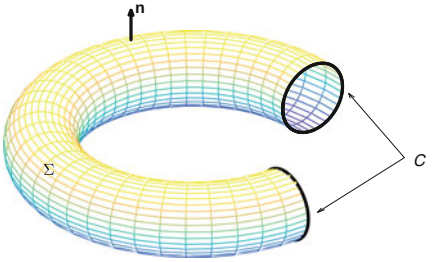
`ones` Creates a matrix of all ones

`rand` Generates a matrix of uniformly distributed random numbers

`size` Gets the dimensions of a matrix

`zeros` Creates a matrix of all zeros

Chapter 9
Multidimensional Calculus



The development of one-variable calculus in the seventeenth and eighteenth centuries was a tremendous boon to man's understanding of the laws of the earth and the universe. Over the centuries, humans have learned to model the phenomena they see around them via mathematical equations. The components of these equations are functions that stand in for the observed quantities. And more often than not, the evolution of a system is expressed in terms of rates of change, that is derivatives of the functions that portray the quantities involved. Thus, one is led to differential equations.

Practitioners often speak of the methods of solving a differential equation as "integrating the equation." Thus, it is not at all surprising that both differential calculus and integral calculus have played a fundamental role in the development of classical (and modern) mathematics. Part of the reason for their success is the inherent relationship between the two processes—expressed in the most important theorem in one-variable calculus: The *Fundamental Theorem of Calculus* [FTC]. Here it is:

Theorem 9.1. (FTC). *Let $f(t)$ be a continuous function on an interval $a \leq t \leq b$. Define a new function $F(t) = \int_a^t f(s)\,ds$. Then F is differentiable on (a, b) and $F'(t) = f(t)$, $a < t < b$. Moreover, if $G(t)$ is any other differentiable function satisfying $G'(t) = f(t)$, $a < t < b$ (sometimes called a* primitive *or* antiderivative *of f), then there is a constant c so that $G(t) = F(t) + c$, for all $t \in (a, b)$. In particular, $\int_a^b f(t)\,dt = G(b) - G(a)$ for any primitive G of f.*

Now, in fact, most phenomena in the real world involve multiple variables, so that the study of scalar functions of a single variable in almost all cases represents an oversimplification. The differential equations that arise are often *partial differential equations*. Therefore, we attempt to cope with this by introducing *vector* or *multivariable calculus*. And indeed, the subject of multivariable calculus, sometimes called the *calculus of vector fields*, has as one of its primary goals the generalization from one to two, three, or more dimensions, the components of one-variable calculus and most especially the FTC. There are, however, two aspects in which the generalization can take place—one in the functions f, F and one in the regions on which they are defined. To best understand the possibilities, it is helpful to think of

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_9

the FTC as a relationship between the values of f on $[a, b]$ with those of a primitive F on the *boundary* of $[a, b]$. It is in that form that the four generalizations of the FTC that we are about to present—the *Fundamental Theorem of Line Integrals*, *Green's Theorem*, *The Divergence Theorem* and *Stokes' Theorem*—should be understood.

We shall make use of some of these tools in the next chapter in which we present some beautiful applications of multivariable calculus, or the calculus of vector fields, to several central topics in modern physics.

9.1 The Fundamental Theorem of Line Integrals

We'll begin by generalizing the interval in the FTC. In its place, we shall look at curves (in 2- or 3-space) or surfaces (in 3-space). We could be pedantic and give rigorous definitions of these objects as spaces, every one of whose points has a neighborhood that is "diffeomorphic" to an open interval or open disk, respectively. Instead, we shall assume that our curves and surfaces are given by parametric equations. Thus a *smooth* curve $\mathscr{C}$ is the image of a continuously differentiable function

$$\gamma : [a, b] \to \mathbb{R}^p,$$

where $p = 2$ or 3. (Actually, for the reasons we discussed in Chapter 3, one should also assume that the derivative of γ does not vanish in the interval. Otherwise, the curve may not have a well-defined tangent direction at points where the velocity vanishes.) Similarly, a *smooth* surface $\boldsymbol{\Sigma}$ is the image of a continuously differentiable function

$$\boldsymbol{\sigma} : [a, b] \times [c, d] \to \mathbb{R}^3.$$

A *tangent vector* to the curve at any point $\gamma(t_0)$, $t_0 \in (a, b)$, is given by $\gamma'(t_0)$. A *normal* to the surface at a point $\boldsymbol{\sigma}(u_0, v_0)$ is given by

$$\mathbf{N}(u_0, v_0) = \frac{\partial \boldsymbol{\sigma}}{\partial u}(u_0, v_0) \times \frac{\partial \boldsymbol{\sigma}}{\partial v}(u_0, v_0).$$

Part of the definition of *smoothness* is that neither the tangent vector nor the normal vector vanishes. We'll denote by $\mathbf{n}$ the unit normal obtained by dividing $\mathbf{N}$ by its length. We'll denote the *unit tangent vector* on a curve by τ and, as we said, the unit normal on a surface by $\mathbf{n}$. Curves, as we have defined them, always have an *orientation*—they have clearly defined start and end points and the direction between them is evident. Parametrized surfaces are also oriented—the unit normal $\mathbf{n}$, which is a continuous function of the parameters, determines the direction of flow of matter through the surface (think of it as a porous membrane) in a consistent fashion.

Now for our first generalization, replace the interval in the FTC by a smooth curve in either 2-space or 3-space. So suppose we have a curve $\mathscr{C}$ given by a continuously differentiable parametrization

$$\gamma : [a, b] \to \mathbb{R}^p,$$

where $p = 2$ or 3. Next, suppose f is a continuous function defined on an open set $\mathscr{O}$ in $\mathbb{R}^p$ that contains the curve $\mathscr{C} = \gamma([a, b])$. Functions such as f are often called *scalar fields* since they can be used to represent physical fields such as the gravitational or electrostatic potential. Then the generalization of the FTC we seek should relate the values of a 'primitive' of f on the boundary of the curve—i.e., the 2-point set $\{\gamma(a), \gamma(b)\}$—to the integral of f over $\mathscr{C}$.

In fact, we can integrate both scalar and vector fields over $\mathscr{C}$. Such integrals are called—somewhat imprecisely—*line integrals*. Thus, if $g(t)$ is a continuous scalar-valued function defined on $\mathscr{O}$ and $\mathbf{F}\colon \mathscr{O} \to \mathbb{R}^p$ is a continuous vector field on $\mathscr{O}$, then the definitions of the corresponding *line integrals* are:

$$\int_{\mathscr{C}} g\, ds = \int_a^b g(\gamma(t))\|\gamma'(t)\|\, dt$$
$$\int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r} = \int_a^b \mathbf{F}(\gamma(t)) \cdot \gamma'(t)\, dt.$$

Note. It is a rather routine exercise in "change of variables" to see that these basic line integrals are independent of the choice of parametrization of the curve $\mathscr{C}$.

Before continuing with the theory, we pause to indicate how to compute line integrals in MATLAB.

Example 9.2. Compute the line integral $\int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}$ when

$$\mathbf{F} = (\cos y, \cos z - x \sin y, -y \sin z)$$

and $\mathscr{C}$ is one of two paths from (0, 1, 0) to (0, 0, 1), the first being the straight-line path and the second being an arc of a circle of radius 1 centered at the origin in the y-z plane.

Solution 9.3. The first step is to parametrize the curve. In the first case we take $\gamma(t) = (0, 1 - t, t)$, $0 \le t \le 1$, and in the second case we take $\gamma(t) = (0, \cos(t), \sin(t))$, $0 \le t \le \frac{\pi}{2}$. Now we proceed in MATLAB as follows:

```
>> syms x y z t real
>> F = [cos(y), cos(z) - x*sin(y), -y*sin(z)];
>> gam1 = [0, 1-t, t]; gam2 = [0, cos(t), sin(t)];
>> int1 = int(subs(F,[x,y,z],gam1)*transpose(diff(gam1,t)),t,0,1)

int1 =
-1

>>  int2=int(subs(F,[x,y,z],gam2)*transpose(diff(gam2,t)),t,0,pi/2)

int2 =
-1
```

Note that in Solution 9.3, the two integrals turned out to be the same. One might well wonder if this is an accident or is due to something fundamental. The *Fundamental Theorem of Line Integrals* [FTLI] says the following:

Theorem 9.4. *Let f be a scalar function that is continuously differentiable on an open set $\mathscr{O}$ in $\mathbb{R}^p$. Let $\mathbf{A}, \mathbf{X}$ be two points of $\mathscr{O}$, connected by a smooth curve $\mathscr{C}$ lying in $\mathscr{O}$, that is, we have a smooth parametrization $\gamma : [a, b] \to \mathscr{O}$, $\gamma(a) = \mathbf{A}$, $\gamma(b) = \mathbf{X}$. Then*

$$\int_{\mathscr{C}} \nabla f \cdot d\mathbf{r} = f(\mathbf{X}) - f(\mathbf{A}),$$

where ∇f denotes the gradient of f, that is, $\nabla f = (f_x, f_y, f_z)$. Conversely, if $\mathbf{F}$ is a continuous vector field on $\mathscr{O}$, $\mathbf{A} \in \mathscr{O}$, and if we assume that for every $\mathbf{X} \in \mathscr{O}$, $\int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}$ is independent of the curve connecting $\mathbf{A}$ to $\mathbf{X}$, then if we set

$$\varphi(\mathbf{X}) = \int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r},$$

then φ is differentiable on $\mathscr{O}$ and

$$\nabla \varphi = \mathbf{F}$$

on $\mathscr{O}$.

Proof. We give the proof of the first statement. The rate of change of f along $\mathscr{C}$, at a point $\gamma(t)$, is exactly the directional derivative $D_{\gamma'(t)} f(\gamma(t))$, which by the directional derivative formula (5.3) is the dot product $\nabla f \cdot d\mathbf{r}$. Thus the first statement of the theorem is exactly the FTC for $t \mapsto f(\gamma(t))$.

The other direction is also straightforward. Suppose the path independence criterion is satisfied. To show that φ is differentiable with the indicated gradient, we compute its directional derivative at $\mathbf{X}$ in a direction $\mathbf{u}$ and show that it is well-defined and equal to the dot product of $\mathbf{u}$ and $\mathbf{F}$. By definition

$$D_{\mathbf{u}} \varphi(\mathbf{X}) = \lim_{t \to 0^+} \frac{\varphi(\mathbf{X} + t\mathbf{u}) - \varphi(\mathbf{X})}{t}.$$

To compute this, since we have independence of the path, we are free to choose $\mathscr{C}$ so that $\mathbf{X} = \gamma(b)$ and $\mathbf{u} = \gamma'(b)$. Then let $\mathscr{C}'$ be the straight-line path starting at $\mathbf{X}$ with constant velocity $\mathbf{u}$ and let $\mathscr{C} + \mathscr{C}'$ be the concatenation of the path from $\mathbf{A}$ to $\mathbf{X}$ and the straight-line path $\mathscr{C}'$. This is smooth because the velocity vectors match at the junction point. Then

$$\varphi(\mathbf{X} + t\mathbf{u}) = \int_{\mathscr{C}+\mathscr{C}'} \mathbf{F} \cdot \mathbf{r} = \varphi(\mathbf{X}) + \int_0^t \mathbf{F} \cdot \mathbf{u}\, dt,$$

and differentiating gives that $D_{\mathbf{u}} \varphi(\mathbf{X}) = \mathbf{u} \cdot \mathbf{F}(\mathbf{X})$. □

Next, we want to say a little more about the issue of when the condition in the FTLI is true—i.e., when the line integral is independent of path. This involves both a local and a global condition. The global condition is on the nature of the region $\mathscr{O}$ (simple connectivity), and will be relegated to Theorem 9.6 and Problem Set I. But the local condition on $\mathbf{F}$ (namely, that it is conservative), is easy enough to explain. Note that if this condition is satisfied, then $\mathbf{F}$ is the gradient of a function φ. That means the components F_j, $1 \leq j \leq 3$, of $\mathbf{F}$ are the partial derivatives of φ. If we differentiate again, we get second (partial) derivatives of $\mathbf{F}$. But the second partials of a smooth function have to satisfy a symmetry condition, namely

$$\frac{\partial^2 \varphi}{\partial x_j \partial x_k} = \frac{\partial^2 \varphi}{\partial x_k \partial x_j}.$$

So this imposes a condition on $\mathbf{F}$. To state it, we need to consider the *curl* of $\mathbf{F}$:

$$\mathbf{curl}\,\mathbf{F} = \left(\frac{\partial F_3}{\partial y} - \frac{\partial F_2}{\partial z}, \frac{\partial F_1}{\partial z} - \frac{\partial F_3}{\partial x}, \frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right).$$

We will employ the usual mnemonic:

$$\mathbf{curl}\,\mathbf{F} = \nabla \times \mathbf{F}.$$

Then we have the following result:

Proposition 9.5. *If φ is a smooth function of x, y, and z, then* $\mathbf{curl}\big(\nabla \varphi\big) = \mathbf{0}$.

Proof. Just take the derivatives and use equality of the mixed second partials. □

There is also a (much harder) result going the other way, which we will state now without giving a complete proof.

Theorem 9.6. *If $\mathbf{F}$ is a smooth vector field in a region $\mathscr{O}$ "without holes" (e.g., an open ball or open solid box) and if* $\mathbf{curl}\,\mathbf{F} = \mathbf{0}$, *then $\mathbf{F}$ is the gradient of a function φ.*

Proof. (Sketch). By Theorem 9.4, we neeed to show that if we have two paths γ_1 and $\gamma_2 \colon [0, 1] \to \mathscr{O}$ with the same beginning and ending points, then $\int_{\gamma_1} \mathbf{F} \cdot d\mathbf{r} = \int_{\gamma_2} \mathbf{F} \cdot d\mathbf{r}$. This is equivalent to showing that $\oint_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r} = 0$, where $\mathscr{C}$ is the closed path following first γ_1 and then traversing γ_2 in the opposite direction. Now we apply the "no holes" condition; this implies that $\mathscr{C}$ bounds a surface patch $\mathscr{R}$ inside $\mathscr{O}$. The integral of $\mathbf{F}$ around $\mathscr{C}$ can be computed in terms of a surface integral of $\mathbf{curl}\,\mathbf{F}$ over $\mathscr{R}$, using Stokes' Theorem (Theorem 9.9 below), and thus it vanishes. □

See below for more information on the curl. We shall also give physical interpretations and explanations for line integrals and surface integrals of vector fields below. But in the meantime, let's go back to Example 9.2 and see if $\mathbf{curl}\,\mathbf{F}$ vanishes.

```
>>  curl(F, [x,y,z])
```

```
ans =
 0
 0
 0
```

This suggests that $\mathbf{F}$ is indeed the gradient of some function φ. Since the first component of $\mathbf{F}$ is $\cos(y)$, it looks as if we should try taking $\varphi = x\cos y + f(y,z)$, so that $\frac{\partial\varphi}{\partial x} = \cos y$. Taking the z-derivative, since we want $\frac{\partial\varphi}{\partial z} = -y\sin z$, we get $\frac{\partial f}{\partial z} = -y\sin z$. This suggests taking $f(y,z) = y\cos z$. And indeed,

$$\frac{\partial}{\partial y}(x\cos y + y\cos z) = -x\sin y + \cos z,$$

which is the second component of $\mathbf{F}$. So with this choice, $\mathbf{F} = \nabla\varphi$. We can confirm this with MATLAB:

```
>> F - jacobian(x*cos(y) + y*cos(z), [x,y,z])
```

```
ans =
[ 0, 0, 0]
```

Alternatively, MATLAB (or more precisely, the Symbolic Math Toolbox) has a command for automating this process of recovering φ from $\mathbf{F}$:

```
>> phi = potential(F)
```

```
phi =
x*cos(y) + y*cos(z)
```

The value that we found for the line integrals is (according to Theorem 9.4) computed by

```
>> subs(phi, [x,y,z], [0,0,1]) - subs(phi, [x,y,z], [0,1,0])
```

```
ans =
-1
```

9.2 Green's Theorem

Next we go back to our general presentation of vector calculus and allow the curve $\mathscr{C}$ to close up on itself, and in so doing we shall change our reference—namely, the curve becomes the boundary and its interior becomes the domain of integration. We restrict at first to 2-space. So let $\mathscr{C}$ be a simple smooth closed curve, given by a parametrization $\gamma : [a,b] \to \mathbb{R}^2$, $\gamma(a) = \gamma(b)$, γ of course smooth and with no self-intersections other than at the end points (that is, $\gamma(c) = \gamma(d)$ implies that either $c = d$, or $c = a$ and $d = b$, or vice versa). We assume that the region enclosed by $\mathscr{C}$ is a bounded open set consisting of all of the interior of $\gamma([a,b])$. We also assume that

$\mathscr{C}$ is oriented "counterclockwise," or more precisely, that as one traverses the curve $\mathscr{C}$, the region $\mathscr{O}$ is always to one's left. Our next generalization of the Fundamental Theorem of Calculus is usually called *Green's Theorem*.

Let $\mathbf{F}$ be a smooth vector field defined on an open set $\mathscr{O}$ containing $\mathscr{C}$ and its interior $\mathscr{R}$. Then $\int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}$, often denoted $\oint_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}$ (as a reminder that $\mathscr{C}$ closes up), is obviously going to play the role of the evaluation of the primitive on the boundary. The other side of the equation will involve integration over $\mathscr{R}$ of something related to the curl of $\mathbf{F}$. Green's Theorem asserts that:

Theorem 9.7. (Green's Theorem).

$$\iint_{\bar{\mathscr{R}}} (\nabla \times \mathbf{F}) \cdot \mathbf{k}\, dA = \oint_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}. \tag{9.1}$$

This is not the usual formulation of of Green's Theorem. We have stated it this way so as to make the analogy with the FTC as precise as possible. Here is the usual formulation. Let M, N be the component functions of $\mathbf{F}$, i.e., $\mathbf{F} = (M, N)$. Then the third (or $\mathbf{k}$) component of $\nabla \times \mathbf{F}$ is easily seen to be $\frac{\partial N}{\partial x} - \frac{\partial M}{\partial y}$. And so (9.1) becomes

$$\iint_{\bar{\mathscr{R}}} \left(\frac{\partial N}{\partial x} - \frac{\partial M}{\partial y} \right) dx\, dy = \oint_{\mathscr{C}} M\, dx + N\, dy, \tag{9.2}$$

the usual formulation of Green's Theorem.

Proof. (Sketch). We give a quick sketch of the proof just for the case where $\mathscr{C}$ is the boundary of a rectangle $\bar{\mathscr{R}}$, bounded, say, by the vertical lines $x = x_0$ and $x = x_1$ and the horizontal lines $y = y_0$ and $y = y_1$, in order to indicate the connection with the FTC. This situation is illustrated in Figure 9.1. It suffices to prove equation (9.2) for this case. Applying the Fundamental Theorem of Calculus to each of the two terms in the integral on the left, we obtain

$$\begin{aligned} &\int_{y_0}^{y_1} \int_{x_0}^{x_1} \left(\frac{\partial N}{\partial x} - \frac{\partial M}{\partial y} \right) dx\, dy \\ &\qquad = \int_{y_0}^{y_1} (N(x_1, y) - N(x_0, y))\, dy - \int_{x_0}^{x_1} (M(x, y_1) - M(x, y_0))\, dx. \end{aligned}$$

The integral over x on the right is the sum of the line integrals over the two horizontal line segments (solid arrows), and the integral over y on the right is the sum of the line integrals over the two vertical line segments (dashed lines). Thus (9.2) holds. □

Corollary 9.8. (Two-Dimensional Divergence Theorem).

$$\iint_{\bar{\mathscr{R}}} (\nabla \cdot \mathbf{F})\, dA = \oint_{\mathscr{C}} \mathbf{F} \cdot \mathbf{n}\, ds, \tag{9.3}$$

where $\mathbf{n}$ *is the unit outward-pointing normal vector to the curve* $\mathscr{C}$ *and* $\nabla \cdot \mathbf{F}$ *is a shorthand for* $\frac{\partial M}{\partial x} + \frac{\partial N}{\partial y}$.

Proof. This reduces down to (9.2) applied not to **F** but to the "rotated" vector field $\mathbf{G} = (-N, M)$, which is perpendicular to **F** (since their dot product is zero). Indeed

$$(\nabla \times \mathbf{G}) \cdot \mathbf{k} = \frac{\partial M}{\partial x} + \frac{\partial N}{\partial y} = \nabla \cdot \mathbf{F}$$

and $\mathbf{G} \cdot d\mathbf{r} = \mathbf{F} \cdot \mathbf{n}\, ds$. □

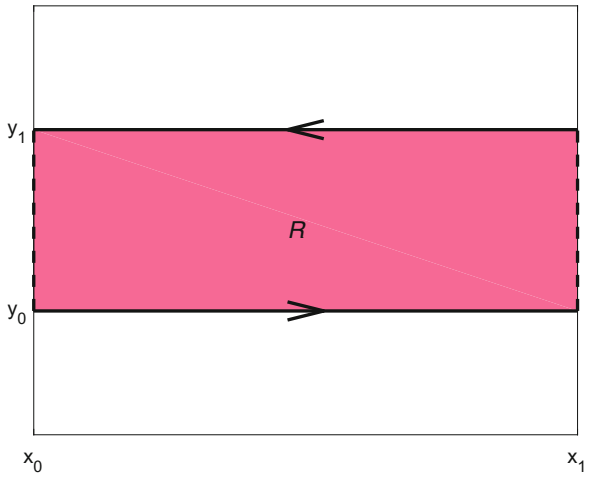


Fig. 9.1 A case of Green's Theorem

Green's Theorem is very useful. It is frequently used to evaluate a complicated line integral over a closed curve by converting it into a simpler double integral. Sometimes, it is used in reverse to evaluate a tricky double integral by converting it into a manageable line integral. See the problems for some examples.

9.3 Stokes' Theorem

Next we free the bounding curve $\mathscr{C}$ from the confines of a plane. Assume that the smooth closed curve $\mathscr{C}$ in 3-space, parametrized by $\gamma : [a, b] \to \mathbb{R}^3$, bounds an open orientable surface $\boldsymbol{\Sigma}$ in $\mathbb{R}^3$. Since we have assumed a parametrization for the curve $\mathscr{C}$, it is of course orientable. We suppose that the orientations of the curve $\mathscr{C}$ and surface $\boldsymbol{\Sigma}$ are compatible. By that we mean the tangent vector, its derivative and the normal vector satisfy a *right-hand rule*. As we saw in Chapter 3, the unit tangent τ and its derivative (with respect to arc length) $\frac{d\tau}{ds}$ are perpendicular and determine the osculating plane of the of the curve. The unit normal **n** is perpendicular to that plane and the compatibility of the orientations amounts to the fact that the triple $(\tau, \frac{d\tau}{ds}, \mathbf{n})$

satisfies the usual right-hand rule as one traverses the closed curve. Then Green's Theorem generalizes to Stokes' Theorem:

Theorem 9.9. (Stokes' Theorem). *Let* $\mathbf{F}$ *be a smooth vector field on an open set* $\mathcal{O}$ *in* $\mathbb{R}^3$ *containing* $\mathscr{C} \cup \boldsymbol{\Sigma}$. *Then*

$$\iint_{\boldsymbol{\Sigma}} (\nabla \times \mathbf{F}) \cdot \mathbf{n}\, dS = \oint_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}.$$

It behooves us to say something more concrete about what we mean by surface integrals to be sure that the meaning of the terms in Stokes' Theorem is clear. First of all, just as the line integral $\int_{\mathscr{C}} 1\, ds = \int_{\mathscr{C}} \|\gamma'(t)\|\, dt$ yields the arc length of the curve, so does the integral

$$\iint_{\boldsymbol{\Sigma}} 1\, dS = \iint_{\boldsymbol{\Sigma}} \left\| \frac{\partial \boldsymbol{\sigma}}{\partial u} \times \frac{\partial \boldsymbol{\sigma}}{\partial v} \right\| du\, dv$$

yield the surface area. Thus, one performs surface integrals of a scalar g and vector $\mathbf{F}$ fields as follows:

$$\iint_{\boldsymbol{\Sigma}} g\, dS = \int_a^b \int_c^d g(\boldsymbol{\sigma}(u,v))\, \|\mathbf{N}(u,v)\|\, du, dv$$
$$\iint_{\boldsymbol{\Sigma}} \mathbf{F} \cdot \mathbf{n}\, dS = \int_a^b \int_c^d \mathbf{F}(\boldsymbol{\sigma}(u,v)) \cdot \mathbf{N}(u,v)\, du\, dv,$$

where $\mathbf{N}(u,v) = \frac{\partial \boldsymbol{\sigma}}{\partial u} \times \frac{\partial \boldsymbol{\sigma}}{\partial v}$ is the canonical normal vector (not necessarily of unit length) determined by the parametrization $\boldsymbol{\sigma}(u,v)$. The sort of configuration envisioned in Theorem 9.9 is illustrated in Figure 9.2.

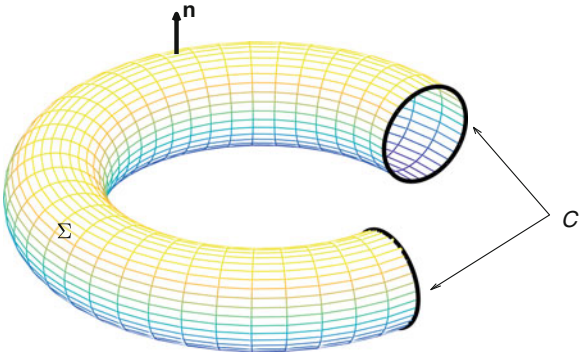


Fig. 9.2 A case of Stokes' Theorem

9.4 The Divergence Theorem

Finally, here is the fourth generalization. Instead of a closed curve bounding an open surface, we take a closed surface $\boldsymbol{\Sigma}$ bounding an open solid region $\mathscr{V}$ in $\mathbb{R}^3$. In analogy with the Green/Stokes situation—in which the boundary integration is over the closed curve, this time the boundary integration will be over the closed surface, that is

$$\iint_{\boldsymbol{\Sigma}} \mathbf{F} \cdot \mathbf{n}\, dS.$$

It must be related to the integral of the "derivative" of the vector field $\mathbf{F}$ over the interior. But this time, the derivative is measured by the *divergence* of $\mathbf{F}$ rather than its curl. The divergence of $\mathbf{F}$ is the *trace* (sum of the diagonal entries) of the matrix of partial derivatives $\left(\frac{\partial F_i}{\partial x_j}\right)$. In other words,

$$\operatorname{div} \mathbf{F} = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} + \frac{\partial F_3}{\partial z},$$

and is denoted by the shorthand $\nabla \cdot \mathbf{F}$. This is precisely the three-dimensional version of the quantity that appeared in Corollary 9.8. We arrive at the *Divergence Theorem*:

Theorem 9.10. *Let* $\mathbf{F}$ *be a smooth vector field on an open set* $\mathscr{O}$ *in* $\mathbb{R}^3$ *containing* $\mathscr{V} \cup \boldsymbol{\Sigma}$. *Then*

$$\iint_{\boldsymbol{\Sigma}} \mathbf{F} \cdot \mathbf{n}\, dS = \iiint_{\mathscr{V}} \nabla \cdot \mathbf{F}\, dV. \tag{9.4}$$

Proof. It's a bit technical to cover the general case, but we can easily do the case of a rectangular box, where $\mathscr{V}$ is given by $x_0 < x < x_1$, $y_0 < y < y_1$, and $z_0 < z < z_1$. In this case, the right-hand side of (9.4) is a sum of three integrals, while the left-hand side of (9.4) is a sum of integrals over the six sides of the box. The first term on the right is $\iiint \frac{\partial F_1}{\partial x}\, dV$, which we can write as

$$\int_{z_0}^{z_1} \int_{y_0}^{y_1} \left(\int_{x_0}^{x_1} \frac{\partial F_1}{\partial x}\, dx \right) dy\, dz.$$

Apply the FTC to the quantity inside the parentheses, and it becomes $F_1(x_1, y, z) - F(x_0, y, z)$. Thus the first term on the right of (9.4) becomes

$$\int_{z_0}^{z_1} \int_{y_0}^{y_1} F_1(x_1, y, z)\, dy\, dz - \int_{z_0}^{z_1} \int_{y_0}^{y_1} F_1(x_0, y, z)\, dy\, dz.$$

This is exactly what appears on the left-hand side of (9.4) when we sum the contributions of the faces $x = x_1$ and $x = x_0$ with the outward-pointing orientations. The other terms are handled similarly. □

Note the obvious parallel between Theorem 9.10 and Corollary 9.8. These are really the same theorem, but one in three dimensions and the other in two. The one-dimensional version of this theorem is just the FTC. Channeling the previous remarks about the use of Green's or Stokes' Theorems, this result is often utilized to convert an intractable surface integral into a manageable volume (i.e., triple) integral—or vice versa.

Now just as Theorem 9.4 and Theorem 9.6 give a necessary and sufficient condition for a vector field to be the gradient of a scalar function, there is also a similar necessary and sufficient condition for a vector field to be the curl of another vector field, called a *vector potential*. Here is the result.

Theorem 9.11. *Let* $\mathbf{F}$ *be a smooth vector field on an open set* $\mathcal{O}$ *in* $\mathbb{R}^3$. *If* $\mathbf{F} = \mathbf{curl}(\mathbf{A})$ *for some smooth vector field* $\mathbf{A}$ *on* $\mathcal{O}$, *then* $\operatorname{div}\mathbf{F} = 0$. *Conversely, if* $\operatorname{div}\mathbf{F} = 0$, *then* $\mathbf{F}$ *is* locally *the curl of another vector field* $\mathbf{A}$, *and this will be true globally if* $\mathcal{O}$ *has "no holes."*

Proof. The first statement is a simple calculation again based on the equality of the mixed second partial derivatives. In fact, we can check it in MATLAB as follows:

```
>> syms x y z A1(x,y,z) A2(x,y,z) A3(x,y,z)
>> F = curl([A1(x,y,z), A2(x,y,z), A3(x,y,z)], [x,y,z]);
>> divergence(F, [x,y,z])

ans =
0
```

In the other direction, if $\mathbf{F} = (F_1, F_2, F_3)$ is divergence-free, then at least locally, we can integrate F_3 and $-F_2$ (respectively) with respect to x, to get functions A_2 and A_3 satisfying

$$\frac{\partial A_2}{\partial x} = F_3, \quad \frac{\partial A_3}{\partial x} = -F_2.$$

Then

$$\mathbf{curl}(0, A_2, A_3) = \left(\frac{\partial A_3}{\partial y} - \frac{\partial A_2}{\partial z}, F_2, F_3\right).$$

So we will have constructed the desired vector field $\mathbf{A}$ provided $F_1 = \frac{\partial A_3}{\partial y} - \frac{\partial A_2}{\partial z}$. But since $\operatorname{div}\mathbf{F} = 0$,

$$\frac{\partial F_1}{\partial x} = -\frac{\partial F_2}{\partial y} - \frac{\partial F_3}{\partial z} = \frac{\partial^2 A_3}{\partial x \partial y} - \frac{\partial^2 A_2}{\partial x \partial z} = \frac{\partial}{\partial x}\left(\frac{\partial A_3}{\partial y} - \frac{\partial A_2}{\partial z}\right).$$

So F_1 agrees with $\frac{\partial A_3}{\partial y} - \frac{\partial A_2}{\partial z}$ up to a function with vanishing x-derivative, i.e., depending on y and z alone. We can then go back and modify A_2 and A_3 by functions of y and z so that the correction factor vanishes. □

The argument that appeared in this proof is implemented in MATLAB in the command **`vectorPotential`**. Here is an illustration:

```
>> F = [-sin(y)-cos(x)*cos(z)*sin(y), cos(y)*cos(z), ...
    sin(y)*sin(z)-sin(x)*sin(y)*sin(z)];
>> simplify(divergence(F, [x,y,z]))

ans =
0

>>  A = transpose(vectorPotential(F, [x,y,z]))

A =
[ cos(y)*sin(z), sin(y)*(z + cos(x)*sin(z)), 0]
```

We can check this and see that **A** is indeed a vector potential for **F**:

```
>>  simplify(F - transpose(curl(A, [x,y,z])))

ans =
[ 0, 0, 0]
```

Note, by the way, the vector potentials for a divergence-free vector field are highly non-unique. They are only determined up to addition of an arbitrary gradient field. That is the reason why one can do what physicists call *gauge fixing*—adding extra assumptions (such as setting the first component of **A** to 0 in our proof of Theorem 9.11 above) in order to pin down **A** more precisely. The vector potential **A** is what physicists call a *gauge field*.

9.5 Vector Calculus and Physics

We conclude this chapter with some physical interpretations of the quantities we have introduced here.

1. **Work**. If a vector field **F** represents the force exerted on an object as it travels along a curve $\mathscr{C}$ (e.g., the force due to gravity), then the work done on the object as it traverses the curve is just $\int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}$.
2. **Conservative vector fields**. A vector field **F** on a region $\mathscr{R}$ is called *conservative* if $\int_{\mathscr{C}} F \cdot d\mathbf{r}$ is independent of path between any two points in $\mathscr{R}$. We shall see in the next chapter that both *gravitational fields* and *electromagnetic fields* are conservative. In particular, for example, the work done in moving an object in a gravitational field is independent of path.
3. **Conservation of energy**. Now according to the FTLI, every conservative field is also a gradient field; that is, there is a continuously differentiable scalar field φ so that $\mathbf{F} = \nabla\varphi$ on $\mathscr{R}$. In fact, the FTLI says that the reverse is true as well. The function φ is called a *potential function* for the conservative field **F**. We shall also see in the next chapter that φ represents the *potential energy* of a particle moving in the gravitational field **F**, and that it plus the *kinetic energy* of the particle (that is, $\frac{1}{2}mv^2$) is constant. This is the law of *conservation of energy*.
4. **Conservative Fields and Green's Theorem**. Restrict to the plane momentarily. Suppose $\mathbf{F} = (M, N)$ is a gradient field and the potential function has continuous

second derivatives. Then clearly, since its mixed partials must be equal, we have $\partial N/\partial x = \partial M/\partial y$. What about the converse? This is a special case of Theorem 9.6. It's enough to show that $\mathbf{F}$ is path independent, or equivalently that $\oint_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r} = 0$ for any closed path. We wish to apply Green's Theorem and to do that for all paths, we need to assume that the region in question $\mathscr{R}$ *has no holes* or in usual mathematical parlance is *simply connected*. We'll see what happens in one of the problems when that condition is violated.

5. **Flux through a surface**. It is common to think of a vector field as representing a flow of some sort—say a fluid or gas moving through space. When it encounters a surface in its path, then we can interpret (assuming the surface is essentially a porous membrane) $\iint_{\Sigma} \mathbf{F} \cdot \mathbf{n}\, dS$ as the amount of flux through the surface.
6. **Interpretation of div and curl**. We know how to integrate vector fields—along both curves and surfaces. What about differentiating them? There are two ways represented by the divergence and the curl. Recall that if $\mathbf{F} = (M, N, P)$, these are defined by

$$\operatorname{div} \mathbf{F} = \nabla \cdot \mathbf{F} = \frac{\partial M}{\partial x} + \frac{\partial N}{\partial y} + \frac{\partial P}{\partial z}$$

$$\mathbf{curl}\, \mathbf{F} = \nabla \times \mathbf{F} = \left(\frac{\partial P}{\partial y} - \frac{\partial N}{\partial z}, \frac{\partial M}{\partial z} - \frac{\partial P}{\partial x}, \frac{\partial N}{\partial x} - \frac{\partial M}{\partial y} \right)$$

But what about an intrinsic definition? In fact, the divergence tells us about the strength of the flow out of a point; while the latter tells us about the strength of rotation of the vector field about a point. More specifically, the divergence of the vector field $\mathbf{F}$ is normally defined at follows: For any point P in the region on which the field is defined, we set

$$\operatorname{div}(P) = \lim_{\text{vol} \to 0} \frac{\int_{\Sigma} \mathbf{F} \cdot \mathbf{n}\, dS}{\text{vol}(\Sigma)}$$

It is not difficult to show that the divergence thus defined, when computed in Cartesian coordinates, agrees with the formula above. Incidentally, the notation $\nabla \cdot \mathbf{F}$ is suggested by formally considering

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right).$$

Now for the curl. The *curl* of a vector field $\mathbf{F}$, denoted $\mathbf{curl}\, \mathbf{F}$, or $\nabla \times \mathbf{F}$, and sometimes $\mathbf{rot}\, \mathbf{F}$, at a point is defined in terms of its projection onto various lines through the point. If $\mathbf{n}$ is any unit vector, the projection of the curl of $\mathbf{F}$ onto $\mathbf{n}$ is defined to be the limiting value of a closed line integral in a plane orthogonal to $\mathbf{n}$ as the path used in the integral becomes infinitesimally close to the point, divided by the area enclosed. Thus, implicitly, the curl is specified by

$$(\nabla \times \mathbf{F}) \cdot \mathbf{n} = \lim_{A \to 0} \left(\frac{1}{|A|} \oint_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r} \right),$$

where $\oint_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}$ is a line integral along the boundary of the area in question, and $|A|$ is the magnitude of the area. Again, it is a fairly routine exercise to derive from this definition the expression of $\nabla \times \mathbf{F}$ in Cartesian coordinates.

7. **Conservative fields and the curl**. It is clear that if $\mathbf{F}$ is conservative, then because $\mathbf{F} = \nabla\varphi$ for some potential φ, $\nabla \times \mathbf{F} = \nabla \times (\nabla\varphi) = 0$, this time because of the earlier given coordinate formulas for the curl. The converse, that is, $\nabla \times \mathbf{F} = 0 \Rightarrow \mathbf{F}$ is conservative, can only be concluded if the region of definition is simply connected. Incidentally, another formal computation shows that

$$\nabla \cdot (\nabla \times \mathbf{F}) = 0$$

because the vector $\nabla \times \mathbf{F}$ must be "perpendicular" to both $\mathbf{F}$ and ∇. Finally,

$$\nabla^2\varphi = \nabla \cdot \nabla\varphi = \frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} + \frac{\partial^2\varphi}{\partial z^2}.$$

Functions for which $\nabla^2\varphi = 0$ are called *harmonic functions*. Some properties of these functions were explored in Problem 5.2.

Problem Set I. Multivariable Calculus

Problem 9.1. This problem is adapted from [1], Problem Set C, Problem 8, and is intended to illustrate the Fundamental Theorem of Calculus (FTC). The differential equation

$$y' = e^{-t^2}$$

cannot be solved in terms of elementary functions, but it can be solved using MATLAB.

(a) Apply **int** to the function $t \mapsto e^{-t^2}$. The antiderivative cannot be expressed in terms of elementary functions, but MATLAB writes it in terms of **erf**, a *special function* known as the *error function*.

(b) Use the MATLAB differentiation operator **diff** to see that

$$\frac{d}{dt}(\text{erf}(t)) = \frac{2}{\sqrt{\pi}} e^{-t^2}.$$

In fact,

$$\text{erf}(t) = \frac{2}{\sqrt{\pi}} \int_0^t e^{-s^2}\, ds.$$

Use **help erf** to confirm this formula.

(c) Although one does not have an elementary formula for this function, the numerical capabilities of MATLAB mean that we "know" this function as well as we "know" elementary functions like $\tan t$. To illustrate this, evaluate $\text{erf}(t)$ at $t = 0$, 1, and 10.5, and plot $\text{erf}(t)$ on $-2 \le t \le 2$.

(d) Compute $\lim_{t\to\infty} \text{erf}(t)$ and $\int_{-\infty}^{\infty} e^{-s^2}\, ds$.

(e) Next, solve the initial value problem

$$y' + 2ty = 1, \qquad y(0) = 0$$

using **dsolve**. You should get a formula involving a function called **erfi**. Type **help erfi** to see how this function is related to **erf**. Also solve the initial value problem by hand-calculation. (Multiply both sides by e^{t^2} and observe that the left-hand side now becomes the derivative of $e^{t^2} y(t)$.) Your answer will involve an integral that cannot be evaluated by elementary techniques. Since solutions of initial value problems are unique, these two formulas (the one obtained by MATLAB and the one obtained by hand-calculation) must be the same. Explain how they are related. Hint: Use **int** to evaluate $\int_0^t e^{s^2}\, ds$.

Problem 9.2. This problem is intended to provide some practice in having MATLAB compute line integrals, in preparation for more sophisticated problems below. Use MATLAB to compute $\int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r} = \int_a^b \mathbf{F}(\gamma(t)) \cdot (x'(t), y'(t), z'(t))\, dt$ where the vector field $\mathbf{F}$ and the curve $\gamma(t) = (x(t), y(t), z(t))$ are given by:

(a) $\mathbf{F} = (2xy, x^2 + y^2)$, $\gamma(t) = (\cos t, \sin t)$, $t \in [0, 2\pi]$.
(b) $\mathbf{F} = (x, y^2)$, $\gamma(t) = (t, t^2)$, $t \in [0, 1]$.
(c) $\mathbf{F} = (xz, yz, z)$, $\gamma(t) = (\cos t, \sin t, t)$, $t \in [0, \pi]$.
(d) $\mathbf{F} = (y, x)$, $\gamma(t) = \begin{cases} (t+1, 0), & -1 \le t \le 0 \\ (\cos t, \sin t), & 0 \le t \le \pi/2 \\ (0, 1 + \pi/2 - t), & \pi/2 \le t \le \pi/2 + 1. \end{cases}$

Note that in cases (a), (b), and (d), $\mathbf{F}$ is conservative. Apply **`potential`** to it in each case and recover the results of the integrations.

Problem 9.3. Illustrate independence of path for line integrals of conservative vector fields by integrating the gradient of the scalar field $f(x, y) = x^2 + y^2$ over the following three paths that connect the origin to the point $(1, 1)$:

(a) $\gamma(t) = (t, t)$, $t \in [0, 1]$.
(b) $\gamma(t) = (t, t^2)$, $t \in [0, 1]$.
(c) $\gamma(t) = (\frac{1+\sqrt{2}\cos t}{2}, \frac{1+\sqrt{2}\sin t}{2})$, $t \in [-\frac{3\pi}{4}, \frac{\pi}{4}]$.

Now repeat the exercise for the gradient of the scalar field $e^{-(x^2+y^2)}$.

Problem 9.4. One can sometimes use Green's Theorem to compute areas of two-dimensional regions. For example, consider a region $\mathscr{R}$ bounded by a closed curve $\mathscr{C}$. Now to compute the area of $\mathscr{R}$, apply Green's Theorem to the vector field $\mathbf{F} = (-y, x)$. Here in the standard terminology of Green's Theorem, $M(x, y) = -y$ and $N(x, y) = x$. But then $\frac{\partial N}{\partial x} - \frac{\partial M}{\partial y} = 2$. Therefore, twice the area of $\mathscr{R}$ is given by the line integral $\int_{\mathscr{C}} M\,dx + N\,dy$, where of course $\mathscr{C}$ is the given closed curve. So the area we seek is just

$$\frac{1}{2} \oint_{\mathscr{C}} -y\,dx + x\,dy.$$

(a) The following example is taken from [2]. Consider the region $\mathscr{R}$ bounded by the curve $\gamma(t) = (\sin 2t, \sin t)$, $t \in [0, \pi]$. Draw the curve using MATLAB. Compute the area of $\mathscr{R}$ by evaluating the appropriate line integral using MATLAB.
(b) Carry out the same process to find the area inside one arch of a cycloid: $x = t - \sin t$, $y = 1 - \cos t$, $0 \le t \le 2\pi$.
(c) Compute the area inside one leaf of the lemniscate $x^4 = x^2 - y^2$ given parametrically by $\gamma(t) = (\sin t, \sin t \cos t)$, $t \in [0, \pi]$.

Problem 9.5. Now for a problem in which we use Green's Theorem to convert a line integral into a double integral.

(a) Compute the work done by the force field $\mathbf{F} = (-y + 5x, 3y + x)$ in moving a particle once around the ellipse $x^2 + 9y^2 = 9$ in the counterclockwise direction.

(b) Do likewise, but assume the trajectory is counterclockwise around the square whose corners are at $(\pm 1, \pm 1)$.

(c) Consider the field $\mathbf{F} = \left(\frac{y}{x^2+y^2}, \frac{-x}{x^2+y^2}\right)$. Let $\mathscr{C}$ be the unit circle traversed counterclockwise. Show that $\mathbf{F}$ is conservative in that $\frac{\partial N}{\partial x} = \frac{\partial M}{\partial y}$. But show that $\oint_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r} \neq 0$. Why does this not violate Green's Theorem? Can you find a function $\varphi(x, y)$ so that $\nabla\varphi = \mathbf{F}$? What is special about φ that helps to explain the peculiar behavior?

Problem 9.6. Corroborate the two-dimensional Divergence Theorem by evaluating

$$\iint_{\mathscr{R}} (\nabla \cdot \mathbf{F})\, dA \quad \text{and} \quad \oint_{\mathscr{C}} \mathbf{F} \cdot \mathbf{n}\, ds$$

for each of the following:

(a)
$$\mathbf{F} = \frac{r}{\|\mathbf{r}\|} = \left(\frac{x}{\sqrt{x^2+y^2}}, \frac{y}{\sqrt{x^2+y^2}}\right),$$
where $\mathscr{R}$ is the unit disk and $\mathscr{C}$ is the unit circle, traversed counterclockwise.

(b)
$$\mathbf{F} = (-y, x),$$
where $\mathscr{R}$ is the square with corners at $(\pm 1, \pm 1)$ and $\mathscr{C}$ is its boundary, traversed counterclockwise.

(c) $\mathbf{F} = \nabla f$, where $f(x, y) = e^{-(x^2+y^2)}$, $\mathscr{C}$ is the circle $(x-1)^2 + y^2 = 1$, oriented counterclockwise, and $\mathscr{R}$ is its interior. In this case, you will not be able to evaluate either integral symbolically. Use **`double`** to convert your incomplete symbolic integrals into numerical values and compare the two.

Problem 9.7. Use Stokes' Theorem to evaluate the line integral $\int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r}$ where the curve $\mathscr{C}$ is the intersection of the elliptic cylinder $2x^2 + y^2 = 1$ with the plane $z = x$, traversed counterclockwise when viewed from above; and the vector field $\mathbf{F}$ is given by $\mathbf{F} = (y^3, z^2, x)$.

Problem 9.8. Corroborate Stokes' Theorem by evaluating

$$\int_{\mathscr{C}} \mathbf{F} \cdot d\mathbf{r} \quad \text{and} \quad \iint_S (\nabla \times \mathbf{F}) \cdot \mathbf{n}\, dS$$

for each of the following:

(a) $F = (x^2 y, xy^2, z)$ and the surface S is the portion of the hyperboloid $z = xy$ cut off by the circular cylinder $x^2 + y^2 = 1$, and the curve $\mathscr{C}$ is the boundary of that surface, traversed counterclockwise when viewed from above.

(b) $\mathbf{F} = (y, z, x)$ and the surface S is the portion of the upper hemisphere $x^2+y^2+z^2 = 4$, $z \geq 0$ inside the cylinder $x^2 + y^2 = 1$, and the curve $\mathscr{C}$ is the circular intersection of the cylinder and the hemisphere, traversed counterclockwise when viewed from above.

Problem 9.9. Corroborate the three-dimensional Divergence Theorem by evaluating

$$\iiint_{\mathscr{V}} (\nabla \cdot \mathbf{F})\, dV \text{ and } \iint_S \mathbf{F} \cdot \mathbf{n}\, dS$$

for each of the following:

(a) $\mathbf{F} = (x^2y^2, y^2z^2, z^2x^2)$ and $\mathscr{V}$ is the volume bounded on the top by the sphere $x^2 + y^2 + z^2 = 2$ and on the bottom by the cylindrical cone $z = \sqrt{x^2 + y^2}$.
(b) Take the same $\mathbf{F}$, but replace the cylindrical cone by the paraboloid $z = x^2 + y^2$. (The upper boundary of $\mathscr{V}$ remains the sphere $x^2 + y^2 + z^2 = 2$.)

Problem 9.10. Use the Divergence Theorem to convert the surface integral $\iint_S \mathbf{F} \cdot \mathbf{n}\, dS$ into a volume integral $\iiint_{\mathscr{V}} \nabla \cdot \mathbf{F}\, dV$, then evaluate the volume integral. In this exercise, you should take $\mathbf{F}$ = the gravitational field exerted by a mass at the point $(0, 0, 2)$. You may assume the mass has been chosen so that the constant Gm in the formula (10.1) for $\mathbf{F}$ is equal to 1, in which case

$$\mathbf{F} = -\frac{\mathbf{r} - \mathbf{r}_0}{\|\mathbf{r} - \mathbf{r}_0\|^3}, \mathbf{r}_0 = (0, 0, 2).$$

Furthermore, S is the cylinder bounded on its sides by the curve $x^2 + y^2 = 1$ and on the top and bottom by the planes $z = \pm 1$, and finally, the volume $\mathscr{V}$ is the inside of the cylinder. It is likely that you will have to evaluate the integral you set up numerically. (Be patient: it might take MATLAB a while to crank out an answer.)

Problem 9.11. Show that the vector field

$$\mathbf{F} = \big(e^y \sin(x)\sin(z) + e^z \cos(x)\cos(y),\, e^z \sin(x)\sin(y) + e^x \cos(y)\cos(z),\, e^x \sin(y)\sin(z) + e^y \cos(x)\cos(z)\big)$$

is divergence-free and find a vector potential for it.

Glossary of MATLAB Commands

`cross` The cross product of two vectors

`curl` The curl of a vector field

`diff` Differentiates an expression

`divergence` The divergence of a vector field

dot The dot product of two vectors

double Converts the (possibly symbolic) expression for a number to a numerical (double-precision) value

dsolve Symbolic differential equation solver

erf The *error function* $\mathbf{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$; see also **erfi**

fplot Easy function plotter

fplot3 Easy 3D function plotter

gradient Computes the gradient of a scalar field

int Integrates an expression

laplacian Computes the Laplacian of a scalar field

limit Computes a limit

potential Computes a potential for a conservative vector field

real Follows **syms** to insure variables are real

simplify Used to simplify a symbolic expression. Note that sometimes you will need to increase the option **Steps** for best results.

solve Symbolic equation solver

subs Substitute for a variable in an expression

syms Set up one or more symbolic variables

vectorPotential Computes a vector potential for a divergence-free vector field

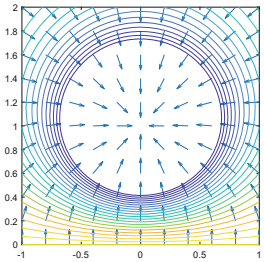
Options to MATLAB Commands

Steps Specifies the number of steps that **simplify** will use in its attempt to algebraically simplify an expression

References

1. B. Hunt, R. Lipsman, J. Osborn, J. Rosenberg, *Differential Equations with MATLAB*, 3rd edn. (Wiley, Hoboken, 2012)
2. D. Nykamp et al., *Math Insight*, http://mathinsight.org, Thread: Multivariable Calculus (2017)

Chapter 10
Physical Applications of Vector Calculus



This chapter represents the culmination of multivariable calculus. We investigate the remarkable physical applications of vector calculus that provided the original motivation for the development of this subject in the seventeenth, eighteenth, and nineteenth centuries. The vector fields that we examine arise naturally in celestial mechanics, electromagnetism, and fluid flow. We will use the basic concepts of vector calculus to derive fundamental laws of physics in these subjects. In the attached Problem Set I, you will have a chance to use MATLAB to solve some interesting physical problems that would be difficult or impossible to tackle with pencil and paper alone.

10.1 Motion in a Central Force Field

In Chapter 4, *Kinematics*, we studied the motion of an object in a central force field as an application of the theory of curves in 3-space. We now reexamine the same subject from the point of view of the calculus of vector fields. We begin with the study of the motion of a *particle*, that is, of an object whose size is negligible compared to the distances over which it moves. For all practical purposes we may assume the object is located at a single point in space. In fact, the theory we develop does not apply very well to the objects usually known as particles in physics, such as electrons, photons, and positrons. True particles are so small that classical physics does not provide a very good approximation to their actual behavior. They need to be studied with quantum mechanics, which involves a more complicated vector calculus than we will study here. The theory we develop, however, works quite well when applied to "celestial particles" such as planets and comets moving around a star.

We assume that the location of our particle is represented by a *position* vector $\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$ in $\mathbb{R}^3$, which is a function of the time t. We also assume the particle obeys *Newton's law of motion*, which says that $\mathbf{F} = m\mathbf{a} = m\ddot{\mathbf{r}}$, where $\mathbf{F}$ is the total force on the particle, m is its mass, $\mathbf{a}$ is its acceleration, and we use dots to indicate time derivatives.

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_10

Suppose that the force on the particle is always a scalar multiple of the position vector, with the scalar factor a function of $r = \|\mathbf{r}\|$, but not of time. This amounts to assuming three things:

- the magnitude of the force on the particle only depends on the distance $r = \|\mathbf{r}\|$ from the origin;
- the force on the particle always points either toward or away from the origin; and
- whether the force points inward or outward (i.e., is *attractive* or *repulsive*) also depends only on $r = \|\mathbf{r}\|$.

When these conditions are satisfied, we say the object is moving in a *central force field*. In symbols, $\mathbf{F} = f(r)\mathbf{r}$. The classical example occurs when $\mathbf{r}(t)$ represents the position of a planet in a solar system with the sun at the origin. We assume the mass m of the planet is negligible compared to the mass M of the sun, so that the sun remains fixed. The planet orbits the sun under the influence of gravity, which is an attractive force always pointing inward toward the sun. *Newton's law of gravitation* says that

$$\mathbf{F} = -\frac{GMm}{\|\mathbf{r}\|^3}\,\mathbf{r}, \tag{10.1}$$

where G is a universal constant. The minus sign occurs because gravity is attractive, hence tends to pull objects in the direction of decreasing r. As we pointed out in Chapter 4, Newton's law of gravitation (10.1) is an *inverse square law*, since

$$\|\mathbf{F}\| = \frac{GMm}{\|\mathbf{r}\|^2}.$$

For the time being, let's see how far we can progress without assuming (10.1), since there are examples of central force fields in physics where the dependence of $\mathbf{F}$ on $\mathbf{r}$ is different. We assume only that $\mathbf{F} = f(r)\mathbf{r}$. Applying Newton's law of motion, we arrive at the fundamental differential equation governing the motion of the object:

$$\mathbf{F} = f(r)\mathbf{r} = m\ddot{\mathbf{r}}, \tag{10.2}$$

or

$$\dot{\mathbf{v}} = \ddot{\mathbf{r}} = \frac{f(r)}{m}\,\mathbf{r} = g(r)\mathbf{r}.$$

Let's take the curl of the vector field $\mathbf{F}(\mathbf{r}) = f(r)\mathbf{r} = f(r)(x\mathbf{i} + y\mathbf{j} + z\mathbf{k})$ that gives the force on the particle. We can do this in MATLAB as follows:

```
>> syms x y z real; syms f(r)
>> rvec = [x, y, z];
>> r = sqrt(dot(rvec, rvec))

r =
(x^2 + y^2 + z^2)^(1/2)

>> curl(f(r)*rvec)
```

```
ans =
0
0
0
```

Thus

$$\mathbf{curl}\,\mathbf{F} = \nabla \times \mathbf{F} = \mathbf{0}. \tag{10.3}$$

This suggests that $\mathbf{F}$ should be a *conservative* force field, i.e., that $\mathbf{F} = -\nabla\phi$ for some function ϕ. (Vanishing of the curl does not quite imply that $\mathbf{F}$ is the gradient of a function, because of possible singularities of $\mathbf{F}$ at the origin.) Indeed, suppose $\phi(\mathbf{r})$ depends only on $r = \|\mathbf{r}\|$, so that we can write $\phi(r)$ without fear of confusion. We compute the gradient:

```
>> gradient(phi(r))
```

```
ans =
(x*D(phi)((x^2 + y^2 + z^2)^(1/2)))/(x^2 + y^2 + z^2)^(1/2)
(y*D(phi)((x^2 + y^2 + z^2)^(1/2)))/(x^2 + y^2 + z^2)^(1/2)
(z*D(phi)((x^2 + y^2 + z^2)^(1/2)))/(x^2 + y^2 + z^2)^(1/2)
```

The output is equal to $(\phi'(r)/r)\mathbf{r}$, so $\mathbf{F} = -\nabla\phi$ provided we choose ϕ such that

$$\frac{\phi'(r)}{r} = -f(r).$$

In other words, we should choose $\phi(r)$ to be an antiderivative of $-rf(r)$. Such a choice of ϕ is called a *potential function* for the central force field. If $f(r) \to 0$ rapidly enough as $r \to \infty$ so that the following improper integral converges, then we can normalize ϕ by taking

$$\phi(r) = \int_r^\infty x f(x)\, dx. \tag{10.4}$$

In this case, $\phi(r) \to 0$ as $r \to \infty$.

Finally, define

$$E = \frac{m}{2}\|\mathbf{v}\|^2 + \phi(r), \tag{10.5}$$

which we think of as a function of t by evaluating it along the path of the particle. This expression is called the *energy* of the particle. The first term is the *kinetic energy*; the second, the *potential energy*. If we express the kinetic energy as a dot product, $\frac{1}{2}m\mathbf{v}\cdot\mathbf{v}$, its time derivative becomes

$$\frac{d}{dt}\left(\frac{m}{2}\mathbf{v}\cdot\mathbf{v}\right) = \frac{m}{2}\left(\dot{\mathbf{v}}\cdot\mathbf{v} + \mathbf{v}\cdot\dot{\mathbf{v}}\right) = m\mathbf{a}\cdot\mathbf{v} = \mathbf{F}\cdot\mathbf{v} = f(r)\mathbf{r}\cdot\dot{\mathbf{r}}.$$

Now ϕ was chosen so that $\mathbf{F} = -\nabla\phi$. By the chain rule, the time derivative of the potential energy is:

$$\frac{d\phi}{dt} = \frac{\partial\phi}{\partial x}\frac{dx}{dt} + \frac{\partial\phi}{\partial y}\frac{dy}{dt} + \frac{\partial\phi}{\partial z}\frac{dz}{dt} = \nabla\phi \cdot \dot{\mathbf{r}} = -\mathbf{F} \cdot \mathbf{v}.$$

This precisely cancels the time derivative of the kinetic energy, and so $\dot{E} = 0$. This is the *law of conservation of energy*; it says that the quantity E remains fixed once and for all.

To summarize, assuming we have an object moving in a central force field according to Newton's laws of motion, we have used vector calculus to derive the law of conservation of energy. We have also demonstrated that the force field must be conservative. Recall that under the same hypotheses, we derived in Chapter 4 the law of conservation of angular momentum, and we showed that the motion of such an object is planar.

10.2 Newtonian Gravitation

In the previous section, we discussed Newton's law of gravitation, but only in the special case of a single object moving in a central force field. A more general and interesting problem is the *n-body problem*, the study of the motion of n bodies (which we again assume are "particles," i.e., of negligible size compared to the distances between them), moving under the influence of their mutual gravitational attractive forces. If $\mathbf{r}_i$ denotes the position vector of the ith particle and m_i is its mass ($i = 1, 2, \ldots, n$), then each particle exerts a force on the others given by the inverse square law (10.1) as discussed above, so there are equations of motion analogous to (10.2) above. In particular, the force on the ith particle is given by

$$m_i\ddot{\mathbf{r}}_i = -\sum_{j\neq i} Gm_i m_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\|\mathbf{r}_i - \mathbf{r}_j\|^3}.$$

While these equations appear quite complicated, a few important features deserve comment. The first feature, which is quite special to gravitation, is the fact that we can cancel m_i from both sides of the equation. (There is an m_i on the right, since the force between the ith and jth particle is proportional to the product of the masses $m_i m_j$; while there is an m_i on the left, since Newton's law of motion says that the force on the ith particle is equal to $m_i\ddot{\mathbf{r}}_i$.) We obtain

$$\ddot{\mathbf{r}}_i = -\sum_{j\neq i} Gm_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\|\mathbf{r}_i - \mathbf{r}_j\|^3}, \tag{10.6}$$

whose right-hand side *no longer involves the mass of the* ith *particle*. It only involves the position $\mathbf{r}_i$ of the ith particle and the positions and masses of the other particles. We may think of it as a *vector field* (i.e., a vector-valued function of $\mathbf{r}$), evaluated at $\mathbf{r} = \mathbf{r}_i$.

We can use MATLAB to visualize this vector field for various configurations of masses. For example, with an object of mass 2 located at $(1,\ 0,\ 0)$ and with an object of mass 1 located at $(-1,\ 0,\ 0)$, the vector field looks like Figure 10.1:

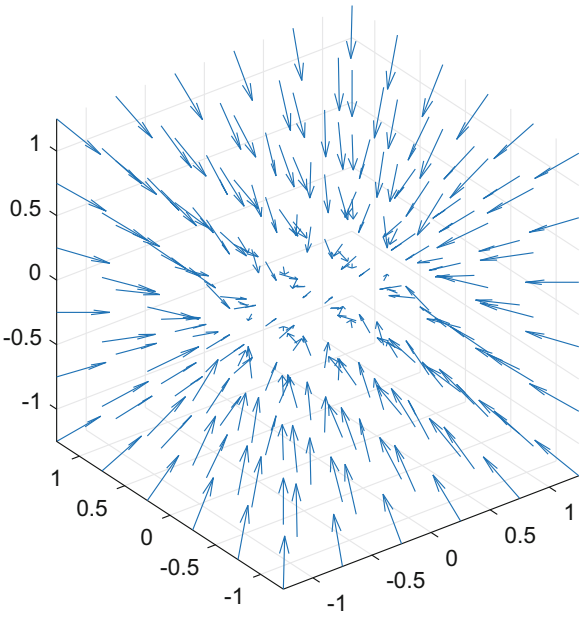


Fig. 10.1 A gravitational force field

We will now find that we can reformulate the laws of gravitation by considering the flux of the gravitational vector field. Recall that the *flux* of a vector field $\mathbf{V}$ over an oriented surface S is the surface integral $\iint_S \mathbf{V} \cdot \mathbf{n}\, dS$, where $\mathbf{n}$ is the unit "outward pointing" normal vector. Let's take S to be any simple closed piecewise smooth surface (for instance, a sphere, or the surface of a cube) enclosing all of the particles except for the ith. Then the flux through S of the vector field specified by the right-hand side of (10.6) is given by

$$\iint_S \left(-\sum_{j \neq i} G m_j \frac{\mathbf{r} - \mathbf{r}_j}{\|\mathbf{r} - \mathbf{r}_j\|^3} \right) \cdot \mathbf{n}\, dS = -\sum_{j \neq i} G m_j \iint_S \frac{\mathbf{r} - \mathbf{r}_j}{\|\mathbf{r} - \mathbf{r}_j\|^3} \cdot \mathbf{n}\, dS. \quad (10.7)$$

To evaluate (10.7), we only need to compute

$$\iint_S \frac{\mathbf{r} - \mathbf{r}_j}{\|\mathbf{r} - \mathbf{r}_j\|^3} \cdot \mathbf{n}\, dS. \quad (10.8)$$

If S were a sphere of radius r_0 centered at $\mathbf{r}_j$, then (10.8) would become

$$\begin{aligned}\iint_{\|\mathbf{r}-\mathbf{r}_j\|=r_0} \frac{\mathbf{r}-\mathbf{r}_j}{\|\mathbf{r}-\mathbf{r}_j\|^3}\cdot\mathbf{n}\,dS &= \iint_{\|\mathbf{r}\|=r_0} \frac{\mathbf{r}}{\|\mathbf{r}\|^3}\cdot\mathbf{n}\,dS\\ &= \iint_{\|\mathbf{r}\|=r_0} \frac{\mathbf{n}}{r_0^2}\cdot\mathbf{n}\,dS\\ &= \frac{1}{r_0^2}\iint_{\|\mathbf{r}\|=r_0} dS\\ &= \frac{1}{r_0^2}4\pi r_0^2\\ &= 4\pi.\end{aligned} \tag{10.9}$$

Away from the origin, we can compute the divergence of $(1/r^3)\mathbf{r}$ by

```
>> simplify(divergence((1/r^3)*rvec))

ans =
0
```

Thus, away from the point $\mathbf{r}=\mathbf{r}_j$ where the integrand in (10.8) blows up, we have

$$\operatorname{div}\frac{\mathbf{r}-\mathbf{r}_j}{\|\mathbf{r}-\mathbf{r}_j\|^3} = 0. \tag{10.10}$$

This means the flux integral (10.8) is independent of the choice of the surface S. To see that, apply the *Divergence Theorem* to the shell region between S_1 and S_2, where S_1 and S_2 are simple closed piecewise smooth surfaces around $\mathbf{r}_j$, with S_1 inside S_2. Since (10.10) holds inside the shell region, the Divergence Theorem gives

$$\iint_{S_2}\cdots - \iint_{S_1}\cdots = \iint_{S_2-S_1}\cdots = 0,$$

(where for convenience we have suppressed the integrand). Hence (10.8) is also independent of S, provided S encloses all the points $\mathbf{r}_j$. Putting everything together, we get *Gauss' law*:

$$\begin{aligned}\iint_S\left(-\sum_{j\neq i} Gm_j\frac{\mathbf{r}-\mathbf{r}_j}{\|\mathbf{r}-\mathbf{r}_j\|^3}\right)\cdot\mathbf{n}\,dS &= -\sum_{j\neq i} Gm_j 4\pi\\ &= -4\pi G(\text{mass enclosed by } S).\end{aligned} \tag{10.11}$$

In addition, the calculation in (10.3) shows that the curl of the right-hand side of (10.6) vanishes (except at the points $\mathbf{r}_j$ where it is undefined).

By extension, we see that according to the laws of Newtonian gravitation, any distribution of masses in space defines a *gravitational field* $\mathbf{G}(\mathbf{r})$, a vector field with

the property that the acceleration due to gravity of a particle of arbitrary mass m with position vector $\mathbf{r}$ is given by the vector field $\mathbf{G}(\mathbf{r})$, regardless of the value of m. This vector field has vanishing curl:

$$\mathbf{curl}\,\mathbf{G} = \mathbf{0}, \tag{10.12}$$

and satisfies Gauss' law:

$$\iint_S \mathbf{G}\cdot\mathbf{n}\,dS = -4\pi G(\text{mass enclosed by } S). \tag{10.13}$$

Thus, vector calculus gives us a very elegant way to reformulate the basic principles of Newtonian gravitation.

These laws apply (as we can see by an appropriate limiting process) also when the distribution of masses in space is continuous rather than discrete. Then we no longer have singularities in $\mathbf{G}(\mathbf{r})$ at point masses. The formulation (10.13) is particularly useful when the mass distribution is spherically symmetric about the origin, i.e., only depends on $r = \|\mathbf{r}\|$. Then the symmetry implies that $\mathbf{G}(\mathbf{r})$ is a scalar function $g(r)$ of r multiplied by the unit vector $\mathbf{n} = \mathbf{r}/r$ in the direction of $\mathbf{r}$. From (10.13), with S the sphere of radius r centered at the origin, we have

$$\begin{aligned} -4\pi G(\text{mass enclosed by } S) &= \iint_S \mathbf{G}\cdot\mathbf{n}\,dS \\ &= \iint_S g(r)\mathbf{n}\cdot\mathbf{n}\,dS \\ &= g(r)\cdot\text{surface area}(S) \\ &= 4\pi r^2 g(r), \end{aligned}$$

and thus

$$\mathbf{G}(\mathbf{r}) = -\frac{G\mathbf{n}}{r^2}(\text{mass enclosed by sphere of radius } r). \tag{10.14}$$

It is also possible to give an equivalent formulation of Gauss' law (10.13) using the Divergence Theorem. If $\rho(\mathbf{r})$ is the density of matter at $\mathbf{r}$, and $\mathscr{R}$ is the solid region bounded by S, then the mass enclosed by S is

$$\iiint_{\mathscr{R}} \rho(\mathbf{r})\,dV.$$

Now (10.13) can be rewritten using the Divergence Theorem as

$$\iiint_{\mathscr{R}} \operatorname{div}\mathbf{G}(\mathbf{r})\,dV = -4\pi G \iiint_{\mathscr{R}} \rho(\mathbf{r})\,dV.$$

This being true for arbitrary regions $\mathscr{R}$, we must have

$$\operatorname{div} \mathbf{G} = -4\pi G\rho. \tag{10.15}$$

Formulas (10.15) and (10.13) give a convenient formulation for studying the gravitational field inside bodies which are no longer considered to be particles (see some applications in the exercises). The formulation is also useful for comparison with Maxwell's laws of electromagnetism, which we will study in the next section. The usual method of solving these equations is to introduce a function ϕ, called the *gravitational potential*, satisfying $\mathbf{G} = -\nabla\phi$. (This is possible since $\mathbf{G}$ is a conservative field by (10.12).) Note that ϕ is only defined up to a constant. (The difference between the values of ϕ at two points in space measures how much work must be done against gravity to move an object from one point to the other.) Substituting the expression for $\mathbf{G}$ into (10.15), we obtain *Poisson's equation*

$$\nabla^2\phi = 4\pi G\rho. \tag{10.16}$$

This is a *second-order* differential equation (it involves second derivatives), but it has the advantage of only involving a scalar function and not a vector field.

To summarize: to understand the gravitational field generated by a continuous distribution of matter, we only have to solve a second-order (scalar) partial differential equation (Poisson's equation).

10.3 Electricity and Magnetism

Electrostatics, the study of electrical forces between static or slowly moving charges, is formulated in mathematical terms in almost the same way as Newtonian gravitation. The key physical fact is *Coulomb's law*, which says the electrical force between two charged objects is an inverse square law. The only differences from the laws of gravity are that:

- the force between charges is proportional to the product of the charges, and does not involve the masses; and
- the force between charges can be either attractive or repulsive. Like charges repel; opposite charges attract.

With only changes in some constants, all the equations we derived for gravitation apply equally well to electrostatics. We spare the reader the chore of rederiving the results; we just state them. First, any distribution of charges in space defines an *electric field* $\mathbf{E}(\mathbf{r})$, a vector field with the property that the electrostatic force on a particle of arbitrary charge q with position vector $\mathbf{r}$ is given by $q\mathbf{E}(\mathbf{r})$. Second, neglecting magnetic effects, the electrical field $\mathbf{E}(\mathbf{r})$ obeys the analogs of (10.12) and (10.13):

$$\mathbf{curl}\, \mathbf{E} = \mathbf{0} \tag{10.17}$$

and *Gauss' law of electrostatics*:

$$\iint_S \mathbf{E} \cdot \mathbf{n}\, dS = 4\pi \,(\text{charge enclosed by } S). \tag{10.18}$$

Since $\mathbf{E}$ is a conservative force field, we can again represent it in terms of a potential function ϕ, now called the *electrostatic potential*:

$$\mathbf{E} = -\nabla\phi.$$

For a point charge q located at the origin in $\mathbb{R}^3$, the electrostatic potential is simply

$$\phi(\mathbf{r}) = \frac{q}{\|\mathbf{r}\|}.$$

In the case of continuous charge distributions, ϕ obeys an analog of Poisson's equation (10.16):

$$\operatorname{div} \mathbf{E} = -\nabla^2\phi = 4\pi\rho, \tag{10.19}$$

where ρ is the charge density. Equations (10.18) and (10.19) are the simplest forms of *Maxwell's equations*, which codify the principles of electromagnetism in terms of vector calculus.

One phenomenon that appears in electrostatics but not in gravitation is the concept of a (perfect) *conductor*. In a perfect conductor, charges instantaneously rearrange themselves so that the electrical field inside the conductor vanishes. Since $\mathbf{E} = -\nabla\phi$, the electrostatic potential ϕ is constant throughout the conductor. In practice, metal objects can usually be treated as conductors for purposes of electrostatics, as can moist earth. (Hence the term "to ground" a piece of equipment, i.e., to connect it to a perfect conductor so as to set $\phi = 0$ on it.) At the opposite extreme from a perfect conductor is a *perfect insulator* or *dielectric* (from the prefix *dia-*, meaning "apart"), inside of which current is unable to flow.

Thanks to the work of Maxwell, we know that it is impossible to give a satisfactory mathematical treatment of electrical forces without considering magnetism at the same time. Indeed, electricity and magnetism are two different aspects of the same fundamental force, usually called *electromagnetism*. Maxwell's equations of electromagnetism are coupled equations for two vector fields, the electric field $\mathbf{E}(\mathbf{r})$ and the magnetic field $\mathbf{B}(\mathbf{r})$. Only in static situations (where all charges are at rest) do these fields become decoupled from one another. The magnetic field, like the electric field, exerts a force on charged objects, but this force is proportional to the speed of the charged object and is perpendicular to its motion. The magnetic force on a charged object with charge q and position vector $\mathbf{r}$ is

$$\mathbf{F}_{\text{mag}} = -\frac{q}{c}\mathbf{B} \times \dot{\mathbf{r}},$$

where c is a universal constant, the speed of light in a vacuum. In a dynamic situation, Maxwell's equations are

$$\begin{aligned} \mathbf{curl}\,\mathbf{E} &= -\frac{1}{c}\dot{\mathbf{B}}, \\ \operatorname{div}\mathbf{E} &= 4\pi\rho, \\ \mathbf{curl}\,\mathbf{B} &= \frac{4\pi}{c}\mathbf{J} + \frac{1}{c}\dot{\mathbf{E}}, \\ \operatorname{div}\mathbf{B} &= 0. \end{aligned} \tag{10.20}$$

Here $\mathbf{J}$ is the *current density* (current per cross-sectional area). The field $\mathbf{J}$ has the same units as $\dot{\mathbf{E}}$, electric field per unit time. The equation for $\mathbf{curl}\,\mathbf{E}$ is a form of *Faraday's law of induction*, which says that changing magnetic fields give rise to an electric field. The equation for $\mathbf{curl}\,\mathbf{B}$ is a form of *Ampère's law*, which says that magnetic fields exert forces on moving charges. The equation for $\operatorname{div}\mathbf{E}$ we recognize as a form of Gauss' law. The vanishing of $\operatorname{div}\mathbf{B}$ means that there are no point sources of magnetic field, that is, no *magnetic monopoles*. (While physical theories involving magnetic monopoles, first proposed by Dirac in 1931, are increasingly in vogue, there is no firm experimental evidence that magnetic monopoles exist in nature.) Since $\operatorname{div}\mathbf{B} = 0$, it follows that we can find a vector field $\mathbf{A}$ (only unique up to addition of the gradient of a function) such that $\mathbf{B} = \mathbf{curl}\,\mathbf{A}$. Such a field is called a *vector potential*, as we discussed in Chapter 9. In analogy with the picture of a gravitational field in Figure 10.1, we display in Figure 10.2 an electrostatic field in which a positive charge of 2 is located at $(1,\ 0,\ 0)$ and a negative charge -1 is located at $(-1,\ 0,\ 0)$.

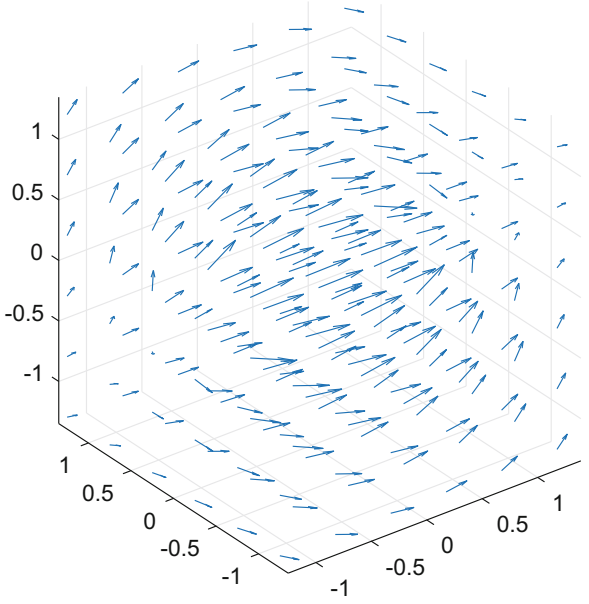


Fig. 10.2 An electrostatic field

Originally, it was thought that the vector potential $\mathbf{A}$ is just a mathematical convenience and that it has no physical meaning itself, i.e., all one can measure is $\mathbf{B} = \mathbf{curl}\,\mathbf{A}$ and not $\mathbf{A}$ itself. Now we know, thanks to quantum mechanics, that this is not completely true. The quantum mechanical equations describing a charged particle in a magnetic field actually involve $\mathbf{A}$ and not just $\mathbf{B}$, and indeed this gives rise to the *Aharonov–Bohm effect*, whereby the wave function of the particle moving along a curve $\mathscr{C}$ changes by an amount proportional to the line integral $\int_{\mathscr{C}} \mathbf{A} \cdot d\mathbf{r}$. This has been observed experimentally.

10.4 Fluid Flow

In this section, we discuss the study of fluids using vector calculus. Fluid flow is a very complicated subject with applications ranging from weather prediction to the design of chemical factories. We can only give the most rudimentary introduction here. This should be enough to convince you of the importance of multivariable calculus for this subject. We consider a fluid (a liquid or a gas, basically "anything that flows") moving in a certain region $\mathscr{R}$ (possibly a container or the interior of a pipe) located in $\mathbb{R}^3$. At a point of $\mathscr{R}$ with position vector $\mathbf{r}$, we can measure various properties of the fluid at that point, such as the temperature T, the pressure p, the density ρ, and the velocity vector $\mathbf{v}$. The temperature, pressure, and density are related by the

equation of state of the fluid. For example, for an "ideal gas," p is proportional to ρT. For liquids, on the other hand, ρ tends to vary very little with p and T. We will mostly be concerned with an idealized *incompressible fluid*, in which ρ is constant. We will also neglect thermodynamic effects and ignore the parameter T, though it is especially important in gas dynamics.

The first basic principle of fluid flow is *conservation of mass*, in other words, that fluid is neither created nor destroyed. That is not to say, however, that it can't flow from place to place. Consider a fixed closed surface S in $\mathscr{R}$. The flux integral $\iint_S \rho \mathbf{v} \cdot \mathbf{n}\, dS$ is the integral over S of the mass flowing across the surface per unit surface area per unit time. It therefore computes the mass of fluid flowing out of S per unit time. As the mass of fluid enclosed by S is

$$\iiint_{\text{interior of } S} \rho \, dV,$$

we obtain the equation

$$\iint_S \rho \mathbf{v} \cdot \mathbf{n}\, dS \;=\; \frac{\partial}{\partial t} \iiint_{\text{interior of } S} \rho \, dV.$$

Applying the divergence theorem to the left-hand side gives

$$\iiint_{\text{interior of } S} \operatorname{div}(\rho \mathbf{v})\, dV \;=\; \iiint_{\text{interior of } S} \frac{\partial \rho}{\partial t}\, dV.$$

Since S was arbitrary, this gives us the equation of conservation of mass:

$$\operatorname{div}(\rho \mathbf{v}) = \frac{\partial \rho}{\partial t}. \tag{10.21}$$

In the case of an incompressible fluid, ρ is constant. So this simplifies to

$$\operatorname{div} \mathbf{v} = 0, \tag{10.22}$$

which is analogous to equations (10.15) and (10.19) in empty and uncharged space, respectively. By the way, this illustrates an important principle of mathematical physics: *very often, totally different physical phenomena are governed by the same mathematical equations.* That is one reason why vector calculus is so powerful.

The next equation of fluid flow corresponds to Newton's law of motion, *force = mass times acceleration*, applied to a small piece of the fluid. But we have to be careful. We've set up our coordinates to remain fixed in space, not to move with the fluid. For simplicity, suppose we're dealing with *steady flow*, so that $\mathbf{v}(\mathbf{r})$ does not vary with time. For a small piece of fluid located at the point with position vector $\mathbf{r}$ at time t, its velocity at time $t + h$ is $\mathbf{v}$ computed at the point to which the piece of

fluid has moved by that time, which is (for small h) approximately $\mathbf{v}(\mathbf{r} + h\mathbf{v}(\mathbf{r}))$. By the chain rule, the acceleration of the piece of fluid is

$$\lim_{h \to 0^+} \frac{\mathbf{v}(\mathbf{r} + h\mathbf{v}(\mathbf{r})) - \mathbf{v}(\mathbf{r})}{h} = \nabla \mathbf{v} \cdot \mathbf{v}.$$

(Since $\mathbf{v}$ is a vector, what we've denoted $\nabla \mathbf{v}$ is really the 3×3 matrix made up of the gradients of the three components of $\mathbf{v}$. Each of these gradients, when dotted with $\mathbf{v}$, gives one of the components of the acceleration.) The second equation of motion for steady flow is therefore of the form

$$\rho \nabla \mathbf{v} \cdot \mathbf{v} = \text{force on the fluid per unit volume.} \tag{10.23}$$

Equation (10.23) can become quite complicated when gravitational, electromagnetic, and thermodynamic effects are taken into account. In addition, we need to include *viscous forces*, forces due to internal friction in the liquid. The simplest case of (10.23) arises when the flow is steady, and we ignore all forces except those due to pressure in the fluid. (Pressure, after all, is force per unit area.) Then (10.23) simplifies to

$$\rho \nabla \mathbf{v} \cdot \mathbf{v} = -\nabla p. \tag{10.24}$$

(The "minus sign" reflects the fact that if the pressure increases in a certain direction, then that will tend to push fluid backward, from the region of high pressure to the region of low pressure.)

Since we usually don't know the variation of the pressure in the fluid, (10.24) might not appear to be of much use. But at least ∇p is a conservative vector field, so the curl of the left-hand side must vanish. Thus, in the very simplest case of fluid flow (steady incompressible flow with no viscous or external forces), we get a pair of coupled equations for the velocity field: (10.22) and

$$\mathbf{curl}\,(\nabla \mathbf{v} \cdot \mathbf{v}) = \mathbf{0}. \tag{10.25}$$

Equation (10.25) is still a formidable equation to solve. However, you can check that it is satisfied when

$$\mathbf{curl}\,\mathbf{v} = \mathbf{0}. \tag{10.26}$$

This is the case of *irrotational flow*, and is a reasonable approximation to the truth when certain thick liquids are flowing rather slowly. Some cases of steady, incompressible, irrotational flow are studied in the following problem set. Note that the pair of equations (10.22) and (10.26) coincides with some special cases of Maxwell's equations (10.20) of electromagnetism.

10.5 Heat and Wave Equations

In this last section, we shall use the Divergence Theorem (somewhat analogous to the way we did in the last section) to derive the fundamental partial differential equation, the *heat equation* that describes the flow of heat in a body; and then we shall use Maxwell's equations to derive another fundamental partial differential equation, the *wave equation*, that describes the propagation of electromagnetic waves through a medium.

10.5.1 The Heat Equation

So suppose that the function $u(t, \mathbf{x})$ measures the temperature of a body (in 3-space) at time t and position $\mathbf{x}$. We list various properties—gleaned from observation and physical laws—that constrain the temperature function u.

- The heat energy in the material is proportional to the temperature, the density of the material and its specific heat capacity. We can summarize that assertion in an equation for the heat
$$Q = \iiint_{\mathscr{V}} \rho\kappa u \, dV, \tag{10.27}$$
where ρ is the density and κ is the heat capacity. These may vary but we shall assume them to be constant here. Moreover, $\mathscr{V}$ is any solid region inside the body.
- The heat transfer through the boundary Σ of the region $\mathscr{V}$ is proportional to the heat conductivity, the gradient of the temperature across the region and to the area of contact. Thus
$$\frac{\partial Q}{\partial t} = \iint_{\Sigma} \eta \nabla u \cdot \mathbf{n} \, d\sigma, \tag{10.28}$$
where η is the conductivity (again assumed constant). The constancy of ρ, κ and η amounts to assuming the substance in question is homogeneous.

Now we differentiate (10.27) with respect to time and then apply the Divergence Theorem to (10.28) to obtain:

$$\begin{aligned} \iiint_{\mathscr{V}} \rho\kappa u_t \, dV &= \frac{\partial Q}{\partial t} \\ &= \iint_{\Sigma} \eta \nabla u \cdot \mathbf{n} \, d\sigma \\ &= \iiint_{\mathscr{V}} \eta \nabla \cdot (\nabla u) \, dV \\ &= \eta \iiint_{\mathscr{V}} \nabla^2 u \, dV. \end{aligned} \tag{10.29}$$

That is

$$\iiint_{\mathscr{V}} [\rho\kappa u_t - \eta\nabla^2 u]\, dV = 0. \tag{10.30}$$

But in a standard trick of multivariable calculus, we observe that the solid region $\mathscr{V}$ is arbitrary inside the body in question. Therefore, the integral—since it integrates to zero over any $\mathscr{V}$—must be identically zero. (Strictly speaking, the conclusion is that it is zero almost everywhere, but all functions that we are dealing with here are continuous.) Hence

$$u_t = k\nabla^2 u = k(u_{xx} + u_{yy} + u_{zz}) \tag{10.31}$$

where $k = \eta/\rho\kappa$. Equation (10.31) is called the *heat equation*. It is one of the most fundamental and studied *partial differential equations* in all of mathematics. You will be able to consider a solution via MATLAB in a special case in Problem 10.10.

10.5.2 The Wave Equation

To make matters simple, let's assume the medium is a vacuum and we have a charge-free, current-free setting. In that case Maxwell's Equations (see (10.20)), take the form:

$$\begin{aligned} \nabla \times \mathbf{E} &= -\frac{1}{c}\dot{\mathbf{B}}, \\ \nabla \cdot \mathbf{E} &= 0, \\ \nabla \times \mathbf{B} &= \frac{1}{c}\dot{\mathbf{E}}, \\ \nabla \cdot \mathbf{B} &= 0, \end{aligned} \tag{10.32}$$

where c is the speed of light and the elevated dot denotes differentiation with respect to t. We take the curl of the two curl equations to obtain

$$\begin{aligned} \nabla \times (\nabla \times \mathbf{E}) &= -\frac{1}{c}\nabla \times \dot{\mathbf{B}} = -\frac{1}{c^2}\ddot{\mathbf{E}} \\ \nabla \times (\nabla \times \mathbf{B}) &= \frac{1}{c}\nabla \times \dot{\mathbf{E}} = -\frac{1}{c^2}\ddot{\mathbf{B}}. \end{aligned} \tag{10.33}$$

But we have the vector identity

$$\nabla \times (\nabla \times \mathbf{v}) = \nabla(\nabla \cdot \mathbf{v}) - \nabla^2 \mathbf{v}.$$

Applying this identity to $\mathbf{E}$ and $\mathbf{B}$ and using $\nabla \cdot \mathbf{E} = \nabla \cdot \mathbf{B} = 0$, we get

$$\begin{aligned} \nabla \times (\nabla \times \mathbf{E}) &= -\nabla^2 \mathbf{E} \\ \nabla \times (\nabla \times \mathbf{B}) &= -\nabla^2 \mathbf{B}. \end{aligned} \tag{10.34}$$

Therefore,

$$\begin{aligned} \nabla^2 \mathbf{E} &= \frac{1}{c^2} \ddot{\mathbf{E}} \\ \nabla^2 \mathbf{B} &= \frac{1}{c^2} \ddot{\mathbf{B}}. \end{aligned} \tag{10.35}$$

If we let $u(t, \mathbf{x})$ be any one of the coordinates of $\mathbf{E}$ or $\mathbf{B}$, we obtain the wave equation as it is usually written

$$u_{tt} = c^2 \left(u_{xx} + u_{yy} + u_{zz} \right),$$

arguably the second most famous partial differential equation. As with the heat equation, we'll explore the application of MATLAB to this partial differential equation in the problem set.

Problem Set J. Physical Applications

Problem 10.1. Here's a famous problem in electrostatics. The objective is to compute the electrostatic potential and electrical field in the half-space $z \geq 0$ of $\mathbb{R}^3$ due to a point charge Q located at the point $(0,\ 0,\ 1)$ (i.e., one unit above the x-y plane), assuming the x-y plane is an infinite conducting plate. This gives a crude idealized model for the electrical field in a thunderstorm: the x-y plane represents the ground and the point $(0,\ 0,\ 1)$ represents the location of a charged thundercloud.

First observe that the electrostatic potential and electrical field are the same as those due to *two* point charges, Q at $(0,\ 0,\ 1)$ and $-Q$ at $(0,\ 0,\ -1)$. Why is this? Add the potentials due to the two charges (separately), and differentiate the potential to compute the electric field. We will look at the cross sections of these fields in the x-z plane. Use the MATLAB commands **`fcontour`** and **`quiver`** to plot the electrostatic potential and electrical field, respectively, in the portion of the x-z plane with $-1 \leq x \leq 1$, $0 \leq z \leq 2$, when $Q = 1$. Combine the two plots on the same graph.

Problem 10.2. Here is another problem in electrostatics. Compute and draw a contour plot of the electrostatic potential ϕ in $\mathbb{R}^3$ due to two parallel conducting circular loops which are oppositely charged (with charges of the same absolute value). For convenience we'll assume the charged loops are located at the circles $\{x^2 + y^2 = 1,\ z = 1\}$ and $\{x^2 + y^2 = 1,\ z = -1\}$. This is a crude model for a *capacitor*, a circuit element consisting of two oppositely charged conductors separated by a small nonconducting gap. Here is how to do the calculation. In the last problem, you computed the electrostatic potential at a point $\mathbf{r}$ in $\mathbb{R}^3$ due to a charge of $+1$ at $(0,\ 0,\ 1)$ and a charge of -1 at $(0,\ 0,\ -1)$. Simply by shifting, this gives the potential at $\mathbf{r}$ due to a charge of $+1$ at $(\cos\theta,\ \sin\theta,\ 1)$ and a charge of -1 at $(\cos\theta,\ \sin\theta,\ -1)$. You then want to integrate this potential over θ as θ runs from 0 to 2π. (By circular symmetry, the charge is uniform in θ.) In fact, MATLAB can do the integral to compute $\phi(x,\ 0,\ z)$. First compute the integrand as a function of x, z, and θ, by replacing $(x,\ y,\ z)$ in the potential field of Problem 10.1 by $(x - \cos\theta,\ \sin\theta,\ z)$, and applying **`simplify`**. (You will probably have to increase **`Steps`** to get the simplest form.) Trying to integrate this directly over θ, as θ runs from 0 to 2π, doesn't seem to work. (Try it!) But you can compute the integral with MATLAB by first computing

$$\int_0^{2\pi} \frac{1}{\sqrt{a - 2\cos(\theta)}}\, d\theta$$

(be sure to tell MATLAB to **`assume`** that $a > 2$) and then writing the integral for the potential in terms of this (for appropriate choices of a). The answer will be in terms of a special function. Draw contour plots of the solution for ϕ in the region $0 \leq x \leq 4$, $0 \leq z \leq 4$. You get the picture for $-4 \leq x \leq 4$, $-4 \leq z \leq 4$, by reflecting. Then differentiate the potential function to compute the electric field. Do not be intimidated by the length and complexity of the answers. Plot the field in the first quadrant of the x-z plane using **`quiver`**.

Problem 10.3. For applications in astrophysics, it is often important to understand the gravitational fields due to various mass distributions. As you have seen in this chapter, there are two main equations for studying such fields: Gauss' law and Poisson's equation. In this problem, we will study both of these equations in the case of mass distributions with cylindrical symmetry.

(a) Suppose the mass density in space, ρ, is a function only of the radial coordinate r in cylindrical coordinates. Then the gravitational field $\mathbf{G}$ must also have cylindrical symmetry, and so (since the force is attractive) points straight inward toward the z-axis. Thus $\mathbf{G}$ is of the form $-g(r)(x\mathbf{i} + y\mathbf{j})/r$, where $g(r)$ is the field strength. (Here $-(x\mathbf{i} + y\mathbf{j})/r$ is a unit vector pointing toward the z-axis.) Let us apply Gauss' law to a right circular cylinder $\mathbf{\Sigma}$ centered along the z-axis, with height h and radius a. Since $\mathbf{G}$ points toward the z-axis, there is no flux through the top and bottom of the cylinder. Thus Gauss' law becomes

$$\text{flux through } \mathbf{\Sigma} = (\text{area of sides of } \mathbf{\Sigma}) \cdot (-g(a)) = -4\pi G(\text{mass enclosed by } \mathbf{\Sigma}),$$

or

$$(2\pi ah)g(a) = 4\pi G \int_0^{2\pi} \int_0^h \int_0^a \rho(r) r \, dr \, dz \, d\theta.$$

(Note the integration factor in cylindrical coordinates.) Compute the θ and z integrals by sight; then solve for $g(a)$ in terms of $\int_0^a \rho(r) r \, dr$. Then specialize in the case of the following density functions:

(i) $\rho(r) = \begin{cases} \rho \text{ (a constant)}, & r \le 1, \\ 0, & r > 1. \end{cases}$

(ii) $\rho(r) = \dfrac{1}{(1 + r^2)^2}$.

(iii) $\rho(r) = e^{-r^2}$.

(iv) $\rho(r) = r^2 e^{-r^2}$.

Evaluate the integrals to compute $g(r)$ as a function of r, using $G = 1$. Plot the results. What happens to the field strength $g(r)$ as $r \to \infty$?

(b) Recall from this chapter that the gravitational field $\mathbf{G}$ is given by $\mathbf{G} = -\nabla\phi$, where ϕ (the gravitational potential) satisfies Poisson's equation

$$\nabla^2\phi = 4\pi G\rho.$$

If ρ is a function of r only, then the same will be true for ϕ, again by symmetry considerations. In the following, it is advisable to use *cylindrical coordinates*. The left-hand side of Poisson's equation will then be $\frac{d^2\phi}{dr^2} + \frac{1}{r}\frac{d\phi}{dr}$. For each of the mass distributions (i)–(iv) of (a), solve Poisson's equation using **`dsolve`** and verify that you get the same formulas for the field strength function as in (a). You

may take $G = 1$. (Notes: The potential functions may involve some functions you haven't seen before. If so, you can learn about them using the MATLAB Help browser. Also, the solution to a second-order differential equation involves two constants of integration. One of these will be constrained by the requirement that ϕ not blow up as $r \to 0_+$. You can use **`series(phi(r), r)`** to compute the first few terms of the power series of the solution function **`phi(r)`**, and then set the coefficients of $\log r$ terms or of negative powers of r to zero. This will still leave one constant of integration undetermined, but it will go away when you take the gradient to compute the field strength.)

Problem 10.4. This problem is almost identical to the last one, except that we will investigate mass distributions with spherical symmetry instead of those with radial symmetry. Take r to be the radial coordinate in *spherical coordinates*, and let $\mathbf{\Sigma}$ be a sphere centered at the origin with radius a. We again assume the mass density ρ is a function of r alone. This time the gravitational field takes the form $\mathbf{G} = -g(r)(x\mathbf{i} + y\mathbf{j} + z\mathbf{k})/r$, and Gauss' law becomes

$$\text{flux through } \mathbf{\Sigma} = (\text{area of } \mathbf{\Sigma}) \cdot (-g(a)) = -4\pi G(\text{mass enclosed by } \mathbf{\Sigma}),$$

or

$$(4\pi a^2)g(a) = 4\pi G \int_0^{2\pi} \int_0^{\pi} \int_0^{a} \rho(r) r^2 \, dr \, \sin\phi \, d\phi \, d\theta.$$

Redo all the calculations of parts (a) and (b) of the last problem for cases (i)–(iv). This time you will need to use spherical coordinates. Note that the radial part of the Laplacian in spherical coordinates is $\frac{d^2 f}{dr^2} + \frac{2}{r}\frac{df}{dr}$.

(In part (b), name the potential function something other than **`phi`** to avoid conflict of notation.) Are there any differences between your results this time and the cylindrical case from Problem 10.3? Pay attention to the rates of decay at infinity.

Problem 10.5. The most fundamental mathematical problem in both gravitation and electrostatics is the solution of *Laplace's equation* $\nabla^2\phi = 0$ for the potential function. (This equation also crops up in problems about fluid flow and heat flow.) Hence, it is of particular importance to have good methods for solving this equation with various boundary conditions. An important special case is the *Dirichlet problem*, where $\nabla^2\phi = 0$ is to be satisfied by a function ϕ defined in a region $\mathscr{R}$, and ϕ is specified *a priori* on the boundary of the region. In this case, it is known under reasonable conditions (see the Appendix below) that the solution is the function ϕ with the specified boundary values for which the energy $\iiint \|\nabla\phi\|^2 \, dV$ is minimized. This suggests the following numerical algorithm for solving the Dirichlet problem: (1) Start with some reasonable function ϕ_0 defined in $\mathscr{R}$ with the given values on the boundary. (This function will not usually satisfy Laplace's equation.) (2) Find a general class of functions ϕ_1 defined in $\mathscr{R}$ which are relatively "tractable" and which *vanish* on the boundary of $\mathscr{R}$. (3) Find the function $\phi = \phi_0 + \phi_1$ for which the energy integral is minimized (among all possibilities for ϕ_1). This should be a good approximation to a solution.

Now let's carry out this algorithm in a simple case. For convenience, we consider a two-dimensional instead of a three-dimensional situation; the mathematics is similar but the calculations are less complicated. Say we want to solve $\nabla^2\phi = 0$ in the interior of the square $\{0 \le x \le 1,\ 0 \le y \le 1\}$, and we're given that ϕ vanishes on the lower three sides of the square and is given on the top side $\{0 \le x \le 1,\ y = 1\}$ by $\phi(x,\ 1) = x(1 - x)$. A function ϕ_0 satisfying these boundary conditions is $\phi_0(x,\ y) = yx(1 - x)$. (Check this.) For a function ϕ_1 vanishing on the sides of the square, we can use

$$\phi_1(x,\ y) = x(1 - x)y(1 - y)(\text{any polynomial in } x,\ y).$$

Let the factor in parentheses be the most general polynomial in two variables of total degree 3. Then compute the energy integral (as a function of the coordinates) and search for a minimum using **`fminsearch`**. You should not expect a simple form for the minimizing function. Draw a contour plot of your approximate solution for ϕ, and compare it with a contour plot of ϕ_0 (which does not satisfy Laplace's equation). Interpret the difference between the two plots.

Problem 10.6. In this problem, we use the discussion of Newtonian gravitation in this chapter to study the gravitational field of the Earth (or other planets).

(a) According to equation (10.14) in this Chapter, the gravitational field at the surface of a spherically symmetric spherical body points in toward the center of the object, with field strength $g = GM/R^2$, where G is the universal gravitational constant, M is the mass of the object, and R is the radius. Compute the mass and density of the Earth (to three significant digits), given that $G = 6.67408 \times 10^{-8}\,\text{cm}^3/(\text{g} \cdot \text{sec}^2)$, the gravitational field g at the surface of the Earth has field strength $978.0\,\text{cm/sec}^2$, and the radius R of the Earth is about 6.37×10^8 cm. Do the same calculation for the Moon, which has radius about 1.74×10^8 cm and gravitational acceleration about $162.7\,\text{cm/sec}^2$. What might account for the difference in densities? (If you don't know, you can find the answer, as well as some additional information relevant to the next part of the problem, in a book on planetary science such as [1].)

(b) The gravitational field of the Earth is of considerable importance for such purposes as rocket and satellite navigation, and it's known to an extraordinary degree of precision. There are variations from the model of a perfectly symmetrical planet due to a variety of factors: inhomogeneity of the Earth's interior, variations in surface topography, and above all, flattening of the Earth at the poles due to the centrifugal force coming from the Earth's rotation. In addition, centrifugal force itself exerts an effect on objects on the Earth's surface which is indistinguishable from weak *negative gravity* (pointing in the opposite direction from true gravity and varying quite considerably with latitude). In the rest of this problem, we will explore some of these factors.

To a reasonable degree of accuracy, the shape of the Earth is a spheroid (i.e., an ellipsoid with radial symmetry about its axis of rotation), with equatorial radius $a = 6.378139 \times 10^8$cm and polar radius c, where the flattening constant is

$$\frac{a-c}{a} = 3.35282 \times 10^{-3}.$$

Our aim in the rest of the problem is to compute the gravitational field strength at the surface of such a spheroid in terms of GM, a, c, and the latitude φ. (Be sure to distinguish between the potential ϕ and the latitude φ. Recall that $\varphi = \pi/2$ corresponds to the North Pole, $\varphi = 0$ corresponds to the equator, and $\varphi = -\pi/2$ corresponds to the South Pole.) You may assume for purposes of this calculation that the density is constant. Here's a hint as to how to proceed. If $a = c$, the spheroid is a sphere and everything is spherically symmetric. In this case, we know from formula (10.14) in this Chapter that the gravitational field strength at a distance $r \le a$ from the center of the object is

$$\frac{G}{r^2}(\text{mass of a sphere of radius } r) = \frac{G}{r^2}\frac{r^3}{a^3} \cdot M = \frac{GMr}{a^3}.$$

The gravitational potential $\phi_{\text{grav}}(r)$ may therefore be taken to be

$$\int_0^r \frac{GMx}{a^3}\,dx = \frac{GMr^2}{2a^3} = \frac{GM}{2a^3}\left(x^2+y^2+z^2\right), \qquad r \le a,$$

and this satisfies Poisson's equation (equation (10.16) in this Chapter) with $\rho = 3M/(4\pi a^3)$, i.e.,

$$\nabla^2\phi_{\text{grav}} = 4\pi G\rho = \frac{3GM}{a^3}, \qquad r \le a.$$

Now the spheroid

$$\frac{\left(x^2+y^2\right)}{a^2} + \frac{z^2}{c^2} = 1 \tag{10.36}$$

is the image of the sphere $x^2+y^2+z^2 = a^2$ under the linear change of coordinates

$$x \mapsto x, \qquad y \mapsto y, \qquad z \mapsto \frac{cz}{a},$$

which has Jacobian

$$\det\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & c/a \end{pmatrix} = \frac{c}{a},$$

so the volume of the spheroid is

$$\frac{c}{a}\frac{4}{3}\pi a^3 = \frac{4}{3}\pi a^2 c.$$

The *gravitational potential* ϕ_{grav} for a spheroidal object of constant density and mass M should therefore satisfy

$$\nabla^2 \phi_{\text{grav}} = 4\pi G\rho = \frac{3GM}{a^2 c} \tag{10.37}$$

in the interior of the object.
However, on a rotating planet, the *effective potential* is

$$\phi_{\text{eff}} = \phi_{\text{grav}} + \phi_{\text{cent}},$$

where ϕ_{cent} is the *centrifugal potential*, i.e., $-\nabla\phi_{\text{cent}}$ is the centrifugal acceleration. Compute ϕ_{cent}, assuming that the spheroid (10.36) is rotating about the z-axis with angular velocity ω. The potential ϕ_{cent} should be a constant multiple of $x^2 + y^2$.

(c) Finally, to compute ϕ_{grav}, we use the principle that the effective potential

should be constant on the surface of the object. (10.38)

This condition corresponds to the physical principle that gravity and centrifugal forces should not cause the oceans to flow from one part of the Earth to another, but that instead, *sea level* should be a surface of constant (effective) potential. Thus the solution for ϕ_{eff} should be of the form

$$\phi_{\text{eff}}(x,\ y,\ z) = C\left(x^2 + y^2 + \left(\frac{az}{c}\right)^2\right),$$

which ensures that (10.38) is automatically satisfied, and that the effective gravitational force $\mathbf{G}_{\text{eff}} = -\nabla\phi_{\text{eff}}$ is normal to the surface of the planet. Compute—in terms of c, G, M, and a—the correct value of C. To do this, subtract off ϕ_{cent}, which you already computed, and adjust C to satisfy (10.37). Then take the gradient to compute the effective gravitational field $\mathbf{G}_{\text{eff}}$. Compute the field strength as a function of φ by putting $\mathbf{r} = (a\cos\varphi,\ 0,\ c\sin\varphi)$, and then evaluating $g = \|\mathbf{G}(\mathbf{r})\|$, φ the latitude along the spheroid (10.36). Note that the effective gravitational field *does not always point toward the center of the object* in the spheroidal case. Compute what you get numerically for $g(\varphi)$ using the parameters of the Earth given above. Substitute the values of a and c given earlier, take $GM = 3.986004 \times 10^{20}\,\text{cm}^3/\text{sec}^2$, and take

$$\omega = \frac{2\pi}{24\,\text{hr}} = \frac{\pi}{12 \cdot 60^2\,\text{sec}}.$$

Compare your result with the International Gravity Formula, which in units of cm/sec^2 is

$$g(\varphi) = 978.03268\left(1 + 0.00527904\sin^2\varphi + 0.0000232718\sin^4\varphi\right).$$

Problem 10.7. Maxwell's equations imply that an electric current induces a magnetic field. Suppose for simplicity that a steady-state current J is flowing in a loop around the unit circle $x^2 + y^2 = 1$ in the x-y plane. The *Biot-Savart law* computes the induced magnetic field at a point in $\mathbb{R}^3$ with position vector $\mathbf{r}$ to be

$$\mathbf{B}(\mathbf{r}) = \frac{J}{c}\int_0^{2\pi} \frac{d}{d\theta}(\cos\theta,\ \sin\theta,\ 0) \times \frac{\mathbf{r} - (\cos\theta,\ \sin\theta,\ 0)}{\|\mathbf{r} - (\cos\theta,\ \sin\theta,\ 0)\|^3}\, d\theta,$$

where c is the speed of light. Because of the rotational symmetry of the problem around the z-axis, it is only necessary to compute this for $\mathbf{r} = x\mathbf{i} + z\mathbf{k}$ in the x-z plane. The numerator in the integrand then becomes

$$\begin{aligned}
(-\sin\theta\,\mathbf{i} + \cos\theta\,\mathbf{j}) &\times (\mathbf{r} - \cos\theta\,\mathbf{i} - \sin\theta\,\mathbf{j})\ d\theta \\
&= \big(-\sin\theta\,\mathbf{i} \times (x\mathbf{i} + z\mathbf{k} - \cos\theta\,\mathbf{i} - \sin\theta\,\mathbf{j}) \\
&\qquad + \cos\theta\,\mathbf{j} \times (x\mathbf{i} + z\mathbf{k} - \cos\theta\,\mathbf{i} - \sin\theta\,\mathbf{j})\big)\, d\theta \\
&= \left(z\sin\theta\,\mathbf{j} + \sin^2\theta\,\mathbf{k} - x\cos\theta\,\mathbf{k} + z\cos\theta\,\mathbf{i} + \cos^2\theta\,\mathbf{k}\right)\, d\theta \\
&= (z\sin\theta\,\mathbf{j} + (1 - x\cos\theta)\mathbf{k} + z\cos\theta\,\mathbf{i})\ d\theta.
\end{aligned}$$

The denominator is

$$\begin{aligned}
\|\mathbf{r} - (\cos\theta,\ \sin\theta,\ 0)\|^3 &= \left((x - \cos\theta)^2 + \sin^2\theta + z^2\right)^{3/2} \\
&= \left(x^2 - 2x\cos\theta + \cos^2\theta + \sin^2\theta + z^2\right)^{3/2} \\
&= \left(x^2 - 2x\cos\theta + 1 + z^2\right)^{3/2},
\end{aligned}$$

which is a non-negative even function of θ. The product of the denominator with $\sin\theta$ is an odd function of θ, and so the **j**-term in the integral cancels out. Thus there are only two components of the integral to compute. Take the constant J/c to be 1.

(a) Compute the **i** and **k** components **`field1`** and **`field3`** of the magnetic field as functions of x and z, and use the command **`quiver`** to view the magnetic field in the region $-3 \le x,\ z \le 3$ of the x-z plane. The integrals are complicated and involve special functions, so depending on how you set things up, you may or may not be able to do the integrals symbolically. But you should at least be able to define a function to compute them numerically. The picture of the field should resemble the apparent patterns that you get by placing iron filings on a sheet of paper in the magnetic field, as you might remember from high-school physics.

(b) The **k**-component of the field is the only component that is non-zero in the x-y-plane, and it clearly has rotational symmetry around the z-axis. Plot with **fplot3** the **k**-component of the field as a function of x for $0 \le x \le 3$ when $y = z = 0$.
(c) It is often convenient to visualize the magnetic field in terms of "lines of force." These are the curves traced out by a point with position vector $\mathbf{r}$ satisfying the differential equation of motion

$$\dot{\mathbf{r}} = \mathbf{B}(\mathbf{r}).$$

Use your answer to (a) and **ode45** to compute a few lines of magnetic force around the current loop, and then plot these in the x-z plane using **plot**. Here are a few hints on how to set this up. You have to be careful since the equations are quite complicated and take a long time to solve, even on a fast computer. Please be patient! If **f1** and **f2** are the functions you obtained for the **i** and **k** components of the field, then the commands

```
traj = @(t, x) [f1(x(1), x(2)); f2(x(1), x(2))];
[t, xa] = ode45(traj, [0 3], [x0; 0];
plot(xa(:, 1), xa(:, 2))
```

will plot the line of force in the x-z plane through the point $(x_0, 0, 0)$ for $0 \le t \le 3$. Try taking $x_0 = (10j + 1)/60$, $0 \le j \le 5$ and use **hold on** to plot all the trajectories on the same axes. (Be careful with the semicolons here. The first argument to **ode45** has to be a function, whose output and second input are both column vectors, not row vectors. If the commands still take too long on your computer, you may need to settle for fewer trajectories or a shorter time interval.) You should see a familiar pattern of nested loops of magnetic lines of force. The picture looks more impressive if you reflect it across the x- and y-axes to see the picture in the whole plane.

Problem 10.8. This problem deals with steady incompressible fluid flow, as discussed at the end of this chapter.

(a) Verify that equation (10.25), for steady incompressible fluid flow, is satisfied when $\mathbf{curl}\,\mathbf{v} = \mathbf{0}$ (the irrotational case). To do this, first check the identity

$$\nabla\mathbf{v} \cdot \mathbf{v} = (\mathbf{curl}\,\mathbf{v}) \times \mathbf{v} + \tfrac{1}{2}\nabla(\mathbf{v} \cdot \mathbf{v}),$$

which holds for general vector fields $\mathbf{v}$.
(b) Using the identity from (a), in the irrotational case where $\mathbf{curl}\,\mathbf{v} = \mathbf{0}$, deduce from equation (10.24) the principle known as *Bernoulli's law* for steady, irrotational, incompressible fluid flow; namely, that $p + (\rho\|\mathbf{v}\|^2/2)$ remains constant.
(c) Suppose the equations $\operatorname{div}\mathbf{v} = 0$ and $\mathbf{curl}\,\mathbf{v} = \mathbf{0}$ for steady, irrotational, incompressible fluid flow are satisfied for a fluid contained in a region $\mathscr{R}$ of $\mathbb{R}^3$. Then, at least locally in $\mathscr{R}$, $\mathbf{v}$ is a conservative field and we can write $\mathbf{v} = \nabla\phi$

for some *potential function* ϕ. The condition $\operatorname{div} \mathbf{v} = 0$ then becomes *Laplace's equation* $\nabla^2 \phi = 0$. Since fluid cannot flow through the sides of a closed container, we have the additional boundary condition $\partial \mathbf{v}/\partial n = 0$ on the boundary of $\mathscr{R}$, or $\mathbf{n} \cdot \nabla \phi = 0$, where $\mathbf{n}$ is the outward unit normal. Show that if $\mathscr{R}$ is the half-space given by $x \geq 0$ in Cartesian coordinates (x, y, z), then $\phi(x, y, z) = y$ is a valid solution to the equations. Draw contours of the potential function and plot the velocity field for $0 \leq x \leq 3$ and $-3 \leq y \leq 3$. Superimpose the two plots to show the potential function and the velocity field simultaneously.

(d) Suppose $\mathscr{R}$ is the wedge given in cylindrical coordinates (r, θ, z) by $|\theta| \leq \pi/3$. Show that $\phi(r, \theta, z) = r^{3/2} \sin(3\theta/2)$ satisfies Laplace's equation and the boundary condition for steady, irrotational, incompressible fluid flow in $\mathscr{R}$. (*Hint*: Compute the Laplacian in cylindrical coordinates.) Again, draw contours of the potential function and plot the velocity field, then superimpose the two. You can take $0 \leq x \leq 3$ and $-3 \leq y \leq 3$. (Hint: To block out the part of this rectangular region that is not physically relevant, use the command **`fill`** to create a blackened triangle with vertices $(0, 0)$, $(\sqrt{3}, 3)$, and $(0, 3)$. Then superimpose it onto the plots to block out the irrelevant part of the first quadrant. Similarly with the fourth quadrant.) Explain what is going on in the flow.

(e) In fluid flow problems, it is often useful to look at the *streamlines*, that is, the curves that would be traced by objects "drifting with the current." Since the streamlines must everywhere be tangent to the velocity field, they must be perpendicular to the level curves of the potential function ϕ. (Why? If necessary, refer back to Chapter 5 on *Directional Derivatives* and to Problem 5.2 in Problem Set E.) Show that the gradient of the function $h = r^{3/2} \cos(\frac{3}{2}\theta)$ is orthogonal to the gradient of ϕ, and thus that the streamlines are the level curves of h. Draw a contour plot of h and superimpose it on your plots in (d) to visualize the streamlines for the flow.

Problem 10.9. Here is another problem on steady, incompressible, irrotational fluid flow. Consider such a flow in the half-space $x > 0$, but around an *obstacle* located at $\{0 \leq x \leq 1, \ y = 0\}$. (In this problem, we have assumed that the z-coordinate plays no role, so the problem is effectively two-dimensional.) Thus, the region $\mathscr{R}$ where Laplace's equation is to be satisfied is

$$\mathscr{R} = \{(x, y) \in \mathbb{R}^2 : x > 0, \text{ and } x > 1 \text{ if } y = 0\}.$$

Show that $\phi(x, y) = \sqrt{r} \sin(\theta/2)$ satisfies Laplace's equation and the correct boundary condition for the potential function, provided that (r, θ) are polar coordinates for the points $(x^2 - y^2 - 1, 2xy)$ with $-\pi < \theta < \pi$. First check that if $(x, y) \in \mathscr{R}$, then the point $(x^2 - y^2 - 1, 2xy)$ can't lie on the negative x-axis, so that we can indeed choose θ to lie in the range $-\pi < \theta < \pi$. Then check the equations. As in the last problem, draw contours of the potential function and plot the velocity field for $0 \leq x \leq 3$ and $-3 \leq y \leq 3$; then superimpose the two. (You may want to adjust the lengths of the vectors.) As before, you can blacken out the obstacle by creating a blackened line segment with the command **`fill`**. Explain what is going on in

the flow. As in the last problem, show that the streamlines are the level curves of $g(x, y) = \sqrt{r}\cos(\theta/2)$, and plot them using a contour plot of the function g. Combine the streamlines with the vector field in a single picture. (*Hint*: At a certain point, you will need to use the *arctan* function. Be sure to invoke MATLAB's 4-quadrant version, **`atan2`**, rather than the single-quadrant version **`atan`**.)

Problem 10.10. In this problem, we solve an instance of the heat equation, as discussed in Section 10.5.1. For simplicity we'll deal just with a one-dimensional bar of material corresponding to the interval $[0, \pi]$. Assume that the ends of the bar are immersed in ice water and thus kept at a constant temperature of 0 (in degrees Celsius). Assume that before time $t = 0$, the whole bar is heated to a constant temperature of 1 (in suitable units) and that in these same units, the constant k in the heat equation is 1. We then end up with the following problem for $t \geq 0$:

$$\begin{cases} u_t = u_{xx}, & 0 < x < \pi,\ t > 0, \\ u(x, 0) = 1, & 0 < x < \pi, \\ u(0, t) = 0, & t > 0, \\ u(\pi, t) = 0, & t > 0. \end{cases} \tag{10.39}$$

We will solve the problem using some elementary Fourier Series. For those unfamiliar with the subject, there are numerous basic references, e.g., [2].

(a) Solve (10.39) by assuming a solution of the form

$$u(x, t) = \sum_{n=1}^{\infty} a_n(t)\, \sin(n\, x).$$

Note that this already satisfies the boundary condition at $x = 0$ and $x = \pi$. Since

$$\frac{d^2}{dx^2} \sin(n\, x) = -n^2 \sin(n\, x),$$

the differential equation will be satisfied provided that $a_n(t)$ solves the differential equation $y'(t) = -n^2 y(t)$. Solve this using **`dsolve`**; you should find that $a_n(t) = b_n e^{-n^2 t}$ for some constants b_n.

(b) Solve for the constants b_n so that the formula for $u(x, 0)$ matches the Fourier series of the periodic step function which is $+1$ for $0 < t < \pi$ and -1 for $-\pi < t < 0$. That means you should get the same result on both sides when you multiply by $\sin(m\, x)$ and integrate over $[0, \pi]$. Use the fact that

$$\int_0^{\pi} \sin(n\, x)\, \sin(m\, x)\, dx = 0$$

when $n \neq m$, so you just need to compute

$$\int_0^\pi \sin(n\,x)\,dx \quad \text{and} \quad \int_0^\pi \sin^2(n\,x)\,dx,$$

which you can compute with MATLAB.

(c) Plot the solution for $t = 0.1$ and for $t = 0.2$, by taking 40 terms in the series. (This is enough to get reasonably accurate results.) Also plot $u(\frac{\pi}{2}, t)$ as a function of t and observe how it decays to 0 as $t \to \infty$.

Problem 10.11. In this problem, we solve an instance of the wave equation, as discussed in Section 10.5.2. For simplicity, we'll deal just with a "one-dimensional" wave corresponding to an "infinite" elastic string defined initially on the interval $(-\infty, \infty)$, but vibrating in 2-space $-\infty < x, y < \infty$. This obviates the need for boundary conditions, but we shall assume initial conditions on position and velocity. Thus the wave equation in this instance becomes, for $t \geq 0$:

$$\begin{cases} u_{tt} = u_{xx}, & -\infty < x < \infty,\ t > 0, \\ u(x,0) = f(x), & -\infty < x < \infty, \\ u_t(x,0) = g(x), & -\infty < x < \infty, \end{cases} \tag{10.40}$$

We have simplified matters slightly by assuming the wave equation coefficient is 1.

(a) In fact, it was observed by the eighteenth century French mathematician, Jean le Rond D'Alembert, that a solution to this problem is given by the following formula:

$$u(t,x) = \frac{1}{2}\left[f(x-t) + f(x+t) + \int_{x-t}^{x+t} g(s)\,ds \right]. \tag{10.41}$$

Use MATLAB to verify that D'Alembert's equation (10.41) is in fact a solution to the problem posed, that is, it solves the wave equation and the two initial conditions.

Now let's solve some specific cases depending on choices of f and g.

(b) Solve the wave equation for the infinite string in case

$$f(x) = \begin{cases} 2, & -1 < x < 1 \\ 0 & \text{elsewhere.} \end{cases}$$

and g is identically zero. Draw the graph of f. Then draw the position of the string at times $t = 1/2, 1, 2$. You may need to adjust **`axis`** to see the displaced string more clearly.

(c) Next, make a movie, using the MATLAB commands **`getframe`** and **`movie`**, by letting the time vary between $t = 0$ and $t = 5$, with step size $1/2$.

(d) Repeat part (b) but with the smooth (instead of step) function

$$f(x) = \frac{1}{1+x^2}.$$

(e) Repeat again, but with

$$f(x) = \exp(-x^2).$$

(f) Next we'll assume no initial displacement, but a non-trivial initial velocity; that is, we'll assume f is identically zero, but g is not. Solve the wave equation for the infinite string in case

$$g(x) = \begin{cases} 1-x, & -1 < x < 0 \\ x-1, & 0 < x < 1, \\ 0 & \text{elsewhere.} \end{cases}$$

and f is identically zero. Draw the graph of g. Then draw the position of the string at times $t = 1/2, 1, 2$.

(g) Repeat with g given by

$$g(x) = \frac{-1}{1+x^2}.$$

Glossary of MATLAB Commands and Options

`assume` Set assumptions on a symbolic object

`atan2` Finds the polar coordinate θ in terms of y and x

`cross` The cross product of two vectors

`curl` The curl of a vector field

`dsolve` Symbolic differential equation solver

`fcontour` Easy 2D contour plotter

`fill` Creates a colored polygon

`fminsearch` Searches numerically for a minimum of a function of a single variable

`fplot` Easy function plotter

`fplot3` Easy 3D function plotter

`gradient` Computes the gradient of a scalar field

`heaviside` The Heaviside step function

`int` Integrates an expression

`laplacian` Computes the Laplacian of a function

matlabFunction Converts a symbolic expression to a vectorized numerical function

meshgrid Sets up a numerical 2D grid of values of x and y, for example for use with **quiver** or **contour**

quiver Draws a 2D vector field; use **meshgrid** first

series Computes a series expansion

transpose Computes the transpose of a matrix

Options to MATLAB Commands

LevelList Specifies number of contours that **fcontour** will display

MaxFunEvals Sets maximum number of function evaluations allowed in a computation or process

Order Specifies how many terms to compute in a series expansion

Steps Specifies the number of steps that **simplify** will use in its attempt to algebraically simplify an expression

Appendix: Energy Minimization and Laplace's Equation

In this Appendix, we briefly explain why the energy minimization approach to Laplace's equation, outlined and used in Problem 10.5, gives the correct answer. Consider the *Dirichlet problem* in a bounded region $\mathscr{R}$ in $\mathbb{R}^3$, which is to find a function ϕ defined in a region $\mathscr{R}$, where ϕ is given on the boundary of the region and ϕ solves the equation $\nabla^2\phi = 0$ inside $\mathscr{R}$. We take it for granted that there is indeed a unique solution. (This requires proof, which for "reasonable" regions $\mathscr{R}$ is supplied in courses on partial differential equations.) Call the solution ϕ_0. (This is not necessarily the same as the starting function ϕ_0 for the algorithm described in Problem 10.5. But if it were, the algorithm would immediately terminate and yield the correct answer.) Then a general function in $\mathscr{R}$ with the given boundary values is of the form $\phi = \phi_0 + \phi_1$, where ϕ_1 *vanishes* on the boundary. Let

$$E(\phi) = \iiint_{\mathscr{R}} \|\nabla\phi\|^2 \, dV.$$

We want to show that $E(\phi) \geq E(\phi_0)$, with equality only if $\phi = \phi_0$, i.e., only if $\phi_1 = 0$ everywhere. But

$$\begin{aligned} E(\phi) &= \iiint_{\mathscr{R}} \nabla\phi \cdot \nabla\phi \, dV \\ &= \iiint_{\mathscr{R}} (\nabla\phi_0 + \nabla\phi_1) \cdot (\nabla\phi_0 + \nabla\phi_1) \, dV \\ &= \iiint_{\mathscr{R}} \nabla\phi_0 \cdot \nabla\phi_0 \, dV + 2 \iiint_{\mathscr{R}} \nabla\phi_1 \cdot \nabla\phi_0 \, dV + \iiint_{\mathscr{R}} \nabla\phi_1 \cdot \nabla\phi_1 \, dV \\ &= E(\phi_0) + 2 \iiint_{\mathscr{R}} \nabla\phi_1 \cdot \nabla\phi_0 \, dV + \iiint_{\mathscr{R}} \|\nabla\phi_1\|^2 \, dV. \end{aligned}$$

The last term on the right is ≥ 0, and we can compute the second term using the Divergence Theorem, since by the product rule,

$$\operatorname{div}(\phi_1 \nabla\phi_0) = \nabla\phi_1 \cdot \nabla\phi_0 + \phi_1 \nabla^2\phi_0,$$

and $\nabla^2\phi_0 = 0$ by assumption. So

$$\begin{aligned} 2 \iiint_{\mathscr{R}} \nabla\phi_1 \cdot \nabla\phi_0 \, dV &= 2 \iiint_{\mathscr{R}} \operatorname{div}(\phi_1 \nabla\phi_0) \, dV \\ &= 2 \iint_{\text{boundary } \mathscr{R}} (\phi_1 \nabla\phi_0) \cdot \mathbf{n} \, dS \\ &= 0, \end{aligned}$$

since $\phi_1 = 0$ on the boundary of $\mathscr{R}$. Thus we have seen that

$$E(\phi) = E(\phi_0) + \iiint_{\mathscr{R}} \|\nabla\phi_1\|^2 \, dV,$$

which shows that

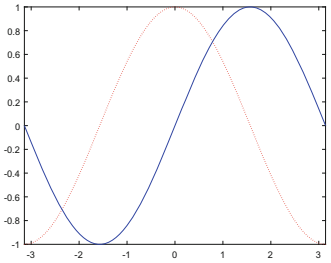
$$E(\phi) \geq E(\phi_0),$$

with equality only if $\nabla\phi_1 = 0$ everywhere. But this means ϕ_1 is a constant function, and since $\phi_1 = 0$ on the boundary of $\mathscr{R}$, $\phi_1 = 0$ everywhere.

References

1. G.S.J. Consolmagno, M.W. Schaefer, *Worlds Apart: A Textbook in Planetary Sciences*. (Prentice Hall, Englewood Cliffs, New Jersey, 1994)
2. H.F. Davis, *Fourier Series and Orthogonal Functions*. (Dover Publications, New York, 1989) (Reprint of the 1963 edn.)

Chapter 11
MATLAB Tips



In this chapter, we will use a question-and-answer format to discuss common difficulties encountered by our students. Some of the difficulties are intrinsically mathematics issues; but most are deeply intertwined with students' attempts to apply MATLAB to solve mathematics problems.

Question 1 *Why is MATLAB not doing anything? I entered a command, and nothing is happening!*

Answer First of all, if you are working in the Command Window, make sure you typed ENTER at the end of the command. Otherwise, MATLAB may simply be waiting for you to finish typing! Assuming you typed ENTER, MATLAB is not idle—it is probably churning away on a computation that is either very slow or that it cannot complete. While it is working, MATLAB will display the word *Busy* at the bottom border of the Command Window. For instance, when you enter the first input line of a session, MATLAB may take a while to respond because it is busy loading the main part of the program.

There is always a chance that some typing or syntax error has entrapped MATLAB in a process that will not terminate. If you feel that a response is overdue, click anywhere in the Command Window and type CTRL + C, that is hold down the CTRL key and press "C".

If you are working on a script in the MATLAB Editor, then of course nothing will happen until you evaluate a cell. The most common way to do so—assuming you are in a specific cell of the script—is to click on "Run Section" in the menu bar at the top. But again, if nothing seems to be happening, you may be caught in a loop or an unending function evaluation; you can stop that by typing CTRL + C in the Command window.

Question 2 *I am puzzled by the format of MATLAB's response to my input. Sometimes the response line is indented and sometimes not. Is there any rhyme or reason to this?*

Answer MATLAB is distinguishing—as should you—between numeric and symbolic data. Numerical output, such as the response to the input `85 - 56` will be

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_11

indented; whereas symbolic output, such as the response to **`syms x y; z = x^2 + y^2`**, is left-justified, never indented. This is handy when you are debugging as it offers a clue to the nature of your output. Incidentally, regarding other data classes, character string output is not indented, while function handles are.

Question 3 *I am confused between the concepts of a function and an expression in MATLAB. For example, I have to do some manipulations with the combination of monomials* $x^2 + x + 1$. *How do I enter it in MATLAB and then how do I manipulate it?*

Answer Generally $x^2 + x + 1$ will connote an *expression* in the symbolic variable x. If for example, you want to evaluate it at $x = 3$, you would input: **`syms x; expr = x^2 + x + 1; subs(expr, x, 3)`**. On the other hand, should you want to differentiate the function $x^2 + x + 1$, then a better method would be **`f = @(t) t^2 + t + 1; syms x; diff(f(x))`**.

Question 4 *I am having trouble keeping straight the different kinds of brackets and braces that MATLAB uses. What do I use where?*

Answer *Parentheses* are used both for grouping arithmetic expressions and for enclosing inputs to a MATLAB command, a script, or an anonymous function. They are also used for referring to an entry in a vector or matrix. *Square brackets* are used for defining vectors or matrices. *Single quote marks* are used for defining strings. Finally *braces* or *curly brackets* are used in cell arrays; more specifically if you want to construct an array of strings, then it has to be done with braces, since brackets when applied to strings are interpreted as concatenation.

Question 5 *Sometimes I am completely startled by the output I see; it is is not at all in conformance with what I expect. What's going on?*

Answer There are various reasons for this eventuality, but all trace back to a failure to comply with basic MATLAB protocols. For example, you may have forgotten an old variable definition. So be careful not to use variables you don't remember defining; and clear variables you no longer need—especially in a long session. Another cause is inattention to precedence of arithmetic operations. For example, if you type **`16^1/2 - 1`** expecting the answer **`3`**, you will be surprised by the output **`7`** because **`16`** is raised to the power **`1`** before the division by **`2`**. Yet another cause of output surprises is an attempt to refer to old output with **`ans`**. If you decide in a session that you wish to refer to prior output that was unnamed, then execute the command again with output assigned to a new variable. (The UP-ARROW key or Command History Window are useful for recalling the command in order to edit it.) If you depend on **`ans`**, even when it works as intended, you have less flexibility to edit your input—for example, to fix mistyped input in the Command Window, or to insert commands into a script file. Finally, do not use the same name for two different functions or variables, and in particular, do not overwrite the names of any of MATLAB's built-in functions.

Question 6 *I am baffled by the inconsistencies I see in how to enter options or modifiers to a MATLAB command. For example, when using* **plot**, *I often want to specify a color or curve style or font size, but I never can figure out how to enter these things.*

Answer Alas, there is some merit to this complaint. But generally speaking, you enter a modifier to a MATLAB command—within the parentheses that establish the command—by entering the option within single quotes (since it is almost always a character string) and then the choice of option value in single quotes if it is a character string and without quotes if it is a numerical value. For example, if you want to plot $\sin(x)$ on the interval $[0, 2\pi]$ but with a thick line style, you would type

```
>>  X=0:0.1:2*pi; Y=sin(X); plot(X, Y, 'LineWidth', 4)
```

However, if you want to plot the graph in red, it is enough to enter

```
>>  X=0:0.1:2*pi; Y=sin(X); plot(X, Y, 'r')
```

And finally, if you want to give the figure a title, then you have to do that on a new line via

```
>>  title('Sine Curve')
```

After prolonged use, you will develop a feel for the slickest way to enter options to a MATLAB command. Incidentally, the help text on a command often lists the most useful options that go along with said command.

For certain commands, such as **ode45** and **fminsearch**, the options are sufficiently complicated that a special command is needed to enter them. This command is **odeset** in the case of **ode45** and **optimset** in the case of **fminsearch**. See the help on these commands for more information.

Question 7 *I get confused about the differences between* **=** *and* **==**. *How can I tell which one to use?*

Answer The equal sign (**=**) is used to assign a value to a variable; the double equal sign **==** is used to specify an equation. Thus, for example

```
>>  syms x y(x); z = x+1; dsolve(x*diff(y) + 1 == y, x)
```

will set the symbolic variable **z** equal to **x+1** and solve the differential equation $x\frac{dy}{dx} + 1 = y$.

Question 8 *Why doesn't MATLAB simplify expressions like* $\sqrt{x^2}$*? It should report it as* x.

Answer MATLAB errs on the side of caution. After all, it does not know whether x is positive or negative. The value of $\sqrt{x^2}$ is $\pm x$ accordingly. But the expression $z/\sqrt{z^4}$ is $1/z$ regardless of the sign of the variable. And MATLAB can't cope with that one either. But in fact, if you specify that your variable is real, MATLAB does indeed cope with the latter. The input **syms z real; z/sqrt(z^4)** returns the value **1/z**. You can also sometimes achieve greater simplification with the **assume** command, as in the following example:

```
>>  syms x; assume(x>0); simplify(sqrt(x^2))

ans =
x
```

Question 9 *Sometimes the algebraic expression I see as a result of a computation is more complicated than I expect. Applying* **simplify** *often helps, but not always. Is there a more powerful tool?*

Answer Yes, there is an option, **Steps**, that will increase the number of algorithms that MATLAB uses in its attempt to simplify an expression. The following example is taken directly from the MATLAB *Help Documentation*.

```
>>  syms x; f = ...
   ((exp(-x*i)*i)/2 - exp(x*i)*i)/2/(exp(-x*i)/2 + exp(x*i)/2);
>> simplify(f)

ans =
-(exp(x*2i)*1i - 1i)/(exp(x*2i) + 1)

>>  simplify(f, 'Steps', 10)

ans =
2i/(exp(x*2i) + 1) - 1i

>>  simplify(f, 'Steps', 30)

ans =
((cos(x) - sin(x)*1i)*1i)/cos(x) - 1i

>>  simplify(f, 'Steps', 50)

ans =
tan(x)
```

Question 10 *I want to combine figures generated earlier in a session, but I can't figure out how to do it. I can of course generate one figure, then type* **hold on** *and then generate the second figure. But this is repetitive; I have generated and named the figures earlier; why can't I just issue one command now to display them on the same graph?*

Answer Here the answer depends on whether you still have both figure windows open simultaneously or not. If you want to close a figure, the best option is to use the command **savefig** to save the figure in a file. You can then reopen it with **openfig** when you need it, and by applying **hold on**, you can then add something to it. Note that the option **'compact'** to **savefig** reduces the amount of storage space required, but has the slight drawback that the resulting file can't be used in

versions of MATLAB before R2014b. Also note that if you close the figure window before saving it, then it is too late to do this and you will need to regenerate the figure window.

If you have two figure windows open simultaneously and you want to copy a plot from one into the other, then you can use **copyobj** as in the following example:

```
>>  fplot(@sin, [-pi, pi], 'b'); sincurve = get(gca, 'Children');

>>  figure; fplot(@cos, [-pi, pi], 'r:'); copyobj(sincurve, gca)
```

The result is shown in Figure 11.1. Note that you have to execute the **copyobj** command *before* closing the first figure window.

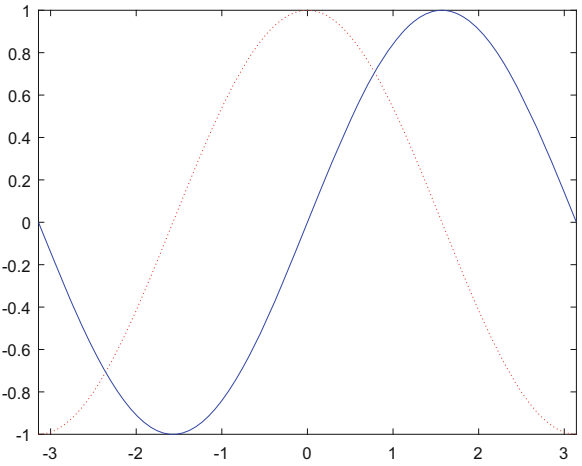


Fig. 11.1 Pasting one plot into another

Question 11 *The top of one of my plots seems to have been cut off. Another plot seems to include too much, so that the detail is washed out. What can I do about these plots?*

Answer Use the command **axis** to adjust the axes so as to render the most relevant parts of your plots.

Question 12 *Sometimes when I plot a function I get a warning message like*

```
Warning: Function fails on array inputs. Use element-wise operators
to increase speed.
```

or worse, an error message

```
Dimensions of matrices being concatenated are not consistent.
```

What is going on and how can I fix it?

Answer The phenomenon just described will occur if you try something like

```
>>  fplot(@(x) 1, [-1, 1])
```

or

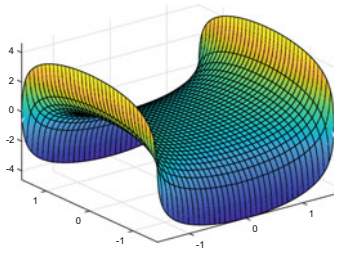
```
>>  verticalRegion(@(x) 0, @sin, 0, pi)
```

An even more inscrutable error message will result if you try

```
>>  fplot(matlabFunction(sym(1)), [-1, 1])
```

The problem in all these cases has to do with the difference between 0 or 1 as a *number* and 0 or 1 as a function of x. The warning message "Function fails on array inputs" will also be displayed if you try to plot a non-vectorized function like `@(x) x^2`. The thing to remember is that almost all MATLAB plotting routines want a *function* as input. This function should take a *vector* `x` to a vector of values *of the same size* as `x`. A way to achieve this is to replace `@(x) 0` with `@(x) zeros(size(x))` or `@(x) 2` with `@(x) 2*ones(size(x))`. Similarly you should replace `@(x) x^2` with `@(x) x.^2`.

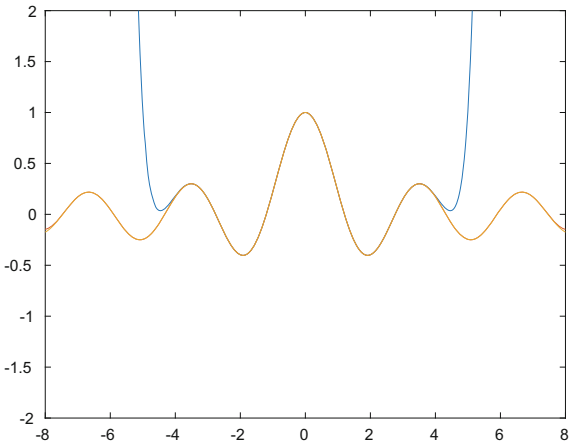
Chapter 12
Sample Solutions



12.1 Problem Set A: Problem 10

1.10(a)

```
f = @(x, k) sum(((-ones(1,k+1)).^(0:k)).* ...
    (x*ones(1,k+1)).^(2*(0:k))./factorial(0:k).^2);
figure; fplot(f(x, 10), [-8,8]), axis([-8,8,-2,2])
hold on
fplot(f(x, 20), [-8,8]), axis([-8,8,-2,2])
fplot(f(x, 40), [-8,8]), axis([-8,8,-2,2])
hold off
```



We see that f_{10} and f_{20} have almost the same graph over the interval $-3.5 < x < 3.5$, and that f_{20} and f_{40} have almost the same graph over the interval $-8 < x < 8$. We can understand this in terms of the alternating series test. For an alternating

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8_12

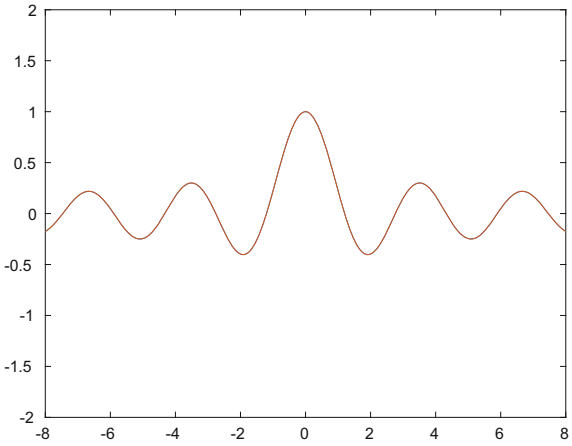
series whose terms decrease in absolute value, the error due to truncation is bounded by the absolute value of the first term omitted. The term with n = 22 for x = 8 has absolute value

```
8^44/factorial(22)^2

ans =

    0.0043
```

This is too small to have much effect on the picture at the scale we are interested in.

1.10(b)

```
syms n
symsum(((-1)^n)*x^(2*n)/factorial(n)^2, n, 0, Inf)

ans =

hypergeom([], 1, -x^2)


figure; fplot(ans, [-8, 8]), axis([-8,8,-2,2])
hold on
fplot(f(x, 40), [-8,8]), axis([-8,8,-2,2])
hold off
```

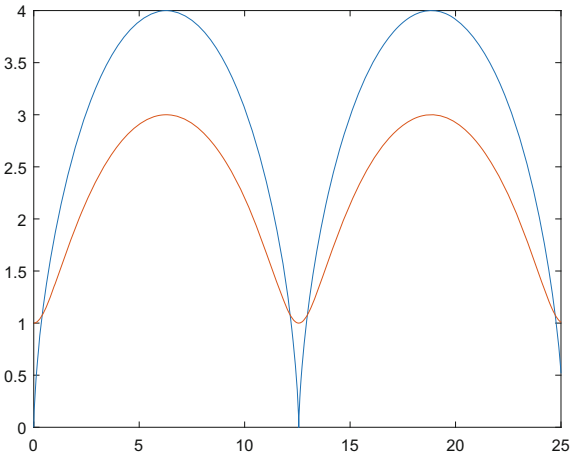


We see that the two plots almost coincide.

12.2 Problem Set B: Problem 21

First we set up the parametrizations of the cycloid and trochoid; and then we plot them.

```
clear all; close all
syms t s x y u v; cycl = [2*t - 2*sin(t), 2 - 2*cos(t)];
troch = [2*s - sin(s), 2 - cos(s)];
x = cycl(1); y = cycl(2);
u = troch(1); v = troch(2);
figure; fplot(x, y, [0 4*pi])
hold on; fplot(u, v, [0 4*pi])
axis([0 25 0 4])
```



It looks from the picture as if there are 4 intersection points, with t and s in each case just above or below a multiple of 2*pi. So the following finds the intersection points and the values of t and s corresponding to them.

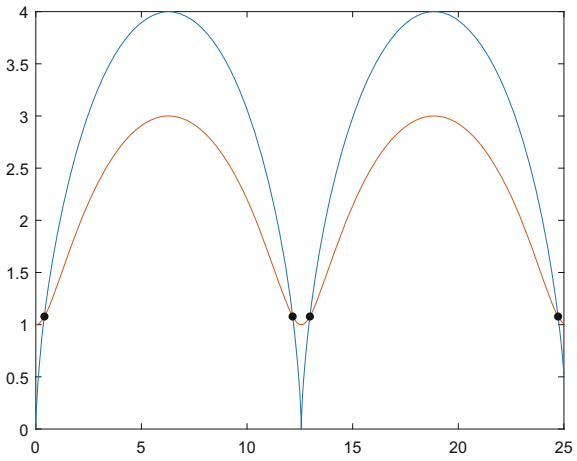
```
sol1 = vpasolve(cycl == troch, [t, s], [.1 .1]);
disp(['t = ', num2str(double(sol1.t)), '; s = ', ...
    num2str(double(sol1.s))])
disp(['[x, y] = ', num2str(double(subs(cycl, t, sol1.t)))])
sol2 = vpasolve(cycl == troch, [t, s], [2*pi-.1, 2*pi-.1]);
disp(['t = ', num2str(double(sol2.t)), '; s = ', ...
    num2str(double(sol2.s))])
disp(['[x, y] = ', num2str(double(subs(cycl, t, sol2.t)))])
sol3 = vpasolve(cycl == troch, [t, s], [2*pi+.1, 2*pi+.1]);
disp(['t = ', num2str(double(sol3.t)), '; s = ', ...
```

```
    num2str(double(sol3.s))])
disp(['[x, y] = ', num2str(double(subs(cycl, t, sol3.t)))])
sol4 = vpasolve(cycl == troch, [t, s], [4*pi-.1, 4*pi-.1]);
disp(['t = ', num2str(double(sol4.t)), '; s = ', ...
    num2str(double(sol4.s))])
disp(['[x, y] = ', num2str(double(subs(cycl, t, sol4.t)))])
```

```
t = 1.0918; s = 0.39826
[x, y] = 0.40871      1.0783
t = 5.1914; s = 5.8849
[x, y] = 12.1577      1.07826
t = 7.375; s = 6.6814
[x, y] = 12.9751      1.07826
t = 11.4745; s = 12.1681
[x, y] = 24.724      1.07826
```

Now let's mark the four points of intersection on the graph.

```
hold on;
P1 = double(subs(cycl, t, sol1.t));
P2 = double(subs(cycl, t, sol2.t));
P3 = double(subs(cycl, t, sol3.t));
P4 = double(subs(cycl, t, sol4.t));
plot(P1(1), P1(2), '.k', 'MarkerSize', 15)
plot(P2(1), P2(2), '.k', 'MarkerSize', 15)
plot(P3(1), P3(2), '.k', 'MarkerSize', 15)
plot(P4(1), P4(2), '.k', 'MarkerSize', 15)
```



12.3 Problem Set C: Problem 14

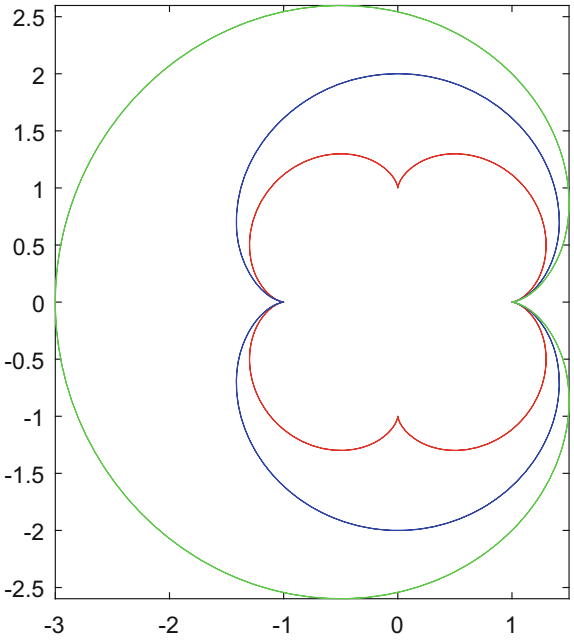
(a) parametric equations

```
syms a b t real; assume(a>0 & b>0)
r = [(a+b)*cos(t) - b*cos((a+b)*t/b), ...
  (a+b)*sin(t) - b*sin((a+b)*t/b)];
```

(b) graphs

Let's fix a = 1 (no loss of generality, since we can rescale) and try b = 0.25, 0.5, and 1.

```
r1 = subs(r, [a, b], [1, 1/4]);
r2 = subs(r, [a, b], [1, 1/2]);
r3 = subs(r, [a, b], [1, 1]);
fplot(r1(1), r1(2), [0, 4*pi], 'r'), hold on
fplot(r2(1), r2(2), [0, 4*pi], 'b')
fplot(r3(1), r3(2), [0, 4*pi], 'g'), hold off
axis equal
```



(c) speed and curvature

```
veclength = @(x) sqrt(x*transpose(x));
vel = diff(r,t);
speed = veclength(vel);
accel = diff(vel, t);
vta = simplify(cross([vel,0], [accel,0]));
curvature = simplify(veclength(vta)/speed^3);
sqspeed = simplify(speed^2, 'Steps', 50);
pretty(sqspeed)
sqcurv = simplify(curvature^2, 'Steps', 50);
pretty(sqcurv)
```

```
       2 /    / t (a + b) \          \2          2
(a + b)  | sin| --------- | - sin(t) |  + (a + b)
         \    \     b     /          /

   /    / t (a + b) \          \2
   | cos| --------- | - cos(t) |
   \    \     b     /          /

                2   2
         sign(#1)  #1
------------------------------
        2   2          2   2 3
((a + b)  #2  + (a + b)  #3 )

where

                /                     / t (a + b) \         2 \
                |                   cos| --------- | (a + b)  |
                |                      \     b     /          |
       #1 == (a + b) | cos(t) (a + b) - ------------------------- | #3 + (a + b)
                \                                 b             /

      /                     / t (a + b) \         2 \
      |                   sin| --------- | (a + b)  |
      |                      \     b     /          |
      | sin(t) (a + b) - ------------------------- | #2
      \                               b             /

           / t (a + b) \
   #2 == sin| --------- | - sin(t)
            \     b     /

           / t (a + b) \
   #3 == cos| --------- | - cos(t)
            \     b     /
```

We can see from this that the curvature vanishes only when “#1” vanishes. The curvature blows up (goes to infinity) and the speed vanishes when “#2” and “#3” vanish simultaneously, i.e., when sin(t*(a+b)/b) = sin(t) and cos(t*(a+b)/b) = cos(t). This always happens at t=0 (the cusp on the right on the figure) but whether it happens at other points or not depends on whether a/b is rational or not and what it’s exact value is.

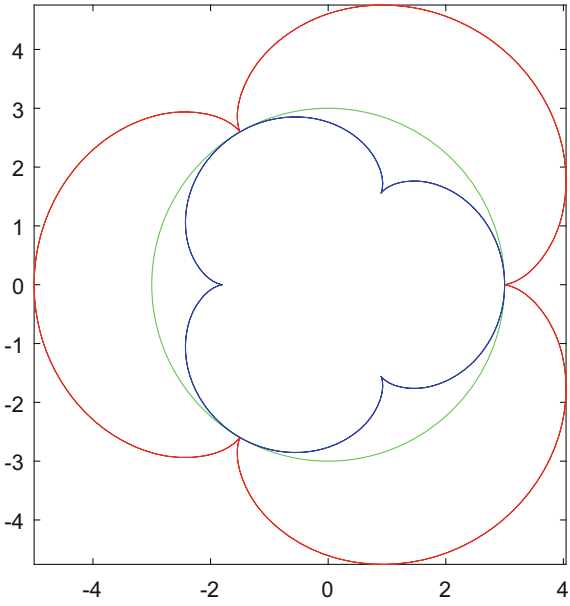
(d) the evolute

```
UT = [vel,0]/speed;
UB = simplify(vta/veclength(vta));
UN = cross(UB, UT); UN = UN(1:2);
evolute = simplify(r + UN/curvature);
pretty(evolute)
```

```
-- --                        / t (a + b) \    (a + b) #2 #4
|  |  cos(t) (a + b) - b cos| --------- | - -------------,
-- --                        \     b     /          #1

                           / t (a + b) \    (a + b) #2 #3 -- --
  sin(t) (a + b) - b sin| --------- | - --------------  |  |
                           \     b     /          #1       -- --

where

                 /                       / t (a + b) \         2 \
                 |                     cos| --------- | (a + b)   |
                 |                        \     b     /           |
        #1 == (a + b) | cos(t) (a + b) - ------------------------- | #4 + (a + b)
                 \                                 b              /

      /                      / t (a + b) \         2 \
      |                    sin| --------- | (a + b)   |
      |                       \     b     /           |
      | sin(t) (a + b) - ------------------------- | #3
      \                                 b             /

                 2   2           2   2
   #2 == (a + b)  #3  + (a + b)  #4

            / t (a + b) \
   #3 == sin| --------- | - sin(t)
            \     b     /

            / t (a + b) \
   #4 == cos| --------- | - cos(t)
            \     b     /
```

Now let's specialize the case a = 3, b = 1. We will plot the fixed circle in green, the epicycloid curve in red, and the evolute in blue.

```
r = simplify(subs(r , [a, b], [3, 1]));
evolute = simplify(subs(evolute , [a, b], [3, 1]));
fplot(r(1), r(2), [0, 4*pi], 'r'), hold on
fplot(3*cos(t), 3*sin(t), [0, 2*pi], 'g')
fplot(evolute(1), evolute(2), [0, 4*pi], 'b'), hold off
axis equal
```



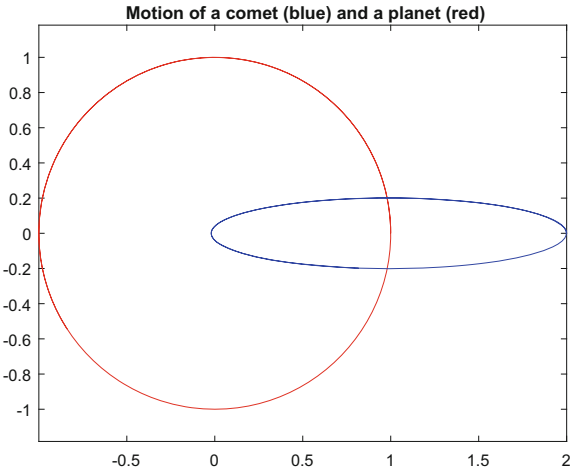
A very pretty picture! The evolute looks like a shrunken and rotated version of the same curve.

12.4 Problem Set D: Problem 6

(a)

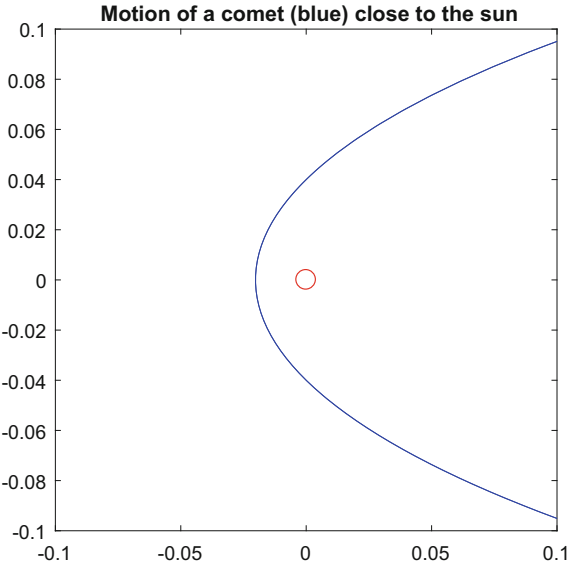
Let w(1), w(2) represent the coordinates x1 and y1, w(3), w(4) represent the coordinates x2 and y2, w(5), w(6) represent the components of the velocity of the planet, w(7), w(8) represent the components of the velocity of the comet. First we assume no interactions.

```
opts = odeset('RelTol',1e-5,'AbsTol',1e-7);
[tsol, wsol] = ode45(@(t, w) [w(5);w(6);w(7);w(8); ...
     -w(1)./(w(1).^2+w(2).^2).^(3/2); ...
     -w(2)./(w(1).^2+w(2).^2).^(3/2); ...
     -w(3)./(w(3).^2+w(4).^2).^(3/2); ...
     -w(4)./(w(3).^2+w(4).^2).^(3/2)], ...
     [0, 10], [1; 0; 2; 0; 0; 1; 0; 0.1], opts);
plot(wsol(:,1), wsol(:,2), 'r'), hold on;
plot(wsol(:,3), wsol(:,4), 'b'), hold off; axis equal
title('Motion of a comet (blue) and a planet (red)')
```



Let's blow up the picture to make sure the comet doesn't crash into the sun.

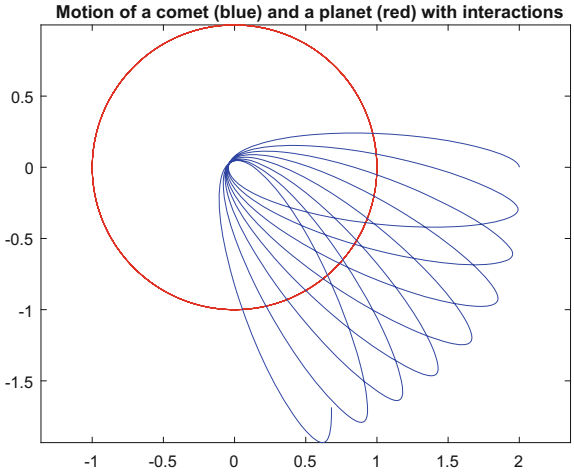
```
axis([-.1, .1, -.1, .1]), hold on
plot(0, 0, 'or', 'MarkerSize', 10), hold off
title('Motion of a comet (blue) close to the sun')
```



(b)

Now we put in the interaction term. This affects the equation for the comet by addition of a term involving the vector difference between the positions of the planet and the comet. The equation of motion of the planet is unchanged.

```
[tsol, wsol] = ode45(@(t, w) [w(5);w(6);w(7);w(8); ...
     -w(1)./(w(1).^2+w(2).^2).^(3/2); ...
     -w(2)./(w(1).^2+w(2).^2).^(3/2); ...
     -w(3)./(w(3).^2+w(4).^2).^(3/2) - ...
  0.1*(w(3)-w(1))./((w(3)-w(1)).^2+(w(4)-w(2)).^2).^(3/2); ...
     -w(4)./(w(3).^2+w(4).^2).^(3/2) - ...
  0.1*(w(4)-w(2))./((w(3)-w(1)).^2+(w(4)-w(2)).^2).^(3/2)], ...
     [0, 50], [1; 0; 2; 0; 0; 1; 0; 0.1], opts);
plot(wsol(:,1), wsol(:,2), 'r'), hold on;
plot(wsol(:,3), wsol(:,4), 'b'), hold off; axis equal
title(['Motion of a comet (blue) and a planet (red) ', ...
  'with interactions'])
```



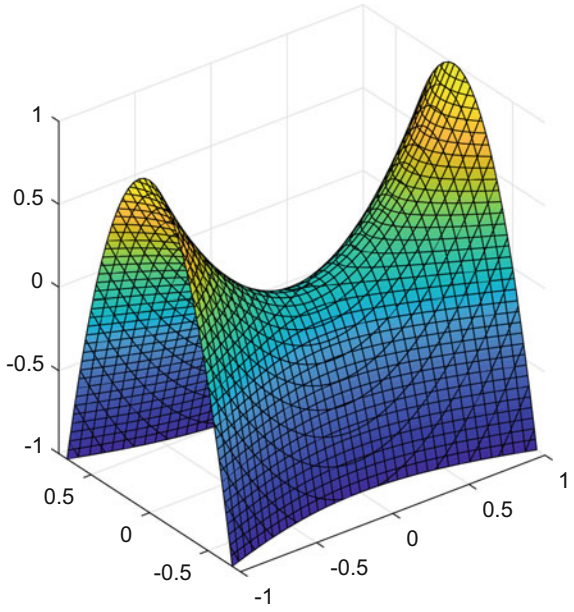
Note that the comet "precesses" around instead of staying in an elliptical orbit.

12.5 Problem Set E: Problem 5, Parts (b) and (c)

```
clear all; close all
```

(b) A hyperbolic paraboloid or saddle surface

```
syms x y z; h = x^2 - 4*y^2 - z;
figure; fimplicit3(h, [-1, 1, -1, 1, -1, 1]); axis equal
```



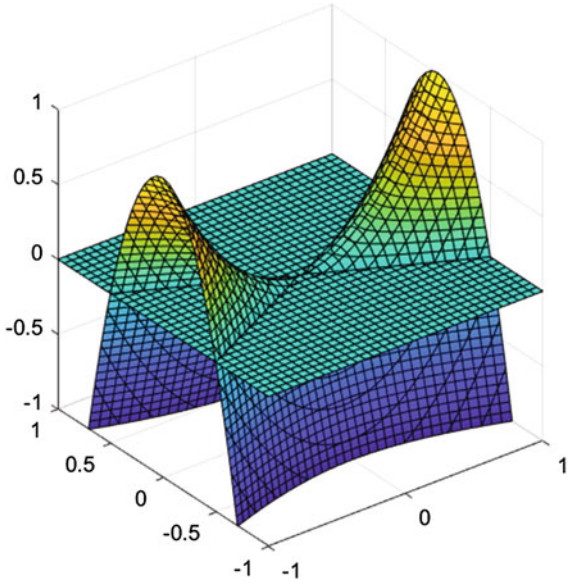
A classic saddle surface. Now for the tangent plane.

```
syms x y real; gradh = jacobian(h)
subs(gradh, [x,y, z], [0,0,0])
```

```
gradh =

[ 2*x, -8*y, -1]


ans =

[ 0, 0, -1]
```

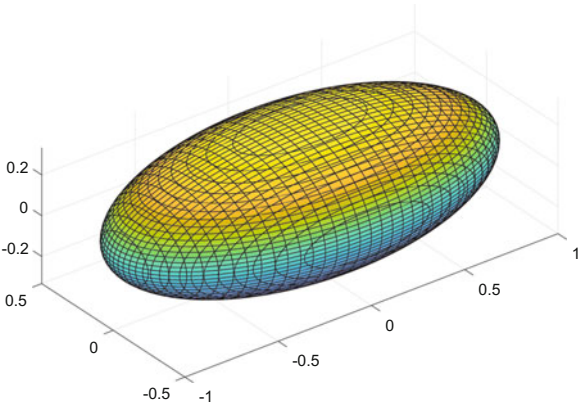
So the normal vector is (0, 0, -1) (or its negative). This verifies that the tangent plane is the x-y plane. Let's add it to the graph.

```
hold on; fsurf(x, y, 0, [-1 1 -1 1])
```



(c) The ellipsoid

```
syms x y z; h = x^2 + 4*y^2 + 9*z^2;
figure; fimplicit3(h-1, [-1 1 -0.5 0.5 -1/3 1/3]); axis equal
```



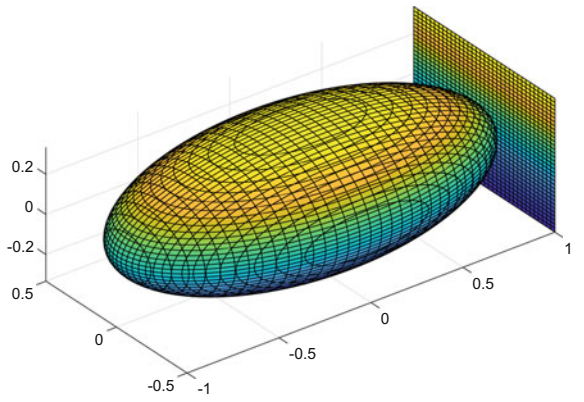
Note that the surface looks something like a football. Now we find the gradient and tangent plane.

```
syms x y real; gradh = jacobian(h)
subs(gradh, [x,y,z], [1,0,0])
```

```
gradh =

[ 2*x, 8*y, 18*z]

ans =

[ 2, 0, 0]
```

The tangent plane is thus normal to (2, 0, 0) or 2**i**, and so is parallel to the y-z plane. The tangent plane is thus x = 1. This is not the graph of a function of two variables, so we display it as a parametric plot of a surface.

```
hold on; syms u v; fsurf(1, u, v, [-1/2 1/2 -1/3 1/3])
```



12.6 Problem Set F: Problem 10, Part (e)

We begin by reproducing the commands that set up the curvature computations that we use for the surfaces in this problem, and of course for all the others in the chapter and problem set.

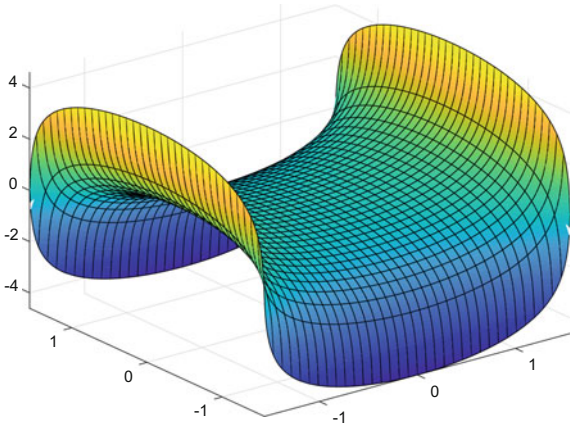
```
veclength = @(vec) sqrt(dot(vec, vec));
normalvector = @(surf, u, v) ...
    simplify(cross(diff(surf, u), diff(surf, v)));
tangentplane = @(surf, u, v, x, y, z) dot([x, y, z] - ...
    surf, normalvector(surf, u, v)== 0);
unitnorm = @(surf, u, v) normalvector(surf, u, v)/...
    veclength(normalvector(surf, u, v));

E = @(surf, u, v) dot(diff(surf, u), diff(surf, u));
F = @(surf, u, v) dot(diff(surf, u), diff(surf, v));
G = @(surf, u, v) dot(diff(surf, v), diff(surf, v));
e = @(surf, u, v) dot(unitnorm(surf, u, v), diff(surf, u, 2));
f = @(surf, u, v) dot(unitnorm(surf, u, v), diff(surf, u, v));
g = @(surf, u, v) dot(unitnorm(surf, u, v), diff(surf, v, 2));

shapeoperator = @(surf, u, v) ...
((E(surf,u,v)*G(surf,u,v)-F(surf,u,v)^2)^(-1)) ...
   *[[e(surf,u,v)*G(surf,u,v)-f(surf,u,v)*F(surf,u,v), ...
f(surf,u,v)*E(surf,u,v) - e(surf,u,v)*F(surf,u,v)]; ...
   [f(surf,u,v)*G(surf,u,v) - g(surf,u,v)*F(surf,u,v), ...
g(surf,u,v)*E(surf,u,v) - f(surf,u,v)*F(surf,u,v)]];

principalcurves = @(surf, u, v) eig(shapeoperator(surf, u, v));
gausscurv = @(surf, u, v) det(shapeoperator(surf, u, v));
meancurv = @(surf, u, v) (1/2)*trace(shapeoperator(surf, u, v));
```

Now for the two surfaces, Scherk's and Enneper's, both of which are, as we shall see, minimal surfaces.

(i) Scherk's Minimal Surface

```
close all
syms u v real
scherk = [u, v, log(cos(v)/cos(u))]
meancurv(scherk, u, v)
figure; fsurf(scherk(1), scherk(2), scherk(3), ...
   [(-pi/2)+0.01, (pi/2)-0.01, (-pi/2)+0.01, (pi/2)-0.01])
```

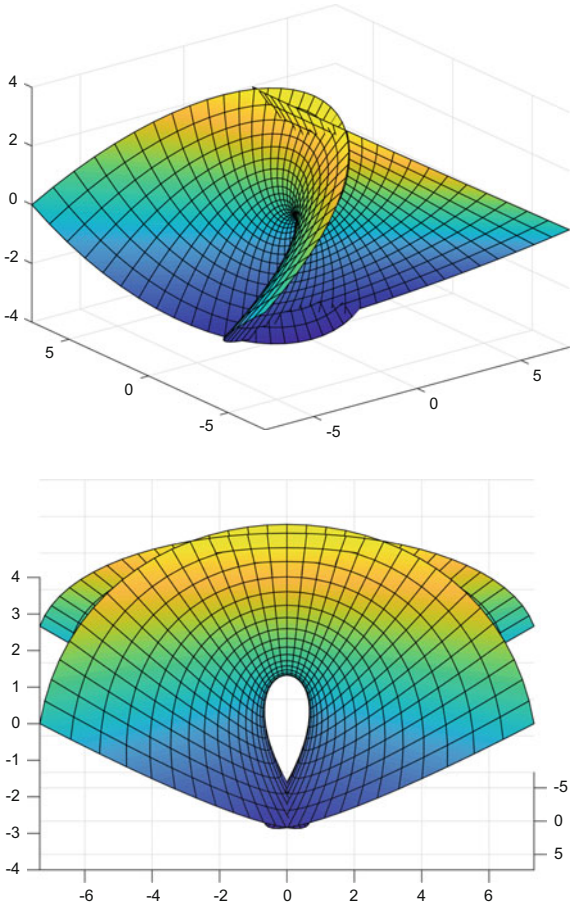
```
scherk =

[ u, v, log(cos(v)/cos(u))]

ans =

0
```



(ii) Enneper's Minimal Surface

We'll give two views – the default, and another in which we can see the "hole" better.

```
enneper = [u - u^3/3 + u*v^2, v - v^3/3 + u^2*v, u^2 - v^2]
simplify(meancurv(enneper, u, v))
figure; fsurf(enneper(1), enneper(2), enneper(3), [-2 2 -2 2])
figure; fsurf(enneper(1), enneper(2), enneper(3), [-2 2 -2 2])
view([3 0 1])


enneper =

[ - u^3/3 + u*v^2 + u, u^2*v - v^3/3 + v, u^2 - v^2]

ans =

0
```



12.7 Problem Set G: Problem 10, Part (b)

This is a problem illustrating the Lagrange multiplier method. We are trying to maximize and minimize a function $f(x,y)$ on the ellipse $g(x,y) = x^2 + 3y^2 = 1$.

(i)

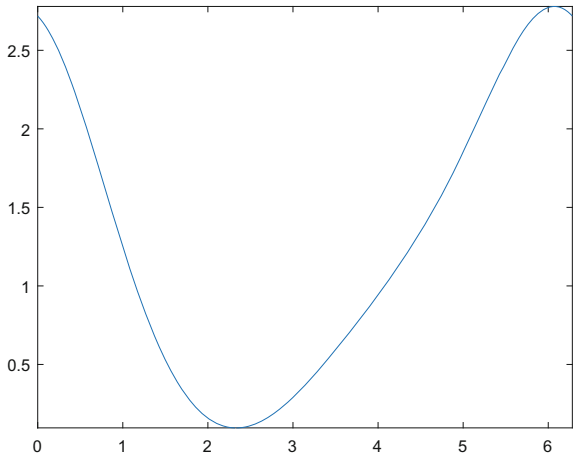
First, we reduce to a one-variable problem and use `fminbnd`.

```
syms x y t real
f = exp(x) - sin(y);
g = x^2 + 3*y^2;
onevarfunct = subs(f, [x, y], [cos(t), sin(t)/sqrt(3)])
```

```
onevarfunct =

exp(cos(t)) - sin((3^(1/2)*sin(t))/3)
```

Let's plot this one-variable function to see where the maximum and minimum seem to be located.

```
fplot(onevarfunct, [0, 2*pi])
```



It seems from this that the maximum occurs near $t = 6$ and the minimum occurs near $t = 2.3$. Indeed, we see:

```
F = matlabFunction(onevarfunct);
xmin = fminbnd(F, 0, 2*pi)
F(xmin)
xmax = fminbnd(matlabFunction(-onevarfunct), 0, 2*pi)
F(xmax)


xmin =

    2.3313

ans =

    0.0957
```

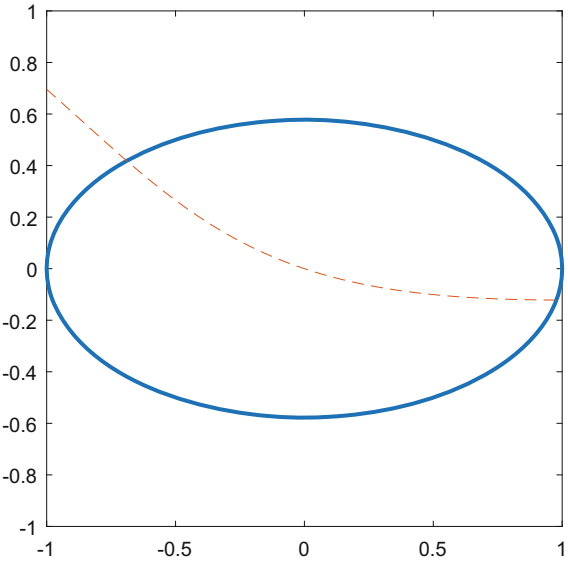
```
xmax =

    6.0708

ans =

    2.7793
```

(ii)

Now we use the Lagrange multiplier method. We will plot the constraint curve in a heavy line style and the Lagrange multiplier equation with a dashed curve. After eliminating λ, we want to set to zero the expression

```
lagr = diff(f,x)*diff(g,y) - diff(f,y)*diff(g,x)
fimplicit(g - 1, [-1, 1, -1, 1], 'LineWidth', 2), hold on
fimplicit(lagr, [-1, 1, -1, 1], '--'), hold off, axis equal
```

```
lagr =

2*x*cos(y) + 6*y*exp(x)
```



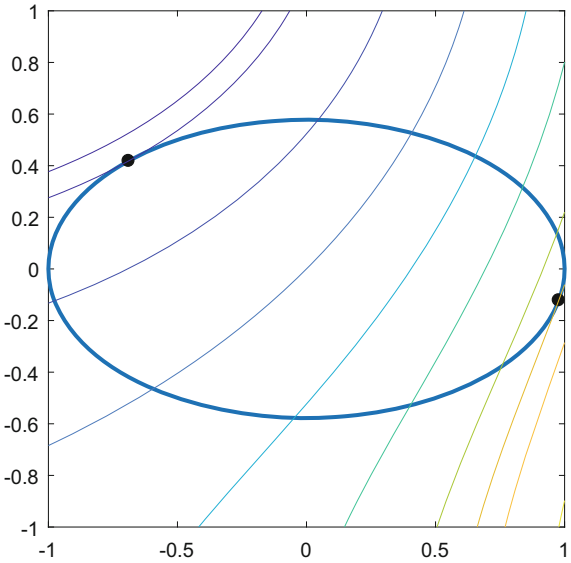
We can see two intersection points, one near (-.6, .4) and one near (.95, -.1). Indeed, using the script `newton2d` from Problem 7.5, we can get fairly accurate values for x and y, and compute the values of the objective function f at the two points.

```
pt1 = newton2d(g-1, lagr, x, y, -.6, .4); pt1
minval = double(subs(f, [x,y], pt1))
pt2 = newton2d(g-1, lagr, x, y, .95, -.1); pt2
maxval = double(subs(f, [x,y], pt2))
```

```
pt1 =

   -0.6893    0.4183

minval =

    0.0957

pt2 =

    0.9775   -0.1217

maxval =

    2.7793
```

Clearly, `pt2` gives the maximum value, around 2.7793, and `pt1` gives the minimum value, around 0.0957. These match what we found by the other method. Finally, we show that the maximum and minimum occur where the level curves of f are tangent to the ellipse. The dark dots indicate the maximum and minimum points, with the maximum in the lower right and the minimum in the upper left.

```
list = [minval, maxval, -1:0.5:4];
fimplicit(g - 1, [-1, 1, -1, 1], 'LineWidth', 2), hold on
plot(pt1(1), pt1(2), '.k', 'MarkerSize', 20)
plot(pt2(1), pt2(2), '.k', 'MarkerSize', 20)
fcontour(f, [-1,1,-1,1], 'LevelList', list),hold off,axis equal
```



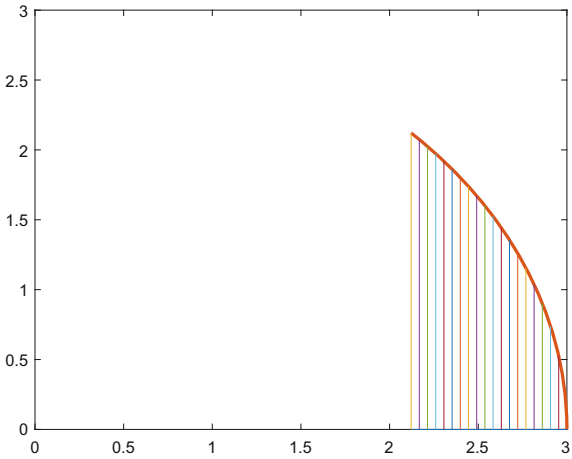
12.8 Problem Set H: Problem 2, Parts (c) and (d)

```
syms x y r theta real
```

(c)

Let's first draw the picture with `verticalRegion` to visualize what's going on.

```
verticalRegion(@(x) zeros(size(x)), @(x) (9-x.^2).^(1/2), ...
  3/sqrt(2), 3)
axis([0, 3, 0, 3])
```

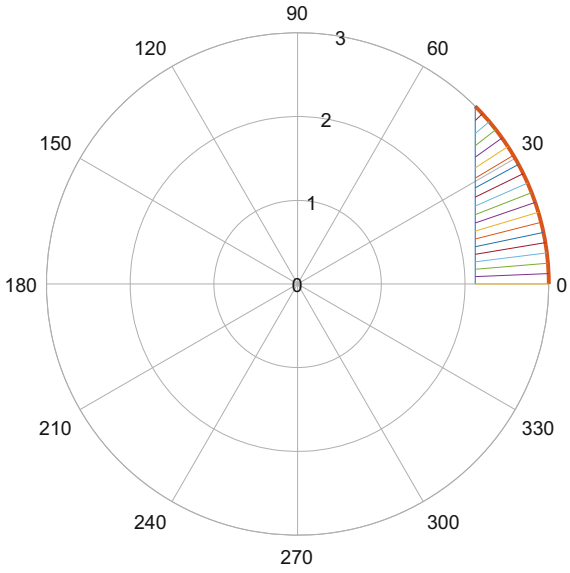


So this is the piece of the disk of radius 3 around the origin in the first quadrant to the right of the line `x = 3/sqrt(2)`. This line has polar equation $r\cos(\theta) = 3/\sqrt{2}$ or $r = \frac{3}{\sqrt{2}}\sec(\theta)$. This line meets the circle `r = 3` where $\cos(\theta) = 1/\sqrt{2}$ or $\theta = \pi/4$.

```
polarRegion(@(theta) 3*sec(theta)/sqrt(2), ...
   @(theta) 3*ones(size(theta)), 0, pi/4)
int(int(r^3, r, 3*sec(theta)/sqrt(2), 3), theta, 0, pi/4)
```

```
ans =
```

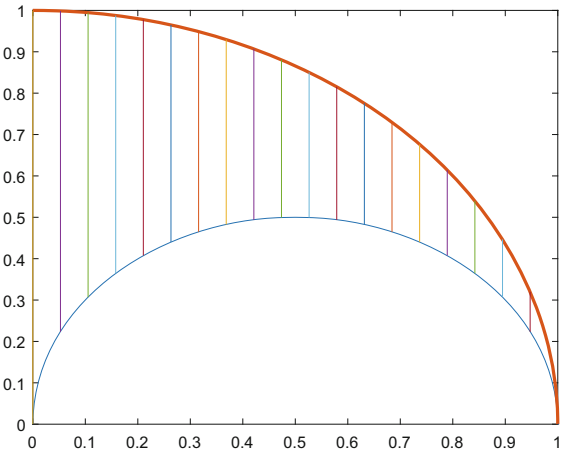
```
(81*pi)/16 - 27/4
```



(d)

Let's first draw the picture with `verticalRegion` to visualize what's going on.

```
verticalRegion(@(x) (x-x.^2).^(1/2), @(x) (1-x.^2).^(1/2), 0, 1)
```



So this is the region between two circles in the first quadrant. The outer circle is `r = 1` and the inner circle is

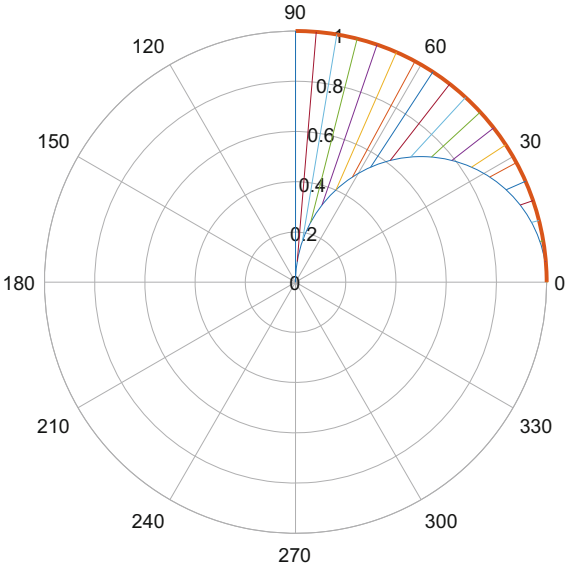
```
simplify(subs(y^2+x^2==x, [x, y], [r*cos(theta),r*sin(theta)]))
```

```
ans =

r == cos(theta) | r == 0
```

So this is $r = \cos(\theta)$.

```
polarRegion(@(theta) cos(theta), ...
   @(theta) ones(size(theta)), 0, pi/2)
int(int(r^4, r, cos(theta), 1), theta, 0, pi/2)
```

```
ans =

pi/10 - 8/75
```



12.9 Problem Set I: Problems 4(b) and 5(a)

```
clear all; close all
```

4(b) Using Green's Theorem to compute an area

We compute the area inside one arch of a cycloid by using Green's Theorem to convert the area integral into a line integral. The line integral we need to compute is:

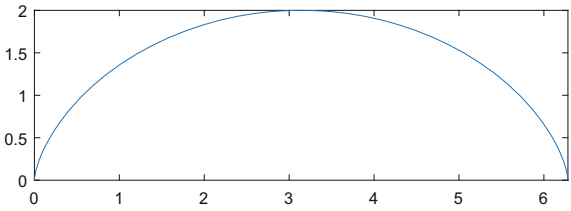
$$-\frac{1}{2}\oint_C ydx + xdy$$

where C is the boundary of the arch traversed counterclockwise.

On the base of the area in question, i.e., along the x-axis, both y and dy are zero, so we only need to evaluate along the arch. We'll draw the cycloid just for the record.

```
syms t; cycloid = [t - sin(t),1 - cos(t)];
fplot(cycloid(1), cycloid(2), [0 2*pi])
axis equal
area = (1/2)*(int(-(1-cos(t))*(1 - cos(t)) + ...
   (t - sin(t))*sin(t), [2*pi, 0]))
```

```
area =

3*pi
```



5(a) Using Green's Theorem to convert a line integral into a double integral

We are asked to compute the work done by a force field on a particle moving around an ellipse, which, of course, is a line integral computation. We use Green's Theorem to convert the line integral into a double integral, which we evaluate via MATLAB.

```
clear all; close all
syms x y z real; F = [-y+5*x, 3*y + x];
G = curl([F,0], [x,y,z]); G(3)

ans =

2
```

Therefore, by Green's Theorem, the line integral $\oint_C(-y+5x)dx+(3y+x)dy =$ two times the area of the ellipse $x^2+9y^2=9$, which is π times the product of the lengths of the two semi axes, i.e., 3 and 1. The value of the line integral is therefore 6π. Let's verify that by having MATLAB do the computation.

```
syms u v real; 2*4*int(int(1, v, 0, sqrt(1-(u/3)^2)), u, 0, 3)


ans =

6*pi
```

12.10 Problem Set J: Problem 3, Parts (a) and (b), but only subpart (ii)

```
clear all
close all
```

(a) Gauss' Law

Let `g(r)` be the gravitational field strength, `rho(r)` the mass density, `gravconst` the gravitational constant. Then we have

```
syms a r z theta h rho(r) gravconst real
g = @(a) 4*pi*gravconst*int(int(int(rho(r)*r,r,0,a),z,0,h),...
    theta, 0, 2*pi)/(2*pi*a*h);
g(a)
```

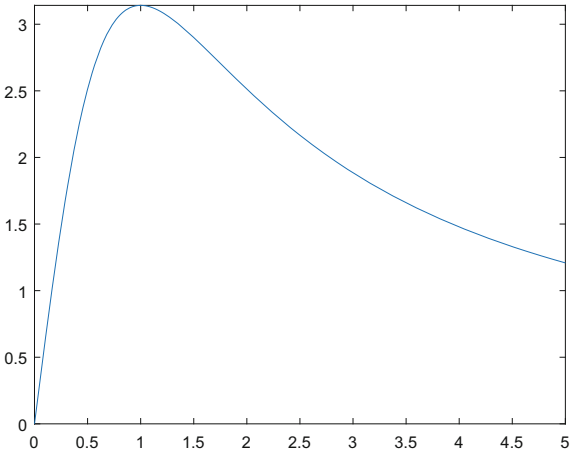
```
Warning: Can only make assumptions on variable names, not 'rho(r)'.

ans =

(4*pi*gravconst*int(r*rho(r), r, 0, a))/a
```

We use this expression below.

(ii) Decrease like the 4th power

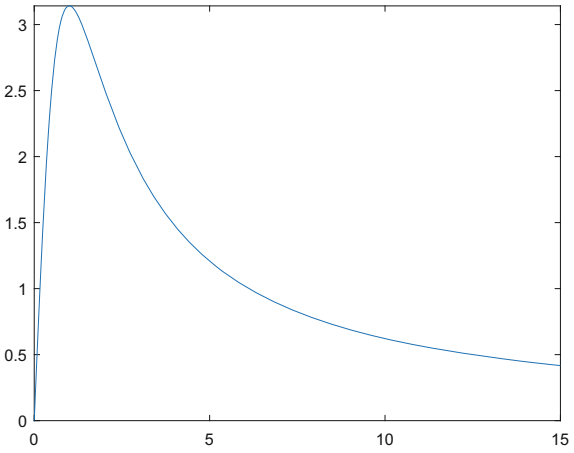
```
gravconst=1; rho = @(r) 1/(1+r^2)^2
gravfieldstrength = simplify(4*pi*gravconst*int(r*rho(r),r,0,a)/a)
figure; fplot(gravfieldstrength, [0 5])
```

```
rho =

  function_handle with value:

    @(r)1/(1+r^2)^2

gravfieldstrength =

(2*pi*a)/(a^2 + 1)
```



The field strength increases initially—apparently linearly for a while—reaches a maximum at about $r = 1$, then begins to decay, rather slowly it would appear, although the equations suggest in inverse proportion to the distance. This is rather slow decay, so the field is still substantial far out from the mass. Here's another graph that substantiates the claim.

```
figure; fplot(gravfieldstrength, [0 15])
```



(b) Poisson's Equation

We use the Laplacian in cylindrical coordinates (actually, it is essentially polar coordinates here since no `z` is involved in the computation). In fact $\nabla^2(\rho(r)) = \rho''(r) + (1/r)\rho'(r)$ (using that the derivative with respect to θ is also zero).

(ii)

```
syms r real; rho = @(r) 1/(1+r^2)^2
syms t phi(t) real;
sol = dsolve(diff(phi(r),r)/r + diff(phi(r),r,2) == ...
    4*pi*gravconst/(1 + r^2)^2, r);
simplify(sol)
```

```
rho =

  function_handle with value:

    @(r)1/(1+r^2)^2

Warning: Can only make assumptions on variable names, not 'phi(t)'.

ans =

C4 + pi*log(r^2 + 1) + C5*log(r) - 2*pi*log(r)
```

The constant in front of the $\log r$ must match 2π and so the term we must deal with is $\text{const} + \pi \log(1 + r^2)$, whose gradient is proportional to $r/(1 + r^2)$, which again is what we got in part (a).

Index

R.L. Lipsman and J.M. Rosenberg, *Multivariable Calculus with MATLAB®*,
DOI 10.1007/978-3-319-65070-8

D

E